applied sciences

MDPI

Article

Real Driving Cycle-Based State of Charge Prediction for EV Batteries Using Deep Learning Methods

Seokjoon Hong, Hoyeon Hwang, Daniel Kim, Shengmin Cui and Inwhee Joe *

Department of Computer Science, Hanyang University, Seoul 04763, Korea; daniel379@hanyang.ac.kr (S.H.); kd3122@hanyang.ac.kr (H.H.); danielkim96@hanyang.ac.kr (D.K.); shengmincui@hanyang.ac.kr (S.C.)
* Correspondence: iwjoe@hanyang.ac.kr; Tel.: +82-02-2220-1088

Abstract: An accurate prediction of the State of Charge (SOC) of an Electric Vehicle (EV) battery is important when determining the driving range of an EV. However, the majority of the studies in this field have either been focused on the standard driving cycle (SDC) or the internal parameters of the battery itself to predict the SOC results. Due to the significant difference between the real driving cycle (RDC) and SDC, a proper method of predicting the SOC results with RDCs is required. In this paper, RDCs and deep learning methods are used to accurately estimate the SOC of an EV battery. RDC data for an actual driving route have been directly collected by an On-Board Diagnostics (OBD)-II dongle connected to the author's vehicle. The Global Positioning System (GPS) data of the traffic lights en route are used to segment each instance of the driving cycles where the Dynamic Time Warping (DTW) algorithm is adopted, to obtain the most similar patterns among the driving cycles. Finally, the acceleration values are predicted from deep learning models, and the SOC trajectory for the next trip will be obtained by a Functional Mock-Up Interface (FMI)-based EV simulation environment where the predicted accelerations are fed into the simulation model by each time step. As a result of the experiments, it was confirmed that the Temporal Attention Long–Short-Term Memory (TA-LSTM) model predicts the SOC more accurately than others.

Keywords: electric vehicle; real driving cycle; recurrent neural network; simulation; state of charge; temporal attention

Citation: Hong, S.; Hwang, H.; Kim, D.; Cui, S.; Joe, I. Real Driving Cycle-Based State of Charge Prediction for EV Batteries Using Deep Learning Methods. *Appl. Sci.* **2021**, *11*, 11285. https://doi.org/10.3390/app112311285

Academic Editors: João M. F. Rodrigues, Pedro J. S. Cardoso and Marta Chinnici

Received: 23 October 2021
Accepted: 22 November 2021
Published: 29 November 2021

1. Introduction

Since Electric Vehicles (EVs) do not emit CO_2, they have a great potential to prevent air pollution [1]. In addition, it is attractive for drivers to use EVs, because the cost of charging the battery of an electric vehicle is much cheaper than the cost of refueling [2]. In recent years, the performance of electric vehicle batteries has greatly improved and, consequently, there are EVs capable of driving more than 500 km when fully charged. However, there are few battery-charging stations compared to conventional gas stations, and batteries take a long time to charge [3]. Therefore, when driving an electric vehicle, it is very important to predict the driving distance through the state-of-charge (SOC) of the battery.

Accurate battery SOC measurement is an important feature in the BMS of electric vehicles, which can be implemented with microcontrollers (MCUs). Recently, as pre-trained neural network models have been installed and used in automotive MCUs, it has become possible to apply AI to the Battery Management System (BMS) [4]. In addition, among recent studies, a method to more accurately measure SOC using a cloud data center that provides high-capacity and high-performance calculations for big-data-based, data-driven Deep Learning (DL), and interworking with the vehicle's BMS, was also proposed [5].

Future applications for fully Connected and Autonomous Vehicles (CAVs) include an AR-based navigation system, a video-conferencing application, real-time image-processing and inferencing solutions implemented by a neural network model equipped on board [6,7]. Since these applications require a high level of computation power and low power consumption, various hardware implementations such as a Graphics Processing Unit (GPU),

field-programmable gate array (FPGA), and Application-Specific Integrated Circuits (ASIC) are employed. Specifically, a Tesla Full Self-Driving (FSD) computer, built on ASIC, meets 50 Tera floating point operations per second (TFLOPS) of the Neural Network model, and uses 100 W or less power consumption during computation [8]. To benefit from these state-of-the-art embedded systems adapted to the full CAVs, it is important to both estimate the current SOC and predict the future SOC to aid the drivers in decision-making.

It is important to accurately measure the current SOC, but it is also necessary to predict the future SOC based on the current driving data. An accurate prediction of the SOC can help determine the possible driving distance and charging time. There are two main methods to predict the SOC of an EV battery. The first one is predicting the SOC through simulation based on the vehicle dynamics. This approach was implemented by [9], which used standard driving cycles (SDC) as input of the simulation model. The second method consists of predicting the SOC through Machine Learning (ML) techniques using the battery data collected while driving the electric vehicle [10]; for example, using information from the battery itself (voltage, current, temperature) and ML algorithms to predict the future SOC.

Recently, there have been studies to predict SOC through complex mathematical formulas, considering the actual driving cycle of electric vehicles, but ML was not used.

In this paper, we propose an algorithm that accurately predicts the SOC of an EV battery by acquiring actual driving information through OBD-II, dividing and preprocessing this driving information by section based on GPS information and DTW, training a neural network with the dataset and running simulations using Functional Mock-Up Units (FMU).

2. Related Work

This section classifies the related studies into two major categories: modeling and simulation, and SOC estimation and prediction in the EV domain. Since there have been significant research efforts in these fields, we are focused on work that has been conducted rather recently, and discuss the limitations of these works.

First of all, in the field of modeling and simulation, extensive research has been carried out to improve the accuracy of EV energy consumption estimation with high-fidelity simulation models [11–17]. By using simulation environments such as MATLAB/Simulink and DACCOSIM, each physical component of an EV is mathematically modeled to form a subsystem block, which is composed of primitive blocks. The simulation is performed by the subsystem blocks mapping the input signals to the output values, which are passed to the subsequent blocks over time during the simulation. Recently, distributed co-simulation based on Functional Mock-up Interface (FMI), which provides a standardized interface to ease the sharing of different models [18], has been widely used. Techniques to reduce the simulation time while maintaining the simulation accuracy have also been developed. S. Hong proposed a redundancy reduction algorithm (RRA) to increase the simulation speed by adaptively adjusting the simulation step size, increasing it when a specific pattern of the cycle is detected as repeated, and decreasing it when a zero-crossing point is detected [19]. However, this is mainly applicable to the SDCs, especially the New European Driving Cycle (NEDC), where many repetitive parts are known ahead of the simulation.

These studies heavily rely on the SDCs to validate the methods. However, as mentioned above, SDCs have inherent limitations, as they are not real driving cycles (RDCs). Even though a variety of SDCs have been developed to emulate urban, rural, and highway environments, they do not reflect personalized factors, i.e., individual drivers' habits, which are highly complicated as they dynamically change over time [20,21]. Moreover, they can hardly incorporate factors such as traffic conditions, traffic signals, and time of driving, which are unique to each driving instance [22]. Therefore, in order to provide more personalized simulation results, both the mathematically defined models and data-driven methods that learn the drivers' patterns from the previous RDCs are required.

Next, there have been a myriad of studies in the area of SOC estimation and prediction. In the application of the pure EV, accurate prediction of the battery SOC consumption is highly important, since this would directly lead to an accurate prediction of the EV range [23]. According to the literature [24–26], methodological approaches to the SOC estimation can be divided into two: ML-based and non-ML-based methods. In brief, Coulomb counting and open-circuit voltage (OCV) methods have been used as conventional methods, while, recently, ML models consisting of deep neural networks (DNNs) have been intensely applied [24–26]. In the ML-based data-driven approach, the SOC values are predicted by automatically adjusting parameters to minimize the resultant error by learning from the pattern of the input features such as current, voltage, and temperature [27–29]. In this way, the models are fitted to the specific real-world measurement data, which enables predictions with enhanced accuracy under similar conditions.

However, few studies have been conducted on the prediction of future SOC trends through simulation. Most of the research has been focused on the internal parameters of the battery management system (BMS) [30] to estimate the SOC value. That is, research on the relationship between the EV subsystems in an integrated view to improve the performance of the simulation, as well as reducing the overhead of the tedious model-based simulation, has been insufficient. If an individual driver's OBD data can be repeatedly collected on the road, one can simulate the SOC patterns for the next cycle for the same segment of the route with the velocity and acceleration data as inputs to an EV model. In this way, an approximate trajectory of the SOC consumption can be obtained, which can, in turn, help the driver be aware of the possible range and adjust her driving behavior from the perspective of an SOC.

3. Overall Procedure for Real Driving Cycle-Based SOC Estimation

This paper proposes a method to predict SOC using a real-world driving cycle. The proposed SOC prediction method using real-world driving cycle is as shown in the Figure 1.

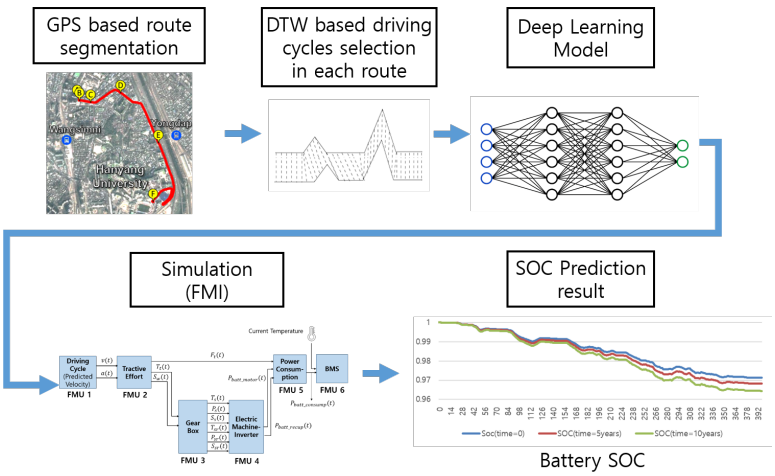


Figure 1. Procedure of Real Driving cycle based SOC estimation.

The tested driving routes run from a fixed site to the other, i.e., from the homeplace of one of the authors to a parking lot in the Hanyang University campus. The route consists of several intersections between the start point and the destination point. The driving data were collected onboard while the author was driving, using an application named Torque Pro running on an Android device connected to a Bluetooth dongle, which supports the OBD-II specifications. The OBD data were comprised of various second-by-second temporal data, such as GPS latitude, longitude, velocities, etc., and were saved in a Comma-Separated-Value (CSV) format. The route is highlighted, as shown in Figure 2. The route was extracted and drawn on the map using the Google API with the GPS latitude and longitude information from the OBD data.

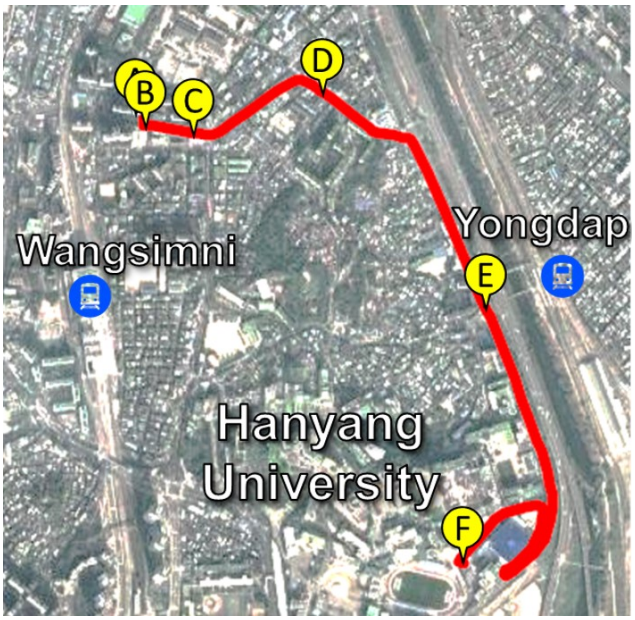


Figure 2. Real Driving Cycle with Google MAP.

Since the overall driving cycles are highly dependent on the operations of en-route traffic lights, to enhance the accuracy of the predicted driving cycles, we first divided each of the end-to-end routes into sub-routes by the traffic lights (as can be seen from Figure 2 as circular markers). The segments obtained from different instances of the route were then compared to one another. To find the most similar intervals across all of the instances of the route, a Dynamic Time Warping (DTW) algorithm was employed. After grouping the segments of the route by their similarity, different Machine Learning (ML) algorithms were used to learn from them and predict future driving cycles.

To validate the performance of the prediction of the SoC consumption, we generated the eventual SoC values by an Electric Vehicle (EV) simulation model, implemented based on Functional Mock-up Units (FMU). The FMU-based EV simulation model reads as inputs the velocities, calculates the electric current from power consumption and regeneration by acceleration and deceleration, respectively, and writes as outputs the resultant SoC values to a file at each second. As expected, the test results showed that the SoC values could be predicted with high accuracy, as long as the driving cycle was predicted with high accuracy.

The paper is structured as follows: first, we proposed the preprocessing method using the DTW algorithm to group the driving intervals by similarity. Secondly, we proposed a deep learning algorithm that accurately predicts the speed after learning from similar driving cycles. The speed was also predicted using the existing Seasonal Autoregressive Integrated Moving Average (SARIMA) model. Finally, to evaluate the models, the predicted speeds by both algoritms were used as input to a simulation model, which computes the SOC of the driving cycle.

4. Methodology

4.1. Dynamic Time Warping (DTW) for Similarity of Real Driving Cycles

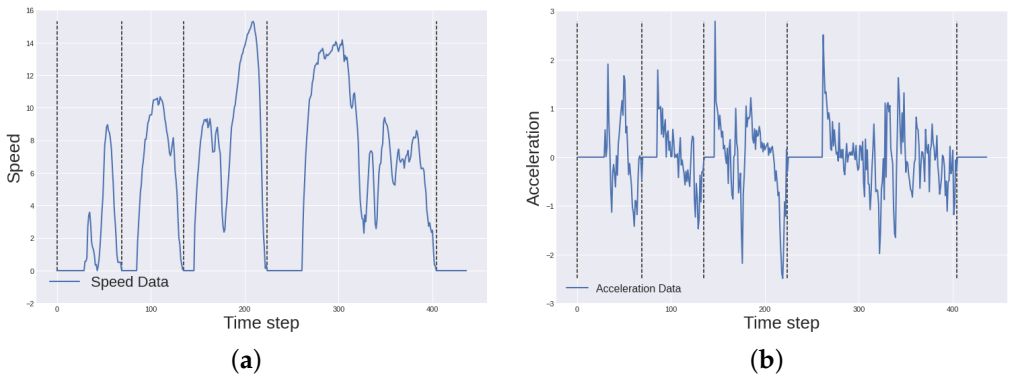
First proposed by [31] in the speech recognition field, the DTW is an algorithm that measures similarity between time series, which may vary in length. The technique performs non-linear warpings so that the time series are stretched or shrinked in order to find the optimal alignment between them. It has a wide range of applications, such as data mining, gesture recognition, robotics, speech-processing, manufacturing and medicine [32].

We propose a method to find similar driving routes based on the DTW algorithm, using the acceleration of the vehicle as input. The reason why acceleration is used is that the SOC is closely related to power consumption, which, in turn, is closely related to the acceleration of the vehicle. There were more than 30 driving-cycle-related data collected

and, due to the large noise and uncertainty existent in the dataset, only similar segments were used for the prediction. Therefore, the goal of the algorithm is to find the most similar acceleration time series segments to use them as input to the ML models.

The algorithm that was used can be described in the following manner:

- First of all, the *base* time series which we want to use to find other similar time series must be defined. For this present work, we defined the route shown in the map of the Figure 2, represented by the time series in the Figure 3, as the base route to which the remaining 30 time series will be compared.
- Secondly, the base time series may further be split into an arbitrarily number of base segments. Considering the Figures 2 and 3, the markers from A to F and the dotted black vertical lines represent situations in which the velocity of the car reached zero or, in other words, the start and the end points of each of the four segments (A to C, C to D, D to E, E to F). Note that the point B was not taken into account, because we regarded it as noise due to its proximity to point A, and that the each of the base segments are shown in greater detail in the subfigures of Figure 4.
- Thirdly, for each of the four *base* segments, we set the lower bound and the upper bound times for the similar patter- finder algorithm by subtracting and adding an ϵ arbitrary small amount of time. For example, the second *base* segment was extracted from second 69 to 135 of the *base* time series. If $\epsilon = 50$, then the lower-bound and upper-bound times for a similar pattern-finder algorithm will be, respectively, 19 s and 185 s, as shown in Table 1.
- With the lower and upper bounds of the algorithm defined, we split the period between the lower and upper bounds into smaller search periods, spaced by an arbitrary time interval. Intuitively, the lower the interval is, the higher the accuracy of the algorithm. These smaller search periods are shown in Table 1 for an interval of five seconds between each period.
- Next, we looped through the search periods.
- Inside the search-periods-loop, we loop through the remaining 30 time series, which will be compared to the *base* time series.
- Next, we extract the corresponding search period of the time series to be used in the comparison.
- Next, we obtain the distance between the extracted time series and the *base* segment by DTW.
- Finally, after the two loops are finished, we rank the all the extracted time series according to their cost, computed by DTW, and obtain the 10 periods that have the lowest cost, which will form the training and test datasets for the machine learning models.



(**a**) (**b**)

Figure 3. Speed and acceleration over time of the base time series. (**a**) Speed $\times$ time of the base time series. (**b**) Acceleration $\times$ time of the base time series.

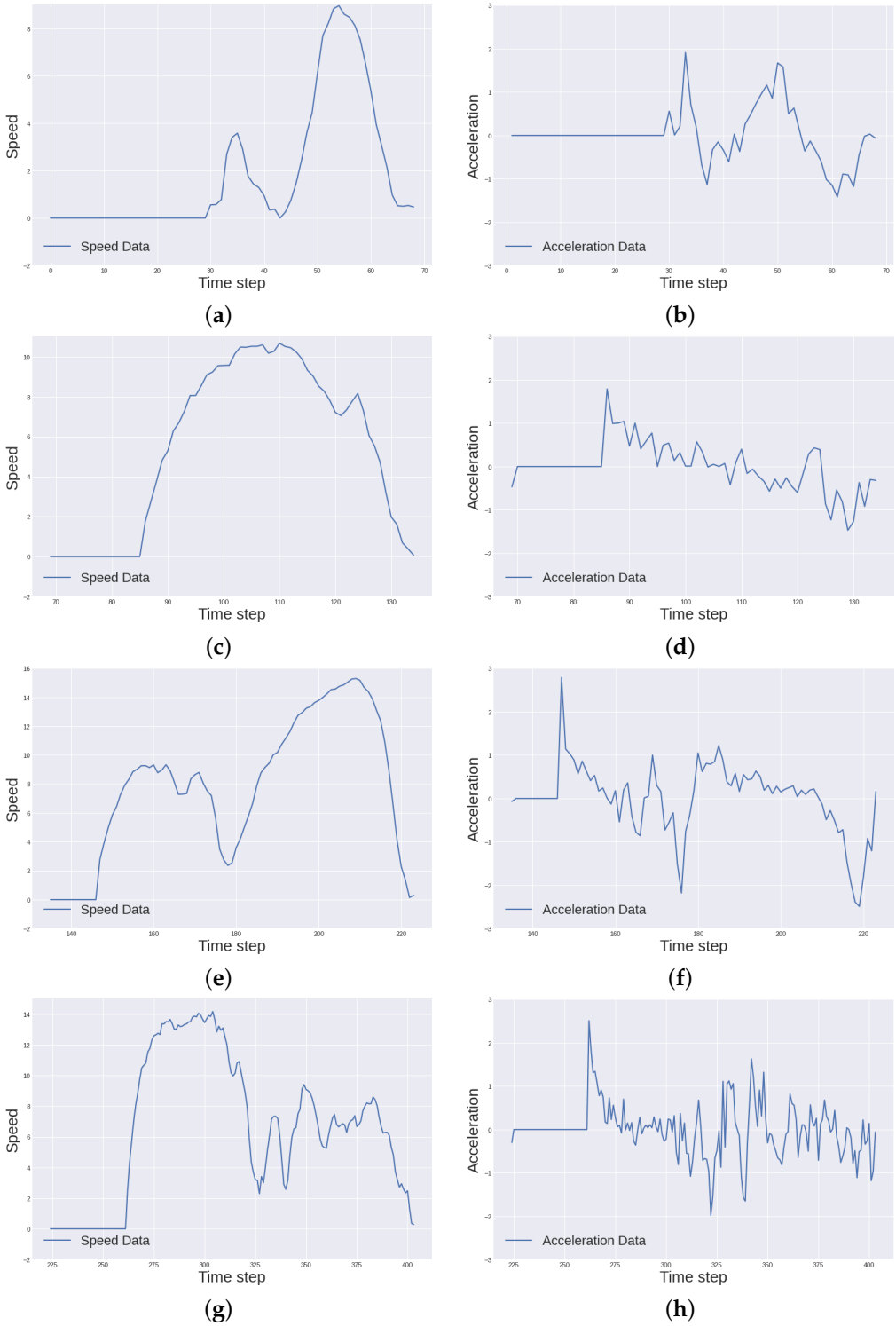


Figure 4. Speed and acceleration over time of the four base segments. (**a**) Speed × Time (A to C segment); (**b**) Acceleration × Time (A to C segment); (**c**) Speed × Time (C to D segment); (**d**) Acceleration × Time (C to D segment); (**e**) Speed × Time (D to E segment); (**f**) Acceleration × Time (D to E segment); (**g**) Speed × Time (E to F segment); (**h**) Acceleration × Time (E to F segment).

Table 1. Table showing the start and end time of the segments and the search periods.

	First Base Segment	Second Base Segment	Third Base Segment	Fourth Base Segment
Start Point (s)	0	69	135	224
End Point (s)	69	135	224	404
Search Start Point (s)	0	19	85	174
Search End Point (s)	119	185	274	454
examles of Search periods for interval = 5 s (start, end)	(0, 69), (5, 74), ... (45, 114), (50, 119)	(19, 85), (24, 90), ... (114, 180), (119, 185)	(85, 174), (90, 179), ... (180, 269), (185, 274)	(174, 354), (179, 359), ... (269, 449), (274, 454)

A summary of the algorithm is shown in Algorithm 1.

Algorithm 1 Similar time series finding algorithm

```
Set base_time_series
Split base_time_series into base_segments
for bs in base_segments do
    Set lower_bound and upper_bounds
    Set interval
    Compute search_periods
    Initialize empty list costs
    for period in search_periods do
        for ts in time_series_list do
            Extract period from ts
            cost ← DTW(extracted_ts_period, bs)
            Append cost to costs
        end for
    end for
    get_n_series_least_cost(costs, num_series)
end for
```

4.2. Deep Learning-Based SOC Prediction

In this section, our intention is to predict the future speed from the past speed of the EV; hence, we consider this task as a univariate time series prediction task. Given the observed time series data $\mathbf{x} = [x_1, x_2, \ldots, x_T] \in \mathbb{R}^T$, the task is to predict the future value $x_{T+1} \in \mathbb{R}$. Formally, we intend to predict $\hat{x}_{T+1} \in \mathbb{R}$ through a function f, as follows:

$$\hat{x}_{T+1} = f(x_1, x_2, \ldots, x_T), \tag{1}$$

where $f(\cdot)$ is a linear or nonlinear function that needs to be learned.

Our main contribution is presenting a model for predicting speed, which is based on an LSTM with a temporal attention mechanism. The architecture of the prediction model is shown in Figure 5. First, we use an LSTM layer to encode the information from the input sequence into a feature representation. The final prediction is then made by utilizing the temporal attention mechanism over the output features of the LSTM layer.

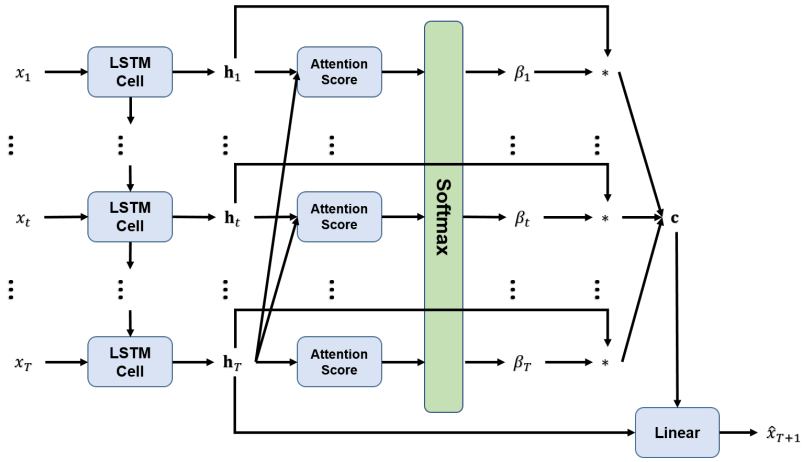


Figure 5. Architecture of TA-LSTM.

Recurrent Neural Network (RNN) is one of the family of neural networks specialized in processing sequential data. If a general feed-forward neural network approach is used to process fixed-length data, the network has separate parameters for each input feature, so it needs to learn them separately at each time position. In contrast, an RNN shares weights across multiple time steps. This sharing of weights is important, as it allows for the generalization of unseen sequences and the sharing of statistical strength across time steps. Long short-term memory (LSTM) and gated recurrent unit (GRU), which are variants of RNN, are commonly used for handling sequential data such as language modeling [33,34] and time series prediction [35,36]. LSTM outperforms RNN and GRU for many sequence prediction tasks because of its gating systems, so we chose LSTM for this task. The forward process of an RNN cell is defined as:

$$\mathbf{f}_t = \sigma(\mathbf{W}_f[x_t, \mathbf{h}_{t-1}] + \mathbf{b}_f) \tag{2}$$
$$\mathbf{i}_t = \sigma(\mathbf{W}_i[x_t, \mathbf{h}_{t-1}] + \mathbf{b}_i) \tag{3}$$
$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{W}_c[x_t, \mathbf{h}_{t-1}] + \mathbf{b}_c) \tag{4}$$
$$\mathbf{c}_t = \mathbf{f}_t \cdot \mathbf{c}_{t-1} + \mathbf{i}_t \cdot \tilde{\mathbf{c}}_t \tag{5}$$
$$\mathbf{o}_t = \sigma(\mathbf{W}_\mathbf{o}[x_t, \mathbf{h}_{t-1}] + \mathbf{b}_o) \tag{6}$$
$$\mathbf{h}_t = \mathbf{o}_t \cdot \tanh(c_t) \tag{7}$$

where $\mathbf{W}_f, \mathbf{W}_i, \mathbf{W}_c, \mathbf{W}_o \in \mathbb{R}^{m\times(m+1)}$ are weight matrices, and $\mathbf{b}_f, \mathbf{b}_i, \mathbf{b}_c, \mathbf{b}_o \in \mathbb{R}^m$ are bias to be learned. $\mathbf{f}_t$ is the forget gate, which determines how much previous information is forgotten, $\mathbf{i}_t$ is the input gate, which determines how much new information is added, and $\mathbf{o}_t$ is the output gate, which, along with the current cell state $\mathbf{c}_t$, determines the output of this cell.

Recently, researchers have proposed multiple attention mechanisms for time series tasks and achieved better results than LSTM and GRU [37–39]. Inspired by the dynamic spatial–temporal attention mechanism [39], we employ the temporal attention mechanism to predict the acceleration of EVs in this work, since our task is univariate time series forecasting, which does not require consideration of spatial attention. Therefore, the temporal attention mechanism assigns a weight to each hidden state by correlating the output at each time step with the output at the last time step. Intuitively, for the prediction of the next time step, the current state is very important, but we cannot ignore the state of previous time steps. A general RNN-based prediction model either uses a fully connected layer to connect the hidden states of all time steps, giving a relatively accurate view of the hidden states of all time steps, or uses the last hidden state for prediction. However, we believe that temporal attention chooses the balanced option of adaptively attributing all time steps an attention score according to the relevance of different time steps to the state of the last time step, and finally connecting the fused context vector with the last hidden

state for prediction, which, in turn, enhances the contribution of the current state to the prediction of the next time step.

First, the attention score is computed based on the relevance between the current state and the last state.

$$\text{Attention Score}(\mathbf{h}_t, \mathbf{h}_T) = \mathbf{h}_t^T \mathbf{W}_s \mathbf{h}_T, \quad 1 \leq t \leq T \tag{8}$$

where $\mathbf{W}_s \in \mathbb{R}^{m \times m}$ is learnable weights, which can be trained jointly with the LSTM layer to adaptively learn the correlation of hidden state of each time step with the last time step.

Then, for each time step, the attention score is converted into probabilistic form using the Softmax function, and the attention weight for this timepoint is obtained by

$$\beta_t = \frac{\exp(\text{Attention Score}(\mathbf{h}_t, \mathbf{h}_T))}{\sum_{j=1}^{T} \exp(\text{Attention Score}(\mathbf{h}_j, \mathbf{h}_T))} \tag{9}$$

where the attention weight $\beta_t \in \mathbb{R}$ demonstrates the importance of hidden state h_t for prediction and the Softmax function is applied to ensure all β sum to 1.

Next, the context vector is obtained by aggregating the hidden states of the RNN layer:

$$\mathbf{c} = \sum_{t=1}^{T} \beta_t \mathbf{h}_t \tag{10}$$

where c is a weighted sum of all hidden states and can be considered as an adaptive selection of relevant hidden states among all time steps.

Finally, the prediction $\hat{x}_{T+1}$ can be obtained through the linear combination of context vector and the last hidden state:

$$\hat{x}_{T+1} = \mathbf{W}_l^T [\mathbf{c}, \mathbf{h}_T] + b_o \tag{11}$$

where $\mathbf{W}_l \in \mathbb{R}^{2m}$ and $b_o \in \mathbb{R}$ are learnable parameters.

The proposed model is differentiable, so the learnable parameters can be updated by back propagation. The mean squared error (MSE) is applied as loss function:

$$Loss(x_{T+1}, \hat{x}_{T+1}) = \frac{1}{N} \sum_{i=1}^{N} (x_{T+1}^i - \hat{x}_{T+1}^i)^2 \tag{12}$$

where N is the number of samples.

5. Experimental Results

Before applying the proposed method to the RDCs, we checked the prediction results of an SDC using Seasonal Auto Regressive Integrated Moving Average (SARIMA), a conventional time-series forecasting model. This model is an extended version of ARIMA, supporting the modeling of the seasonal components of the time-series data. We used NEDC as the SDC and predicted the velocity after training the model. The mean absolute percentage error (MAPE) is used as criteria for evaluating the performance.

$$MAPE = \frac{1}{N} \sum_{t=1}^{N} \left| \frac{x_t - \hat{x}_t}{x_t} \right| \tag{13}$$

where x_t and $\hat{x}_t$ are ground truth and predicted value, respectively.

As can be seen from Table 2, the SDC can be predicted accurately enough without using the DL model. However, the accuracy for the RDC is much lower than the SDC. Therefore, we will use the proposed DL method to improve the predictions for the RDC.

Table 2. MAPE velocity results of SARIMA for the SDC (NEDC) and RDC.

Model	SDC (NEDC)	RDC (Four Road Segments)			
		AtoC	CtoD	DtoE	EtoF
SARIMA	0.014	0.06	0.06	0.12	0.35

To validate our proposed model using RDC, we collected real EV driving data through OBD II. Our dataset of EV speed was obtained from actual driving and divided into four road segments: location A to location C, location C to location D, location D to location E, and location E to location F. Each route has 11 driving cycles, and we took the seen driving cycles of each route as the training set and the rest as the test set. We first verified the performance of our speed predictions. by analyzing the performance of our model using the grid search method with different combinations of hyperparameters. The purpose was to find the best combination of hyperparameters and then verify the effect of each component on the results. Once we obtained the best model, we predicted the speed/acceleration along with the SOC values and compared them with some commonly used methods.

5.1. Speed/Acceleration Prediction

5.1.1. Performance with Different Hyperparameters

First, we integrated all the training sets in the dataset and adopted a five-fold cross validation (CV) method to find the best combination of hyperparameters. We implemented our model using the pytorch library and trained and tested it on an Nvidia GTX 1080 Ti GPU. Adam optimizer was used to train the models. The hyperparameters we need to consider here are the time step length T and the hidden state size m of the LSTM layer. For the proposed model, we set T as varying among $[5, 10, 15]$ and m as varying among $[16, 32, 64, 128]$. The average RMSE results for the 5-fold CV are tabulated in Table 3. First, we describe the effect of the time step length T. It is obvious from Table 3 that the average RMSE results for $T = 10$ and $T = 15$ are notably larger than for the case of $T = 5$. Therefore, for speed prediction, the best choice of time step length is 5. Then, for the case $T = 5$, the average RMSE value is the smallest when the hidden state size m equals to 32. In deep learning, too few parameters can lead to underfitting problems while too many parameters can lead to overfitting problems. Therefore, we believe that $m = 16$ belongs to underfitting, $m = 64$ and $m = 128$ belong to overfitting. Thus, the optimal combination of hyperparameters for our model for this dataset is $\{T = 5, m = 32\}$.

Table 3. RMSE results of 5-fold CV.

Hyperparameters		Average RMSE
T	***m***	
5	16	0.4908
	32	0.4880
	64	0.4926
	128	0.4926
10	16	0.4988
	32	0.4998
	64	0.5013
	128	0.4982
15	16	0.5007
	32	0.5055
	64	0.5025
	128	0.5059

5.1.2. Performance with Different Models

First, to verify that the LSTM in TA-LSTM is the preferred encoding layer, we compared LSTM and other RNN variants on four road segments of the dataset. RNN and GRU are commonly used for time series forecasting, and all three models follow the five-fold CV approach introduced in the previous section to identify the best combination of hyperparameters. All three models obtained the minimum average RMSE at $\{T = 5, m = 32\}$. The first seven driving cycles of each road segment were used as the training set and the remaining four driving cycles were used as the test set, and each road segment was trained and tested separately. In addition, 20% of the data in the training set were randomly selected as the validation set during the training process, and the best-performing model in the validation set was saved for testing. The best test results for each model after training multiple times are shown in Table 4, as can be seen in Table 4, LSTM performs the best in all four road sections. Therefore, we choose LSTM as the coding layer in this task.

Table 4. RMSE results of different models.

Models	AtoC	CtoD	DtoE	EtoF
RNN	0.4485	0.3702	0.4549	0.4924
GRU	0.4369	0.3689	0.4572	0.4873
LSTM	0.4240	0.3556	0.4524	0.4864

Then, to verify the effectiveness of temporal attention, we compared the LSTM with TA-LSTM. The training process and the allocation of training and test sets are the same as in the previous experiment. The results of the comparison are depicted in Figure 6. The comparison shows that the addition of temporal attention yields better results.

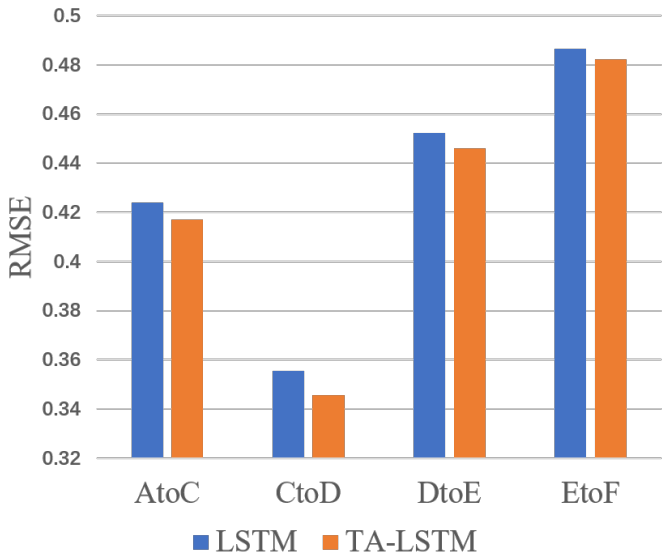
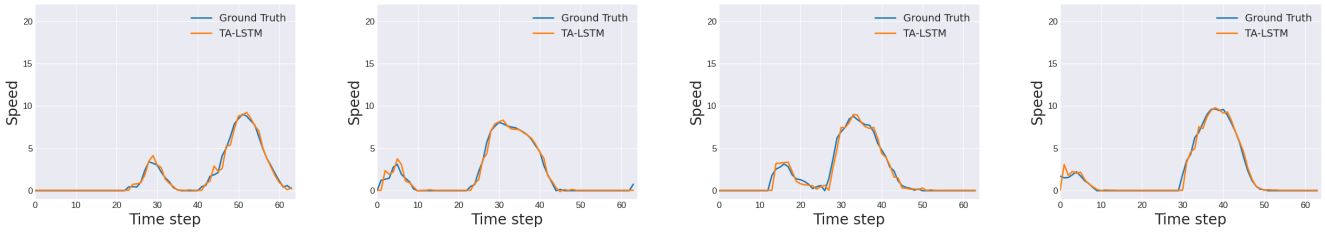


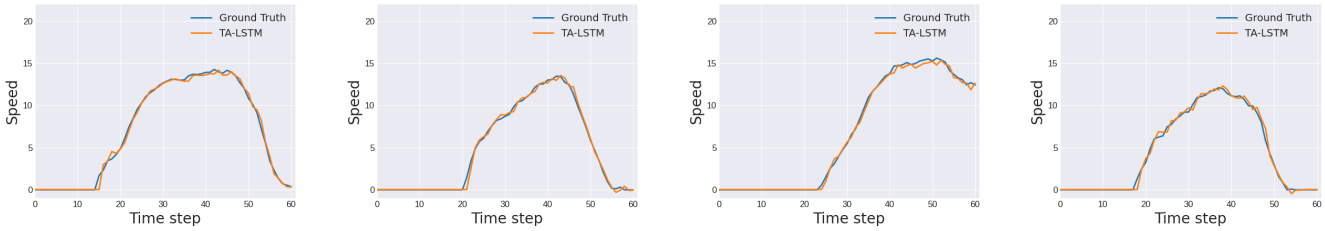
Figure 6. The effect of temporal attention on RMSE results.

Since temporal attention can be combined with various variants of RNN, we compared the performance of different encoding layers combined with temporal attention. From Tables 4 and 5, it can be seen that adding temporal attention to all three different encoding layers can improve the prediction accuracy. TA-LSTM has the best performance. Therefore, we chose TA-LSTM for the task of predicting speed. The comparison between the predicted and true values of each driving cycle for each road section is shown in Figures 7–10. As can be observed from these figures, our model can fit the real data very well in most cases.



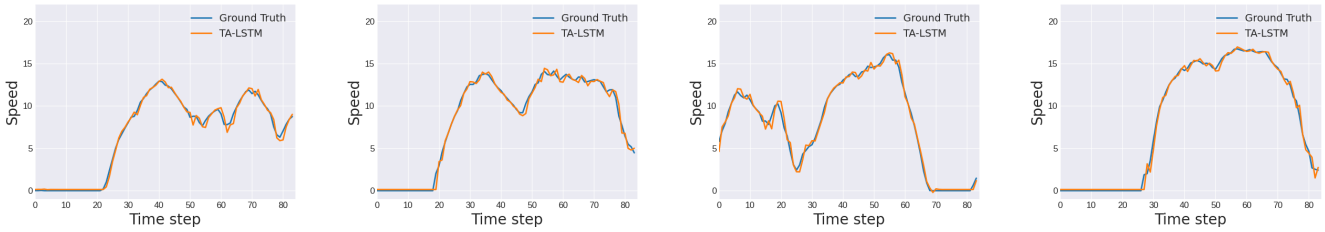
(**a**) (**b**) (**c**) (**d**)

Figure 7. Speed prediction of route A to C: (**a**) cycle 8; (**b**) cycle 9; (**c**) cycle 10; (**d**) cycle 11.



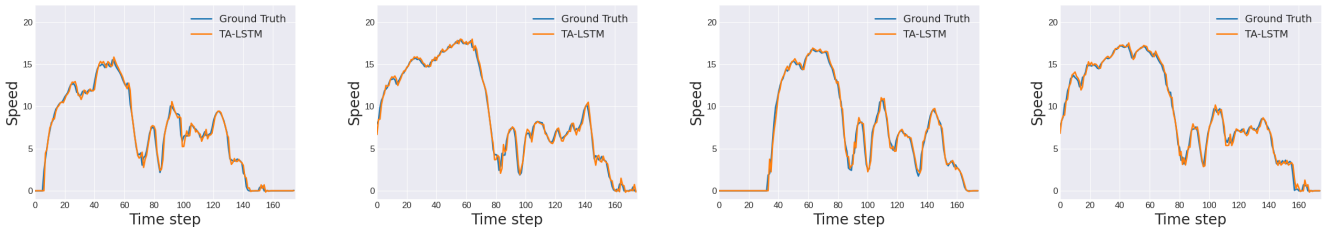
(**a**) (**b**) (**c**) (**d**)

Figure 8. Speed prediction of route C to D: (**a**) cycle 8; (b) cycle 9; (**c**) cycle 10; (**d**) cycle 11.



(**a**) (**b**) (**c**) (**d**)

Figure 9. Speed prediction of route D to E: (**a**) cycle 8; (**b**) cycle 9; (**c**) cycle 10; (**d**) cycle 11.



(**a**) (**b**) (**c**) (**d**)

Figure 10. Speed prediction of route E to F: (**a**) cycle 8; (**b**) cycle 9; (**c**) cycle 10; (**d**) cycle 11.

Table 5. RMSE results of temporal attention with different models.

Models	AtoC	CtoD	DtoE	EtoF
TA-RNN	0.4412	0.3614	0.4515	0.4870
TA-GRU	0.4295	0.3586	0.4527	0.4832
TA-LSTM	0.4171	0.3456	0.4461	0.4821

5.2. SOC Prediction Using FMI-Based EV Simulation

After predicting the speed of the EV, we subsequently utilized the predicted results and our simulator to predict the SOC.

We used FMI-based EV simulation with the previously predicted velocity and acceleration as inputs to predict the EV's SOC. Our EV Simulation model is shown in Figure 11. As can be seen in the figure, given the inputs, the FMU2 to FMU6 are responsible for computing the SOC using the Formulas (14)–(38), whose variables are detailed in Table 6. Additionally, the lookup2d function in Formulas (29)–(31) performs the same operation as lookup2d in MATLAB/Simulink. The necessary efficiency values of the electric machine were obtained through the efficiency curves given in [9].

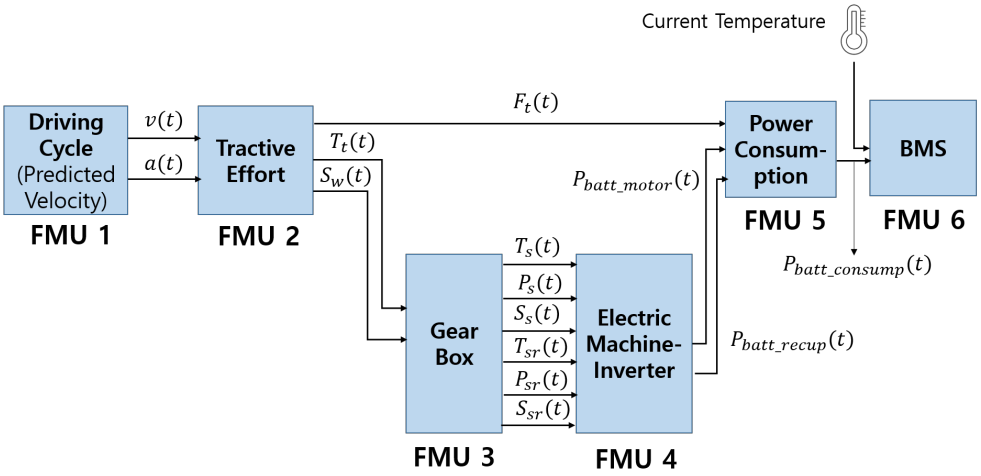


Figure 11. FMI-based model for EV power consumption.

Table 6. Parameters for FMU-based EV simulation (Default setting).

Parameters	Description	Values
m	Mass of the vehicle (kg)	1000
r_w	Wheel radius (m)	0.2736
g	Gravity acceleration (m/s^2)	9.81
ρ	Air density	1.2
A	Front area of vehicle (m^2)	2.36
α	Angle of driving surface (rad)	0
μ_{rr}	Rolling resistance coefficient	0.015
C_d	Aerodynamic drag coefficient	0.3
η_g	Gearbox efficiency	0.98
G	Gearbox ratio	8.59
C_0	Initial capacity (C)	720,000
R_{bi}	Internal resistance (Ω)	0.008
E_{b0}	Open-circuit voltage (V)	53.6
K_{ct}	Linear t dependency of the capacity C	6.084436×10^{-10}
t_{bu}	battery usage time (s)	0
T_{ref}	Reference temperature (°C)	20 °C
T_{curr}	Ambient temperature (°C)	20 °C
$alphaC$	Linear temperature coefficient of capacity (K^{-1})	0

$$F_t = F_{rr} + F_{ad} + F_{hc} + F_{la} + F_{wa} \tag{14}$$

$$F_{rr} = \mu_{rr} m g \tag{15}$$

$$F_{ad} = \frac{1}{2} \rho A C_d v^2 \tag{16}$$

$$F_{hc} = mg\sin(\alpha) \tag{17}$$

$$F_{la} = ma \tag{18}$$

$$F_{wa} = 0.05 \times F_{la} \tag{19}$$

$$T_t = F_t \times r_w \tag{20}$$

$$P_t = F_t \times v \tag{21}$$

$$\omega_w = \frac{v}{r_w} \tag{22}$$

$$S_w = \frac{30}{\pi} \times \omega_w \tag{23}$$

$$T_s = \frac{T_t}{\eta_g \times G}(P_t > 0) \tag{24}$$

$$T_{sr} = -\eta_g \times \frac{T_t}{G}(P_t < 0) \tag{25}$$

$$S_s = S_{sr} = G \times S_w \tag{26}$$

$$P_s = T_s \times S_s \times \frac{\pi}{30} \tag{27}$$

$$P_{sr} = T_{sr} \times S_s \times \frac{\pi}{30} \tag{28}$$

$$\text{Efficiency}(\eta) = \text{lookup2d}(\text{Torque}(\text{Nm}), \text{Speed}(\text{rpm})) \tag{29}$$

$$\eta_m = \text{lookup2d}(T_s, S_s) \tag{30}$$

$$\eta_r = \text{lookup2d}(T_{sr}, S_{sr}) \tag{31}$$

$$P_{bm} = \frac{P_s}{\eta_m} \tag{32}$$

$$P_{br} = \eta_r \times P_{sr} \tag{33}$$

$$P_{bc}(t) = \begin{cases} P_{bm}(t) + P_{aux}, & F_t(t) > 0 \\ P_{br}(t) + P_{aux}, & F_t(t) < 0 \end{cases} \tag{34}$$

$$I_B(t) = \frac{E_{B0}}{2 \times R_{Bi}} - \sqrt{\left(\frac{E_{B0}}{2 \times R_{Bi}}\right)^2 - \frac{P_{bc}}{R_{Bi}}} \tag{35}$$

$$Q(t) = \int_0^t I_B dt \tag{36}$$

$$C = C_0 * (1 - K_{Ct} * t_{bu}) * \left(1 + alphaC * \left(T_{curr} - T_{ref}\right)\right) \tag{37}$$

$$\mathrm{SOC}(t) = \frac{\mathrm{C} - Q(t)}{\mathrm{C}} \tag{38}$$

To verify the effectiveness of our speed prediction model for the final SOC prediction, we compared the proposed model results with the speed and SOC prediction obtained with the Seasonal Autoregressive Integrated Moving Average (SARIMA) model. The final SOC prediction results are shown in Table 7. As can be seen from the table, the RMSE of TA-LSTM drops 56.69%, 84.97%, 82.34%, and 91.62% compared to SARIMA in the AtoC, CtoD, DtoE, and EtoF sections, respectively.

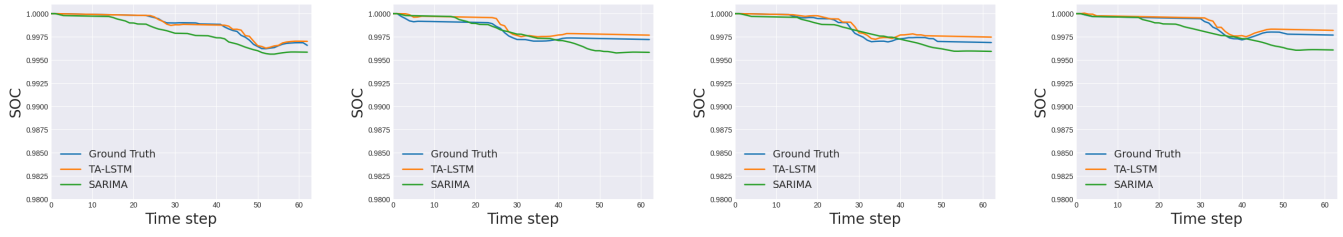
Table 7. RMSE ($\times 10^{-2}$) results of SOC prediction.

Models	AtoC	CtoD	DtoE	EtoF
SARIMA	0.0820	0.0998	0.1512	0.3938
TA-LSTM	0.0356	0.0150	0.0267	0.0330

The predicted SOC results made by TA-LSTM and SARIMA for the driving cycle of each road section are shown in Table 8 and Figures 12–15. From Table 8, we can see that TA-LSTM performed better than SARIMA in every driving cycle. In addition, the RMSE results of TA-LSTM for all driving cycles in the CtoD and DtoE sections are less than 0.0004. Although the RMSE for the AtoC and EtoF section is slightly worse than the first three sections, the RMSE results are less than 0.0006. The comparison between the real SOC and the predicted values shows that the predicted results of our model are very close to the real values. This indicates that our proposed method is able to perform SOC prediction properly.

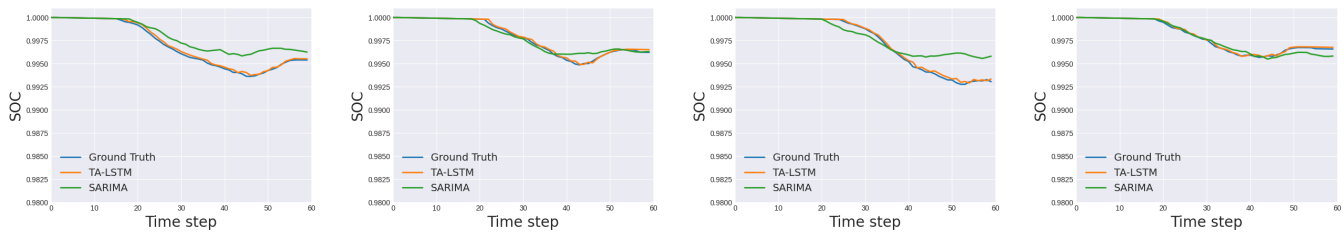
Table 8. RMSE ($\times 10^{-2}$) results of SOC prediction of each driving cycle.

Cycle	AtoC		CtoD		DtoE		EtoF	
	SARIMA	TA-LSTM	SARIMA	TA-LSTM	SARIMA	TA-LSTM	SARIMA	TA-LSTM
8	0.0892	0.0161	0.1288	0.0170	0.1236	0.0195	0.2505	0.0176
9	0.0792	0.0499	0.0411	0.0139	0.1179	0.0237	0.5170	0.0252
10	0.0604	0.0368	0.1429	0.0163	0.1408	0.0381	0.2615	0.0288
11	0.0950	0.0310	0.0339	0.0122	0.2060	0.0214	0.4711	0.0509



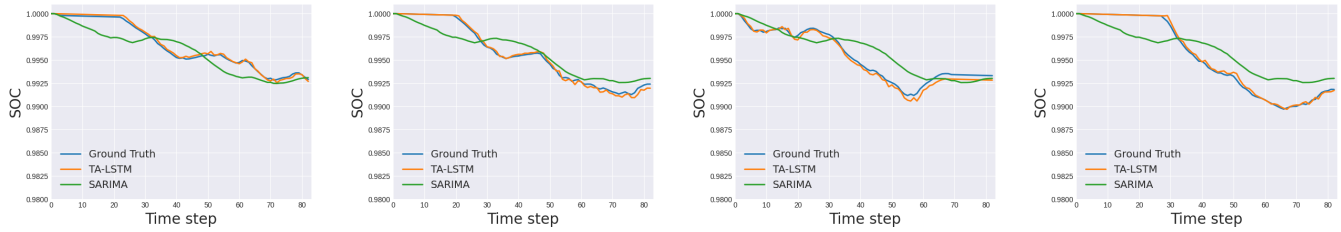
(**a**) (**b**) (**c**) (**d**)

Figure 12. SOC prediction of route A to C: (**a**) cycle 8; (**b**) cycle 9; (**c**) cycle 10; (**d**) cycle 11.



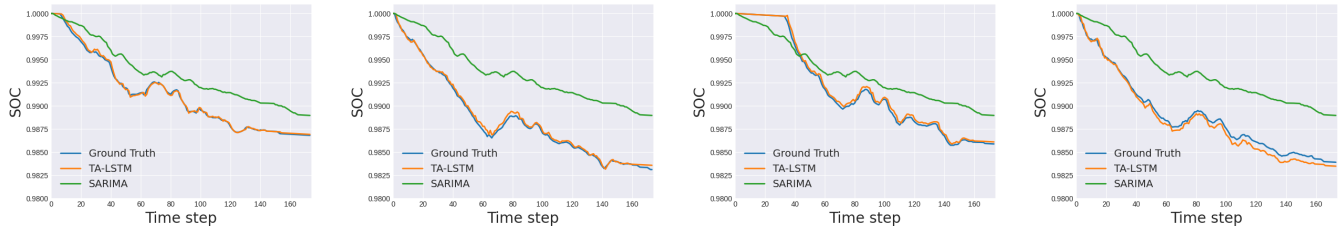
(**a**) (**b**) (**c**) (**d**)

Figure 13. SOC prediction of route C to D: (**a**) cycle 8; (**b**) cycle 9; (**c**) cycle 10; (**d**) cycle 11.



(**a**) (**b**) (**c**) (**d**)

Figure 14. SOC prediction of route D to E: (**a**) cycle 8; (**b**) cycle 9; (**c**) cycle 10; (**d**) cycle 11.



(**a**) (**b**) (**c**) (**d**)

Figure 15. SOC prediction of route E to F: (**a**) cycle 8; (**b**) cycle 9; (**c**) cycle 10; (**d**) cycle 11.

In addition, in order to simulate the battery model by reflecting more realistic situations, we checked the change in SOC by varying the ambient temperature and battery usage time. The real driving cycle, the parameters of the battery model, and the ambient temperature and battery operating time values for the simulation are shown in Table 9.

Table 9. Parameters for the battery model simulation.

Parameters	Description	Values
Driving cycle	Real driving cycle	Cycle 8
K_{ct}	Linear t dependency of the capacity C	6.084436×10^{-10}
t_{bu}	battery usage time (s)	0, 157,698,305 s (5 years), 315,396,610 s (10 years)
T_{ref}	Reference temperature (°C)	20 °C
T_{curr}	Ambient temperature (°C)	0, 20, 30 °C
$alphaC$	Linear temperature coefficient of capacity (K^{-1})	0.03 (0)

From the results in Figure 16, it can be confirmed that the battery SOC decreased faster because the battery capacity decreases as the temperature decreases.

From the results of Figure 17, it can be confirmed that the battery SOC decreased more quickly because the battery capacity decreases as the battery usage time increases.

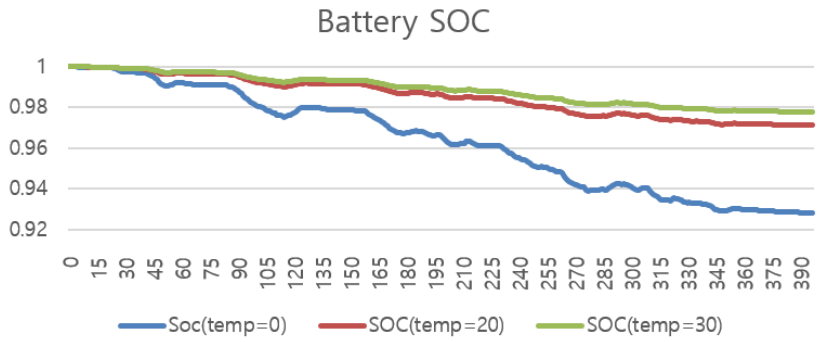


Figure 16. Temperature effect on the battery SOC.

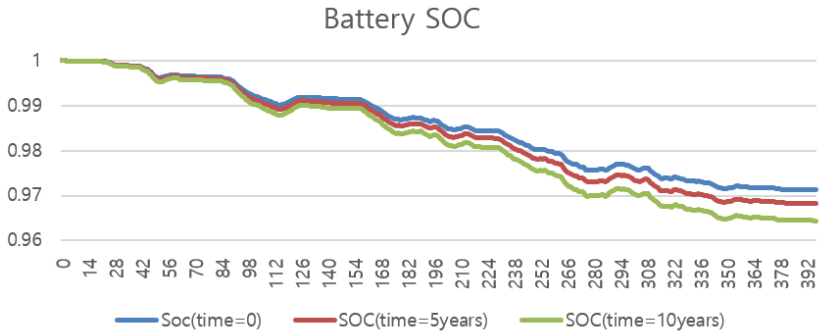


Figure 17. Battery aging effect on the battery SOC.

6. Conclusions

In our paper, we proposed a method for predicting the real driving-cycle-based SOC for EV batteries using deep-learning-based algorithms. For the RDC data, the authors used the data collected through the OBD II standard while driving from home to university in an actual vehicle. The proposed method first classifies detailed routes based on the section with traffic lights using GPS information, and finds the most similar patterns through DTW among the driving cycles of each section. Next, we predicted these sections by learning speed using TA-LSTM based on temporal attention, and predicted SOC based on this predicted speed. From the experimental results, it was confirmed that the case of using TA-LSTM could most accurately predict the speed, and that this could be very close to the actual values when used as a basis to predict the SOC. Therefore, if the proposed method is applied to the Real Driving Cycle, in which the same section is repeatedly driven, an accurate SOC prediction can be expected.

All the algorithms we propose can be implemented as an in-vehicle embedded system. It has been confirmed that DL models can be pre-trained and executed in an MCU. FMU simulation for the battery SOC can also be executed in an embedded system, such as an MCU or a sensor node [4,40]. If a high-capacity memory is required due to an increase in the demands of storing a massive amount of data or improving the calculation speed, a prediction can be performed in conjunction with a cloud data center or an edge server through the vehicular network, such as the Vehicle to Infrastructure (V2I) network.

The idea of the paper is to propose an algorithm that accurately predicts the SOC value of the next time step. However, predicting the SOC after multiple time steps may also be useful to the driver. Therefore, in a future work, the algorithm could be adapted so that it may predict, for example, the SOC after the completion of the entire driving cycle.

Author Contributions: Methodology, S.H., H.H. and D.K.; Project administration, S.H.; Software, S.H., H.H., D.K. and S.C.; Supervision, I.J.; Writing–original draft, S.H. and H.H. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported partly by the Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (No. 2020-0-00107, Development of the technology to automate the recommendations for big data analytic models that define data characteristics and problems), and partly by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2019R1I1A1A01058964).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

ASIC	Application Specific Integrated Circuits
BMS	Battery Management System
CAVs	Connected and Autonomous Vehicles
CSV	Comma Separated Value
CV	Cross Validation
DL	Deep Learning
DNNs	Deep Neural Networks
DTW	Dynamic Time Warping
EV	Electric Vehicle
FGPA	Field-programmable gate array
FMI	Functional Mock-up Interface
FMU	Functional Mock-up Units
FSD	Full Self-Driving
GPS	Global Positioning System
GPU	Graphics Processing Unit
GRU	Gated recurrent unit
MAPE	Mean Absolute Percentage Error
MCU	Microcontroller
ML	Machine Learning
MSE	Mean Squared Error
NEDC	New European Driving Cycle
OBD	On Board Diagnostics
OCV	Open circuit voltage
RDC	Real Driving Cycle
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
RRA	Redundancy reduction algorithm
SARIMA	Seasonal Autoregressive Integrated Moving Average
SDC	Standard Driving Cycle
SOC	State of Charge
TA-LSTM	Temporal Attention Long Short-Term Memory
TFLOPS	Tera floating point operations per second
V2I	Vehicle to Infrastructure
Variables	
α	Angle of the driving surface, rad
η_g	Gearbox efficiency
η_m	Efficiency of power consumed (motoring mode)
η_r	Efficiency of power generated (regenerative braking mode, W
α_c	Linear temperature coefficient of the capacity C, K^{-1}
μ_{rr}	Coefficient of rolling resistance
ρ	Density of the air
A	Frontal area of the vehicle, m^2
a	Aceleration of the vehicle, m/s^2
C	Battery capacity, Ah
C_d	Aerodynamic drag coefficient
E_{b0}	Open-circuit voltage of the battery, V
F_t	Traction force of the vehicle, N
F_{ad}	Aerodynamic drag, N

F_{hc}	Hill climbing force, N
F_{la}	Force required to give linear acceleration, N
F_{rr}	Rolling resistance force of the wheels, N
F_{wa}	Force required to give angular acceleration to the rotating motor, N
G	Gear ratio of differential
g	Gravity acceleration, m/s^2
I_B	Battery current
m	Vehicle mass, kg
P_s	Shaft power of electric machine (motoring mode), W
P_t	Traction power, W
P_{aux}	Power consumed by auxiliary loads, W
P_{bc}	Total power consumed in the EV, W
P_{bm}	Power consumed by electric machine (motoring mode), W
P_{br}	Power consumed by electric machine (regenerative braking mode), W
P_{sr}	Shaft power of electric machine (regenerative braking mode), W
Q	Total charge of the battery
r_w	Wheel radius, m
R_{Bi}	Internal resistance of the battery, Ω
S_s	Shaft angular velocity of electric machine (motoring mode), rpm
S_W	Angular velocity of the wheels, rpm
S_{sr}	Shaft angular velocity of electric machine (regenerative braking mode), rpm
T_s	Shaft torque of electric machine (motoring mode), Nm
T_t	Traction torque, Nm
T_{ref}	Reference temperature, K
T_{sr}	Shaft torque of electric machine (regenerative braking mode, Nm)
v	Velocity of the vehicle, m/s
w_W	Angular velocity of the wheels, rad/s

References

1. Cao, J.; Emadi, A. A New Battery/UltraCapacitor Hybrid Energy Storage System for Electric, Hybrid, and Plug-In Hybrid Electric Vehicles. *IEEE Trans. Power Electron.* **2012**, *27*, 122–132. [CrossRef]
2. Sivak, M.; Schoettle, B. Relative Costs of Driving Electric and Gasoline Vehicles in the Individual US States, University of Michigan. Report No. SWT-2018-1. 2018. Available online: http://websites.umich.edu/~umtriswt/PDF/SWT-2018-1.pdf (accessed on 19 November 2021).
3. Moghaddam, Z.; Ahmad, I.; Habibi, D.; Phung, Q.V. Smart Charging Strategy for Electric Vehicle Charging Stations. *IEEE Trans. Transp. Electrif.* **2018**, *4*, 76–88. [CrossRef]
4. STMicroelectronics. Artificial Intelligence (AI) Plugin for Automotive SPC5 MCUs. Available online: https://www.st.com/en/development-tools/spc5-studio-AI.html (accessed on 19 November 2021).
5. Li, S.; He, H.; Li, J.; Wang, H. Big data driven Deep Learning algorithm based Lithium-ion battery SoC estimation method: A hybrid mode of C-BMS and V-BMS. In Proceedings of the Applied Energy Symposium: MIT A+B, Boston, MA, USA, 22–24 December 2019.
6. Lu, S.; Shi, W. The Emergence of Vehicle Computing. *IEEE Internet Comput.* **2021**, *25*, 18–22. [CrossRef]
7. Liu, L.; Lu, S.; Zhong, R.; Wu, B.; Yao, Y.; Zhang, Q.; Shi, W. Computing Systems for Autonomous Driving: State of the Art and Challenges. *IEEE Internet Things J.* **2021**, *8*, 6469–6486. [CrossRef]
8. Findelair, A. Tomorrow's Car Silicon Brain, How Is It Made? Towards Data Science. Available online: https://towardsdatascience.com/tomorrows-car-silicon-brain-how-is-it-made-9090e1f06c9d (accessed on 23 March 2021).
9. Bhatt, A. Planning and application of Electric Vehicle with MATLAB®/Simulink®. In Proceedings of the 2016 IEEE International Conference on Power Electronics, Drives and Energy Systems (PEDES), Trivandrum, India, 14–17 December 2016; pp. 1–6.
10. Wei, M.; Ye, M.; Li, J.B.; Wang, Q.; Xu, X. State of Charge Estimation of Lithium-Ion Batteries Using LSTM and NARX Neural Networks. *IEEE Access* **2020**, *8*, 189236–189245. [CrossRef]
11. Amrhein, M.; Krein, P.T. Dynamic simulation for analysis of hybrid electric vehicle system and subsystem interactions, including power electronics. *IEEE Trans. Veh. Technol.* **2005**, *54*, 825–836. [CrossRef]
12. Lv, C.; Zhang, J.; Li, Y.; Yuan, Y. Mechanism analysis and evaluation methodology of regenerative braking contribution to energy efficiency improvement of electrified vehicles. *Energy Convers. Manag.* **2015**, *92*, 469–482. [CrossRef]
13. Fiori, C.; Ahn, K.; Rakha, H.A. Power-based electric vehicle energy consumption model: Model development and validation. *Appl. Energy* **2016**, *168*, 257–268. [CrossRef]
14. Genikomsakis, K.N.; Mitrentsis, G. A computationally efficient simulation model for estimating energy consumption of electric vehicles in the context of route planning applications. *Transp. Res. Part D Transp. Environ.* **2017**, *50*, 98–118. [CrossRef]

15. Liu, K.; Wang, J.; Yamamoto, T.; Morikawa, T. Exploring the interactive effects of ambient temperature and vehicle auxiliary loads on electric vehicle energy consumption. *Appl. Energy* **2018**, *227*, 324–331. [CrossRef]
16. Iora, P.; Tribioli, L. Effect of Ambient Temperature on Electric Vehicles' Energy Consumption and Range: Model Definition and Sensitivity Analysis Based on Nissan Leaf Data. *World Electr. Veh. J.* **2019**, *10*, 2. [CrossRef]
17. Luin, B.; Petelin, S.; Al-Mansour, F. Microsimulation of electric vehicle energy consumption. *Energy* **2019**, *174*, 24–32. [CrossRef]
18. Blochwitz, T.; Otter, M.; Akesson, J.; Arnold, M.; Clauß, C.; Elmqvist, H.; Friedrich, M.; Junghanns, A.; Mauß, J.; Neumerkel, D.; et al. Functional Mockup Interface 2.0: The Standard for Tool independent Exchange of Simulation Models. In Proceedings of the 9th International MODELICA Conference, Munich, Germany, 3–5 September 2012; pp. 173–184.
19. Hong, S.; Lim, D.; Joe, I.; Kim, W. F-DCS: FMI-Based Distributed CPS Simulation Framework with a Redundancy Reduction Algorithm. *Sensors* **2020**, *20*, 252. [CrossRef] [PubMed]
20. Bär, T.; Nienhüser, D.; Kohlhaas R.; Zöllner, J.M. Probabilistic driving style determination by means of a situation based analysis of the vehicle data. In Proceedings of the 14th International IEEE Conference on Intelligent Transportation Systems (ITSC), Washington, DC, USA, 5–7 October 2011; pp. 1698–1703. [CrossRef]
21. Ellison, A.B.; Greaves, S.P.; Bliemer, M.C.J. Driver behaviour profiles for road safety analysis. *Accid. Anal. Prev.* **2015**, *76*, 118–132. [CrossRef]
22. Wu, X.; He, X.; Yu, G.; Harmandayan, A.; Wang, Y. Energy-Optimal Speed Control for Electric Vehicles on Signalized Arterials. *IEEE Trans. Intell. Transp. Syst.* **2015**, *16*, 2786–2796. [CrossRef]
23. Varga, B.O.; Sagoian, A.; Mariasiu, F. Prediction of Electric Vehicle Range: A Comprehensive Review of Current Issues and Challenges. *Energies* **2019**, *12*, 946. [CrossRef]
24. How, D.N.T.; Hannan, M.A.; Hossain Lipu, M.S.; Ker, P.J. State of Charge Estimation for Lithium-Ion Batteries Using Model-Based and Data-Driven Methods: A Review. *IEEE Access* **2019**, *7*, 136116–136136. [CrossRef]
25. Xiong, R.; Cao, J.; Yu, Q.; He, H.; Sun, F. Critical Review on the Battery State of Charge Estimation Methods for Electric Vehicles. *IEEE Access* **2018**, *6*, 1832–1843. [CrossRef]
26. Zhang, R.; Xia, B.; Li, B.; Cao, L.; Lai, Y.; Zheng, W.; Wang, H.; Wang, W. State of the Art of Lithium-Ion Battery SOC Estimation for Electrical Vehicles. *Energies* **2018**, *11*, 1820. [CrossRef]
27. Yang, F.; Song, X.; Xu, F.; Tsui, K. State-of-Charge Estimation of Lithium-Ion Batteries via Long Short-Term Memory Network. *IEEE Access* **2019**, *7*, 53792–53799. [CrossRef]
28. Chemali, E.; Kollmeyer, P.J.; Preindl, M.; Emadi, A. State-of-charge estimation of Li-ion batteries using deep neural networks: A machine learning approach. *J. Power Sources* **2018**, *400*, 242–255. [CrossRef]
29. Babaeiyazdi, I.; Rezaei-Zare, A.; Shokrzadeh, S. State of charge prediction of EV Li-ion batteries using EIS: A machine learning approach. *Energy* **2021**, *223*, 120116. [CrossRef]
30. Xing, Y.; Ma, E.W.M.; Tsui, K.L.; Pecht, M. Battery Management Systems in Electric and Hybrid Vehicles. *Energies* **2011**, *4*, 1840–1857. [CrossRef]
31. Sakoe, H.; Chiba, S. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Trans. Acoust. Speech Signal Process.* **1978**, *26*, 43–49. [CrossRef]
32. Keogh, E.J.; Pazzani, M.J. Derivative Dynamic Time Warping. In Proceedings of the 2001 SIAM International Conference on Data Mining, Chicago, IL, USA, 5–7 April 2001.
33. Sundermeyer, M.; Schluter, R.; Ney, H. LSTM Neural Networks for Language Modeling. In Proceedings of the Thirteenth Annual Conference of the International Speech COMMUNICATION association, Portland, OR, USA, 9–13 September 2012.
34. Sutskever, I.; Martens, J.; Hinton, G. Generating Text with Recurrent Neural Networks. In Proceedings of the 28th International Conference on Machine Learning, Bellevue, WA, USA, 28 June–2 July 2011.
35. Fu, R.; Zhang, Z.; Li, L. Using LSTM and GRU neural network methods for traffic flow prediction. In Proceedings of the 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC), Wuhan, China, 11 November 2016.
36. Cui, S.; Joe, I. Collision prediction for a low power wide area network using deep learning methods. *J. Commun. Netw.* **2020**, *22*, 205–214. [CrossRef]
37. Qin, Y.; Song, D.; Cheng, H.; Cheng, W.; Jiang, G.; Cottrell, G.W. A dual-stage attention-based recurrent neural network for time series prediction. In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, Melbourne, Australia, 19–25 August 2017.
38. Fan, C.; Zhang, Y.; Pan, Y.; Li, X.; Zhang, C.; Yuan, R.; Wu, D.; Wang, W.; Pei, J.; Huang, H. Multi-horizon time series forecasting with temporal attention learning. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 25 July 2019.
39. Cui, S.; Joe, I. A Dynamic Spatial-Temporal Attention-Based GRU Model With Healthy Features for State-of-Health Estimation of Lithium-Ion Batteries. *IEEE Access* **2021**, *9*, 27374–27388. [CrossRef]
40. Bertsch, C.; Neudorfer, J.; Ahle, E.; Armugham, S.; Ramachandran, K.; Thuy, A. FMI for Physical Models on Automotive Embedded Targets. In Proceedings of the 11th International Modelica Conference, Versailles, France, 21–23 September 2015. [CrossRef]