Machine Learning Engineer Nanodegree

Capstone Project
State-of-Charge Estimation with Neural Networks

Gergo Kiss
February 3rd, 2020

I. Definition

Project Overview

An alternative solution is applied for battery state-of-charge estimation by using data driven nonlinear models to compare their predictions with a modeless approach as coulomb counting. I develop software in a renewable energy source project where we measure batteries' voltage, temperature, SOC and some other stats. We are using a printed circuit board (later PCB) to measure the battery and control it by turning it on or off or in other words engaging it or bypassing it with the integrated electronics on the circuit board. The batteries can be connected through the PCBs thus the connected system can be used for EV car charging, peak load shaving on grids or simply as an energy buffer.

The batteries are individually controlled and measured which allows the user to set different DC voltage levels without DC/DC converters. Therefore, our technology is different than the usual BMS where the batteries only connected in series via power connection and only one measuring unit reads the overall voltage of the connected batteries. There are studies such as "Battery state of charge estimation using a load-classifying neural network" and "State-of-charge prediction of batteries and battery–supercapacitor hybrids using artificial neural networks" which are detailing the effectiveness of neural networks in the field. I had the chance to look at the first publication which details load profile based neural networks and I will use it as example for input selection, but I also tested our dataset with other neural net settings.

Problem Statement

To measure a battery SOC, you must fully discharge it until it reaches its lower voltage limit. At this point of time the battery has 0% SOC. The charging process starts, and a current measurement unit tells the PCB how much current runs through the battery. The PCB uses the current value and duty time of the battery to estimate the SOC by calculating the capacity which is gained or lost inside the battery. This modeless method is coulomb counting. Since no measurement can be perfect, this method suffers from long-term drift and lack of a reference point. Therefore, the SOC must be re-calibrated on a regular basis, such as by resetting the SOC to 0% when a charger determines that the battery is fully discharged. In this report, I will represent how a data driven non-linear neural network model can be applied for this regression problem.

1. Translate raw measurement data from ‘battery_data_1.xls’ and ‘battery_data_2.xls’ to ‘.csv’ file format excluding all unused parameters. Also calculate the time difference between measurements to replace raw timestamps. If prep. is done, import the datasets into python.
2. Try out and implement few of the following options for the solution model:
    a. I selected neural network models and I will show the optimization steps to reach high accuracy.
    b. decision trees, random forest or other regression models were not used.
3. Experiment with the chosen models by changing hyperparameters.
4. Train the selected model(s).
    a. Try different episode numbers, batch sizes and other training parameters.
    b. I added an extra step here which is to change the shape and the variables of the input data during training iterations because this turned out to be the most influential factor in the neural network performance.
5. Evaluate the model performance by using the chosen metrics.
    a. I choose to use mean absolute error instead of mean absolute percentage error due to the fact the SOC value can be 0 which can result division by zero. Therefore, checking the absolute difference between the predicted value and the reference seems to be the best metric for this application.
6. If necessary, select the best performing model and try to fine-tune it by making small adjustments.
7. Derive the conclusion.
8. Discuss future development potential.
9. Overall try to reach more than 90% accuracy if that seems to be not possible then discuss what can be missing from the dataset and include into future development.

Metrics

For simplicity and due to the lack of different SOC estimation results, the benchmark model has 100% performance. Therefore, the solution model will be compared to the benchmark model SOC values. Metric is chosen to be the mean absolute error in Equation 1 and the related accuracy in Equation 2. Since we are interested in how close the predicted value of the solution model to the actual value given by the benchmark model is.

$$Error = \frac{1}{n}\sum\left|SOC_{benchmark\ model} - SOC_{predicted}\right|$$

Equation 3 – Error function where n is the number of samples.

$$Accuracy = 1 - Error$$

Equation 4– Accuracy of a prediction

II. Analysis

Data Exploration

I will use 2 datasets. Both datasets are generated by our PCB measurements and by a laboratory power supply (during charging) or an electric load (during discharging). The first dataset includes 5 full charging cycles where only 1 battery is used. It is fully charged until it reaches the higher voltage cut- off limit and then it is fully discharged until it reaches the lower voltage cut-off limit (5 times).

The second dataset includes data about 5 partial charging cycles where the battery is part of a battery module which contains 24 batteries in total. In this scenario, the batteries are connected to maintain a constant voltage based on a topology selection algorithm. The topology of the engaged batteries is always changing during charge/discharge. Therefore, each battery spends time in turned off or turned on mode during the partial charging procedure. I will use the measurement data of only 1 cell to keep the dataset simplified for this project.

The first dataset called 'battery_data_1.csv' which includes approx. 165000 measurements. The second dataset called 'battery_data_2.csv' which includes approx. 107000 measurements.

The inputs originally were decided to be the voltage, status, temperature, current, mode and time. The output is the SOC.

INPUT						OUTPUT
U_b	B_E	T = [T_1,T_2,T_3]	I_m	mode	delta_time	SOC
Battery Voltage [V]	Battery Status Bypass = 0 Engage = 1	Battery Temperature [°C]	Current Measurement [A]	Charge = 1 Discharge = -1 Off = 0	Time between 2 log. [s]	State-of-Charge 100 Ah = 100%

Table 1 – Description of the dataset

During the experimentation with Neural Network models, I decided to augment the dataset to allow more various data input variations. Table 2 shows the extra and augmented input parameters. I also decided to round the measured voltage values up to two decimals, since the chosen datasets have no frequently changing values such as in pulse charging measurement data. Therefore, smoothening the voltage measurements with rounding the values can be beneficial for model training.

Extra INPUT		Augmented INPUT					
date	run_time	sample_id	actual_time	dV	dV2	dV3	C
DD-MM-YY HH:MM:SS	Always SUM of the previous delta_time values	running number for samples identification	Always SUM of the difference between the first date value and the current date value	Equation 3	Equation 4	Equation 5	Equation 6

Table 2 – Description of the added input parameters

The augmented data is created by using the already registered data values such as I_m, B_E, mode, delta_time.

$dV = U\_b_t - U\_b_{t-1}$ Equation 3 – dV is the difference between the current voltage and the previous voltage measurement.

$dV2 = U\_b_t - U\_b_{t-1} * e^{-delta\_time}$ Equation 4 – dV2 is 2nd derivative of the voltage difference in discrete terms.

$dV3 = U\_b_t - U\_b_{t-1} * e^{-\frac{delta\_time}{10}}$ Equation 5 – dV3 is modified 2nd derivative of the voltage difference in discrete terms.

$C = \sum delta\_time * I\_m * B\_E * mode$ Equation 6 – C is capacity where delta_time is different than duty_time. Because delta_time is the time between two data logging and duty_time is the time between two SOC calculation with coulomb counting. Thus, the resulted C values differ from the output SOC values.

<u>Data sample from dataset_2:</u>

```
sample_id,date,actual_time,mode,B_E,I_m,U_b,T_1,T_2,T_3,delta_time,runtime,dV,dV2,dV3,C,SoC

11,05-08-19 09:42:59,241885,1,1,33.300,2.94,24.1,23.9,23.9,0.67830,5.81861,0,0,0,0.047547876,0.054

12,05-08-19 09:42:59,241885,1,1,33.300,2.94,24.1,23.9,23.9,0.66024,6.47884,0,0,0,0.053822106,0.060

13,05-08-19 09:43:00,241886,1,1,33.300,2.94,24.1,24.7,23.9,0.71278,7.19162,0,0,0,0.05992931,0.067

14,05-08-19 09:43:01,241887,1,1,33.300,2.94,24.2,24.0,23.9,0.64484,7.83646,0,0,0,0.066522507,0.073

15,05-08-19 09:43:01,241887,1,1,33.300,2.94,24.2,24.0,23.9,0.67170,8.50816,0,0,0,0.072487262,0.079
```

Battery limits:

The limits of the battery are presented so the following Figure 1 and Figure 2 characteristics are easier to understand.

Lower Cut-off Voltage	Higher Cut-off Voltage	Nominal Voltage	Charge current MAX	Discharge current MAX	Standard current charge/discharge
2.5 V	3.7 V	3.2 V	200 A	300 A	33 A

Table 3 – Battery Charging Specification

The datasets contain only several current level values and the number of samples in each current level is different. Therefore, SOC value offset in the prediction can be partially resulted by the lack of multiple current levels in the datasets.

Exploratory Visualization

Both figures show the SOC output calculated by coulomb counting and the battery voltage U_b. As you can see on Figure 1 the SOC goes higher than 100%. It can happen because of two main reasons. Firstly, if the battery is new then its capacity can be higher than the nominal capacity which is 100 Ah here. Secondly the battery operates in a range of current levels during charge/discharge. Therefore, the energy loss during the charging procedure can vary and effect the battery capacity.

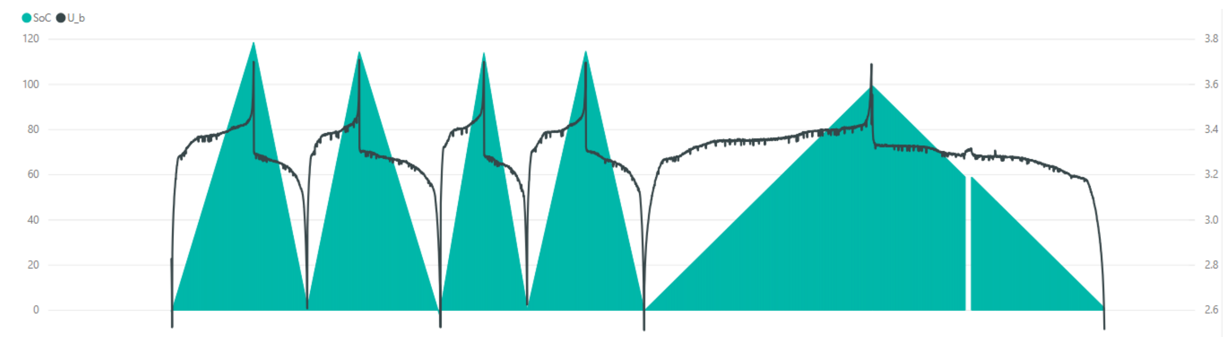


Figure 1 – [SOC, U_b] / runtime plot from 1st dataset, left vertical axis SOC, right vertical axis U_b

Figure 2 represents a different charging profile where the battery has rest time, so the 'time' input parameter either can be rest time if the status is 0 or duty time if the status is 1. In the first case the SOC should stay the same while in the second case SOC should increase or decrease based on the mode value.

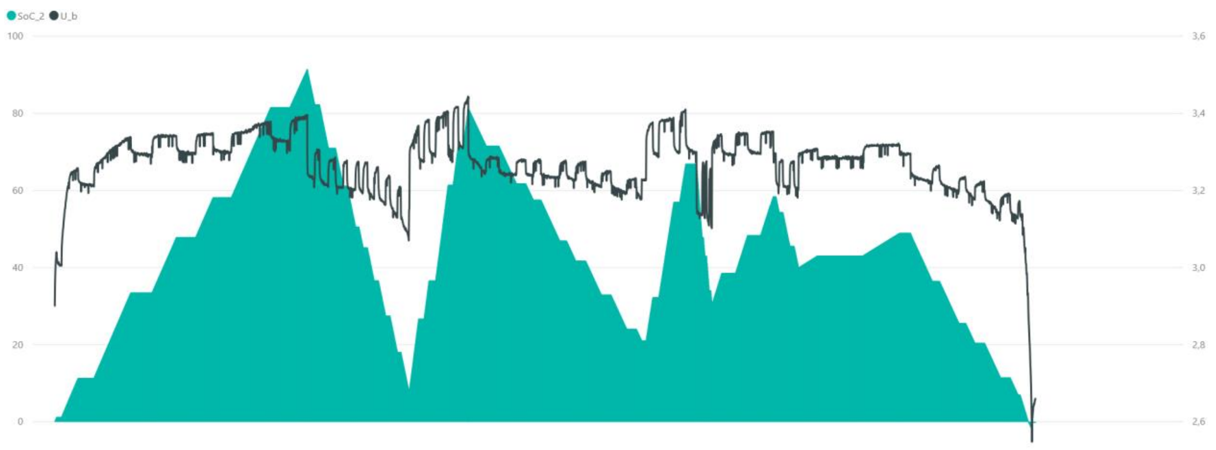


Figure 2 – [SOC, U_b] / runtime plot from 2nd dataset, left vertical axis SOC, right vertical axis U_b

I experimented with these two datasets by using one for training, the other one for testing and vice versa because firstly I was separately checking the two different solutions for the two datasets. Then I decided to use always 1 for training and the other for testing because the voltage value characteristics are different in the two datasets and this can be a good sensitivity test for the solution model.

Algorithms and Techniques

The best fitting algorithm is selected after multiple hundreds of iterations ran on a neural network class by changing the hyperparameters and the input shape and size in each class instance. The models were created in keras. The used parameters to find the best performing model are the following:

- Input X has 16 variation because the datasets can be used twice (once for training, once for testing) and 8 type of input value settings were chosen for overall performance evaluation.
    - Dataset_1 for training | Dataset2_for testing
    - Dataset_2 for training | Dataset1_for testing
        - x_15p: Use all input variables except date
        - x_9p: Use only I_m, U_b, T_2, delta_time, runtime, dV, dV2, dV3, C values
        - x_6p: Use only I_m, U_b, dV, dV2, dV3, C values
        - x_5p_1: Use only I_m, U_b, delta_time, runtime, dV values
        - x_5p_2: Use only I_m, U_b, dV, dV2, dV3 values
        - x_3p: Use only I_m, U_b, delta_time values
        - x_load: Use I_m, U_b, dV, dV2, dV3, C divided into 3 column groups based on charging mode of the battery. This input shape was inspired by [1].
            - 1st column group has 6 data columns for the 6 values, but every value is equal to 0 where **I_m ≥ 0**. This column group in the pandas data frame has only values if the battery is **charged**.
            - 2nd column group has 6 data columns for the 6 values, but every value is equal to 0 where **I_m != 0**. This column group in the pandas data frame has only values if the battery is **in standby or idle mode**.
            - 3rd column group has 6 data columns for the 6 values, but every value is equal to 0 where **I_m ≤ 0**. This column group in the pandas data frame has only values if the battery is **discharged**.
        - x_load_2: Use I_m, U_b, dV, dV2, dV3 divided into 3 column groups based on charging mode of the battery. Similarly, as above but without C parameter.
- Hyperparameters
    - Layer size of neural network
    - Number of nodes in the layers
    - Number of epochs during training
    - Batch size of training
    - Other parameters were considered and tested as well but did not stay as dynamic parameters.
        - Optimizers and learning rates
            - adam, sgd, rmsprop
            - adadelta has been selected as fix optimizer.
        - Batch normalization layers were unnecessary due to the normalized data.
        - Dropout layers did not show significant increase in model performance.
        - Regularizers also did not show increase in model performance.

The parameters were tuned several times before the final solution has been found. Therefore, some parameter settings might have missing performance values such as error and accuracy because the worst performances were always interrupted due to the long training time of neural networks.

Benchmark

The values of the ‘SOC’ column in the datasets are the results of my benchmark model called coulomb counting. The nominal capacity is 100 Ah. Simplified coulomb counting looks like the following.

```
if mode == 1:
    SOC += delta duty time [s] * measured current [A] / (3600 * nominal capacity)
if mode == -1:
    SOC -= delta duty time [s] * measured current [A] / (3600 * nominal capacity)
```

III. Methodology

Data Preprocessing

Data preprocessing is in the function called data_prep(data1, data2) in SOC_estimator.py where all the above mentioned X input is created by using dataset 1 and dataset 2.

1. Modify I_m where the new I_m = I_m*B_E*mode.
2. Separate X and Y (input and output).
3. Normalize data.
   a. I_m and mode ϵ [-1, 1]
   b. rest of the parameters ϵ [0, 1]
4. Create 8 variations of input X while both 1st and 2nd datasets are used for training and testing.
5. Input X data frame to numpy array
6. Return a list of x_train, y_train, x_test, y_test arrays.

The various data input resulted great results but also some weird neural networks which tries to fit the voltage characteristic on the SOC curve. All results and their identification will be discussed in Results section.

Implementation

The implementation started with transfer learning where I used a predefined keras neural network model from a previous Udacity project to start with. Then the raw measurement data of batteries was fed to the model which resulted poor predictions. Normalization and augmentation of data were necessary. During this early phase of experimentation, I also found out the batch size ‘sweet spot’ is around 500 because the data include multiple measurement batches which barely changes in values if the size is not big enough. After the data structures were defined and few pre-runs of rudimentary models have been executed, I started to log performance and plot the predictions. The following step-by-step list describes the implementation:

1. Load data from csv.
2. Data preprocessing with data_prep(data1, data2).
3. Describe hyperparameter settings.
4. Create data containers for error, accuracy, error standard deviation.
5. Loop through all hyperparameters.
   I. Define neural network parameters.
   II. Train model.
   III. Validate model by predicting SOC values of x_test.
   IV. Calculate mean absolute error, error_std and accuracy
   V. Save overall performance plot and predictions plot

Refinement

The improvement along the implementation was the gained knowledge about which hyperparameters do not affect the model performance so those can be left out. The usage of different inputs also was a big improvement since the chosen parameters and augmented values in the input had significant role in the overall good performance of the final solution candidates.

The implementation of the different parameter settings can be split three groups:

d1, d2 = dataset1 for training, dataset2 for training

1. **Neural net ID 1 – 80**

   X = [d1:[x_15p, x_9p, x_5p_1, x_5p_2, x_3p, x_6p, x_load, x_load_2],
   d2:[x_15p, x_9p, x_5p_1, x_5p_2, x_3p, x_6p, x_load, x_load_2]]

   - Input X has 16 variations but only d1:[x_15p, x_9p, x_5p_1 and part x_5p_2] ran because, x_5p_1 and x_5p_2 preformed poorly and the software was interrupted.
   - hidden_layer_size = [1, 2]
   - number_of_nodes = [[18,12,64], [36,24,24]] which means 2 types of node selection will be tried out by the model.
     i. If hidden_layer_size = 1 then only the first node number applies 18 or 36.
     ii. If hidden_layer_size = 2 then the first and second node numbers apply 18 or 36 and 12.
     iii. If hidden_layer_size = 3 then all node numbers apply which only was used in the rudimentary model but here max 2 hidden layers were used.
   - number_of_epochs = [50, 150, 250]
   - batch_sizes = [1000, 500]
   - optimizers: adadelta

   All the performance results of this part were overwritten by the second implementation phase. Therefore, only the plots can be used to define the best neural net ID and then the parameter settings must be reverse engineered based on the software for loop structure.

   ‘41_trend.png’ and ‘41.png’ shows that the neural network with 1 hidden layer, 18 nodes, with batch size 1000 during 250 epochs which is trained on x_9p input is the best performing model when dataset 1 is used for training and dataset 2 for testing. The accuracy was more than 97%.

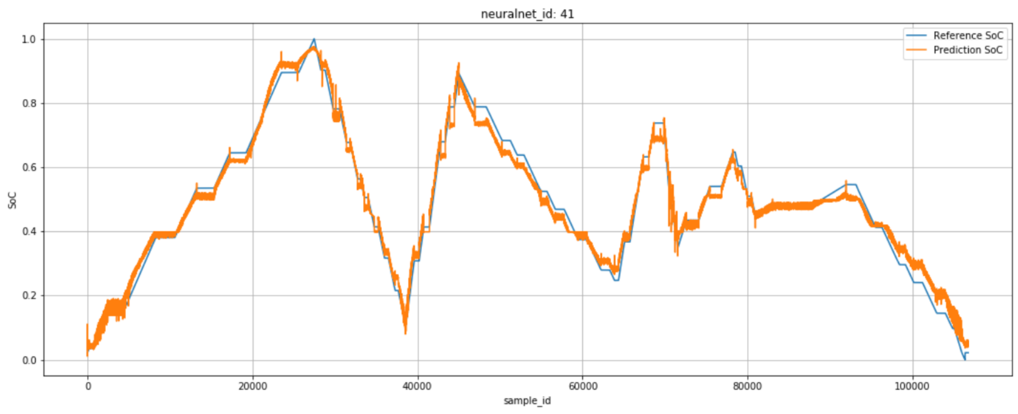


Figure 3 – SOC / sample_id plot, where prediction is more than 97% (snippet from ’41.png’)

2. **Neural net ID 100 – 193**

   X = [d1:[x_load, x_load_2], d2:[x_load, x_load_2]]

   - x_load_2 was preforming poorly therefore the software was interrupted 1 last time.
   - no significant change according to models where x_15p and x_9p are used.
   - I still wanted to check the performance of the models with x_15p, x_9p but when d2 used as training set.

3. **Neural net ID 200 – 264**

   X = [d2:[x_15p, x_9p, x_3p], d1:[x_15p, x_9p, x_3p]]

   - X starts with d2's x_15p, x_9p because these data inputs resulted more than 90% accuracy for the models (original project goal was reached). Also, if the x_3p proves to be a bad input selection then the software can be interrupted again.
   - Software was interrupted at x_3p. At this point I already had the performance of the models for the best parameter settings.

Multiple models reached close to 97% accuracy and the best turned out to be neural net ID 213 which is a neural network with 2 hidden layers, 18-12 nodes, with batch size 500 during 50 epochs which is trained on x_15p input. In this case dataset 2 was used to create train data and dataset 1 to validate and test models.

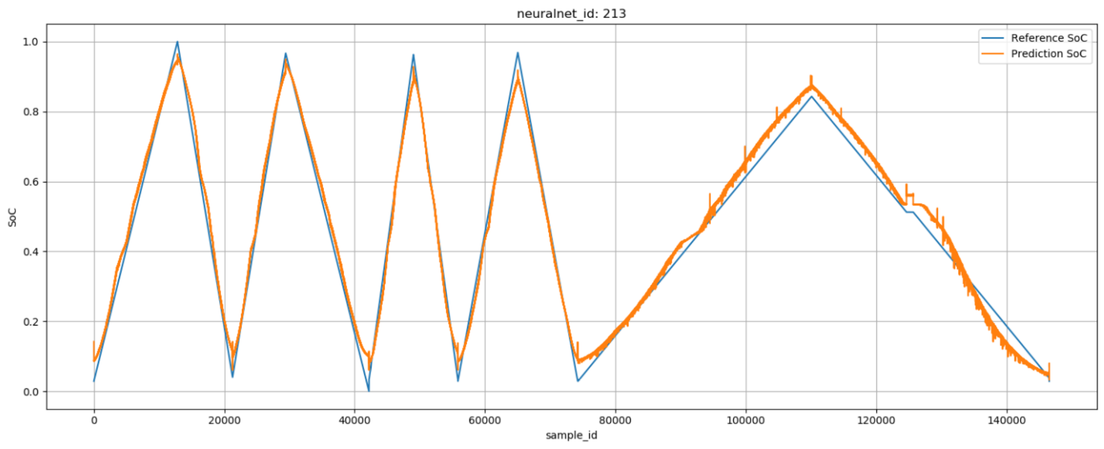


Figure 4 – SOC / sample_id plot, where prediction is 97% (snippet from '213.png')

The neural net model at ID 41 is identical to ID 228 which had 96% accuracy so in overall the best model properties are the following:

- ✓ Hidden layer size: 1
- ✓ Number of nodes: 18
- ✓ Number of epochs: 250
- ✓ Batch size: 1000
- ✓ Optimizer: adadelta
- ✓ Input: x_9p

This algorithm outperformed even the load-classified neural networks, so it was selected as final solution.

IV. Results

Model Evaluation and Validation

The final model was chosen because it reached more than 96% accuracy in both times when the model was trained on dataset 1 or 2. The model was tested on 146565 + 106710 data samples which could indicate that the model is robust enough but it would be also interesting to see how the model would perform if it is trained on pulse-charging data which is unavailable because we did not measure the battery characteristics under quickly fluctuating current values yet. The final model reasonable and aligning with solution expectations which was to reach at least 90% accuracy. The model and input base could be further approved but I will discuss that under Improvements.

Performance tested on data 2:

data\plot\trend\41_trend.png:

- Accuracy is 97.4%
- Mean abs error is 2.6%
- Error's stand. deviation is 1.6%

Performance tested on data 1:

data\plot\trend\228_trend.png:

- Accuracy is 96.2%
- Mean abs error is 3.8%
- Error's stand. deviation is 3.4%

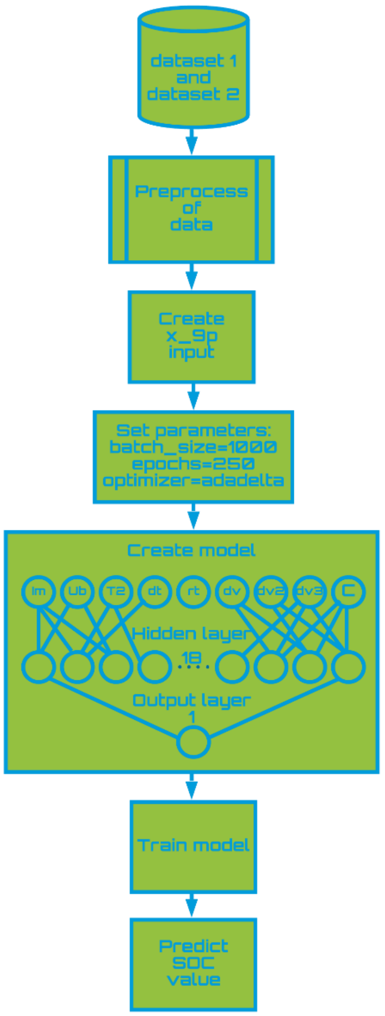


Figure 5 – Final Model

Justification

The final solution results are should not be better than the benchmark solution only if there will be data in hand which is generated by better SOC measurement approaches than discretized coulomb counting. Because the model is data driven and the used data was generated by our own system. The goal was to get close as possible to 100% prediction accuracy which was met. Basically, the model was forced to understand coulomb counting by getting C as an input parameter.

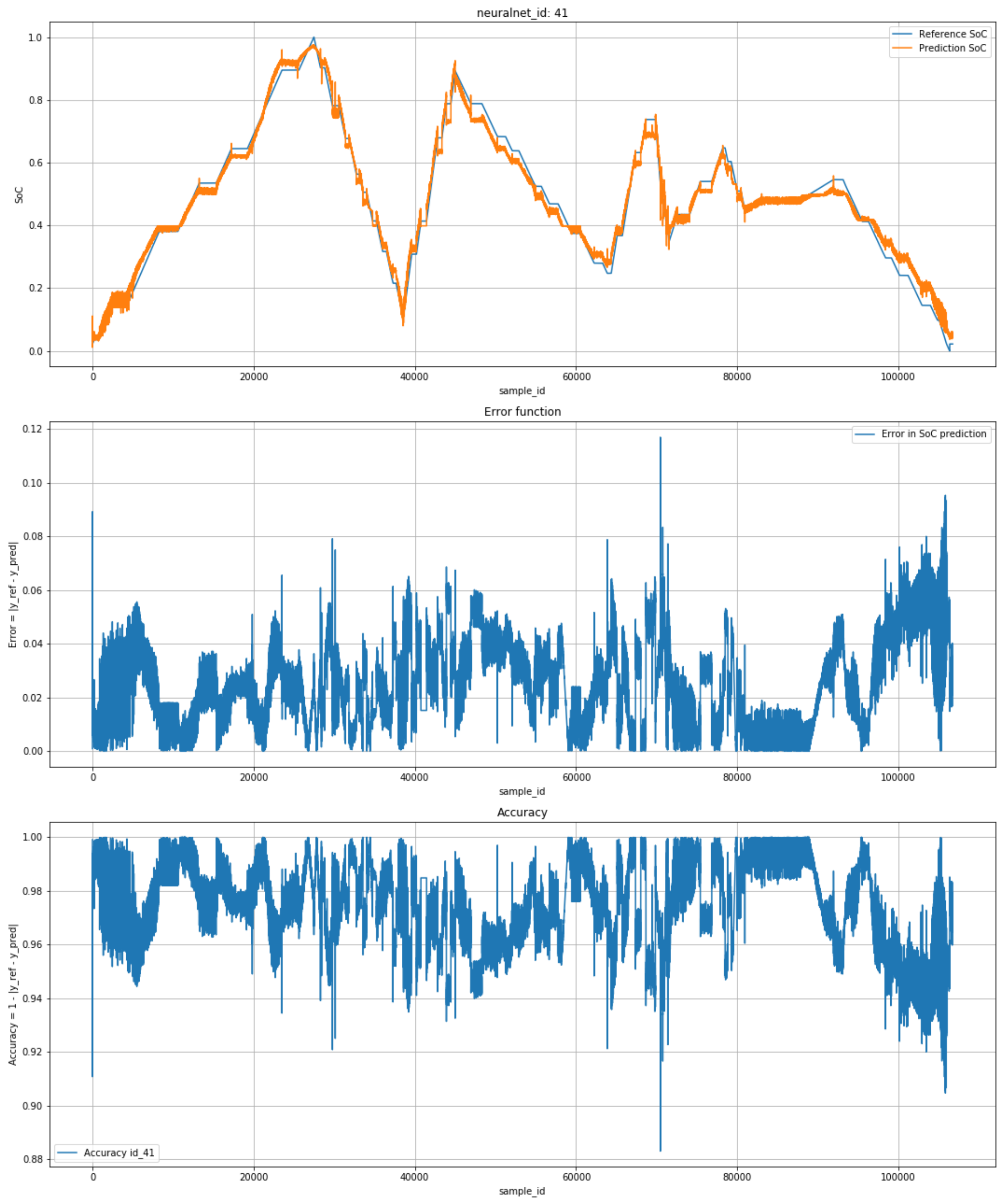
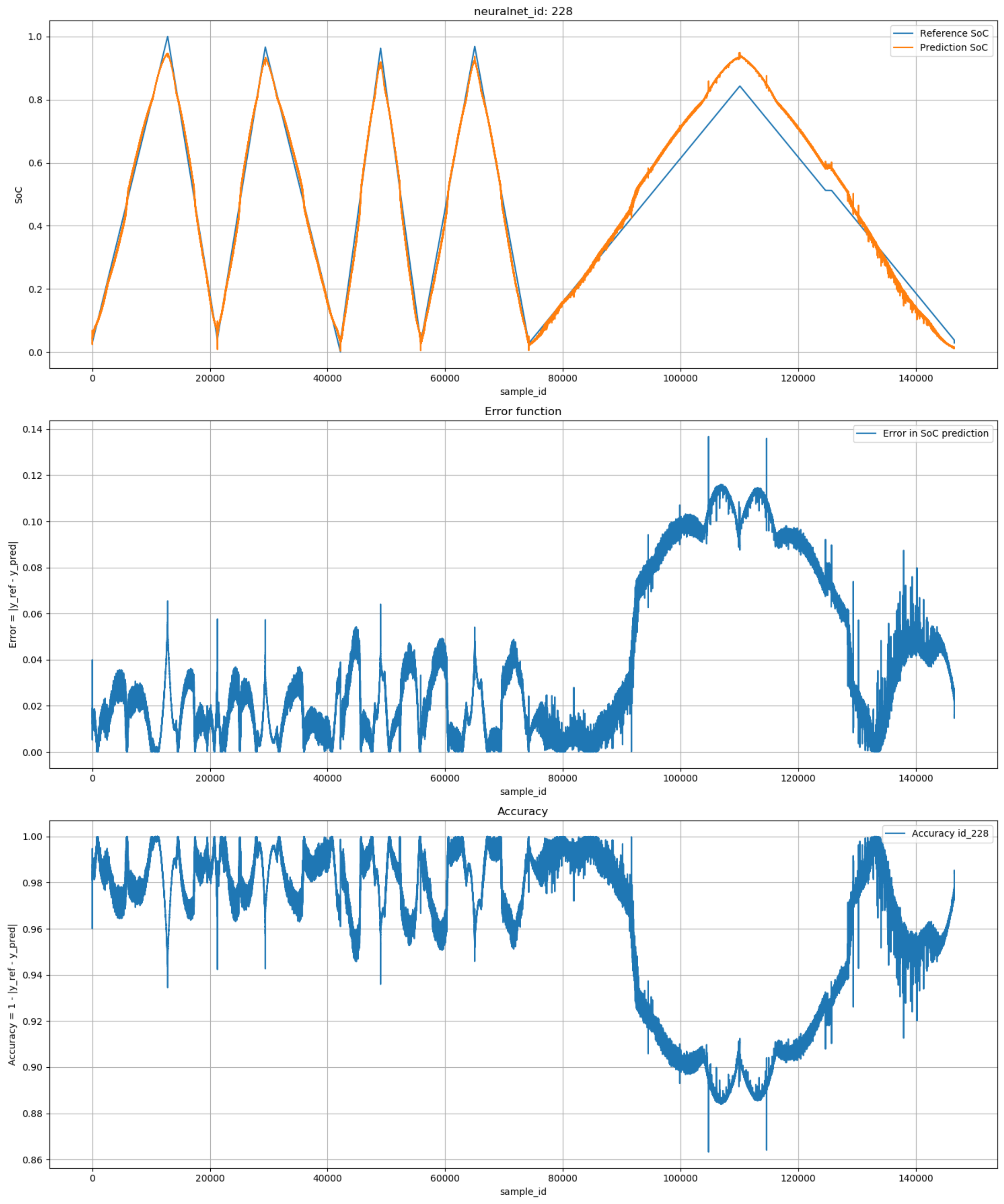


Figure 6 – data\plot\ 41.png Predictions, error, accuracy of final solution

Figure 6 shows the best result where the final solution was validated by predicting values of dataset 2. It preformed good because the training data had samples on the full scale of SOC values. The predictions were more undershooting which is much safer for the battery and its lifespan.

The big error at the last charging cycle on Figure 7 is resulted because the training data only has 10A charging information between 40-50% SOC. The last cycle is at 10A charge level therefore above 50% the model is slightly unstable.



\

V. Conclusion

Free-Form Visualization

The following Figure 8 shows that how significant is to have appropriate input selection, otherwise the neural net will predict random characteristics. The black-box behavior of the model could be a drawback of using neural networks if the used input is inadequate for the regression problem.

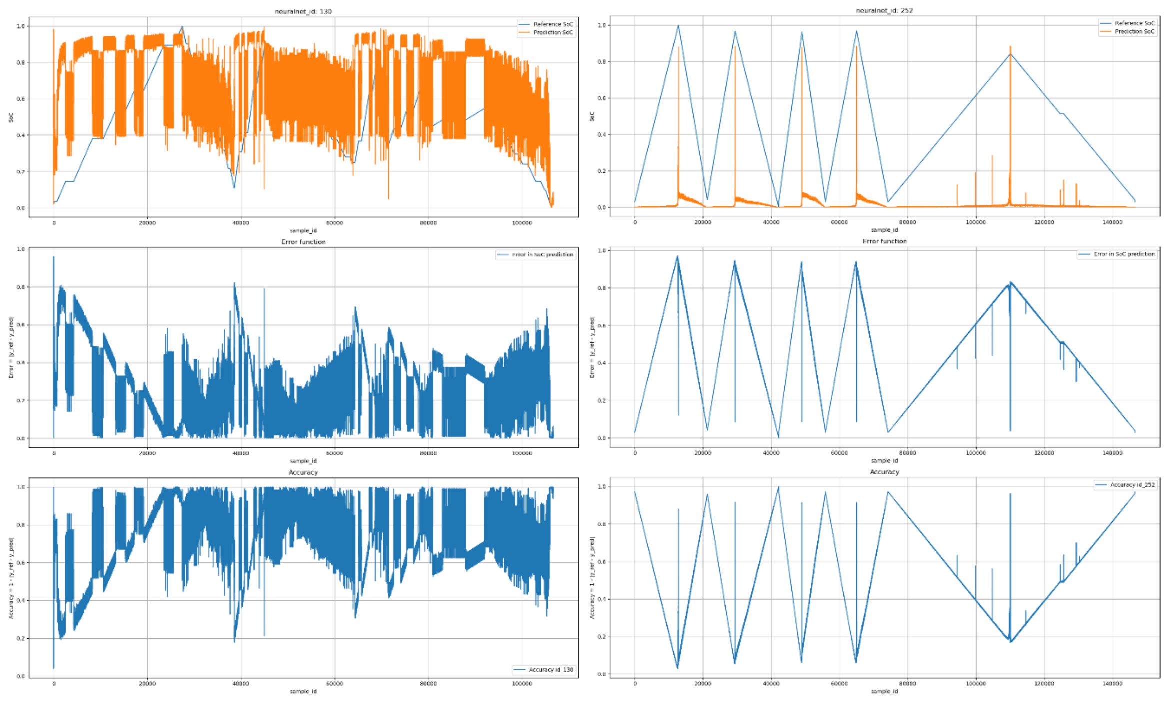


Figure 8 – Weird SOC Predictions, error, accuracy of x_load2 and x_3p based models

Reflection

I wanted to see if I can teach a neural network to predict battery State-of-Charge values.

1. The solution Selection of the used data sets were a smart choice because the models can be tested against different SOC characteristics.
2. Augmenting the dataset was necessary to get good results.
3. Figuring out which data input variations were the best performing was difficult and interesting.
4. The nested for loop of hyperparameters was generating multiple parameter settings for train.
5. Models had been evaluated.
6. Final solution was found.

The solution will be used later to predict other datasets too. My concern is the model will preform poorly under those SOC characteristics which were not covered during training in this project.

Improvement

The solution can be improved in several ways but the most influential change would be a larger more various data set for training and for testing.

- ✓ The biggest optimization could be if a highly accurate battery management system would be used to measure SOC (instead of coulomb counting based on PCB meas.) in a controlled environment with fix temperature. This setup would result a model which predicts SOC without accumulated errors where temperature could be introduced as new significant feature in SOC prediction.
- ✓ The training data should be more various in current level values. The frequency of current value change should be also increased to register extremist voltage variations during SOC calculation. F.e.: pulse-charging data. It would be interesting to see the effects of more dramatic voltage changes on the SOC prediction.
- ✓ Other regression models should be tried for comparison.