Modelling State-of-Health for a Li-ion Battery

Karthik Nataraj, Hampus Carlens and Julian Cooper

1 Motivation

Lithium ion batteries are of great and increasing importance in today's society due to their high energy density. Li-on batteries' performance capability can be characterized by their State of Health (SOH). SOH is a measure of usable capacity over rated capacity. Accurately predicting how many cycles (i.e charge then discharge) a battery can perform, at any given time, before reaching it's end of useful life is important for reliability. However, prediction is notoriously difficult due to poor understanding of how the measurable parameters (voltage, current, temperature, etc.) effect the SOH. Today three main methods are used: electrochemical, equivalent circuit, and data-driven models.

There have been many previous studies into data-driven models for SOH prediction. Some of the most cited recent publications are [6], [4] and [2], all of which produce predictors for RUL with Mean Absolute Percentage Error (MAPE) ranging from 2-15%. The lowest errors are obtained using deep neural networks such as in [4] and [2], with MAPE of 2%. Multiple authors (including [2] and [5]) point towards the importance of future work in trying to scale these models, to make them suitable for front-end embedded systems. In [6] the authors opt for a simple linear model over neural networks. Given information from the first 100 cycles only, they manage to achieve MAPE of 13% for RUL predictions.

In this project, we develop a scalable (and explainable) white box model to predict the SOH curve until end-of-life threshold (as opposed to a point estimate of RUL) given the first 100 cycles. Since our predicted SOH curve implies an RUL prediction, we will also compare our implied RUL prediction error to that of previous studies.

2 Data & Methodology

2.1 Battery cycle data

The dataset we have selected contains approximately 96,700 cycles (approx. 780 cycles per battery for 124 batteries). For each cycle the authors captured voltage, applied current and temperature sampled at 2.5 second intervals. This is the largest publicly available dataset for identical commercial lithium-ion batteries cycled under controlled conditions, and is the same source used by two recent papers that motivated our project: "Machine learning pipeline for battery state-of-health estimation" (Nature, 2021)[4] and "Data-driven prediction of battery cycle life before capacity degradation" (Nature, 2019)[6]. See Appendix 5.1 for exploratory data analysis.

2.2 Evaluation criteria

Our goal is to build a model that takes information from the first 100 cycles and predicts State-of-Health (SOH) for the remaining cycles until end-of-life threshold (80% of nominal). For all model variants, we used 70% of the data for training, leaving 30% for out-of-sample testing. Given the small number of total observations (124 batteries), we opt for cross-validation for tuning hyperparameters to avoid further reducing our training set.

To evaluate out-of-sample performance we use two error metrics:

- State of Health Curve (SOH): Compute Mean Square Error (MSE) for predicted SOH values, one value per cycle for each battery used in error calculation.
- Remaining Useful Life (RUL): Compute Mean Absolute Percentage Error (MAPE) for predicted RUL values (or equivalently when SOH goes below 80% of nominal capacity), one value per battery used.

While these error metrics are related (RUL = first cycle for which SOH dips below end-of-life threshold), they do measure different things. For example, we might correctly predict the number of cycles before end-of-life (let's say 10,000), but guess

CS229: Machine Learning, Autumn 2022, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

that the path is linear instead of parabolic or logistic. This might mean that at 50 cycles our prediction of available capacity (SOH) is much worse than the true value despite good RUL accuracy.

2.3 Model development

Our journey towards an effective white box predictor for the SOH curve can be understood as a progression through three model variants.

1. Neural Network: Predicts RUL given information from the first 100 cycles. We built this model to convince ourselves that the model performance from [6] was reproducible (and equally that we were interpreting their engineered feature variables correctly), and perform feature importance analyses to identify which variables might be most valuable for SOH prediction in subsequent models.
2. Time Series Models: Our first attempt to produce a white box model of the SOH curve, rather than RUL point prediction. We tried auto-regression and exponential smoothing methods, but both suffered from the same issue: we could not effectively "pool" data across batteries, leading to poor performance.
3. Bayesian Inference Model: Our second idea for building a white box model of the SOH curve involved imposing the known physics of our problem on our model specification. We know that battery discharge capacity has an exponential decay relationship with cycle number, and a fixed y-asymptote at the nominal capacity. Therefore, we can impose a functional form that reflects this physical behaviour and only ask our model to learn the unknown parameters of this function.

3 Experiments & Discussion

The following sections discuss the design choices and results from each of the models we developed in pursuit of a white box predictor for the SOH curve.

3.1 Neural Network

Our Neural Network model predicts RUL of a battery given data features derived only from its first 100 cycles. The purpose was to improve results from [6] and perform feature importance analsyses to inform subsequent time series and Bayesian Inference models.

During development of our Neural Network model we needed to make several design choices, including (a) hyperparameter tuning and (b) feature selection.

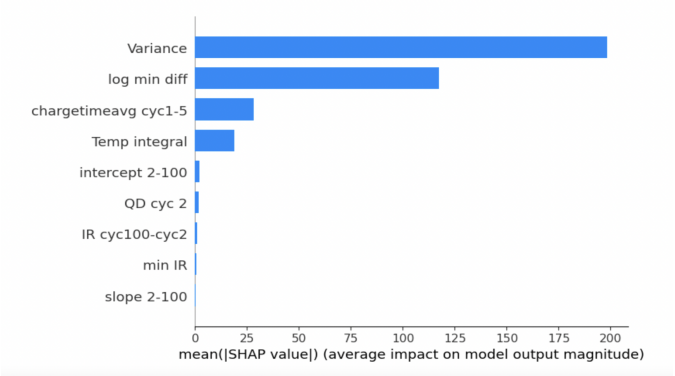
(a) **Hyperparameter selection**: The 2-layer feed-forward, densely connected neural network proposed in [4] was successful, so we used that and initially performed a random search hyperparameter tuning experiment on the learning rate (sampling from .01, .001, .0001) and number of neurons in each of two hidden layers (first layer ranging from $\approx 500 - 2000$ nodes, second from $\approx 0 - 500$). We used early stopping based on validation MSE loss, with 50% of training data used for validation. The learning rate range seemed reasonable as per the experiment in [1] and ranges for the nodes of hidden layers covered a reasonably wide span, and further would result in an architecture similar to that proposed in [4].

The goal of this experiment was to see how helpful tuning would be on a problem with this little data, before launching a full gridsearch over these and more hyperparameters (like number of layers and mini-batch size, for example). Our initial feature set consisted of the original 20 features proposed in Supplementary Note 1 of [6], along with average temperature and current over the first 100 battery cycles.

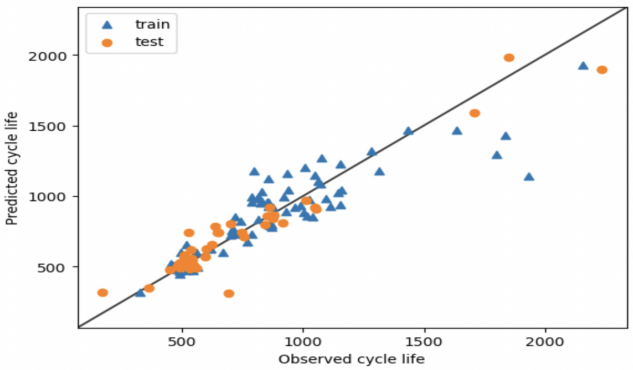
However, since $n = 81$ was so small the optimal parameters that the tuning algorithm selected were not optimal on the test set, in particular the learning rate chosen by the tuning procedure was .01 while a rate of .001 performed almost 5% better on the test set. Therefore further tuning would not be helpful and we used the network architecture determined by the preliminary tuning, consisting of 2 hidden layers, first having 1076 neurons and the second 96. We then used the learning rate of .001 with the Adam optimizer, ReLU activation, and full-batch gradient descent (since the training dataset is already small).

(b) **Feature engineering** The model with the full feature set converged very slowly and had poor validation performance even with l1/l2 regularization. So instead of letting the model eliminate the noisy features, we condensed the original feature set ourselves using Shapley values, following the work in [3], to derive a less noisy model based on the 3 features with the highest mean feature importance scores (see (a) of 1). This analysis serves the dual purpose of improving our prediction accuracy, while also informing the Bayesian Inference Model (Point 3 of 2.3) later on, which requires very few variables in order to train in a reasonable amount of time.

Model Evaluation Our final model yielded a MAPE of $\approx 10.8\%$, outperforming the best model in the paper which obtained a MAPE of 13%. (b) in Figure 1 below visually shows the predictions vs. actuals:



(a) Top 9 feature importance scores for non time-series variables; Refer to [6] for variable formulae



(b) Predicted vs. Actual cycle lives

Figure 1: Descriptive plots for neural network approach to predicting

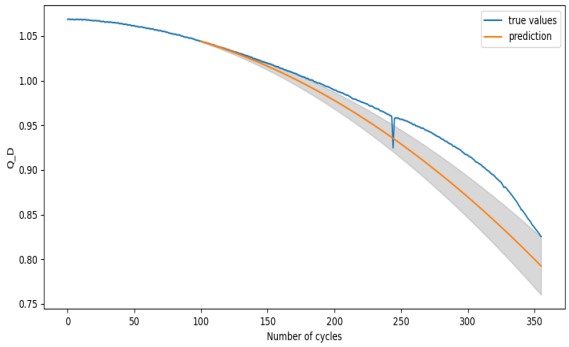
3.2 Time Series Models

The purpose of using time series models was to predict the entire capacity degradation curve, not only the RUL as with the neural net. We experimented with both Exponential Smoothing and Auto-Regressive Integrated Moving Average (ARIMA) time series model variants. Of these two, ARIMA was the most promising variant, so this report focuses on those results.

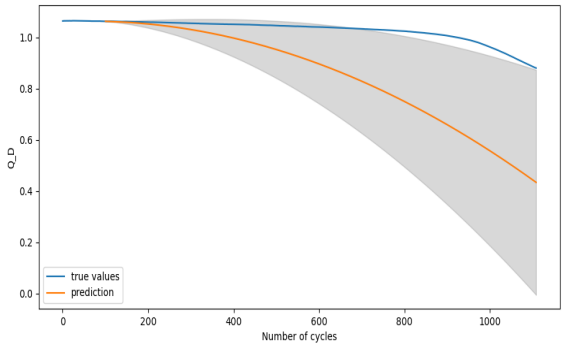
There were several design choices made during model development, including (a) choice of hyperparameters, (b) varying number of initialization cycles, and (c) use of exogenous variables.

(a) **Choice of hyperparameters**: The ARIMA model consists of three parts corresponding to three defining parameters, p, d, q, where p = number of lags used, d = degree of differencing needed to make the data stationary, and q = order of moving average. We used the Box-Jenkins method with battery degradation curves from our training set to derive p (partial auto-correlation), d (differencing required to make stationary) and q (auto-correlation). We then sense checked these with results from auto-arima[7] which performs an automatic search for the best parameters based on an information criterion (AIC). We found that p=2, d=2, q=2 worked well, and in particular that our results were very sensitive to d being at least 2.

(b) **Exogenous variables**: ARIMA models can be used with or without exogenous variables. An exogenous variable is in our case a parallel time series of known values that can be imposed on the model/used as a weighted input. In the ARIMA model without exogenous variables we only considered one measurement per cycle, namely the discharge capacity Q_D. We tried adding exogenous variables to the ARIMA model, however, since the battery life cycles were of different lengths it was hard to consistently apply exogenous variables in training and testing. One approach we experimented with using exogenous data from our longest cycle life battery in the training set. However the results were mixed, particularly for shorter cycle lives. A future study could tackle this more rigorously with regressors with a normalized cycle life scale.



(a) ARIMA, 100 cycles, on b2c3, great fit



(b) ARIMA, 100 cycles, on b3c3, bad fit

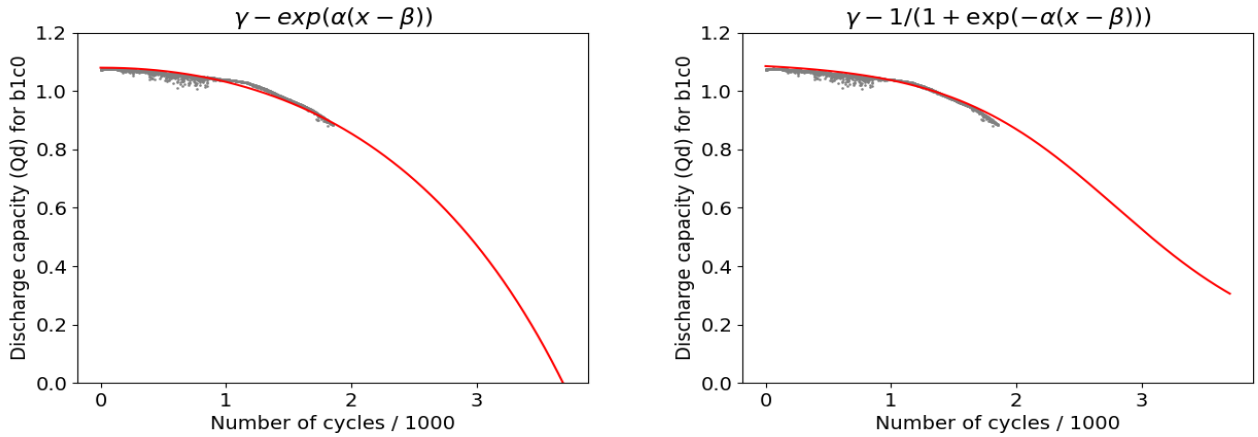
Model evaluation: Using ARIMA(2,2,2) and intializing each model with the first 100 cycles only, we achieve a MSE of 0.049 for SOH prediction and MAPE of 26.5% for RUL prediction. An example of a very good and a very bad fit is presented in figures 2a and 2b. Figure 2a performs so much better than 2b because the first 100 cycles already indicate the decay trend that will follow.

3.3 Bayesian Inference Model

We explored Bayesian Inference as a way to impose more structure (known physics) on the problem. Main idea: since we know the discharge capacity curve of each battery must be a decay curve, why don't we specify such a functional form and only ask our model to learn the shape and translation parameters. Another anticipated benefit of this approach was that we would be able to effectively pool (and learn from) data across batteries - a significant limitation of our ARIMA models.

In constructing our Bayesian Inference model we made a number of design choices, including (1) functional form, (2) parameterization and (3) selection of priors.

(a) **Function form**: We investigated two different functional forms that described the physical behaviour we would expect in our region of interest: (a) shifted exponential decay and (b) inverse sigmoid.



(a) Shifted exponential decay

(b) Inverse sigmoid

Figure 3: Comparison of functional forms fit to data for an example battery cell

While the exponential decay curve seemed like a more natural fit (no inflexion point where curvature reverses), we found that the inverse sigmoid also described the data well in our region of interest (y = 0.8 to 1.2) and its parameters were more interpretable. For example, in our inverse sigmoid formulation, α corresponds to shape (or rate of decay), β corresponds to horizontal translation of our sigmoid midpoint, and γ is our y-asymptote (which we can set equal to our nominal discharge capacity for each battery). In contrast, for exponential decay, changing α or β will both affect shape and horizontal translation.

(b) **Parameterization**: Having selected a functional form, we then needed to decide how to relate our function parameters α (rate of decay), β (translation) and γ (y-asymptote) to our data features (inspired by neural network variable importance from Section 3.1).

- x_1: nominal discharge capacity (= Qd after first cycle)
- x_2: variance between cycles 10 and 100 of Qd difference as a function of voltage
- x_3: log of magnitude of the minimum of the Qd difference
- x_4: average charge time for cycles 2 through 6
- x_5: sum of average temperature for cycles 2 through 100

We choose linear models for α (= $a^\top x$) and β (= $b^\top x$) based on the effectiveness of the linear model from [6] for predicting RUL (closely related to translation in our case). Then for γ we further restrict our formulation by specifying that the y-asymptote must equal the nominal discharge capacity (x_1).

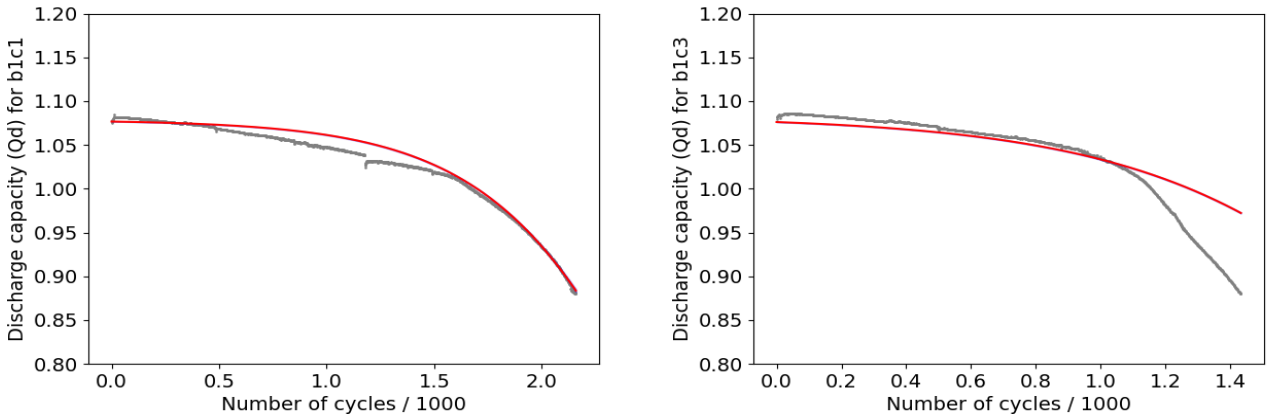
(c) **Prior specification**: We make an assumption that our labels y (discharge capacity for each cycle for each battery) are generated from a normal distribution with mean $\hat{y}$ (our predictions based on learned parameters) and variance σ^2 (which we learn as a parameter).

$$y \sim \text{Normal}\left(\gamma - \frac{1}{1+\exp(-\alpha(x-\beta))}, \quad \sigma^2\right)$$

For our model parameters, we impose informative priors on a_0, b_0 based on aggregate analysis of our data: $a_0 \sim N(3,1)$ and $b_0 \sim N(2,1)$, and weakly informative (standard normal) priors on the remaining a_i, b_i parameters since we did not have pre-existing intuition for these relationships. Finally, for variance, we impose a more traditional gamma prior $\sigma^2 \sim \text{Gamma}(1,2)$.

To sample from our joint posterior, we use the Hamiltonian Monte Carlo No U-Turn sampling method (Stan in-built). See Appendix 5.2 for detailed analysis of posterior marginals and sampling efficiency.

Model evaluation: On out-of-sample test data, our model achieves MSE of 0.015 for SOH prediction and MAPE of 33.5% for RUL prediction. While the curve fit is an improvement on our time series models, the RUL is worse.



(a) SOH MSE = 0.0008, RUL MAPE = 1.5%

(b) SOH MSE = 0.022, RUL MAPE = 38.5%

Figure 4: Predicted discharge capacity over normalized cycle life

Our model predicts well for batteries with mid-long cycle lives (Figure 4a), but often overshoots for short cycle life batteries (e.g. Figure 4b). This is partly due us having so few short cycle life training observations (variance), but also may be indicative of inflexibility in the model parameterization (bias). For example, our linear model for the shape and translation terms might be too simplistic. Examples like Figure 4b disproportionately impact the RUL MAPE performance relative to our ARIMA model.

4 Conclusions

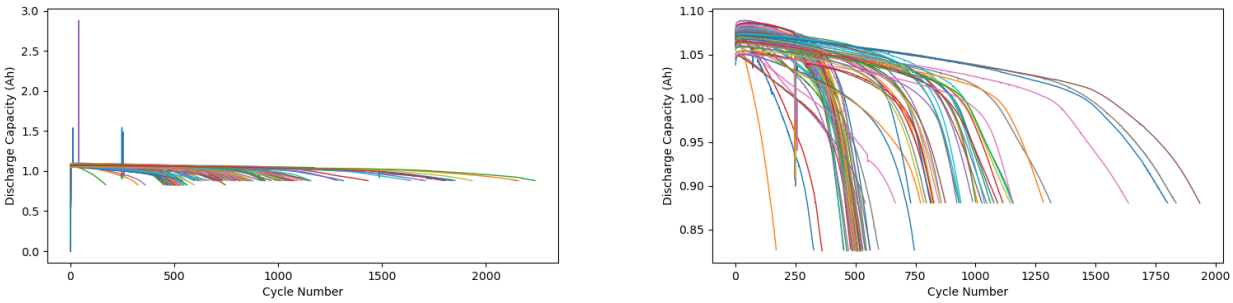
In general, while we were able to confirm that a relatively simple model can predict RUL accurately (MAPE $\approx 11\%$), finding a white box model to predict both RUL and the SOH curve proved much more difficult. This is likely due to the error propagation in predicting the decay curve, which causes predicted RUL (the predicted decay curve endpoint in this case) to either over or undershoot drastically. Our Bayesian and ARIMA models were each best on different metrics: Bayesian for SOH MSE (0.015 vs 0.049) and ARIMA for RUL MAPE (26.5% vs 33.5%).

There are a few areas of future study that would be interesting to pursue as next steps. First, given the partial success of the ARIMA models, we could try using a Long Short Term Memory (LSTM) Neural Network to see if we can improve predictions for the decay curve. This would no longer be a white box model that could be used in online systems, but might still be useful for offline use cases. Second, our ARIMA models struggled when the first 100 cycles were flat, which suggests we could improve by adding well-chosen exogneous variables that encode an downwards trend (or inflexion point). Given the varying number of cycles for each battery prediction, any exogneous variable would need to be flexible in length and therefore model generated (e.g. regressors). Third, one idea for improving our bayesian formulation would be to refine the parameterization for shape and translation of our inverse sigmoid. We currently use a linear model, but a deeper understanding of the physics involved might lead us to specify something more complex (e.g. with interaction terms).

5 Appendix

5.1 Data exploration

Discharge capacity vs cycle number. Our goal is to predict the discharge curves below given information from the first 50-100 cycles. A few things to note. First, the batteries in our population are either rated at 1.1 Ah or 1.05 Ah nominal capacity. By nominal here we mean manufacturer rated initial capacity. Second, in practice, our initial discharge capacity rarely perfectly matches the nominal capacity rating and so instead of two discrete starting points at 1.1 and 1.05, our starting points range continuously in that range. Third, our data is meant to reflect cycling each battery from its initial capacity to 80% of its nominal. This explains the two distinct end points we observe in the right hand chart at 0.88 Ah and 0.84 Ah, the 80% thresholds of 1.1 Ah and 1.05 Ah nominal capacities respectively. Last, in plotting these curves we identified some obvious outliers. Batteries in the left hand plot with capacities either above 1.2 or below 0.8 are erroneous measurements that we remove from the data before training.



(a) Complete dataset

(b) Outliers removed

Figure 5: Discharge capacity by cycle number for 124 batteries

Distribution of our target variable. Cycle life is the number of cycle it takes for a given battery to reach 80% of its nominal capacity. This is our target variable. When plotting the complete dataset we identified a skewed normal distribution. Ideally we want to roughly maintain this distribution for our validation and test data. To achieve this we re-used logic from "Data-driven prediction of battery cycle life before capacity degradation" [6] to split our train, validation and test data.

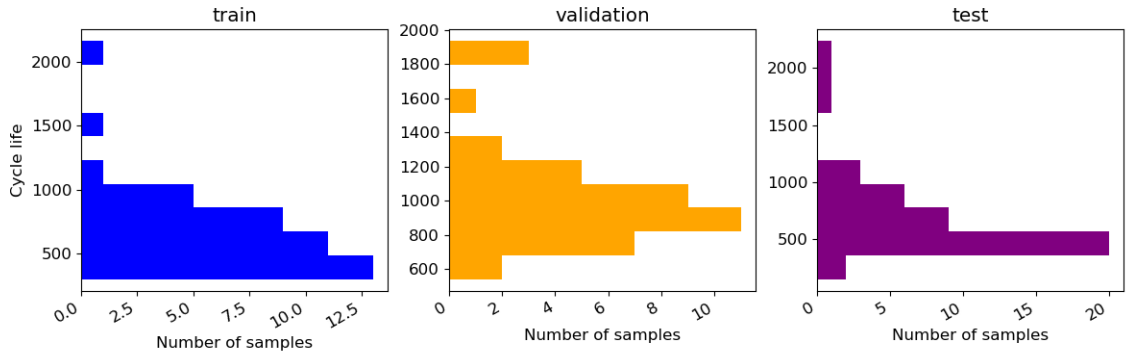
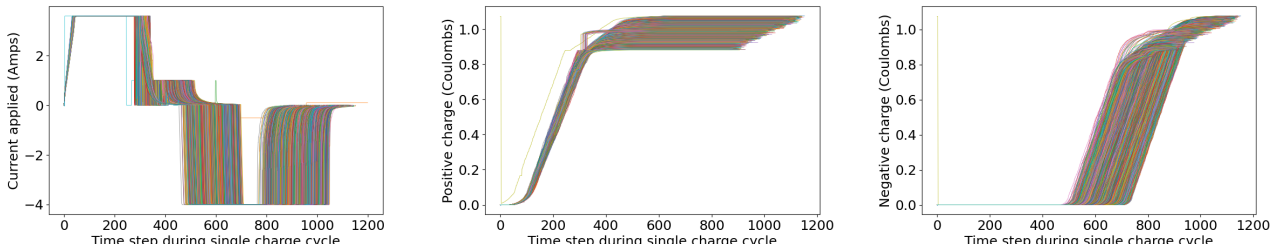


Figure 6: Distribution of target for train, validation and test data

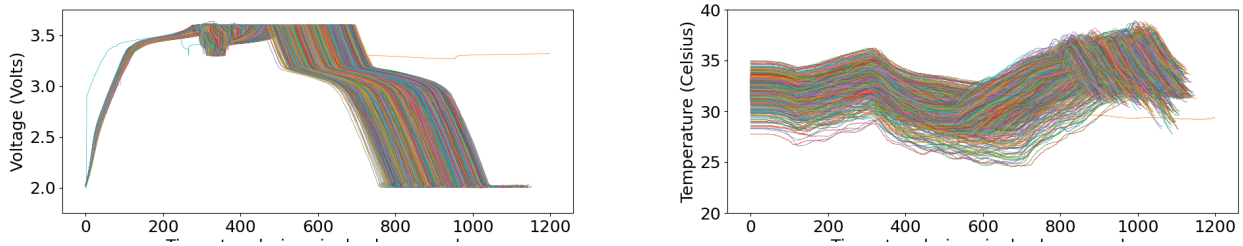
Applied current and charge cycle. The charts below illustrate current, charge capacity, and discharge capacity for all cycles of an example battery data point (ref: b1c0). We can immediately confirm that the vast majority of cycles follow similar charge profiles. In particular, applying positive current (charging) for first 500-700 time steps, then switching to an applied negative current (discharging) until depleted at the end of the cycle. The charge capacity (Qc) and discharge capacity (Qd) charge plots also show this transition, with Qc increasing up until the changeover to negative applied current, and Qd increasing only after the changeover.



(a) Applied current (b) Charge capacity (Qc) (c) Discharge capacity (Qd)

Figure 7: Charge and discharge cycles for an example battery (ref: b1c0)

We also investigated how voltage and temperature vary over the cycles of the same example battery. It is interesting to note that while temperature has a roughly gaussian distribution throughout any given cycle, the voltage measure has almost no variance during charge but significant variance across cycles during discharge.



(a) Voltage (volts) (b) Temperature (degrees celsius)

Figure 8: Voltage and temperature over cycles for an example batter (ref: b1c0)

Correlation with cycles @ 5% fade. Finally, we wanted to directly plot some measure of capacity fade (in this case number of cycles to reach 5% decrease from nominal) during the initial cycles against cycle life to see if we can recover the correlated behaviour we expect between early trajectory and end point of the discharge capacity curve. Encouragingly, we see the two are highly correlated, achieving a pearson correlation coefficient of approximately 0.94.

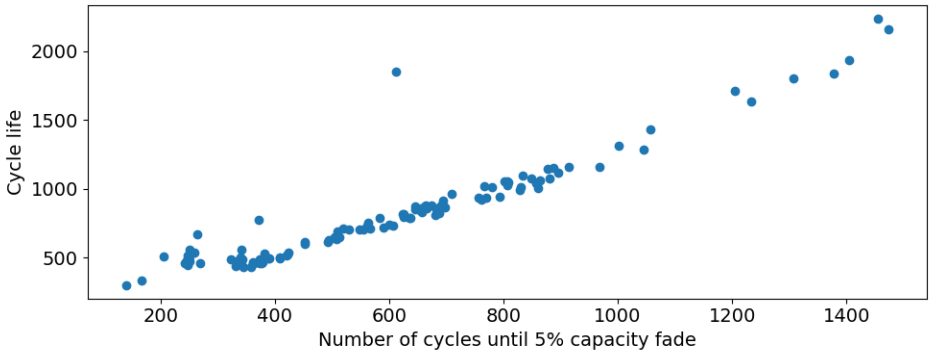
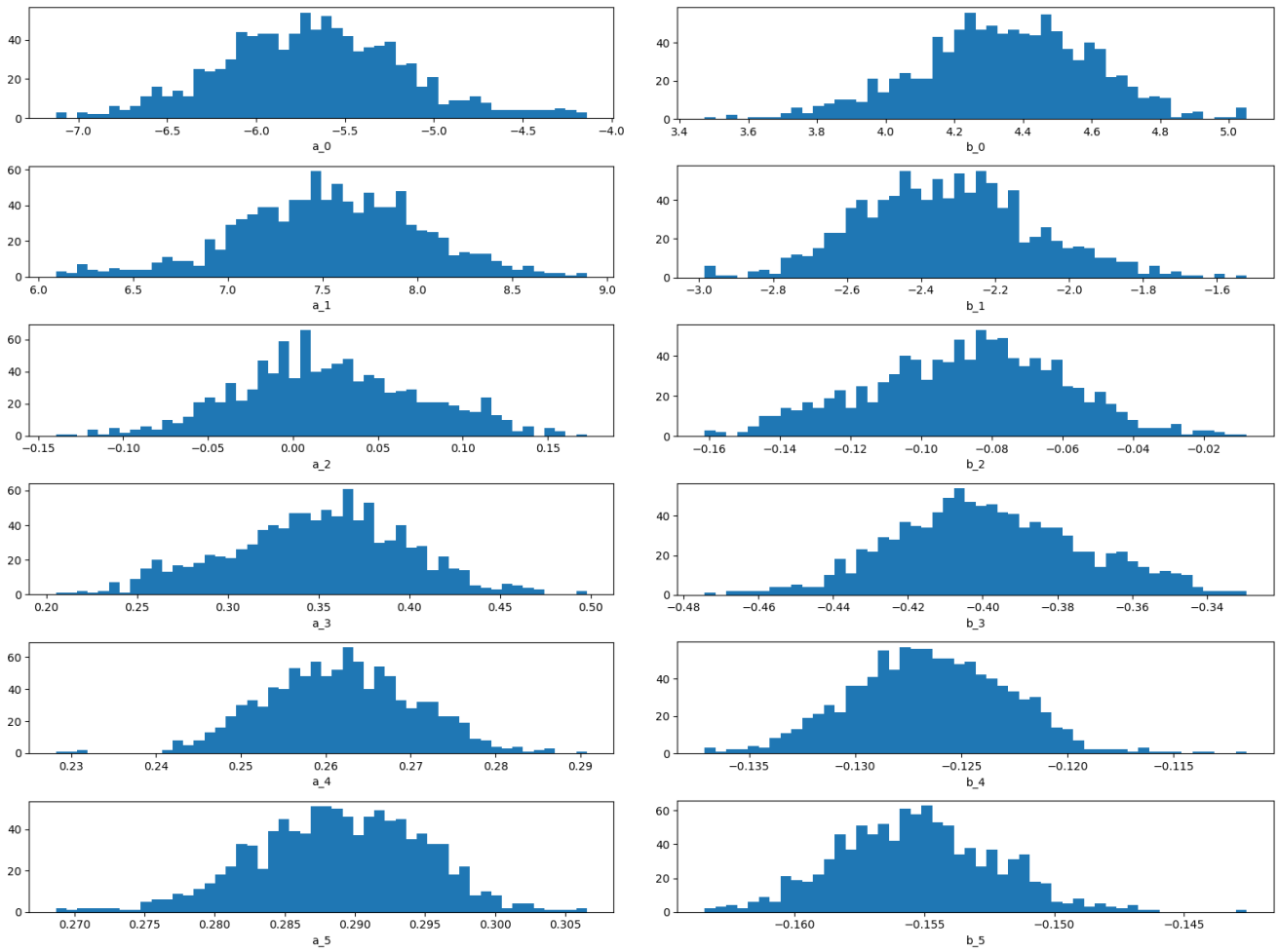


Figure 9: Scatter plot demonstrating correlation between cycle life and 5% capacity fade

5.2 Hamiltonian Monte Carlo Sampling

Sampled posterior marginals. Our sampled posterior marginal histograms are what we use to make predictions. We can take the mean from each (converges to MLE for large training dataset) or use information from these distributions to generate predictions that reflect prediction uncertainty (eg. +/- standard deviation).

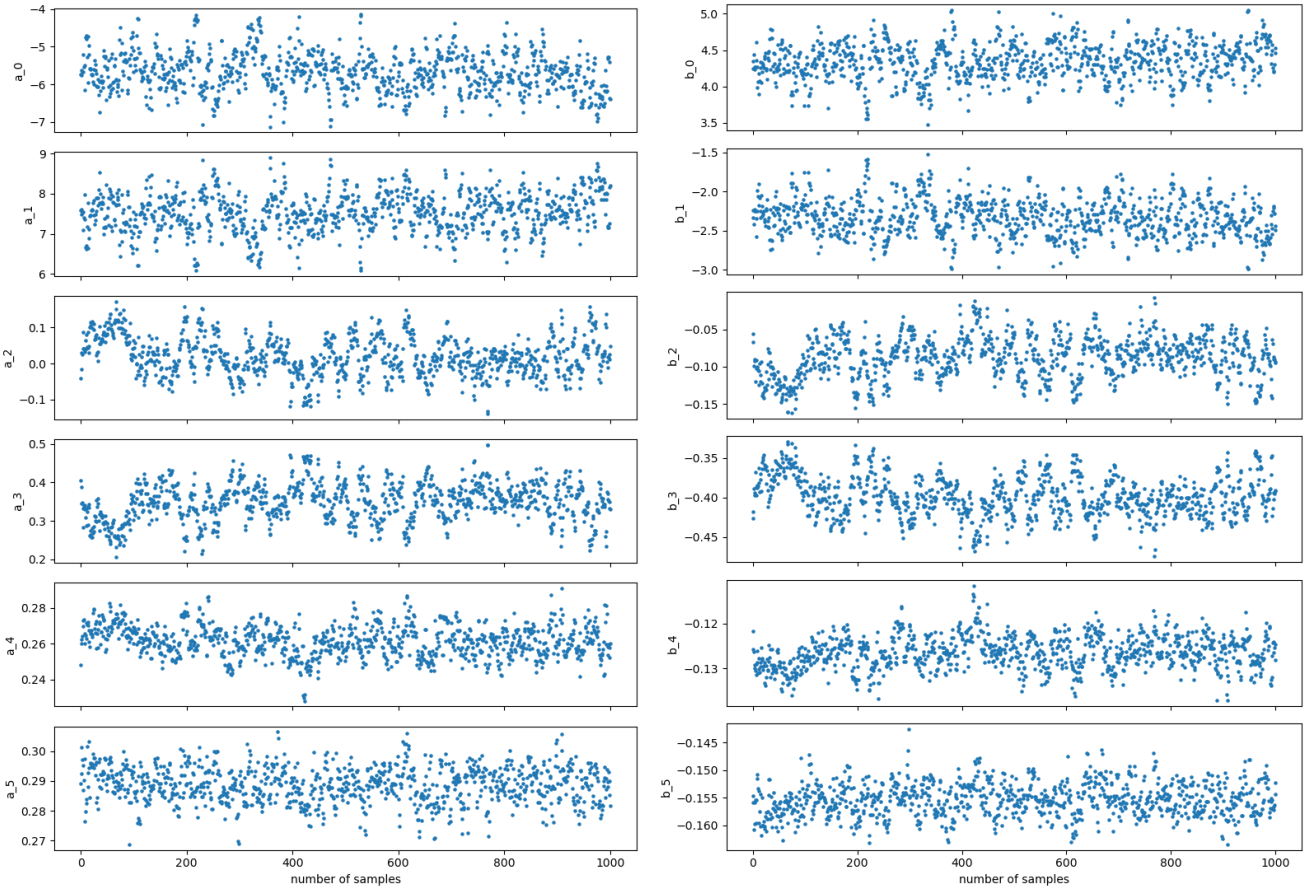


(a) Rate of decay parameters (α)

(b) Translation parameters (β)

Figure 10: Sampled posterior marginals for alpha and beta linear model parameters

Sampling efficiency. Our trace plots (only samples after burn-in shown) give a sense of the path that our MCMC sampler (No U-turn Hamilton Monte Carlo) took through the posterior space. AS we had hoped, these show reasonable variation, but no major divergences or long periods of repeated value where the sampler was stuck.



(a) Rate of decay parameters (α)

(b) Translation parameters (β)

Figure 11: Trace plot of HMC sampling for alpha and beta linear model parameters

When considering sampling efficiency we also consider auto-correlation between samples. Ideally, we want to take gradient steps through the posterior space such that we maximize the number of "effective" (uncorrelated) samples. In this case, from 1,000 draws we generate 175 effective samples.

Contributions

Julian owned development of the Bayesian Inference model. Hampus owned development of the ARIMA and Exponential Smoothing models. Karthik owned development of the Neural Network model. Everyone contributed to data exploration, fit evaluation methodology and brainstorming on / sense checking each others' results and design choices.

References

[1] Introduction to the keras tuner. `https://www.tensorflow.org/tutorials/keras/keras_tuner`. Accessed: 2022-12-02.

[2] Sungwoo Jo, Sunkyu Jung, and Taemoon Roh. Battery state-of-health estimation using machine learning and preprocessing with relative state-of-charge. *Energies*, 14(21), 2021.

[3] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. NIPS'17, page 4768–4777, Red Hook, NY, USA, 2017. Curran Associates Inc.

[4] Darius Roman, Saurabh Saxena, Valentin Robu, Michael Pecht, and David Flynn. Machine learning pipeline for battery state-of-health estimation. *Nature Machine Intelligence*, 3(5):447–456, 2021.

[5] Darius V. Roman, Ross W. Dickie, David Flynn, and Valentin Robu. A review of the role of prognostics in predicting the remaining useful life of assets. In Marko Čepin and Radim Briš, editors, *Safety and Reliability. Theory and Applications*, pages 897–904. CRC Press, 2017. 27th European Safety and Reliability Conference 2017, ESREL 2017 ; Conference date: 18-06-2017 Through 22-06-2017.

[6] Kristen A Severson, Peter M Attia, Norman Jin, Nicholas Perkins, Benben Jiang, Zi Yang, Michael H Chen, Muratahan Aykol, Patrick K Herring, Dimitrios Fraggedakis, et al. Data-driven prediction of battery cycle life before capacity degradation. *Nature Energy*, 4(5):383–391, 2019.

[7] Taylor G. Smith et al. pmdarima: Arima estimators for Python, 2017–. [Online; accessed <today>].