Graph Neural Network Model for Predicting Electric Vehicle Battery Voltage

Volcov Sabina
Faculty of Computer Science
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
volcov.vl.sabina@student.utcluj.ro

Lungu Ioan Stelian
Faculty of Industrial Engineering, Robotics and Production Management
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
lungu.ho.ioan@student.utcluj.ro

***Abstract*— The aim of this paper is to present a solution for battery voltage estimation in solar-powered electric vehicles based on electrical signals collected from the motor controller, thermistors and solar panel cells. This study proposes a Graph Neural Network (GNN) model to accurately predict the battery's health in terms of voltage. The model leverages the interconnected nature of sensor data to capture complex relationships, offering a robust approach for real-time estimation. Experimental results highlight the effectiveness of the proposed method in improving both prediction accuracy and generalization across various conditions.**

***Keywords*—graph neural network, artificial intelligence, machine learning, electric vehicle, battery voltage prediction**

I. Introduction

Artificial Intelligence (AI) has been extensively used to optimize performance, enhance decision-making and improve efficiency across various engineering domains, including robotics [1], smart home systems [2] and renewable energy technologies [3]. In the transportation sector, AI-driven models are integral to energy management and performance optimization, particularly in the advancement of sustainable mobility solutions [4].

In the context of solar-powered electric vehicles, real-time monitoring of system components and the selection of optimal driving strategies are critical for maximizing energy efficiency and ensuring overall functionality. A major challenge in this field lies in the effective management of battery load based on continuous electrical sensor readings, which helps optimize energy consumption. To address this issue, Machine Learning (ML) techniques can be employed to develop predictive models for motor efficiency and battery health estimation. However, traditional models often struggle to capture the intricate and dynamic relationships between electrical inputs and battery performance.

This study presents a solution for battery voltage estimation by implementing and evaluating a Graph Neural Network (GNN) model that utilizes motor controller data, alongside battery pack measurements from thermistors and solar cells, to predict battery voltage with the lowest possible errors. GNNs are particularly well-suited for this application due to their capability to model graph-structured data, making them an effective framework for learning from time-dependent sensor data in electric vehicle systems.

II. Related Work

Many approaches have been explored for performance optimization in energy systems. Traditional methods, such as Naïve Bayes [5], k-NN [6] or direct mathematical formulae [7] work well on small datasets with few features, but fail to capture the relationships that inevitably appear between some of the input data. Recent advancements in deep learning have shown promise in addressing these limitations. However, these models typically assume grid-like or sequential data structures, which may not fully represent the interconnected nature of sensor data in electric vehicles.

Graph Neural Networks (GNNs) have emerged as a powerful alternative for modeling such relational data. GNNs have been successfully applied in social network analysis, molecular property prediction, and recommendation systems [8]. Their ability to capture dependencies between nodes in a graph makes them particularly suitable for sustainable solar-powered systems, where sensor data can be naturally represented as a graph of interconnected components.

Some studies already attempted to integrate the GNNs complexity in solar power prediction, primarily focusing on meteorological data such as temperature, humidity and solar irradiance to improve forecasting accuracy [9]. In response, recent work has aimed to develop more comprehensive models by incorporating satellite cloud maps, real-time cloud tracking and ground sensors. This more extensive data web provides a better understanding of how atmospheric conditions evolve over time and how they influence solar power output [10]. Other papers focused on fault diagnosis by monitoring current and voltage production of photovoltaic systems [11].

The experimental results show that these new GNN models achieve high prediction accuracy and strong robustness, demonstrating better generalization ability in varying conditions. Additionally, they offer higher computational efficiency compared to older forecasting methods. Overall, this data-driven, AI-powered approach represents a promising direction toward maximizing solar power potential and enabling a cleaner, more sustainable energy future.

III. Methodology

A. GNN model

The architecture used in this paper is a Graph Neural Network (GNN), a machine learning architecture designed to process data interpretable in the form of a graph. Unlike traditional neural networks that operate on grid-like structures, such as images or sequences, GNNs employ representations using nodes and edges to model relationships between interconnected entities [8].

a) GNN Input Representation

The input to the GNN consists of a feature matrix $X \in R^{N \times F}$, where N represents the number of time steps (or nodes in the graph) and F denotes the number of extracted features. The features used include:

- Temporal information: the timestamp converted into elapsed seconds of the day.
- Battery parameters: battery voltage, current, maximum and minimum cell voltages and cell temperatures.
- Motor controller readings: voltage and current values from the direct (Vd, Id) and quadrature (Vq, Iq) axes of the motor controller.
- Two engineered features: voltage difference (ΔV) and power-to-voltage ratio (P/V).

Each row in X represents a sample (or observation) and each column is a separate attribute (or feature). Since GNNs typically perform better on positive values, the columns are split into two: one for the positive part and one for the negative part. The features are normalized using a MinMaxScaler to ensure that they are within the same scale (between 0 and 1), an important operation for neural network, as it helps speed up convergence and avoids issues caused by some features dominating the others.

To define the graph structure, an adjacency matrix is constructed based on temporal connectivity, where each node (or time step) is connected to its immediate predecessor and successor, forming a chain-like graph. Self-loops are added to preserve individual node information during message passing and ensure the neural network captures the self-interactions of the data. The graph structure is represented as an edge index E, stored as a list of connections between nodes.

In each forward pass, the GNN receives both features (X) and the graph structure to compute predictions, considering all the relationships between consecutive samples.

b) GNN Layers

The model in this study consists of 5 convolutional layers, each with 256 hidden dimensions, Rectified Linear Unit (ReLU) activation function, a Dropout layer, and one fully-connected layer for the output, as shown in Figure 1. The core mechanism behind the Graph Neural Network is message passing, where each node in the graph aggregates information (messages) from its neighbors to update its representation in each layer. After a forward pass, the convolutional layers' output can be formulated as:

$$H^{(\lambda+1)} = \int(\Gamma X N X o\nu\varpi(H^{(\lambda)}, E)), \qquad (1)$$

where $H^{(l)}$ represents the feature matrix at layer l, and σ is the activation function, ReLU:

$$P\varepsilon\Lambda Y(\xi) = \mu\alpha\xi(0, \xi) \qquad (2)$$

The formula for the graph convolution being:

$$\Gamma X N X o\nu\varpi(H^{(\lambda)}, E) = \Re * H^{(\lambda)} * \Omega^{(\lambda)}, \qquad (3)$$

with Â – the normalized adjacency matrix E, and $W^{(l)}$ – the matrix of weights for layer l.

At the end of each forward pass of the entire model, the loss function is computed and backpropagated to update the weights on each layer before the next training epoch.

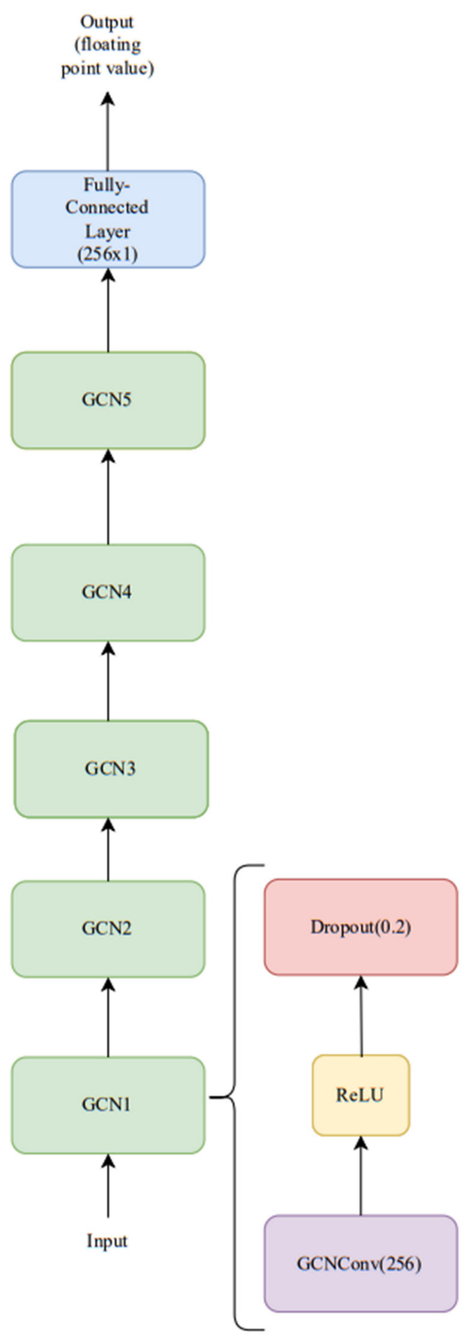
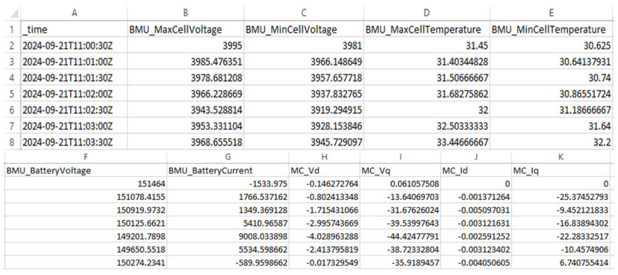


Fig. 1. Figure 1. GNN achitecture

B. Dataset description

The data used for the task of battery voltage estimation was collected through the sensors on the Solis racing car during the iLumen European Solar Challenge 2024 competition, over the course of 8 hours and multiple circuit laps. The values were stored in an InfluxDB container via Controller Area Network (CAN) bus communication and queried in .csv format. The battery can provide at most 160V (volts) and 21A (ampers), with the values for current and voltage at any given time measured in millivolts and milliampers, respectively.

An example of raw entries for the model is the one depicted in Figure 2:



	A	B	C	D	E
1	_time	BMU_MaxCellVoltage	BMU_MinCellVoltage	BMU_MaxCellTemperature	BMU_MinCellTemperature
2	2024-09-21T11:00:30Z	3995	3981	31.45	30.625
3	2024-09-21T11:01:00Z	3985.476351	3966.148649	31.40344828	30.64137931
4	2024-09-21T11:01:30Z	3978.681208	3957.657718	31.50666667	30.74
5	2024-09-21T11:02:00Z	3966.228669	3937.832765	31.68275862	30.86551724
6	2024-09-21T11:02:30Z	3943.528814	3919.294915	32	31.18666667
7	2024-09-21T11:03:00Z	3953.331104	3928.153846	32.50333333	31.64
8	2024-09-21T11:03:30Z	3968.655518	3945.729097	33.44666667	32.2

F	G	H	I	J	K
BMU_BatteryVoltage	BMU_BatteryCurrent	MC_Vd	MC_Vq	MC_Id	MC_Iq
151464	-1533.975	-0.146272764	0.061057508	0	0
151078.4155	1766.537162	-0.802413348	-13.64069703	-0.001371264	-25.37452793
150919.9732	1349.369128	-1.715431066	-31.67626024	-0.005097031	-9.452121833
150125.6621	5410.96587	-2.995743669	-39.53997643	-0.003121631	-16.83894302
149201.7898	9008.033898	-4.028963288	-44.42477791	-0.002591252	-22.28332517
149650.5518	5534.598662	-2.413795819	-38.72332804	-0.003123402	-10.4574906
150274.2341	-589.9598662	-0.017329549	-35.9189457	-0.004050605	6.740755414

Fig. 2. Input Example

Therefore, the features used for battery voltage prediction were: time (converted to seconds during preprocessing), BMU_MaxCellVoltage (maximum voltage from all the cells, in millivolts), BMU_MinCellVoltage (minimum voltage from all the cells, in millivolts), BMU_MaxCellTemperature (maximum temperature on the cells, in °C), BMU_MinCellTemperature (minimum temperature on the cells, in °C), BMU_BatteryCurrent (current recorded on the battery, in milliampers), MC_Vd, MC_Vq and MC_Id, MC_Iq (the direct and quadratic voltages and currents recorded on the Motor controller MC), all split into their positive and negative parts, as well as two new engineered features: voltage differences and power-to-voltage ratio.

IV. Setup

A. Training and Optimization

The model was trained over 350 epochs using the AdamW optimizer with an initial learning rate of 0.001 and weight decay of 1×10^{-5}. A ReduceLROnPlateau scheduler was added to adjust the learning rate dynamically based on validation loss. The most efficient loss function for the battery voltage estimation task turned out to be Huber Loss, which provides robustness to outliers in the data.

The Huber Loss function is a combination of the mean squared error (MSE) and mean absolute error (MAE), with a threshold δ. It is defined as:

$$\Lambda_{\text{TM}}(\psi, \psi\sim) = \begin{cases} \frac{1}{2}(y - y\sim)^2, & if\ |y - y\sim| \leq \delta \\ \delta\left(|y - y\sim| - \frac{1}{2}\delta\right), & otherwise \end{cases} \quad (4)$$

where y is the true value (ground truth), y~ is the predicted value and δ is the threshold that defines the point where the loss function transitions from quadratic (MSE) to linear (MAE). In this study, δ is set to 0.5.

If the error is small, this function behaves like mean squared error. However, if the error is large, it behaves like mean absolute error to reduce sensitivity to outliers. This allows for robust handling of noisy and missing data while maintaining smoothness near the true values.

For the training of the GNN model, the dataset was split into 80% training set and 20% testing set, ensuring that edge connectivity is preserved for both sets. During each epoch, gradient clipping (max_norm=5.0) was applied to stabilize updates. Early stopping was implemented based on validation performance, with the best-performing model checkpoint saved for final evaluation.

To match the scale of the target values, the final predictions were rescaled to their original range using the inverse transformation of the MinMaxScaler, ensuring accurate interpretation of results.

B. Implementation

The implementation of this model is intended for Python with PyTorch and Torch geometric to define the necessary layers. These libraries offer flexibility in designing and training AI models with various predefined and custom architectures, for any kind of data. PyTorch provides an intuitive framework for building neural networks, while Torch Geometric extends PyTorch by offering specialized tools for efficiently handling graph structures, performing message passing in GNNs. Figure 3 presents an example of using PyTorch to define the model used for this experiment:

```python
import torch
from torch_geometric.nn import GCNConv

class GNN(torch.nn.Module):
    def __init__(self, input_dim, hidden_dim, output_dim):
        super(GNN, self).__init__()
        self.conv1 = GCNConv(input_dim, hidden_dim)
        self.conv2 = GCNConv(hidden_dim, hidden_dim)
        self.conv3 = GCNConv(hidden_dim, hidden_dim)
        self.conv4 = GCNConv(hidden_dim, hidden_dim)
        self.conv5 = GCNConv(hidden_dim, hidden_dim)
        self.fc = torch.nn.Linear(hidden_dim, output_dim)
        self.dropout = torch.nn.Dropout(0.2)

    def forward(self, x, edge_index):
        x = self.conv1(x, edge_index).relu()
        x = self.dropout(x)
        x = self.conv2(x, edge_index).relu()
        x = self.dropout(x)
        x = self.conv3(x, edge_index).relu()
        x = self.dropout(x)
        x = self.conv4(x, edge_index).relu()
        x = self.dropout(x)
        x = self.conv5(x, edge_index).relu()
        x = self.dropout(x)
        x = self.fc(x)
        return x
```

Fig. 3. Example of model definition in PyTorch

To prepare the dataset for use in a Graph Neural Network (GNN), a structured preprocessing pipeline was followed that transforms the data in the correct format for the model. The dataset was originally stored in a .csv file and contained both positive and negative numerical values. The preprocessing involves cleaning and structuring the data into a graph format.

a) Libraries used for preprocessing

- pandas: for reading and processing .csv files.
- numpy: for numerical operations such as absolute values and clipping.

b) Data processing pipeline:

1. Loading the file using pandas.read_csv() and keeping only the numerical columns;
2. Handling negative values by splitting the columns with both positive and negative values into two separate ones and the converting the columns with all negative values to all positive;
3. Structuring data for GNN input by creating a graph representation where each row represents a node and relationships are determined based on timestamps;
4. Constructing the adjacency matrix using the existing PyTorch functions;

5. Converting data to PyTorch Geometric format: creating a new torch_geometric.data.Data object;

These preprocessing steps ensured that the dataset was properly formatted for GNN training, with all features transformed into positive values and the relationships tbetween them correctly defined.

V. Simulation Results

The trained GNN model was assessed using several regression metrics, as described in Table 1:

TABLE I. SIMULATION RESULTS

Metric	Formula(5, 6, 7, 8)	Value for the trained model
Mean Absolute Error (MAE)	$\frac{1}{n}\sum_{i=1}^{n} \lvert y_i - y\sim_i \rvert$	**718.1757**
Root Mean Squared Error (RMSE)	$\sqrt{\frac{1}{n}\sum_{i=1}^{n} (y_i - y\sim_i)^2}$	**1265.4047**
R-Squared (R²) Score	$1 - \frac{\sum_{i=1}^{n}(y_i - y\sim_i)^2}{\sum_{i=1}^{n}(y_i - \overline{y})^2}$	**0.8496**
Mean Absolute Percentage Error (MAPE)	$\frac{100}{n}\sum_{i=1}^{n} \left\lvert \frac{y_i - y\sim_i}{y_i} \right\rvert$	**0.0060**
where y – true values, y~ - predicted values, $\overline{y}$ – mean of true values		

The mean absolute error shows that the average magnitude of the differences between the actual and predicted values is about 718, an acceptable value because of the scale of the battery voltages, which are calculated in millivolts, and not in volts. The root mean squared error suggests that the model occasionally predicts large deviations from the ground truth, which is normal due to the inherent variations in battery voltage.

However, the R-squared score is a value close to 1, meaning that the model explains about 85% of the variance in the battery voltage predictions. The mean absolute percentage error is also a strong indicator of good performance, suggesting that the predictions are extremely accurate when it comes to the percentage error relative to the true values.

During the 350 epochs of training, the loss decreased, as illustrated in Figure 4, indicating that the model learned and improved its predictions over time. The fluctuations in the graph suggest some periods of instability, but the general downward trend implies a progress in training.

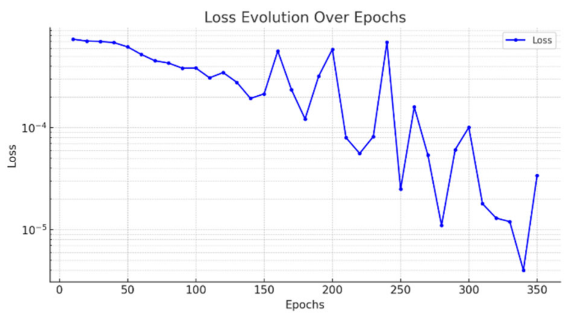


Fig. 4. Loss evolution during training (logarithmic scale)

The final, best model resulted in predictions very close to the actual target values, as shown in Figure 5. Although there are areas where the model might struggle, such deviations are to be expected and do not undermine the GNN's general ability to make accurate estimations.

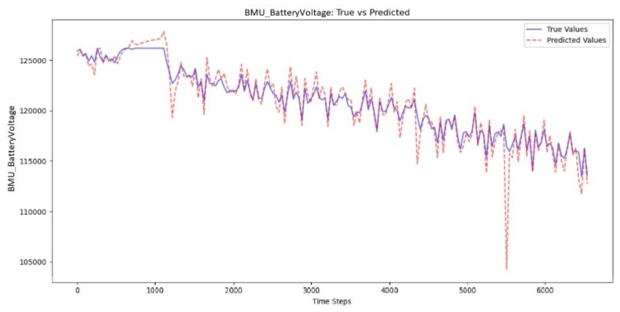


Fig. 5. True values and values predicted by the model

VI. Conclusions

Graph Neural Networks effectively capture the non-trivial dependencies between electric vehicle parameters and, unlike traditional models, can adapt to dynamic, time-varying electrical conditions, making them suitable for real-time applications. Leveraging both structural and temporal data, GNNs enhance prediction accuracy and robustness, ensuring reliable performance in tasks that involve sensor monitoring. Their adaptability makes them particularly useful for optimizing energy management in electric systems.

As a result, GNN-based models offer a promising approach for improving efficiency and sustainability in modern transportation systems, especially in solar-powered electric vehicles.

Acknowledgment

The authors would like to express their sincere gratitude to the TUCN Solis Solar Racing Team for generously providing the data used in this research. Their dedication to advancing sustainable and innovative solar-powered vehicle technology has been instrumental in this study. Without their efforts in gathering and maintaining high-quality telemetry data from the Solis racing car, this work would not have been possible.

References

[1] M. W. Otte, "A survey of machine learning approaches to robotic path-planning," Department of Computer Science, University of Colorado at Boulder, Boulder, CO, 2023.

[2] S. Park, "Machine learning-based cost-effective smart home data analysis and forecasting for energy saving," Buildings, vol. 13, no. 9, p. 2397, 2023.

[3] V. S. K. Reddy, S. T. Saravanan, N. T. Velusudha, and T. S. Selwyn, "Smart grid management system based on machine learning algorithms for efficient energy distribution," E3S Web of Conferences, vol. 387, p. 02005, 2023.

[4] „How AI is making electric vehicles safer and more efficient", Available: https://www.ibm.com/think/topics/ai-ev-batteries

[5] R. A. Tharakan, R. Joshi, G. Ravindran, and N. Jayapandian, "Machine learning approach for automatic solar panel direction by using Naïve Bayes algorithm," in Proceedings of the Fifth International Conference on Intelligent Computing and Control Systems (ICICCS 2021), ISBN: 978-0-7381-1327-2, Bangalore, India, 2021.

[6] N. Uddin, E. Purwanto, and H. Nugraha, "Machine learning based modeling for estimating solar power generation," 3S Web of Conferences, vol. 475, p. 03009, 2024.

[7] J. Gallagher and S. Clarke, "Energy-efficient route prediction for solar-powered vehicles," Green Energy and Intelligent Transportation, vol. 2, no. 1, p. 100063, Feb. 2023.

[8] CS224W: Machine Learning with Graphs, Stanford online, Fall 2024. Available: https://web.stanford.edu/class/cs224w/index.html

[9] V. Manimegalai, V. Deekshitha, T. Dhivya Shri, D. Harini, V. Mohanapriya, and A. Elakya, "Graph Neural Networks for Hyper-Accurate Solar Power Forecasting," Proc. 7th Int. Conf. Inventive Comput. Technol. (ICICT 2024), ISBN: 979-8-3503-5929-9, 2024.

[10] H. Yang, G. Wang, M. Su, J. Liu, and R. Zhang, "Photovoltaic power generation prediction based on graph neural network and superimposed satellite cloud images," 2023 IEEE Power and Energy Conference (POWERCON), Jinan, China, 2023.

[11] J. Van Gompel, D. Spina, and C. Develder, "Cost-effective fault diagnosis of nearby photovoltaic systems using graph neural networks," Ghent, Belgium, 2023.