Graphical Convolutional Networks Based Regenerative Brake Energy Prediction System for solar-powered electric vehicles

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

***Abstract*— *This study presents an innovative approach for predicting **regenerative braking energy** in **solar vehicles** using **Graphical Convolutional Networks (GCNs)**. The proposed system focuses on the **Solis EV4** solar vehicle, using **vehicle parameters** to forecast the potential for energy regeneration during braking events. A key contribution is the **automated preprocessing** methodology for manipulating **mixed-sign parameters** in vehicle data. The design implements a **multi-layer GCN architecture** that processes **temporal sequences** of vehicle data to provide **real-time predictions**. The results demonstrate exceptional accuracy, with an R2 score of 0.9977 and an RMSE error of 0.0367 kW, validating the effectiveness of the proposed approach.*

Keywords— graphical convolutional networks, artificial intelligence, machine learning, electric vehicle *regenerative braking*

I. INTRODUCTION

Networks Artificial intelligence (AI) has become an essential technology in many areas, from image and speech recognition to natural language processing and decision-making. One of the most important branches of AI is deep learning, which focuses on developing algorithms that can learn and adapt from experience [3]. One of the key concepts in deep learning is data representation. In many cases, data can be represented in the form of graphs, which are data structures composed of vertices and edges. Graphs can be used to represent complex relationships between objects, such as social networks, transport networks, or molecular structures.

Graph Convolutional Network (GCN) are a class of deep learning models specifically designed to process structured data in the form of graphs. GCNs were introduced as an extension of convolutional neural networks (CNNs), which are very effective for Euclidean data such as images or text. However, CNNs cannot directly process graph structures, which require a special approach [1],[2].

GCNs work by propagating and aggregating information between graph vertices, allowing the model to learn representations that capture both the local properties of the vertices and the overall structure of the graph. This capability makes them particularly useful in applications where relationships and interdependencies between components are crucial to solving the problem. One such field is that of electric and solar vehicles, where numerous parameters – from operating conditions and battery status to route characteristics – interact in complex ways to determine system performance [4],[5].

In particular, the accurate prediction of regenerative energy recoverable by braking is a significant challenge, as it depends on multiple interdependent variables that can be naturally modeled as a graph. The vertices of this graph can represent different parameters of the vehicle, and the edges can shape the relationships and influences between these parameters. [1], [2]

II. DATA PREPROCESSING AND EXPERIMENTAL SETUP

The dataset used in this study was collected during the ILUMEN 2024 solar car competition in Belgium, specifically from the Solis EV4 vehicle of the Technical University of Cluj-Napoca (UTCN) team. The Solis EV4 is the latest and most efficient solar vehicle developed by the Solis team, incorporating numerous improvements over previous models [6].

The collected dataset comprises multiple vehicle performance parameters, measured and recorded directly from the Solis EV4 during the competition runs. These include fundamental electrical parameters (bus voltage and current), drivetrain data (motor speed), dynamic measurements (vehicle speed and distance traveled), and critical energy metrics (power consumption and regeneration, energy consumption per kilometer). Together, these parameters create a detailed representation of the solar vehicle's behavior and energy management during actual racing conditions[4], [6].

A. Data Preprocessing

To ensure that the data is in a format suitable for training the GCN model, we have pre-processed it in such a way that there are no negative values. The reasons for this decision are:

- Data Normalization: I used MinMaxScaler to normalize the data, ensuring that all features are on the same scale. Normalization contributes to the stability of the training and helps to converge the model faster [9].
- Elimination of Negative Values: The separation of positive and negative components allows the model to identify specific patterns and independently analyze the behavior of each component. This approach improves the ability to generalize and provides a deeper understanding of the relationships between parameters. Components.

- Model Architecture: The model uses 5 active GCNConv layers. Which convolutional layers are activated by the ReLU function for each layer.Dropout (0.) is included for regularization and prevention of model overfitting.

In addition to normalization, we've introduced new features to improve the model's ability to learn relationships between vehicle parameters. New features include.

```
data['power_ratio'] = data['Power_Consumption_kW'] /
(data['Power_Regeneration_kW'] + 1e-6)
data['velocity_ratio'] = data['MC_MotorVelocity_pos'] /
(data['MC_MotorVelocity_neg'] + 1e-6)
data['current_ratio'] = data['MC_BusCurrent_pos'] /
(data['MC_BusCurrent_neg'] + 1e-6)
```

Fig. 1. Feature Engineering Example

B. *Data Splitting*

For training and evaluation of the model, dataset was divided into 2 main parts: 80% of total data was allocated fortraining and validation process, and remaining 20% was reserved for final testing. From 80% allocated to training, an additional subdivision was made into training data (80%) and validation data (20%), using the following code:

```
train_split = int(0.8 * len(x))
x_train, x_test = x[:train_split], x[train_split:]
y_train, y_test = y[:train_split], y[train_split:]
```

Fig. 2. Data Splitting

The remaining 20% of total dataset was kept separate and unused during training and validation process. This dataset serves exclusively for the final evaluation of the model, simulating real-world scenarios where the model needs to make predictions on completely new and unknown data. This approach allows us to accurately assess the model's actual generalizability and robustness under practical conditions of use.

TABLE I. DATA PROCESSED

time	2024-09-2	2024-09-2	2024-09-2	2024-09-2	2024-09-2	2024-09-2	2024-09-2	2024-09-2	2024-09-2	2024-09-2	2024-09-2	2024-09-2	2024-09-20T14:31:02.0140192
MC_BusVoltage	134.2277	134.23	134.2438	134.2384	134.2389	134.2373	134.2407	134.2384	134.2395	134.2419	134.2386		
time_diff	0.185762	0.067141	0.294513	0.198986	0.049828	0.410612	0.193648	0.013602	0.237771	0.386882	0.048419		
Power_Consumption_kW	0	0	0	0	0.024611	0.025112	0.024163	0.024246	0.023173	0	0.000214		
Power_Regeneration_kW	0	0	0	0	0	0	0	0	0	0.056799	0		
Energy_Consumption_kWh	0	0	0	0	3.41E-07	2.86E-06	1.30E-06	9.16E-08	1.53E-06	0	2.87E-09		
Energy_Regeneration_kWh	0	0	0	0	0	0	0	0	0	6.10E-06	0		
Vehicle_Speed_mps	0.29249	0.292146	0.287143	0.287753	0.283804	0.30283	0.323581	0.345927	0.368397	0.391513	0.413886		
Vehicle_Speed_kph	1.052963	1.051726	1.033716	1.03591	1.021696	1.090186	1.164682	1.245339	1.326029	1.409448	1.562171		
Distance_km	5.43E-05	1.96E-05	8.46E-05	5.73E-05	1.42E-05	0.000124	6.27E-05	4.71E-06	8.78E-05	0.000151	2.10E-05		
Consumption_per_km	0	0	0	0	0.024089	0.023035	0.020742	0.01947	0.017426	0	0.000137		
MC_BusCurrent_pos	0	0	0	0	0.183338	0.187075	0.179994	0.18062	0.172627	0	0.001591		
MC_BusCurrent_neg	0	0	0	0	0	0	0	0	0	0.423108	0		
MC_MotorVelocity_pos	0	0	0	0	0	0	0	0	0	0	0		

III. METHOLOGY

Graph Convolutional Networks have achieved promising results in various areas of machine learning, especially the processing of structured data in the form of graphs [1], [2]. In the next part we will expose the approach of implementing GCN to make predictions of regenerative energy in electric vehicles [7].

The architecture of the implemented model is presented in Fig. 3, highlighting the flow of information through the network. Starting from the input features, the data passes through five GCN layers that perform convolution operations combined with ReLU activation and dropout mechanisms [1], [3]. A fully connected layer processes the final GCN layer's output to generate the regenerative energy prediction [7].

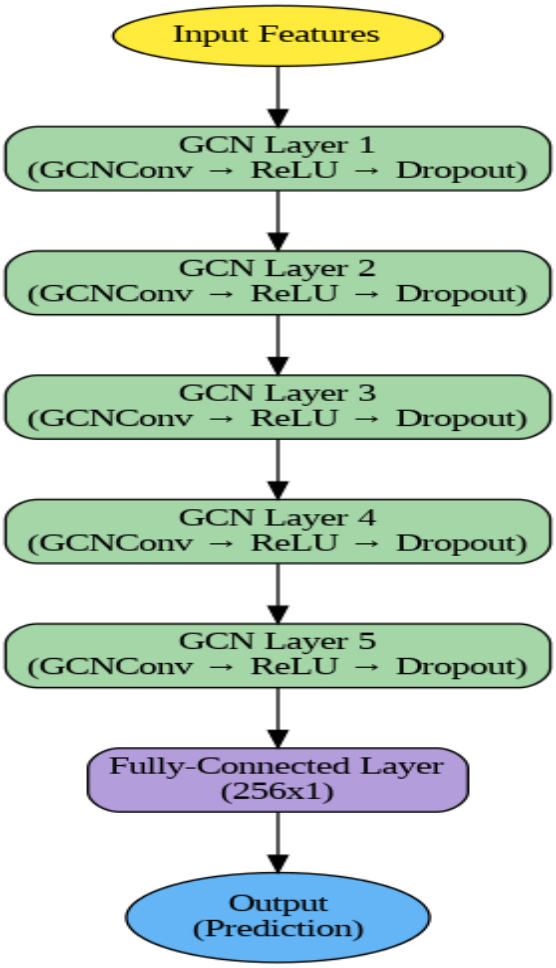

Fig. 3. GCN Model Architecture

The GCN model is structured in 3 main components:

A. *Layers of Input*

The input layer is responsible for processing the 17 features that were obtained by preprocessing the data, described in Chapter 2, and which includes a combination of the original parameters and features derived by feature engineering.

B. *Layers GCN Layers of Input*

The architecture includes 5 consecutive GCN layers, each implementing the following fundamental operation:

$$H^{l+1} = \sigma\left(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{l}W^{l}\right) \quad (1)$$

Equation (1) represents the core mathematical operation performed in each Graph Convolutional Network (GCN) layer. Here, H^(l+1) denotes the feature matrix of the next layer, while H^l represents the current layer's features. The matrix Ã is the adjacency matrix with added self-connections, describing how nodes in the graph are connected. D̃ is the degree matrix derived from Ã, and its negative square root D̃^(-1/2) serves as a normalization factor. W^l represents the learnable weight matrix for the current layer, and σ is a non-linear activation function, typically ReLU. This operation effectively aggregates information from neighboring nodes while maintaining the graph's structural properties. The normalization term D̃^(-1/2) prevents the scaling of feature vectors from varying too much across nodes with different degrees, ensuring stable training. Through this operation, each node updates its features by considering both its own information and the weighted influence of its neighbors, enabling the network to learn increasingly complex graph-level representations [1].

Layers are configured with a hidden size of 256 units for each layer, Dropout of 0.3 between layers for regularization, and ReLU activation after each convolutional layer. To improve the learning process, we have implemented the Huber Loss cost function, which provides an optimal compromise between Mean Squared Error (MSE) and Mean Absolute Error (MAE). This feature is more robust to outliers compared to MSE and offers more stable gradients than MAE. The Huber Loss formula is defined as follows:

$$L_\delta(a) = \begin{cases} \frac{1}{2}x^2, & if\ |x| \le \delta, \\ \delta\left(|x| - \frac{1}{2}\delta\right), & if\ |x| > \delta, \end{cases} \qquad (2)$$

Where δ is the parameter that controls the transition between quadratic (for small errors) and linear (for large errors) behavior. In our implementation, we chose δ = 0.7 based on empirical experiments that showed the best compromise between training stability and prediction accuracy[11].

C. *Stratul de Outpu*

The final layer consists of a linear transformation that produces the prediction for regenerative energy, using a linear activation function. Optimization is performed using the AdamW algorithm with adaptive learning rate:

$$\theta_t = \theta_{t-1} - \alpha * m_t / \left(\sqrt{v_t} + \varepsilon\right) - \lambda * \theta_{t-1} \qquad (3)$$

Where θ_t is the parameters at step t, α is the learning rate (initially set to 0.01), m_t and v_t are the first- and second-order moments, and λ is the weight decay factor (1e-4). The learning rate is dynamically adjusted using a ReduceLROnPlateau scheduler with a waiting period of 50 epochs and a reduction factor of 0.1, which allows the model to find better local minimums as the training progresses [10].

D. *Implement model*

The implementation is carried out in PyTorch Geometric, using the following structure that allows efficient processing of data in the form of a graph and the application of convolution operations [1]. The model's architecture is implemented in a class that inherits torch.nn.Module, providing flexibility in parameter management and data forward propagation [2]. The following code shows the basic structure of the implementation:

```
class GCN(torch.nn.Module):
    def __init__(self, input_dim, hidden_dim, output_dim):
        super(GCN, 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(0.4)
    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. 4. Model implementation

The figure below illustrates the process of information propagation in a GCN network, which consists of three main steps: Feature Propagation, Linear Transformation and Nonlinearity. The model initially receives an input graph, where each vertex is represented by a feature vector. At each GCN layer, information is propagated between the connected nodes by filling the feature matrix with a normalized adjacency matrix, facilitating the exchange of information between neighbors. Then, a linear transformation is applied by supplementing with an array of trainable weights, followed by a ReLU activation to introduce nonlinearity. This process is iterated over K-1 times, where K is the total number of layers of the model. Finally, the predictions are generated by applying a softmax function to the final representations of the vertices. This approach allows for efficient propagation of information in the GCN grid and is used in this model to improve the prediction of regenerative energy in electric vehicles [1], [7].

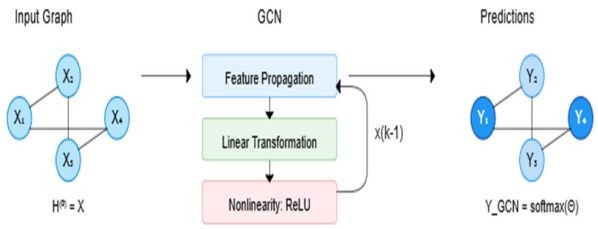

Fig. 5. Propagation of information in a Graph Convolutional Network

IV. EXPERIMENTS

This section presents the experimental results obtained from the implementation and evaluation of the GCN model for the prediction of regenerative energy in electric vehicles. The experiments were carried out using data collected during the competition in Belgium, divided into training set (80%) and test set (20%)

A. Experimental Setup

To evaluate the model's performance, we used three standard metrics in regression problem:

- MAE (Mean Absolute Error): measures the average of absolute errors [12].

$$\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \quad (4)$$

- RMSE (Root Mean Squared Error): measures the square root of the mean of the square errors [12].

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \quad (5)$$

- R^2 Score: indicates the proportion of the variance of the dependent variable that is predictable from the independent variable

$$1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \quad (6)$$

B. Experimental Setup

The model was trained for 250 epochs, showing consistent improvement in performance throughout the training process. The training loss evolved from an initial value of 0.0776 at epoch 10 to a final value of 0.0092 at epoch 250, demonstrating stable convergence. Significant improvements were observed in the early stages of training, particularly between epochs 20 and 40, where the loss decreased from 0.0739 to 0.0173, indicating rapid learning of the underlying patterns in the data.

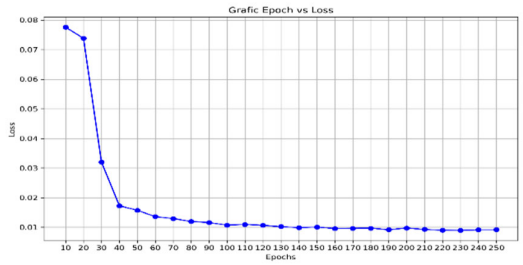

Fig. 6. Training Loss Evolution over 250 Epochs

C. Results and Analysis

TABLE II. GCN MODEL PERFORMANCE

Metric	Training Set(80%)	Testing Set(20%)
MAE	0.0143	2.1699
RMSE	0.0367	2.3250
R^2 Score	0.9977	-8.5345

The experimental results are presented in Table 2, which compares the performance of the model on the drive set and on the test set.

On the test set, we see a significant degradation in the model's performance, which raises questions about its ability to generalize. A negative R^2 score of -8.5345 indicates that the model fails to explain the variability of the data. Also, the prediction errors, measured by MAE (2.1699) and RMSE (2.3250), are significantly higher than on the training set, which highlights a problem of generalization of the model, which fails to maintain the same level of performance when applied to new and unknown data.

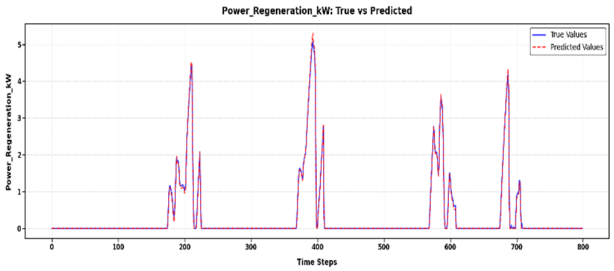

Fig. 7. True vs Predicted: Power Regeneration (Training)

On the test set, we see a significant degradation in the model's performance, which raises questions about its ability to generalize. A negative R^2 score of -8.5345 indicates that the model fails to explain the variability of the data. Also, the prediction errors, measured by MAE (2.1699) and RMSE (2.3250), are significantly higher than on the training set, which highlights a problem of generalization of the model, which fails to maintain the same level of performance when applied to new and unknown data.

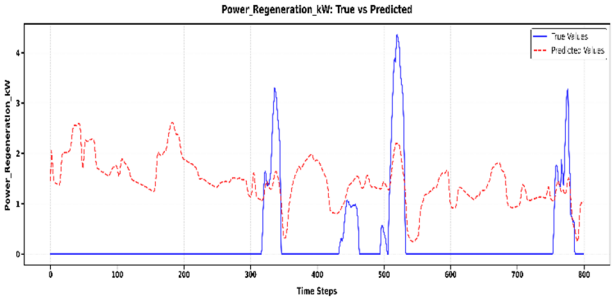

Fig. 8. True vs Predicted: Power Regeneration (Testing)

V. CONCLUSION

In this study, we developed a Graph Convolutional Network (GCN) artificial intelligence model for predicting Power_Regeneration_kW values. The model was trained and tested using an EV-specific dataset, and its performance was evaluated by standard metrics.

The results obtained indicate that the model effectively learned the relationships from the training set, obtaining an R^2 Score of 0.9977, which shows that it explains 99.81% of the variability of the real data. Also, the errors of the MFA (0.0143) and RMSE (0.0367) are very small, confirming that the predictions are extremely close to the real values.

However, the evaluation on unknown data revealed a significantly worse performance, with the model obtaining a negative R^2 Score (-8.5345), and the MAE (2.1699) and

RMSE (2.3250) errors being considerably higher. This suggests that the current architecture cannot make reliable predictions in new scenarios, possibly due to the lack of additional relevant features such as GPS data, which could give the model a clear reference point based on location and route [7], [8].

ACKNOWLEDGMENT

I would like to express my sincere gratitude to Solis UTCN project and Solis team for granting access to their valuable competition database. This research would not have been possible without their willingness to share data collected during ILUMEN 2024 solar car competition. The comprehensive dataset from Solis EV4 vehicle provided the foundation for developing and validating GCN-based regenerative brake energy prediction system presented in this paper.

I am also grateful to open-source community for developing and maintaining the tools and frameworks that made this research possible. Their contributions to the field of machine learning, particularly in the development of Graph Convolutional Networks, have been invaluable for this work.

REFERENCES

[1] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," in International Conference on Learning Representations (ICLR), 2017.

[2] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," AI Open, vol. 1, pp. 57-81, 2020.

[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436-444, 2015.

[4] M. Hannan, M. S. H. Lipu, A. Hussain, and A. Mohamed, "A review of lithium-ion battery state of charge estimation and management system in electric vehicle applications: Challenges and recommendations," Renewable and Sustainable Energy Reviews, vol. 78, pp. 834-854, 2017.

[5] P. Zhang, F. Yan, and C. Du, "A comprehensive analysis of energy management strategies for hybrid electric vehicles based on bibliometrics," Renewable and Sustainable Energy Reviews, vol. 48, pp. 88-104, 2015.

[6] B. Wang, J. Xu, B. Cao, and X. Zhou, "A novel multimode hybrid energy storage system and its energy management strategy for electric vehicles," Journal of Power Sources, vol. 281, pp. 432-443, 2015.

[7] Z. Chen, B. Xu, M. Wang, and H. Li, "Regenerative Braking System Optimization Using Artificial Neural Networks in Electric Vehicles," Energy Conversion and Management, vol. 230, 113796, 2021.

[8] L. Li, X. Yan, W. Wang, and Z. Wang, "Deep Learning for Energy Management in Electric Vehicles: A Comprehensive Review," IEEE Access, vol. 9, pp. 51023-51040, 2021.

[9] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825-2830, 2011.

[10] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," in International Conference on Learning Representations (ICLR), 2019.

[11] P. J. Huber, "Robust Estimation of a Location Parameter," The Annals of Mathematical Statistics, vol. 35, no. 1, pp. 73-101, 1964.

[12] T. Chai and R. R. Draxler, "Root mean square error (RMSE) or mean absolute error (MAE)?," Geoscientific Model Development Discussions, vol. 7, pp. 1525-1534, 2014.