Open Access

Journal of Energy and Power Technology

Original Research

Online State of Charge Prediction in Next Generation Vehicle Batteries Using Deep Recurrent Neural Networks and Continuous Model Size Control

Steven Hespeler [1, †], Donovan Fuqua [2, *]

1. College of Engineering, New Mexico State University, Las Cruces, NM, USA; E-Mail: schesp@nmsu.edu
2. College of Business, New Mexico State University, Las Cruces, NM, USA; E-Mail: dfuqua@nmsu.edu

† These authors contributed equally to this work.

* **Correspondence:** Donovan Fuqua; E-Mail: dfuqua@nmsu.edu

Academic Editor: Zhao Yang Dong

Special Issue: Modeling and Control of Fuel Cell Systems

Journal of Energy and Power Technology
2021, volume 3, issue 1
doi:10.21926/jept.2101xxx

Received: July 31, 2020
Accepted: December 29, 2020
Published:

Abstract

This investigation presents a data-driven Long-short Term Memory battery model for predicting State of Charge for lithium-ion batteries *LiFePO₄* for next-generation vehicle operations. Our modified algorithm builds and updates a model using multivariate inputs that include physical properties, voltage, current, and ambient temperature during operations. The primary research goal is to improve prediction performance on future values from multiple training examples using an online learning scheme. Initial results demonstrate excellent predictions that outperform results from literature and other neural network algorithms. Due to computing constraints in on-board vehicle systems, the authors develop online training with autonomous control of lag (window width). The control algorithm embeds in the model with rules that govern and adjust lag during training. This method ensures the

minimization of computational cost and prediction errors with the use of standard computing equipment during driving conditions.

Keywords
Long-short term memory; continuous model size control; battery management system

1. Introduction

With the increase of devices and applications that require sophisticated, next-generation batteries, management and control have become critical to maintaining the safety and reliability of these systems. Battery Management Systems (BMS) are used to monitor the overall health of batteries and are customizable to fit a variety of different battery types and various applications. Critical BMS parameters and functions established by Xing, Y, *et al.* include data acquisition, safety protection, ability to determine and predict state evaluation, ability to control battery charging and discharging, cell balancing, thermal management, delivery of battery status updates to a user interface, communication with all battery components, and prolonged battery life [1].

While battery operational conditions are different for different applications, this research focuses on next-generation automobiles [2]. In the case of Fuel Cell Vehicle (FCV) and Electric Vehicle (EV) operation, the BMS controls many parameters and functions - none being more important than state determination. Researchers consider state of charge (SOC) to be the most influential of the state determination parameters [3]. SOC provides information about the battery's current and remaining life, which is useful for protecting the battery from over-charging and over-discharging [3]. Furthermore, an accurate estimation of the battery state assures reliable and safe operating conditions for the user. In the case of battery operation, SOC is the equivalent of a fuel gauge and indicates to the user how much energy is available for usage. Therefore, accurate SOC prediction remains one of the main challenges in the successful operation of FCVs [2].

SOC estimation methods are typically complex, with scarce literature found to provide a detailed explanation of algorithmic approaches to the problem. Most of the methods have significant issues that become more apparent during the aging of the battery, temperature fluctuation, and change in discharge cycles [4]. Also, many of the methods produce inaccurate estimations of SOC because of the high sensitivity that lithium-ion batteries have to internal/external factors and complex electrochemical reactions. This problem results in a model attempting to evaluate complex calculations with high computation cost and negligence on the effects of time. A small number of Machine Learning (ML) algorithms have been utilized in an attempt to accurately predict SOC due to the ability to adapt and self-learn on a complex nonlinear dataset [5].

He et al. developed one such architecture using a Back Propagation Neural Network (BPNN) along with an Unscented Kalman Filter (UKF). This method estimates SOC during different driving conditions and directly tests recurrent neural network algorithms with algorithmic adjustments against BPNN and BPNN+UKF methods [6].

In a BMS, SOC is widely considered the most influential and essential parameters for battery-operated applications. SOC is defined as the percentage of residual capacity and the total capacity of the battery cell [3, 7-9]. When a battery discharges, the formula to calculate SOC is the ratio of

releasable capacity, $Q_{releasable}$, relative to the rated capacity, Q_{rated}, this percentage can be expressed in the following formula: $SOC = Q_{releasable} * Q_{rated} * 100$. It is desirable to maintain SOC percentage within certain limits, typically between 20 to 95 percent capacity [10].

While current literature demonstrates that there is no way to measure SOC of the chemical energy directly and precisely, Chang recommends four categories that distribute the many different methods of addressing the issue of SOC estimation [11]. SOC is estimated through direct measurement through physical battery properties such as terminal voltage, book-keeping evaluation, and indirect methods utilizing discharging current as one of the inputs. Adaptive systems automatically adjust the SOC when subjected under various discharging conditions. Finally, hybrid methods employ the advantageous parts of each SOC estimation method to provide estimates. The majority of estimation issues occur at full to partial SOC because the change in impedance is small resulting in a significant prediction error [12].

The voltage measurement method (or terminal voltage method) is based on evaluating terminal voltage drops due to internal impedances as the battery discharges [11]. Sato et al. proposed the following SOC estimation equation that literature accepts as the current standard: $SOC = \alpha V + \beta R + \gamma\sqrt{V} + \partial\sqrt{R} + C$ [13].

Typically, AC Impedance measurements are used as a method to indirectly determine battery capacity by inputting a sinusoidal, controlled current or voltage test signal of a single frequency or combination of signals with different frequencies [14]. Huet states that impedance is defined by: $|Z| = \frac{V_{max}}{I_{max}e^{j/\varphi}}$ [15]. Therefore, the electrochemical impedance of a battery is a frequency-dependent complex number characterized either by its real and imaginary parts. Through the mid to late seventies, bridges and lock-in amplifiers were utilized to measure impedance. Later on, impedance measurements were performed by frequency response analyzers based on harmonic analysis [15]. Early literature (before 1977) had conducted experimentation with the procedure involving the battery reaching equilibrium, which results in precise measurement. However, more recent research argues against this procedure and proposes the method of impedance measurement carried out while the battery is charging or discharging [16, 17]. However, this alteration creates additional errors and increases the measurement period.

Newer literature suggests a model-based approach to predict a state of charge, health, and life (SOC, SOH, SOL). Kozlowski used three models (neural networks, fuzzy logic, and auto-regressive moving average (ARMA) to identify electrochemical parameters in four battery types [18]. These models were offline, where there was a cutoff between training, test, and validation. Offline training, however, is less helpful for real-time SOC measurement. Hung et al. state that the motivation for online estimation is 1) the need for initial values or historical data, 2) the inability to perform real-time detection, and 3) the failure to determine the actual SOC from calculations in the event of fluctuations in SOH [19]. Bundy et al. present a multivariate method for predicting SOC using electrochemical data of a nickel-metal hydride (NiMH) battery [20]. This method generates predictions through partial least squares (PLS) regressions. These predictions then evaluate the electrochemical impedance spectra and estimate SOC.

For this research, artificial intelligence methods are utilized to predict SOC in an online training environment. Literature includes techniques involving backward propagation neural networks (BPNN), artificial neural networks (ANN), fuzzy logic, radial basis function neural networks (RBFNN), support vector machines (SVM), Kalman filters (KF) and particle filters (PF) [5, 21-23]. In the early

2000s, literature saw several ANN models utilized to estimate SOC in NiMH batteries [24]. Shen et al. created an input layer consisted of temperature, discharge, and regenerative capacity distribution through a total of six input neurons. The hidden layer consists of ten neurons, suggesting that increasing the number of neurons past ten has "no significant improvement in the estimation accuracy." Cai et al. estimated SOC of a high powered NiMH battery with an ANN [25]. In the study, researchers used a three-layered, feed-forward, back-propagation ANN. Since battery behavior is complex and nonlinear, input selection was accomplished by initially testing several different battery parameters (terminal voltage, discharge current, time-average voltage, etc.). Then the correlation coefficient was calculated based on variables and SOC. They rank results in order of the highest correlation to the least correlated. The researchers selected five variables as input neurons to the model: battery discharge current (I), accumulated ampere-hours (Ah), battery terminal voltage (V), time-average terminal voltage (TAV), and twice time-average terminal voltage (TTAV). It is worth noting that the linear correlation coefficients of the variables were very high for several of the selected variables (>0.942). Yanqing presented a novel approach for online SOC determination by using n neural network-based model and neuro-controller. First, Yanqing erects a nonlinear dynamic system cell model, that can be represented by discrete state-space form calculations [5]. Linda et al. published research on predicting SOC with a feed-forward neural network model using voltage, current, and ambient temperature [26].

Support vector machine (SVM) is a minimization-maximization algorithm that produces hyperplanes within data. Researchers have used SVM to estimate SOC of a variety of batteries [27-31]. The fuzzy logic (FL) method is a rule-based option for nonlinear data. FL models include four parts: a relationship in rule-based input-output, the membership function for both input and output, reasoning, and defuzzification of outputs. Singh et al. use an FL based SOC meter developed for use in a Lithium-ion battery in a portable defibrillator [32]. In a study conducted by Hu et al., fuzzy adaptive federated filtering is utilized for SOC estimations for series-connected battery packs to combat inconsistencies in battery cell state [50]. The investigation found that online and offline parameters experience less than 0.4% and 1%, respectively.

Current literature based on electrochemical battery modeling indicates a keen interest in SOC estimation methods with an attempt to create an efficient BMS [33-38]. Much of the research conducted focuses on external battery influences that do not consider internal dynamics nor energy losses of the battery [39-41]. Other studies estimate SOC without online parameters, which become inaccurate as the battery ages [42]. Some research has investigated internal dynamics and online SOC determination of batteries but does not take into account temperature effects on battery performance [43]. Data-driven methods present a new possibility of explaining internal battery dynamics from an applied systems approach. An intelligent/online BMS that considers internal dynamics and temperature effects work to ensure the Lithium-ion batteries operate efficiently for vehicle operation throughout the life of the battery. Unlike the approaches used in literature, this investigation treats the battery performance data as a time series to ensure the time-dependent relationships of the parameters are accurately monitored.

The issue of computational time during SOC prediction was addressed in the work completed by Kim in [51], which demonstrates a battery model capable of limiting computational time while introducing a robust slider mode observer to compensate for model error. The model restricts SOC error to less than 3% in most cases. In another study, Skrylnyk et al. investigate slider mode along

with fuzzy logic modeling for SOC estimation during autonomous solar application [52]. In the study, authors report excellent estimation capabilities with the tradeoff of increased training time.

On an application level, methods like ANN, FL, and KF have their own set of issues. ANN algorithms require a substantial amount of data before they can increase accuracy, which prevents these models from being rapid. FL method requires definitions of its membership functions, making this method undesirable for large models. KF is suited more for linear systems, of which battery models are not. Other versions of the KF like EKF and UKF methods can be challenging to tune and will provide inaccurate estimations if nonlinearities in the battery model are severe [6].

Based on the literature review, the following knowledge gaps are directly addressed in this research:

1) There is limited research on state of charge prediction using state of the art RNN architectures, notably long short-term memory neural networks.

2) Feature selection using statistical learning can determine variable importance in the state of charge prediction that has previously been assumed or reported through observational evidence.

3) There is limited research on comparison of multiple methods for SOC prediction.

4) On-boarding an online algorithm will require controlling data size that can be accomplished through continuously controlling lag as a hyperparameter during learning.

In this study, data used was collected by the CALCE lab at the University of Maryland on $LiFePO_4$ batteries under dynamical stress testing (DST), US06 highway driving schedule, and the federal driving schedule. We thank CALCE for the use of their data and for their willingness to answer questions about testing procedures [42].

2. Materials and Methods

Time computations are performed on a personal computer platform to mimic the capabilities of a next-generation vehicle platform. This setup used an Intel Core i3 7100U running up to 2.4 GHz/s and 8 GB of RAM. For hyperparameter search and model development, the New Mexico State University Discovery Computing Cluster was utilized. All computations were executed on Python 3 with the Tensor Flow 2.1.0 package.

2.1 Feature Selection

An essential aspect of this investigation was to identify the most influential physical battery features to input into the predictive battery model. This dataset is highly dynamical with high dimensionality; therefore, it was essential to decide which features were necessary for high prediction accuracy and which weren't. Therefore, non-informative variables were removed from the dataset through analysis using the Random Forest technique.

2.2.1 Random Forest Definition

Random Forest (RF) is a method focused on reducing variance by utilizing the bagging (bootstrap aggregation) technique to comprise a mean of noisy but unbiased models [44]. RF works through a tree-growing process in which random selection of input variables bootstraps on a dataset. The idea is to de-correlate the trees without increasing variance [45]. RF introduces randomness to the tree-growing process. RF selects m variables by randomly pulling from p variables so that $m = \sqrt{p}$. The

driving force behind this process is the regression predictor, mathematically represented as (for regression):

$$\hat{f}_{rf}^{B}(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x) \quad (1)$$

Where *B* is the total number of trees that we are predicting on a new point *x*.

2.2.2 Feature Importance

Random Forests techniques are capable of producing feature importance charts, as seen in Section 3.1. In this method, the importance measure determines the split point when growing the tree. Therefore, highly correlated features show little to no importance measure. The process uses Out of Bag (OOB) samples at the bth tree to pass down the tree to record prediction accuracy [44]. Then, values at the *j*th variable are arbitrarily permuted from the OOB samples and again accuracy is computed and recorded. The computed accuracy is then averaged across all trees and used a metric for determining importance at the *j*th feature with the Random Forest.

2.2 LSTM

2.2.1 Definition

Long-Short Term Memory (LSTM) is a specific type of recurrent neural network (RNN) architecture developed to solve the vanishing gradient problem [46, 47]. The LSTM does not experience long term dependency issues seen in other RNN architectures. In practice, the LSTM has demonstrated a superior ability to learn long-range dependencies as compared to simplified RNNs [48]. As seen in Figure 1, this model introduces a memory cell. RNNs utilize long-term memory in the form of weights, which change slowly during training and short-term memory in the form of temporary activations which pass from one node to another [48]. The memory cell in a LSTM network has intermediate storage in the form of gates within each hidden layer. Figure 1 depicts an LSTM architecture that contains four parts within the memory block: an input gate (i), a forget gate (f), an output gate (o), and cell state (C_{t-1}). The forget gate is responsible for deciding which information is retained or discarded from the cell state. The input gate determines which values will be updated to a vector of new candidate values (C_t). Finally, the output gate decides what information will be passed to the cell state in the next time step.

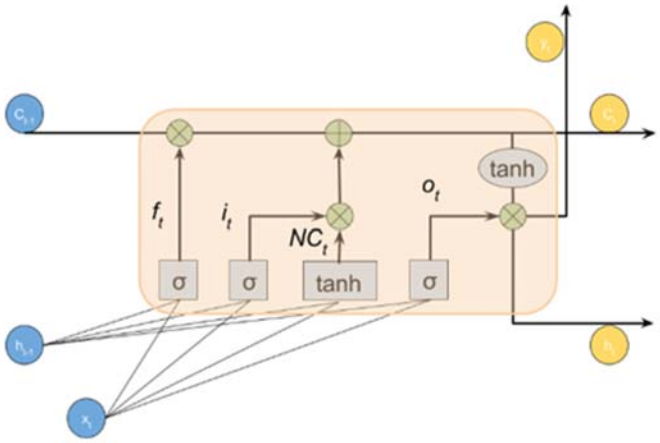

Figure 1 LSTM memory cell.

LSTM connections are represented through the following equations:

$$f_t = \sigma\left(W_{fh}h_{t-1} + W_{fx}x_t + b_f\right) \tag{2a}$$

$$i_t = \sigma(W_{ih}h_{t-1} + W_{ix}x_t + b_i) \tag{2b}$$

$$C_t = f_t \cdot c_{t-1} + i_t \cdot NC_t \tag{2c}$$

$$NC_t = \tanh\left(W_{NC_th}h_{t-1} + W_{NC_tx}x_t + b_{NC_t}\right) \tag{2d}$$

$$o_t = \sigma(W_{oh}h_{t-1} + W_{ox}x_t + b_o) \tag{2e}$$

$$h_t = o_t \cdot \tanh(c_t) \tag{2f}$$

In these equations, σ is a random sigmoid activation, (i_t, f_t, h_t, c_t, and o_t) are vectors from the input, forget gate, hidden gate, cell gate, and output, b_t is the bias vector, and W represents weight vectors at various portions of the LSTM.

2.3 Data

2.3.1 Collection of Data

CALCE collected and reported all data used in this experiment [42]. The battery tested was lithium-ion ($LiFePO_4$) and was subjected to three battery testing load profiles; dynamical stress testing (DST), US06 highway driving schedule, and federal urban driving schedule (FUDS). DST data is a basic loading profile built for battery testing. US06 and FUDS are complex, highly nonlinear data that simulates real life driving cycles. Figure 2 depicts the load profiles of the DST (top), US06 (middle), and FUDS (bottom) datasets.

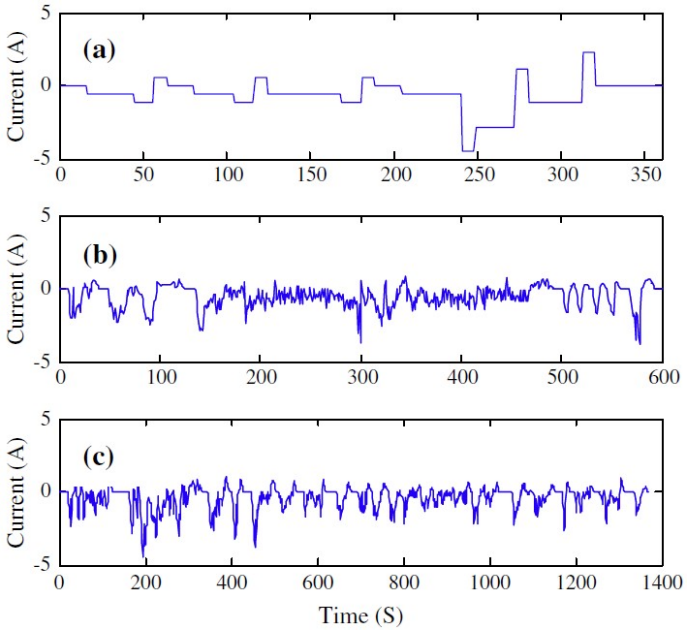

Figure 2 Loading profiles(current) of DST (top), US06 (middle), and FUDS (bottom).

2.3.1.1 Load Profiles. The DST test is a testing procedure established by the US Department of Energy Vehicle Technologies Program used throughout literature to evaluate the performance of EVs [49]. The test validates models and algorithmic accuracy [25]. During the trial, a battery experiences different DST cycles that alter SOC from 90% to 20%.

US06 is a high acceleration aggressive driving schedule identified as the "Supplemental FTP" driving schedule EPAUS06. Figure 3 (a) shows the aggressive driving schedule. Figure 3 (b) shows the noticeably more complex FUDS. Researchers subjected all the loading profiles to varying ambient temperatures; 0, 10, 20, 30, 40, and 50 $Celsius$. This investigation focused on the 0 $Celsius$ datasets to test our algorithm.

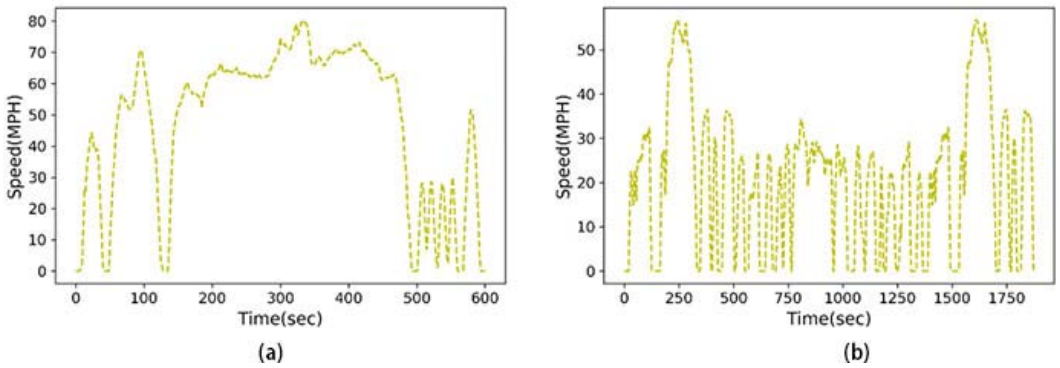

Figure 3 (a) US06 highway driving schedule (b) FUDS highway driving schedule.

2.3.2 Data Formatting and Reshape

Raw data was imported to the predictive battery model as individual load profiles in the comma-separated value (CSV) format. Then, each dataset is cleaned and prepared. The small number of rows containing empty (NaN) values were eliminated and included the current, voltage,

temperature, and SOC features. Once the cleaning was complete, the time series data was normalized to a range of (0, 1).

Next, the data was reshaped and prepared for training on a time-series cross-validation split. Experiments were run for all the data according to the following combinations of lag capacity, horizon, and load profile, with 36 experiments completed in total.

In our construct, a walk-forward validation model was created where the train and test index splits into feature train and test, along with predictor train and test tuples. Since this is a time-series and the data is auto-correlated, our function trains on the feature and predictor training set then uses the *X*th test sample to generate a $\hat{y}_{+1}$ prediction. Walk-forward validation iterates through the train and test sets. Since this is a time-series dataset, we predict a value *x* at time *T* from T_{-H}, where *H* is the pre-set horizon value.

Using the TensorFlow package in Python, the model is formatted to a sequential model with LSTM of 50 input layers (determined through hyperparameter grid search along with batch size and epochs). The raw data (after cleaning and data prep) has 6000 to 8000-time steps (n) with four features (p). Raw data was reshaped into a 3D tensor with the shape of [batch, time steps, feature]. A *reshape()* function is executed that accepts a tuple argument. The data now takes the form of:

$$[X\ train, (X\ train[0], X\ train[1]), 1] \tag{3}$$

Where *X train [0]* has the shape *[batch, time steps, 1]* and *X train [1]* has the shape *[batch, time steps, +1, 1]*.

Each experiment runs for 1,000 iterations. Once the input is reshaped, the model is fit to an epoch cycle of 100 and a batch size of 50. The model trains by slicing inputs into batch sizes and iterating over the specified number of epoch cycles. At the end of each epoch, the model iterates over the validation dataset and computes the validation loss.

2.4 LSTM-CLC

2.4.1 Network Architecture

Figure 4 depicts how our algorithm predicts SOC on the battery model. The model is compiled with a loss calculation of MAE, using the Adam optimizer. Using a walk-forward validation technique, the model predicts a pre-set horizon value (1, 3, or 5), indicating 10, 30, and 50 seconds in the future as the time-series is in a 10-second format.

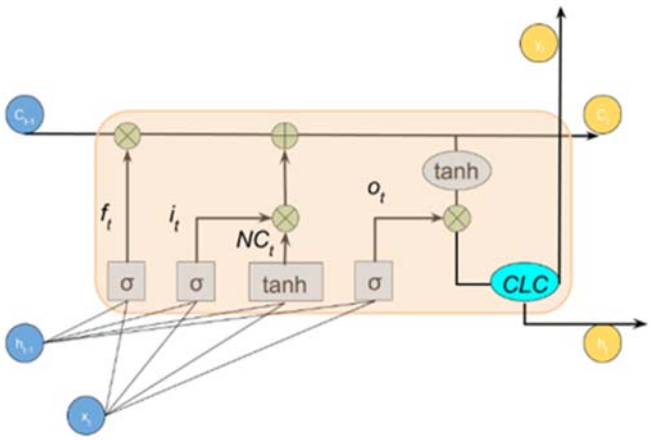

Figure 4 Illustration of LSTM-CLC memory block.

Here, the forget, input, and output gates all function as a vanilla LSTM. However, once the LSTM initiates the backward pass, the lag capacity adjusts (or remains the same from the previous time step), and the next forward pass begins. This next pass uses the lag capacity to train on the allowable data size set by the lag capacity rules. The algorithm runs until all iterations are complete, in this case, 1,000 iterations.

During each grid search, testing parameters are recorded. RMSE, computation time, loss, validation loss, predicted value, and expected value are all appended to respective variables. Once the experiment has completed, the variables are saved and stored in a CSV file for further analysis, discussed in Section 3 and 4.

Figure 5 depicts how the lag capacity could expand during any experiment. However, restriction of the data size up to the lag capacity during experimentation mimics onboard computational capacities. An important note is that during each iteration, an observation has only one time to be in the test set. This type of cross validation ensures low bias in each model.

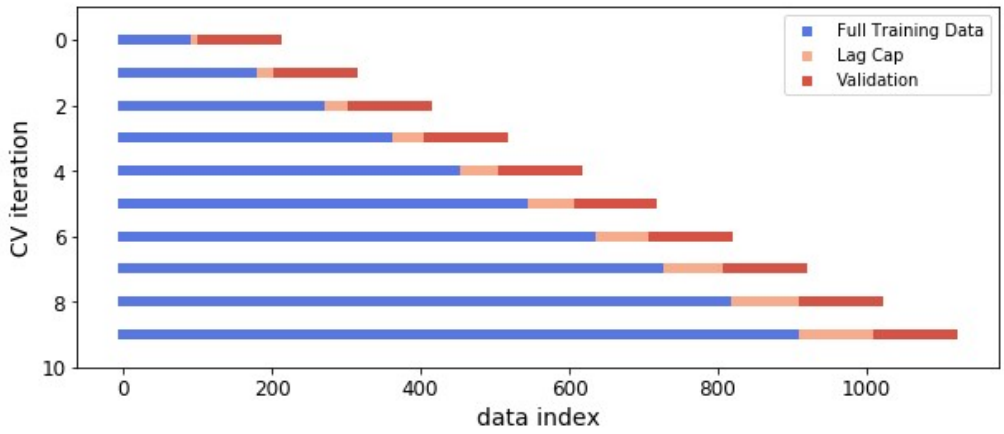

Figure 5 Visualization of train set modification with walk forward validation.

2.4.2 Pseudo Algorithm

The algorithm is explained in pseudo code below.

Algorithm 1: Controlling Lag- Optimal Model Size

Input : $M = \{(x_i, y_i)\}_{i=1}^{n}$
Output: (W, b, l)
Initialize Weights, bias, and lag;
Forward pass and loss calculated;
while *Train Index, Test Index in TSCV* **do**
  Backward pass, Adam optimization, update $(\frac{\partial W}{\partial t}, \frac{\partial b}{\partial t})$;
  if $e > 3\sigma$ **then**
    $lag = lag + 1$;
  else
    $lag = lag$;
  end
  if $e > 2\sigma$ **then**
    $lag = lag + 1$;
  else
    $lag = lag$;
  end
  if $\epsilon[t] < [t-1] < [t-2]$ **then**
    $lag = lag - 1$;
  else
    $lag = lag$;
  end
  if $lag > lag\ cap$ **then**
    lag=lag cap;
  else
    $lag = lag$;
  end
end
RMSE calculation and comparison of Y^{pred} to Y^{act}

Figure 6 Pseudocode for LSTM-CLC algorithm.

2.4.3 Rules Governing Lag Control

2.4.3.1 Rule 1- Data Capacity. Walk-forward validation is utilized for sequential predictive models to split data into training and test sets. Since the data in sequential models are time dependant, the order of arrival is important. Features must remain in order as they arrived in the dataset. Applying the walk-forward method to an application like HEV proposes difficulty due to ever-increasing memory tax as the model learns. Although with our other rules, lag is capped to a maximum value in the model, which allows the training index to reach back only as far as the lag cap will allow.

This cap minimizes the maximum allowed size of the training index at time t within the planning horizon *[0, T].* The control algorithm implements the cap by disallowing the training set to be larger than that of the lag cap. If the size of the training set extends beyond the lag cap, the model is adjusted to include the previous training set within the lag cap only. Four different lag caps are run in the experiments: 10, 20, 30, and 40. The lag caps are somewhat arbitrary and can be increased when there is additional on-board computational power.

2.4.3.2 Rule 2- Sampling Distribution. The lag control algorithm monitors and adjusts to diminishing standard error values. If three consecutive error values are decreasing, that is if $\epsilon[0] < \epsilon[-1]\ and\ \epsilon[-1] < \epsilon[-2]$, the lag value will pull back from lag to *lag -1*. This cap continuously monitors and adjusts the window width to ensure lag controls sampling distribution (standard error).

2.4.3.3 Rule 3 Outlier Detection. There are three specific cases that the control algorithm adjusts lag based on outlier detection. If the standard error of the residuals is outside two standard deviations from the mean on a particular iteration, lag adjusts from *lag* to *lag +1*.

3. Results

3.1 Feature Analysis

Figure 7 shows the relative variable importance in each of the driving loading profiles. Although there are differences in the importance levels, we see that current is the most significant predictive feature for SOC prediction, followed by voltage, and then ambient temperature. Test time (degradation of the battery), discharge capacity, charge energy, and charge capacity have a measurable but less significant effect on SOC.

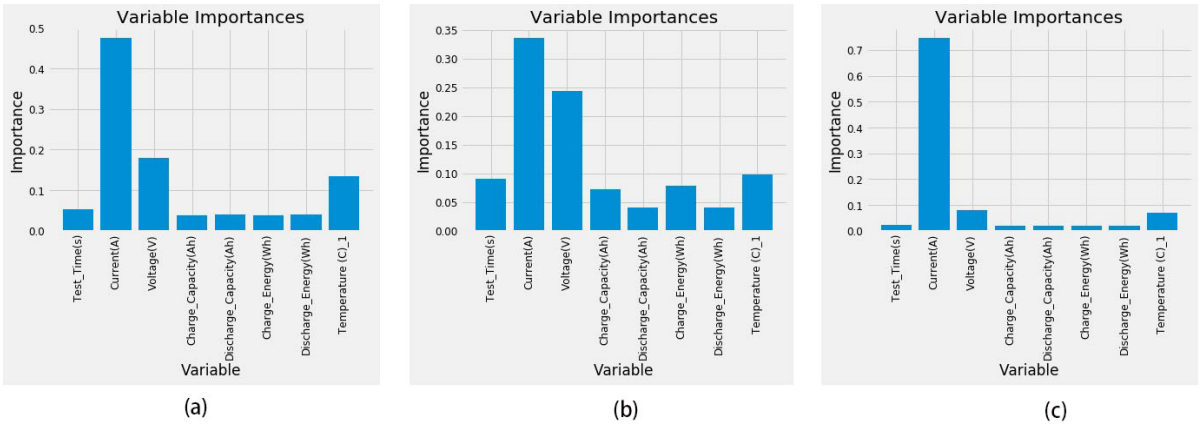

Figure 7 Feature importance comparison (random forest) for a) DST, b) FUDS, c) US06.

We can see that importance for variable influence changes based on driving cycle, indicating that variable importance is not collinear with the increase of dynamical data.

3.2 Residual Analysis

During online training, residuals are plotted for the loading profiles in Figure 8. In Figure 9, we perform QQ plots to check the normality of the residuals, given the assumption that our lag control model identifies an outlier that is three standard deviations from the mean. Figures 8 and 9 show that FUDS has higher residuals than DST or US06, but has a more normal distribution. While there is a lack of residual normality in DST, it is also the simplest of the loading profiles and has the most autocorrelation. There is confidence that while the outlier rule might not help with the DST model, it also does not impede learning.

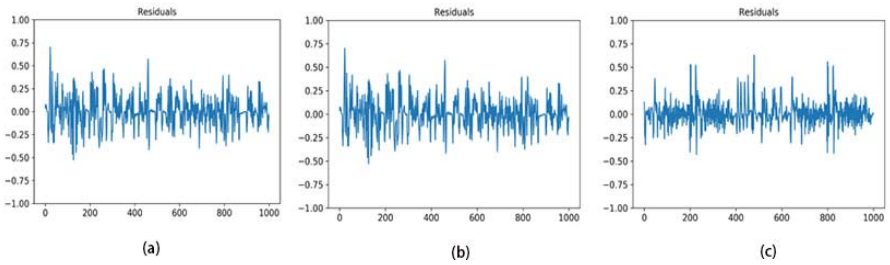

Figure 8 Residual time series plots for validation results for a) DST, b) FUDS, and c) US06.

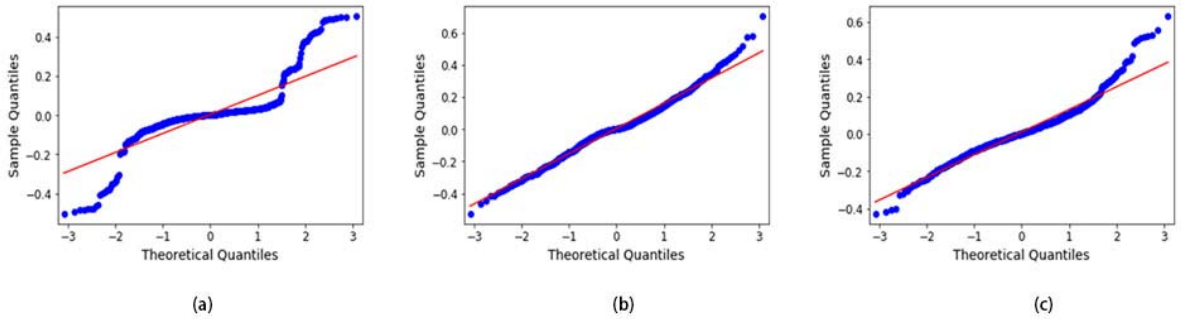

Figure 9 QQ Plots of residuals for a) DST, b) FUDS, c) US06.

Residual analysis demonstrates a lack of cyclic nature and patterns, which indicates that the errors are random and are a result from the model.

3.2 Model Performance

In Table 6, the algorithm is compared with results from literature and computations made on CPU. The first two rows include the NN (neural network, an artificial neural network) and the NN+UKF (neural network with unscented Kalman Filter), results were taken from He (2014) [42]. These two rows do not include results from the DST load profile because the authors from the study trained on that dataset. Results from the vanilla LSTM demonstrate the power of this algorithm on time-series data which set the desire to perform online LSTM and on-board the algorithm to HEV application. The models were trained and tested on each load profile which is likely the reason that the model performs at such a high level. Literature involving SOC prediction using SVM does not perform tests on the DST, US06, and FUDS datasets, therefore tests were conducted with the three most common kernel types using the skLearn toolset in Python and included in the comparison. Finally, results were averaged over ten runs and employ the CLC modification to show that the LSTM+CLC has the best predictive performance.

Table 1 Comparison of different predictive models.

	DST	US06	FUDS
NN (42)	N/A	4.100	4.200
NN+UKF (42)	N/A	2.400	2.200

Vanilla LSTM	0.372	0.464	0.493
SVM (RBF) +CLC	0.126	0.155	0.210
SVM (Poly) + CLC	0.119	0.124	0.201
SVM (Linear) +CLC	0.099	0.128	0.193
LSTM + CLC	**0.076**	**0.102**	**0.166**

In Table 7, average RMSE values are displayed (over ten runs) with a ten-second (H=1), thirty-second (H=3), and fifty-second (H=5) future prediction. It was anticipated that predictions would degrade given longer horizon time however, the results demonstrate robust future prediction capabilities. Longer predictive horizons benefit from a larger lag cap. Table 8 shows the standard deviations in each of the average runs. This table illustrates relatively low variability between runs and demonstrates that the CLC is effective at controlling outliers.

Table 2 Average (Mean) of RMSE results.

	H=1			H=3			H=5		
	DST	US06	FUDS	DST	US06	FUDS	DST	US06	FUDS
Lag Cap 10	0.064	0.091	0.140	0.078	0.111	0.202	0.084	0.120	0.212
Lag Cap 20	0.070	0.088	0.132	0.076	0.104	0.180	0.083	0.113	0.184
Lag Cap 30	0.069	0.087	0.128	0.076	0.104	0.172	0.082	0.112	0.180
Lag Cap 40	0.070	0.086	0.125	0.078	0.102	0.170	0.084	0.112	0.178

Table 3 Standard deviation of RMSE results.

	H=1			H=3			H=5		
	DST	US06	FUDS	DST	US06	FUDS	DST	US06	FUDS
Lag Cap 10	0.011	0.008	0.024	0.012	0.013	0.029	0.012	0.013	0.027
Lag Cap 20	0.011	0.006	0.020	0.010	0.014	0.023	0.011	0.012	0.022
Lag Cap 30	0.012	0.006	0.018	0.011	0.011	0.022	0.011	0.011	0.024
Lag Cap 40	0.012	0.006	0.018	0.012	0.013	0.023	0.012	0.011	0.024

Table 9 illustrates ten-second (H=1) predictions for each of the loading profiles. A significant result is that with the simulated on-board system (Core i3 processor), average computational time was 2.05 seconds per online prediction. Therefore, despite using an advanced deep learning technique, this BMS would receive a forecast for an event 8 seconds in the future with excellent prediction performance.

Table 4 Results of ten-second horizon.

Lag Cap	Load Profile	Ave SDE	Max SDE	Ave Time	Max Time	Total Time
10	DST	0.064	0.080	2.065	3.577	2061.141
	US06	0.091	0.127	2.053	3.351	2048.867
	FUDS	0.140	0.208	2.100	3.150	2095.649
20	DST	0.069	0.087	2.051	2.755	2051.213
	US06	0.087	0.123	2.035	3.053	2034.995
	FUDS	0.131	0.169	2.058	3.225	2057.557
30	DST	0.069	0.087	2.108	3.948	2108.107
	US06	0.086	0.112	2.128	5.079	2127.751
	FUDS	0.127	0.152	2.164	4.512	2164.085
40	DST	0.070	0.088	2.031	2.440	2030.791
	US06	0.085	0.123	2.039	2.851	2039.138
	FUDS	0.124	0.155	2.037	2.854	2037.492

In Tables 10 and 11, the predictive horizon is pushed to thirty seconds (H=3) and fifty seconds (H=5) and have roughly the same computational time per iteration. It can be stated that horizon window has minimal effect on computational time in the model.

Table 5 Results of thirty-second horizon.

Lag Cap	Load Profile	Ave SDE	Max SDE	Ave Time	Max Time	Total Time
10	DST	0.096	0.129	2.037	3.125	2037.435
	US06	0.126	0.171	2.145	4.049	2144.721
	FUDS	0.200	0.268	2.171	5.119	2171.391
20	DST	0.098	0.129	2.062	3.610	2062.110
	US06	0.121	0.179	2.067	2.649	2066.734
	FUDS	0.179	0.223	2.071	2.939	2071.450
30	DST	0.097	0.128	2.173	3.643	2173.444
	US06	0.119	0.172	2.145	3.370	2144.582
	FUDS	0.171	0.219	2.110	3.347	2110.102
40	DST	0.095	0.128	2.152	2.591	2152.111
	US06	0.119	0.177	2.154	4.550	2153.868
	FUDS	0.169	0.220	2.145	4.156	2144.919

Table 6 Results of fifty-second horizon.

Lag Cap	Load Profile	Ave SDE	Max SDE	Ave Time	Max Time	Total Time
10	DST	0.117	0.145	2.230	4.335	2229.806
	US06	0.089	0.131	2.034	3.051	2030.082
	FUDS	0.211	0.287	2.060	2.876	2060.138
20	DST	0.115	0.143	2.131	3.769	2131.025
	US06	0.127	0.146	2.092	3.221	2092.484
	FUDS	0.183	0.251	2.154	4.162	2154.042
30	DST	0.110	0.144	2.171	5.362	2171.285
	US06	0.125	0.156	2.166	4.142	2165.733
	FUDS	0.180	0.251	2.203	3.992	2202.841
40	DST	0.105	0.144	2.168	3.138	2168.166
	US06	0.125	0.157	2.169	3.957	2168.595
	FUDS	0.177	0.251	2.169	3.164	2168.849

Figure 10 displays RMSE plots through 1000 iterations for all experiments including DST, US06, and FUDS load profiles. Datasets with less variability (DST) train similarly, while there is more training variability in complex driving profiles (FUDS). These plots demonstrate that training takes roughly 200-300 iterations before the model reaches a constant prediction error. Training error increases as the datasets become more dynamical, which is expected.

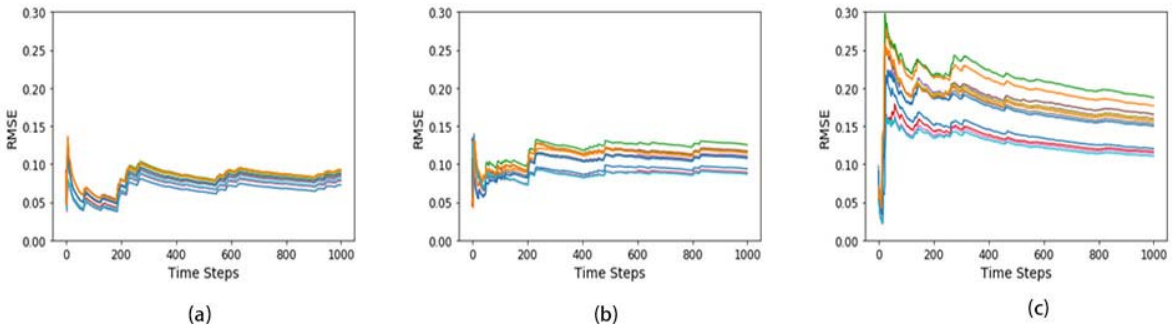

Figure 10 RMSE Values for runs of a) DST, b) US06, and c) FUDS.

In Figure 11, the effect of controlling training data size is shown and how valuable the CLC is for a possible on-board application. Figure 11a shows the necessity of adding a lag control as training exponentially grows and becomes computationally too complex after 500 iterations. This additional time and computational complexity does not equate to better training or a lower RMSE as seen in Figure 11b. It can be seen that model performance remains excellent with minimal computation time in 11b.

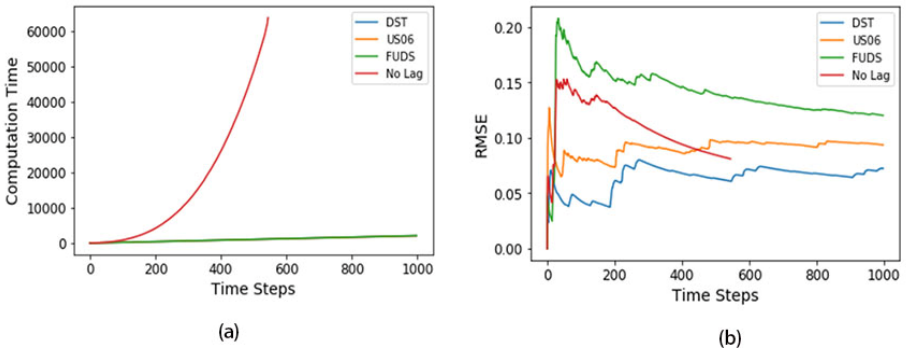

Figure 11 a) Comparison of computation of load profiles with a lag cap and one profile with a lag cap and b) Standard error plots of the same load profiles.

In Figure 12, Figure 13, Figure 14, Figure 15, Figure 16 and Figure 17, actual values versus predictive values are plotted with varying values of lag capacity limits and validation horizon. As stated in Section 2, it was ensured that training and validation sets were separate during online training. Therefore, an apparent decrease of predictive power can be seen with broader training horizons and a smaller increase of power with larger lag capacity limits.

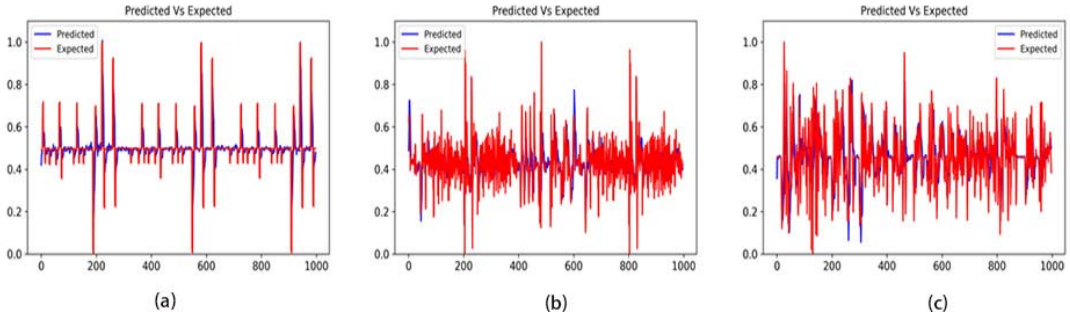

Figure 12 Expected vs. predicted plots with horizon set at one and lag cap set to 10 for a) DST, b) US06, c) FUDS.

False positive predictions (Type II error) increases slightly as the prediction horizon increases from 10 to 30 seconds. Although an error of this type would be hazardous during operation, we don't see false positive predictions to be a significant issue with this battery model.

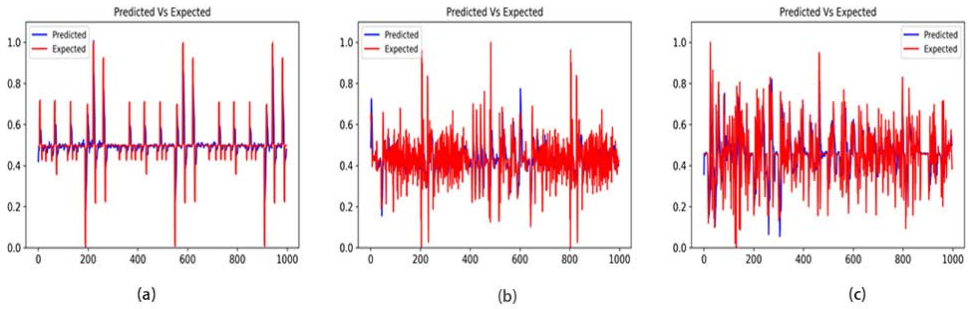

Figure 13 Expected vs. predicted plots with horizon set at three and lag cap set to 10 for a) DST, b) US06, c) FUDS.

Finally, extension of the prediction window out to 50 seconds demonstrates excellent prediction between actual and predicted plots. Compared to the 30 second horizon plot there is a slight increase in type II error however, even at the extended prediction window there doesn't appear to be enough error to cause the model to fail to meet expectations.

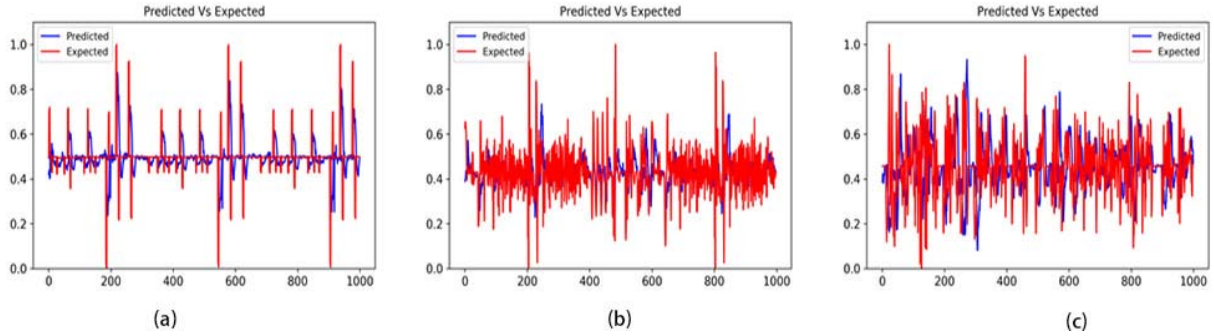

Figure 14 Expected vs. predicted plots with horizon set at five and lag cap set to 10 for a) DST, b) US06, c) FUDS.

Plots 15 through 17 show the prediction performance while extending the training window to 40. When comparing these plots to corresponding plots above (same horizon windows), it is determined the expanding the training size from 10 to 40 doesn't have a drastic effect on performance.

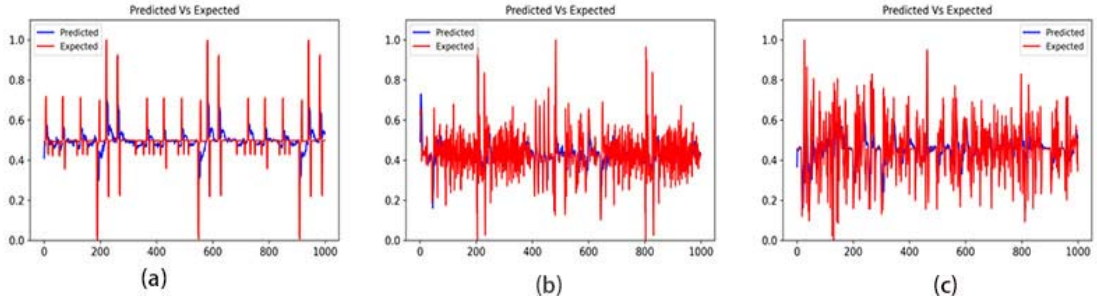

Figure 15 Expected vs. predicted plots with horizon set at one and lag cap set to 40 for a) DST, b) US06, c) FUDS.

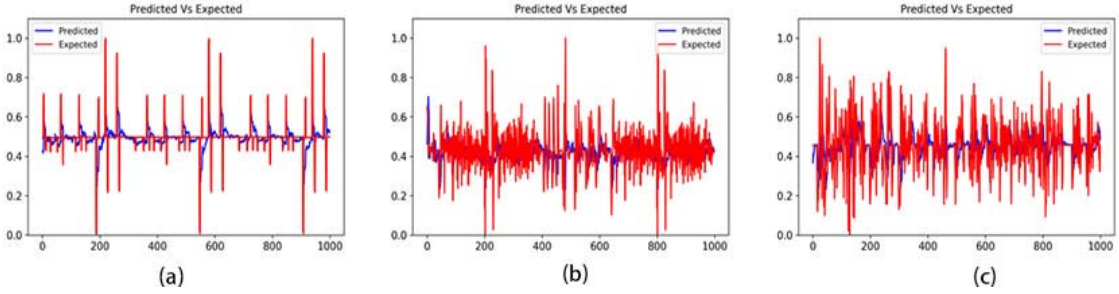

Figure 16 Expected vs. predicted plots with horizon set at three and lag cap set to 40 for a) DST, b) US06, c) FUDS.

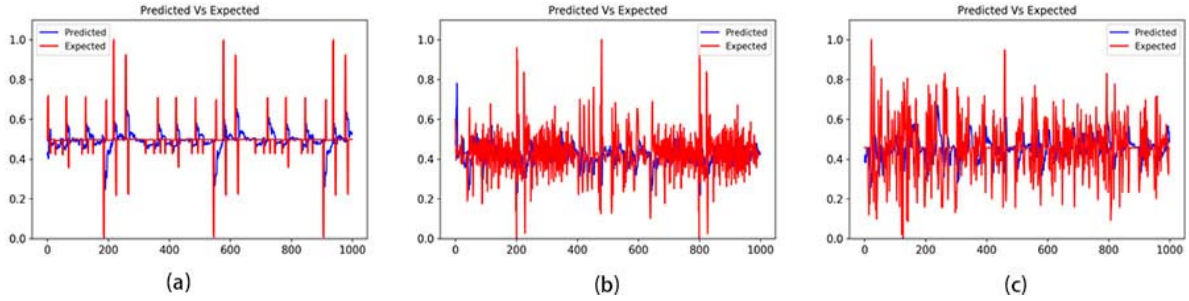

Figure 17 Expected vs. predicted plots with horizon set at five and lag cap set to 40 for a) DST, b) US06, c) FUDS.

Using offline training (where one section of the data is trained and validate once over another portion of the data), plots are generated (Figure 18) to ensure that bias is not introduced to the model. Although the following plot is only a snapshot, training and validation losses are consistent during training. It can be seen visually that 20 epochs are where training levels stabilize during offline training.

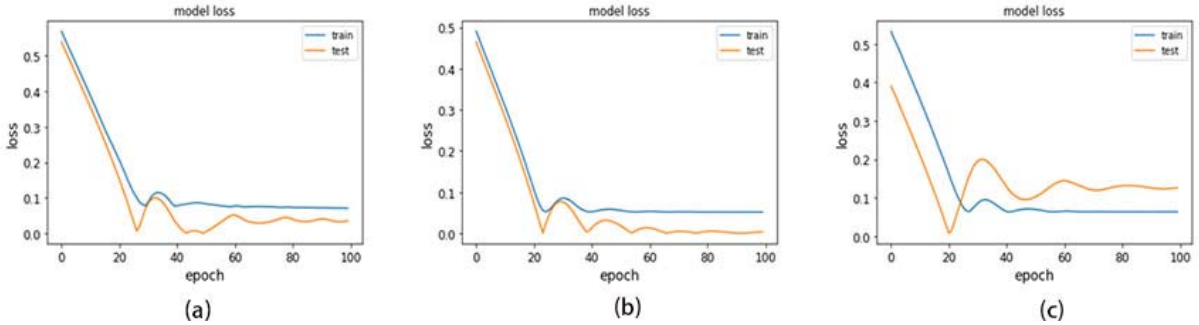

Figure 18 Training versus validation loss for offline profiles a) DST, b) US06, c) FUDS.

4. Conclusions

Although many consider deep learning to be a "black box" method, this study provides illumination into a methodology for using advanced recurrent neural networks within a vehicle battery management system. By examining feature importance, hyperparameter impact, contrasting model performance, and developing an algorithm for vehicle on-boarding, it is shown that recurrent neural networks combined with online training are a viable and impactful solution for next-generation vehicles.

In this research, feature analysis using a random forest algorithm to determine the impact of various parameters on the state of charge prediction was conducted. Although the literature assumes the impact of variables, this research quantifies the impact of temperature, voltage, and current on the state of charge. Next, hyperparameter search and comparison between an LSTM algorithm and models used in literature for the state of change prediction was performed. Tests were conducted on a high-performance cluster to perform gradient search to optimize the number of epochs, batch size, and gradient methods.

Although the first two objectives to this research are new applications for state of charge prediction, the methods are within deep learning literature. However, to on-board a recurrent

neural network into a vehicle system required moving away from high-performance cluster computing. By considering computational performance and the notion that a late predictor is worthless, a rules-based model was created and a continuous training data size control mechanism embedded into a recurrent neural network architecture. Through this mechanism, the feasibility of online predictions from 10 seconds up to 50 seconds in the future is demonstrated.

5. Conclusions and Future Studies

In this study, the feasibility of using advanced deep learning for online battery management in a vehicle platform is examined. Limiting training time (∼2 sec) and computational complexity allows for this battery predictive model to be on-boarded to a HEV application without hindering model performance. While these results indicate that continuous data size control allows for the mechanism of optimizing energy pulls from fuel cells to batteries, we plan to improve our model for eventual vehicle tests.

One limitation of this study is the use of only three driving profiles. Actual driving consists of a combination of many iterations of stop-and-go traffic, open freeway, urban driving, and city driving. To improve the BMS, the plan is to use a classifier in coordination with a recurrent neural network to enhance regression. More data would allow the use of a Conv-LSTM (Convolutional Neural Network classifier embedded in a Long Short-Term neural network regression algorithm) that optimizes data size and improves predictions during online training.

The authors are interested in studying capacity fade over long term use and implementing these effects into our model. This is an open area of research that could benefit from deep learning. Capacity fade is not considered in this study and is a limitation of this model since a battery performs differently as it ages. Future experiments are planned to account for how aging batteries in a HEV application effect state of charge, state of health, and overall effects on a BMS.

Acknowledgments

The authors wish to thank the Center for Advanced Life Cycle Engineering (CALCE) at the University of Maryland for the use of their battery data. The authors also want to thank the Graduate School, College of Engineering, and College of Business at New Mexico State University for their support.

Author Contributions

Both authors designed the model, organized the computational framework, and analyzed the data. Steven Hespeler carried out the implementation and performed the calculations. Donovan Fuqua provided academic guidance and planning for the research. Both authors conceived the study and wrote the manuscript.

Funding

The authors thank the Graduate School and College of Business at New Mexico State University for partial funding of this research.

Competing Interests

The authors have declared that no competing interests exist.

References

1. Xing Y, Ma EW, Tsui KL, Pecht M. Battery management systems in electric and hybrid vehicles. Energies. 2011; 4: 1840-1857.
2. Piller S, Perrin M, Jossen A. Methods for state-of-charge determination and their applications. J Power Sources. 2001; 96: 113-120.
3. Hannan MA, Lipu MS, Hussain A, Mohamed A. A review of lithium-ion battery state of charge estimation and management system in electric vehicle applications: Challenges and recommendations. Renew Sustain Energy Rev. 2017; 78: 834-854.
4. Zenati A, Desprez P, Razik H. Estimation of the SOC and the SOH of Li-ion batteries, by combining impedance measurements with the fuzzy logic inference. Proceedings of the IECON 2010-36th Annual Conference on IEEE Industrial Electronics Society; 2010 November 7-10th; Glendale, California, USA. Piscataway Township: Institute of Electrical and Electronics Engineers.
5. Shen Y. Adaptive online state-of-charge determination based on neuro-controller and neural network. Energy Convers Manag. 2010; 51: 1093-1098.
6. He Y, Liu X, Zhang C, Chen Z. A new model for State-of-charge (SOC) estimation for high-power Li-ion batteries. Appl Energy. 2013; 101: 808-814.
7. Waag W, Fleischer C, Sauer DU. Critical review of the methods for monitoring of lithium-ion batteries in electric and hybrid vehicles. J Power Sources. 2014; 258: 321-339.
8. Cuma MU, Koroglu T. A comprehensive review on estimation strategies used in hybrid and battery electric vehicles. Renew Sustain Energy Rev. 2015; 42: 517-531.
9. Sauer DU, Bopp G, Jossen A, Garche J, Rothert M, Wollny M. State of charge-What do we really speak about? Proceedings of the 21st international telecommunications energy conference; 1999 June 9th; Copenhagen, Denmark. Piscataway Township: Institute of Electrical and Electronics Engineers.
10. Chiasson J, Vairamohan B. Estimating the state of charge of a battery. IEEE Trans Control Syst Technol. 2005; 13: 465-470.
11. Chang WY. The state of charge estimating methods for battery: A review. ISRN Appl Math. 2013; 2013: 953792.
12. Coleman M, Lee CK, Zhu C, Hurley WG. State-of-charge determination from EMF voltage estimation: Using impedance, terminal voltage, and current for lead-acid and lithium-ion batteries. IEEE Trans Ind Electron. 2007; 54: 2550-2557.
13. Sato S, Kawamura A. A new estimation method of state of charge using terminal voltage and internal resistance for lead acid battery. Proceedings of the Power Conversion Conference-Osaka 2002 (Cat No 02TH8579); 2002 April 2-5th; Osaka, Japan. Piscataway Township: Institute of Electrical and Electronics Engineers.
14. Robinson RS. System noise as a signal source for impedance measurements on batteries connected to operating equipment. J Power Sources. 1993; 42: 381-388.
15. Huet F. A review of impedance measurements for determination of the state-of-charge or state-of-health of secondary batteries. J Power Sources. 1998; 70: 59-69.
16. Stoynov Z, Savova-Stoynov B, Kossev T. Non-stationary impedance analysis of lead/acid batteries. J Power Sources. 1990; 30: 275-285.

17. Blanchard P. Electrochemical impedance spectroscopy of small Ni- Cd sealed batteries: Application to state of charge determinations. J Appl Electrochem. 1992; 22: 1121-1128.
18. Kozlowski JD. Electrochemical cell prognostics using online impedance measurements and model-based data fusion techniques. 2003 IEEE Aerospace Conference Proceedings (Cat No 03TH8652); 2003 March 8-15th; Big Sky, Montana, USA. Piscataway Township: Institute of Electrical and Electronics Engineers.
19. Hung MH, Lin CH, Lee LC, Wang CM. State-of-charge and state-of-health estimation for lithium-ion batteries based on dynamic impedance technique. J Power Sources. 2014; 268: 861-873.
20. Bundy K, Karlsson M, Lindbergh G, Lundqvist A. An electrochemical impedance spectroscopy method for prediction of the state of charge of a nickel-metal hydride battery at open circuit and during discharge. J Power Sources. 1998; 72: 118-125.
21. Liu J, Saxena A, Goebel K, Saha B, Wang W. An adaptive recurrent neural network for remaining useful life prediction of lithium-ion batteries. Proceedings of the annual conference of the prognostics and health management society 2010; 2010 October 10-16th; Portland, Oregon, USA. New York: The Prognostics and Health Management Society.
22. Charkhgard M, Farrokhi M. State-of-charge estimation for lithium-ion batteries using neural networks and EKF. IEEE Trans Ind Electron. 2010; 57: 4178-4187.
23. Xu L, Wang J, Chen Q. Kalman filtering state of charge estimation for battery management system based on a stochastic fuzzy neural network battery model. Energy Convers Manag. 2012; 53: 33-39.
24. Shen WX, Chau KT, Chan CC, Lo EW. Neural network-based residual capacity indicator for nickel-metal hydride batteries in electric vehicles. IEEE Trans Veh Technol. 2005; 54: 1705-1712.
25. Cai C, Du D, Liu Z, Ge J. State-of-charge (SOC) estimation of high power Ni-MH rechargeable battery with artificial neural network. Proceedings of the 9th International Conference on Neural Information Processing, 2002. ICONIP '02; 2002 November 18-22; Singapore. Piscataway Township: Institute of Electrical and Electronics Engineers.
26. Linda O, William EJ, Huff M, Manic M, Gupta V, Nance J, et al. Intelligent neural network implementation for SOCI development of Li/CFx batteries. Proceedings of the 2009 2nd International Symposium on Resilient Control Systems; 2009 August 11-13th; Idaho Falls, Idaho, USA. Piscataway Township: Institute of Electrical and Electronics Engineers.
27. Zhang Y, Song W, Lin S, Feng Z. A novel model of the initial state of charge estimation for $LiFePO_4$ batteries. J Power Sources. 2014; 248: 1028-1033.
28. Antón JC, Nieto PJ, de Cos Juez FJ, Lasheras FS, Vega MG, Gutiérrez MN. Battery state-of-charge estimator using the SVM technique. Appl Math Model. 2013; 37: 6244-6253.
29. Wu X, Mi L, Tan W, Qin JL, Zhao MN. State of charge (SOC) estimation of Ni-MH battery based on least square support vector machines. Adv Mat Res. 2011; 211-212: 1204-1209.
30. Chen Y, Long B, Lei X. The battery state of charge estimation based weighted least squares support vector machine. Proceedings of 2011 Asia-Pacific Power and Energy Engineering Conference; 2011 March 25-28; Wu Han, China. Piscataway Township: Institute of Electrical and Electronics Engineers.
31. Shi QS, Zhang CH, Cui NX. Estimation of battery state-of-charge using v-support vector regression algorithm. Int J Automot Technol. 2008; 9: 759-764.
32. Singh P, Fennie Jr C, Reisner D. Fuzzy logic modelling of state-of-charge and available capacity of nickel/metal hydride batteries. J Power Sources. 2004; 136: 322-333.

33. Zhong F, Li H, Zhong S, Zhong Q, Yin C. An SOC estimation approach based on adaptive sliding mode observer and fractional order equivalent circuit model for lithium-ion batteries. Commun Nonlinear Sci Numer Simul. 2015; 24: 127-144.
34. Zhang Z, Cheng X, Lu Z, Gu D. SOC estimation of lithium-ion batteries with AEKF and wavelet transform matrix. IEEE Trans Power Electron. 2017; 32: 7626-7634.
35. Lin X. Theoretical analysis of battery SOC estimation errors under sensor bias and variance. IEEE Trans Ind Electron. 2018; 65: 7138-7148.
36. Ouyang Q, Chen J, Zheng J, Hong Y. SOC estimation-based quasi-sliding mode control for cell balancing in lithium-ion battery packs. IEEE Trans Ind Electron. 2018; 65: 3427-3436.
37. Guo Y, Zhao Z, Huang L. SoC estimation of lithium battery based on improved BP neural network. Energy Procedia. 2017; 105: 4153-4158.
38. He T, Li D, Wu Z, Xue Y, Yang Y. A modified luenberger observer for SoC estimation of lithium-ion battery. Proceedings of the 2017 36th Chinese Control Conference (CCC 2017); 2017 July 26-28; Dalian, China. Piscataway Township: Institute of Electrical and Electronics Engineers.
39. Purvins A, Sumner M. Optimal management of stationary lithium-ion battery system in electricity distribution grids. J Power Sources. 2013; 242: 742-755.
40. Zong Y, Mihet-Popa L, Kullmann D, Thavlov A, Gehrke O, Bindner HW. Model predictive controller for active demand side management with pv self-consumption in an intelligent building. Proceedings of the 3rd IEEE PES Innovative Smart Grid Technologies (ISGT) Europe Conference; 2012 October 14-17th; Berlin, Germany. Piscataway Township: Institute of Electrical and Electronics Engineers.
41. Castillo-Cagigal M, Gutiérrez A, Monasterio-Huelin F, Caamaño-Martín E, Masa D, Jiménez-Leube J. A semi-distributed electric demand-side management system with PV generation for self-consumption enhancement. Energy Convers Manag. 2011; 52: 2659-1666.
42. He W, Williard N, Chen C, Pecht M. State of charge estimation for Li-ion batteries using neural network modeling and unscented Kalman filter-based error cancellation. Int J Electr Power Energy Syst. 2014; 62: 783-791.
43. Li J, Danzer MA. Optimal charge control strategies for stationary photovoltaic battery systems. J Power Sources. 2014; 258: 365-373.
44. Hastie T, Tibshirani R, Friedman J. Random forests. In: The elements of statistical learning. New York: Springer; 2009. pp. 587-604.
45. Kuhn M, Johnson K. Regression Trees and Rule-Based Models. In: Applied Predictive Modeling. New York: Springer; 2013. pp. 173-220.
46. Sak H, Senior A, Beaufays F. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. Proceedings of the Interspeech 2014: 15th Annual Conference of the International Speech Communication Association; 2014 September 14-18th; Singapore. Baixas: International Speech Communication Association.
47. Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput. 1997; 9: 1735-1780.
48. Lipton ZC, Berkowitz J, Elkan C. A critical review of recurrent neural networks for sequence learning. Available from: https://arxiv.org/abs/1506.00019.
49. USABC electric vehicle battery test procedures manual. Revision2. Washington, DC, USA. United States Department of Energy. 1996.
50. Hu L, Hu X, Che Y, Feng F, Lin X, Zhang Z. Reliable state of charge estimation of battery packs using fuzzy adaptive federated filtering. Appl Energy. 2020; 262: 114569.

51. Kim I. The novel state of charge estimation method for lithium battery using sliding mode observer. J Power Sources. 2006; 163: 584-590.
52. Skrylnyk O, Lepore R, Ioakimidis CS, Remy M, Frère M. State-of-charge observers for lead-acid storage units used in autonomous solar applications. J Energy Storage. 2017; 14: 1-7.