CS230

Deep learning-based capacity estimation for lithium-ion electric vehicle batteries

Kevin Russell Moy
Department of Energy Resources Engineering
Stanford University
kmoy14@stanford.edu

Abstract

The capacity of lithium-ion batteries decreases over the lifetime of the battery. This capacity fade is a marker for state-of-health (SOH) of the battery; as a battery is used, its SOH decreases from 100% to 0%. Knowing the SOH and capacity fade is important for the safe operation of lithium-ion batteries. This paper proposes a deep neural network to predict electric vehicle (EV) battery capacity, given realistic information (current and voltage) measured as the EV is driven. Between an MLP and CNN model, the CNN model provides superior performance in predicting capacity, able to achieve 0.015% error with only 2 minutes of current and voltage driving data.

1 Introduction

As lithium-ion batteries are deployed and used in applications such as electric vehicles, their capacities decrease (fade); this capacity fade is highly dependent on operating conditions such as driving patterns and ambient temperatures [6]. Accurately estimating the capacity of an EV battery is important to evaluate state-of-health (SOH) of the battery and ensure that the EV remains in safe operating conditions; for example, the "mileage remaining" of an EV will change as a battery ages, so shorter trips must be taken to avoid being stranded on the highway. Furthermore, EV batteries can be recycled for "second-life" usage, where they are recycled and re-used for stationary grid energy storage systems [3]. Estimating capacity is also important to understand how much capacity remains for usage in the stationary grid storage second-life of the battery.

Capacity is also cumbersome to measure in real-life as it is done experimentally. In the laboratory, the battery is charged to 100% state of charge (SOC), and then discharged at a very low current to 0% SOC. The battery is discharged as close to 0 Amps as possible, because discharged capacity decreases as discharge current increases. It is unrealistic to assume that EV owners will wait 20+ hours for the EV to measure in real-time the capacity of its battery. Therefore, this paper leverages laboratory-collected data via deep learning, using collected battery telemetry similar to that collected during EV driving to predict capacity fade as a measure of SOH. The outcome will be a model that estimates the capacity of the battery given operational voltage and current data taken at different stages of the battery's life.

2 Related Work

Previous papers have used convolutional neural networks, which used charging segments based on data collected with constant-current/constant-voltage (CC/CV) charge and discharge [4, 8]. Conversely, this project will focus on discharging data with current and voltage driven by more realistic electric

CS230: Deep Learning, Spring 2022, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

vehicle drive cycle profile data. Beyond this, other papers have used long short-term memory recurrent neural networks (LSTM-RNN) to estimate capacity [10, 9]; however, as the dataset does not currently span a long enough capacity fade trajectory (only having gone down to ≈90% nominal capacity), so we cannot yet use this method in any meaningful fashion.

3 Dataset and Features

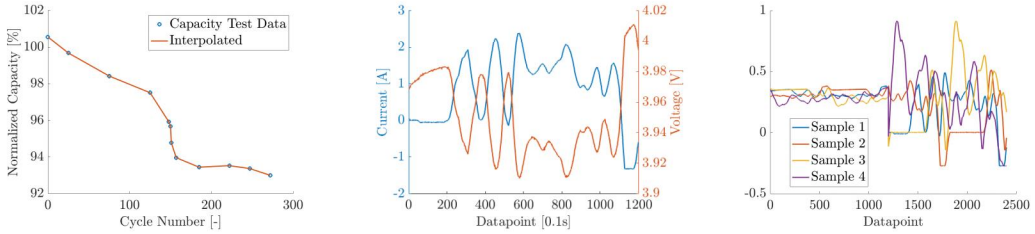
The data for this project will be drawn from a dataset collected by the Stanford Energy Control Lab. The dataset consists of data collected from several lithium-ion batteries cycled according to a protocol which closely mimics electric vehicle driving. The data for each battery consists of two parts:

1. Cycle data, where one cycle represents a full discharge with an electric vehicle "UDDS" driving current profile from 80% SOC down to 20% SOC, and charging at a fixed constant current, and
2. Diagnostic data, which is taken at intermittent cycle numbers to measure capacity and resistance of the battery.

Both parts consist of current and voltage data collected at 0.1 second resolution, and the battery temperature was held constant in a thermal chamber at 23°C. We use data from four different cells: W5, W8, W9, and W10. A detailed description of the experimental setup and dataset format can be found in [7]. We can leverage the current and voltage data as they are indicative of resistance (e.g. via Ohm's Law: V/I = R) and the voltage will change for the fixed UDDS current profile as the battery ages.

3.1 Pre-processing

We use the cycle capacity data as the output. The diagnostic capacity values are only measured at intermittent cycle numbers, so the battery capacity is linearly interpolated between these values for each intermediate cycle integer number. The capacity is normalized to the nominal cell capacity of 4.85 Ampere-hours (Ah) and reported as a percentage out of 100. As an example, the capacity values for each cycle for the W8 cell are shown in Figure 1a.



(a) (b) (c)

Figure 1: (a) W8 cell data for capacity as a function of cycle number. Note capacity can go above 100%, since the cell started above nominal capacity. (b) Example of W8 cell data current/voltage segment. (c) Example of four input samples, of concatenated normalized voltage and current.

We use the cycle discharge data as the input. For each cycle, the UDDS discharge current and voltage are segmented into 1200 datapoints (120 seconds, or 2 minutes) each. An example of a pair of current and voltage segments is shown in Figure 1b. These segments overlap by 600 datapoints. As each cycle lasts approximately 5 hours (or 180,000 datapoints), each cycle has approx. 300 corresponding current/voltage segment pairs. The current is normalized by the nominal capacity (4.85Ah) and the voltage is normalized by the nominal voltage (3.63V) in order to keep the values of the input between $[-1, 1]$. Four examples of inputs are shown in Figure 1c.

4 Methods

This paper uses two different deep learning models to estimate capacity as a function of measured current and voltage. Both models are implemented in TensorFlow [1] and Keras [2], and trained on

this loss function using the Adam optimization algorithm with default hyperparameters. The W8 cell dataset is used as training and validation data with an 80%-20% train-dev split, and the other three cell datasets (W5, W9, and W10) are used as test data. We make the assumption that the dev and test sets are drawn from the same distribution given that all cells are nearly identical, and that within the small time window of current and voltage segments drawn from for the model inputs, that the cells behave nearly identically to each other.

The metric used to compare between the different methods will be the root-mean-squared percent error (RMSPE) of the test set. For example, given M test set examples and M predictions $\widehat{y}$ on ground-truth labels y, the test set performance is given by

$$RMSPE_{test} = \sqrt{\frac{1}{M}\sum_{i=1}^{M}(y_i - \widehat{y}_i)^2} * 100\% \tag{1}$$

As the original capacity is scaled out of 100%, rescaling by this provides the RMSPE.

4.1 Multi-Layer Perceptron (MLP)

The first approach uses a standard fully-connected MLP. Each pair of current and voltage segments is concatenated into a 2400×1 vector as the input to the neural network. KerasTuner [5] is used to tune the number of hidden layers as well as the number of nodes in each hidden fully-connected layer (details in Appendix 1). From random search of hyperparameter values in KerasTuner, the following fully-connected hidden layer sizes are used: $[1200, 1200, 2400, 200, 20]$, each with ReLU activation on each node. The output layer is a linear regression on the output of the last hidden layer to predict capacity. A diagram of the network structure is shown in Figure 2. The model is trained for 5000 epochs with a batch size of 500. [1]

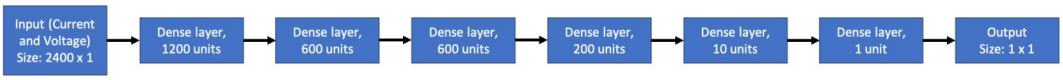


Figure 2: Plot of entire MLP model architecture.

The loss function for the MLP model is the mean-absolute error (MAE) between estimated capacity $\widehat{y}_i$ and true outputs y_i across all training examples K as shown in Equation 2. The mean-squared error (MSE; see Equation 3) showed poor convergence and extreme overfitting to the training data (see Appendix 2), so MAE was used for its superior results.

$$\mathcal{L} = \frac{1}{K}\sum_{i=1}^{K}|y_i - \widehat{y}_i| \tag{2}$$

4.2 Convolutional Neural Network (CNN)

The second approach uses a CNN. The structure was hand-tuned given the many parameters chosen, e.g., number of layers in each CNN, but also the layer type in each layer. For example, the SeparableConv1D layer in Keras was originally considered as an option for the first hidden layer, as it treats each input channel separately before combining them, but it did not provide fast loss convergence during training, and so is not presented in this paper (but is available in the GitHub repo). [2] However, we use a similar concept from SeparableConv1D in the final architecture. The current and voltage are treated as separate 1D channels of dimension 1200×1, each with its own CNN block shown in Figure 3.

In general, the network was designed so that the number of parameters within each was evenly distributed; for example, making sure that the output of the CNN block was small enough so that

[1] Model can be found at `https://github.com/kevinrussellmoy/cs-230-final-project/blob/main/MLP.ipynb`.

[2] SeparableConv1D model can be found at `https://github.com/kevinrussellmoy/cs-230-final-project/blob/main/CNN_MSE_v2.ipynb`.

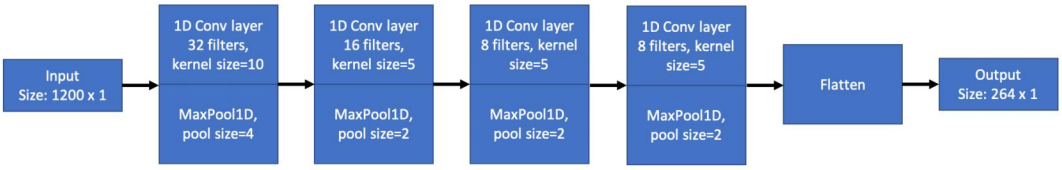


Figure 3: Plot of CNN block. Unless otherwise noted, all hyperparameters left at default values (e.g. valid padding for all 1D Conv layers.)

the densely-connected layers did not contain too many parameters. The network hyperparameters were then hand-tuned to minimize the output loss. The output of each CNN is concatenated and then combined in a series of densely-connected layers with ReLU activation. As in the MLP model, the output layer is a single-output regression for the estimated capacity. A diagram of the entire model structure is shown in Figure 4. For use in this model, the capacity values are scaled using the `MinMaxScaler` in `sklearn` to have minimum 0 and maximum 1 across all samples.

The loss function for the CNN models is the mean-squared error (MSE) between estimated capacity $\widehat{y}_i$ and true outputs y_i across all training examples K as shown in Equation 3.

$$\mathcal{L} = \frac{1}{K}\sum_{i=1}^{K}(y_i - \widehat{y}_i)^2 \tag{3}$$

The model is trained for 500 epochs with batch size 92. [3]

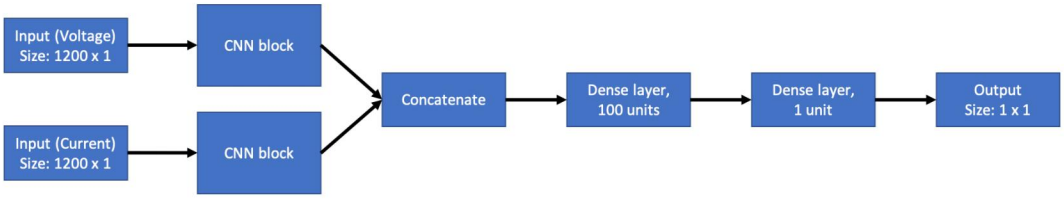


Figure 4: Plot of entire CNN model architecture with CNN block denoted as in Figure 3.

5 Results

The models are evaluated on the W5, W9, and W10 cell test datasets, and the test RMSPE is used to compare them. The results are shown in Table 1.

Table 1: Validation Results

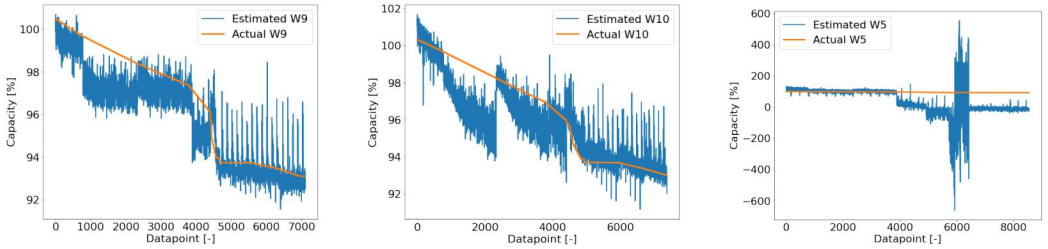
Model	Test Data	RMSPE [%]
MLP	W9	1.15
	W10	2.22
	W5	6143.19
CNN	W9	0.015
	W10	0.020
	W5	83.16

Figure 5 shows the predicted (estimated) CNN model output vs. actual capacity for the test data cells, while Figure 6 shows the MLP model output vs. actual capacity. We can see that both models perform best on the W8 cell, followed by the W10 cell, as evidenced by how well the capacity trajectory is traced out by both models. The W5 cell capacity is poorly estimated by both models, predicting

[3]Model can be found at `https://github.com/kevinrussellmoy/cs-230-final-project/blob/main/CNN_final_model.ipynb`.

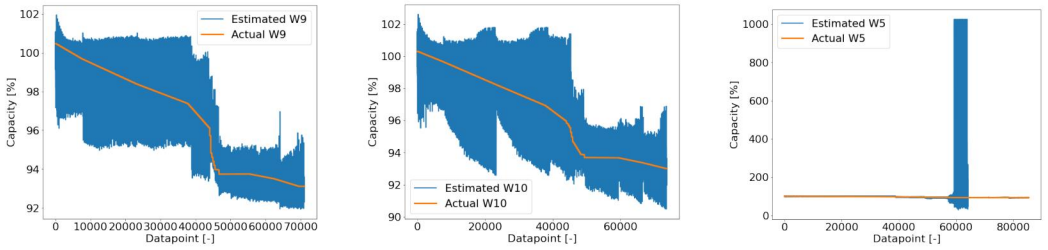
capacities well in excess of 100% and below 0%. This may show that our initial assumption was incorrect and that the W5 data represents some distribution of current and voltage segments not seen in the W8, W9, and W10 data. Therefore, future work would include more diverse data from different cells in the training dataset.

Comparing the two models, the CNN performs uniformly better than the MLP, and appears to be more robust to the distribution of the test dataset, as seen by the lower RMSPE in W5 from the CNN compared to the MLP. Additionally, the MLP model took 5000 epochs to train compared to the CNN, which only required a tenth of the time.



(a) (b) (c)

Figure 5: Validation results for the CNN trained on W8 cell data on (a) W9 cell data for capacity, (b) W10 cell data for capacity, (c) W5 cell data for capacity. Note capacity can go above 100%, since the cell started above nominal capacity.



(a) (b) (c)

Figure 6: Validation results for the MLP trained on W8 cell data on (a) W9 cell data for capacity, (b) W10 cell data for capacity, (c) W5 cell data for capacity. Note capacity can go above 100%, since the cell started above nominal capacity.

6 Conclusions

Two deep learning models were developed and presented to solve the problem of estimating capacity from current and voltage measurements that could reflect real-time data during EV driving. Of these two, the CNN provides the best balance of accuracy, as measured by the RMSPE of the predicted capacity, as well as the computation time, and may even overfit to the training data as exhibited by the W5 test results. Meanwhile, the MLP can only discern general trends in the capacity fade among all cells. We hypothesize that the CNN is able to treat the current and voltage as separate signals and reduce the signals down to key features in their separate 1D CNNs, and so can fit better to the training data.

Future work would include incorporating more experimental data into the training set, as well as different pre-processing methods, such as employing the Fourier Transform to convert the current and voltage data into their frequency components as model inputs. We could also consider hyperparameter tuning within the CNN framework, adding more layers of different sizes, as well as tuning the length of the input current and voltage segments. Finally, as more experimental data is collected on these cells, we would like to incorporate some memory or dependence on the previous known/estimated capacity state to predict future capacity, which would lend well to some recurrent neural network architectures over these longer capacity fade trajectories.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] F. Chollet. keras. `https://github.com/fchollet/keras`, 2015.

[3] H. Engel, P. Hertzke, and G. Siccardo. Second-life ev batteries: The newest value pool in energy storage, May 2019.

[4] Y. Li, K. Li, X. Liu, and L. Zhang. Fast battery capacity estimation using convolutional neural networks. *Transactions of the Institute of Measurement and Control*, 2020.

[5] T. O'Malley, E. Bursztein, J. Long, F. Chollet, H. Jin, L. Invernizzi, et al. KerasTuner. `https://github.com/keras-team/keras-tuner`, 2019.

[6] E. Paffumi and G. Martini. Real-world mobility and environmental data for the assessment of in-vehicle battery capacity fade. *World Electric Vehicle Journal*, 12(1), 2021.

[7] G. Pozzato, A. Allam, and S. Onori. Lithium-ion battery aging dataset based on electric vehicle real-driving profiles. *Data in Brief*, 41, 2022.

[8] C. Qian, B. Xu, L. Chang, B. Sun, Q. Feng, D. Yang, Y. Ren, and Z. Wang. Convolutional neural network based capacity estimation using random segments of the charging curves for lithium-ion batteries. *Energy*, 227, 2021.

[9] G.-W. You, S. Park, and D. Oh. Diagnosis of electric vehicle batteries using recurrent neural networks. *IEEE Transactions on Industrial Electronics*, 64(6), 2017.

[10] Y. Zhang, R. Xiong, H. He, and M. G. Pecht. Long short-term memory recurrent neural network for remaining useful life prediction of lithium-ion batteries. *IEEE Transactions on Vehicular Technology*, 67(7), 2018.

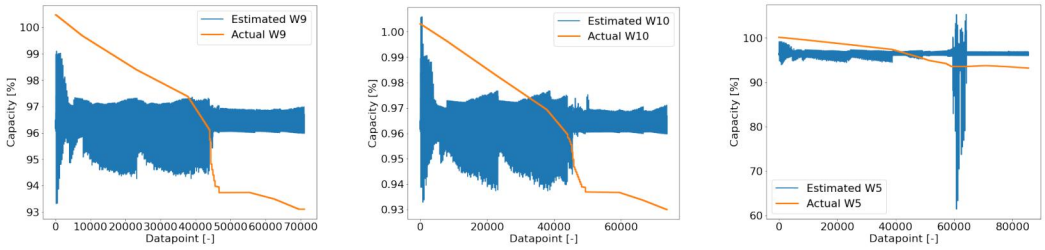
Appendix 1: KerasTuner configuration

The KerasTuner tuner for the MLP was configured with the following values. Note the network starts with "large" hidden layers with size close to the input size (2400×1), while the succeeding layers narrow the network down before the single regression output layer. The tuner ran for 5 epochs and two runs (one train of 5 epochs per run) for each model configuration.

1. Number of "large" hidden layers: $\{1, 2, 3\}$
2. Number of nodes in each "large" hidden layers: $\{600, 1200, 2400\}$
3. Number of nodes in second-to-last hidden layer: $\{50, 100, 200\}$
4. Number of nodes in second-to-last hidden layer: $\{10, 20, 50\}$

Appendix 2: MLP results with MSE loss

The KerasTuner tuner for the MLP with MSE loss returned the following layer sizes for the hidden layers: $[1200, 600, 600, 200, 10]$.



(a) (b) (c)

Figure 7: Validation results for the MLP trained on W8 cell data with MSE loss on (a) W9 cell data for capacity, (b) W10 cell data for capacity, (c) W5 cell data for capacity. Note capacity can go above 100%, since the cell started above nominal capacity.

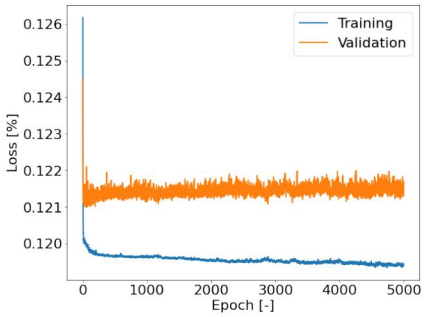


Figure 8: Plot loss as a function of epoch for MLP trained with MSE loss, exhibiting clear overfitting to the train data.