Open Access

Journal of Energy and Power Technology

LIDSEN Publishing Inc.

Article Type: Research Article

ONLINE STATE OF CHARGE PREDICTION IN NEXT GENERATION VEHICLE BATTERIES USING DEEP RECURRENT NEURAL NETWORKS AND CONTINUOUS MODEL SIZE CONTROL

Hespeler, Steven [1,†], Fuqua, Donovan [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(s): dfuqua@nmsu.edu

Academic Editor(s): Name(s)

Special Issue:

Journal of Energy and Power Technology
2020, volume2,issue 1
doi:10.21926/jept.1901xxx

Received:
Accepted:
Published:

Abstract

This investigation presents a data-driven Long-short Term Memory (LSTM) battery model for predicting State of Charge (SOC) for lithium-ion batteries ($LiFePO_4$) for next-generation vehicle operations. Our modified LSTM algorithm builds and updates a model using multivariate inputs that include physical properties, voltage, current, and ambient temperature during operations. As a primary research goal, we want to improve prediction performance on future SOC 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 onboard vehicle systems, we 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
Deep Learning; Continuous Control; Battery Management

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, we focus our research 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 (6). In our test, we directly test our recurrent neural network algorithms against BPNN and BPNN+UKF methods.

In a BMS, SOC is widely considered the most influential and essential parameters for battery-operated applications. We define SOC 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). We can estimate SOC 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 our research, we rely on artificial intelligence methods 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 an ANN-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 ANN 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).

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). Finally, 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.

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, we identify the following knowledge gaps that we directly address in our 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) We see limited research on comparison of multiple methods for SOC prediction.

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

In this study, we utilize data 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

We conducted time computations 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, we utilized the New Mexico State University Discovery Computing Cluster. We executed all computations on Python 3 with the TensorFlow 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. We aimed to remove non-informative variables from the dataset, and to do this; we employed 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):

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

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 jth 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 jth feature with the Random Forest.

2.2 LSTM

2.2.1 Definition

Long-Short Term Memory (LSTM) is a specific type of 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.

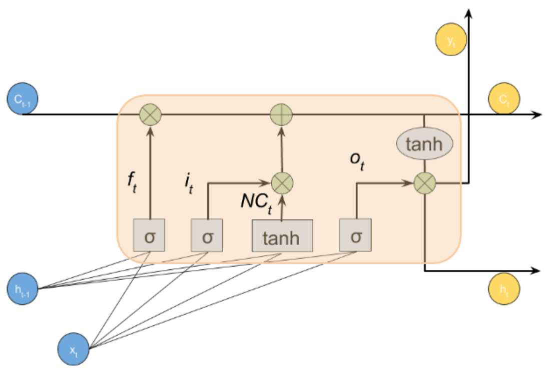

Figure 1 LSTM Memory Cell

We represent LSTM connections 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 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 a LiFePO4 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.

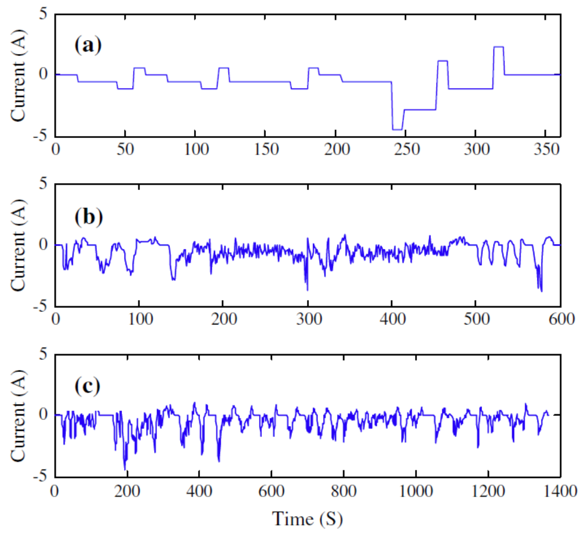

Figure 2 Loading profiles(current) of DST (top), US06 (middle), and FUDS (bottom)

<u>Load Profiles.</u> 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 $Celsius$. For this investigation, we focused on the 0 $Celsius$ datasets to test our algorithm.

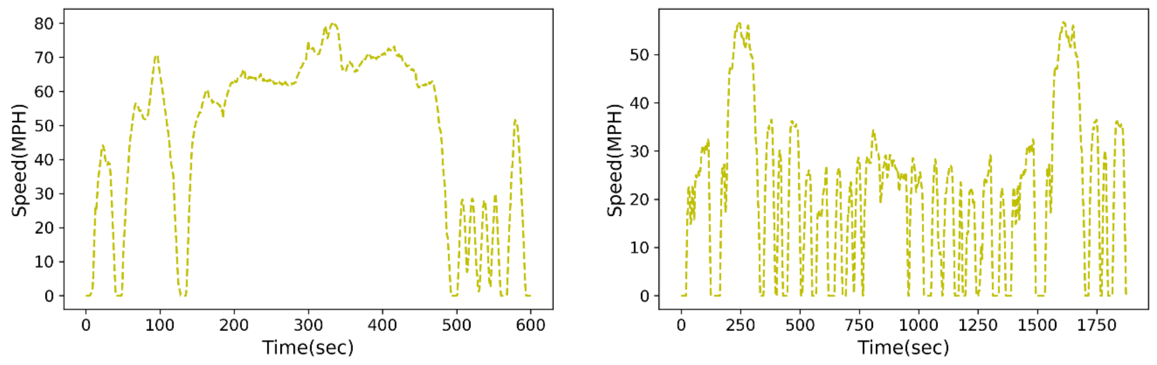

Figure 3 (a) US06 Highway Driving Schedule (b) FUDS Highway Driving Schedule

2.3.2 Data Formatting and Reshape

We import raw data to the predictive battery model as individual load profiles in the comma-separated value (CSV) format. Then, each dataset is cleaned and prepared. We eliminated the small number of rows containing empty (NaN) values and included the current, voltage, temperature, and SOC features. Once the cleaning was complete, we normalized the time series data to a range of (0, 1).

Next, we reshaped the data and prepared for training on a time-series cross-validation split. We ran experiments for all the data according to the following combinations of lag capacity, horizon, and load profile, with 36 runs completed for each dataset:

Table 1 Combinations of Lag Capacity, Horizon, and Load Profile

		Horizon		
Lag Capacity		1	3	5
10	DST			
	US06			
	FUDS			
20	DST			

	US06			
	FUDS			
30	DST			
	US06			
	FUDS			
40	DST			
	US06			
	FUDS			

In our construct, we create a walk-forward validation model 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 Xth 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). We reshape raw data into a 3D tensor with the shape of [batch, time steps, feature]. We perform this with a *reshape()* function 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 we reshape the input, we fit the model 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.

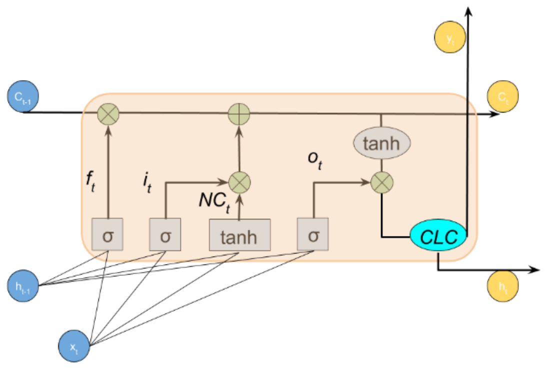

Figure 4 Illustration of LSTM-CLC memory block

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, we record testing parameters. 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, we restrict the data size up to the lag capacity during experimentation. 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.

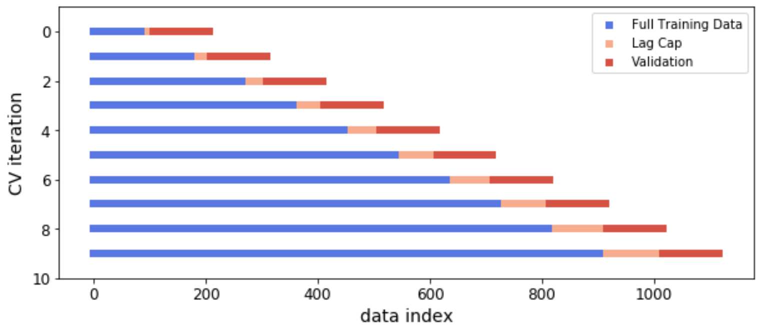

Figure 5 Visualization of Train Set Modification with Walk Forward Validation

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

Rule 1- Data Capacity

We use walk-forward validation 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, we cap lag 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. We run four different lag caps in the experiments: 10, 20, 30, and 40. The lag caps are somewhat arbitrary and can be increased when there is additional onboard computational power.

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).

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

In Figure 7, we show 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.

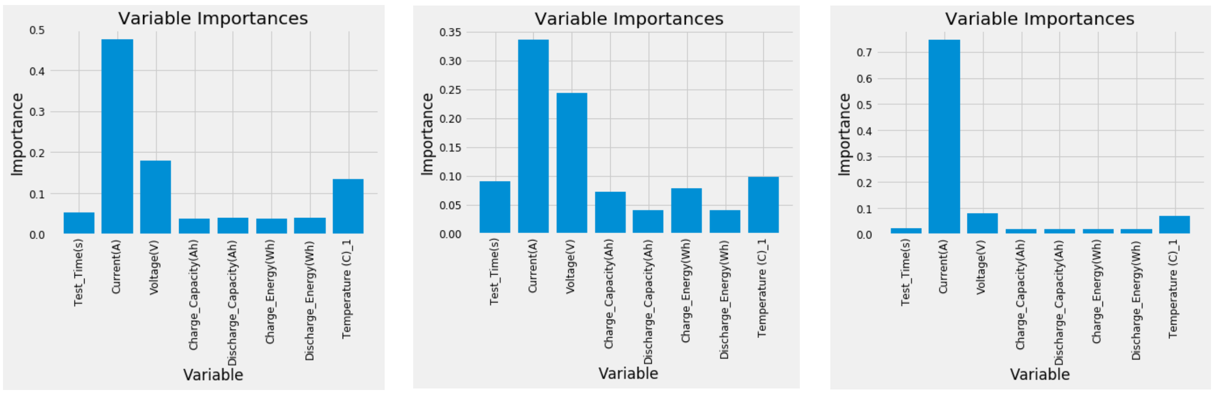

Figure 7 Feature Importance Comparison (Random Forest) for a) DST, b) FUDS, c) US06

3.2 Residual Analysis

During online training, we plot residuals 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. Therefore, we have confidence that while the outlier rule might not help with the DST model, it also does not impede learning.

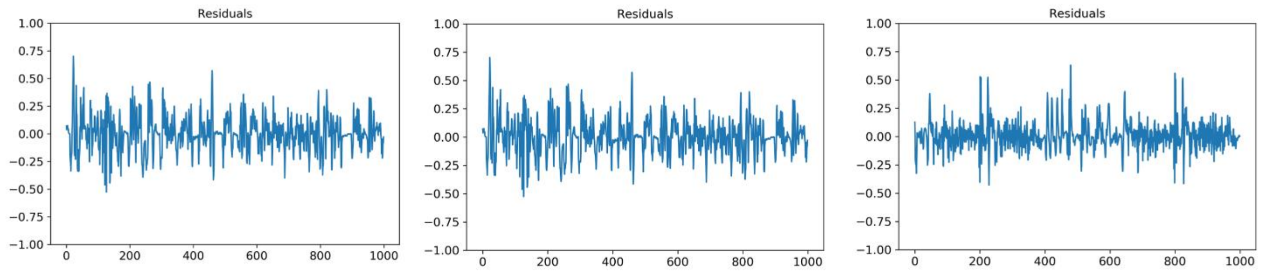

Figure 8 Residual *time series plots for validation results for a) DST, b) FUDS, and c) US06*

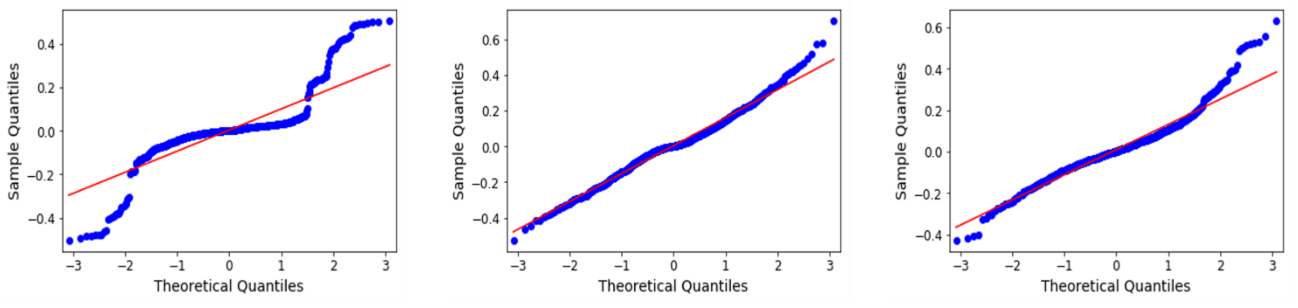

Figure 9 QQ Plots of Residuals for a) DST, b) FUDS, c) US06

In Tables 2-5, we illustrate the residual analysis for lag caps from 10 to 40. These tables show that residuals are consistent, given the size of the training data sets.

Table 2 Residual Analysis with a Lag Cap of 10

		H=1			H=3			H=5	
	DST	US06	FUDS	DST	US06	FUDS	DST	US06	FUDS
mean	-0.0038	0.0061	-0.0014	0.0028	0.0126	0.0096	0.0006	0.0006	0.0010
std	0.0723	0.0935	0.1202	0.1047	0.1292	0.1764	0.1288	0.0896	0.1876
min	-0.3700	-0.2960	-0.4240	-0.4921	-0.4078	-0.6602	-0.5508	-0.2602	-0.6468
25%	-0.0066	-0.0506	-0.0728	-0.0053	-0.0599	-0.0946	-0.0069	-0.0556	-0.1158
50%	0.0012	0.0012	-0.0021	0.0023	0.0128	0.0012	0.0030	-0.0057	0.0020
75%	0.0135	0.0521	0.0577	0.0234	0.0760	0.1054	0.0269	0.0442	0.0990
max	0.2920	0.7006	0.7028	0.5091	0.8305	0.7680	0.5059	0.4821	0.8368

Table 3 Residual Analysis with a Lag Cap of 20

		H=1			H=3			H=5	
	DST	US06	FUDS	DST	US06	FUDS	DST	US06	FUDS
mean	-0.0064	0.0066	0.0014	-0.0036	0.0123	0.0047	-0.0039	0.0098	-0.0013

std	0.0775	0.0886	0.1163	0.1067	0.1227	0.1600	0.1263	0.1282	0.1656
min	-0.3535	-0.2817	-0.4119	-0.4906	-0.3860	-0.5400	-0.5421	-0.4043	-0.4623
25%	-0.0072	-0.0483	-0.0634	-0.0086	-0.0573	-0.0833	-0.0129	-0.0586	-0.1020
50%	0.0017	-0.0029	-0.0025	0.0009	0.0034	-0.0016	0.0009	0.0035	-0.0007
75%	0.0128	0.0498	0.0554	0.0153	0.0717	0.0876	0.0201	0.0679	0.0925
max	0.2630	0.4998	0.4732	0.5160	0.8096	0.6316	0.5045	0.6409	0.7017

Table 4 Residual Analysis with a Lag Cap of 30

		H=1			H=3			H=5	
	DST	US06	FUDS	DST	US06	FUDS	DST	US06	FUDS
mean	-0.0040	0.0077	0.0061	-0.0008	0.0112	0.0064	-0.0001	0.0092	0.0020
std	0.0786	0.0874	0.1138	0.1068	0.1198	0.1529	0.1208	0.1251	0.1591
min	-0.3219	-0.2902	-0.3594	-0.4919	-0.3812	-0.5037	-0.5056	-0.4296	-0.5266
25%	-0.0133	-0.0466	-0.0564	-0.0132	-0.0557	-0.0806	-0.0169	-0.0589	-0.0841
50%	0.0011	-0.0001	-0.0022	0.0007	0.0024	-0.0015	0.0009	0.0029	-0.0012
75%	0.0170	0.0506	0.0638	0.0191	0.0661	0.0839	0.0216	0.0656	0.0891
max	0.2725	0.4965	0.4637	0.5111	0.7805	0.6310	0.5029	0.6380	0.7006

Table 5 Residual Analysis with a Lag Cap of 40

		H=1			H=3			H=5	
	DST	US06	FUDS	DST	US06	FUDS	DST	US06	FUDS
mean	-0.0018	0.0074	0.0074	0.0030	0.0114	0.0068	0.0040	0.0096	0.0044
std	0.0799	0.0864	0.1104	0.1039	0.1191	0.1501	0.1146	0.1239	0.1565
min	-0.3181	-0.2933	-0.3469	-0.4882	-0.3956	-0.4954	-0.5035	-0.4283	-0.5256
25%	-0.0233	-0.0448	-0.0522	-0.0192	-0.0536	-0.0761	-0.0203	-0.0560	-0.0815
50%	0.0013	-0.0026	-0.0004	0.0015	0.0031	-0.0006	0.0019	0.0014	0.0015
75%	0.0202	0.0503	0.0639	0.0205	0.0667	0.0847	0.0227	0.0639	0.0895
max	0.3137	0.4918	0.4641	0.5167	0.7704	0.6358	0.5057	0.6299	0.7014

3.2 Model Performance

In table 6, we compare our algorithm with comparable results from literature and our computations. For the NN (neural network, an artificial neural network) and the NN+UKF (neural network with unscented Kalman Filter), we take results from He (2014) (42). Because literature involving SOC prediction using SVM does not perform tests on the DST, US06, and FUDS datasets, we executed the tests with the three most common kernel types using the skLearn toolset in Python. We average results over ten runs and employ the CLC modification to show that the LSTM+CLC has the best predictive performance.

Table 6 Comparison of Different Predictive Models

	DST	US06	FUDS
NN	N/A	4.1	4.2

NN+UKF	N/A	2.4	2.2
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, we discuss the average RMSE values (over ten runs) with a ten-second (H=1), thirty-second (H=3), and fifty-second (H=5) future prediction. While we anticipated that our predictions would degrade given longer horizon time, the results demonstrate robust future prediction capabilities. We notice that 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.

Table 7 Average (Mean) of RMSE Results

	H=1			H=3			H=5		
Lag Cap	DST	US06	FUDS	DST	US06	FUDS	DST	US06	FUDS
10	0.06445	0.09118	0.13968	0.07783	0.11061	0.20160	0.08426	0.12008	0.21192
20	0.06957	0.08762	0.13206	0.07573	0.10358	0.17953	0.08273	0.11270	0.18426
30	0.06945	0.08727	0.12826	0.07618	0.10444	0.17234	0.08187	0.11210	0.18049
40	0.07045	0.08646	0.12505	0.07819	0.10165	0.16978	0.08389	0.11154	0.17812

Table 8 Standard Deviation of RMSE Results

	H=1			H=3			H=5		
Lag Cap	DST	US06	FUDS	DST	US06	FUDS	DST	US06	FUDS
10	0.01069	0.00757	0.02355	0.01200	0.01320	0.02858	0.01218	0.01272	0.02725
20	0.01107	0.00636	0.01975	0.01041	0.01358	0.02336	0.01130	0.01188	0.02231
30	0.01232	0.00602	0.01809	0.01113	0.01055	0.02247	0.01146	0.01074	0.02393
40	0.01242	0.00579	0.01783	0.01199	0.01312	0.02295	0.01213	0.01076	0.02429

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

Table 9 Results of Ten-Second Horizon

Lag Cap	**Load Profile**	**Ave SDE**	**Max SDE**	**Ave Time**	**Max Time**	**Total Time**
10	DST	0.06386627	0.079912443	2.065271543	3.577	2061.141
	US06	0.090597997	0.127160811	2.052972946	3.351	2048.867
	FUDS	0.139593303	0.207880845	2.099848697	3.15	2095.649
20	DST	0.069081643	0.086606597	2.051213	2.755	2051.213
	US06	0.08690564	0.123244956	2.034995	3.053	2034.995

	FUDS	0.131442564	0.169158205	2.057557	3.225	2057.557
30	DST	0.069153844	0.087470076	2.108107	3.948	2108.107
	US06	0.085772121	0.111791685	2.127751	5.079	2127.751
	FUDS	0.127146547	0.152129962	2.164085	4.512	2164.085
40	DST	0.070078222	0.088397603	2.030791	2.44	2030.791
	US06	0.085484205	0.123453846	2.039138	2.851	2039.138
	FUDS	0.124347128	0.154540519	2.037492	2.854	2037.492

In Tables 10 and 11, we push the predictive horizon to thirty seconds (H=3) and fifty seconds (H=5) and have roughly the same computational time per iteration.

Table 10 ~~Standard Deviation and Time~~ Results of Thirty-Second Horizon

Lag Cap	Load Profile	Ave SDE	Max SDE	Ave Time	Max Time	Total Time
10	DST	0.096124299	0.128579439	2.03743	3.125	2037.435
	US06	0.126353772	0.17080093	2.14472	4.049	2144.721
	FUDS	0.200245166	0.268061164	2.17139	5.119	2171.391
20	DST	0.097730039	0.129244181	2.06211	3.61	2062.11
	US06	0.121353343	0.178857826	2.06673	2.649	2066.734
	FUDS	0.178642592	0.222952451	2.07145	2.939	2071.45
30	DST	0.097121299	0.128353603	2.17344	3.643	2173.444
	US06	0.118900968	0.171584763	2.14458	3.37	2144.582
	FUDS	0.171409009	0.219296343	2.11010	3.347	2110.102
40	DST	0.094969001	0.128417297	2.15211	2.591	2152.111
	US06	0.11867284	0.177030567	2.15386	4.55	2153.868
	FUDS	0.169420723	0.21987988	2.14491	4.156	2144.919

Table 11 ~~Standard Deviation and Time~~ Results of Fifty-Second Horizon

Lag Cap	Load Profile	Ave SDE	Max SDE	Ave Time	Max Time	Total Time
10	DST	0.117482411	0.145291832	2.229806	4.335	2229.806
	US06	0.088886014	0.131248766	2.034150	3.051	2030.082
	FUDS	0.210808044	0.286806077	2.060138	2.876	2060.138
20	DST	0.114770244	0.143235343	2.131025	3.769	2131.025
	US06	0.12659186	0.145549367	2.092484	3.221	2092.484
	FUDS	0.183420291	0.251015491	2.154042	4.162	2154.042
30	DST	0.110015253	0.14435831	2.171285	5.362	2171.285
	US06	0.125402212	0.155607189	2.165733	4.142	2165.733
	FUDS	0.179694366	0.250905301	2.202841	3.992	2202.841
40	DST	0.105431892	0.144039809	2.168166	3.138	2168.166
	US06	0.124709532	0.157193981	2.168595	3.957	2168.595
	FUDS	0.177334507	0.251008321	2.168849	3.164	2168.849

In Figure 10, we show multiple runs (through 1000 steps) for the FUDS, US06, and DST loading profiles. With these profiles, we demonstrate that patterns with less variability (DST) train the same while there is more training variability in complex driving profiles (FUDS).

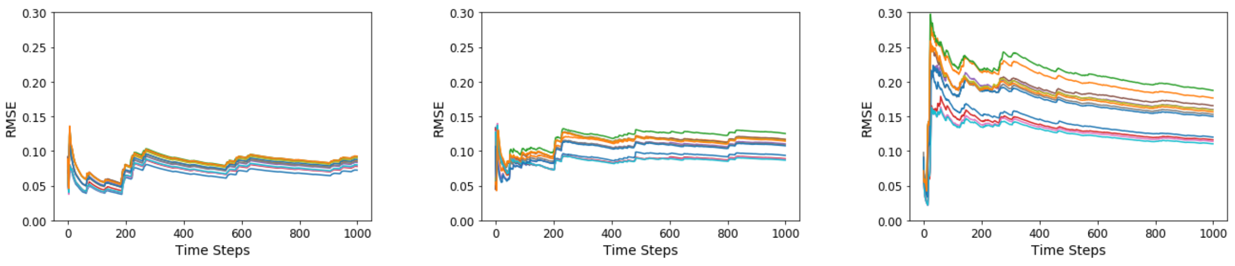

Figure 10 RMSE Values for runs of a) DST, b) US06, and c) FUDS

In Figure 11, we show the effect of controlling data size. In 2a, we show the necessity of adding a lag variable as training becomes too long after 500 iterations. In 2b, we show that while training time increases, the additional time does not equate to better training or a lower RMSE.

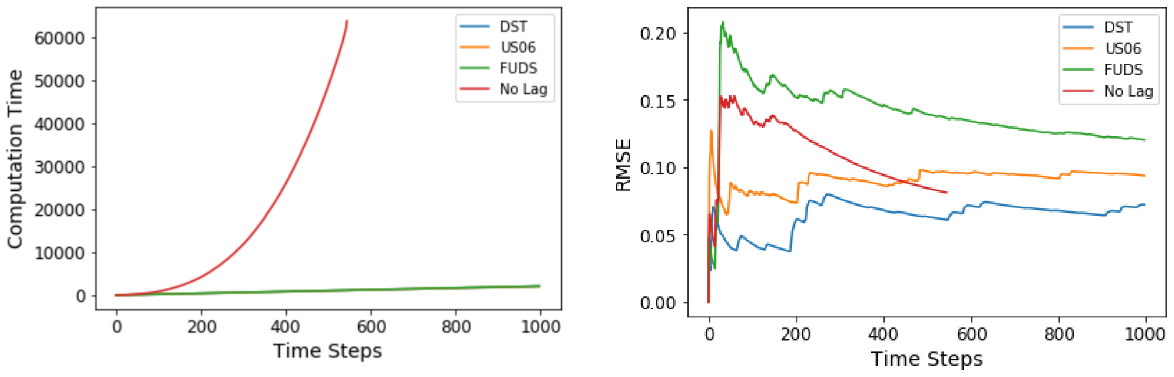

Figure 11 a) Comparison of Computation of load profiles with a lag cap and one profile with a lag cap and b) Standard error plots of the same load profiles

In Figures 12-17, we show plots of actual values versus predictive values at varying values of lag cap and validation horizon. As stated in Section 2, we ensured that our training and validation sets were separate during online training. Therefore, we see an apparent decrease of predictive power with broader training horizons and a smaller increase of power with larger lag caps.

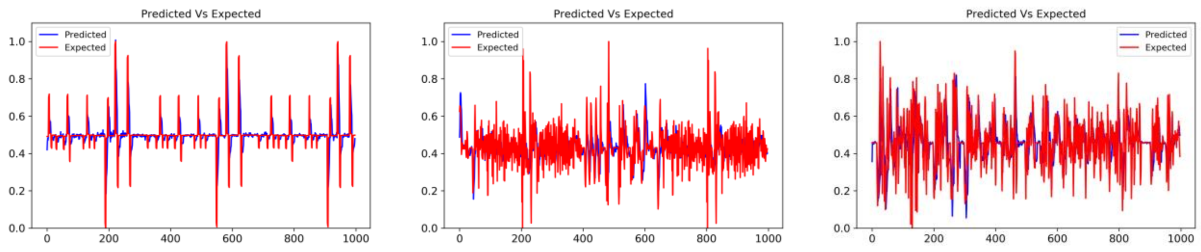

Figure 12 Expected vs. Predicted Plots with horizon set at one and lag cap set to 10 for a) DST, b) US06, c) FUDS

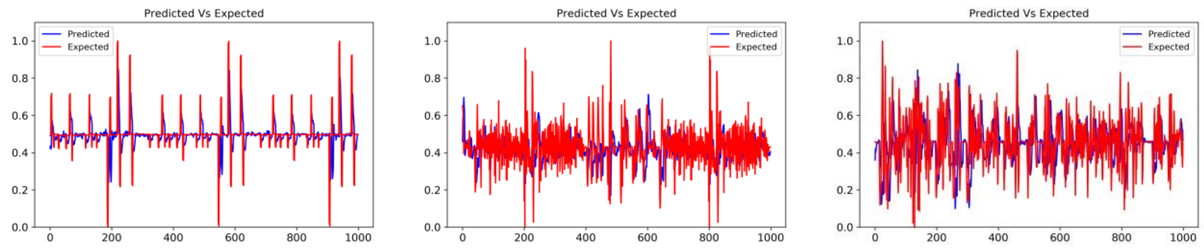

Figure 13 Expected vs. Predicted Plots with horizon set at three and lag cap set to 10 for a) DST, b) US06, c) FUDS

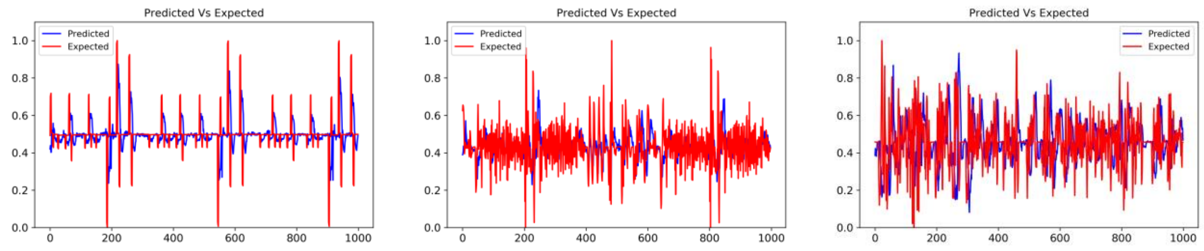

Figure 14 Expected vs. Predicted Plots with horizon set at five and lag cap set to 10 for a) DST, b) US06, c) FUDS

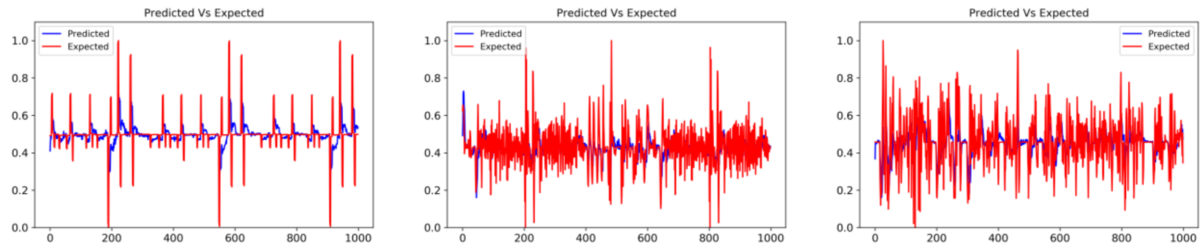

Figure 15 Expected vs. Predicted Plots with horizon set at one and lag cap set to 40 for a) DST, b) US06, c) FUDS

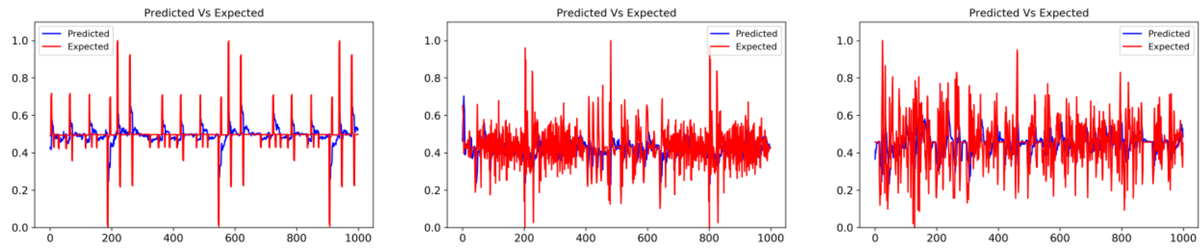

Figure 16 Expected vs. Predicted Plots with horizon set at three and lag cap set to 40 for a) DST, b) US06, c) FUDS

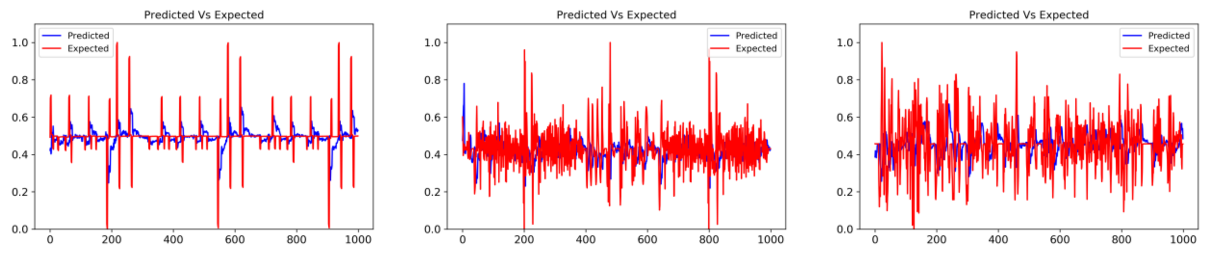

Figure 17 Expected vs. Predicted Plots with horizon set at five and lag cap set to 40 for a) DST, b) US06, c) FUDS

Using offline training (where we train one section of the data and validate once over another portion of the data), we generate plots (Figure 18) to ensure that we have not introduced bias in our model. Although the following plot is only a snapshot, we see that our training and validation losses are consistent during training. We also understand that 20 epochs are where training levels stabilize during offline training.

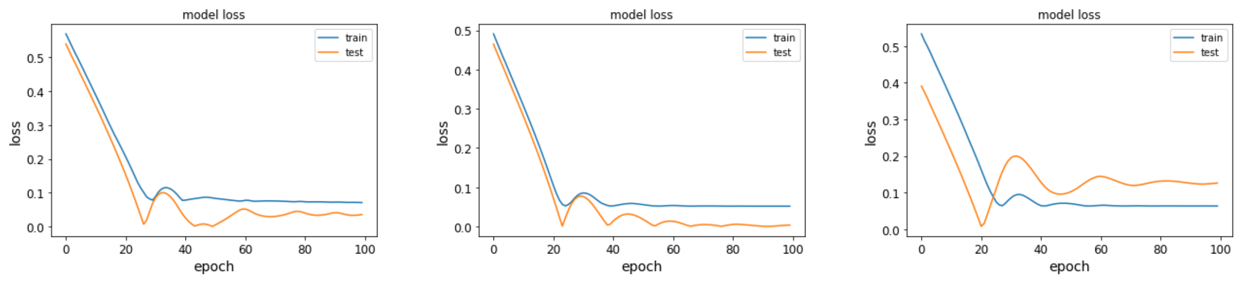

Figure 18 Training versus Validation Loss for offline profiles a) DST, b) US06, c) FUDS

4. Discussion

Although many consider deep learning to be a "black box" method, we hope this study provides illumination into a methodology for using recurrent neural networks within a vehicle battery management system. By examining feature importance, hyperparameter impact, contrasting model performance, and developing an algorithm for vehicle onboarding, we show that recurrent neural networks and online training are useful for next-generation vehicles.

In this research, we conduct a feature analysis using a random forest algorithm to determine the impact of various variables on the state of charge prediction. Although the literature assumes the impact of variables, our research quantifies the impact of temperature, voltage, and current on the state of charge. Next, we perform a hyperparameter search and comparison between our LSTM algorithm and models used in literature for the state of change prediction. We conducted these tests on a high-performance cluster to perform gradient search to optimize our 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 onboard 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, we develop a rules-based continuous data size control mechanism into our recurrent neural network architecture. Through this mechanism, we demonstrate the feasibility of online predictions from 10 seconds up to 50 seconds in the future.

5. Conclusions and Future Studies

In this study, we demonstrated the feasibility of using advanced deep learning for online battery management in a vehicle platform. While our results indicate that continuous data size control allows us the mechanism for 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 (FUDS, US06, and DST). Actual driving consists of combinations of many iterations of stop-and-go traffic, open freeway, urban driving, and city driving. To improve BMS predictions, we hope to use a classifier in coordination with a recurrent neural network to enhance regression. More data would allow us to use 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. We are also interested in studying capacity fade over long term use and its impact on the state of charge and state of health of batteries. We feel that this is an open area of research that could benefit from deep learning.

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. We also want to thank the Graduate School, College of Engineering, and College of Business at New Mexico State University for their support.

Additional Materials (if any)

No additional materials are attached to this manuscript. We present coding and datasets at https://github.com/Tov-Nephesh.

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

We 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 EWM, Tsui KL, Pecht M. Battery management systems in electric and hybrid vehicles. Energies. 2011;4(11):1840–57.

2. Piller S, Perrin M, Jossen A. Methods for state-of-charge determination and their applications. J Power Sources. 2001;96(1):113–20.

3. Hannan MA, Lipu MSH, 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;78:834–54.

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. In: IECON 2010-36th Annual Conference on IEEE Industrial Electronics Society. 2010. p. 1773–8.

5. Shen Y. Adaptive online state-of-charge determination based on neuro-controller and neural network. Energy Convers Manag. 2010;51(5):1093–8.

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;101:808–14.

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;258:321–39.

8. Cuma MU, Koroglu T. A comprehensive review on estimation strategies used in hybrid and battery electric vehicles. Renew Sustain Energy Rev. 2015;42:517–31.

9. Sauer DU, Bopp G, Jossen A, Garche J, Rothert M, Wollny M. State of Charge--What do we really speak about. In: The 21st international telecommunications energy conference. 1999. p. 6–9.

10. Chiasson J, Vairamohan B. Estimating the state of charge of a battery. In: Proceedings of the 2003 American Control Conference, 2003. 2003. p. 2863–8.

11. Chang W-Y. The state of charge estimating methods for battery: A review. ISRN Appl Math. 2013;2013.

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(5):2550–7.

13. Sato S, Kawamura A. A new estimation method of state of charge using terminal voltage and internal resistance for lead acid battery. In: Proceedings of the Power Conversion Conference-Osaka 2002 (Cat No 02TH8579). 2002. p. 565–70.

14. Robinson RS. System noise as a signal source for impedance measurements on batteries connected to operating equipment. J Power Sources. 1993;42(3):381–8.

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(1):59–69.

16. Stoynov Z, Savova-Stoynov B, Kossev T. Non-stationary impedance analysis of lead/acid batteries. J Power Sources. 1990;30(1–4):275–85.

17. Blanchard P. Electrochemical impedance spectroscopy of small Ni- Cd sealed batteries: Application to state of charge determinations. J Appl Electrochem. 1992;22(12):1121–8.

18. Kozlowski JD. Electrochemical cell prognostics using online impedance measurements and model-based data fusion techniques. In: 2003 IEEE Aerospace Conference Proceedings (Cat No 03TH8652). 2003. p. 3257–70.

19. Hung M-H, Lin C-H, Lee L-C, Wang C-M. State-of-charge and state-of-health estimation for lithium-ion batteries based on dynamic impedance technique. J Power Sources. 2014;268:861–73.

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(2):118–25.

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. 2010.

22. Charkhgard M, Farrokhi M. State-of-charge estimation for lithium-ion batteries using neural networks and EKF. IEEE Trans Ind Electron. 2010;57(12):4178–87.

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(1):33–9.

24. Shen WX, Chau KT, Chan CC, Lo EWC. Neural network-based residual capacity indicator for nickel-metal hydride batteries in electric vehicles. IEEE Trans Veh Technol. 2005;54(5):1705–12.

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. In: Proceedings of the 9th International Conference on Neural Information Processing, 2002 ICONIP'02. 2002. p. 824–8.

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. In: 2009 2nd International Symposium on Resilient Control Systems. 2009. p. 57–62.

27. Zhang Y, Song W, Lin S, Feng Z. A novel model of the initial state of charge estimation for LiFePO4 batteries. J Power Sources. 2014;248:1028–33.

28. Antón JCÁ, Nieto PJG, de Cos Juez FJ, Lasheras FS, Vega MG, Gutiérrez MNR. Battery state-of-charge estimator using the SVM technique. Appl Math Model. 2013;37(9):6244–53.

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. In: Advanced Materials Research. 2011. p. 1204–9.

30. Chen Y, Long B, Lei X. The battery state of charge estimation based weighted least squares support vector machine. In: 2011 Asia-Pacific Power and Energy Engineering Conference. 2011. p. 1–4.

31. Shi Q-S, Zhang C-H, Cui N-X. Estimation of battery state-of-charge using v-support vector regression algorithm. Int J Automot Technol. 2008;9(6):759–64.

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(2):322–33.

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(1–3):127–44.

34. Zhang Z-L, Cheng X, Lu Z-Y, Gu D-J. SOC estimation of lithium-ion batteries with AEKF and wavelet transform matrix. IEEE Trans Power Electron. 2017;32(10):7626–34.

35. Lin X. Theoretical Analysis of Battery SOC Estimation Errors under Sensor Bias and Variance. IEEE Trans Ind Electron. 2018;

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(4):3427–36.

37. Guo Y, Zhao Z, Huang L. SOC Estimation of lithium battery based on improved BP neural network. Energy Procedia. 2017;105:4153–8.

38. He T, Li D, Wu Z, Xue Y, Yang Y. A modified luenberger observer for SOC estimation of lithium-ion battery. In: Control Conference (CCC), 2017 36th Chinese. 2017. p. 924–8.

39. Purvins A, Sumner M. Optimal management of stationary lithium-ion battery system in electricity distribution grids. J Power Sources. 2013;242:742–55.

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. In: Innovative Smart Grid Technologies (ISGT Europe), 2012 3rd IEEE PES International Conference and Exhibition on. 2012. p. 1–8.

41. Castillo-Cagigal M, Gutiérrez A, Monasterio-Huelin F, Caamaño-Mart\'\in 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(7):2659–66.

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;62:783–91.

43. Li J, Danzer MA. Optimal charge control strategies for stationary photovoltaic battery systems. J Power Sources. 2014;258:365–73.

44. Hastie T, Tibshirani R, Friedman J. Random forests. In: The elements of statistical learning.

Springer; 2009. p. 587–604.

45. Kuhn M, Johnson K. Regression trees and rule-based models. In: Applied predictive modeling. Springer; 2013. p. 173–220.

46. Sak H, Senior A, Beaufays F. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In: Fifteenth annual conference of the international speech communication association. 2014.

47. Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput. 1997;9(8):1735–80.

48. Lipton ZC, Berkowitz J, Elkan C. A critical review of recurrent neural networks for sequence learning. arXiv Prepr arXiv150600019. 2015;

49. Hunt G, others. USABC electric vehicle battery test procedures manual. Washington, DC, USA United States Dep Energy. 1996;