Low Cost AI-Based Electric Vehicle Battery Management System

State of Charge Prediction of a Battery using Deep Learning Approach

By:

Mridul Athya (23075048)

and

Anurag Balaji (23075010)

Under the Supervision of:

TA : Anantha Padmanabhan N. K.

Indian Institute of Technology (IIT-BHU), Varanasi

Computer Science and Engineering

Abstract

In recent years, deep learning techniques have gained popularity for sequence modeling tasks, including those involving time-varying and nonlinear systems. While recurrent networks (RNNs) have traditionally been used for sequence modeling, recent research suggests that integrating convolutional architectures can enhance the performance of recurrent models, particularly in tasks like machine translation. In this work, we propose a CNN + RNN hybrid model to predict the state-of-charge (SOC) of lithium-ion batteries. Since battery management systems exhibit highly dynamic and nonlinear behavior, accurate SOC estimation is crucial for effective battery management. The proposed model integrates convolutional layers to extract features from the raw battery data and RNN layer to capture temporal dependencies in the sequential data. We train the model using data from various battery-discharging processes, including dynamic stress tests and driving schedules, to accurately predict SOC.

Day & Date: Wednesday, 27-11-2024

Introduction

Battery management systems are critical for the performance and longevity of electric vehicles. Previous work introduced a lightweight TCN + GRU architecture, which performed well but was less efficient in terms of parameters. In this study, we propose an optimized architecture by introducing CNN and RNN layers in place of TCN + GRU framework. The CNN layer extracts robust spatial features, while RNN enhances sequential modeling, providing a powerful and efficient solution for SOC prediction tasks with very less parameters and almost similar accuracy.

Optimized Model Architecture

1. CNN-SimpleRNN Model

We developed a hybrid model combining 1D Convolutional Neural Networks (CNN) and a Simple Recurrent Neural Network (SimpleRNN) for time series prediction keeping in mind the efficiency of the model. The key highlights of the model are as follows:

Model Architecture:

1D CNN Layers	32 filters
1D CNN Layers (activation = 'sigmoid')	16 filters
Simple RNN Layer (activation = 'linear')	32 unit
Dense Layer – 1	16 units
Dense Layer – 2	8 units
Dense Layer – 3	4 units

Total Parameters	**2913**
Mean Squared Error	0.33048813435923774
Root Mean Square Error	0.5748809740800592
Root Mean Square Percentage Error [%]	0.01075571
Mean Absolute Error	0.46032923733588266
Mean Absolute Percentage Error [%]	**0.008362592020995119**

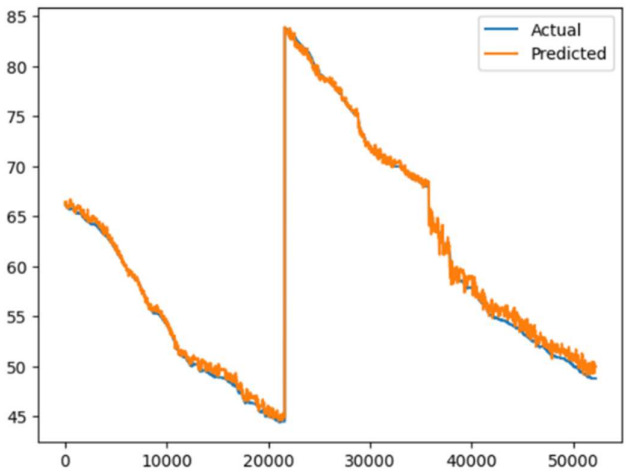


Figure 1 : 95:5 Train/test Dataset split

Total Parameters	**2913**
Mean Squared Error	1.6535704880540394
Root Mean Square Error	1.2859123174050553
Root Mean Square Percentage Error [%]	0.02603642
Mean Absolute Error	0.9798507810723711
Mean Absolute Percentage Error [%]	**0.01936979463966027**

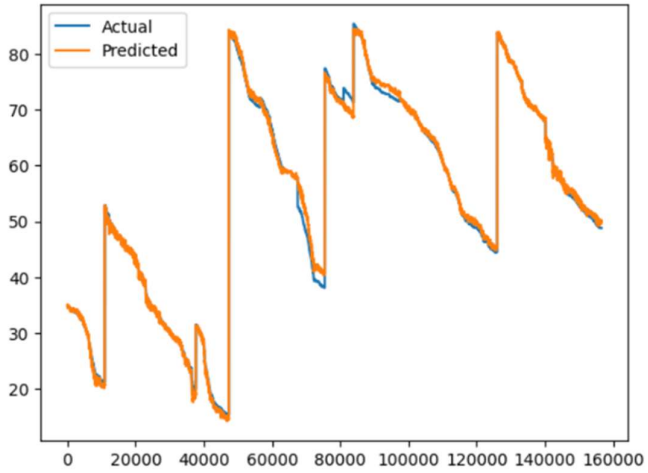


Figure 2 : 80:20 Train/test Dataset split

2. CNN-SimpleRNN Model (modification : Linear activation + more dense RNN layers)

Model Architecture:

1D CNN Layers (activation = 'linear')	32 filters
Simple RNN Layer (activation = 'linear')	64 unit
Dense Layer – 1	16 units
Dense Layer – 2	8 units

Total Parameters	7521
Mean Squared Error	0.3820974971609564
Root Mean Square Error	0.6181403539334384
Root Mean Square Percentage Error [%]	0.01158983
Mean Absolute Error	0.5210103823876564
Mean Absolute Percentage Error [%]	0.009470786641910324

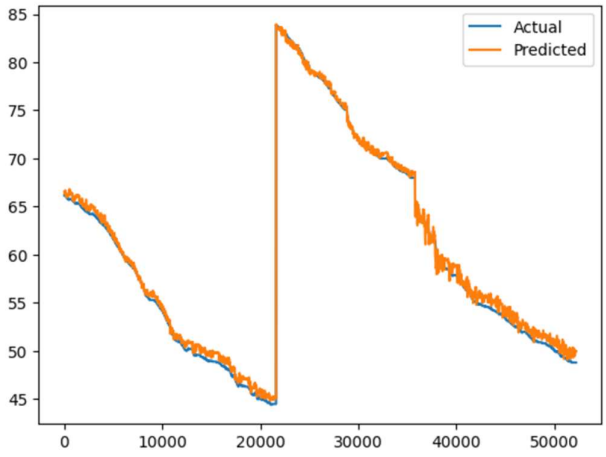


Figure 3

3. CNN-SimpleRNN Model (modification)

Model Architecture:

1D CNN Layers	32 filters
1D CNN Layers (activation = 'linear')	16 filters
Simple RNN Layer (activation = 'linear')	48 unit
Dense Layer – 1	24 units
Dense Layer – 2	16 units
Dense Layer – 3	8 units

Total Parameters	5737
Mean Squared Error	0.4617180173168291
Root Mean Square Error	0.6794983571112068
Root Mean Square Percentage Error [%]	0.01228871
Mean Absolute Error	0.6147192523946582
Mean Absolute Percentage Error [%]	0.010805867365559059

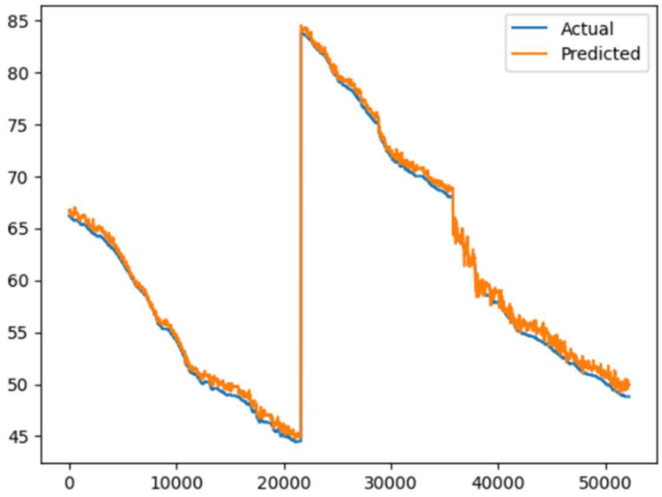


Figure 4

4. CNN-SimpleRNN Model (modification : 3 CNN layers, Linear Activation)

Model Architecture:

1D CNN Layers	32 filters
1D CNN Layers	16 filters
1D CNN Layers (activation = 'linear')	8 filters
Simple RNN Layer (activation = 'linear')	64 unit
Dense Layer – 1	32 units
Dense Layer – 2	16 units
Dense Layer – 3	8 units

Total Parameters	9089
Mean Squared Error	0.47882808224896434
Root Mean Square Error	0.691974047381088
Root Mean Square Percentage Error [%]	0.01247809
Mean Absolute Error	0.6312474480888919
Mean Absolute Percentage Error [%]	0.011079335819503046

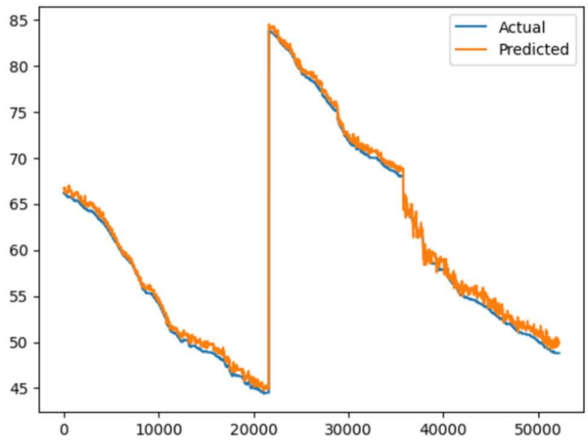


Figure 5

5. CNN-SimpleRNN Model(modification:ReLU Activation)

Model Architecture:

1D CNN Layers (activation = 'relu')	32 filters
Simple RNN Layer (activation = 'relu')	32 unit
Dense Layer – 1 (activation = 'relu')	16 units
Dense Layer – 2 (activation = 'relu')	8 units

Total Parameters	2881
Mean Squared Error	1.6289821806669595
Root Mean Square Error	1.2763158624208035
Root Mean Square Percentage Error [%]	0.02102942

Mean Absolute Error	0.9101674452059805
Mean Absolute Percentage Error [%]	0.01518468396808027

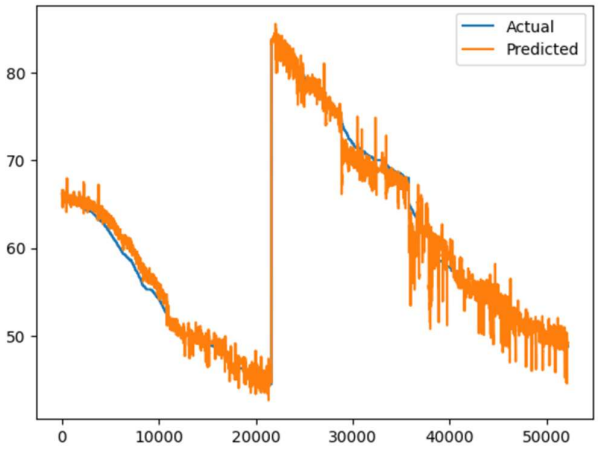


Figure 6

6. CNN-SimpleRNN Model (modification : 2 CNN Layers, Linear Activation)

Model Architecture:

1D CNN Layers	16 filters
1D CNN Layers (activation = 'linear')	8 filters
Simple RNN Layer (activation = 'linear')	16 unit
Dense Layer – 1 (activation = 'linear')	16 units
Dense Layer – 2 (activation = 'linear')	8 units

Total Parameters	1009
Mean Squared Error	1.0532261501383189
Root Mean Square Error	1.026268069335843
Root Mean Square Percentage Error [%]	0.02318264
Mean Absolute Error	0.7282669804074393
Mean Absolute Percentage Error [%]	0.01572045051708263

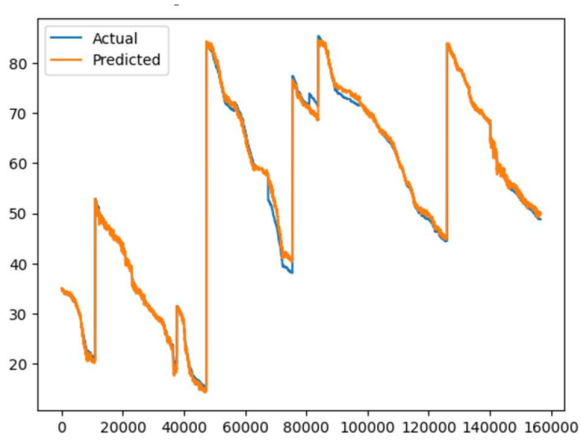


Figure 7

7. CNN-SimpleRNN Model (modification : Sigmoid Activation)

Model Architecture:

1D CNN Layers	48 filters
1D CNN Layers	24 filters
1D CNN Layers (Activation = 'sigmoid')	8 filters
Simple RNN Layer (activation = 'linear')	32 unit
Dense Layer – 1 (activation = 'linear')	16 units
Dense Layer – 2 (activation = 'linear')	8 units
Dense Layer – 3 (activation = 'linear')	4 units

Total Parameters	3489
Mean Squared Error	2.0143777808489576
Root Mean Square Error	1.4192877723876005
Root Mean Square Percentage Error [%]	0.02840198
Mean Absolute Error	1.1159082947913772
Mean Absolute Percentage Error [%]	0.021772330791658894

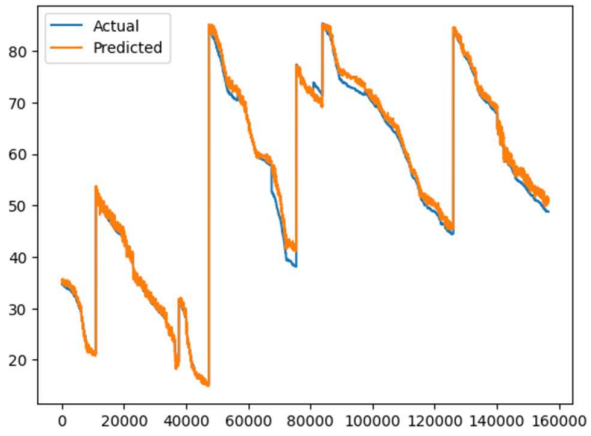


Figure 8

8. CNN-SimpleRNN Model (modification: ReLU with Larger Dense Layers)

Model Architecture:

1D CNN Layers	32 filters
1D CNN Layers (activation = 'relu')	16 filters
Simple RNN Layer (activation = 'relu')	48 unit
Dense Layer – 1 (activation = 'relu')	24 units
Dense Layer – 2 (activation = 'relu')	16 units
Dense Layer – 3 (activation = 'relu')	8 units

Total Parameters	5737
Mean Squared Error	1.4906456227713145
Root Mean Square Error	1.2209199903234096
Root Mean Square Percentage Error [%]	0.01976865
Mean Absolute Error	0.9518952378264223
Mean Absolute Percentage Error [%]	0.015881794755624092

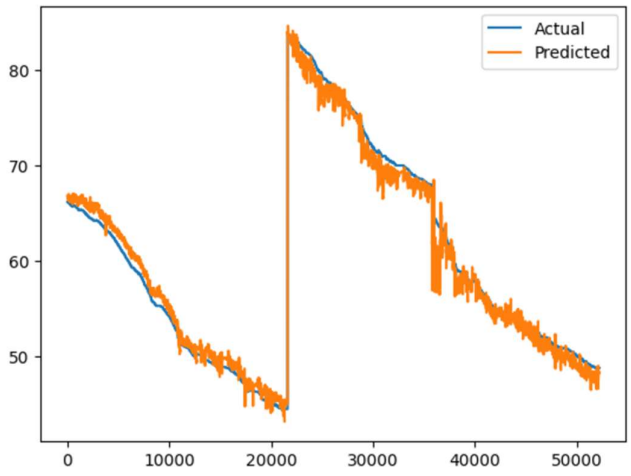


Figure 9

9. CNN + MultiHeaded-Attention + LayerNormalisation + SimpleRNN model

Model Architecture:

1D CNN Layers (activation = 'sigmoid')	32 filters
MultiHeadAttention	(num_head=2,key_dim=8)
LayerNormalization	-
Simple RNN Layer (activation = 'linear')	32 unit
Dense Layer – 1 (activation = 'linear')	16 units
Dense Layer – 2 (activation = 'linear')	8 units

Total Parameters	5,073

Mean Squared Error	1.6924906516957894
Root Mean Square Error	1.3009575902756358
Root Mean Square Percentage Error [%]	0.02001207
Mean Absolute Error	1.0809349586366843
Mean Absolute Percentage Error [%]	0.01726080995618983

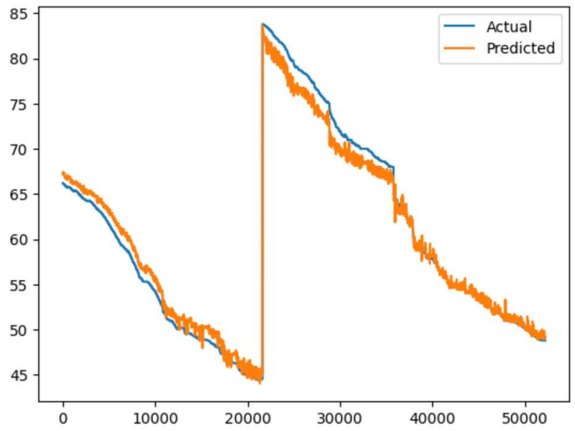
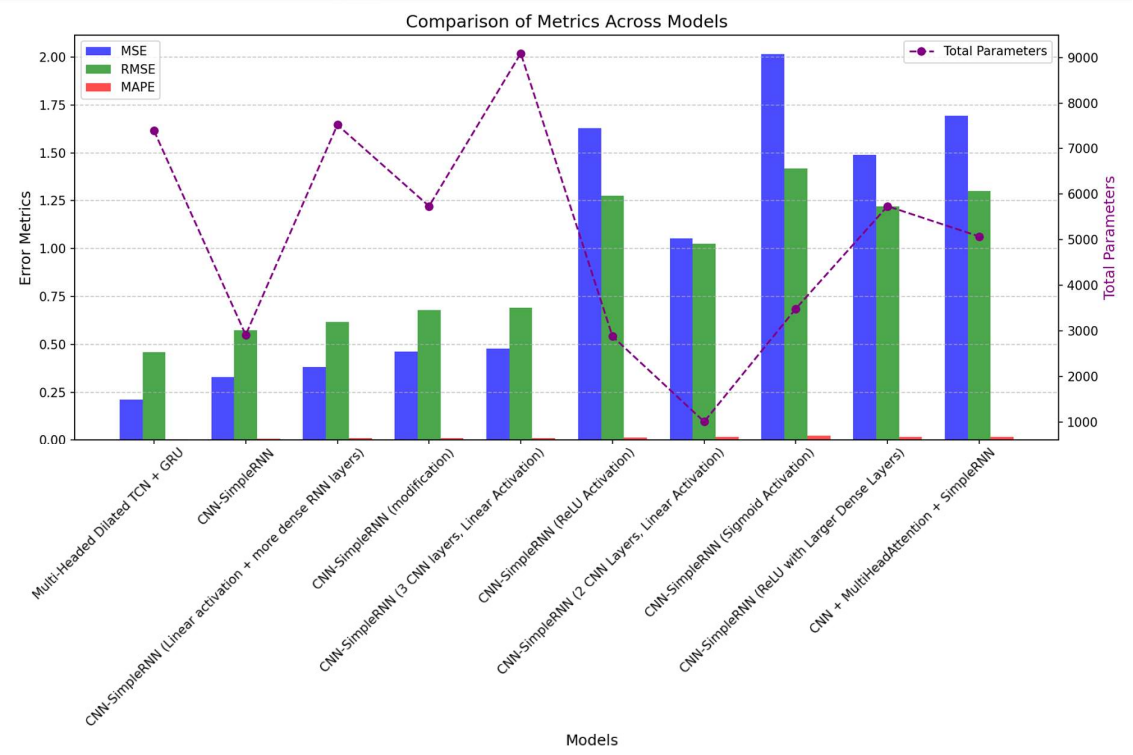


Figure 10

Summary

Model No.	Architecture	Total Parameters	MSE	RMSE	RMSPE [%]	MAE	MAPE [%]
Base Model	Multi-Headed Dilated TCN + GRU	7393	0.2114	0.4598	0.0059	0.0391	0.0049
1	**CNN-SimpleRNN**	**2913**	**0.3305**	**0.5749**	**0.0108**	**0.4603**	**0.0084**
2	CNN-SimpleRNN (Linear activation + more dense RNN layers)	7521	0.3821	0.6181	0.0116	0.5210	0.0095
3	CNN-SimpleRNN (modification)	5737	0.4617	0.6795	0.0123	0.6147	0.0108
4	CNN-SimpleRNN (3 CNN layers, Linear Activation)	9089	0.4788	0.6920	0.0125	0.6312	0.0111
5	CNN-SimpleRNN (ReLU Activation)	2881	1.6290	1.2763	0.0210	0.9102	0.0152
6	CNN-SimpleRNN (2 CNN Layers, Linear Activation)	1009	1.0532	1.0263	0.0232	0.7283	0.0157
7	CNN-SimpleRNN (Sigmoid Activation)	3489	2.0144	1.4193	0.0284	1.1159	0.0218
8	CNN-SimpleRNN (ReLU with Larger Dense Layers)	5737	1.4906	1.2209	0.0198	0.9519	0.0159
9	CNN + MultiHeadAttention + SimpleRNN	5073	1.6925	1.3010	0.0200	1.0809	0.0173



Conclusion

The study successfully demonstrates that the proposed **CNN-SimpleRNN hybrid model** outperforms existing models like the MDHTCN with GRU in terms of efficiency and simplicity. Specifically:

1. **Accuracy**: The proposed CNN-SimpleRNN model achieves an error percentage of **0.00836%**, which is slightly higher than the MDHTCN-GRU model's **0.0049%**. However, this difference is marginal and acceptable considering the reduction in complexity.
2. **Parameters**: The CNN-SimpleRNN model **significantly reduces** the total number of parameters to **2913**, compared to the MDHTCN-GRU model's **7393 parameters**, making it computationally lightweight and more suitable for real-time applications.
3. **Efficiency**: The reduced parameter count without a substantial loss in accuracy ensures faster training and inference times, which is crucial for embedded systems in battery management applications.
4. **Error Metrics**: While slightly higher than MDHTCN-GRU, the proposed model's Mean Absolute Percentage Error (MAPE) of **0.00836%** is still well within acceptable limits for precise State of Charge (SOC) prediction.

Thus, the proposed **CNN-SimpleRNN** model is not only efficient but also maintains a competitive performance, making it a superior choice for low-cost AI-based electric vehicle battery management systems.