AI-driven power consumption analysis for a solar electric vehicle

Petric Alexandra
Faculty of Automation and Computer Science
Cluj-Napoca, Romania
petric.al.alexandra@student.utcluj.ro

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

***Abstract*— The purpose of this paper is to emphasize the dynamics between Solis vehicle's velocity and energy consumption and regeneration, analysis based on the data gathered from the latest competition in Belgium. We have trained an AI model to give us the statistics that will furthermore allow us to improve our future vehicle model's performance.**

Keywords— graphical convolutional networks, artificial intelligence, machine learning, electric vehicle *energy consumption.*

I. INTRODUCTION

Solar-powered vehicles promise responsible energy consumption, but their efficiency stems from advancements in battery storage, electrical efficiency and efficient driving patterns. The approach was to create an AI-based model using GCN (Graph Convolutional Networks) to analyze telemetry data and get insights on possible future improvements.

GCN is a multilayer graph convolutional neural network, where each convolutional layer processes only first-order neighborhood information, and by stacking several convolutional layers, information transfer in multiple-order neighborhoods can be achieved [1].

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 [2], [3].

The GCN model serves three primary functions in optimizing energy consumption for our solar vehicle:

1. Estimating real-time energy consumption - The model analyzes telemetry data and provides energy consumption predictions based on motor velocity and power usage.
2. Simulating vehicle energy behavior in different conditions - By modifying input parameters, the model can simulate how energy consumption fluctuates under various driving patterns and environmental conditions.
3. Optimizing energy management strategies - The insights from the model can be used to develop improved driving techniques, and fine-tune vehicle parameters to maximize efficiency in competitions.

Based on the results obtained, the team can further improve strategies in competitions and increase overall energy performance.

II.GCN ARCHITECTURE

In the model discussed here, the focus is on analyzing the relationship between Motor Velocity and Power Consumption. Each entry in the dataset represents a vertex in the graph, and edges connect each consecutive timestamp at which data is registered. GCNs are composed of stacked graph convolutional layers in a similar way that traditional CNNs are composed of convolutional layers. Each convolutional layer takes as input the nodes' vectors from the previous layer (for the first layer this would be the input feature vectors) and produces corresponding output vectors for each node [4]. The model consists of 7 convolutional layers, all having a consistent width of 128 units and a dropout of 0.3 between them for regularization.

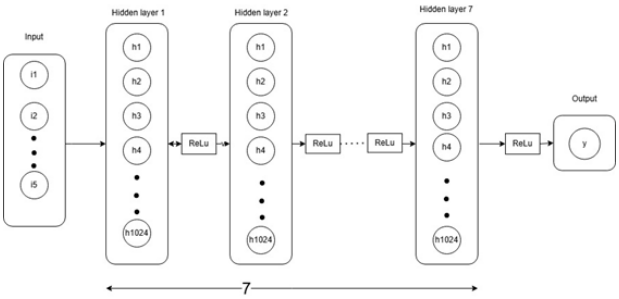

Fig. 1. Structure of GCN.

Each convolutional layer is defined through the following mathematical formula:

$$H = \sigma(\tilde{D}^{\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}X\Theta) \quad (1)$$

where, H represents the matrix of node embeddings h_m, X denotes the matrix of node features x_m, and $\sigma(\cdot)$ is an activation function, in our case ReLU. The matrix $\tilde{A}$ corresponds to the graph adjacency matrix, enhanced with self-loops, while $\tilde{D}$ represents the degree matrix, also adjusted with self-loops. Lastly, Θ is a trainable parameter matrix that helps optimize the model.

The ReLU activation function is responsible for the management of negative input data and is a part of the data pre-processing process. The function returns 0 if it receives any negative input, but for any positive value it returns that value back, therefore it can be written as [5], [6]:

$$ReLu(x) = x^+ = max(0,x) = \frac{x+|x|}{2} = \begin{cases} x \; if \; x > 0 \\ 0 \; if \; x \leq 0 \end{cases} \quad (2)$$

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. Below is the basic representation:

```
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.conv6 = GCNConv(hidden_dim, hidden_dim)
        self.conv7 = GCNConv(hidden_dim, hidden_dim)
        self.fc = torch.nn.Linear(hidden_dim, output_dim)
        self.dropout = torch.nn.Dropout(0.3)

    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.conv6(x, edge_index).relu()
        x = self.dropout(x)
        x = self.conv7(x, edge_index).relu()
        x = self.dropout(x)
        x = self.fc(x)
        return x
```

Fig. 2. GCN code representation.

To better illustrate how information flows through a GCN, the image below depicts how predictions are generated based on known factors[7]:

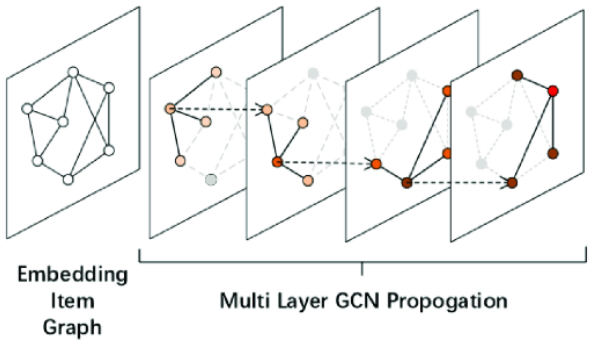

Fig. 3. Schematic of information propagation in GCN [7].

III. DATASET DESCRIPTION AND PREPROCESSING

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 [8]. The Solis EV4 is the latest and most efficient solar vehicle developed by the Solis team, incorporating numerous improvements over previous models.

The input values are represented by sets of (Timestamp, Motor Velocity, Power Consumption (+) and Power Consumption (-)).

A. Data Preprocessing

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

1. Data Normalization: It was applied MinMaxScaler to scale the data. This normalization enhances training stability and accelerates model convergence, as the MinMaxScaler transforms features by scaling each feature to a given range.
2. Elimination of Negative Values: By separating positive and negative components, the model can recognize distinct patterns and analyze the behavior of each component independently.

Below is an example of processed data:

TABLE I. PROCESSED DATA SAMPLE

time	time_diff	Power_Co	Energy_Co	Distance_	Consumpt	MC_Motor	MC_MotorVelocity_ne
2024-09-2	0.185762	0	0	5.43E-05	0	0	10.94464
2024-09-2	0.067141	0	0	1.96E-05	0	0	10.93179
2024-09-2	0.294513	0	0	8.46E-05	0	0	10.74459
2024-09-2	0.198996	0	0	5.73E-05	0	0	10.76739
2024-09-2	0.049928	0.024611	3.41E-07	1.42E-05	0.024089	0	10.61965
2024-09-2	0.410612	0.025112	2.86E-06	0.000124	0.023035	0	11.33155
2024-09-2	0.193648	0.024163	1.30E-06	6.27E-05	0.020742	0	12.10805
2024-09-2	0.013602	0.024246	9.16E-08	4.71E-06	0.01947	0	12.94423
2024-09-2	0.237771	0.023173	1.53E-06	8.78E-05	0.017426	0	13.82243
2024-09-2	0.386892	0	0	0.000151	0	0	14.65
2024-09-2	0.048419	0.000214	2.87E-09	2.10E-05	0.000137	0	16.23742
2024-09-2	0.203844	0	0	9.07E-05	0	0	16.64333
2024-09-2	0.380726	0	0	0.000171	0	0	16.79007
2024-09-2	0.065232	0	0	2.81E-05	0	0	16.10052
2024-09-2	0.196775	0	0	8.31E-05	0	0	15.80134
2024-09-2	0.187776	0	0	7.72E-05	0	0	15.37923
2024-09-2	0.186235	0.004467	2.31E-07	7.29E-05	0.003168	0	14.65604
2024-09-2	0.05625	0.007364	1.15E-07	2.23E-05	0.005168	0	14.81149
2024-09-2	0.304244	0.010669	9.02E-07	0.000119	0.007555	0	14.67861
2024-09-2	0.245606	0.012365	8.44E-07	9.87E-05	0.008551	0	15.03076
2024-09-2	0.213641	0.014425	8.56E-07	8.84E-05	0.009678	0	15.49177

IV. TRAINING PROCESS

The training process of the Graph Convolutional Network (GCN) involves iteratively updating the model's weights to minimize the prediction error. The input features are forwarded through multiple graph convolutional layers, where each node aggregates information from its neighbors, capturing spatial relationships in the data.

A. Loss function

The Huber loss function describes the penalty incurred by a wrong prediction procedure. The formula for it is:

$$L_\delta(a) = \begin{cases} \frac{1}{2}x^2, & if \; |x| \leq \delta, \\ \delta\left(|x| - \frac{1}{2}\delta\right), & if \; |x| > \delta, \end{cases} \quad (3)$$

By its mathematical definition, it does not penalize more for outliers, leading to more stable and reliable predictions, and provides an optimal compromise between Mean Squared

Error (MSE) and Mean Absolute Error (MAE). The loss parameter used is δ = 1, which was chosen based on empirical testing.

A. Optimization

For optimization, the model uses *AdamW*, a variant of *Adam* (short for Adaptive Moment Estimation) which helps models train efficiently while maintaining good generalization by properly handling weight decay. In this case, the parameters used are a learning rate $\alpha = 0.001$ and a weight decay $\lambda = 1 \times 10^{-4}$. Such values ensure that overfitting of the data is avoided. The general formula for the *AdamW* optimizer is:

$$\theta_t = \theta_{t-1} - \alpha * m_t / \left(\sqrt{v_t} + \varepsilon\right) - \lambda * \theta_{t-1} \quad (4)$$

Below are the exact parameters used in the code:

```
optimizer = torch.optim.AdamW(model.parameters(), lr=0.01, weight_decay=1e-4)
scheduler = torch.optim.lr_scheduler.ReduceLROnPlateau(optimizer, 'min', patience=50, verbose=True)
criterion = torch.nn.HuberLoss(delta= 0.8)
```

Fig. 4. *AdamW* code representation.

B. Use of data

The telemetry dataset is split into train data and test data on a 80% - 20% ratio, to ensure balanced learning and minimize possible prediction errors. Through experimentation, we determined that 200 epochs provide the best balance between model performance and avoiding overfitting on the training data. This dataset is intended solely for the final evaluation of the model, mimicking real-world situations where the model must predict outcomes based on entirely new and unfamiliar data.

The evolution of the training loss over the 200 epochs is illustrated in the graph below:

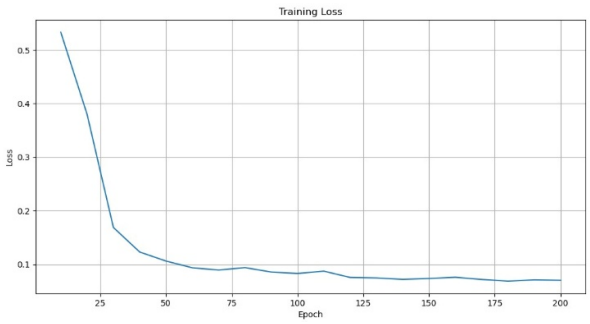

Fig. 5. Training loss over 200 epochs.

V. EXPERIMENT

The model's results are interpretable through code-generated diagrams that will be displayed in the following.

A. Experimental Setup

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

- MAE (Mean Absolute Error): measures the average of absolute errors [9].

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

- RMSE (Root Mean Squared Error): measures the square root of the mean of the square errors. [10]

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

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

$$1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \quad (7)$$

B. Results and Analysis

Below is displayed the prediction of Power Consumption over different time steps, prediction using solely the dataset provided and the 80 - 20% data split:

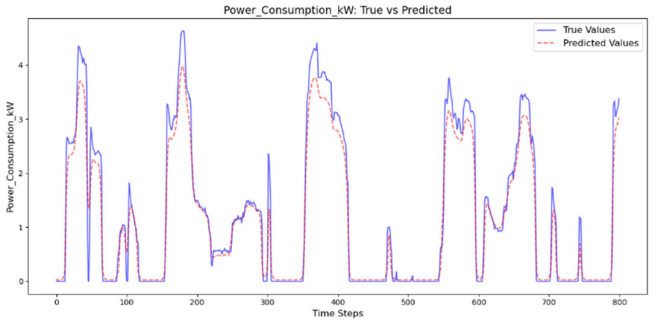

Fig. 6. Comparison of actual vs. predicted power consumption values on training data.

As for testing the model on entirely new data, the graph below is the best representation:

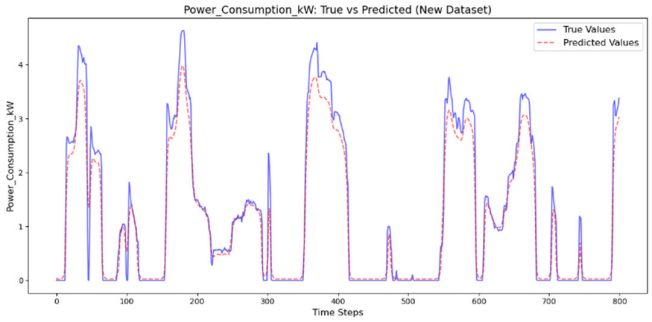

Fig. 7. Comparison of actual vs. predicted power consumption values on unseen test data.

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

TABLE II. GCN MODEL PERFORMANCE

Metric	Test Dataset	Train Dataset
MAE	0.3095	0.1507
RMSE	0.5371	0.2560
R^2 Score	0.9214	0.9782

On the test set, a slight degradation in the model's performance is visible, from 97% to 92%.

These test dataset results provide an overview of the model's accuracy when handling unseen data. These predictions are critical for optimizing the vehicle's energy consumption in competitions. The 92% efficiency indicates that the model can reliably predict energy usage, allowing for:

- Better route and driving strategy planning, reducing unnecessary power consumption.
- Informed decision-making during competitions, enabling the team to allocate energy more efficiently, and identification of potential inefficiencies in energy management.

By leveraging these results, Solis can enhance performance and sustainability, potentially gaining an advantage in future races.

VI. CONCLUSION

Predicting energy consumption based on parameters like motor velocity is crucial for optimizing the performance of solar vehicles in real-world conditions. AI-based models, like the one presented, not only provide accurate predictions but also offer insights into ideal values and conditions for maximizing energy efficiency, ultimately promoting sustainability. With a larger dataset, models like this have the potential to evolve from theoretical applications to industrial usability, further advancing the future of solar-powered transportation.

ACKNOWLEDGMENT

I would like to express my gratitude towards Solis UTCN project and to the Solis team for granting access to their valuable competition database, and for offering their valuable input. 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 our GCN-based system for predicting power consumption based on motor velocity.

REFERENCES

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

[2] K. Liu, Y. Yang, W. Zhang, and H. Liu, "A Graph Convolutional Neural Network-based Method for Short-term Photovoltaic Power Forecasting," in Proc. 2020 IEEE International Conference on Smart Grid and Clean Energy Technologies (ICSGCE), pp. 304-309, 2020

[3] Peter J. Huber. "Robust Estimation of a Location Parameter." Ann. Math. Statist. 35 (1) 73 - 101, March, 1964.

[4] M. N. Bernstein, "Graph Convolutional Neural Networks," *Personal Blog*, 24 September 2023.

[5] J. Brownlee, "A Gentle Introduction to the Rectified Linear Unit (ReLU)," *Machine Learning Mastery*, 8 January 2019.

[6] D. Liu, "A Practical Guide to ReLU," *Medium*, 30 November 2017.

[7] M. Zhang and Z. Yang, "GACOforRec: Session-Based Graph Convolutional Neural Networks Recommendation Model," *IEEE Access*, vol. 7, pp. 114077-114085, 2019.

[8] 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.

[9] *Solis EV4, Solis UTCN*. https://solis.utcluj.ro/en/articles?id=7

[10] 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