

Applying Machine Learning Techniques to Lithium-ion Cell Research

by

Samuel Buteau

Submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy

at

Dalhousie University
Halifax, Nova Scotia
February 2021

© Copyright by Samuel Buteau, 2020

Table of Contents

List of Tables	viii
List of Figures	xi
Abstract	xiii
List of Symbols Used	xiv
Chapter 1 Introduction.....	1
1.1 Optimization.....	2
1.1.1 The Optimization Problem.....	2
1.1.2 Gradients.....	3
1.1.3 Gradient Descent.....	4
1.1.4 Loss Functions Defined by Datasets.....	5
1.1.5 Stochastic Gradients.....	6
1.1.6 ADAM	7
1.2 Neural Networks	7
1.2.1 Tensors.....	7
1.2.2 Basic Mathematical Definition of Neural Networks	9
1.2.3 Universal Function Approximation	11
1.2.4 Backpropagation	11
1.2.5 Fully Connected Layers	12
1.2.6 Convolutional Layers.....	17

1.2.7	Averaging and Self-Attention Layers	19
1.3	Advice for Building Intuition.....	24
Chapter 2	Mathematical Properties of Electrochemical Impedance Spectroscopy Data and Equivalent Circuit Models of Lithium-ion Cells	26
2.1	Basic Mathematical Definitions of EC models and EIS Data.....	27
2.2	The Difficulty of Choosing an EC Model.....	31
2.3	A Realistic EC for Lithium-ion Cells.....	40
2.4	Enforcing Constraints on EC Parameters.....	44
2.5	Symmetries within a single EC Model.....	46
2.6	The Difficulty of Choosing EC Model Parameters Uniquely	50
2.7	A Simple Measure of EC Parameter Complexity	53
2.8	Acknowledgements	55
Chapter 3	Robust Fitting of Lithium-ion Cell EIS to EC Models.....	56
3.1	Introduction	56
3.1.1	Related Works.....	58
3.2	Formalizations of the Fitting Problem	60
3.3	A Comparison of Various Approaches to the Fitting Problem	63
3.3.1	Individual Fitting	63
3.3.2	Hybrid Approach	66
3.3.3	Supervised Approach	66
3.3.4	Unsupervised Approach.....	67

3.4	Results	71
3.4.1	Reproducibility and Access to Code.....	85
3.5	Conceptual Discussion of Tricks.....	87
3.5.1	Defining a Prior Distribution on the EC Parameters	87
3.5.2	Leveraging Symmetries to Compress the Prior	88
3.5.3	Modelling Deviations from the Averaged Prior	89
3.5.4	Penalizing Deviations from the Prior.....	90
3.5.5	Breaking the Symmetry in the EC Model.....	91
3.5.6	Penalizing Complexity.....	92
3.5.7	Automatically Setting the Relative Importance of the Penalties	92
3.5.8	Generating Fake Data to Improve Robustness of the Inverse Model	93
3.5.9	Augmenting Real Data to Improve Robustness of the Inverse Model	94
3.5.10	Error Rescaling	94
3.6	Extension to Multiple EC Models.....	95
3.6.1	Multi-Task Learning, Positive Transfer, and Negative Transfer.....	102
3.7	Key Implementation Details and Intuitive Guide to their Impact on Performance	104
3.7.1	What Separates Good from Bad Choices of Neural Networks.....	104
3.7.2	Symmetry and Convolutional Layers	104
3.7.3	Variable Numbers of Frequencies and Fully-Convolutional Architecture	108
3.7.4	Coordination of Local Processes	108

3.7.5	Batch Diversity and Masks	109
3.7.6	Rebalancing the Dataset.....	111
3.7.7	Finetuning with ADAM.....	112
3.8	Future Work: The Transformer Architecture	114
3.8.1	The Transformer Architecture as a Coordination Mechanism	114
3.8.2	The Dual Role of Log-Frequencies as Inputs of Sequence Layers.....	115
3.8.3	Positional Encodings of Log-Frequencies	116
3.8.4	A Concrete Proposal for Implementing the Transformer Architecture for Future Researchers	117
3.9	Conclusions	122
3.10	Acknowledgements	122
Chapter 4 Interpretability in the Determination of the Electrolyte Concentration in Lithium-Ion Cells Using Fourier Transform Infrared Spectroscopy		
4.1	Introduction	125
4.2	Methods.....	128
4.2.1	Data acquisition	128
4.2.2	Physical underpinnings, Beer's law, and the Proposed Models.....	132
4.2.3	Measuring performance and penalizing bad model properties.	136
4.2.4	All equations for the Linear-VM model	138
4.2.5	Code Structure and Content.....	140
4.2.6	Addendum: Spectra Measured on Other Apparatus	141

4.2.7	Future Work: Fine-tuning	142
4.3	Results and Discussion.....	143
4.4	Conclusions	155
4.5	Acknowledgements	156
Chapter 5	A Database for Lithium-ion Data	157
5.1	Motivation for a Data Processing System	157
5.2	Requirements and Desired Properties of a Data Processing System	161
5.3	Design of the System and Fundamentals of its Workings	163
5.3.1	Lithium-ion Cell Ontology (Structure of the Cell Metadata)	163
5.3.2	Obtaining Unique Names at Scale	166
5.3.3	Linking Experimental Files to the Appropriate Cells	168
5.3.4	Searches Based on the Ontology of Lithium-ion Cells.....	171
5.3.5	Data Processing and Grouping the Cycles Automatically	177
5.3.6	Creating and Annotating Datasets	181
5.3.7	Outputting Experimental Data in a Simple Format (for Humans).....	187
5.3.8	Outputting Data in a Complete Format (for Machine Learning Algorithms).....	188
5.4	User Manual and Frequently Asked Questions.....	189
5.4.1	Access to the Universal Battery Database	190
5.4.2	Searching the Database	192
5.4.3	Datasets	215

5.4.4	Registering Cells.....	221
5.4.5	Fixing Bad File Metadata and Bad Cycling Data.....	223
5.5	Future Work.....	227
5.5.1	Forms to Formsets.....	227
5.5.2	Static Database Information.....	228
5.5.3	Anomaly Warnings and Troubleshooting Help.....	229
5.5.4	Outputting Data in a Complete Format through the User Interface.....	230
5.6	Acknowledgements.....	231
Chapter 6	Conclusion and Outlook.....	232
	Bibliography.....	236

List of Tables

Table 1	The weight ratios of the dataset samples with known weight ratios. (part 1).....	129
Table 2	The weight ratios of the dataset samples with known weight ratios. (part 2).....	130
Table 3	The weight ratios of the dataset samples with known weight ratios. (part 3).....	131
Table 4	The weight ratios of the dataset samples with known weight ratios. (part 4).....	132
Table 5:	A few examples of how various informal searches can be represented as restricted formal searches.	177
Table 7	Examples of dataset filters to isolate “Check-up” cycles.	185
Table 8	Examples of dataset filters to get the various cycles except the “Check-up” cycles.....	186
Table 9	Examples of dataset filters to get the various cycles in a “Rate Map”.	186
Table 10	The option combinations to trigger a simple search or a powerful search on electrolytes or dry cells.	201

List of Figures

Figure 2.1	Example of one equivalent circuit (left, original) that can mimic another's impedance spectrum (right, converted) by using different values for the parameters.	34
Figure 2.2	Impedance formulas of a resistor and a capacitor, respectively.	34
Figure 2.3	Example of how a simple trend in one equivalent circuit looks quite different and more complicated in another equivalent circuit, even if the underlying impedance spectra are exactly the same.	36
Figure 2.4	Example of explicit conversion from a Voight-type circuit configuration to a Ladder-type circuit configuration.	40
Figure 2.5	Equivalent circuit used to fit the impedance spectra in this thesis.	42
Figure 2.6	Equivalent circuit components used in Section 2.3..	44
Figure 2.7	Reparameterizations used in this thesis..	46
Figure 2.8	Useful symmetry of the equivalent circuit model..	49
Figure 3.1	Graphical illustration of the <i>fitting problem</i> 's structure.	61
Figure 3.2	Error of the inverse model for two test datasets before and after finetuning.	74
Figure 3.3	Complexity metric of the inverse model for two test datasets before and after finetuning.	75
Figure 3.4	Some fits of the FRA test dataset using the inverse model.	78
Figure 3.5	Some fits of the EIS test dataset using the inverse model.	79
Figure 3.6	Some fits of the FRA test dataset using the inverse model followed by finetuning..	80
Figure 3.7	Some fits of the EIS test dataset using the inverse model followed by finetuning....	81
Figure 3.8	Some fits in the FRA dataset shown in original scale.	84
Figure 3.9	Error of an inverse model trained with 10 percent of the data.	86

Figure 3.10	Graphical illustration of the multiple-equivalent-circuit-models variant of the fitting problem.....	96
Figure 3.11	Various equivalent circuits supported by the software (part 1).	100
Figure 3.12	Various equivalent circuits supported by the software (part 2).	101
Figure 4.1	Comparison of two models: the cross-validation <i>performance overview</i>	145
Figure 4.2	Comparison of two models: the cross-validation <i>predictions of mass ratios</i>	146
Figure 4.3	Evaluation of the linear-vibration-modes model: cross-validation <i>reconstruction of spectra</i>	148
Figure 4.4	Evaluation of the constant-vibration-modes model: cross-validation <i>reconstruction of spectra</i>	149
Figure 4.5	Evaluation of the physical plausibility of the vibration modes of the linear-vibration-modes model.	151
Figure 4.6	Model parameters <i>X</i> during cross-validation.	152
Figure 4.7	Model parameters <i>A</i> during cross-validation.	153
Figure 5.1	Schematic of the various modalities of battery data.	158
Figure 5.2	A hierarchical breakdown of a cell into components.....	163
Figure 5.3	A more detailed breakdown of the components of the cell into subcomponents. ...	164
Figure 5.4	Illustration of the connection between filenames, cycling data, valid file metadata, and cell IDs.....	171
Figure 5.5	The front page of the software accessed from a computer within the lab's network.....	191
Figure 5.6	Overview of the search page, as it would show up when clicking on the hyperlink Search Stuff	195
Figure 5.7	Illustration of the structure of the possible searches, called the “search tree”.....	196
Figure 5.8	Zoomed-in view of the electrolyte section of the search page.....	198

Figure 5.9	Results of a simple search for electrolytes based on the input of Figure 5.8.	199
Figure 5.10	Zoomed-in view of the dry cell section of the search page, with an example input for the dry cell substructure, as it would appear just before the search.	202
Figure 5.11	Results of a simple search for <i>dry cells</i> based on the input of Figure 5.10.	203
Figure 5.12	Results of a simple search for <i>dry cell boxes</i>	204
Figure 5.13	Results of a powerful search for dry cells based on the dry cell box ID input node.	205
Figure 5.14	View of the search page, with an example input for the <i>dry cell box ID</i> and the <i>electrolyte substructure</i> similar to Figure 5.13 and Figure 5.8 respectively.	207
Figure 5.15	Results of a simple search for wet cells based on the inputs of Figure 5.14.	208
Figure 5.16	Zoomed-in view of adding some cells to a dataset.	209
Figure 5.17	A search for valid cycling data based on experimental conditions and filename substructure.	210
Figure 5.18	The results of the search from Figure 5.17.	211
Figure 5.19	The results of the search from Figure 5.17 with the “Show Visuals” option checked.	212
Figure 5.20	The steps required to force the system to reimport the files associated with given cells	214
Figure 5.21	The overview of all datasets.	215
Figure 5.22	Creating a new dataset step by step.	216
Figure 5.23	The listing of an example dataset.	217
Figure 5.24	The default edit page for an example dataset.	218
Figure 5.25	How to give a specific name to a cell in the context of a dataset.	218
Figure 5.26	The edit page after having set a specific name for cell “10001”.	219
Figure 5.27	How to set the most inclusive of all filters.	220

Figure 5.28	How to output the dataset immediately to CSV format.	221
Figure 5.29	Registering Cells (part 1)	222
Figure 5.30	Registering Cells (part 2)	222
Figure 5.31	Fixing File Metadata.	224
Figure 5.32	The “Fix Bad Cycles” page.	225
Figure 5.33	Zooming into a given cycle number region on the “Fix Bad Cycles” page.	226
Figure 5.34	View of a cycle after “cursing” region C in Figure 5.33.....	227

Abstract

Progress in lithium-ion cells research is largely a matter of determining which aspect of the cell's design and operation will lead to longer life, higher energy-density, and lower costs. These attributes can easily be characterized empirically but determining if a cell will last 10 or 20 years in the naïve way is much too slow (i.e. it would take between 10 to 20 years). The same attributes could in principle be determined from theory alone, but this is a very challenging problem and is currently unsolved.

It would therefore seem that the way forward is to leverage some experimental results obtained in a reasonable time to estimate the key attributes of various cell designs and make progress towards better designs. The *development of models and tools using data* (i.e. machine learning) is a powerful and much studied toolbox which in principle is ideally suited to the task at hand, but care must be taken in its application. If this thesis withstands the test of time, it will likely do so as the initiation of a process of cross-pollination of the field of machine learning towards lithium-ion cells research. To this end, we offer two clear applications of machine learning to the understanding of specific measurements. We apply machine learning to impedance spectroscopy and then to Fourier-transform infrared spectrometry. Finally, we offer an example of a data processing system scaled to support the long-term cycling data of a laboratory in the real-world.

As such, it is our hope that the process of cross-pollination will be helped by these concrete in-depth examples of applying the techniques of machine learning, and by a scalable system which organized tens of thousands of experiments to be used for future inquiry.

List of Symbols Used

Symbol	Definition
\mathbb{R}	The set of all real numbers, such as 0, -1, π .
\mathbb{R}^N	The set of all tuples of N real numbers. Often considered as the set of all real-valued vectors with N components.
$\mathbb{R}^N \times \mathbb{R}^M$	The set of pairs where the first element belongs to \mathbb{R}^N and the second belongs to \mathbb{R}^M .
$\prod_{j=1}^n \mathbb{R}^{N_j}$	A shorthand notation for $\mathbb{R}^{N_1} \times \mathbb{R}^{N_2} \times \dots \times \mathbb{R}^{N_n}$. The set of tuples where the j -th element belongs to \mathbb{R}^{N_j} .
$R, Q, \varphi, \omega_c, R_1, R_2, R_3$	Original parameters of the EC model from Figure 2.5. Not all of these parameters are referred by name in the text, but R denotes a resistance, R_1, R_2, R_3 denotes the resistances of the 3 ZARCs respectively, ω_c is a "characteristic frequency," and φ is an "exponent" of the given circuit.
$\sigma(x)$	The logistic function. The formula is $\sigma(x) = \frac{1}{1+\exp(-x)}$.
$r, q, \phi, w_c, w_{c1}, w_{c2}, w_{c3}$	Reparameterized parameters of the EC model from Figure 2.5, with the reparameterization given in Figure 2.7. Not all of these parameters are referred by name in the text, but r denotes a log-resistance, w_c is a "characteristic log-frequency," w_{c1}, w_{c2}, w_{c3} are the characteristic log-

frequencies of the 3 ZARC elements respectively, and ϕ is a "reparameterized exponent" of the given circuit.

$(w_i, Z_i)_{i=1}^N$

A measured spectrum, given as a list of log-angular frequencies and corresponding impedances. More precisely, w_i is the logarithm of the i -th angular frequency, and Z_i is the corresponding complex impedance.

r_α, w_α

The scaling and shifting parameters, which allow to rescale (the impedance of) and shift (the log-angular frequencies of) the measured spectra. More precisely, r_α is the logarithm of the scaling of the impedance and referred to as "the log-resistance scale parameter." Similarly, w_α is the shifting of the log-angular frequencies and referred to as "the log-frequency shift parameter."

$\theta_{EC,\mu}, \theta_{EC,\sigma}$

The prior distribution over EC parameters. More precisely, $\theta_{EC,\mu}$ are supposed to be the average values of the EC parameters (they are defined or chosen before running the program, simply based on the understanding of what constitutes typical reasonable values). Similarly, $\theta_{EC,\sigma}$ are supposed to be the standard deviations of the EC parameters (they are defined or chosen before running the program, simply based on the understanding of the range of reasonable values).

$C(R_1, R_2, R_3)$

The complexity metric, defined as $C(R_1, R_2, R_3) = \frac{(\sqrt{R_1} + \sqrt{R_2} + \sqrt{R_3})^2}{R_1 + R_2 + R_3}$.

$l_1, l_{1/2}$	The "L-1 norm" l_1 is the sum of the absolute values of the elements of a vector. Similarly, the "L-1/2 pseudo-norm" $l_{1/2}$ is the square of the sum of squared roots of the absolute values of the elements of a vector.
$\text{stopgrad}(x)$	The "stopgrad of x ", $\text{stopgrad}(x)$, is the same as x except that when computing the derivatives with respect to some variable y , we set the derivative of $\text{stopgrad}(x)$ with respect to y to 0. In other words, we treat it as a constant, but the value of the constant is set to x after the differentiation. For instance, the derivative of $y \cdot \text{stopgrad}(y^n)$ with respect to y is $1 \cdot y^n + y \cdot 0 = y^n$, for any real number n .
$P_i(\theta_{\text{Inv}}; S)$	The i -th penalty term added to the average MSE in the section Guiding the optimization. For instance, $P_1(\theta_{\text{Inv}}; S)$ would be the penalty due to deviations from the prior, $P_2(\theta_{\text{Inv}}; S)$ would be the penalty that breaks the symmetry in the EC model between the various ZARCs, and so on.
$\theta_{\text{EC}} = (r_{\text{elec}}, w_{c1}, \dots)$	The EC model's parameters are a small vector of real numbers. See Figure 2.5 for the model. Note that here, we consider the reparameterized parameters, given in Figure 2.7. For example, r_{ohm} is the log-resistance of the electrolyte resistor, w_{c1} is the characteristic log-frequency of the first electrochemical ZARC.
$Z_{\text{EC}}(\omega; \theta_{\text{EC}})$	The impedance of the EC model is a function which returns a complex number (the impedance) when evaluated at an angular frequency ω ,

where the parameters of the EC model are given by θ_{EC} . Figure 2.5 shows the EC model itself, and Figure 2.7 gives the equations for each component.

$MSE(\theta_{EC}; s)$

The mean squared error of a fit is a function which returns a positive number (the error) when evaluated with some EC parameters θ_{EC} and a measured spectrum s . Let the spectrum s be given as a list of angular frequencies ω_i and the corresponding impedances Z_i , where $i = 1, 2, \dots, N$.

Then,

$$MSE(\theta_{EC}; s) = \frac{1}{N} \sum_{i=1}^N |Z_{EC}(\omega_i; \theta_{EC}) - Z_i|^2$$

In words, it is the mean of the squared differences between the measured impedances and the impedance of the EC model.

θ_{EC}^*

The optimal EC parameters for a given measured spectrum is a small vector of real numbers. Let s be the measured spectrum under consideration.

Then, $\theta_{EC}^* = \operatorname{argmin}_{\theta_{EC}} MSE(\theta_{EC}; s)$

In words, the optimal EC parameters for a given spectrum are the EC parameters which minimize the mean squared error for that given spectrum.

θ_{Inv} The inverse model's parameters are a large vector of real numbers. In this paper, they are the parameters of the neural network representing the inverse model.

$f_{\text{Inv}}(s; \theta_{\text{Inv}})$ The inverse model is a function which, given as input a spectrum s as well as the inverse model's parameters θ_{Inv} , will return a small vector of real numbers (an approximation of the optimal EC parameters for the given spectrum). In Chapter 3, this function is computed as a neural network.

$\mathcal{L}(\theta_{\text{Inv}}; S)$ The average MSE error over a set S of spectra is a function returning a positive number (the average MSE) when evaluated with the inverse model's parameters θ_{Inv} and a set S of spectra. Let s_j be the j -th spectrum in the set S with $j = 1, 2, \dots, M$.

$$\text{Then, } \mathcal{L}(\theta_{\text{Inv}}; S) = \frac{1}{M} \sum_{j=1}^M \text{MSE}(f_{\text{Inv}}(s_j; \theta_{\text{Inv}}); s_j)$$

In words, the average of the MSE over a set of spectra.

θ_{Inv}^* The optimal inverse model's parameters for a given set of measured spectra is a large vector of real numbers. Let S be the set of measured spectra under consideration.

Then, $\theta_{\text{Inv}}^* = \operatorname{argmin}_{\theta_{\text{Inv}}} \mathcal{L}(\theta_{\text{Inv}}; S)$

In words, the optimal inverse model's parameters for a given set of spectra are the θ_{Inv} parameters which minimize the average MSE for that given set of spectra when the output from the inverse model is used to produce the EC parameters for each spectrum.

Chapter 1 Introduction

This thesis discusses the *development of models and tools using data* (i.e. machine learning) in the context of lithium-ion cell research. Machine learning is a powerful toolbox, but care must be taken in its application, especially on a novel domain.

Chapter 1 provides a minimal foundation in machine learning. Chapter 2 analyses the topic of electrochemical impedance spectroscopy to extract key mathematical properties. Chapter 3 then leverages such properties to create a robust solution with machine learning techniques. Chapter 4 studies interpretable predictions (i.e. where the process is understandable by a human) in the context of Fourier-transform infrared spectrometry. Finally, Chapter 5 illustrates how data processing systems can be built to benefit the daily operation of a laboratory and increase the scale of feasible machine learning projects.

The applications discussed were those that seemed most relevant and achievable at the time the author studied in the lab. Namely, Chapter 2 and Chapter 3 focus on electrochemical impedance spectrometry, Chapter 4 focusses on Fourier-transform infrared spectrometry, and Chapter 5 focussed on a database for long-term cycling data and structural description of lithium-ion cells.

Thorough introductions to the topic of lithium-ion cells, their known physical characteristics, and the measurements which can be performed on them exist. We shall assume that the interested reader can refer to those introductions as needed¹⁻⁶.

Similarly, quality references on the topics of machine learning, optimization, and neural networks exist. We shall assume that the interested reader can use these references as needed⁷⁻¹⁰. Indeed, all that follows in the introduction is either directly discussed in these standard references⁷ or is a straight-forward application of undergraduate mathematics.

Next, we recapitulate various important points in the theory of optimization (Section 1.1) and neural networks (Section 1.2) useful to understand the rest of the thesis. Finally, Section 1.3 gives guidelines for building a useful intuition about machine learning.

For readers with a “physics background” (if you want to see something used before it is built), the thesis might be more enjoyable if Chapter 1 is treated as an occasional reference and Chapters 2 through 5 as the interesting applications. In the extreme case, it might be advisable to start at Chapter 5 and work your way backwards until Chapter 1.

For readers with a “mathematical background” (if you want to start from nothing and build concepts in order with the usage at the end), the thesis should probably be read in order, and the main references⁷ should be kept nearby while reading the thesis.

1.1 Optimization

1.1.1 The Optimization Problem

Given a function from N real numbers to a single real number (henceforth called *the loss function*^{11–13}), the minimization problem (henceforth called *the optimization problem*) is to find settings of the inputs such that the output is as small as possible. More formally, given a function $\text{Loss}: \mathbb{R}^N \rightarrow \mathbb{R}$, we say that an input $x^* \in \mathbb{R}^N$ is a *minimum* of the function Loss and we write

$$x^* = \underset{x \in \mathbb{R}^N}{\operatorname{argmin}} \text{Loss}(x)$$

if for every possible inputs $x' \in \mathbb{R}^N$, $\text{Loss}(x^*) \leq \text{Loss}(x')$.

The optimization problem is to find such a minimum.

The introductory literature is full of examples and the rest of this thesis contains various examples as well. To keep things concrete and simple, we also give a minimalist example. Consider the

space of triplets of real numbers \mathbb{R}^3 with $(x_1, x_2, x_3) \in \mathbb{R}^3$, and consider the loss function given by:

$$\text{Loss}((x_1, x_2, x_3)) = \left(1 - \sqrt{x_1^2 + x_2^2 + x_3^2}\right)^2$$

This loss function will be minimized whenever $\sqrt{x_1^2 + x_2^2 + x_3^2} = 1$, which means that a minimum $x^* = (x_1, x_2, x_3)$ must be a unit vector in the usual Euclidian 3-D space.

1.1.2 Gradients

To properly discuss optimization, we must remind the reader about a few mathematical objects.

Linear functions $f: \mathbb{R}^N \rightarrow \mathbb{R}$ are functions which satisfy $f(\alpha\mathbf{u} + \beta\mathbf{v}) = \alpha f(\mathbf{u}) + \beta f(\mathbf{v})$ for all scalars $\alpha, \beta \in \mathbb{R}$ and all input vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^N$.

Affine functions are simply the sum of a linear function with a constant.

Differentiable functions $f: \mathbb{R}^N \rightarrow \mathbb{R}$ are functions which can be approximated locally by a linear (affine) function. Namely, for a point $\mathbf{a} \in \mathbb{R}^N$, and for any point $\mathbf{a} + \Delta\mathbf{a} \in \mathbb{R}^N$ such that $|\Delta\mathbf{a}|$ is small, there will exist a vector $\nabla\mathbf{f} \in \mathbb{R}^N$ such that $f(\mathbf{a} + \Delta\mathbf{a}) \approx f(\mathbf{a}) + \Delta\mathbf{a} \cdot \nabla\mathbf{f}$.

Here, $\mathbf{u} \cdot \mathbf{v}$ is the dot product of two vectors.

This vector $\nabla\mathbf{f} \in \mathbb{R}^N$ is known as the gradient of f evaluated at point \mathbf{a} . Each component of the gradient is a partial derivative with respect to a component of the input.

Note that for a function with *vector outputs*, one can consider the gradients of each of the outputs, and assemble them into a matrix where rows correspond to input indices and columns to output indices (or vice-versa), and elements are the partial derivatives of a given output with respect to a

given input. This is called the *Jacobian*^{14,15}. For instance, if $\mathbf{y} = f(\mathbf{x})$, then the position i,j in the Jacobian would be $\frac{\partial y_j}{\partial x_i}$.

This can be applied recursively. For instance, the gradient of a scalar function f is a function with vector outputs, so if everything is twice differentiable, we can take the *Jacobian* of the gradient of f , which is known as the *Hessian*¹⁶ of f . Concretely, it is a matrix of second order partial derivatives of f with respect to its inputs. For instance, if $\mathbf{y} = f(\mathbf{x})$, then the position i,j in the Hessian would be $\frac{\partial^2 y}{\partial x_i \partial x_j}$.

1.1.3 Gradient Descent

Given a starting point a and an affine loss function which (by definition) can be written as $\text{Loss}(a + \Delta a) = \text{Loss}(a) + \Delta a \cdot \nabla \text{Loss}$, what is the direction in input space from a which most reduces the value of Loss? In other words, if we are restricted to producing a new input $a + \Delta a$ such that $|\Delta a| = \eta$ is fixed, then what choice of Δa will yield the smallest possible value for Loss?

Simple algebra reveals that

$$\Delta a = -\frac{\eta}{|\nabla \text{Loss}|} \nabla \text{Loss}$$

produces the smallest possible value under these constraints.

More generally, the negative direction of the *gradient* $-\frac{1}{|\nabla \text{Loss}|} \nabla \text{Loss}$ is the direction in which Loss *diminishes most rapidly*. This remains true locally for differentiable functions despite the fact that the gradient of $\text{Loss}(x)$ is not constant with respect to x in general.

Therefore, a simple iterative algorithm to attempt to solve the optimization problem of a differentiable loss function $\text{Loss}(x)$ is to:

- 1 Choose a starting point $x^{(0)} = x^{(start)}$.
- 2 Compute the gradient of the loss function at the present point $g = \nabla \text{Loss}(x^{(t)})$.
- 3 Take a step $x^{(t+1)} = x^{(t)} - \eta g$
- 4 Go back to step 2 with a higher value of t unless $\text{Loss}(x^{(t+1)})$ is sufficiently small or t is sufficiently large.

This procedure is known as gradient descent¹⁷⁻¹⁹.

1.1.4 Loss Functions Defined by Datasets

Typically in machine learning, there will be a model $f: \mathbb{R}^N \times \mathbb{R}^M \rightarrow \mathbb{R}$ which is written as $f(x; \theta)$ where $x \in \mathbb{R}^N$ is the *input* of the model and $\theta \in \mathbb{R}^M$ are the *parameters* of the model, and then a possible loss function could be defined as the mean squared error²⁰ over a dataset $\mathcal{D} = \{(x^{(i)}, y_i) | i = 1, \dots, m\}$ where $x^{(i)} \in \mathbb{R}^N$ and $y_i \in \mathbb{R}$ for $i = 1, \dots, m$. Namely,

$$\text{Loss}(\theta) = \frac{1}{|\mathcal{D}|} \sum_{(x^{(i)}, y_i) \in \mathcal{D}} |f(x^{(i)}; \theta) - y_i|^2$$

This example is a standard supervised learning problem setting. In general, though the optimizer views the loss function as a function of a given set of parameters θ , it is possible to construct such a loss function out of pieces that have other dependencies (such as $f(x; \theta)$) and data (such as $\{(x^{(i)}, y_i) | i = 1, \dots, m\}$).

A reader of a previous draft of this thesis remarked that the relationship between this section and “the price of rice in China” was unclear and requested an example to clarify that.

Therefore, consider a setting where $N = 2$, and x represents a pair of values, where the first element is the number of acres of land in China devoted to rice cultivation in a given year and the

second element is the number of people living in China in the same year. Correspondingly, y would be the average price of rice sold in China over that same year relative to the value of gold. For this example, the model could be selected with various forms, but an example of a linear model would make $M = 2$ and θ would represent a pair of values such that the model is defined as $f((x_1, x_2); (\theta_1, \theta_2)) = \theta_1 x_1 + \theta_2 x_2$.

Then, the dataset could consist of $\{(x^{(i)}, y_i) | i = 1, \dots, 100\}$ with $x^{(i)}$ representing the land and population for the i -th year of the 19-th century, and similarly with y_i representing the price of rice in China (in grams of gold) for that given year. For a given setting of (θ_1, θ_2) , the value of $\text{Loss}(\theta)$ would therefore represent the mean squared error over the dataset for the very simple pricing model $f(x; (\theta_1, \theta_2))$.

1.1.5 Stochastic Gradients

As the size of the dataset increases, the cost of computing the exact gradients of the loss function described in Section 1.1.5 increases proportionally. For the large datasets often required to properly define the loss function, **such computations quickly become infeasible**.

Since gradient descent is an iterative method, a key idea is to replace the gradients of the loss function at each step by *stochastic approximations* which only depend on small subsets of the data.

For instance, the gradients of $\text{Loss}(\theta) = \frac{1}{|\mathcal{D}|} \sum_{(x^{(i)}, y_i) \in \mathcal{D}} |f(x^{(i)}; \theta) - y_i|^2$ can be estimated by the gradients of $\frac{1}{|\mathcal{D}'|} \sum_{(x^{(i)}, y_i) \in \mathcal{D}'} |f(x^{(i)}; \theta) - y_i|^2$ where $\mathcal{D}' \subset \mathcal{D}$ is a random subset of the data with size $|\mathcal{D}'| < |\mathcal{D}|$ (typically can be any number of datapoints, from a single datapoint up to a million points depending on what is practical to compute on the provided hardware). The key point is that this subset of the data is chosen at random at each iteration of the gradient descent, essentially imitating gradient descent but with some noise added due to the choice of random subset²¹.

1.1.6 ADAM

The famous ADAM (which stands for adaptive moment estimation) algorithm^{17,22,23} is a variation on gradient descent which estimates running averages of the gradient (vector of partial derivatives) and the averages of the squares of the gradient (vector of squares of partial derivatives), and takes a step proportional to the ratio of the averaged gradient and the square root of the squares of the gradient. In general, taking a step proportional to the averaged gradient provides a *momentum* to the gradient descent algorithm such that the steps will only slowly change when the gradient changes.

By dividing each average partial derivative by the square root of the average squares of the same partial derivative, we will get a number close to +1 or -1 in cases where the gradient is constant

for several steps (since $|x| = \left| \frac{1}{n} \sum_{i=1}^n x \right| = \sqrt{\frac{1}{n} \sum_{i=1}^n x^2}$ for any number x), but in cases where x varies a lot over several steps, the denominator will become much larger relative to the numerator, and we will get a number close to 0. Hence, for each partial derivative, the steps will be either positive or negative (depending on the sign of the average partial derivative), and their magnitude will be controlled by the relative scale of the variance over the last several steps of gradient descent.

1.2 Neural Networks

1.2.1 Tensors

To properly discuss neural networks, we must remind the reader about a few mathematical objects.

First, elements of a space such as \mathbb{R}^N are referred to as vectors, and elements of $\mathbb{R}^N \times \mathbb{R}^M$ are referred to as matrices. In general, such elements are collections of real numbers which can be indexed by one natural number (vectors) or two natural numbers (matrices). This notion can be generalized to collections of real numbers which can be indexed by n natural numbers (henceforth

called n -tensors), and denote the space to which they belong as $\prod_{j=1}^n \mathbb{R}^{N_j}$, where the j -th index must take values between 1 and N_j inclusively. If this view is unclear, an equivalent view is that n -tensors are real-valued functions from tuples of n natural numbers.

In this view, if A is a 3-tensor, in $\mathbb{R}^N \times \mathbb{R}^M \times \mathbb{R}^P$, then for every natural $1 \leq i \leq N, 1 \leq j \leq M, 1 \leq k \leq P$, A outputs a real number, written A_{ijk} . As a shorthand, we denote the bounds on individual indices by saying that A is an N by M by P tensor. Here, (N, M, P) is called the *dimensionality* of A and 3 is called the *arity* of A . Note that some important neural network operations are defined for n -tensors of a fixed dimensionality, while others only require a fixed arity.

A similar object would be a set S of n -tensors $S \subset \prod_{j=1}^n \mathbb{R}^{N_j}$ where any given element of this set is a n -tensor. Here the number of tensors in S is called the size of S . Such sets of tensors with every tensor having the same dimensionality are called *uniform sets*. In general sets of tensors need not be uniform. Note that some important neural networks are defined for uniform sets of tensors with arbitrary set size.

Finally, note that a n -tensor can be viewed as a sequence of $(n - 1)$ -tensors. For instance with a N by M by P tensor, we could say A is an N -sequence of M by P tensors and write $(A_i)_{i=1}^N$ where each A_i is a M by P tensor with elements defined as $(A_i)_{jk} = A_{ijk}$ (this view of a tensor as a sequence of tensors is sometimes called *currying* in reference to mathematician Haskell Curry²⁴). In this case, A would be viewed as a sequence of matrices. Here, N is the *sequence length*. Such sequences where each tensor has the same dimensionality are called *uniform sequences*. In general, sequences of tensors need not be uniform. Note that some important neural networks are defined for uniform sequences of tensors with arbitrary sequence length.

When it will be desired to generically refer to a space potentially containing tensors, sets of tensors, or sequences of tensors, we shall use the letters \mathcal{T} , \mathcal{U} , or \mathcal{V} and call it a *space of tensor-like objects*.

In the greatest generality, sequences of sequences (etc.) or sets of sets (etc.) of tensors are also possible.

1.2.2 Basic Mathematical Definition of Neural Networks

A neural network is simply a function of a tensor-like object $x \in \mathcal{T}$ (the input of the neural network), a tensor-like object $\theta \in \mathcal{U}$ (the neural network parameters), and returning a tensor-like object $y \in \mathcal{V}$ (the output of the neural network). Mathematically, we denote a given model as $y_{\text{model}}(x; \theta)$ where “model” is replaced by a name to distinguish a given neural network from another.

The reason for calling these objects *networks* is that they can be represented graphically as networks (also known as graphs) and composed together into bigger networks. A name more familiar to physicists might have been neural *circuits*, since the neural networks considered here can be represented graphically as circuits and composed into bigger circuits, but for historical reasons, the name is neural *network*. Indeed, if this section seems too abstract on first read, the reader is encouraged to skip to Section 2.1 (discussing equivalent circuits to model impedance spectra) and immediately come back to reread the current section. There is a strong parallel between equivalent circuits and neural networks, though the specific ways of building a bigger neural network out of smaller neural networks differs from the ways of building a bigger equivalent circuit out of smaller equivalent circuits.

Given two neural networks denoted as $y_1(x; \theta_1)$, $y_2(x; \theta_2)$ such that the set of possible *outputs* of the first is contained in the set of *inputs* of the second, we can define a third neural network called “serial composition of network 1 with network 2” as

$$y_{1 \rightarrow 2}(x; \theta_1, \theta_2) = y_2(y_1(x; \theta_1); \theta_2)$$

Similarly, if the two networks always produce outputs which can be summed given the same input, we can define another neural network called “the parallel sum of models 1 and 2” as

$$y_{1+2}(x; \theta_1, \theta_2) = y_1(x; \theta_1) + y_2(x; \theta_2)$$

This can be generalized further to “the parallel application of models 1 and 2”

$$y_{1||2}(x; \theta_1, \theta_2) = (y_1(x; \theta_1), y_2(x; \theta_2))$$

which is a neural network outputting a 2-sequence of tensor-like objects.

If the neural network *parameters* θ are fixed, the neural network $y(x; \theta)$ can be viewed as a function from *input* x to *output* y .

The main reasons for studying neural networks is that they can represent very complicated functional dependencies (Section 1.2.3), and they give rise to tractable optimization problems (Section 1.2.4). The optimization problem here is finding the parameters θ which minimize a loss function over a dataset.

1.2.3 Universal Function Approximation

A reader skeptical that very complicated functional dependencies can be well approximated is invited to look at the literature on Universal Function Approximation⁸ (especially the referenced visual proofs). For many common instances of neural networks this literature proves the following:

For any function f (satisfying some non-pathology criteria), any probability distribution over the inputs of f , and any given desired precision ε , there always exists a setting of the parameters θ such that the given neural network, when viewed as a function from input to output, approximates f within precision ε on average over the given probability distribution on inputs.

Note that this only holds in the limit of very big neural networks.

The details are not of much relevance in the context of this thesis, since the primary question is to achieve some *practical* precision with a given function and a given dataset.

However, it is worth noting that the results can often extend to not only approximate well the function f itself, but also that the neural network's derivatives can approximate the derivatives of f while the outputs of the neural network approximates f itself²⁵.

1.2.4 Backpropagation

Neural networks are usable in practice because the outputs are easily differentiable with respect to the parameters and with respect to the inputs as well.

This property in turn has the advantage that if it holds for neural network 1 and for neural network 2, then it also holds for the serial composition of network 1 with network 2 and similarly with the parallel sum. More generally, when making big networks out of smaller networks each having this property, the big network will also have this property, allowing gradients to be computed and optimization to be performed in a tractable manner.

Recall the rules of differentiation such that *the derivative of the sum is the sum of the derivatives*, *the derivative of the products follows the Leibnitz rule*, and *the derivative of a composition follows the chain rule*. These rules can be applied in a generic algorithm called *backpropagation* which in essence allows one to simply define the partial derivatives for individual simple neural networks, as well as the way in which the simple neural networks are composed together into the final complex neural network, and from these pieces of information, to mechanically express the partial derivatives of the outputs of the final complex neural network with respect to its parameters and inputs²⁶.

Backpropagation is the way gradients of the loss function are computed in practice for neural networks. And it is why there is a constant emphasis on the differentiability of the neural networks introduced in the literature.

Typically, neural networks which are very simple in structure, and which are not implemented as compositions of other smaller neural networks are called *layers*, and practical neural networks are often defined as combinations of predefined layers.

Once again we mention that many excellent references exist on the topic⁷, but we also recapitulate a few key layers for each type of tensor-like object described in Section 1.2.1.

- For tensors of fixed dimensionality, Section 1.2.5 discusses fully connected layers.
- For uniform sequences of tensor, Section 1.2.6 discusses convolutional layers.
- For uniform sets of tensors, Section 1.2.7 discusses averaging and self-attention layers.

1.2.5 Fully Connected Layers

The most basic neural network worthy of the name is a linear function from vectors to vectors. For 1-tensor inputs $x \in \mathbb{R}^N$ with dimensionality N , and 1-tensor outputs $y \in \mathbb{R}^M$ with dimensionality

M , the parameters of the linear layer are $\theta = W \in \mathbb{R}^N \times \mathbb{R}^M$ a 2-tensor (i.e. a matrix) often denoted by the letter W .

Then, the layer is defined such that $y = y_{\text{linear}}(x, W)$, with $y_j = \sum_{i=1}^N W_{ij}x_i$.

A close cousin of the linear function is the affine function²⁷, with parameters $\theta = (W, b)$ with $W \in \mathbb{R}^N \times \mathbb{R}^M$ and $b \in \mathbb{R}^M$. Then the affine layer is defined such that $y = y_{\text{affine}}(x, (W, b))$, with

$$y_j = b_j + \sum_{i=1}^N W_{ij}x_i$$

Note that it is possible to define affine or linear transformations between any space of n -tensors with fixed dimensionality. For instance, if the inputs are 3 by 4 by 5 tensors and the outputs are 7 by 8 tensors, then the parameters will be $\theta = (W, b)$ with W a 3 by 4 by 5 by 7 by 8 tensor and b a 7 by 8 tensor such that $y_{lm} = b_{lm} + \sum_{i=1}^3 \sum_{j=1}^4 \sum_{k=1}^5 W_{ijklm}x_{ijk}$.

Note on the Input and Output Dimensionality for Linear Models

The dimensionality of the input will determine how large a dataset this neural network will require to generalize well.

This is because a linear function is entirely determined by its output on a *basis* of the input space.

If P vectors $x^{(1)}, \dots, x^{(P)}$ are chosen at random in \mathbb{R}^N , then with probability 1,

- If P is smaller than N , then these vectors will be linearly independent, and no matter the choice of corresponding vectors in the output space $y^{(1)}, \dots, y^{(P)}$, there will be an infinity of possible linear functions which sends these inputs to these outputs. Namely, for $k=1, \dots, P$, $y^{(k)} = y_{\text{linear}}(x^{(k)}, W)$. We say that any dataset thus chosen will *underdetermine* the parameters of the neural network.

- If P equals N , then these vectors will be linearly independent, and no matter the choice of corresponding vectors in the output space $y^{(1)}, \dots, y^{(P)}$, there will be a single possible linear function which sends these inputs to these outputs. We say that any dataset thus chosen will *exactly determine* the parameters of the neural network.
- If P is greater than N , then these vectors will not be linearly independent, and there will therefore exist choices of corresponding vectors in the output space $y^{(1)}, \dots, y^{(P)}$, for which no linear function could send these inputs to these outputs. We say that any dataset thus chosen will *overdetermine* the parameters of the neural network (indeed either the constraints cannot be satisfied or there is a smaller subset of the dataset which would lead to the same linear functions being possible).

This means that, for a fixed dimensionality of the input space, the same size of dataset is needed to exactly determine the linear function's parameters *no matter how large the output space is!* For instance, with an input dimensionality of 3 and output dimensionality of 10^{10000} , a dataset containing 3 linearly independent input values will exactly determine the parameters of the linear function. This neural network has 3×10^{10000} parameters!

A first order approximation to this phenomenon would relate linearly the number of parameters to the smallest number of datapoints in a dataset that exactly determines (or at least overdetermines) a neural network. As a formula:

$$\text{Data Requirement} = \frac{\text{Number of Parameters}}{\text{Supervision Strength}}$$

Where the *supervision strength* is simply the ratio of number of parameters to data requirement, but very roughly speaking, it tells us “how many parameters does a single datapoint in the dataset allow to determine.”

Based on this simple view, what happens is that for a given input dimensionality, as the output dimensionality is increased, the number of parameters increases linearly, but so does the supervision strength.

In order to reproduce exactly a dataset given by $x^{(1)}, \dots, x^{(P)}$ and $y^{(1)}, \dots, y^{(P)}$, one can define an optimization problem over a linear neural network by giving the following loss function:

$$Loss(W) = \frac{1}{P} \sum_{k=1}^P |y^{(k)} - y_{\text{linear}}(x^{(k)}, W)|^2$$

which can be more explicitly given as:

$$Loss(W) = \frac{1}{P} \sum_{k=1}^P \sum_{j=1}^M \left(y_j^{(k)} - \sum_{i=1}^N W_{ij} x_i^{(k)} \right)^2$$

This formula allows us to understand why the “supervision strength” is proportional to the output dimensionality in this case. Indeed, the loss function can be viewed as the sum over j of

$$\frac{1}{P} \sum_{k=1}^P \left(y_j^{(k)} - \sum_{i=1}^N W_{ij} x_i^{(k)} \right)^2$$

which constrains directly only the j -th component of the output, and without which all the parameters W_{ij} for that fixed j value (in total N parameters in our case) would be underdetermined.

We could view the *number of independent loss functions* contained within the *actual loss* as a measure of the “supervision strength,” though many other factors can contribute when considering more complicated neural networks with more complicated loss functions.

Non-linearities

Finally, it is common practice to serially compose linear and affine functions with a so-called elementwise non-linearity. Let $\rho: \mathbb{R} \rightarrow \mathbb{R}$ be any function from real numbers to real numbers, then we define a simple neural network called *the elementwise application of ρ* such that for 1-tensor inputs $x \in \mathbb{R}^N$ with dimensionality N , and 1-tensor outputs $y \in \mathbb{R}^N$ with the same dimensionality, it is defined as $y = y_\rho(x)$, with $y_j = \rho(x_j)$. Note that there are no parameters for this operation, but the serial composition of an affine layer with the elementwise application of ρ

is $y = y_{\text{affine} \rightarrow \rho}(x, (W, b))$, with $y_j = \rho(b_j + \sum_{i=1}^N W_{ij}x_i)$.

There are various common choices for the non-linearity²⁸ such as:

- *relu* (x), such that $relu(x) = x$ if $x \geq 0$, but $relu(x) = 0$ if $x < 0$. It stands for rectified linear unit.
- *tanh* (x) such that $tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$, which tends to -1 when x goes to $-\infty$, +1 when x goes to ∞ , and equals 0 when x equals 0. It stands for hyperbolic tangent.
- *sigmoid* (x) such that $sigmoid(x) = \frac{1}{2} + \frac{1}{2}tanh(x)$. It has limits between 0 and 1.

As a shorthand, the serial composition of an affine layer with the elementwise application of a well-known non-linearity would be called for instance *a fully-connected layer with relu activation*²⁹.

Non-linearities are not always needed (for instance, at the very output of a complicated neural network which predicts values between $-\infty$ and ∞ , there would be no use for a non-linearity). However, in order to satisfy the conditions of the Universal Function Approximation discussed in Section 1.2.3, it is not possible to simply serially compose affine layers without non-linearities.

Indeed, it can be shown that serially composed affine layers are always equivalent to a single affine layer. Yet, through the magic of the Universal Function Approximation theorems, any of the three non-linearities discussed above, as well as many others, would allow a net gain in the kinds of functions which may be expressed through the serial composition of many fully-connected layers with such activations. Readers still confused about how this all works are invited to look at proofs of the Universal Function Approximation theorems, since these proofs construct an actual neural network (by explaining how to choose its parameters) to approximate the target function⁸.

1.2.6 Convolutional Layers

Given a fully connected layer with ρ activations $y = y_{\text{affine} \rightarrow \rho}(x, (W, b))$ between 1-tensors of dimensionality N and 1-tensors of dimensionality M , there is a trivial way to produce a neural network which takes uniform sequences of 1-tensors to uniform sequences of 1-tensors. Namely, by applying $y_{\text{affine} \rightarrow \rho}$ *elementwise*. In other words, if x is now a sequence of 1-tensors and y is a sequence of 1-tensors, define $y = y_{\text{conv},1}(x, (W, b))$ by $y_k = y_{\text{affine} \rightarrow \rho}(x_k, (W, b))$ for all the values of the sequence index k .

More generally, given a fully connected layer with ρ activations $y = y_{\text{affine} \rightarrow \rho}(x, (W, b))$ between 2-tensors of dimensionality K by N and 1-tensors of dimensionality M , it is possible to define a neural network which takes as input any sequence of 1-tensors (dimensionality N) and returns a sequence of 1-tensors (dimensionality M) by applying $y_{\text{affine} \rightarrow \rho}$ repeatedly on all possible segments of K consecutive 1-tensors in the input. There are three obvious ways to do this.

- First, by only applying $y_{\text{affine} \rightarrow \rho}$ to places in the sequence where K consecutive 1-tensors exist. In other words, $y_k = y_{\text{affine} \rightarrow \rho}\left((x_{k+d})_{d=0}^K, (W, b)\right)$ is only defined for elements k

in the sequence such that elements k to $k+K$ exist in the sequence. In this way, the output sequence will be shorter than the input sequence.

- Second, by applying $y_{\text{affine} \rightarrow \rho}$ to all places in the sequence with the convention that anytime the equation requires a x_{k+d} outside the sequence, a tensor of appropriate size filled with 0 values will be used instead. In this way, the output sequence will have the same length as the input.

If K is an odd number, then there is another set of possibilities which is intuitively nicer to keep track of, namely, at each element in the input sequence, the segments are *centered around* that element instead of *starting at* that element:

- First, by only applying $y_{\text{affine} \rightarrow \rho}$ to places in the sequence surrounded by $\frac{K-1}{2}$ 1-tensors.

In other words, $y_k = y_{\text{affine} \rightarrow \rho} \left((x_{k+d})_{d=-\frac{K-1}{2}}^{\frac{K-1}{2}}, (W, b) \right)$ is only defined for elements k in

the sequence such that elements $k - \frac{K-1}{2}$ to $k + \frac{K-1}{2}$ exist in the sequence. In this way, the output sequence will be shorter than the input sequence.

- Second, by applying $y_{\text{affine} \rightarrow \rho}$ to all places in the sequence with the convention that x_{k+d} outside the sequence are substituted by tensors filled with 0 values.

Section 3.7.2 illustrates these general ideas with a concrete example and attempts to justify this type of layers on an actual problem of relevance to lithium-ion research.

Due to strange historical reasons, such neural network is called a 1-D *convolution*³⁰ with *kernel* W and *bias* b , even though the mathematical name for such operation should be *autocorrelation*. Compared to the convolutions seen in physics textbooks, the summation indices run backwards.

In precisely the same way that a single index may be extended to arbitrary values (going from 2-tensors to sequences of 1-tensors), it is possible to extend two indices (going from 3-tensors to *sequences of sequences* of 1-tensors), and this is routinely done in images (2-D convolutions). Indeed, the same concepts can be generalized to n -D convolutions for any positive integer n .

In the literature, the various numbers of dimensions have names. The output dimension is the number of *filters*, the last dimension of the input is the number of *channels*, and the other dimensions of the input are the *kernel width* (for 1-D convolutions), the *kernel width and height* (for 2-D convolutions), and generally the *receptive field dimensions*.

Why is it called the *receptive field* dimensions? This is because the output at position k or position (w,h) in the output only depends on part of the input (a segment of positions corresponding to the receptive field dimensions).

1.2.7 Averaging and Self-Attention Layers

Fully-Connected layers go from tensors of fixed dimensionality to tensors of fixed dimensionality. Convolutions go from sequences to sequences. What about cases where the input or output is a uniform set? To address this, we first consider cases where the input is a uniform sequence and the output is a single tensor of fixed dimensionality.

A simple solution is to average elementwise across the sequence. For instance given a sequence of vectors $x^{(1)}, \dots, x^{(P)}$ in \mathbb{R}^N , one can produce a single output vector $x^{(\text{average})}$ in \mathbb{R}^N with

$$x^{(\text{average})}_i = \frac{1}{P} \sum_{k=1}^P x^{(k)}_i.$$

Notice that indeed, this works for *uniform sets* as well as uniform sequences. For instance, if $S \subset \mathbb{R}^N$ is a finite set of vectors, then one can produce a single output vector $x^{(\text{average})}$ in \mathbb{R}^N with

$$x^{(\text{average})}_i = \sum_{x \in S} \frac{1}{|S|} x_i.$$

In cases where a more complicated relationship between inputs and outputs must be approximated, simple averaging does not suffice. In simple averaging, the multiplier for all the elements is the same, (namely $\frac{1}{|S|}$). A generalization which unlocks much more powerful function approximation is to allow weighted averaging. Indeed, if the input is a set of pairs where the first element of the pair is a positive real number and the second element of the pair is a vector (i.e. $S \subset \mathbb{R} \times \mathbb{R}^N$), then one can produce a single output vector $x^{(\text{average})}$ in \mathbb{R}^N with

$$x^{(\text{average})}_i = \frac{\sum_{(w,x) \in S} w x_i}{\sum_{(w',x') \in S} w'}$$

Note that the denominator is simply a renormalization constant to ensure that the multipliers $\frac{w}{\sum_{(w',x') \in S} w'}$ sum to 1.

The restriction that the weights should be positive is annoying when they come from the output of previous neural network layers, so it can be removed by taking as input the *logarithms of weights* u instead of the weights w themselves, then one can produce a single output vector $x^{(\text{average})}$ in \mathbb{R}^N with

$$x^{(\text{average})}_i = \frac{\sum_{(u,x) \in S} \exp(u) x_i}{\sum_{(u',x') \in S} \exp(u')}$$

In general, the logarithms of weights will be the outputs of some neural network, and the vectors to be averaged will themselves be the outputs of some neural network. This way of averaging is referred to as an *attention mechanism*.

This arrangement is useful as a final conversion from a set (or sequence) to a single tensor, but it assumes that a neural network converted the initial inputs to a form suitable for averaging. In the case of sequences, this can be done with convolutions, but what about the case of sets?

To answer this question, we shall proceed in three steps. First, we shall describe a scheme to capture interactions of a single tensor with a whole set of tensors. Second, we shall generalize to sets interacting with other sets. Third, we shall consider the special case of a set of tensors interacting with itself.

First, imagine the task of learning the dynamics of a small comet as it travels through a solar system filled with large planets and stars. Though this example is motivating, the task is really to define a simple neural network layer which will be applicable to many problems.

In particular, imagine that each large body in the solar system is represented as a tensor (a vector in \mathbb{R}^N for simplicity), and the small comet is represented as a tensor (a vector in \mathbb{R}^M for simplicity). These tensors are given as inputs, and the task is to predict the final position of the comet. Furthermore, assume that the large bodies in the solar system are held in a static configuration by some external force.

Then, an intuitive way the problem may be approached is by composing layers which each take a small time step for the comet, where the dynamics are hopefully simpler, but there could be other less intuitive ways of decomposing the problem into a large sequence of neural network layers.

The attention mechanism which has seen the largest successes³¹ circa 2020 AD can be presented in this setting as follows:

For a set of 1-tensor inputs $S \subset \mathbb{R}^N$ with dimensionality N , a 1-tensor input $z \in \mathbb{R}^M$ and 1-tensor output $y \in \mathbb{R}^O$, the parameters of the *simple dot-product attention layer* are $\theta = (Q, K, V)$ with the *query projection* $Q \in \mathbb{R}^M \times \mathbb{R}^P$, the *key projection* $K \in \mathbb{R}^N \times \mathbb{R}^P$ and the *value projection* $V \in \mathbb{R}^N \times \mathbb{R}^O$.

Then, the layer is defined such that $y = y_{\text{dot-attention}}((S, z), (Q, K, V))$, with

$$y_j = \frac{\sum_{(u,v) \in T} \exp(u) v_j}{\sum_{(u,v) \in T} \exp(u)}$$

and

$$T = \{(u, v) \mid u = \sum_{p=1}^P q_p k_p, v_j = \sum_{i=1}^N V_{ij} x_i, q_p = \sum_{j=1}^M Q_{jp} z_k, k_p = \sum_{i=1}^N K_{ip} x_i, x \in S\}$$

Since this is a bit hard to read, we summarize it step by step:

1. The query projection is applied to the input z (the comet) to obtain the query q given by

$$q_p = \sum_{j=1}^M Q_{jp} z_k.$$

2. For every $x \in S$ (large body in the solar system),

- a. The key projection is applied to produce a key k given by $k_p = \sum_{i=1}^N K_{ip} x_i$.
- b. The value projection is applied to produce a value v given by $v_j = \sum_{i=1}^N V_{ij} x_i$.
- c. The dot product of the unique query q and the key k (for that element of the set) is computed to produce the logarithm of the weight $u = \sum_{p=1}^P q_p k_p$.

3. The values are averaged according to the logarithms of the weights to produce the output.

One can see how such a layer may produce the necessary information to compute the force exerted on the comet by the large bodies in the solar system, which could then be used to update the representation of the comet, and so on.

In the same way, one could imagine having many comets in the same solar system, and the simple dot product layer between a set and a single tensor generalizes as follows:

For a set of 1-tensor inputs $S \subset \mathbb{R}^N$ with dimensionality N , a set of 1-tensor input $Z \subset \mathbb{R}^M$ and a set of 1-tensor outputs $\Omega \subset \mathbb{R}^O$ with $|Z| = |\Omega|$ but allowing $|Z| \neq |S|$, the parameters of the *simple dot-product cross-attention layer* are $\theta = (Q, K, V)$ with the *query projection* $Q \in \mathbb{R}^M \times \mathbb{R}^P$, the *key projection* $K \in \mathbb{R}^N \times \mathbb{R}^P$ and the *value projection* $V \in \mathbb{R}^N \times \mathbb{R}^O$.

Then, the layer is defined such that $\Omega = y_{\text{dot-cross-attention}}((S, Z), (Q, K, V))$, where $y_{\text{dot-cross-attention}}((S, Z), (Q, K, V)) = \{y_{\text{dot-attention}}((S, z), (Q, K, V)), |z \in Z\}$ is simply the elementwise application of the simple dot-product attention.

Finally, imagine that it is desired to describe the dynamics of all the solar system objects (large bodies and comets) interacting together without artificial statis of the large bodies. By making the two input sets in cross-attention the same, one obtains a powerful layer which captures pairwise interactions within a set, and it is called *self-attention*.

Formally, For a set of 1-tensor inputs $S \subset \mathbb{R}^N$ with dimensionality N and a set of 1-tensor outputs $\Omega \subset \mathbb{R}^O$ with $|S| = |\Omega|$, the parameters of the *simple dot-product self-attention layer* are $\theta = (Q, K, V)$ with the *query projection* $Q \in \mathbb{R}^N \times \mathbb{R}^P$, the *key projection* $K \in \mathbb{R}^N \times \mathbb{R}^P$ and the *value projection* $V \in \mathbb{R}^N \times \mathbb{R}^O$.

Then, the layer is defined such that $\Omega = y_{\text{dot-self-attention}}(S, (Q, K, V))$, where $y_{\text{dot-self-attention}}(S, (Q, K, V)) = y_{\text{dot-cross-attention}}((S, S), (Q, K, V))$.

Note that in practice³¹⁻³³ D of these layers with outputs in $\mathbb{R}^{\frac{O}{D}}$ can be applied in parallel to obtain a set of 2-tensor outputs $\Omega \subset \mathbb{R}^D \times \mathbb{R}^{\frac{O}{D}}$ and this is called *multi-headed dot-product self-attention*.

1.3 Advice for Building Intuition

The performance of machine learning system is difficult to predict, and in general no method is better than another on all problems³⁴. However, in the opinion of the author, the greatest boost to success is to develop a strong intuition about the relationship between the **model architecture**, the **mathematical properties of the application domain**, and the **performance of the system**. Therefore, the text attempts to capture arguments and reasoning about machine learning system performances at the level of rigor the author used to guide his decisions. It would be quite challenging to formalize all these arguments into proofs (indeed, most of these arguments are not valid in general settings and depend strongly on various features of the given problem) and it would be counter productive since the value of such arguments is to reduce the number of trial and errors necessary to obtain an adequate solution to a given problem. As such, these arguments represent near-immediate intuitions to guide exploration, they cannot afford a lengthy validation and correction process, and their use can tolerate a degree of inaccuracy. Of course, seeing specific instances of the author's far-from-perfect intuition in action is not enough to fully develop this skill in the reader. To this more ambitious end, the author offers the following advice:

1. **When modifying a machine learning design and running a numerical experiment, seek not to increase the performance metrics at all cost; instead, seek to be the least confused by the differences in performance between various alternative designs, and**

when a change in performance is confusing, strive to understand it. In the author's experience, not only does this lead to much simpler and bug-free code, but it also is the most powerful way to improve one's intuition on the relevant problems one is tasked to solve.

2. Once adequate performance has been demonstrated, attempt to reproduce a similar performance with the minimal possible complexity of design. Here, complexity refers to the length of the code.
3. Once a simple implementation has adequate performance, attempt to reproduce a similar performance using components commonly used in the machine learning literature.
4. If the common components cannot reproduce the target performance, investigate, and find the smallest change to the common component which reproduces the target performance.

Chapter 2 Mathematical Properties of Electrochemical Impedance Spectroscopy Data and Equivalent Circuit Models of Lithium-ion Cells

Despite the accuracy and non-intrusive nature of Electrochemical Impedance Spectroscopy (EIS), the impedance spectra of commercial lithium-ion cells are notoriously hard to interpret. Such measurements contain information about multiple distinct steps that the lithium ions must undergo to travel from one electrode to the other. To extract physical insights from EIS datasets, one must

- Choose an Equivalent Circuit (EC) model which will represent the data.
- Reliably find estimates for the parameters of the EC model which closely match a given impedance spectrum (henceforth called *the inference problem*).
- Investigate trends across time, state of charge, and even different chemistries.

High quality introductions to EIS, EC models, and their relationship to lithium-ion cells exist^{1,35-37}, and a passing familiarity with these concepts will be assumed of the reader. This Chapter will instead focus on illustrating a key component of a robust tool based on machine learning model: understanding the mathematical properties of the data.

The Chapter is organised as follows:

- Section 2.1 defines EIS data and EC models as mathematical objects
- Section 2.2 discusses the difficulty of the choosing an EC model
- Section 2.3 presents a reasonable choice of EC model

- Section 2.4 rewrites the model equations to simplify the parameter space, helping to simplify the inference problem
- Section 2.5 defines symmetries of the model equations, also helping to simplify the inference problem.
- Section 2.6 describes the multiplicity of EC parameters which may lead to approximately the same EIS observations
- Section 2.7 defines a simple metric to help compare different solutions to the inference problem and thus reduce the multiplicity mentioned in Section 2.6

As will be shown in this Chapter and Chapter 3, basic physical and mathematical insights about the data can be leveraged to guide design decisions of machine learning solutions, both enabling better outcomes, and building confidence into the solution.

This chapter is substantially taken from a corresponding paper³⁸ though the paper explores the conversions between various ECs more thoroughly and was published along with a simple implementation of the conversion formulas, accessible at <https://github.com/Samuel-Buteau/Explicit-Conversion-Equivalent-Circuits-EIS>. Furthermore, to streamline the thesis as a whole, substantial components of the paper³⁹ corresponding to Chapter 3 have been refactored into this Chapter.

2.1 Basic Mathematical Definitions of EC models and EIS

Data

When modulating the voltage across the terminals of a lithium-ion cell with a small amplitude sinusoidal wave, there will be a time-dependent current response. In the linear regime, the response will also be sinusoidal. Furthermore, the ratio of amplitudes between the excitation and the

response, as well as the phase shift, will be independent of the excitation amplitude, only depending on frequency. This amplitude ratio and phase shift (an angle) can be represented together as a complex number, called *impedance*.

Generally, using angular frequency (i.e. frequency multiplied by 2π) will make the formulas nicer. Therefore, unless otherwise mentioned, ω shall be used to denote angular frequency of the excitation, and the notation will be abused by calling ω the *frequency*.

Equivalent Circuit Model Definition

An Equivalent Circuit (EC) model is simply a function which takes as inputs the frequency ω and a vector of n real numbers $\theta_{EC} \in \mathbb{R}^n$ called the *EC model parameters*, and which returns a complex number, called *the impedance of the model at the given frequency*. Mathematically, we denote a given model as $Z_{\text{model}}(\omega; \theta_{EC})$ where “model” is replaced by a name to distinguish a given EC model from another.

The reason for calling these objects *circuits* is that they can be represented graphically as circuits (graphs) and can be composed together into bigger circuits.

For instance, given two EC models denoted as $Z_1(\omega; \theta_{EC,1})$ and $Z_2(\omega; \theta_{EC,2})$, we can define a new EC model called “model 1 in series with model 2” as

$$Z_{1-2}(\omega; \theta_{EC,1}, \theta_{EC,2}) = Z_1(\omega; \theta_{EC,1}) + Z_2(\omega; \theta_{EC,2})$$

Similarly, we can define yet another EC model called “model 1 in parallel with model 2” as

$$Z_{1\parallel 2}(\omega; \theta_{EC,1}, \theta_{EC,2}) = \frac{1}{\frac{1}{Z_1(\omega; \theta_{EC,1})} + \frac{1}{Z_2(\omega; \theta_{EC,2})}}$$

If the EC model parameters are fixed, an EC model can be viewed as a function from frequency to impedance. Compare these definitions to those of a neural network found in Section 1.2.2.

Impedance Spectrum Definition and Relationship to Equivalent Circuit models

An EIS, or an impedance *spectrum* is a set of pairs of frequencies and impedances $\{(\omega_i, Z_i) | i = 1, \dots, m\}$.

Obviously, if an EC model $Z_{\text{model}}(\omega; \theta_{\text{EC}})$ together with the corresponding EC parameters θ_{EC} , and a set of frequencies $\{\omega_i | i = 1, \dots, m\}$ are all fixed, then they can be combined into an impedance spectrum, namely $\{(\omega_i, Z_{\text{model}}(\omega_i; \theta_{\text{EC}})) | i = 1, \dots, m\}$. This procedure will be called *sampling EC model $Z_{\text{model}}(\omega; \theta_{\text{EC}})$ with parameters θ_{EC} at the frequencies $\{\omega_i | i = 1, \dots, m\}$.*

In general, an impedance spectrum measured from an actual physical system $\{(\omega_i, Z_i) | i = 1, \dots, m\}$, could be arbitrary, and need not correspond to sampling a sampling EC model.

However, the abstraction with which physical systems are typically modelled is that there exists an EC model $Z_{\text{ideal}}(\omega; \theta_{\text{EC}})$ together with fixed EC parameters θ_{EC} underlying the physical system during the measurement, and that under idealized conditions, the measured spectrum $\{(\omega_i, Z_i) | i = 1, \dots, m\}$ would be approximately equal to the sampling of the ideal model with appropriate parameters at the given frequencies. In other words, $Z_i \approx Z_{\text{ideal}}(\omega_i; \theta_{\text{EC}})$ under ideal conditions. However, since things are not ideal, what is measured is instead the ideal impedance *plus some noise term* so that $Z_i = Z_{\text{ideal}}(\omega_i; \theta_{\text{EC}}) + \varepsilon_i$ for some set $\{\varepsilon_i | i = 1, \dots, m\}$.

Actually, these noise terms again don't have to obey any assumption, but for good quality data with small noise terms, these terms can usually be modelled adequately by a complex random variable where both the real and imaginary part are each the sum of a normally distributed real

random variable (the absolute error) and a normally distributed real random variable multiplied by the modulus of the ideal impedance $Z_{\text{ideal}}(\omega_i; \theta_{\text{EC}})$ (the relative error).

To make things simple, consider the case where only the absolute error is significant.

The main problem around experimental impedance spectra is the *inference problem*. Namely, given an EC model which is thought to adequately approximate the ideal EC model for a given physical system, and given an impedance spectrum from that physical system, the problem is to estimate the plausible values of the EC parameters θ_{EC} which produced the observation.

To make things even simpler, the problem may be further broken into two subproblems: 1) finding the most plausible EC parameters θ_{EC}^* given the observation $\{(\omega_i, Z_i) | i = 1, \dots, m\}$ and 2) estimating the uncertainties on the parameters such that any parameter within these uncertainties are still quite plausible.

If all possible values of the parameters are equally likely before seeing the data, and the noise is assumed to be normally distributed, then finding the most plausible EC parameters is equivalent to minimizing the mean squared error of reconstruction

$$\text{MSE}(\theta_{\text{EC}}) = \frac{1}{m} \sum_{i=1}^m |Z_i - Z_{\text{ideal}}(\omega_i; \theta_{\text{EC}})|^2$$

This would imply that the noise terms are as small as possible which is the most likely outcome when observing independent normally distributed random variables centered around 0.

Since one almost never has access to the exact ideal model underlying a physical system, the ideal model is replaced by some other model chosen to represent the physical system, so the mean squared error of reconstruction henceforth refers to

$$\text{MSE}(\theta_{\text{EC}}) = \frac{1}{m} \sum_{i=1}^m |Z_i - Z_{\text{chosen}}(\omega_i; \theta_{\text{EC}})|^2$$

where $Z_{\text{chosen}}(\omega_i; \theta_{\text{EC}})$ is the choice for the EC model under consideration.

Note however that minimizing an equation which is non-linear in the parameters θ_{EC} , such as the one above, can itself be a challenging task. This minimization problem is henceforth called the *fitting problem*.

To clarify, to solve the inference problem, one must solve the fitting problem, but solving the fitting problem may not be sufficient by itself, since other considerations may make one solution to the fitting problem more plausible as a solution to the inference problem than another.

2.2 The Difficulty of Choosing an EC Model

Chapter 3 considers the problem of fitting tens of thousands of impedance spectra to a physical model to extract some trends. Perhaps the most basic part of this problem is to choose an EC model to fit the spectra. However, there are multiple models discussed in the literature with various interpretations. Choosing between these models based only on the spectra themselves is difficult. For instance, it is known^{36,40} that many of the circuits discussed in the literature, despite having different physical interpretations, are mathematically equivalent or approximately equivalent to each other in the sense that they can produce exactly the same spectra, although not with the same parameters.

To better understand this phenomenon at a theoretical level, the simple case of circuits made up of resistors and capacitors is explored, though circuits used in practice contain more realistic components (constant phase elements) and correspondingly are plagued with a larger set of possible conversions between EC models.

Furthermore, the existence of relatively simple formulas to convert between various circuit topologies helps explain why, in the context of Chapter 3, a neural network model which estimates the parameters of one EC could easily be extended to estimate the parameters of various ECs for which conversion rules exist.

Note, however, that the conversions alluded to below are only tools, and not meant to replace a physical analysis of the impedance. Most ECs that will fit a complex spectrum have nothing to do with the underlying physics, and yet these conversions will allow them to fit equally well the experimental data than the physically meaningful EC. Many unusual topologies are reachable with these conversions, but this is just a consequence of making the transformations as general and simple as possible. Judgement must be used when using these tools.

It is worth illustrating what is meant by “converting between two ECs.” To accomplish this, the simplest non-trivial example will be investigated.

Figure 2.1 shows two ECs and Figure 2.2 gives a reference for the two components used in this introductory discussion. The two ECs look different, but they can produce **exactly the same** impedance spectra, when their parameters are chosen appropriately. Below Figure 2.1, the formula for the impedance of each EC is shown. Each EC contains two resistors and one capacitor. The original EC on the left has three parameters (R_{11} , R_{12} , C_{11}). If these values are fixed (e.g. $R_{11} = 1 \Omega$, $R_{12} = 10 \Omega$, $C_{11} = 1 \text{ F}$), then the impedance spectrum is fully determined. To say that the original EC can be converted to the *converted* EC is to say that one can determine the values of the *converted* parameters (R_{21} , R_{22} , C_{21}) so that the impedance of the *converted* EC will be **exactly the same as the impedance of the original EC at all frequencies**. This conversion is given in the third column of Figure 2.1 and has previously been published⁴⁰.

The notation works as follows. The parameters have two indices. The first index determines whether the parameter belongs to the original EC (index 1) or the “converted” EC (index 2). The second index serves to differentiate the various resistors and the various capacitors within a single EC.

Original	Converted	Formula
		$R_{21} = R_{12} + R_{11}$ $R_{22} = (R_{12} + R_{11}) \frac{R_{12}}{R_{11}}$ $C_{21} = C_{11} \left(\frac{R_{11}}{R_{12} + R_{11}} \right)^2$
$Z_1(\omega) = R_{12} + \frac{1}{\frac{1}{R_{11}} + C_{11}(i\omega)}$		$Z_2(\omega) = \frac{1}{\frac{1}{R_{21}} + \frac{1}{R_{22} + \frac{1}{C_{21}(i\omega)}}$

Figure 2.1 Example of one EC (left, original) that can mimic another's impedance spectrum (right, converted) by using different values for the parameters. The impedance formulas are shown under the ECs.

Component Name	Circuit Representation	Formula
Resistor		$Z_{\text{ohm}}(\omega) = R$
Capacitor		$Z_{\text{cap}}(\omega) = \frac{1}{i\omega C}$

Figure 2.2 A reference for the impedance formulas of a resistor and a capacitor, respectively.

By tracking the impedance of a given electrochemical cell through time, charge-discharge cycle number, cell potential, etc., a dynamic characterization of the cell could be obtained. Yet, different models might produce very different trends. Figure 2.3 shows an example of this phenomenon based on the conversion in Figure 2.1. On the left side, the original EC of Figure 2.1 has three

parameters, and all parameters except one are kept constant. R12 is increased linearly in 1 Ω steps from 1 Ω to 10 Ω . All the parameters of the two ECs are plotted as a function of R12. On the right side of Figure 2.3, the “converted” EC parameters are plotted against the original value of R12.

As Figure 2.3 shows, the trends in the converted parameters are more complicated. Indeed, all three parameters change, most trends are non-linear, and even though the capacitance does not vary in the first EC, it has a very dramatic variation in the second.

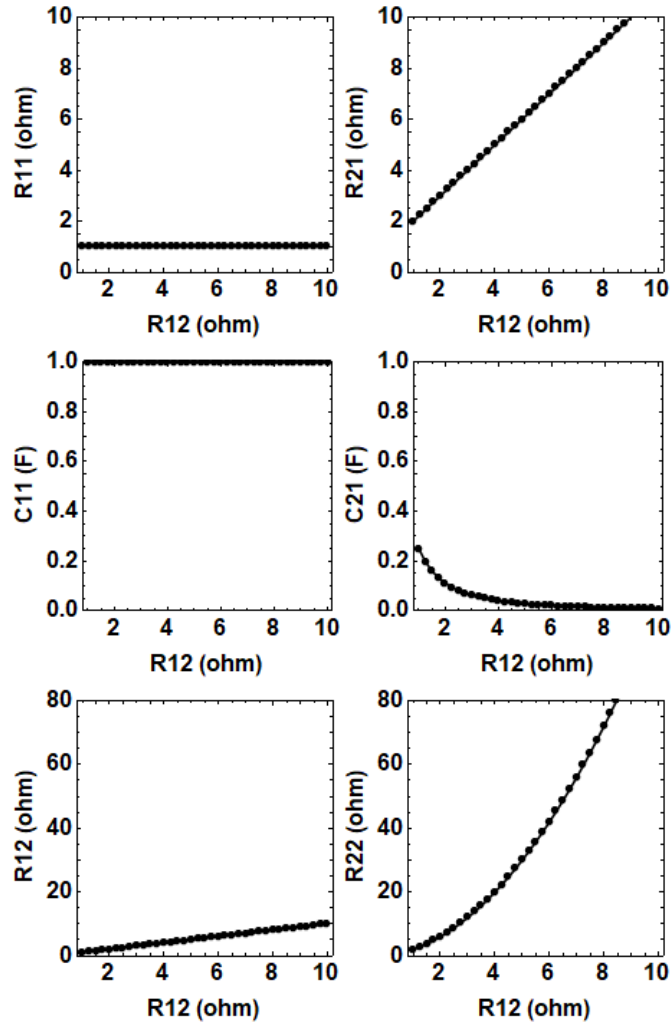


Figure 2.3 Example of how a simple trend in one EC (original EC of Figure 2.1) looks quite different and more complicated in another EC (converted EC of Figure 2.1), even if the underlying impedance spectra are exactly the same.

Looking at Figure 2.1 and Figure 2.3, it is natural to wonder if there are some limits on the conversion formulas. In other words, are there any parameter choices in the original EC which, after applying the conversion formula, would not yield the same impedance spectrum?

As it turns out, the only case where this happens is if the formulas require a division by zero. In the case of Figure 2.1, this would correspond to cases where R_{11} is zero or $R_{11} + R_{12}$ is zero. Assuming that resistances are positive, this corresponds to cases where R_{11} is zero. For more complicated formulas the story is the same. If the conversion formula requires division by zero, then the ECs most likely cannot give identical spectra. In the exceptional case that the ECs can give identical spectra with the problematic parameters, then a different formula would be required. Looking at Figure 2.1, one might get the impression that conversions, either implicit or explicit, only work for simple ECs. However, this is not the case. To illustrate this, Figure 2.4 shows two complicated ECs which look quite different, and whose topologies are quite different. Despite this, one can explicitly convert from the upper EC to the lower based on the formulas described in the paper³⁸.

To keep the focus of the thesis on developing robust machine learning models in the context of lithium-ion research, most of the details present in the paper are omitted here. But here follows a sketch of the construction of the formulas:

- The various core components of all the circuits are represented in a unified form.
- Formulas are given to convert *to* this unified form *from* various circuits of interest.
- Formulas are given to convert *to* various circuits of interest *from* this unified form.
- This creates a graph of conversions where each node is an EC and each arrow is an elementary conversion. Then, for various common cases, the conversions are composed (i.e. applied one after the other) to create a simple program to convert between ECs of interest, such as in Figure 2.4

Figure 2.4 uses the code to convert from a Voight configuration (with a series capacitor) to a ladder configuration (with a capacitor embedded in the inner part of the ladder). Figure 2.4c) shows the two circuits considered. Concretely, Figure 2.4 shows 3 examples of conversion. For each example, parameter values were chosen for the Voight configuration, and the impedance spectrum was computed and plotted. Next, the circuit was converted to the ladder configuration, and the impedance spectrum computed and plotted again for the new configuration. The impedance curves from the two configurations are on top of each other in the figure. For simplicity, the parameter values of the Voight configuration are all the same in the three examples except for the series capacitor, which took on the values, 10, 100, and 1000 F, respectively.

Figure 2.4a) shows the impedance spectra. From left to right, Figure 2.4 shows a Nyquist plot, then the real Bode plot, then the imaginary Bode plot. The different grayscale colors correspond to different initial values for the series capacitor.

Figure 2.4b) lists the parameter values before and after conversion. Each row corresponds to a set of parameter values for a given circuit. The first three rows are the original parameter values for the Voight configuration, and the last three rows are the corresponding parameter values for the ladder configuration. The differences before/after conversion are relatively small when the series capacitance is large, but these differences are very large when the series capacitance is small. This phenomenon is related to the overlap between different subcomponents in frequency space (little overlap when C is large; much overlap when C is small).

Note that, despite having the same spectra, and therefore the same relaxation times, the two ECs show very different relationships between the capacitances and resistances. In a Voight-type circuit, it is easy to find the various time constants that determine the relaxation behavior by simply computing $R_i C_i$. But the same approach would yield different results for the ladder-type circuit.

This highlights that the physical meaning cannot be the same for the corresponding components in the two circuits. More concretely, the same observation, if we assume that it comes from a series of successive electrochemical processes in the cell, will give a picture very different than if we assume that the chain of electrochemical processes is embedded in a ladder configuration.

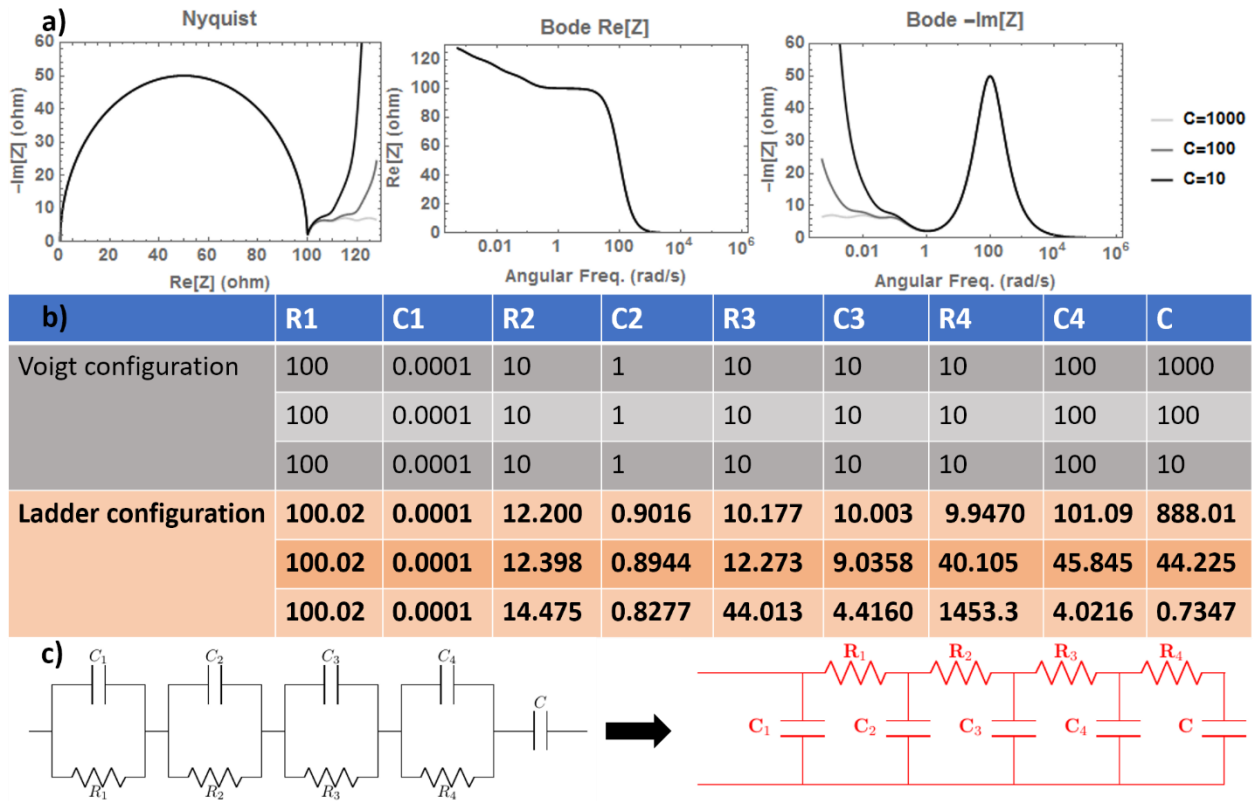


Figure 2.4 An example of explicit conversion from a Voigt-type circuit configuration to a Ladder-type circuit configuration.

2.3 A Realistic EC for Lithium-ion Cells

Though it would be possible to carry the full discussion without ever mentioning a specific choice of EC model, we give a reasonable one here before proceeding with the full discussion. However, see Section 3.6 to see how the results can be extended to encompass various other reasonable choices.

First, we describe the data itself. About 100000 individual impedance spectra were collected from various lithium-ion chemistries, using more than 4 distinct experimental setups^{1,41}. Approximately 90000 spectra were collected automatically for various cells at various **cycle numbers** and various **voltages** (a typical cell producing more than 100 spectra across its lifetime). These systems were run **at 20 and 40 degrees Celsius**. This dataset will be referred to as the **FRA dataset**.

Also, approximately 10000 spectra were collected **manually** for various cells, including **pouch** cells, **coin** cells, cells with both a positive and a negative electrode (**full cells**), cells with two negative electrodes (**symmetric negative cells**), cells with two positive electrodes (**symmetric positive cells**), at various temperatures ranging **from -10 to 40 degrees Celsius**. A typical cell would have less than 10 spectra measured on this system. This dataset will be referred to as the **EIS dataset**.

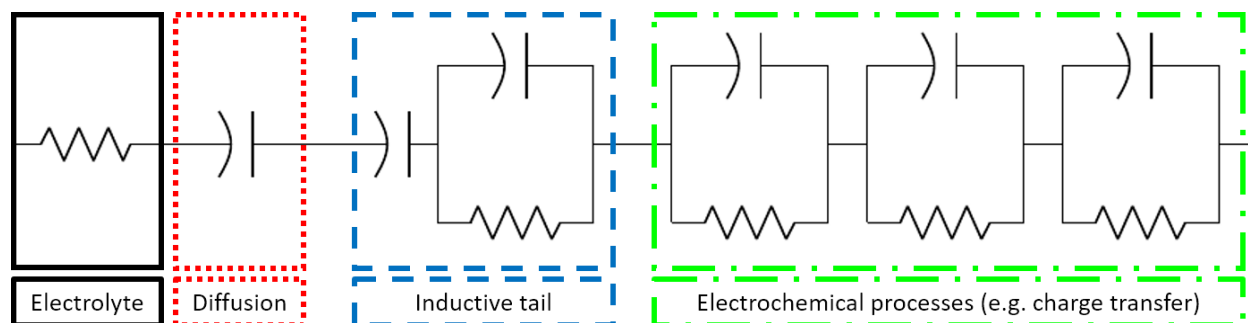


Figure 2.5 The EC model used to fit the spectra. The various sub-ECs connected serially are framed with different colors. From left to right, we have: “Electrolyte”, a series resistor, to model the electrolyte resistance. “Diffusion”, a Constant Phase Element (CPE) representing diffusion; “Inductive tail”, a CPE representing an imperfect inductance and a Resistor-CPE in parallel, also known as ZARC, representing an imperfect inductance with a finite time-constant. Finally, “Electrochemical processes”, three ZARCs each representing an electrochemical process giving rise to a relatively sharp distribution of time-constants.

Given this large dataset, a single EC model was chosen to fit all the spectra. Figure 2.5 illustrates this EC and Figure 2.6 defines the various components mathematically, namely the resistor, the constant phase element (CPE), and the ZARC (literally stands for impedance which looks like an “arc”). Note that similar ECs to the one studied in this paper have been used in the literature⁴².

Note that a CPE can represent various more traditional components when the exponent φ is fixed to certain values. For instance, with an exponent of 1, we obtain an ideal capacitor; with an exponent of 0.5, we obtain a Warburg element; with an exponent of -1, we obtain an ideal inductance. Also note that a ZARC element can represent a resistor in parallel with a capacitor

(when the exponent is 1). For such an EC, there is a frequency (characteristic frequency) above which the current primarily flows through the capacitor, and below which it flows primarily through the resistor. In a lithium-ion cell, the same particle is repeated across an electrode, with fluctuations in the local shape, composition, and even currents. Therefore, it is more typical to see a distribution over a range of characteristic frequencies. This is what is modelled by a ZARC element with an exponent less than 1. The **lower** the exponent, the **broader** the distribution.

As presented, only three different components intervene in this EC model. For instance, the **inductance** and the **diffusion** are **both** modelled by a CPE. This is possible since the exponent (denoted by φ in Figure 2.6) of the diffusion CPE must take values between 0 and 1, and typically will be around 0.5, whereas the exponent of the inductive CPE must take values between -1 and 0, and will be approximately -1. Similarly, the exponent of the electrochemical ZARCs must be between 0 and 1, with typical values between 0.6 and 0.9. These constraints will be generalized below to the notion of “prior knowledge” on the values of the parameters. This leads to the problem of enforcing these constraints.



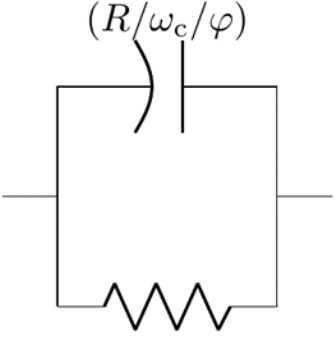
Component Name	Circuit Representation	Formula
Resistor		$Z_{\text{ohm}}(\omega) = R$
CPE		$Z_{\text{CPE}}(\omega) = \frac{1}{Q(i\omega)^\varphi}$
ZARC		$Z_{\text{ZARC}}(\omega) = \frac{R}{1 + (i\frac{\omega}{\omega_c})^\varphi}$

Figure 2.6 The EC model components used in this section. The formulas for the impedance are given in terms of the angular frequency ω of oscillation of the voltage signal applied to the terminals of the sub-ECs.

2.4 Enforcing Constraints on EC Parameters

As a sub-problem of the inference problem, the *fitting problem* is to find the EC parameters θ_{EC} which minimize the mean squared error of reconstruction (see Section 2.1). However, only a restricted set of possible θ_{EC} may be considered. As discussed earlier, the exponents of the inductive CPE and ZARC must be between -1 and 0 (**negative**), while the exponents of the diffusion CPE and electrochemical ZARCs must be between 0 and 1 (**positive**). Furthermore, all

resistances must be **positive or zero**, while all characteristic frequencies, all frequencies, and the Q parameters must be greater than 0 (**positive**).

It is possible to enforce these constraints without having to deal with constrained minimization^{7,43}, namely by rewriting the formulas in a way that has no constraints but is equivalent to the original way. More precisely, the EC model is reparameterized such that 1) the **same set** of spectra can be modelled, 2) the original parameters can easily be **recovered** from the reparameterized version, and 3) **any** value for the new parameters leads to a **valid** spectrum (i.e. respects the constraints on the original parameters).

Figure 2.7 shows the reparameterizations used in this Chapter as well as Chapter 3.

Note that the log scale version of R , Q , frequency, and characteristic frequency is used. Exponentiating ensures positive values. For the exponents φ , we use the logistic⁴⁴ function σ for Diffusion and Electrochemical, which always is more than 0 and less than 1. This function also has nice derivatives around an output of 0.5 (where the derivative is 1), and the derivatives are still large enough until we reach an output of 0.9. However, this function's derivative vanishes as the output approaches 0 or 1. In contrast, the function used for the exponent of the inductance CPE⁴⁵ and Inductance ZARC has good derivatives everywhere (including for outputs near -1), but it isn't invertible (indeed, it is a symmetric function around its origin).

Note that the choice made here might not be the optimal choice⁴⁶. Yet, it satisfies the above goals and is easily implemented in code. When applying this methodology to a different fitting problem, choosing a good reparameterization might require some trial and error. A special care should be taken with the derivative of the model with respect to the new parameters, as this might affect the ability of the neural network to converge to a good "inverse model" (to be defined later). In this

case, the first reparameterization considered was adequate, and generally ease of implementation and conceptual simplicity are also important factors in the choice.


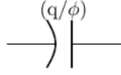
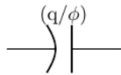
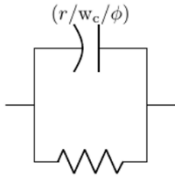
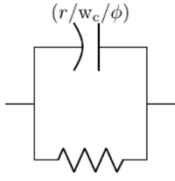
Component Name	Circuit Representation	Conversion Rules	New Formula
Electrolyte Resistor		$R = \exp(r)$ $\omega = \exp(w)$	$Z_{ohm}(w) = \exp(r)$
Diffusion CPE		$Q = \frac{1}{\exp(q)}$ $\varphi = \frac{1}{1 + \exp(-\phi)} = \sigma(\phi)$ $\omega = \exp(w)$	$Z_{CPE,diffusion}(w) = \exp\left(-i\sigma(\phi)\frac{\pi}{2} + q - w\sigma(\phi)\right)$
Inductance CPE		$Q = \frac{1}{\exp(q)}$ $\varphi = -\frac{1}{1 + \phi^2}$ $\omega = \exp(w)$	$Z_{CPE,inductance}(w) = \exp\left(i\frac{1}{1 + \phi^2}\frac{\pi}{2} + q + \frac{w}{1 + \phi^2}\right)$
Electrochemical ZARC		$R = \exp(r)$ $\varphi = \frac{1}{1 + \exp(-\phi)} = \sigma(\phi)$ $\omega_c = \exp(w_c)$ $\omega = \exp(w)$	$Z_{ZARC,electrochem}(w) = \frac{\exp(r)}{1 + \exp\left(i\frac{\pi}{2}\sigma(\phi) + (w - w_c)\sigma(\phi)\right)}$
Inductance ZARC		$R = \exp(r)$ $\varphi = -\frac{1}{1 + \phi^2}$ $\omega_c = \exp(w_c)$ $\omega = \exp(w)$	$Z_{ZARC,inductance}(w) = \frac{\exp(r)}{1 + \exp\left(-i\frac{\pi}{2}\frac{1}{1 + \phi^2} - (w - w_c)\frac{1}{1 + \phi^2}\right)}$

Figure 2.7 The reparameterizations used in this thesis. For each EC component (under “Component Name”), the new parameters are enumerated together with a visual representation of the EC component (under “Circuit Representation”), and the conversion formulas connecting these new parameters to the old are given under “Conversion Rules”, with the new impedance formula given under “New Formula”.

2.5 Symmetries within a single EC Model

In order to illustrate the process of creating a robust solution, it is useful to consider some further mathematical properties of the EC model and corresponding EIS, which will then be exploited as

part of the solution: transformations of the EC parameters which lead to simple transformations of the spectra and vice versa.

The transformations work as follows: given a spectrum and its associated optimal EC parameters, apply special transformations to both the spectrum and the EC parameters to obtain a new spectrum and the optimal EC parameters for the new spectrum.

Thus, we can scale and shift the spectra to ensure that the inverse model need only be applied to spectra with frequencies centered around a log frequency of 0 and with impedances within the complex unit circle. It may not seem like much, but this is a substantial factor in the robustness of the solution.

Figure 2.8 gives the details. If the optimal parameters are transformed according to Figure 2.8, then they will remain the optimal parameters for the transformed spectrum. The point of these transformations is that it is easy to undo them, as long as the scale and shift parameters are recorded. (We can simply apply the transformations with the negative of the scale and shift parameters to undo the transformation). Note that in the case of CPE, the conversion rule is given in terms of the original exponent φ so that the formula would be the same for both the diffusive CPE and the impedance CPE. It should be understood that in both cases, the original exponent should be computed from the reparameterized exponent.

As a future reference, a translation of w_α of the spectra is denoted as $\text{Trans}_{w_\alpha}(\{(w_i, Z_i) | i = 1, \dots, m\}) = \{(w_i + w_\alpha, Z_i) | i = 1, \dots, m\}$ and the corresponding action on the EC parameters is denoted as $\text{Trans}_{w_\alpha}(\theta_{\text{EC}})$ (the specific transformation should be clear based on the context).

Similarly, a scaling of $\exp r_\alpha$ is denoted as

$$\text{Scale}_{r_\alpha}(\{(w_i, Z_i) | i = 1, \dots, m\}) = \{(w_i, \exp r_\alpha Z_i) | i = 1, \dots, m\}$$

and the corresponding action on the EC parameters is denoted as $\text{Scale}_{r_\alpha}(\theta_{\text{EC}})$.


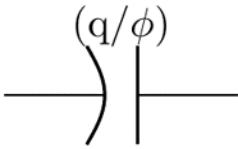
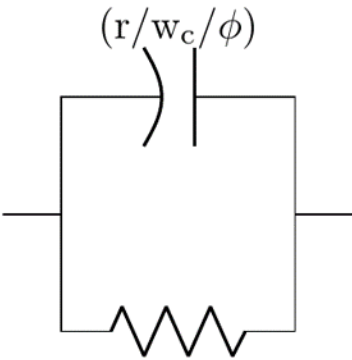
Component Name	Original Representation	Conversion Rules
Spectrum	$(w_i, Z_i)_{i=1}^N$	$w_i \leftarrow w_i + w_\alpha$ $Z_i \leftarrow Z_i \exp(r_\alpha)$
Resistor		$r \leftarrow r + r_\alpha$
CPE		$q \leftarrow q + r_\alpha + w_\alpha \varphi$ $\phi \leftarrow \phi$
ZARC		$r \leftarrow r + r_\alpha$ $w_c \leftarrow w_c + w_\alpha$ $\phi \leftarrow \phi$

Figure 2.8 A useful symmetry of the EC model. It is possible to shift the log-frequencies of the observed spectrum, and scale the observed impedances. For each component of the EC model, the parameters of Figure 2.7 are shown. Then, the log-frequency shift parameter w_α and the log-resistance scale parameter r_α are applied to transform **the original spectrum** according to the first row of the Figure. This transformation corresponds to transformations of the **EC parameters** given under “Conversion Rules”. For instance, as the impedance is scaled by $\exp(r_\alpha)$, the log-resistance of the series resistor must be changed to $r + r_\alpha$.

2.6 The Difficulty of Choosing EC Model Parameters

Uniquely

Section 2.2 discussed how essentially the choice of EC models acts as the choice of “coordinate system”, with well-defined conversion rules when switching between “coordinate-systems”. No one would be tempted to look for trends in the position of a particle with each time step given in a randomly chosen coordinate system. Similarly, if trends in the EC parameters of the fitted EIS of a cell are to be investigated, then a single EC model must be chosen for the investigation. Things will look different for different choices, but there is still a chance of finding a reasonable interpretation, since the conversion rules between EC models are understood (see Section 2.2).

Yet, this does not exhaust the scope of the problem. Indeed, for a given spectrum and a given EC model, there may still be multiple EC parameters which have approximately the same mean squared reconstruction error.

When considering multiple EIS and the trends which may exist across them but *individually* choosing among the multiple EC parameters fitting equally well each spectrum, trends can be obscured.

The simplest trend possible (i.e. EIS more or less constant across measurements) illustrates well the phenomenon. If several EIS are nearly identical, but each fitting problem makes a different choice among the multiple good EC parameters, then each individual number in θ_{EC} may appear to change, oscillate, fluctuate randomly, etc.. It is very difficult to notice even the *constant trend* if the individual fitting problems are not coordinated in some way. Some examples of these multiple choices follow shortly.

Note also, that what is really desired is a *global coordination* such that across hundreds of thousands of spectra measured and fitted at different times in different labs by different people, the choices of parameters have been coordinated to at least make visible the constant trends, and hopefully many other common trends as well. This global coordination cannot be added artificially in practice since it would be too computationally expensive to solve the fitting problem anew for all existing EIS every time a new EIS was measured.

Formally speaking, global coordination relative to the constant trend can be stated as follows: the process which assigns a choice θ_{EC}^* to a given EIS $\{(\omega_i, Z_i) | i = 1, \dots, m\}$ must be deterministic. In other words, there must exist **a function** from EIS to EC parameters which perfectly reproduces the solutions to the fitting problem.

Intuitively, other trends should also be respected by the solution to the fitting problem. For instance, *small* changes to the EIS should lead to *small* changes to θ_{EC}^* (sometimes known as *continuity*).

Despite the fact that most physicists are not accustomed to taking “derivatives” of functions from indexed sets to vectors, for appropriately defined versions of notions like the *Jacobian* and the *Hessian* of such functions, the intuitive picture of what a global coordination on *simple* trends would look like is faithfully captured by saying that the averaged norm of these measures of slope (Jacobian) and curvature (Hessian of every individual parameter) should be minimized across almost every plausible EIS. (See Section 1.1.2 for definitions of Jacobians and Hessians.)

In case this principle of encoding trends as penalties on slopes and curvatures is of interest, a sketch of them would be that for any EIS $\{(\omega_i, Z_i) | i = 1, \dots, m\}$, the EIS can be thought of as a vector in \mathbb{R}^{3m} (for each i , one number for frequency, one for the real part of Z_i and one for the imaginary

part), and the output of the function θ_{EC}^* is a vector in \mathbb{R}^n . On this EIS, the Jacobian is the matrix of partial derivatives of each EC parameter with respect to each input real value, and the norm squared for instance can be the squared sum of all entries in the matrix (a single number per EIS). One could also take all the second partial derivatives of every output component with respect to every input real number, for which the norm could be computed in a similar way (yielding a single number per EIS). Then, the average of these norms across a set of EIS can be minimized, thus also minimizing the underlying average norms. Reference ²⁵ also mentions some tricks to reduce the computational requirements for the minimization.

To conclude this section, some examples of multiple choices for θ_{EC}^* will be discussed.

First, under a limited range of observed frequencies, a ZARC element may either have no contribution to the circuit impedance, or it may be essentially equivalent to a resistor in series with the other components. Indeed, when all observed frequencies are far below the characteristic frequency of a given ZARC, its contribution to the overall impedance is well approximated by a resistor, while far above the characteristic frequency, it has no contribution. Therefore, for θ_{EC}^* with a ZARC having a very low characteristic frequency, there is a family of other possible θ_{EC}^* which would fit the data equally well, by increasing the value of the series resistor and decreasing the value of the low frequency ZARC's resistor, or vice-versa.

A second example which is a bit more worrisome in practice is that two ZARC sub circuits may be interchanged without changing the resulting impedance. Indeed, any permutation of EC parameters corresponding to permuting the order of the ZARCs in the circuit will constitute equivalent choices in terms of mean squared error of reconstruction.

Finally, the main concern is that ZARC elements are essentially a distribution over Voigt elements⁴⁷. In other words, the impedance of a single ZARC can be rewritten as an integral over characteristic frequencies of ZARCs with an exponents equal to 1 (equivalent to a resistor in parallel with a capacitor). The exact formula is not important, but for those interested, it has a hyperbolic sequest shape, namely $\frac{1}{e^x + e^{-x}}$, in the space of logarithmic frequencies, and is similar in appearance to a gaussian though it decays slower away from the peak⁴⁸.

The important part is that a ZARC is essentially a “peak” with the broadness of the peak determined by the exponent, and similarly to the case of gaussian peaks, **a single broad peak can often be approximated by two or more narrower peaks**. Since typical EIS data contain two or more overlapping such peaks, there are many cases where the noise in the data creates ambiguities in the number of peaks and their proportions.

Yet with all the possible choices of parameters which are in some sense equally good, but which must be coordinated globally, there are also choices which solve the fitting problem equally well but which can in fact be shown to be inferior based on other considerations, which is the subject of the next section.

2.7 A Simple Measure of EC Parameter Complexity

Following the principle that the simplest explanation is the most likely, if a single electrochemical ZARC accurately reconstructs the observed spectrum, then a fit using a single ZARC is preferred to a fit using three ZARCs, even if the three ZARCs can fit the data slightly better.

This can be understood as the principle that the simplest explanation has the highest likelihood to be physically relevant. Indeed, having several ZARCs in the idealized model of the physical system coincide in their characteristic frequencies to appear as a single ZARC is less likely

(requires more coincidences) than a single ZARC accounting for the observations. The difference in likelihood is not astronomical and so a significant improvement in the mean squared error of reconstruction can overcome the simplicity consideration.

Having a numerical metric to track the “simplicity” or “complexity” of a given circuit is quite useful to incorporate this consideration into algorithms, but how to do it?

In our model, setting a ZARC resistor to 0 is equivalent to removing that component from the circuit. Let $C(R_1, R_2, R_3) = \frac{(\sqrt{R_1} + \sqrt{R_2} + \sqrt{R_3})^2}{R_1 + R_2 + R_3}$ which we call the complexity metric. This function is related to counting the number of non-zero resistances. For instance, we can see that $C(R, 0, 0) = C(0, R, 0) = C(0, 0, R) = 1$, $C(R, R, 0) = C(R, 0, R) = C(0, R, R) = 2$, and $C(R, R, R) = 3$ for any positive value of R . More generally, the complexity metric measures the degree to which the sum of resistances is spread across all three resistances. As defined, the complexity metric is related to some well studied mathematical objects. Namely, it is also the quotient of the “ $l_{1/2}$ pseudo-norm” and the “ l_1 norm.” Such norms are known to induce sparsity when used as penalty⁴⁹ and usually, simply using the “ l_1 norm” would work. However, in our application, the “ l_1 norm” is the sum of resistances across the ZARCs which is conserved in the context of an infinite span of measured frequencies no matter how many ZARCs are used due to the fact that it represents the difference between the real part of impedance in the low frequency and the high frequency limit for the collection of ZARC components, so the “ $l_{1/2}$ pseudo-norm” is used instead, and we divide by the “ l_1 norm” in order to properly compare the complexity of spectra with different scales. Note that, in the case where a ZARC has a characteristic frequency far above or below the observed frequencies, the “ l_1 norm” is not conserved, so **we penalize** the “ l_1 norm” $R_1 + R_2 + R_3$ directly as well as the “ $l_{1/2}$ pseudo-norm” $(\sqrt{R_1} + \sqrt{R_2} + \sqrt{R_3})^2$.

In conclusion, the “ l_1 norm” and the “ $l_{1/2}$ pseudo-norm” are tools to distinguish between solutions to the fitting problem with comparable mean squared error of reconstruction, and can readily be applied in practice. Furthermore, the complexity metric is a human friendly metric to assess the relative quality of a set of solutions to the fitting problem not explained by a difference in mean squared error of reconstruction.

Armed with these conceptual tools, we next describe the actual solution to the inference problem.

2.8 Acknowledgements

This work was supported financially by the Natural Sciences and Engineering Research Council of Canada (NSERC), and Tesla Motors.

Chapter 3 Robust Fitting of Lithium-ion Cell EIS to EC Models

Easy collection of electrochemical impedance spectra (EIS) at various cycle numbers and various state of charges produce vast amounts of data. The fitting problem, i.e. fitting each spectrum to an equivalent circuit (EC) can lead to physical insights about the evolution of the lithium-ion cell, but it requires good human initial guesses for the EC parameters to reliably converge, making the fitting process labor intensive and difficult to scale. This chapter presents a paradigm to automate the fitting of measured data to physical models, replacing the good human first guesses with an *inverse model* parametrized with an *artificial neural network*. This method is simple to implement, uses principles applicable to a wide variety of fitting problems, and leads to reliable and accurate initial guesses of the EC parameters for a given spectrum. The performance of the system is evaluated on a dataset of about 100000 impedance spectra from lithium-ion cells, achieving a failure of fitting approximately 1% of the dataset, corresponding to the percentage of poor quality data in the dataset.

This chapter is taken in whole from the corresponding paper³⁹, except for some minor reformatting and refactoring into Chapter 2. However, more details are given about the neural network implementation, and the work is expanded to demonstrate how a single neural network can produce good initial guesses for various EC topologies of interest.

3.1 Introduction

As it turns out, for many EC models encountered in the literature, the problem of fitting is difficult. It is a difficult optimization problem for which the typical optimization algorithms are often sensitive to the starting point, and sometimes prone to divergence.

In practice, this means a human has to choose the initial values of the parameters of the EC model, and verify the output to make sure the fitting converged properly. Thus, this analysis method is expensive to use, hard to automate and repeat, and, therefore, less ubiquitous than it otherwise would be.

The key contributions of Chapter 3 can be summarized as:

1. Converting 100000 separate optimization problems into a **single optimization problem** to which human effort, tuning, and (potentially) ad-hoc solutions can be applied and validated.
2. Applying various insights from the large-scale machine learning literature to solve the single optimization problem.
3. A complete open-source solution (with and without a graphical user interface) to the fitting system which can be easily adapted to various impedance fitting workloads, and finally
4. Formulating the problem such that the insights can be generalized to potentially any other problems of fitting the parameters of a physics model to measured data reliably.

As an overview,

- In Section 3.2, the fitting problem is formalized through various approaches. Among these are mentioned individual fitting, clustering of spectra combined with human interaction, a supervised learning approach, and finally the more successful and principled approach of unsupervised learning.
- In Section 3.3, the various approaches are compared both at an abstract level and in terms of the challenges of implementation.

- In Section 3.4, the unsupervised approach is put to the test on a large dataset, conclusively demonstrating the robustness and power of this approach.
- In Section 3.5, the various tricks and practical considerations which can be discussed at a theoretical level are explored.
- In Section 3.6, the ideas are extended to allow a single program to simultaneously solve the fitting problem for a variety of relevant EC models.
- In Section 3.7, the precise details of the neural network⁷ having a material impact on performance are discussed to hopefully help the reader make good choices on their own application domains.
- In Section 3.8, we present future work by discussing ways of making the core model more powerful (using the Transformer architecture).

3.1.1 Related Works

Though the core ideas of this paper have not been applied to electrochemical impedance spectroscopy before, there are some related works.

For some EC models, some reparameterizations (the same model written with a different mathematical formula) have been shown to reduce the impact of poor initial guesses⁵⁰. Similarly, in an application, the actual physical model has been approximated with a simpler “empirical” model in order to make fitting feasible⁵¹.

Also, in order to stabilize fitting and impose some *a priori* preferences for ‘sensible parameter values’, a so-called “prior distribution of the model parameters” has been added to the fitting problem, penalizing both the fit error and the deviation from the *a priori* sensible parameter values⁵². The optimization problem resulting from the combination of *a priori* preferences and

fitting of measured data is well studied and also called “Bayesian” or “maximum *a posteriori*” parameter estimation.

Furthermore, the use of machine learning to predict the values of EC model parameters (shortened to “EC parameters” from now on) is not new. For instance, there has been an application where design parameters for a physical system were linked directly to values for an EC model⁵³. In other words, the conversion from a detailed model of a physical system to a simpler EC model was accomplished using a neural network.

Finally, perhaps the closest related work conceptually, is that of visual estimation of the EC parameters^{54,55} where a set of simple rules is developed to estimate a subset of the EC parameters by first visualizing the impedance spectrum, and for instance, measuring the high frequency limit of the real part of the impedance to estimate a series resistor. By formalizing these rules into a program, one gets automated initial guesses for a subset of the EC parameters, which can be used to initialize the fitting procedure (in our terminology, we would call such a program a “partial inverse model”). However, developing these rules can be tedious and error prone, and requires special knowledge of the EC model equations. For instance, if one adds a single component to an existing EC model, the whole set of rules might have to change, not to mention that special care has to be taken to make the rules robust to unforeseen special cases such as noisy spectra. In contrast, we develop a general method, which can be applied to a wide variety of fitting problems (i.e. not limited to EIS) and can produce a robust set of rules to estimate all the EC parameters based on the raw impedance spectrum. Indeed, the estimates thus produced, even for the complicated EC model used, are robust and accurate, even removing the need for a separate fitting procedure on a significant percentage of the spectra.

3.2 Formalizations of the Fitting Problem

Given an observed impedance spectrum, an optimization algorithm can be applied to the constrained fitting problem, to produce a set of EC parameters θ_{EC}^* minimizing the mean squared error of reconstruction. Therefore, any deterministic optimization algorithm defines a function from spectrum to EC parameters. We call such a function an “inverse model”, since the EC model goes from parameters to spectra, and this function goes in the “inverse” direction. Hopefully, the EC parameters thus produced will be close to optimal, leading to a good fit of the data. If this is reliably the case, across the set of interesting spectra, we say that the inverse model is a “good inverse model.” In practice, for the EC model shown in Figure 2.5, this requires a human in the loop, to find initial guesses for the EC parameters, and potentially restart the procedure.

Figure 3.1 illustrates how the various elements of the fitting problem relate to each other. Given some EC parameters (and a set of frequencies), a synthetic impedance spectrum can be sampled (this spectrum is represented by $Z_{reconstructed}(\omega)$ in Figure 3.1). Given some measured impedance spectrum and a synthetic spectrum, the mean squared error of reconstruction may be computed. These are represented by solid arrows since they can be directly expressed as equations. Furthermore, given a measured spectrum, the previous two relations allow one to choose a set of parameters, and this choice (the output of the inverse model) is a dashed arrow since it cannot easily be given as an equation. The colors and shapes simply represent the fact that $Z_{measured}(\omega)$ is observed while θ_{EC} is not directly observed (unobserved variables are called *latent variables*).

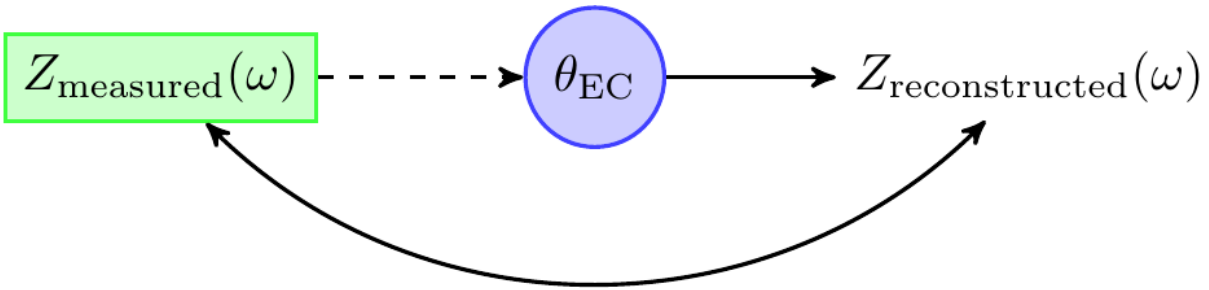


Figure 3.1 A graphical illustration of the fitting problem's structure.

Based on Figure 3.1, there are still various distinct ways of formalizing the idea that “the mean squared error of reconstruction is the criterion for the inverse model.”

- **Individual Fitting (Human in the Loop).** The usual way, often known as least squares fitting, is to say that the inverse model is implicit. Any given spectrum will constitute a *separate* optimization problem of minimizing the mean squared error of reconstruction using a given optimizer and initial parameter guesses provided by a human in the loop. Note that this formalization does not make the inverse model a deterministic function of the measured spectrum, and any properties related to trends are left for the human to enforce. Also note that poor typical initial guesses create the need to use a powerful optimizer. This typically comes at the cost of being prone to divergence.
- **Human-Augmented Individual Fitting (Clustering).** As a historical point of interest, the first attempt by the author to improve trends and reduce the work of fitting 100000 EIS consisted of a distance metric over impedance spectra which was used to group 30 to 60 spectra together for the purpose of choosing the same initial guesses for multiple fits. This was combined with one of the two transformations discussed in Section 2.6 in order to further reduce the differences between spectra in a group. The spectra were then plotted together and initial guesses were given according to visual heuristics^{54,55}. Note that

estimating only a few key parameters (in this case, mostly the characteristic frequencies and resistances of the ZARC elements) is enough to make a reasonable optimizer behave more or less deterministically.

- **Supervised Learning of a Partial Inverse Model (No Humans).** Based on the insight that a human could visually estimate a few key parameters which is a process called a *partial inverse model* (in this case, mostly the characteristic frequencies and resistances of the ZARC elements), it was decided to replace the human by a neural network. The most common approach to training such networks is to create a dataset of pairs $(x_i, y_i)_{i=1}^N$ of inputs and outputs and then train a function $f(x) = y$ to produce the outputs when given the inputs. In this case, the dataset was generated by first selecting θ_{EC} at random, then sampling $Z_{reconstructed}(\omega)$ based on some randomly chosen frequencies. In the above notation, the input x was a visual representation of $Z_{reconstructed}(\omega)$ and the output y was a vector of the few key parameters mentioned earlier, all extracted from the randomly selected θ_{EC} . In other words, a dataset was created where the θ_{EC} were known (not latent), and a neural network was trained to reproduce these values from the dataset. Note that this is not the same as minimising the mean squared error of reconstruction. The error is computed directly on the EC parameters instead. Informally, we would say that the error “has units of EC parameters θ_{EC} rather than units of impedance $Z_{reconstructed}(\omega)$.”
- **Unsupervised Learning of an Inverse Model.** All previous approaches have major setbacks which are explored in Section 3.3. This leads one to ask: *What is the actual problem of interest and is there a way to pursue that more directly?* The problem is to find an inverse model (i.e. a function from measured spectra to EC parameters) which is best according to some criterion. As such, it is itself a single optimization problem. The criterion

is as follows: For the distribution of impedance spectra encountered in practice (in this case, they will be from lithium-ion cells), the best inverse model is *that which minimizes the average mean squared error of reconstruction over these spectra and which behaves best with respect to simple trends of interest*. Note that this can be implemented on a dataset of actual measured impedance spectra, but spectra generated some other way (such as those used for the supervised approach) could also be used. Indeed, by connecting a neural network to an implementation of the sampling of EC parameters, as in Figure 3.1, one obtains a function which takes impedance spectra as input and returns impedance spectra as outputs. The “right” choices of EC parameters for the spectra in a dataset need not be known for the optimization over possible inverse models to be performed.

3.3 A Comparison of Various Approaches to the Fitting

Problem

At a conceptual level, there is only one problem (finding a good inverse model according the various criteria) and pursuing this as directly as possible (using the unsupervised learning approach) has the advantage of elegance. However, understanding the actual trade-offs between the various approaches is instructive, useful, and sometimes the most elegant approach is not the most practical (though it definitely is practical in this case).

3.3.1 Individual Fitting

First, consider the **Individual Fitting** approach. In theory, it confers the following:

- **Flexibility.** If the EC model is changed, an optimization algorithm can immediately be used to fit spectra to the new EC model. For both the supervised and unsupervised

approach, the whole multi-day process of finding a good neural network must be redone for each new EC model (though we note that Section 3.6 overcomes this to some extent).

- **Adaptability.** A very unusual spectrum might benefit from the fact that the optimization algorithm treats each spectrum separately. By default, an inverse model trained on a fixed set of spectra has no defined behavior when applied to a spectrum very different from those in that fixed set used for training. This problem is mitigated by the fact that a very unusual spectrum occurs rarely and can be treated separately. Also, unless the optimization algorithm is of a special kind, there is no convergence guarantee for any spectrum, hence no guarantees for very unusual spectra. Similarly, if the kind of spectra in need of analysis changes significantly over time (for instance, if a laboratory changes their research focus), the criterion at the time the model is used will diverge from the criterion at the time the model is trained. However, this issue is solved by periodically reoptimizing the neural network using all the data available, including the newer spectra, or by somehow having a robust enough selection process such that the inverse model chosen is good over a much wider distribution of possible spectra than those available during training.
- **Component Simplicity.** The general software complexity of setting up and maintaining an inverse model is significant compared with simply using an off-the-shelf fitting software for individual fitting. If such off-the-shelf software gave equal or better fit quality without other software components required, it would be a better practical solution. However, ensuring quality solutions when using individual fitting may require many more components and process complexity in practice (i.e. human-in-the-loop).

On the bad side, individual fitting as an optimization problem on the EC model of Section 2.3 is plagued with bad local minima and as a solution to the inference problem described in Section 2.1 does not address the global coordination at all.

Regarding the first issue, there are in fact many different optimization algorithms varying dramatically in their characteristics.

On one extreme, *random search*⁵⁶ is a very simple and robust algorithm which has no problem with local minima as it searches the set of possible EC parameters globally and is therefore not attracted to local minima but is incredibly slow to find a good solution.

In contrast, *gradient descent* and its variations are moderately fast and stable, but they are attracted to *local minima*, especially in small parameter spaces such as the EC parameters. In other words, for a given impedance spectrum, the space of possible EC parameters is divided into many sub-regions which are the basins of attraction of various local minima, and the optimizer will converge to the local minimum within the sub-region where the initial guesses are chosen. Note that depending on the precise details, the convergence may still be slow, but much faster than random search. Furthermore, with appropriately small steps, convergence is highly likely, if left to optimize long enough.

On the other extreme, various second order methods can converge very fast, but will also diverge more readily, and still will have some trouble with local minima, though their interaction with the basins of attraction is less clear for the same reasons that these algorithms are more prone to divergence: they take larger steps, and the justification for the steps taken helping the optimization have stronger assumptions built-in.

There are ways of trading-off the strengths and weaknesses of the various approaches (for instance by doing a random search on initial guesses each followed by a short gradient descent), and there are many approaches not mentioned here, but the more time spent selecting the perfect algorithm here, the more this “individual fitting” approach becomes similar to the unsupervised approach, since the optimizer will be selected based on a global criterion over plausible impedance spectra. If the search for a good inverse model is performed over a big enough and diverse enough set of possible optimizers, then it is reasonable to expect similar or greater performance to that discussed in Section 3.4 could be achieved. In other words, the unsupervised approach is not in principle limited to considering only neural networks as possible inverse models; more general objects including optimizers could also be considered. Note however, that global coordination may still be a problem and it is beyond the scope of the thesis to optimize over optimizers.

3.3.2 Hybrid Approach

There is not much to say about the human-machine hybrid approach.

3.3.3 Supervised Approach

However, the supervised approach is quite interesting:

- This is the approach that most people with a passing familiarity with machine learning would think of first. It indeed was the case for the author.
- It works much less well than the unsupervised approach, not quite matching the human ability to visually estimate the key EC parameters.
- It works sufficiently well to build a good solution to the fitting problem, when used to constrain the initial guesses. However, it required many human-written heuristics, a well-chosen proprietary optimizer from Mathematica for the individual fitting portion, and many

weeks of tweaking and patching the code. However, the global coordination problem was not properly addressed by this code and the complexity of maintenance was much too high.

3.3.4 Unsupervised Approach

In retrospect, it is obvious that the unsupervised approach is superior. But the real question is what type of reasoning would have led one to conclude *beforehand* that the unsupervised approach would work much better?

The elegance and simplicity argument for using a problem setting which directly corresponds to the desired outcome would have advocated for the unsupervised approach. However, this argument would not have predicted that the supervised approach would fail to meet the human level of performance, or that the unsupervised method would exceed it.

Similarly, this argument would not have predicted that the unsupervised method would naturally (i.e. without special effort) display good global coordination, or that the supervised approach would *not* display it even *with* some significant special effort (i.e. the clustering approach^{57,58}). To emphasize the point, it is unclear to the author whether the supervised approach can in practice display a high level of global coordination even with a lot *more* special effort. If such a thing is possible, the author would bet that the special effort contains something akin to the unsupervised approach and that the supervised part plays at best no role and at worse a detrimental role.

While the elegance argument fails to make bold enough predictions, some intuitive arguments predict boldly **in the wrong direction**. For instance, focussing purely on the apparent difficulty of selecting a neural network would misguide greatly. Indeed, it seems that the supervised dataset simplifies the optimization problem over neural networks by providing the “correct” EC parameters for the given spectra. While both the supervised and unsupervised problem must learn

to correlate the spectrum shape to the correct EC parameters, the unsupervised problem must first determine what the correct EC parameters are. In other words, the unsupervised task must solve the individual fitting problem whereas the supervised task is handed a perfect solution to imitate. Since the task of solving the individual fitting problem is difficult (otherwise this Chapter would not exist), the argument goes that the unsupervised task would do poorly. This of course is a wrong prediction.

Now that some insufficient arguments and wrong arguments have been presented, let us attempt to give good arguments/considerations.

First, since the supervised approach was only tried on parameters which could successfully be estimated visually, the possible solutions for the supervised approach are strictly *less* powerful, but assuming the supervised approach is extended to predict all parameters, the possible candidate inverse models are the same in both approaches. Figure 3.1 illustrates that the two approaches place the neural network in the same place.

Second, assume access to a perfect optimizer for neural networks which takes a loss function defined by the dataset and the selection criterion, and produces a neural network which has the lowest possible loss on the dataset while respecting some continuity condition. Furthermore, assume the largest possible dataset was collected according to the methodology of either the unsupervised or supervised approach. In such a scenario, it is still possible to predict some aspects of the inverse model produced using either the supervised or unsupervised approach:

- In the case of the unsupervised method, the inverse model would perfectly solve the individual fitting problem on every single spectrum in the dataset (i.e. it would associate

to every spectra the choice of EC parameters which minimizes the mean squared reconstruction error).

- However, in the case of the supervised method, assuming there are multiple choices of EC parameters which produce identical spectra (also nearly identical spectra in case the continuity condition would force the neural network to produce nearly identical EC parameters for such spectra), the best which can be achieved is an inverse model which predicts the *average* of all the multiple choices of EC parameters.

When presented with identical spectra from the dataset, the neural network must produce a unique EC parameter vector. If these identical spectra are associated with distinct EC parameter vectors in the dataset, the loss function cannot be perfectly satisfied. The best that can be done is to produce the average of all the EC parameters associated with the same spectrum.

In a slightly more realistic dataset creation effort, there would be imbalances in the multiple choices of EC parameters for a given spectrum. In general, the imbalances would be inconsistent from one spectrum type to the next, which would hurt the global coordination of the solution.

In short, the unsupervised approach has no fundamental limits on its achievable performance, but the supervised approach is limited by the quality of its supervised EC parameters, even in the limit.

Third, the unsupervised approach allows the use of datasets which are strictly larger than those available to the supervised approach. Indeed, any spectrum for which the EC parameters are known can be included in either dataset, but the unsupervised dataset can also include spectra from experiments, as well as spectra sampled from more complicated and diverse EC models than the one presented in Section 2.3.

Furthermore, any transformation of the impedance spectrum which has an unpredictable effect on the corresponding optimal EC parameters may still be included in the unsupervised dataset.

In short, the unsupervised dataset can be much larger and diverse than the supervised dataset. Indeed, there are no limitations on the dataset size for the unsupervised approach and in practice a small amount of effort yielded significant improvements in the quality of the dataset in the unsupervised case.

Based on these considerations, the main imagined hurdle for the unsupervised approach is the fact that it must solve the individual fitting problem which is difficult to do. However, this is not quite accurate. The optimizer applied to the parameters of the neural network will follow gradients in a space with a completely different geometry; one which is much less plagued by bad local minima^{59,60}. Indeed, it is a general result for many domains of application of neural networks that optimization in these very large spaces of neural network parameters is feasible, and various tricks exist to successfully solve these seemingly challenging optimization problems for up to hundreds of billions of parameters³¹. In short, solving the complete fitting problem of finding a good inverse model is not harder than solving every individual fitting problems on a given dataset. Indeed, applying relatively straightforward tricks is expected to solve any issues encountered most of the time on these problems.

Finally, could the good global coordination of the unsupervised approach have been predicted by a good argument? In other words, assuming that the unsupervised approach can reach a given inverse model as a good solution to the fitting problem, and comparing all such inverse models with each other, *is it going to be easier to reach solutions with good global coordination or solutions with bad global coordination?* This is in the end an empirical question, but some intuitions can be used to attempt to estimate these questions beforehand.

If nearby spectra A and B produce incompatible EC parameter choices $\theta_{EC,A}$ and $\theta_{EC,B}$, then essentially the neural network could easily be used to distinguish between A and B more or less reliably. Then, the ability to distinguish A from B must come about and persist in conjunction with the ability to send A to a good choice $\theta_{EC,A}$ and B to a significantly different good choice $\theta_{EC,B}$. It seems unlikely to come about compared to a system which doesn't have the ability to distinguish A from B, and just predicts the same choice for both $\theta_{EC,C}$, and which then incrementally specialises the predictions to lower the mean squared error of reconstruction without a big change in the predictions themselves. Indeed, what would be the incentives for developing the ability to distinguish A from B before the ability to send A to $\theta_{EC,A}$ and B to $\theta_{EC,B}$? A similar analysis would predict that these two separate abilities are difficult to maintain through the training process. This style of thought is still not completely formalized, but it seems useful to develop good expectations about outcomes of sophisticated neural network training processes⁶¹.

3.4 Results

If the only requirement was to produce adequate fits for a **fixed** set of impedance spectra, then the inverse model could always be optimized precisely on all the spectra of interest. Initial experiments showed that the basic setup can accomplish this relatively easily. However, optimizing the inverse model on a large set of spectra can take some time, and if new spectra are constantly measured, this approach would introduce latency and management complexity. It is also quite inefficient to constantly train on a growing set of spectra every time a new spectrum is measured.

Therefore, the proper context to evaluate the performance of the proposed fitting system is as follows: Once the inverse model has been optimized on a set of spectra, it is applied to a set of **new** unseen spectra. Then the following are measured:

- The accuracy of the fits (in the sense of reconstruction error) on the new spectra, as produced by the inverse model.
- The accuracy of the fits on the new spectra when the inverse model is used as initial guesses for a gradient descent optimizer.

If the inverse model produces relatively good EC parameters for a given spectrum, then the simple optimization pass should converge, since only small adjustments of the EC parameters would be required. After doing this, the actual quality of the fits can be determined.

The numerical experiment proceeds as follows.

As detailed in Section 2.3, around 100000 experimental impedance spectra were collected over many years, and are organized into two datasets with different characteristics (the FRA dataset and the EIS dataset). Section 2.3 also details the main differences in those datasets (mostly the temperature of measurement, the types of cells measured, and the ratio of spectra to different cells within the dataset). Each dataset is split at random into two disjoint sets (called *train* set and *test* set respectively), with 1% of the spectra for the train set and 99% for the test set. Since the FRA dataset contains potentially many spectra from the same cell, we enforced the constraint that there be no cell for which spectra exist both in the train FRA dataset and the test FRA dataset.

Then, the inverse model is optimized on a mixture of data from three separate sources: 1) generated data (20 percent of the spectra seen, including repetition), 2) data from the FRA train set (40 percent of the spectra seen, including repetition), and 3) data from the EIS train set (40 percent of the spectra seen, including repetition). Finally, the inverse model is fixed, and is applied to both the FRA test set and the EIS test set to produce EC parameters for each spectrum in the test set.

These EC parameters are used to produce a reconstruction, and the difference between the original spectra and their reconstructions is calculated. Smaller differences mean a better inverse model. However, to visualize these results, we compute a metric (the mean error divided by the standard deviation of the data). Then, the test set is sorted by this metric, and some fits at various points on this sorted list are plotted. Note that, in order to save space, the fits are shown using Nyquist plots (negative imaginary part vs. real part), instead of showing Bode⁶² plots (real part vs. frequency, and imaginary part vs. frequency). From a Nyquist plot, there is no way of knowing whether the frequencies are aligned between fit and original data, as a uniform shift of log-frequency would not change a Nyquist plot. Yet, the optimization has no way of producing good looking Nyquist plots without proper frequency alignment. In all cases shown, the Bode plots would all look better than the corresponding Nyquist plots.

It is not practical to show all the fits in the test set, but Figure 3.4 through Figure 3.7 show fits that have not been “cherry-picked” and illustrate qualitatively the results of Figure 3.2 and Figure 3.3. Similarly, the complexity measure introduced in Section 2.7 is computed once the EC parameters are known, and it is plotted across the dataset. Ideally, the fits should have both low error and low complexity, but experimental data can never have 0 error, and some spectra clearly exhibit more than one active electrochemical ZARC, hence the complexity measure must be superior to 1.

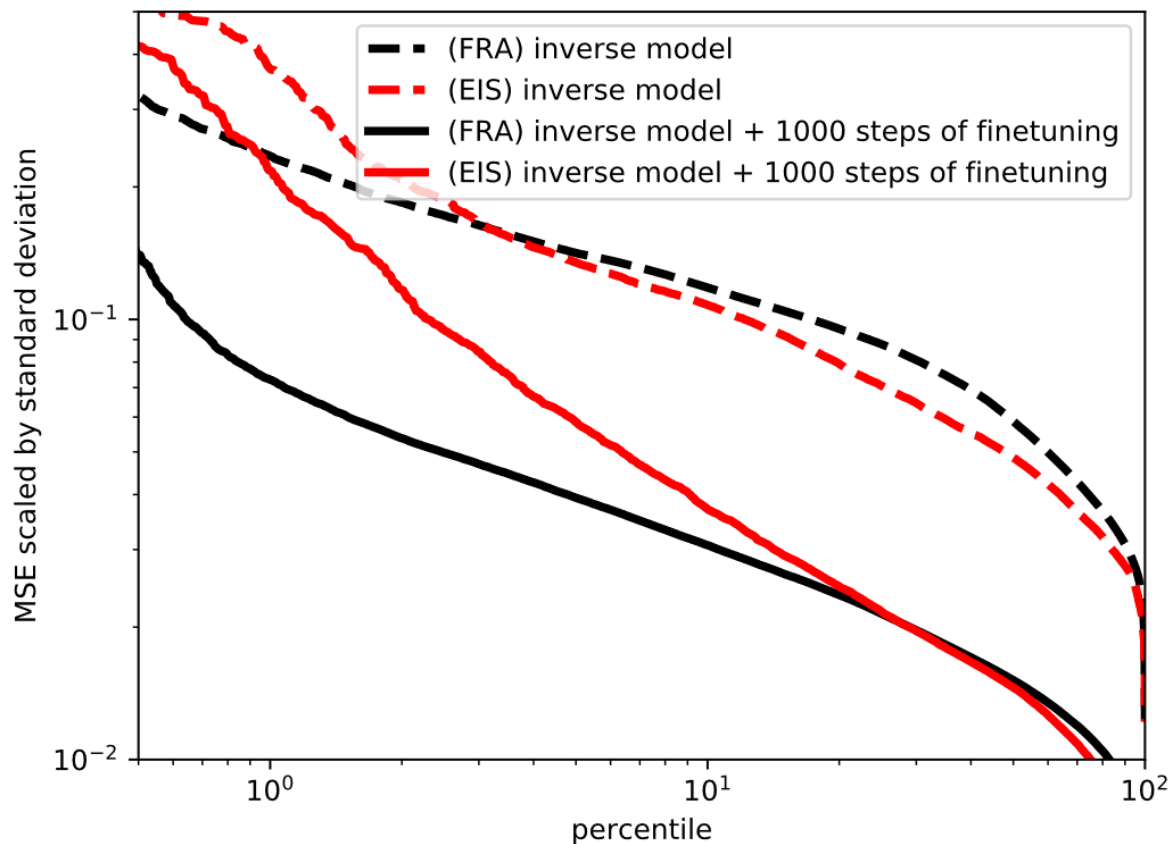


Figure 3.2 The error for both the FRA and EIS test datasets, for the EC parameters directly produced by the inverse model, and those produced by applying 1000 steps of ADAM finetuning to the output of the inverse model. The mean squared error, scaled by the standard deviation, was computed by comparing the reconstructed spectra and the original. These errors were sorted and plotted against the percentile. Therefore, a percentile of 1 represents a fit worse than 99% of all the fits. The horizontal axis represents the whole set of spectra. The inverse model trained for a day. One thousand steps of ADAM finetuning is roughly 0.05 seconds per spectrum. Note that the times were obtained on a 2016 laptop, with an NVIDIA Quadro M1000M graphical processing unit.

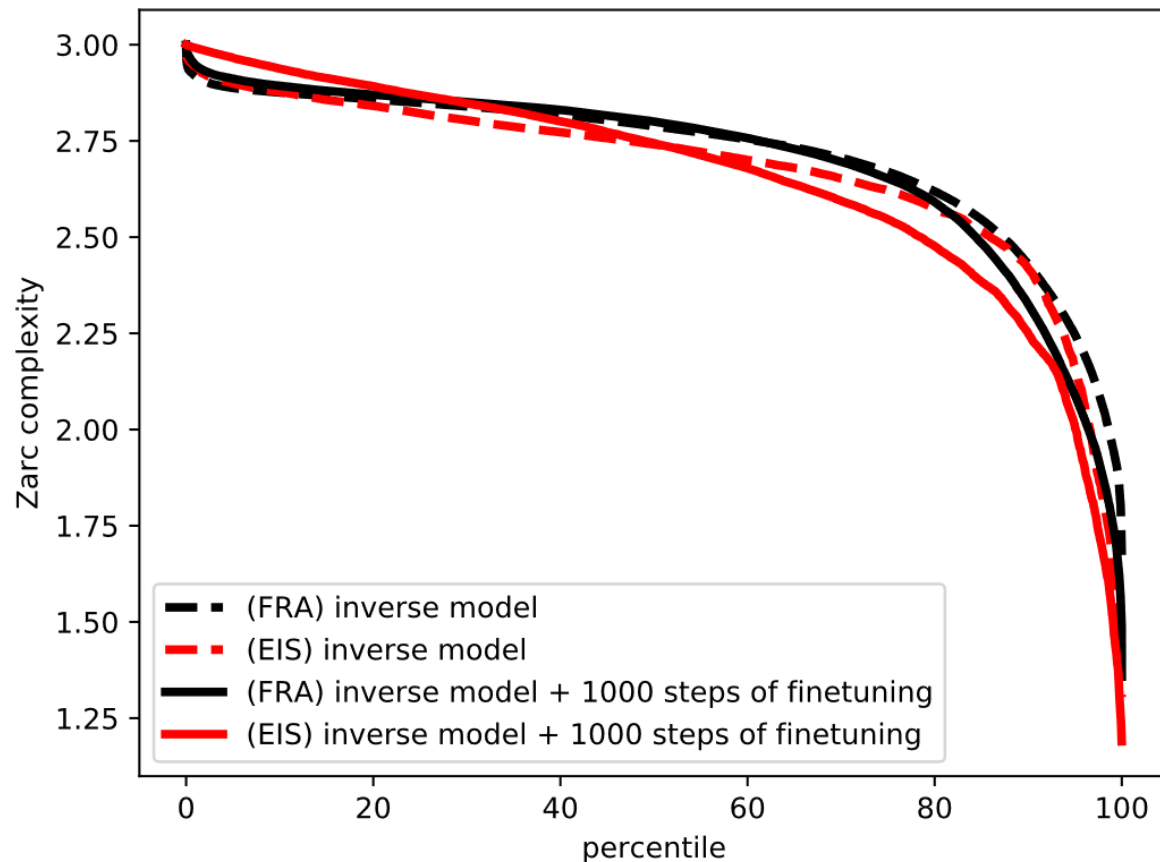


Figure 3.3 The complexity metric is shown for both the FRA and EIS test datasets, for the EC parameters directly produced by the inverse model, and those produced by applying 1000 steps of ADAM finetuning to the output of the inverse model. Figure 3.2 gives details of how the fits were obtained. The complexity decreases slightly during ADAM finetuning.

Figure 3.2 and Figure 3.3 show a quantitative evaluation of the performance of the system (respectively the error and the complexity metrics). The strength of the inverse model optimization is to be able to avoid bad local minima and explore the landscape of possible fits to find an overall good solution. The strength of the finetuning is its stability and precision (it will simply converge to the nearest local minimum).

Initially, the finetuning was a simple gradient descent algorithm (i.e. the derivative of the MSE error is taken with respect to the EC parameters, and the EC parameters are updated by shifting a small amount in the direction that most reduces the error, namely the negative of the gradient). However, the performance was not good. Hence, we implemented the ADAM optimization algorithm²², a variant of gradient descent which in our case performs much better (see Section 1.1.6). Roughly speaking, by computing the gradient at each step, the average and the standard deviation can be estimated, and a step can be taken in the direction of negative average gradient but the size of the step will be smaller if the standard deviation is large.

Figure 3.4 through Figure 3.7 evaluate the fits qualitatively.

For instance, Figure 3.4 shows some fits of the FRA test dataset using the inverse model, at various percentiles of error. The actual data is shown as the dots whereas the spectra reconstructed from the fitted EC parameters are shown with a line of the same color. The big stars are the datapoints which have an angular frequency closest to the characteristic frequency of the corresponding ZARC. There are 3 ZARCs, so three stars should be visible, with the ZARC having the lowest characteristic frequency usually showing up to the right in a Nyquist plot. Furthermore, to help understand the complexity metric, the resistance corresponding to each ZARC was calculated, and is written in the legend. For instance, the blue spectrum in the *percentile 50* plot had its first ZARC at a relatively low frequency, with a resistance of 20, its second ZARC at a medium frequency with a resistance of 21, and its third ZARC at a relatively high frequency with a resistance of 8. Note that the resistances computed would have units of ohm (since they are the bona fide resistances in the original circuit, before reparameterization), but the whole spectrum was scaled such that the maximal observed impedance had magnitude 100. This was done only to simplify

the visualization. Also note that the resistances shown were rounded to the nearest integer to save space. The same goes for Figure 3.4 through Figure 3.7.

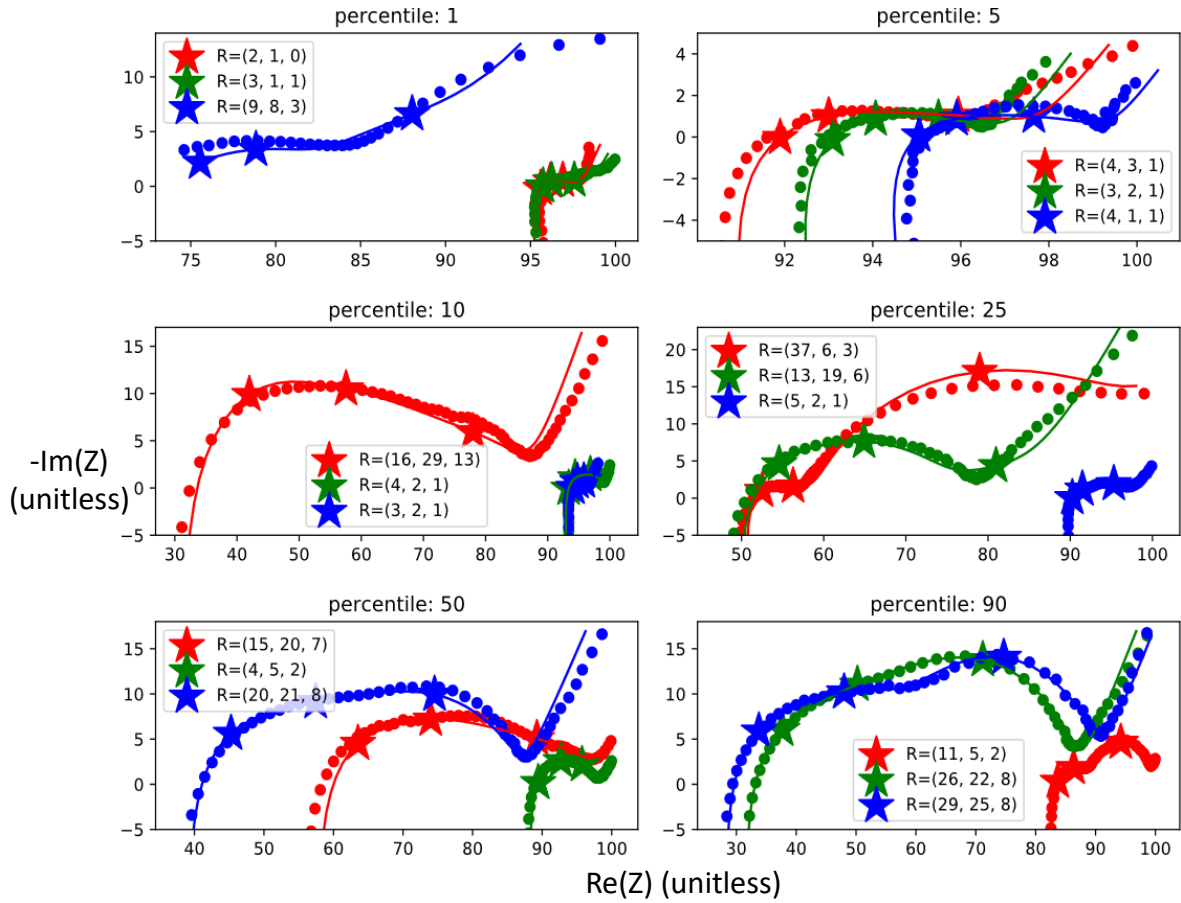


Figure 3.4 Some fits of the FRA test dataset using the inverse model, at various percentile of error. The fits shown are of acceptable quality, but the data itself is of poor quality for the lowest 2 percentiles. The dots and lines represent actual data and reconstructed spectra respectively. The actual data was rescaled such that the largest impedance within each spectrum would equal 100. There are 3 ZARCs, so three stars (each positioned at the dot nearest to the corresponding characteristic frequency) should be visible, with the rightmost star typically representing the lowest characteristic frequency ZARC. The legend shows the resistance corresponding to each ZARC. For instance, the blue spectrum in the *percentile 50* plot had resistances of 20, 21, and 8 for the low, medium, and high frequency ZARCs respectively (when rounded to the nearest integer).

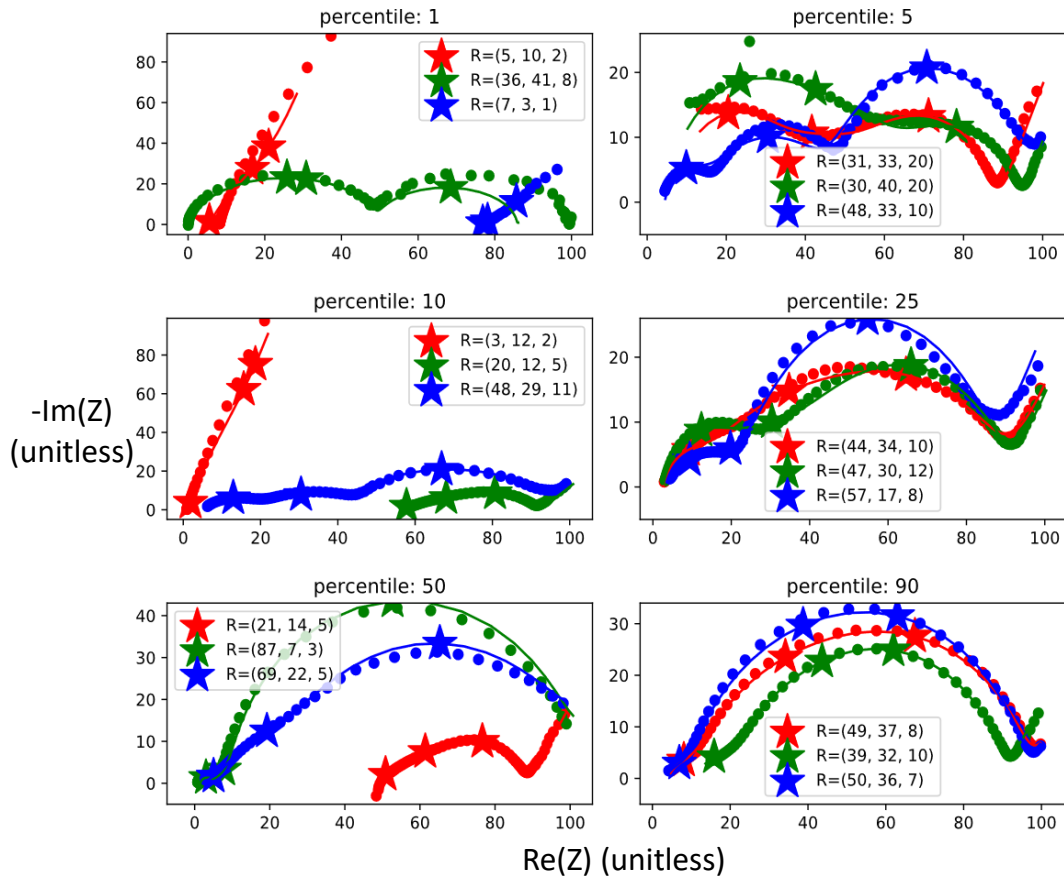


Figure 3.5 Some fits of the EIS test dataset using the inverse model, at various percentile of error.

The fits shown are of acceptable quality, but the data itself is of poor quality for the lowest 4 percentiles. See Figure 3.4 for details about the legend.

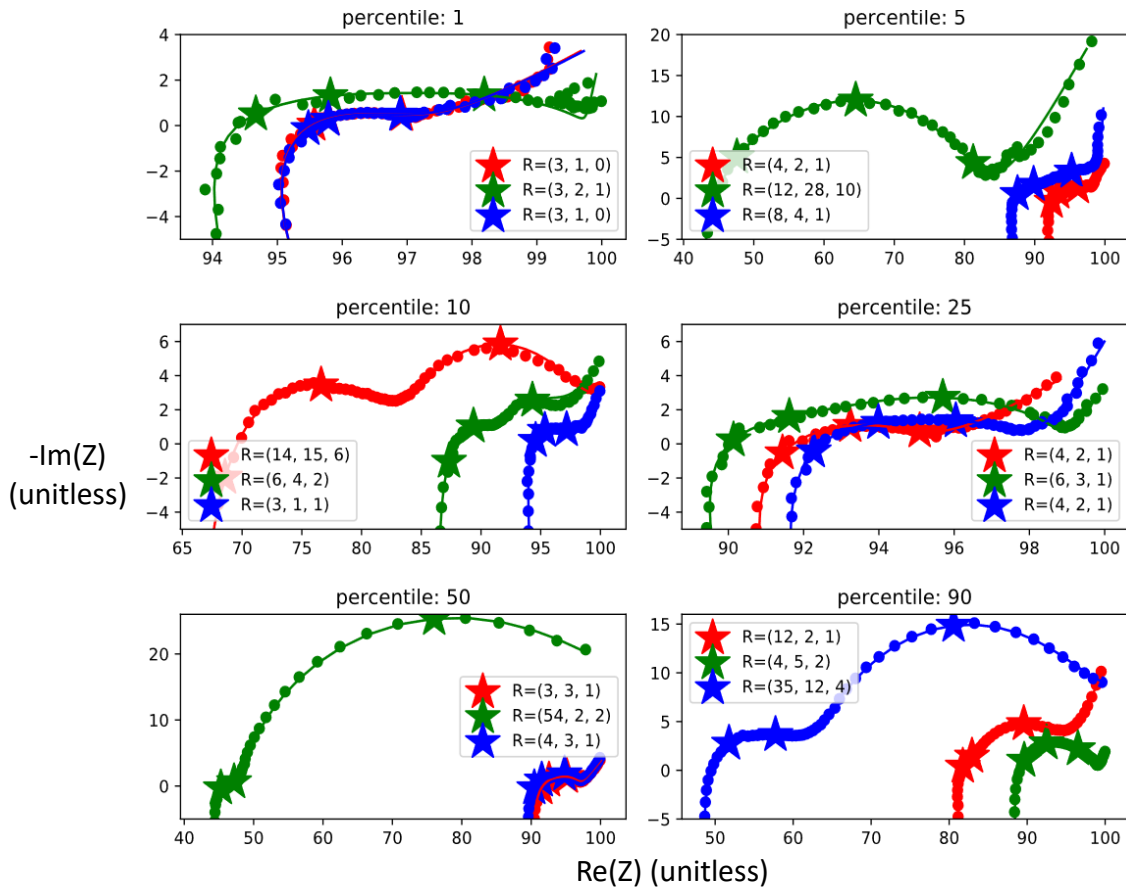


Figure 3.6 Some fits of the FRA test dataset using the inverse model followed by 1000 steps of ADAM finetuning, at various percentile of error. The fits shown are of good quality, but the data itself is of poor quality for the lowest 2 percentiles. See Figure 3.4 for details about the legend.

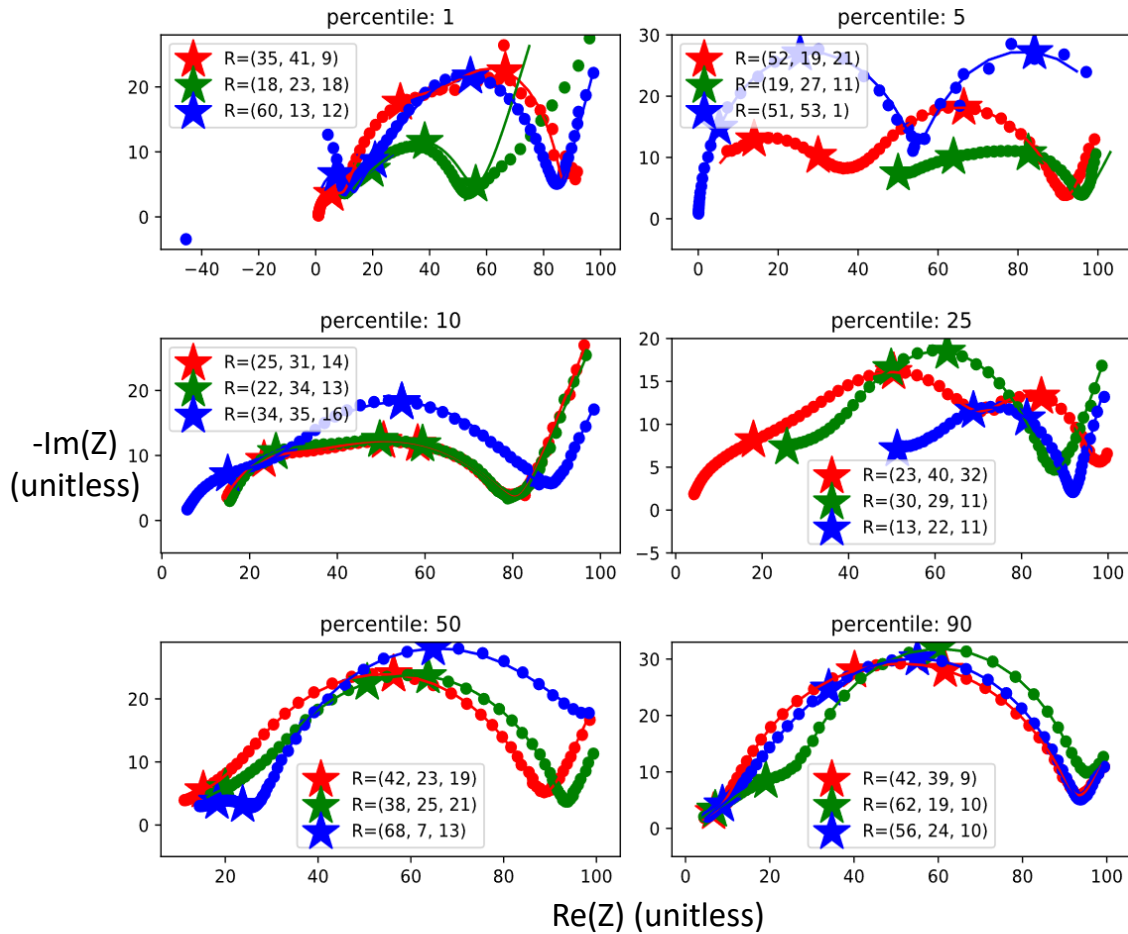


Figure 3.7 Some fits of the EIS test dataset using the inverse model followed by 1000 steps of ADAM finetuning, at various percentile of error. The fits shown are of good quality, but the data itself is of poor quality for the lowest 4 percentiles. See Figure 3.4 for details about the legend.

Looking at Figure 3.4 through Figure 3.7, and confirming by plotting more spectra, the following qualitative conclusions can be reached:

- The inverse model properly fits 98% of the spectra, give or take 2%, but the precision is not optimal.

- The inverse model combined with the finetuning optimization properly fits 99% of the spectra, give or take 1%, and the precision is close to optimal.
- These conclusions hold both on the FRA dataset and the EIS dataset.

Though it is not possible to compare to every other fitting software, the author's experience with freely available fitting software applied to similar data suggests that the inverse model combined with the finetuning optimization is much more reliable than alternatives, and that fitting such a large and diverse dataset with freely available fitting software would be very labor intensive.

By looking at Figure 3.4 and Figure 3.5, anyone who has tried to guess the value of EC parameters visually from the spectra for a circuit such as displayed in Figure 2.5 will see that the inverse model is effective at estimating the optimal EC parameters. Furthermore, given better data and perhaps a larger neural network parametrizing the inverse model, there is no reason why the performance could not be even better. Also, by looking at the low-percentile spectra (i.e. those not well fitted by the inverse model), it is possible to augment the generated data and tweak things until the inverse model properly handles this type of spectrum. However, at any given point, the available data will be incomplete, and there will be corner cases not represented. Therefore, when assessing the performance of a system and quantifying the success rate, it is better to not tweak or generally not try to improve the inverse model on specific cases. Since we have followed this principle, the 98% number given above should be roughly accurate.

These results are encouraging both for the application of EC model fitting, but also for the general approach of training an inverse model, represented as a neural network, and then using a finetuning optimization. We hope that this can be applied to many different difficult fitting problems.

Finally, Figure 3.8 shows a sample of fits in their original scale (before the scaling and shifting), using the combined method (inverse model + finetuning), together with the error metric and the complexity metric for each fit. Though the fitting itself happens in a space where the scales of different spectra are very similar (see Figure 3.4 through Figure 3.7), this method can effectively fit a variety of different spectra at different scales.

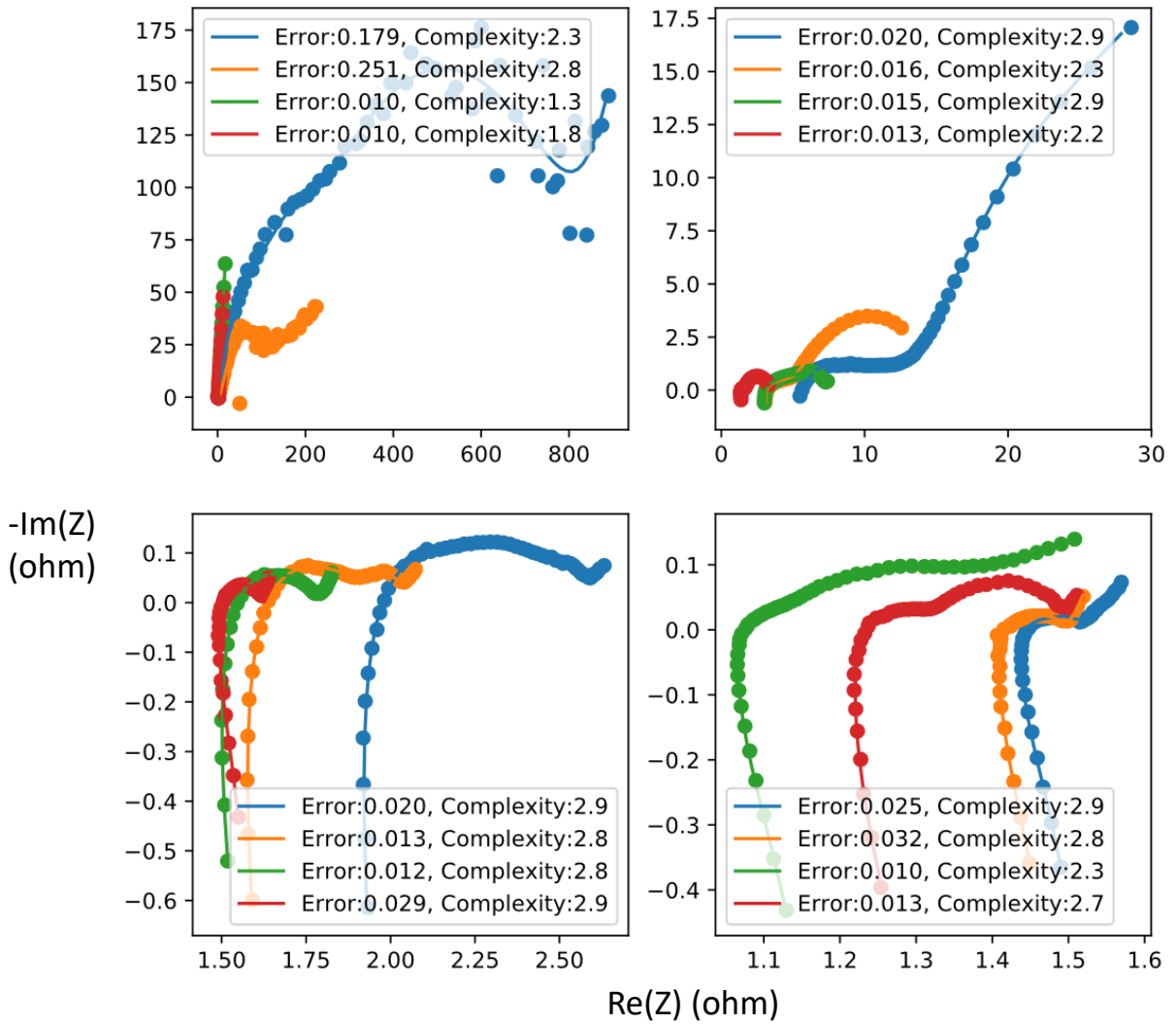


Figure 3.8 Some fits in the FRA dataset shown in original scale. The fits shown were chosen to demonstrate the variety of spectra which can be handled by the system. From thousands of ohms, down to hundreds of milliohms in scale, some fits of various shapes, together with their error metric and complexity metric, are shown. These fits are from the combined method, using the inverse model as well as the fine tuning with ADAM.

3.4.1 Reproducibility and Access to Code

A barebones version of the code is available at <https://github.com/Samuel-Buteau/EISFitting> with all the documentation contained in the README.md file. This codebase contains a pretrained model and the ability to run the model on a directory containing EIS measurements to receive the results. The inverse model is also able to produce results for tens of different EC models. This can be done through a very simple command line interface. We leave the integration of the core software with a truly outstanding user interface as future work.

Many labs interested in applying this technique might be worried that they do not have 100000 spectra available to make this work. However, similar techniques can be applied on a smaller scale to yield good results.

First, note that the two datasets (FRA and EIS) are quite different and producing an inverse model which works well across both tasks is harder than solving each task individually. For the case of the EIS dataset, it contains around 10000 spectra, and we have shown that 1% of the dataset (100 spectra) was sufficient to reach quite good performance.

We note that with more data, the inverse model becomes more precise (for instance, Figure 3.9 shows the performance of a model trained with 10 percent of the data instead of 1 percent, compared with Figure 3.2), but in order to get reasonable starting points for the finetuning, precision is not required for the vast majority of spectra.

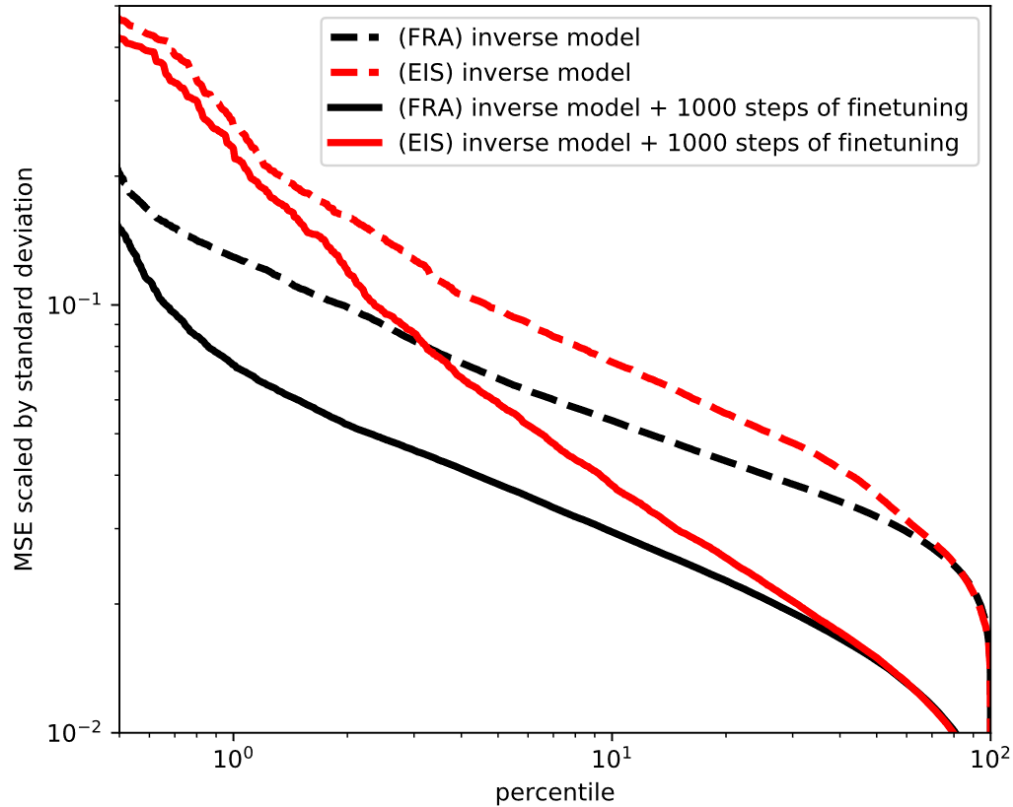


Figure 3.9 The error of a model trained with 10 percent of the data is shown for both the FRA and EIS test datasets, for the EC parameters directly produced by the inverse model, and those produced by applying 1000 steps of ADAM finetuning to the output of the inverse model. The mean squared error, scaled by the standard deviation, was computed by comparing the reconstructed spectra and the original. These errors were sorted and plotted against the percentile. Therefore, a percentile of 1 represents a fit worse than 99% of all the fits. The horizontal axis represents the whole set of spectra. The inverse model trained for a week, but the performance remained stable from day 3 onward.

3.5 Conceptual Discussion of Tricks

Now that the general considerations are spelled out, we detail the various tricks and specific solutions which improve the performance of the system on this particular problem setting of fitting EIS spectra.

The optimization of the inverse model is still a considerable problem, and some tricks go a long way to making it converge quicker, more stably, and favor some desired properties of the solution. These tricks are discussed below. Note that this and the following sections of this chapter are heavy in details, and readers primarily interested in the use of a fitting software need not read them.

3.5.1 Defining a Prior Distribution on the EC Parameters

Even before seeing the spectrum, some values of the EC parameters are less likely than others. Formally, we can represent this knowledge as a probability distribution over θ_{EC} henceforth called the *prior*. Formally speaking, a prior is entirely subjective, and it can be interpreted as a guess of the likelihood of any given θ_{EC} being appropriate for a randomly chosen impedance spectrum. For simplicity, an average value $\theta_{EC,\mu}$ and a standard deviation $\theta_{EC,\sigma}$ is chosen for all the EC parameters (more formally, each parameter follows an independent gaussian distribution), and these values do not depend on individual impedance spectra. Note that the prior only represents one's best guess in a simple form and so it may not have much to do with the actual distribution of θ_{EC} over plausible impedance spectra. When a prior precisely corresponds to the actual distribution of θ_{EC} over plausible impedance spectra, it is called the *optimal* prior. When it is necessary to distinguish an actual choice of prior from the optimal prior, the actual choice is called a *subjective* prior. For the form of prior chosen, increasing the standard deviations is said to *broaden* the prior and decreasing the standard deviations is said to *compress* the prior.

In some sense, choosing a subjective prior⁶³ is related to the supervised approach, but these are different concepts. The supervised approach defines implicitly a prior if one simply ignores the correspondence between specific spectra and specific EC parameters and only looks at the set of EC parameters (to get a gaussian prior out of this, simply compute the average and standard deviation of each parameter). In the case of the unsupervised approach, this correlation between specific spectra and specific EC parameters is not explicitly given, and instead is chosen to serve the overall criterion of producing good reconstructions. On the other hand, the prior can be used as a form of weak supervision such that EC parameters unlikely according to the prior may be slightly penalized as shall be discussed later.

In general, there is nothing preventing one from using criteria more like the supervised approach (i.e. directly defined on EC parameters) in conjunction with criteria more like the unsupervised approach (i.e. directly defined on the reconstructed spectra). *Supervised* and *unsupervised* were convenient words for separating the two approaches, but the generalizable insight from Chapter 3 should not be that “unsupervised is better than supervised” (which is false for many problems). Instead, the insight is that considering many different mechanisms to produce a model may yield better results than simply using the most obvious one.

3.5.2 Leveraging Symmetries to Compress the Prior

Impedance spectra gotten from experiments are extremely diverse in their scale (coin cells typically have much smaller surface area, and thus correspondingly higher impedance than pouch cells), and the frequency ranges over which the characteristic frequencies reside. Therefore, coming up with fixed reasonable values for θ_{EC} to suit all these spectra would be challenging, forcing relatively poor match between the prior and any given spectrum’s actual parameters (i.e. the optimal prior would be very broad).

However, the symmetries discussed in Section 2.5 can be leveraged to scale each impedance spectrum such that the average absolute value of the impedance with respect to frequency equals 1 and such that the logarithm of the frequency averaged by the absolute value of the impedance equals 0. This would compress the optimal prior greatly, since impedance spectra with disproportionately large resistances would have their resistances diminished, and vice-versa.

Correspondingly, it is relatively simple to choose reasonable ranges for each EC parameters knowing that the spectra have been manipulated as above. Indeed, the code has been run with two versions of the prior: once simply by guessing reasonable values (a subjective prior), and once by looking at the final parameters of all the fits, and choosing the average value and standard deviation for each (an approximation to the optimal prior). We note that none of the results change, despite the huge gap between the subjective prior and the (approximated) optimal prior.

Indeed, these symmetries compressed the optimal prior enough to make it easy to choose a relatively good prior. Starting with a good prior when searching for an inverse model is akin to starting with a good initial guess when solving the fitting problem for an individual spectrum, but the process of searching for an inverse model is a lot more resilient, so the only difference between a good and a bad prior in practice is the time it takes the neural network to converge to a good inverse model.

3.5.3 Modelling Deviations from the Averaged Prior

Let $\theta_{EC,\mu}$ be the average values chosen for the EC parameters as defined in the prior distribution above. Then, instead of representing the inverse model by a neural network directly, we represent the **difference between outputs of the inverse model and the average values $\theta_{EC,\mu}$** by a neural network, and it is initialized such that the output is 0. This has the effect of beginning with an

inverse model which is constant and which always produces $\theta_{EC,\mu}$ at the beginning, which are chosen to be reasonable. Then, as the neural network is trained, its outputs will deviate from $\theta_{EC,\mu}$ more and more until it reaches different EC parameter values appropriate for individual spectra. This trick is similar to residual networks⁶⁴ (i.e. make the default prediction better).

3.5.4 Penalizing Deviations from the Prior

As we minimize the mean squared error of reconstruction across the set of spectra with respect to the neural network parameters θ_{Inv} , a penalty which shall be denoted by $\mathcal{L}(\theta_{Inv}; S)$ (see List of Symbols), some EC parameters might fall into a range of values where they have very little impact on the reconstructed spectrum (see Section 2.6).

Furthermore, following the gradient of $\mathcal{L}(\theta_{Inv}; S)$ can lead to exploring inverse models that produce EC parameters where, for instance, the error becomes so large that the numerical precision is insufficient to represent it. By adding a small term in the minimization that pushes the predictions towards typical values, the optimization is more stable. In general, given a probability distribution P and a prediction θ_{EC} , we can compute the *likelihood* of the prediction according to that probability distribution (in our case it is the probability density of sampling the prediction θ_{EC} from the probability distribution P). In practice, we minimize the negative logarithm of the likelihood. For our choice of prior, this is easily computed and essentially is the squared error between $\theta_{EC,\mu}$ and the predictions, weighted by the inverse of the standard deviation $\theta_{EC,\sigma}$ such that parameters which are very broad in the prior will suffer a smaller penalty for predictions which are far from $\theta_{EC,\mu}$, but parameters which are compressed in the prior⁶⁵ will suffer a larger penalty.

3.5.5 Breaking the Symmetry in the EC Model

Since there are 3 ZARC elements all modelling similar processes, the reconstructed spectrum does not change when we interchange the parameters of two ZARCs (see Section 2.6). Therefore, there is nothing in the mean squared error of reconstruction across the set of spectra $\mathcal{L}(\theta_{\text{Inv}}; S)$ to break this symmetry⁶⁶. Indeed, there exists multiple distinct inverse models which minimize equally well $\mathcal{L}(\theta_{\text{Inv}}; S)$ and in fact display the exact same degree of global coordination. Namely, all prediction of a given inverse model can be permuted as above to obtain an equally valid inverse model.

To simplify the interpretation of the solution, and to simplify the prediction task, we penalize predictions of θ_{EC} where the electrochemical ZARCs are not ordered by characteristic frequencies (from lower to higher). Letting w_{c1}, w_{c2}, w_{c3} be the characteristic log-frequencies of the 3 electrochemical ZARCs, this is done by minimizing a penalty which is positive when $w_{c1} > w_{c2}$ or when $w_{c2} > w_{c3}$, and 0 otherwise. To make this differentiable with useful gradients, we compute $\text{relu}(w_{c1} - w_{c2}) = \max(0, w_{c1} - w_{c2})$. When $w_{c1} > w_{c2}$, this function gives a penalty equal to the difference $w_{c1} - w_{c2}$, but when $w_{c1} \leq w_{c2}$, this function is 0. By taking the gradient, we see that this penalty “pushes” w_{c1} toward w_{c2} with a constant “force” when $w_{c1} > w_{c2}$, but has no impact when $w_{c1} \leq w_{c2}$. Similarly, we compute $\text{relu}(w_{c2} - w_{c3})$ for the same reason. Note that despite not having a term to ensure that $w_{c1} < w_{c3}$, this will be optimized for because if $w_{c1} < w_{c2}$ and $w_{c2} < w_{c3}$, then it follows that $w_{c1} < w_{c3}$.

Finally, previous numerical experiments have revealed that it is possible to set w_{c3} to a very large value, in which case the third ZARC will simply behave as a resistor. Similarly, it is possible to set w_{c1} to a very large negative value, in which case the first ZARC has no impact on the impedance spectrum. Both of these are undesirable for two reasons. First, this is against the prior and therefore quite unlikely to represent some real process in the lithium-ion cell. Second, there is

nothing in the measured spectrum which can indicate the presence of ZARC elements with characteristic frequencies far outside the range of measured frequencies. As a safety precaution, we also add the penalties $\text{relu}(w_{min} - w_{c1})$ and $\text{relu}(w_{c3} - w_{max})$, where w_{min} is the logarithm of the smallest observed frequency for a given spectrum, and w_{max} is similarly the logarithm of the largest observed frequency.

3.5.6 Penalizing Complexity

Section 2.7 defined a quantitative measure to track complexity of a given solution. To select an inverse model which produces better solutions, we also penalize the “ l_1 norm” $R_1 + R_2 + R_3$ directly as well as the “ $l_{1/2}$ pseudo-norm” $(\sqrt{R_1} + \sqrt{R_2} + \sqrt{R_3})^2$. In the literature, many ways of penalizing complexity exist⁶⁷, though the penalties often are applied directly to the neural network parameters instead of its output.

3.5.7 Automatically Setting the Relative Importance of the Penalties

Since the most important thing to optimize is the mean squared error of reconstruction across the set of spectra $\mathcal{L}(\theta_{Inv}; S)$ and we don’t really know the target values of all the penalties, instead of minimizing the sum of the penalties, we minimize $\mathcal{L}(\theta_{Inv}; S) + \sum_i P_i(\theta_{Inv}; S)$ where P_i are the penalties. Yet, when taking the gradient of this number with respect to θ_{Inv} , there are some terms without use. Empirically, we observe that the most stable objective to minimize is implemented as $\mathcal{L}(\theta_{Inv}; S) + \text{stopgrad}(\mathcal{L}(\theta_{Inv}; S)) \sum_i P_i(\theta_{Inv}; S)$, where $\text{stopgrad}(\mathcal{L}(\theta_{Inv}; S))$ is treated as a constant with respect to differentiation⁶⁸, but the value of the constant is $\mathcal{L}(\theta_{Inv}; S)$. It is also possible to use statistical properties of the individual terms to set their relative weights, though this was not done here⁶⁹.

3.5.8 Generating Fake Data to Improve Robustness of the Inverse Model

We can use the prior distribution over EC parameters to generate fake spectra, simply by choosing a range of frequency, sampling from the prior distribution of EC parameters and evaluating the EC model on the chosen frequencies and parameters to produce a spectrum. However, to increase robustness, we allow for between 0 and 9 electrochemical ZARCs to be present (the number is chosen at random, with expected number around 3). In practice, spectra generated this way look varied and realistic. However, we have found that by replacing gaussian priors with uniform-over-a-range priors (with the range of possible values centered on the mean of gaussian prior, and the length of the range being proportional to the standard deviation), and playing around with the values, we could increase perceived variation and realism of the fake data⁷⁰. As future work, it would be useful to automatically determine the adjustable parameters of the generating process such that the fake data has a good similarity with the real data, but also has a more broad/comprehensive distribution.

The most straightforward way of doing this is to maintain a table of the N most recent predictions associated with each actual spectra in the dataset. Then, instead of using the prior as a basis for generating fake data, a subset of these previous predictions may be sampled, and some priors (one prior per prediction) with the same standard deviations as the subjective prior for the dataset but the average values taken from the predictions themselves may be created and used in the fake data generation procedure. At the beginning of training, this process would produce the same fake data distribution as the process based on the subjective prior for the whole dataset, but as the inverse model began specializing its predictions to the actual spectra in the dataset, the variation in the generated data would grow.

This future work may allow the system to work even with an experimental dataset with smaller *variety, quality or quantity*.

3.5.9 Augmenting Real Data to Improve Robustness of the Inverse Model

Instead of simply optimizing the inverse model on a fixed number of experimental impedance spectra, we can apply some transformations to these precious experimental spectra to produce a larger set of spectra to optimize on. First, we remove a random number of the higher frequencies from the spectrum (technically, we never remove frequencies lower than the first high frequency where the imaginary impedance becomes non-positive). Then, after the rescaling and frequency shifting, we scale and shift by a small random amount. These transformations^{71,72} extend the set of actual spectra to a continuous manifold of spectra and imposes better robustness constraints on the inverse model. As future work, it may be useful to extend the data manipulations (e.g. superposing real spectra with generated spectra, resampling the frequencies and linearly interpolating).

3.5.10 Error Rescaling

The rescaling of the impedance spectra makes the spectra unitless. Some spectra have a huge spread in their values, while others have a small spread in their values. Without rescaling the error to be comparable, the training signals will be dominated by the samples with large spread in their values. In our case, this is not desirable since most often, this means a fair bit of noise in the spectrum, or a large tail either coming from diffusion or inductance. The rescaling is done by dividing the difference between the reconstructed spectrum and the original by the “empirical standard deviation” of the real part and the imaginary part (respectively). Explicitly, taking all the real parts of impedance for a given spectrum, computing the mean, and then taking the average squared deviation from the mean, we get the empirical variance. The square root of the empirical variance gives the empirical standard deviation.

3.6 Extension to Multiple EC Models

In terms of general usability, an EIS fitting software should allow the user to choose which EC model to use in the fit. The most straightforward way to do this is to train a separate inverse model for each EC model of interest and invoke it on request. However, in practice there are many potential EC models of interest, and the repetition does become prohibitive once more than 10 inverse models must be maintained. In reality, this “problem” is a minor concern for this specific application, but solving it illustrates how the machine learning toolkit may be wielded in creative ways once the basic setup works properly.

The goal is to use a **single neural network** to act as the inverse model for **multiple EC models**. More formally, instead of finding a function from measured spectra $Z_{\text{measured}}(\omega)$ to EC parameters θ_{EC} , we must now construct a function from measured spectra $Z_{\text{measured}}(\omega)$ and EC model identifier ID_{EC} to EC parameters θ_{EC} for that EC model. Since some EC models may have a different number of parameters, we will require a space large enough to contain any EC model’s parameters, and from that space, we will define a projection operation to convert from the larger space to a given EC model’s parameter space.

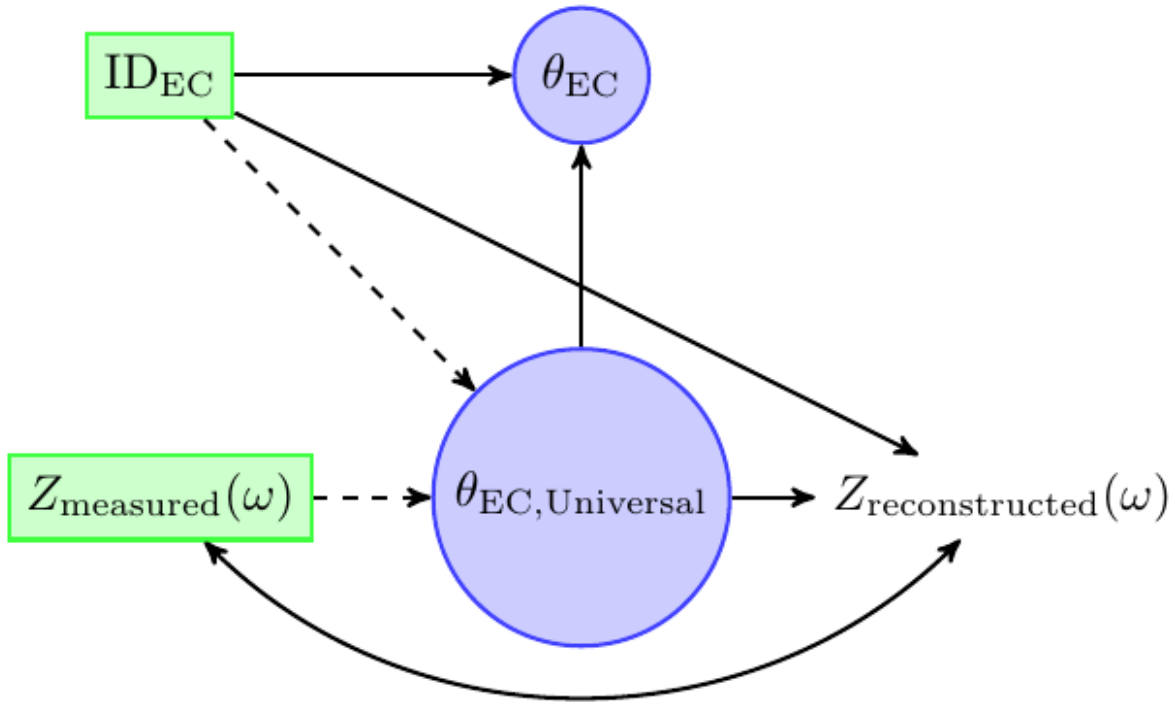


Figure 3.10 A graphical illustration of the multi EC model variant of the fitting problem. The inverse model must now take as inputs $Z_{\text{measured}}(\omega)$ and ID_{EC} (i.e. the representation of the equivalent circuit used) to produce EC parameters in a common space to all the EC models $\theta_{\text{EC,Universal}}$. Given ID_{EC} and $\theta_{\text{EC,Universal}}$, as well as some frequencies, the reconstructed spectrum may be sampled. Finally, given ID_{EC} and $\theta_{\text{EC,Universal}}$, the parameters of the EC model θ_{EC} can be obtained in the space suitable for that given EC model.

A complete list of the various supported circuits can be found in the user manual at <https://github.com/Samuel-Buteau/EISFitting/blob/master/UserManual/manual.pdf>. Figure 3.11 and Figure 3.12 give six examples of circuits which may be of interest.

First, we describe how given ID_{EC} and $\theta_{\text{EC,Universal}}$, as well as some frequencies, the reconstructed spectrum may be sampled. This is in fact straightforward to do with the notion of an EC model

being optionally present within a larger EC model. Recalling Section 2.1 and the concept of two circuits being connected in series, now we introduce a similar definition *for two EC models being serial options*:

Given two EC models denoted as $Z_1(\omega; \theta_{EC,1})$ and $Z_2(\omega; \theta_{EC,2})$, as well as a *switch* variable s (either 0 or 1), we can define a third EC model called “serial option of model 1 and model 2” as

$$Z_{1\oplus 2}(\omega; s, \theta_{EC,1}, \theta_{EC,2}) = sZ_1(\omega; \theta_{EC,1}) + (1 - s)Z_2(\omega; \theta_{EC,2})$$

In the case where $s = 1$, we say that model 1 is *active* and model 2 is *inactive* (and vice versa).

Note that the impedance of a serial option is constant with respect to the parameters of the inactive model. This means that, in practical terms, they can be set to e.g. 0. When both models can be expressed with the same number N_p of parameters and the parameters for the inactive model are always set to 0, it is possible to view the serial option model as only having N_p parameters instead of $2N_p$ parameters:

$$Z_{1\oplus 2}(\omega; s, \theta_{EC,1}, \theta_{EC,2}) = Z_{1\oplus 2}(\omega; s, \theta_{EC}) = sZ_1(\omega; \theta_{EC}) + (1 - s)Z_2(\omega; \theta_{EC})$$

where $\theta_{EC} = s\theta_{EC,1} + (1 - s)\theta_{EC,2}$. In other words, by adding the parameters of the active model to zeroed out parameters for the inactive model, the combined model always has access to the appropriate parameters. Similarly, if model 1 has a smaller number of parameters N_{p1} , as model 2’s number of parameters N_{p2} (or vice-versa), then the parameters of model 1 can be “padded” with an appropriate number of zeros in order to obtain the same number of parameters as model 2. Therefore, the serial option of model 1 and model 2 can be viewed as having $\max(N_{p1}, N_{p2})$ parameters (excluding the switch variable). Note that this would not be feasible if the switch variable were allowed to take intermediate values between 0 and 1.

Finally, a useful special case is when model 2 is *trivial*, in which case we have:

$$Z_{1\oplus}(\omega; s, \theta_{EC}) = sZ_1(\omega; \theta_{EC})$$

As a shorthand, such models would be called “optional model 1”.

Before moving on, let us address similar concepts which might have occurred to the keen reader.

First, the equation for the serial option of model 1 and model 2 may remind one of the correspondence between the impedance of the negative electrode, the positive electrode, and the complete cell. More specifically,

$$\begin{aligned} Z_{\text{full}}(\omega; s, \theta_{EC,\text{pos-pos}}, \theta_{EC,\text{neg-neg}}) \\ = \frac{1}{2}Z_{\text{pos-pos}}(\omega; \theta_{EC,\text{pos-pos}}) + \left(1 - \frac{1}{2}\right)Z_{\text{neg-neg}}(\omega; \theta_{EC,\text{neg-neg}}) \end{aligned}$$

where the impedance of a cell $Z_{\text{full}}(\omega; s, \theta_{EC,\text{pos-pos}}, \theta_{EC,\text{neg-neg}})$ containing both a positive electrode and a negative electrode can be expressed in terms of the impedance of two different cells each containing two electrodes of the same type.

However, this is a different concept since the values of s in a serial option must be either 0 or 1.

Second, if $Z_{1\oplus 2}(\omega; s, \theta_{EC,1}, \theta_{EC,2})$, $Z_1(\omega; \theta_{EC,1})$, and $Z_2(\omega; \theta_{EC,2})$ are known, then as long as $Z_1(\omega; \theta_{EC,1}) \neq Z_2(\omega; \theta_{EC,2})$, the value of the switching variable s can be determined through fitting. However, if only $Z_{1\oplus 2}(\omega; s, \theta_{EC,1}, \theta_{EC,2})$ is known, and the equations for $Z_1(\omega; \theta_{EC,1})$ and $Z_2(\omega; \theta_{EC,2})$ have a scaling symmetry, then it follows that the switching variable s cannot be determined through fitting alone. More concretely, we say that e.g. $Z_1(\omega; \theta_{EC,1})$ has a scaling symmetry if for every parameter setting $\theta_{EC,1}$ and every scaling factor x , then there exists another parameter setting $\theta_{EC,1}'$ such that $Z_1(\omega; \theta_{EC,1}') = xZ_1(\omega; \theta_{EC,1})$. With that said, let us come back to the discussion of serial options.

Serial options form a building block and can be applied as many times as required. The EC models thus obtained have two types of parameters. Namely, they have a set of switch variables which can only take values 0 or 1 and they have a set of usual parameters θ_{EC} , which can vary continuously. Given a bigger EC model with switch variables and an smaller EC model without switch variables, if there exists a set of choices for the switch variables that makes the bigger EC model equivalent to the smaller EC model, then we say that the bigger EC model *encompasses* the smaller EC model (with the given choices for the switch variables). In this way, an EC model with switch variables can be viewed as the set of all the smaller EC models which it can encompass with appropriate choices for the switch variables.

Also note that, for any finite set of small EC models, it is always possible to construct a single big EC model which can encompass all the given set. For instance, Figure 3.11 shows three different EC models, which can all be encompassed by a model which has a ZARC in series with two optional ZARCs. Similarly, to encompass all the models within Figure 3.11 and Figure 3.12 simultaneously, it would suffice to have a serial option of a “CPE in series with a ZARC” and a “ZARC with a nested CPE” instead of “ZARC 1” and “Warburg” in the Figures.

Then, the actual values of the switch variables *is* the representation of ID_{EC} since it determines which small EC model is encompassed by the big EC model. Furthermore, $\theta_{EC,Universal}$ is simply the usual parameters of the big EC model, and the projection to θ_{EC} for any given value of ID_{EC} (i.e. for any choice of a small EC model) is straightforward, amounting to removing the positions in the vector $\theta_{EC,Universal}$ which are not used by the given choice of EC model.

Then, as it turns out, the training procedure is precisely the same as the unsupervised approach from Section 3.5, except that every time a spectrum is presented to the model, a random choice of

ID_{EC} is made, and fed to the inverse model, with the same choice used to produce $Z_{reconstructed}(\omega)$ (see Figure 3.10). For the neural networks described in Section 3.7, the way to include ID_{EC} as an input to the neural network is to replicate it and append it to every observed frequency, so that the neural network will receive $\{(ID_{EC}, \omega_i, Z_i) | i = 1, \dots, m\}$ as input. This is because the neural network is comprised of local processes, each of which may need to adapt depending on the choice of ID_{EC} .

This approach is essentially representing the wanted circuit as a vector, and teaching the network to interpret this representation properly⁷³.

Minimal Options	Full Options	Circuit
<pre> —inductance —zarc_inductance </pre>	<pre> —inductance —zarc_inductance —num_zarcs=3 —no_warburg_inception </pre>	
<pre> —inductance —zarc_inductance —num_zarcs=2 </pre>	<pre> —inductance —zarc_inductance —num_zarcs=2 —no_warburg_inception </pre>	
<pre> —inductance —zarc_inductance —num_zarcs=1 </pre>	<pre> —inductance —zarc_inductance —num_zarcs=1 —no_warburg_inception </pre>	

Figure 3.11 Various EC models supported by the software (part 1). Note the decreasing complexity of the EC models from top to bottom which is the inspiration behind the notion of complexity presented in Section 2.7. More spectra can be represented by the EC model on the top row than on the bottom row, but every spectrum that can be

represented on a given row can still be represented on the rows above. This can easily be achieved by setting the resistance of one or more ZARC element to 0. This would in turn be reflected in the complexity metric of Section 2.7. For instance, on the first row, setting a single ZARC's resistance to 0 would force the complexity metric to be less than or equal to 2, and setting two ZARC's resistances to 0 would force the complexity metric to be equal to 1. In fact, enforcing a complexity metric of 1 guarantees that two ZARC's resistances must be set to 0. Note however that a complexity metric less than 2 may also be achieved with 3 non-zero ZARC resistances (i.e. 98, 1, and 1).

Minimal Options	Full Options	Circuit
<ul style="list-style-type: none"> —inductance —zarc_inductance —warburg_inception 	<ul style="list-style-type: none"> —inductance —zarc_inductance —num_zarcs=3 —warburg_inception 	
<ul style="list-style-type: none"> —inductance —zarc_inductance —num_zarcs=2 —warburg_inception 	<ul style="list-style-type: none"> —inductance —zarc_inductance —num_zarcs=2 —warburg_inception 	
<ul style="list-style-type: none"> —inductance —zarc_inductance —num_zarcs=1 —warburg_inception 	<ul style="list-style-type: none"> —inductance —zarc_inductance —num_zarcs=1 —warburg_inception 	

Figure 3.12 Various EC models supported by the software (part 2).

3.6.1 Multi-Task Learning, Positive Transfer, and Negative Transfer

Using a single neural network to solve the fitting problem for multiple EC models is an architectural choice which may well have an impact on the performance of the system.

This choice is akin to the choice of solving the global fitting problem with a single model instead of solving each individual fitting problem separately.

In general, there are multiple tasks to be solved and there is a choice between solving them with independent neural networks or combining them into a single more challenging task and solving it all with a single model. There are also hybrid possibilities where parts of the neural network are independent for each task and parts are shared but we focus on the two extremes for now.

The question of whether a choice is favorable to another with respect to performance, robustness, data requirements, etc. is referred to as the *valence of transfer*^{74,75} between the individual tasks. The two main possibilities are *positive transfer*⁷⁶, in which case each task benefits from being solved together with all other tasks in a single model (i.e. the best choice is a single model), and *negative transfer*⁷⁷, in which case each task suffers from being solved together with all other tasks (i.e. the best choice is independent models for each task).

This question is ultimately empirical, but how to develop an intuition for whether a multi-task setting would lead to positive or negative transfer? We sketch a mental model to understand the problem:

- For each specific task, there may be various viable solutions on the limited dataset available. Multiple tasks may share *general solutions*, but there may also be *specialized solutions* for given tasks. Each learning process has some probability of producing a given solution.

- When going from individual models for each task to a single multi-task model, the probability associated with general solutions will *increase* and the probability associated with specialized solutions will *decrease*.

The extent to which general solutions can compete with specialized solutions on performance, robustness, etc. will determine if positive transfer or negative transfer will be observed.

- In the best of cases, the general solution displays better performance than specialized solutions and has a reasonable probability of being produced by the training process. This would entail significant positive transfer.
- In the worst of cases, there is no competitive general solution and the multi-task model simply struggles more to produce specialized solutions for each task. This would entail significant negative transfer.

In the case of supporting multiple EC models, there are two convincing arguments to expect significant positive transfer.

- First, a general purpose optimizer is a decent solution to the individual fitting problem for a very large set of possible EC models, and the global coordination problem could be reasonably solved through a penalty term for lack of smoothness of the inverse model, which can be expressed similarly for a very large set of possible EC models.
- Second, Section 2.2 presented various conversion formulas, allowing the solution to the fitting problem for a given EC model to be leveraged to solve the fitting problem for a different EC model. The existence of simple formulas in the simple cases leads one to expect that generalizations thereof would be straightforward to learn in the context of a multi-task model. In other words, the easiest fitting problem among various equivalent such

problems for various EC models could be solved, and then the solution could be converted to solve the fitting problem for the other EC models.

Though these arguments likely do not truly capture the actual solution learned in practice by the multi-task model, they nevertheless should increase our confidence in the positive transfer hypothesis. Empirically, this hypothesis was confirmed.

3.7 Key Implementation Details and Intuitive Guide to their Impact on Performance

Of course, after reading Section 1.2, the reader⁷ will know that “neural network” is a generic term and more details are needed to distinguish between a *good* versus a *poor* implementation.

3.7.1 What Separates Good from Bad Choices of Neural Networks

For a given problem, different choices of neural networks will principally vary by:

- Their ability to represent arbitrary relations between their inputs and outputs (called the network capacity).
- The difficulty of the optimization problem their training process defines (called the optimizability).
- Their computational requirements (memory, time, etc.).
- Their data requirements or data efficiency (i.e. the amount and quality of data needed to achieve a given accuracy).

3.7.2 Symmetry and Convolutional Layers

Since the spectra sampled from an EC model together with the EC model parameters admit of a symmetry under translation in log frequency space (see Section 2.5), and based on the way humans

visually estimate the parameters of an EC model, it is reasonable to expect that a convolutional architecture would increase the data efficiency without compromising too much the network capacity when compared with a fully-connected architecture. Indeed, for a fixed computational budget, a convolutional architecture (i.e. using convolutional layers) may improve the network capacity.

Convolutions have two important features:

1. They have a local receptive field⁷⁸ (i.e. their operation is applied to parts of the input at a time, see Section 1.2.6).
2. The operation applied at every point is the same.

Namely, instead of representing an arbitrary function from a sequence of input features $\{(\omega_i, Z_i) | i = 1, \dots, m\}$ to a sequence of output features $\{F_i | i = 1, \dots, m\}$, they represent a smaller function f (say from 3 neighboring vectors of input features to 1 vector of output features), and the output features are each computed according to $F_i = f((\omega_{i-1}, Z_{i-1}), (\omega_i, Z_i), (\omega_{i+1}, Z_{i+1}))$.

The boundary conditions must be dealt with somehow, since for F_1 , we do not have $(\omega_{1-1}, Z_{1-1}) = (\omega_0, Z_0)$ available, but this is not an interesting choice as it does not affect performance much. Essentially, any request of input features that do not exist are replaced by zeros.

In general, determining the choice of neural network architecture is an empirical problem. But when a lot is known about the underlying setting, it is possible to construct strong arguments to determine which architecture is most suitable, and this type of thinking is useful in the context of lithium-ion research.

Therefore, let us imagine a solution to the fitting problem (i.e. an inverse model) using a convolutional architecture.

First, imagine there exists a non-convolutional approximation to the optimal inverse model which takes as inputs fixed sequences of log-frequencies, and impedances and returns EC parameters in the reparameterized space discussed in Section 2.4. Without loss of generality, consider odd sequences of length $2m_{\text{fix}} + 1$, and express this inverse model as $f_{\text{fix}}\left((w_{i-m_{\text{fix}}}, Z_{i-m_{\text{fix}}}), \dots, (w_i, Z_i), \dots, (w_{i+m_{\text{fix}}}, Z_{i+m_{\text{fix}}})\right)$

It seems reasonable to assume that such inverse models can be selected for any m_{fix} . Furthermore, though they may not be as accurate, inverse models with smaller m_{fix} belong to much smaller function spaces, and hence are more data efficient than their counterparts with larger m_{fix} . As shall be seen, a reasonable approximation to larger models could be obtained by combining smaller models in a way easily expressed with a convolutional architecture.

First, the symmetries discussed in Section 2.5 (e.g. Trans_α) can be applied to the input and the output of f_{fix} such that, if $f_{\text{fix}}\left((w_{i-m_{\text{fix}}}, Z_{i-m_{\text{fix}}}), \dots, (w_i, Z_i), \dots, (w_{i+m_{\text{fix}}}, Z_{i+m_{\text{fix}}})\right)$ is a good inverse model, then so too should

$$\text{Trans}_{w_i} f_{\text{fix}}\left((w_{i-m_{\text{fix}}} - w_i, Z_{i-m_{\text{fix}}}), \dots, (w_i - w_i, Z_i), \dots, (w_{i+m_{\text{fix}}} - w_i, Z_{i+m_{\text{fix}}})\right)$$

In the reasonable case where the log frequencies are equally spaced with spacing Δw , this would correspond to

$$\text{Trans}_{w_i} f_{\text{fix}}\left((-m_{\text{fix}}\Delta w, Z_{i-m_{\text{fix}}}), \dots, (0, Z_i), \dots, (m_{\text{fix}}\Delta w, Z_{i+m_{\text{fix}}})\right)$$

In other words, a good inverse model fundamentally only depends on the spacing between the frequencies, and not on the absolute values of the frequencies.

Furthermore, more or less all of the serial components of the EC model of Figure 2.5 have a relatively small frequency range over which their impedance is not almost constant. Since the

information necessary to estimate a given EC parameter only exists on a smaller range of frequencies, it would likely be feasible to produce a good inverse model in two stages:

1. Produce $\{(w_i, \theta_{EC,i}) | i = 1, \dots, m\}$ where $\theta_{EC,i}$ is the output of smaller f_{fix} (with $m_{fix} < m$) applied to each localized subsets of the input $(w_{i-m_{fix}}, Z_{i-m_{fix}}), \dots, (w_i, Z_i), \dots, (w_{i+m_{fix}}, Z_{i+m_{fix}})$.
2. Produce θ_{EC} by using $\{(w_i, \theta_{EC,i}) | i = 1, \dots, m\}$ as inputs.

When combining the previous two considerations, the decomposition could be:

1. Produce $\{(w_i, \theta'_{EC,i}) | i = 1, \dots, m\}$ where

$$\theta'_{EC,i} = f_{fix} \left((-m_{fix}\Delta w, Z_{i-m_{fix}}), \dots, (0, Z_i), \dots, (m_{fix}\Delta w, Z_{i+m_{fix}}) \right)$$

is only determined by the “shape of the localized impedances”.

2. Produce θ_{EC} by using $\{(w_i, \theta'_{EC,i}) | i = 1, \dots, m\}$ as inputs.

Let us consider how the global estimate of the EC parameters may be simply constructed from local estimates $\{(w_i, \theta_{EC,i}) | i = 1, \dots, m\}$.

For instance, it could be done by averaging each parameter across frequency, or by averaging across frequency according to an estimate of the relevance of each localized subsets of the input $(w_{i-m_{fix}}, Z_{i-m_{fix}}), \dots, (w_i, Z_i), \dots, (w_{i+m_{fix}}, Z_{i+m_{fix}})$ for the prediction of a given EC parameter.

There may be a need to correlate the predictions across various frequencies in order to reliably solve ambiguity problems, but it may still be a simpler task to learn this correlation than to learn f_{fix} for m inputs.

In general, by allowing each localized model to output a general vector of features, the model gains in flexibility without sacrificing significant data efficiency.

But this decomposition of an inverse model with many inputs into the application of many smaller inverse models can be repeated for the individual smaller inverse models (using the same arguments).

In the end, this solution to the problem is quite natural to represent as a convolutional architecture.

3.7.3 Variable Numbers of Frequencies and Fully-Convolutional Architecture

Since the number of frequencies measured per spectrum varies across the dataset, we have chosen a *fully convolutional*^{79,80} neural network architecture, which means that at no point in the sequence of operations does a layer of computation require a fixed input length (though such layers may apply an operation with fixed input lengths repeatedly to a sequence). In practice, this means the layers are either averages over frequencies, or convolutions over frequencies, or generalized averages over frequencies known as attention mechanisms.

As is standard practice with most architectures, various details have been chosen to improve optimizability such as residual blocks⁶⁴, batch renormalization⁸¹, and dropout⁸².

3.7.4 Coordination of Local Processes

Until this point, the neural network can be thought of as a local process which estimates all the EC parameters on subsets of the observed spectra centered around each frequencies, and finally obtains a coherent guess by averaging all the local guesses with equal weight. This does work relatively well for this problem, but in general, it intuitively seems like local processes would have a difficult time coordinating their outputs for parameters which require a global view of the measured spectrum. As a step to allow a bit more coordination between local processes, instead of averaging with uniform weights, the weights are themselves produced by the local processes, and then normalized globally (this is known as attention, or attentive³³ pooling).

3.7.5 Batch Diversity and Masks

Just like it is important to have a diverse dataset to train a neural network robustly, it is also important to produce diverse batches of datapoints in order to compute gradients in a stable way and improve the optimizability of the neural network. Usually, each datapoint has the same dimensionality, and therefore it is quite simple to create such batches: if a datapoint is a tensor with N_1 elements by N_2 elements, ..., by N_m elements, then B such datapoints can be assembled into a tensor with B elements by N_1 elements by N_2 elements, ..., by N_m elements.

However, in the case of impedance spectra, each spectrum may have a different number of frequencies. One solution would be to separate the dataset into subdatasets each only containing spectra with a fixed number of frequencies, allowing the formation of batches of datapoints belonging to a single subdataset.

As it turns out, this greatly reduces the diversity of the possible batches, thus introducing noise into the computed gradients, and reducing the optimizability.

Instead, notice that it is possible to insert a smaller tensor into a bigger tensor by simply adding zeroes into the remaining positions. For instance, it is possible to represent a spectrum $\{(\omega_i, Z_i) | i = 1, \dots, m\}$ of m frequencies as an “ m by 3” tensor, but it is also possible to represent it as an e.g. “ $m + 10$ by 3” tensor (simply by adding zeroes at the end).

In order to keep track of which frequencies are actual and which were just added to make a spectrum fit into a larger tensor space, we add a mask variable mask_i which will equal 1 if the frequency is actual and will equal 0 if the frequency was added artificially. As a concrete example,

imagine the spectrum $\{(-1, -2 + -3j), (1, 2 + 3j), (4, 5 + 6j)\}$. When representing it as a tensor

with 5 frequency positions, it would end up being a “5 by 4” tensor given as
$$\begin{bmatrix} -1 & -2 & -3 & 1 \\ 1 & 2 & 3 & 1 \\ 4 & 5 & 6 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

This flexibility allows to combine any set of B spectra, compute the maximum number of frequencies N_f for that set of spectra, and construct a batch as a “ B by N_f by 4” tensor. As mentioned in Section 3.6, for the case where multiple EC models are supported, each frequency would also receive the set of switch variables, and so the last dimension of the tensor would be larger than 4.

Then, this mask⁸³ must integrate with the various components of the neural network. Most obviously, when computing the mean squared error of reconstruction, each frequency term must be multiplied by the corresponding mask to ensure that the added frequencies do not contribute.

Furthermore, when taking averages across frequencies, the weights are multiplied by the masks to ensure that the local processes centered around an artificial frequency do not influence the averages.

Finally, for the fully convolutional part of the neural network, it turns out to be sufficient to simply feed the mask variables as an input to the neural network. During training a given spectrum will be encountered within different batches of different sizes, and so with a sufficiently wide and deep model, the neural network can learn the irrelevance of the artificial frequencies. The alternative is to carefully craft each layer of convolution to ensure their output are unchanged by the addition of artificial frequencies.

As a matter of computational efficiency, it is faster to sample from a single dataset, so in practice all spectra are represented as large tensors with potentially many artificial frequencies and the whole dataset is kept as a single tensor. Furthermore, the number of actual frequencies for each spectrum is stored into another tensor. Then when a batch is selected, the maximum number of frequencies across the batch is computed, and the batch tensor is sliced (i.e. some artificial frequencies are removed from all the spectra) to the smallest such tensor which still contains all the actual frequencies.

3.7.6 Rebalancing the Dataset

In the case of the impedance spectra dataset (see Section 2.3), there were two distinct sources of data with quite different properties (referred to as the EIS dataset and the FRA dataset), but at bottom, the goal is to find an inverse model which works robustly across many different spectra and these datasets are the only tools we had to accomplish this goal.

As it turns out, there were approximately 10 times more spectra in the FRA dataset. This means that if gradients were computed by sampling spectra at random, the model would have a clear incentive to focus almost exclusively on spectra found in the FRA dataset. A similar trade-off could apply when considering the artificially generated spectra as a separate, essentially infinite dataset.

Often, **the prevalence of a type of data is imperfectly correlated with the importance it should have in shaping the neural network** (mathematically, the coefficient multiplying the term in the loss function corresponding to a type of data represents its importance). In our case, the goal is to have an inverse model which performs just as well on data similar to FRA dataset or to EIS dataset. To better control the importance of each type of data, one must be able to sample a datapoint at random from each type of data. At the most basic level, **the EIS dataset and the FRA dataset**

should be kept separate, and then when the stochastic gradient algorithm used to select the neural network requires a batch of N spectra, a separate batch of $\frac{N}{2}$ spectra should be created from each dataset, and then these batches should be combined.

For the model in this Chapter, all sources of data were sampled more or less equally (see Section 3.4 for the actual rebalancing), but for different applications, it might be better to rebalance differently^{84–86}.

An important consideration when the number of datapoints in a given group is small is that by rebalancing the dataset in this way, not only do you control the relative weight in the loss function for every group of data, but also you affect the quality and diversity of the dataset. An extreme example would be the case where one group has only one datapoint, and the second group has one million datapoints. Then, by rebalancing the dataset equally between these groups, the quality and diversity of the dataset is dramatically reduced (the neural network will encounter a single datapoint over and over).

3.7.7 Finetuning with ADAM

Once sufficiently accurate estimates for the EC parameters of given spectra have been produced by an inverse model, it is possible for a simple optimizer to increase the accuracy of those estimates with respect to the mean squared error of reconstruction without the usual drawbacks of the individual fitting problem. Indeed, by combining good initial guesses (i.e. close to a local minimum) with a restricted optimizer (i.e. only small adjustments), we can improve the reliability of the system. For a sufficiently powerful inverse model, this finetuning step is unnecessary, but robustness is about reducing the requirements on the challenging parts of the system.

Having decided to implement a solution to the individual fitting problem on top of the global fitting problem, the question is how to do so without unnecessary complication.

To this end, we present a way of viewing the individual fitting problem as a special case of the global fitting problem, such that the implementation of both systems may share many redundant components.

Imagine a matrix with as many rows as there are spectra in a given dataset and where each row contains a vector of numbers which could be interpreted as EC parameters. Each spectrum has a unique index (the row number).

How can such a matrix be interpreted as an inverse model? When presented with a given spectrum, instead of using the spectrum itself as an input, it is given the index of the spectrum, and simply outputs the row of numbers corresponding to that index. This “inverse model” can only be applied to the specific spectra in the training dataset.

Note that the training process for such an inverse model corresponds precisely to solving individual fitting problems for each spectrum independently, using the optimizer of the training process. Furthermore, the initial values in a given row of the matrix correspond to the initial guesses for the EC parameters of the corresponding spectrum.

Typically, neural networks are optimized with ADAM, which means that the same training procedure can be applied to this matrix of parameters in order to achieve finetuning on individual fitting problems.

Models which grow with the dataset and do not represent the relationship between input and output using a parameterized function directly are sometimes called nonparametric models.

The main advantages of implementing the finetuning in this way are the simplification of the code and the ability to use the fast parallel computations of neural network libraries without additional work.

Note that it would also be possible to include a gradient descent procedure within our neural network and train this procedure using ADAM; this type of learned optimization⁸⁷ is interesting but in the overkill category for this work.

3.8 Future Work: The Transformer Architecture

3.8.1 The Transformer Architecture as a Coordination Mechanism

In retrospect, given the recent successes of architectures fully based on attention (i.e. the Transformer^{31,33,88}), it seems reasonable that coordination between local processes would be easily achieved by adding a few Transformer³¹ layers after the fully convolutional layers, followed by attentive pooling. Such layers also admit inputs with varying sizes (a basic form of these layers has been introduced in Section 1.2.7) and the fact that every element of the sequence can “interact” with every element of the sequence in every layer is a robust way to allow the neural network to capture arbitrary interactions and coordination across the sequence.

As a future work, the Transformer³¹⁻³³ architecture could be included in the inverse model. In such case, it would be necessary to consider positional encodings as well. This is because situating two samples in a spectrum by their characteristic frequencies is not data efficient for a dot-product attention mechanism (i.e. the attention mechanism of the Transformer, see Section 1.2.7). What matters is the difference between frequencies, but the attention mechanism of Transformers computes similarity between vectors, and therefore the frequencies by themselves are awkward to work with.

3.8.2 The Dual Role of Log-Frequencies as Inputs of Sequence Layers

In the inverse model, the log frequencies within each element of the spectrum are useful for two reasons:

1. Each local process of estimation of the parameters which transform during a log-frequency shift can use the input log-frequency of the center sample to adjust its prediction. For instance, the same shape in the real and imaginary part of impedance seen by a local process must lead to shifted predictions for the characteristic log-frequencies if all the log-frequencies are shifted. See Section 2.5.
2. The “shape” of the impedance spectrum is a feature of the relationship between impedances measured at different log-frequencies. Therefore, there must be a way to compute distances between the log-frequencies at which various impedances were measured.

For the fully-convolutional architecture, the first role of log-frequency is well accomplished by the log-frequency itself (see Section 3.7.2). However, the second use of these log-frequencies is not needed, because convolutional architectures have fixed expectations for the spatial relationship of their inputs (see Section 3.7.2). All that is needed is the spacing between the log-frequencies. Since the convolution is a linear function of its inputs, it can internally compute differences of log-frequencies and therefore has access to all the necessary information in a convenient form.

Contrast this with the self-attention dot-product architecture (i.e. Transformer). Shifting the predictions based on the log frequency at the “center” sample (i.e. the first task) would still easily be accomplished directly on the log frequency as an input. However, determining the distance between the “center” and a given sample (i.e. the second task) could not be done in a data efficient manner by the Transformer’s attention process, which only computes dot-products between

vectors. For instance, there would be no exact way to compute the difference in log-frequencies such that it would be invariant to a uniform shift of all the log-frequencies. Therefore, in order to allow distances between samples to be computed within the self-attention mechanism, some additional values would have to be appended to each sample. These values, referred to as *a positional encoding*⁸⁸⁻⁹⁰ of in this case the log-frequency, are vectors.

3.8.3 Positional Encodings of Log-Frequencies

Each log-frequency is mapped to a vector of values. Each element i of the vector is a periodic function of the log-frequency, such as the complex number $\exp\left(j\frac{2\pi w}{p_i}\right)$, but each element has a unique period p_i . Alternatively (like in the original paper and most commonly in the machine learning literature⁸⁹), each period p_i can be shared by two real numbers in the output vector, namely $\sin\left(\frac{2\pi w}{p_i}\right)$ and $\cos\left(\frac{2\pi w}{p_i}\right)$. Typically, the periods form a geometric progression such that $p_{i+1} = \frac{p_i}{2}$. This representation has empirically been shown to allow learning of dependence on distance between positions in a data efficient way.

In case the reader is puzzled by how the information of the distance between the two positions is accessed by the attention mechanism, consider the following illustrative example.

When multiplying elementwise two of these positional encodings (one for a log-frequency w and the other for a log-frequency w') in the natural way for complex numbers (the first number times the conjugate of the second), one gets $\exp\left(j\frac{2\pi w}{p_i}\right)\exp\left(-j\frac{2\pi w'}{p_i}\right) = \exp\left(j\frac{2\pi(w-w')}{p_i}\right)$, a quantity which only depends on the difference between the two positions (in this case, the log-frequencies). Both the real and imaginary parts will vary between 1 and -1 periodically on a scale given by p_i . By choosing $p_0 = 1$, we get a continuous and complex analog to the binary representation of

$$w - w'$$

where each position in the binary expansion becomes either 0 or 1 with a smaller and smaller period. Explicitly, each element of the positional encoding would give rise to a positive number in the case that the binary coefficient would be 0 and a negative number when the corresponding coefficient would be 1.

By applying a projection of the positional encodings onto one of their components (a linear operation), the dot product will yield a continuous “detector” of the distance between positions $w - w'$ with periodicity p_i .

Therefore, with a relatively simple example of a query projection, the attention mechanism could depend on the distance between the log-frequencies at many scales. Similarly to a Fourier series, various complicated dependencies on $w - w'$ could be obtained by a linear combination of simple dependencies on this decomposition into features of various periodicities. In the literature, the positional encodings do not use complex numbers since the same effects may be obtained in a less direct way with different linear operations but only using real numbers.

3.8.4 A Concrete Proposal for Implementing the Transformer Architecture for Future Researchers

Now that the various pieces of the transformer puzzle have been discussed, future researchers might be interested in trying it. Below, we describe a concrete proposal for future researchers to hopefully get started.

The architecture will represent a spectrum as a matrix with one dimension varying with frequency and one dimension containing the various pieces of information such as the real and imaginary parts of the impedance, the log-frequency itself etc.. Then, this matrix will be passed

through a standard Transformer sequence-to-sequence model, ending with again a matrix where one dimension varies with frequency and the other varies with the “channels” of information.

Then, there will be some sequence-to-vector operation for instance the Transformer used with an output sequence of only one element or alternatively some simple averaging. Finally, there will be a projection into EC parameter space.

This directly corresponds to the convolution-based implementation available at the moment, so the code can be a good starting point here.

Since the transformer architecture to be used is in fact the standard one, we point to the pedagogically excellent implementation <https://github.com/karpathy/minGPT>.

To turn this into a concrete proposal, what remains is to 1) describe the non-standard use of masking within the transformer, 2) describe the precise way in which the initial sequence must be given to the transformer, and 3) give some practical advice about the code.

First the masking. As discussed in Section 1.2.7, any given output of the self-attention mechanism will be a weighted average of the values across the sequence. In the sequence-to-sequence context, these weights will be a tensor with one index for potentially multiple spectra in a batch, one index for the multiple heads of attention, one index for the output position in the sequence, and one index for the input position in the sequence. This tensor is usually referred to as the attention mask and in the context of transformer, we start with so called attention logits (produced by dot-product attention) and apply a softmax along the dimension of input position in the sequence to obtain the attention weights.

This attention mask allows some information to travel from an input position to an output position. If the corresponding weight is 0, then no information travels. For numerical stability

reasons, we typically make the attention logit equal to negative infinity before taking the softmax in order to ensure that no information flows from an input position to an output position. This is called *masking*. Confusingly enough, we would say that we can *mask the attention mask* in order to artificially ensure that no information may travel from a given input position to a given output position. In the language modelling implementation <https://github.com/karpathy/minGPT>, it makes sense to mask any of the attention logits linking a given input position to some earlier output position since the model is supposed to forecast the next word based on all previous words. In our case, this would be detrimental. Therefore, we do not use this type of masking. However, when gathering multiple impedance spectra into a “minibatch” during training, some will have different lengths, and therefore we already have the concept of a mask specifying which position in the sequence corresponds to a measured frequency and which position is just padding.

In the case of a convolutional architecture, it made sense to pass such mask as an input channel, but for the transformer, it does not make sense. Instead, this mask should be used to always mask the attention mask within the transformer so that only positions corresponding to valid measured frequencies could send information to any position.

Similarly, for the last transformation from sequence to vector, a transformer architecture could be used in the setting of sequence-to-vector, and positions in the sequence not corresponding to actual measurements should be masked in the attention mask.

This is how the transformer could be used where the convolutional network was used before. However, now that the concept of masking the attention mask has been introduced, we can do something even better. Namely, during training, when trying to minimize the distance between measured and reconstructed spectrum, we could generate a random mask and use it to further

mask the attention mask of the transformer. This would correspond to trying to estimate the EC parameters from partial observations of the impedance spectra. By turning off the low frequencies systematically, one could force the model to develop some extrapolation capability to infer diffusion parameters based on correlations with other parts of the circuit visible at high frequencies (and vice-versa). By turning off every other frequency, this would force the model to develop some interpolation capability to infer EC parameters which would not act erratically in between observed frequencies. These are simply some illustrative special cases. Generally speaking, one could obtain all these effects by just sampling a random mask every time a spectrum is presented to the network during training. Note that this effect cannot be easily achieved with a convolutional architecture which relies on the spacing between sequence elements.

Second, the presentation of the initial sequence. Concretely, for each frequency, there will be a fixed set of “channels” and this will create a sequence of vectors or matrix to be passed to the transformer. It is crucial to provide the real part of impedance, the imaginary part of impedance, as well as the logarithm of the frequency. This is because some of the EC parameters are not invariant to a translation in log-frequency (for instance the characteristic frequencies of the ZARC elements would change). Beyond this, it is crucial to provide, at each frequency, a copy of the Equivalent Circuit identity in the form of the switch variables discussed in Section 3.6. As stated earlier, it does not make sense to provide the masks indicating measured frequency versus padding since only the measured frequencies will have impact on the answer (and therefore this input channel would effectively be constant across all inputs).

There needs to be some mechanism to do frequency distance comparisons in a translation-invariant way. As discussed in Section 3.8.3, positional encodings should be used here. The

simplest way to implement this is to compute a vector of scaled log-frequencies (i.e. first element is the log frequency, second element is 2 time the log frequency, third element is 4 times,...) and then compute the sine (call it vector A) as well as the cosine (call it vector B). Then the vector A and B would be concatenated onto the channels for each position in the sequence. It is not necessary to introduce these positional encodings at every layer of the transformer; it should only be placed in the very first input.

Once this sequence of vectors has been assembled for each spectrum in the “minibatch”, a linear projection should be applied to every element of the sequence to make the number of channels (i.e. the dimension of the vectors in the sequence of vectors) match what the transformer will operate on. This can be accomplished with a convolution with kernel size of 1 (i.e. a convolution that only depends on a single element of the sequence at a time). This would be the first layer of the neural network, and then the data is ready to be fed into a standard transformer implementation.

Third, the advice about the code. The original paper³⁹ was first implemented using Tensorflow version 1 and then rewritten to work with version 2. However, now that the author has tried using Pytorch¹⁰ instead, he can attest that redoing all of this in Pytorch would be a good thing. Furthermore, the author highly recommends the combination of Pytorch with Einops⁹¹ to allow one to write the various tensor operations much closer to the mathematics, to get much easier to understand error messages when something goes wrong, and to trivially insert sanity checks about the dimensions of various tensors at crucial points in the code. To the extent that it is easier to start with a working model, using the available implementation of this Chapter’s model might be good, but if a future researcher is interested in moving forward and tackling their own projects, one should consider at least trying Pytorch in combination with Einops.

3.9 Conclusions

This chapter presented a general paradigm to automate the fitting of empirical data to physical models, namely to determine an inverse model parametrized with a deep neural network by directly minimizing the mean squared error of the reconstructed empirical data, with a successful application to EC model fitting of impedance spectra of lithium-ion cells (a failure rate of less than 1% and good fit quality on two large and diverse datasets with a single inverse model and using ADAM to finetune the EC parameters). Crucially, this method does not require knowledge of the true EC parameters corresponding to the empirical data, allowing the use of generated data, as well as any available impedance spectra to train the inverse model. This makes the method easy to implement, as well as being flexible.

This application allowed us to illustrate the process by which deeper understanding of the underlying application domain may be leveraged to produce robust machine learning solutions.

3.10 Acknowledgements

This work was supported financially by the Natural Sciences and Engineering Research Council of Canada (NSERC), and Tesla Motors. Sam Buteau acknowledges scholarship support from NSERC. Furthermore, Sam Buteau would like to thank Prof. Yoshua Bengio and the Mila community for useful discussion groups and seminars.

Chapter 4 Interpretability in the Determination of the Electrolyte Concentration in Lithium-Ion Cells Using Fourier Transform Infrared Spectroscopy

Understanding the changes in the electrolyte during lithium-ion cell aging is valuable to improve longevity. Studying this in hundreds or thousands of cells requires a fast and widely available measurement such as Fourier Transform Infrared Spectroscopy (FTIR) of electrolyte samples. This Chapter presents a machine learning model to determine electrolyte composition from FTIR measurements. A carefully prepared dataset of mixtures of 5 electrolyte components (i.e. LiPF₆, EC, EMC, DMC, and DEC), and the code to replicate and extend the model to different electrolyte mixtures are made available. With this model, the mass ratio of salt to total is predicted within an error of 0.4%, and each solvent's mass ratio to total is predicted within an error of 2%. Furthermore, a spectrum calculated based on the predicted component ratios can be compared to the measured spectrum which allows one to detect if unexpected species are present in the electrolyte in significant quantity. A model for mixtures of 5 components can be calibrated well with between 25 and 50 carefully prepared samples so this work can be extended to other systems by simply adding more data and retraining.

This Chapter provides another example of applying machine learning to lithium-ion research, which contrasts with the example contained in Chapters 2 and 3 by only having access to a small dataset (20 to 50 carefully measured samples). As such, it illustrates how to improve the reliability of the model *under duress* or when *some degree of inaccuracy is inevitable*. This is done using simple physical models and various additional penalties (other than minimizing prediction errors) along with some other design tricks.

For some inputs, the “axioms” of the system are respected and so the system can be trusted to have a reasonable accuracy, while for others, the “axioms” of the system are not respected and the system may have poor accuracy. Contrary to many machine learning prediction systems, *our system knows on which inputs its “axioms” are respected and on which they are not.* More specifically

- Instead of always giving confident predictions, the model accompanies its predictions with a measure of its confidence.
- The predictions are accompanied by an *explanation* to make them interpretable.
 - When the model has high confidence, the explanation can be read as: “I believe that the concentration of each components is **this**; at these concentrations, I believe that each component’s contribution should be **this**, which *appropriately reconstructs the measurement.* i.e. my reconstruction for this sample falls within the typical reconstruction error **throughout the spectrum.**”
 - When the model has low confidence, the explanation can be read as: “My best estimate for the concentration of each components is **this**; at these concentrations, I believe that each component’s contribution should be **this**, but it does *not appropriately reconstruct the measurement.* i.e. my reconstruction for this sample falls outside the typical reconstruction error in **these regions of the spectrum.**”
 - In cases where the model’s confidence is low, it is possible, through finetuning, to determine to what extent the error is due to a wrong prediction or to a wrong reconstruction, though this is left as future work.

- The system’s “axioms” can be characterized and understood to a much larger extent than those of a typical neural network, which allows greater trust even when the dataset only allows weak testing.

This chapter is entirely taken from a corresponding paper⁹² (henceforth called *the corresponding article to Chapter 4*) with the only exception of Section 4.2.6 discussing the use of the model on data taken with different experimental settings and Section 4.2.7 which offers a straightforward adaptation of the fine-tuning technique described in Chapter 3 as future work. The other modifications are only to make the thesis more uniform in style. Unfortunately, “EC” stood for “Equivalent Circuit” in Chapters 2 and 3, and it stands for “ethylene carbonate” in the corresponding article to Chapter 4. To remove possible confusion, the acronym has been replaced by Equivalent Circuit in section 4.2.7 or “ethylene carbonate” in the rest of Chapter 4 whenever confusion might occur. **EC never stands for “Equivalent Circuit” within the bounds of Chapter 4.**

4.1 Introduction

Understanding the evolution of the electrolyte during charge-discharge cycling or storage of lithium-ion cells would be valuable to design improved cell chemistries. However, as discussed in a previous paper⁹³, quantitative analysis of electrolyte solutions to determine composition typically employ nuclear magnetic resonance⁹⁴, gas chromatography⁹⁵ and other methods, which have several drawbacks for routine analysis at scales of hundreds to thousands of cells per month. Therefore, a method based on a fast, simple, and inexpensive measurement is useful to develop, with Attenuated Total Reflectance Fourier Transform Infrared Spectroscopy (ATR-FTIR spectroscopy, or simply FTIR) being a prime candidate. The previous paper mentioned earlier⁹³ focussed on a prototype system, only looking at electrolytes consisting of the salt lithium

hexafluorophosphate (LiPF_6), the single linear carbonate dimethyl-carbonate (DMC), and ethylene carbonate (EC). The previous study provided a proof of concept despite being flawed in a few respects:

1. The units of molarity and solvent volume ratios used in that study are dependent on temperature and are not used in industry. Instead, the mass ratios of each component to the total mass is a more robust unit.
2. The samples were prepared by serial dilution, either by hand or with a robot, which allowed the rapid production of a large dataset of samples. However, it was later realized and confirmed experimentally that the rate of evaporation of DMC is substantial and therefore the concentrations produced were not precise. In summary:
 - a. Serial dilutions done by volume (mixing known volumes of solutions obtained themselves by having mixed known volumes of primary solutions) did not produce mass ratios which can be computed without knowing all the intermediate densities.
 - b. Our robot prepared samples with vials opened on the order of tens of minutes even after optimization of the robot's procedure. It was therefore not able to produce samples of known mass ratios, given the evaporation of DMC.
3. Because of these first two points and because of the choice of normalizing absorbance spectra in the analysis by the total absorbance, the article suggested non-linearities, **yet after more careful sample preparations (i.e. using known mass and minimizing evaporation) and thinking, a linear prediction model was deemed appropriate.**
4. It is desired to analyse electrolytes containing ethyl-methyl carbonate (EMC), which can undergo a transesterification process, requiring full mixtures of LiPF_6 , EC, DMC, EMC, and diethyl-carbonate (DEC) to be analysable.

5. The code and data required to reproduce that work and extend it were not made available.

The key contributions of the corresponding article to Chapter 4 can be summarized as:

1. More than 40 samples carefully prepared by hand and covering the whole space of LiPF₆, EC, EMC, DMC, DEC, minimizing evaporation, were prepared (see Table 1 through Table 4 for a listing of all the samples' mass ratios). The precise mass ratios for these samples are known. This dataset is included in the code (as an SQL⁹⁶ database).
2. A large dataset of more than 300 samples which were prepared by a robot. They span the whole space of LiPF₆, EC, EMC, DMC, DEC, but because of the preparation procedure, the mass ratios for these samples are not exactly known.
3. A new model which combines an understanding of the underlying physics and practical machine learning to yield a partly linear model that can be calibrated effectively given a small set of samples with known mass ratios together with (optionally) a large set of samples with unknown mass ratios within the same space of potential components.
4. Code to replicate the study and **to extend the model on different devices as well as to a potentially larger spaces of electrolytes by retraining the model on more data.**
5. An assessment of the model's performance and physical underpinnings. On samples not used for calibration, the model predicts the mass ratios of linear carbonates (EMC, DMC, and DEC) to within 2% error over the full range going from 0% to 100%, the mass ratio of ethylene carbonate (EC) to within 2% error over a range from 0 to 50% (the available dataset's upper bound on ethylene carbonate), and the mass ratio of LiPF₆ within 0.4% over a range from 0 to 20%, which corresponds to a relative error within 2% of upper bound.

The closest related work is a previous paper from the same lab as the author⁹³ which applies FTIR to analysing the electrolyte of aged lithium-ion cells in proof of concept form, but quantitative analysis using FTIR has been done mostly in the context of a straightforward application of Beer's law⁹⁷. Note that in the electrolyte system under study, Beer's law (which roughly states that absorbance is a linear function of concentration) is not directly applicable (see Section 4.2.2), yet it provides a decent baseline upon which we build. This Chapter combines the two approaches to yield a more practical, production-ready solution to analysing the electrolyte of aged lithium-ion cells.

4.2 Methods

4.2.1 Data acquisition

See the previous paper from the same lab as the author⁹³ for the details of equipment and electrolyte solution preparation. To apply the code directly, one ought to use similar settings for the FTIR apparatus, but see Section 4.2.6 for an addendum which removes this limitation. FTIR spectra were collected using a Cary 630 FTIR (Agilent Technologies) equipped with a germanium crystal attenuated total reflectance (ATR) accessory. Sixteen scans were collected for each background and sample measurement, at a resolution of 4 cm^{-1} , using MicroLab PC software. Fourier transforms were performed using HappGenzel apodization, Mertz phase correction, and a zero-fill factor of 2.

With this approach, a dataset of measured spectra has been collected with known mass ratios. Table 1 through Table 4 list all the samples' mass ratios.

LiPF6 mass ratio	EC mass ratio	EMC mass ratio	DMC mass ratio	DEC mass ratio
0	0	0	0	1
0.076	0	0	0	0.924
0.1519	0	0	0	0.8481
0.2279	0	0	0	0.7721
0	0.3333	0	0	0.6667
0.076	0.308	0	0	0.616
0.1519	0.2827	0	0	0.5654
0.2279	0.2574	0	0	0.5148
0	0.1	0.1	0.8	0
0.076	0.0924	0.0924	0.7392	0
0.1519	0.0848	0.0848	0.6785	0
0.2279	0.0772	0.0772	0.6177	0

Table 1 The weight ratios of the dataset samples with known weight ratios. (part 1)

LiPF6 mass ratio	EC mass ratio	EMC mass ratio	DMC mass ratio	DEC mass ratio
0	0	1	0	0
0.076	0	0.924	0	0
0.1519	0	0.8481	0	0
0.2279	0	0.7721	0	0
0	0.5	0	0	0.5
0.076	0.462	0	0	0.462
0.1519	0.424	0	0	0.424
0.2279	0.3861	0	0	0.3861
0	0.5	0	0.5	0
0.076	0.462	0	0.462	0
0.1519	0.424	0	0.424	0
0.2279	0.3861	0	0.3861	0
0	0.3	0.7	0	0
0.076	0.2772	0.6468	0	0

Table 2 The weight ratios of the dataset samples with known weight ratios. (part 2)

LiPF6 mass ratio	EC mass ratio	EMC mass ratio	DMC mass ratio	DEC mass ratio
0.1519	0.2544	0.5937	0	0
0.2279	0.2316	0.5405	0	0
0	0.25	0.05	0.7	0
0.076	0.231	0.0462	0.6468	0
0.1519	0.212	0.0424	0.5937	0
0.2279	0.193	0.0386	0.5405	0
0	0	0	1	0
0	0.25	0.05	0.7	0
0.0727	0.2318	0.0464	0.6491	0
0.1364	0.2159	0.0432	0.6045	0
0.2091	0.1977	0.0395	0.5536	0
0.0748	0	0	0.9252	0
0.1402	0	0	0.8598	0
0.215	0	0	0.785	0

Table 3 The weight ratios of the dataset samples with known weight ratios. (part 3)

LiPF6 mass ratio	EC mass ratio	EMC mass ratio	DMC mass ratio	DEC mass ratio
0	0.3	0	0.7	0
0.0699	0.279	0	0.6511	0
0.131	0.2607	0	0.6083	0
0.2009	0.2397	0	0.5594	0
0	0.3	0.7	0	0
0.0734	0.278	0.6486	0	0
0.1376	0.2587	0.6037	0	0
0.211	0.2367	0.5523	0	0

Table 4 The weight ratios of the dataset samples with known weight ratios. (part 4)

4.2.2 Physical underpinnings, Beer's law, and the Proposed Models

ATR-FTIR measurements produce spectra of absorbance as a function of wavenumber. The peaks in this spectrum correspond to vibration modes in the molecules of the sample under analysis. Indeed, infrared light interacts with a certain volume of sample, occupied by a given number of molecules, each contributing additively to the spectrum by their vibration modes.

When varying the concentration of various molecules, in a textbook case (which is not our case) the following hold:

1. The volume of interaction with the infrared light is constant with respect to concentration.
2. The vibration modes of a given molecule type is the same across the volume of interaction.
3. The vibration modes of a given molecule type **are constant with respect to concentration.**

Together, these give rise to Beer's law. In our case, the third point is problematic since peaks in the absorbance spectrum shift in wavenumber as a function of concentration, and this cannot be explained by constant vibration modes. However, **for some wavenumber regions**, the vibration modes are **more or less constant with respect to concentration.**

The corresponding article to Chapter 4 mostly proposed and described two models, since no improvement was observed with more complex variations.

The first model assumes constant vibration modes for each component, and will be referred to as the Constant-Vibration-Mode model (Constant-VM) and is closely related to Beer's law⁹⁷.

The second model assumes vibration modes which depend linearly on the mass ratios of all the electrolyte components, while at the same time assumes there are enough wavenumbers for which vibration modes are constant to just use these regions during prediction. This model will be referred to as the Linear-Vibration-Modes model (Linear-VM).

This section is quite detailed, trying to precisely describe the models implemented in the code. Most readers mainly interested in using the software or even adapting it to different electrolyte components should skip to Section 4.3 for an evaluation of the performance of the models, and use the current and next subsections as a reference if code modifications are needed, or if they want to propose an alternative model.

To directly apply Beer's law, one must introduce units of mass per total volume, but if the mass-over-volume values for all components are known, then we can compute the mass ratios. If the mass of each component over the total volume of the sample are gathered into a vector of concentrations c , and every absorbance for all measured wavenumbers are gathered into a vector s , then, Beer's law implies that absorbance spectrum s is a linear function of concentration vector c and this relationship can be represented by a matrix multiplication $s = A \cdot c$. In this case, A is a matrix and it is constant over the dataset, corresponding to an absorbance spectrum for each component. These absorbance spectra are called the **fundamental** spectra here. This linear relationship is sometimes referred to as Beer's Law⁹⁷.

However, we wish to go in the opposite direction. Starting with absorbance spectra, we want to predict the mass ratios. Therefore, we define the Constant-VM model as:

$$c = X \cdot s$$

$$m' = \frac{1}{\sum_{i=1}^N c_i} c$$

$$s' = A \cdot c$$

Where X and A are the tunable parameters of the model and are constant matrices. m' is the predicted vector of mass ratios, and s' represents a calculated spectrum based on the predicted mass ratios that can be compared to experiment. s' is called the **reconstructed** spectrum here. In the cases where Beer's law perfectly holds, then the whole dataset can always be perfectly described by a suitable choice of X and A . Note that the code introduces slight tweaks, but they would distract from the point of the article. See the code https://github.com/Samuel-Buteau/Electrolyte_Analysis_FTIR for details.

The Constant-VM model works well and is simplest to implement and understand. However, the dataset is such that it simply cannot be described with constant X, A . Therefore, we consider a simple generalization where A depends on the computed mass ratios m' linearly. This is called the Linear-VM model. Whereas A used to be a constant matrix, indexed by electrolyte component and wavenumber such that A_{ij} would correspond to wavenumber i and electrolyte component j , we replace it by the product of the mass ratios with a 3-dimensional tensor \mathbf{A}_{ijk} . More concretely, we replace the constant A_{ij} by

$$A_{ij} = \sum_{k=1}^N m'_k \mathbf{A}_{ijk}$$

Or in other words, the matrix to convert concentrations into reconstructed spectra is now a linear combination of matrices with the coefficients being the predicted mass ratios.

Now, based on this model, it is not immediately obvious that a constant matrix X can convert from spectra to concentrations. In general, the prediction would need to use non-linearities. However, if we assume that a sufficiently large wavenumber region has constant vibration modes (i.e. would be well fitted with the Constant-VM model), then a constant X could simply use these regions. To see that this is possible, consider that the number of electrolyte components (i.e. 5) is much smaller than the number of measured wavenumbers (above 1000). In our numerical experiments, we considered more complicated functions to go from s to c , but there was no benefit, so we focus on the Linear-VM model and the Constant-VM model, recommending the Linear-VM model. The fact that there was no benefit to adding non-linearities to the prediction is evidence in favor of the existence of enough linear regions to get good predictions.

4.2.3 Measuring performance and penalizing bad model properties.

For the Constant-VM model, given a dataset containing some samples with known mass ratios and some without known mass ratios, the parameters A, X are optimized to minimize various penalties using the ADAM optimizer²².

The penalties to minimize are:

1. The mean squared error between actual mass ratios and predicted mass ratios. (If the samples have known mass ratios).
2. The mean squared error between the original spectra and the reconstructed spectra.
3. Any negative values for the predicted mass ratios are penalized. (In reality, the model would always clamp any negative mass ratio to 0, but this clamping leads to the optimization getting stuck without the penalty on negative values.)
4. We maximize the sum of squares of elements of c while minimizing the sum of squares of elements of X , because this encourages X to ignore the noise in the calibration dataset's absorbance spectra. The details follow. As described in Section 1.2.5, X sends a basis of the space of spectra to the space of concentrations. Assuming no noise in the dataset, and considering the case where Beer's law is valid, we can see that the dataset itself will be contained within a subspace of very small dimension (e.g. 5 out of thousands). As such, this dataset can only constrain the action of X on a basis of that subspace (e.g. 5 independent vectors). For simplicity, consider an orthonormal basis of the smaller subspace which is extended into an orthonormal basis of the full input space. Then, for all these additional basis vectors, the output of X is not determined by the dataset. Ideally, we would like all these basis vectors to be sent to the null vector in the concentration space since this would make X robust to noise (which can always be decomposed into the additional basis vectors).

But this need not happen. In fact, it would be possible for the inverse situation to occur where all the basis vectors corresponding to the dataset without noise would be sent to 0 by X , and only the noise on each element of the dataset would have a non-zero contribution in the multiplication by X . (this can be accomplished since noise on different elements of the dataset is likely to span a large enough subspace such that the whole noise dataset underdetermines X). However, since for a given spectrum, the noise has a much smaller norm than the actual signal, in order to produce the right mass ratios (with comparable norm in the concentration), X would need to itself have a large norm. All things being equal, minimizing the norm of X should force it to use the signal rather than the noise.

For the Linear-VM model, we additionally penalize:

1. Any superfluous dependence of A on the predicted mass ratios. Concretely, if the 3-dimensional tensor \mathbf{A}_{ijk} is as discussed previously, then we want \mathbf{A}_{ijk} to not depend strongly on the index k .
2. The second derivatives of \mathbf{A} and X with respect to wavenumbers should be small, since all the FTIR spectra considered are smooth with respect to wavenumbers. Concretely, if the 3-dimensional tensor \mathbf{A}_{ijk} is as discussed previously, we want the absolute value of $\mathbf{A}_{ijk} - 2\mathbf{A}_{i-1,jk} + \mathbf{A}_{i-2,jk}$ to be small, where i is the wavenumber index. Similarly, if X_{ji} is as discussed previously, then we want the absolute value of $X_{ji} - 2X_{j,i-1} + X_{j,i-2}$ to be small.

The complete penalty is simply a linear combination of all these penalties, and the precise code defining this is reproduced in the next section.

In both of these cases, two types of data can be used for training the model.

1. Supervised data, where the precise mass ratios are known, will be used to tune both the mass ratio predictions and the spectrum reconstruction.
2. Unsupervised data, where the precise mass ratios are not known (in our case this is a much greater dataset), will be used to tune only the spectrum reconstruction. However, since the reconstruction is done by first predicting mass ratios, it will still be useful for improving the quality of the mass ratio predictions.

Finally, since the equation for the reconstructed spectrum does not change as long as $\mathbf{A}_{ijk} + \mathbf{A}_{ikj}$ doesn't change. This equation can be rewritten as follows.

$$s'_i = \frac{1}{\sum_{k=1}^N c_k} \sum_{k=1, j=1}^N c_k \mathbf{A}_{ijk} c_j$$

Without loss of generality, we set all $\mathbf{A}_{ijk} = 0$ for $k > j$.

4.2.4 All equations for the Linear-VM model

In case more details about the penalties are useful, the equations used are given below. However, it is strongly advised to skip this section if such details are not wanted. First, the function relu is defined in Section 1.2.5 as $\text{relu}(x) = x$ if $x > 0$ and $\text{relu}(x) = 0$ if $x \leq 0$.

The equations relating spectra to mass ratios are:

$$c = X \cdot s$$

$$\bar{c} = \text{relu}(c)$$

$$m' = \frac{1}{\varepsilon + \sum_{i=1}^N \bar{c}_i} \bar{c}$$

Here, ε is a very small number added to prevent dividing by 0.

Then the equations for the reconstruction of the spectrum is:

$$s'_i = \frac{1}{\varepsilon + \sum_{k=1}^N \bar{c}_k} \sum_{k=1, j=1}^N \bar{c}_k \mathbf{A}_{ijk} \bar{c}_j$$

Then, the various penalties are:

$$L_{\text{reconstruction}} = \sum_{i=1}^M (s_i - s'_i)^2$$

$$L_{\text{prediction}} = z \sum_{j=1}^N (m_j - m'_j)^2$$

Here, z is 0 if the data is unsupervised and 1 if supervised.

$$L_{\text{positivity}} = \sum_{j=1}^N \text{relu}(-c_j)$$

$$L_{\text{normalization}} = \left(1 - \frac{1}{N^2 M} \sum_{i,j,k} \mathbf{A}_{ijk}^2 \right)^2$$

$$L_{\text{smallx}} = \frac{\sqrt{\frac{1}{NM} \sum_{i,j,k} \mathbf{X}_{ji}^2}}{\varepsilon + \sum_{i=1}^N \bar{c}_i}$$

Let $\tilde{\mathbf{A}}_{ijk} = \mathbf{A}_{ijk} + \mathbf{A}_{ikj}$, then

$$L_{\text{linear}} = \sum_{i,j,k} \left(\tilde{\mathbf{A}}_{ijk} - \frac{1}{N} \sum_k \tilde{\mathbf{A}}_{ijk} \right)^2$$

$$L_{\text{smooth}} = \sum_{i,j,k} (A_{ijk} - 2A_{i-1,jk} + A_{i-2,jk})^2 + \sum_{i,j} (X_{ji} - 2X_{j,i-1} + X_{j,i-2})^2$$

The total penalty is simply the weighted sum

$$L = L_{\text{reconstruction}} + \alpha_1 L_{\text{prediction}} + \alpha_2 L_{\text{positivity}} + \alpha_3 L_{\text{normalization}} + \alpha_4 L_{\text{smallx}} + \alpha_5 L_{\text{linear}} + \alpha_5 L_{\text{smooth}}$$

With this, most of the details of the code have been formulated precisely. Now, we discuss the structure of the code.

4.2.5 Code Structure and Content

The code was written to have a simple way to run the model on some measurements and may be accessed at https://github.com/Samuel-Buteau/Electrolyte_Analysis_FTIR.

There are many possible ways to call the code:

1. After having put some measurement files in a directory, the model can be called by specifying this directory. It will return an excel file with the predicted mass ratios, as well as a directory with a graph of the reconstructed spectra together with the original data for each input file (the extension of the files will be changed from .asp to .png). This is the **run_on_directory** option. For convenience, an excel sheet with the numerical data for measured and reconstructed spectra is outputted as well so the user can make their own graphs (e.g. for publication).
2. The training can be run on the calibration dataset, and thus the model can be updated. This is the **train_on_all_data** option.

3. Cross-validation studies of the model can be run to evaluate it on the calibration dataset. This is the **cross_validation** option.
4. The figures in this paper can be reproduced. This is the **paper_figures** option.
5. There is a functionality to create a dataset, allowing the extension of the dataset, but it might require some modification to adapt to a different lab's workflow. This is the **create_dataset** option.

4.2.6 Addendum: Spectra Measured on Other Apparatus

Note that it is possible to take a spectrum measured at different wavenumbers than those used in the dataset, and through interpolation, estimate what would have been measured at the wavenumbers required by the system.

This was not done in the corresponding article to Chapter 4, but the predictions do not depend on the scale of the FTIR spectrum, and the reconstruction of samples measured on different systems were anecdotally adequate. Since the FTIR spectra live in a very large vector space (dimension greater than 1000), the fact that reconstruction was adequate is strong evidence that the predictions can be transferred adequately from one system to another simply by first interpolating the measurements to the wavenumbers used in the calibration dataset.

Indeed, one can be confident that the system can still determine when its “axioms” are respected or not. *Accurate reconstructions should be a strong evidence of accurate predictions.* Yet, without a careful empirical study, there is no way to know the degradation in the typical reconstruction accuracy that this transfer (i.e. using interpolation to transfer spectra to the wavenumbers used during calibration) may cause. In other words, if reconstruction works, prediction works, but reconstruction might work less often when using some different measurement settings. This latter

limitation could be overcome with more data, but Chapter 4 focusses on the cases where data is not plentiful.

4.2.7 Future Work: Fine-tuning

After reading Chapter 3, the perceptive reader may have gotten the idea that the linear prediction of the electrolyte concentrations is merely an *inverse model*, and its output may be used as initial guesses for an optimizer such as ADAM to minimize the reconstruction error. This indeed could be done with much of the same code, except that the *reconstruction model*'s parameters (namely \mathbf{A}_{ijk}) must be kept fixed during this fine-tuning. Indeed, the *equations for the impedance* of a given Equivalent Circuit model are fixed, and only the Equivalent Circuit *parameters* change.

On the supervised training set, the reconstruction model has already been optimized together with the linear inverse model, so it should be expected that the “initial guesses” would be nearly optimal. However, for new samples which are unlike those in the training set, it is possible that the linear inverse model produces an initial guess which is significantly suboptimal. Indeed, there is a whole subfield of machine learning exploring the deliberate construction of so called *adversarial examples*⁹⁸ which make a given neural network produce a spurious prediction by carefully altering a sample in the dataset. In general, it is hard to characterize how a given component of a neural network might fail, but it is possible that the linear inverse model fails to produce a good initial guess on some atypical samples. For those samples, the question remains of whether a different choice of concentrations would produce a significantly better reconstruction or not for the fixed reconstruction model used. This could be answered directly by the fine-tuning approach.

4.3 Results and Discussion

In order to assess the performance of the model, a percentage of the dataset is held out so that the model does not see this data during training. Then, the trained model is fixed, and applied to the held-out part of the dataset. Thus, we can obtain an approximation of how the model performs on data which was not used during calibration. This can be repeated many times such that all the data is held out at least once, and then the results can be plotted and compiled. This is called Cross-Validation. To simplify the computational demands, we consider held out datasets of 10%, 20%, ... up to 90% successively, and repeat the procedure 8 times or more for each setting, only showing the results on the held-out sets and compiling the various held-out sets to cover the whole dataset. To be clear, there is no selection of the best outcome of this procedure, and during each isolated experiment, the held-out set is only used for testing. Furthermore, when the predictions are shown visually, all the errors for all the held-out sets are shown, thus showing a worst case scatter, and when the prediction errors are averaged, then the average is over all such experiments.

Figure 4.1 shows a quantitative overview comparison of the cross-validation performance of the two models. The performance is shown as a function of the percentage of data which was held-out of the calibration and only used for testing.

On the left we have the absolute value of the difference between measured spectra and the reconstruction of the model. This error is averaged over all wavenumbers and divided by the average absorbance of the measured spectrum. An error of 100% corresponds to as much error as signal. The two models are shown with the same axes, which shows that b) Linear-VM significantly improves over a) Constant-VM.

On the right, we have the absolute difference between actual mass ratio and the prediction of the model, given as a percent of the largest mass ratio for that component within the calibration set (i.e. 0.22 for LiPF₆, 0.5 for EC, and 1 for the linear carbonates EMC, DMC, and DEC). Within the randomness of cross-validation, there is no significant difference between the two models here.

We can also see the error bars, which represent the standard deviation of the average error over the various trials in the cross validation (for instance, if 8 times the dataset has been split into training and testing subsets and the training algorithm has converged to a solution, then the standard deviation of the total error across these 8 trials is given).

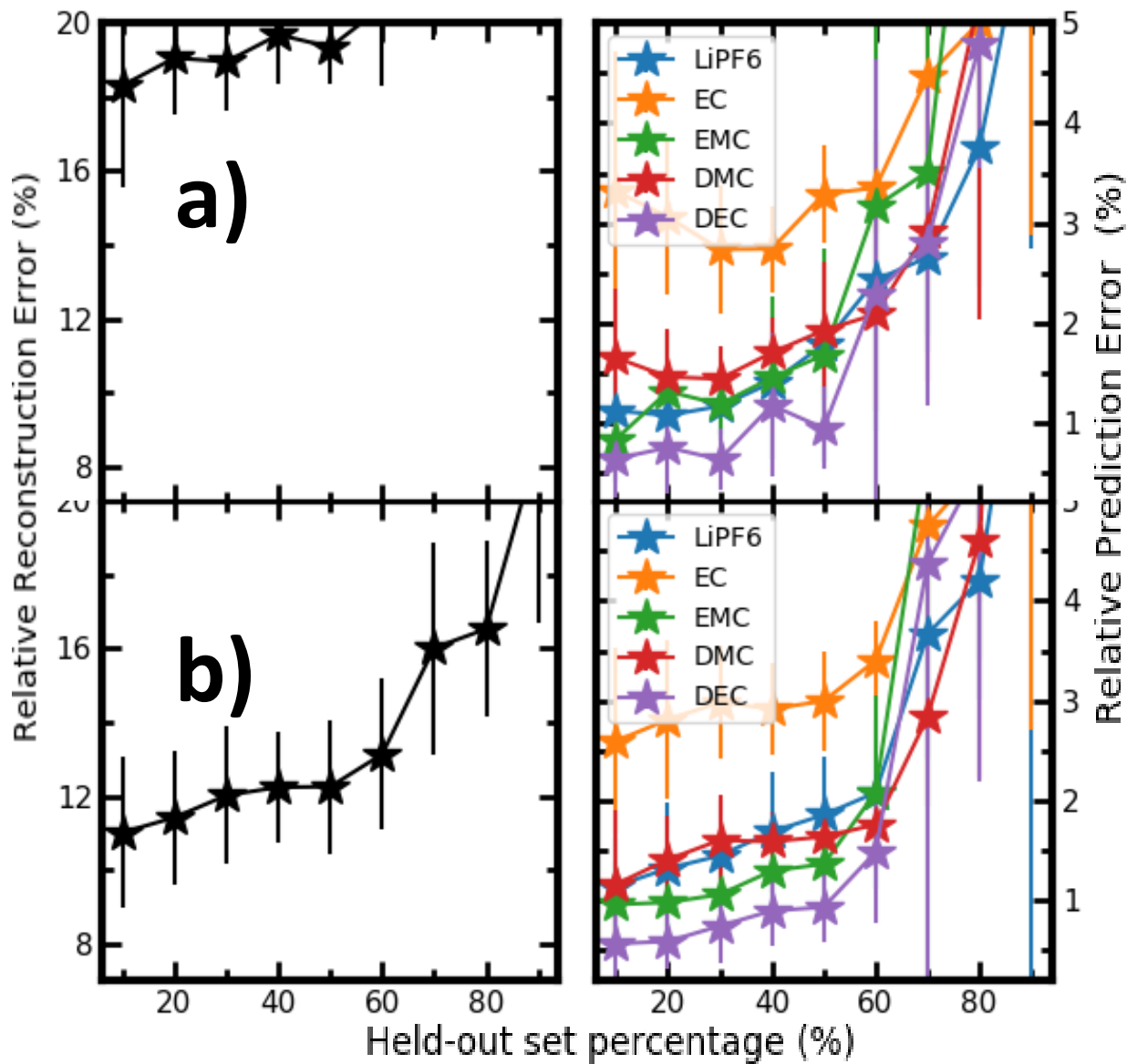


Figure 4.1 Comparison of the cross-validation performance of the two models (overview). Each row corresponds to a different model a) Constant-VM and b) Linear-VM. The performance is shown as a function of the percentage of data which was held-out of the calibration and only used for testing. On the left is shown the reconstruction error between the measured spectrum and the predicted spectrum from the model. Linear-VM is clearly better. On the right is shown the prediction error between the prepared mass ratios and the predicted mass ratios.

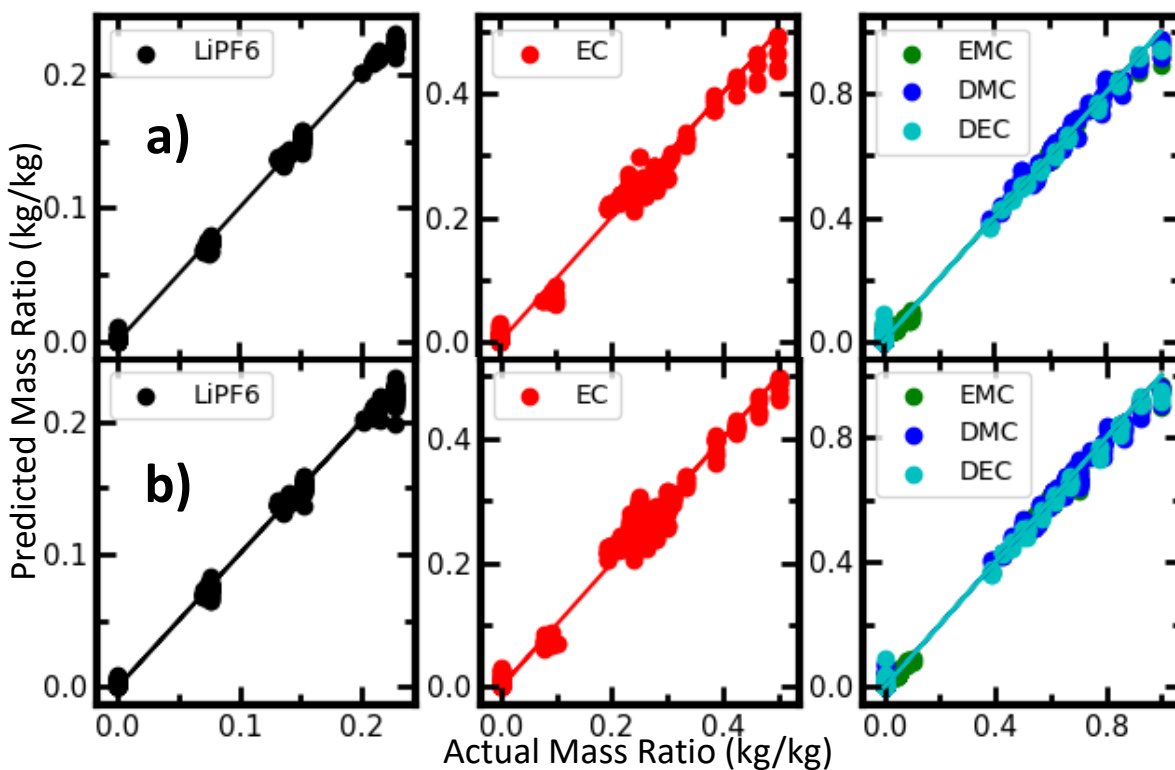


Figure 4.2 Comparison of the cross-validation predictions of the two models for the mass ratios with 30% of the data which was held-out of the calibration and only used for testing. Each row corresponds to a different model a) Constant-VM and b) Linear-VM. For each FTIR spectrum in the test set, the predicted mass ratios are plotted vs the actual known mass ratios, as discrete points, whereas the theoretical "perfect prediction line" which simply shows where perfect predictions would be on the graphs is shown as a full line.

Figure 4.2 shows a visual overview comparison of the prediction performance of the two models. Every single error in prediction over 10 cross-validation experiments is shown, and therefore, the visual scatter of the predictions represents a worst case estimate for this type of data when using 70% of the data was used for training and 30% of the data was used for testing.

Figure 4.3 shows an evaluation of the Linear-VM model's cross-validation reconstruction performance with 30% of the data which was held-out of the calibration and only used for testing. Figure 4.3 should be studied in conjunction with Figure 4.1 as Figure 4.3 gives a qualitative reference to the quantitative study presented in Figure 4.1. In order to compare between the Linear-VM and the Constant-VM model, we look at averages over the held-out data to get a single average absorbance spectrum and a single average absolute error spectrum (the average of absolute error for each wavenumber across all held-out data is plotted versus wavenumber). In order to show that different wavenumber regions have different quality of reconstruction, the wavenumber range has been split into 3 regions (low, medium, high), and plotted on separate rows. As can be seen, most of the reconstruction error occurs in the 800-900 cm^{-1} range, while the middle range (second row) has a decent reconstruction, and the high wavenumber range is quite good. It is instructive to correlate the regions of high error with known vibrational modes. For instance, around 780, the cause is CO_3 non-planar rock on the ethylene carbonate (EC) molecule, around 840, the cause is the $LiPF_6$ t_{1u} mode. Also, around 1150, the cause is CO_2 symmetric stretch on EC, while around 1250, the cause is CO_2 symmetric stretch on linear carbonates. For more details on the causes, see our previous work⁹³.

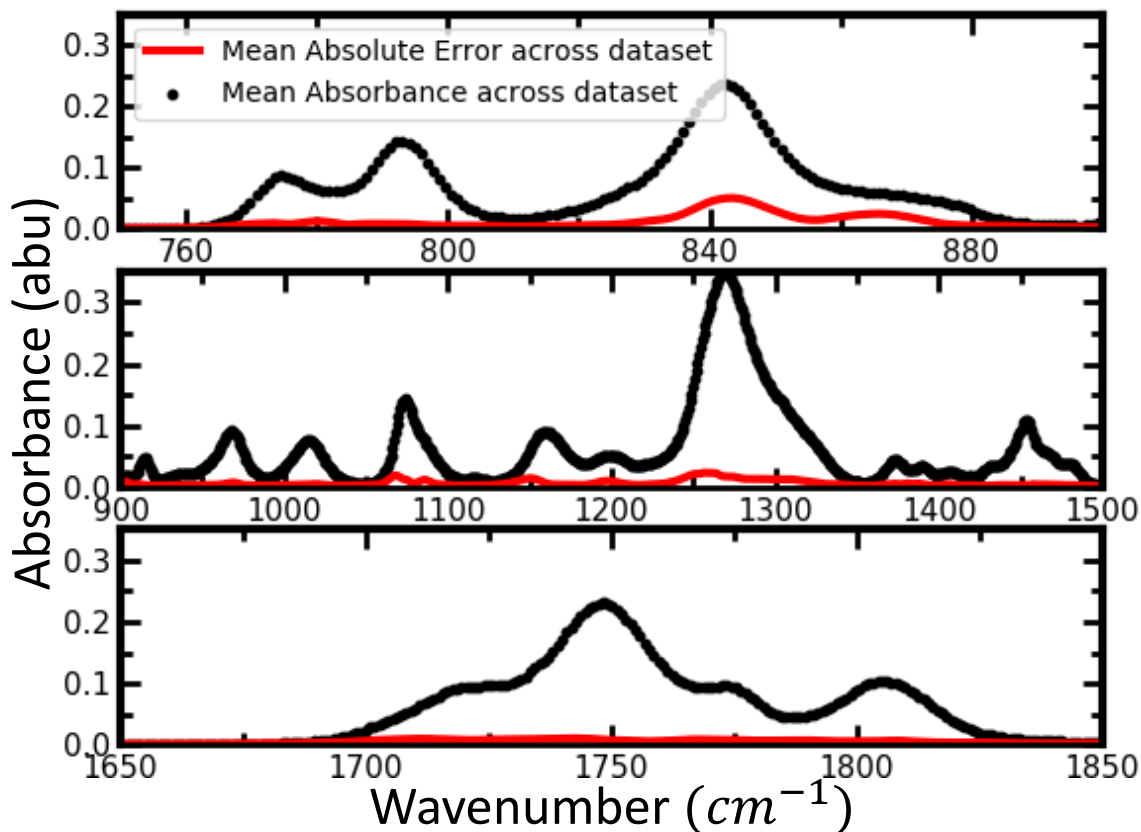


Figure 4.3 Evaluation of the Linear-VM model's cross-validation reconstruction performance with 30% of the data which was held-out of the calibration and only used for testing. In black, the mean absorbance spectrum over the whole dataset is shown, while in red, the mean absolute error of reconstruction (measured in the same units) across the held-out data is shown.

Figure 4.4 shows an evaluation of the Constant-VM model's cross-validation reconstruction performance with 30% of the data which was held-out of the calibration and only used for testing. Figure 4.4 should be studied in conjunction with Figure 4.1 as Figure 4.4 gives a qualitative reference to the quantitative study presented in Figure 4.1. See Figure 4.3 for the details of how this plot was obtained. The error regions from Figure 4.3 are amplified here, but in addition the

high wavenumber region has substantial error. The cause for the high region is mostly the C=O stretch. For more details on the causes, see our previous work.

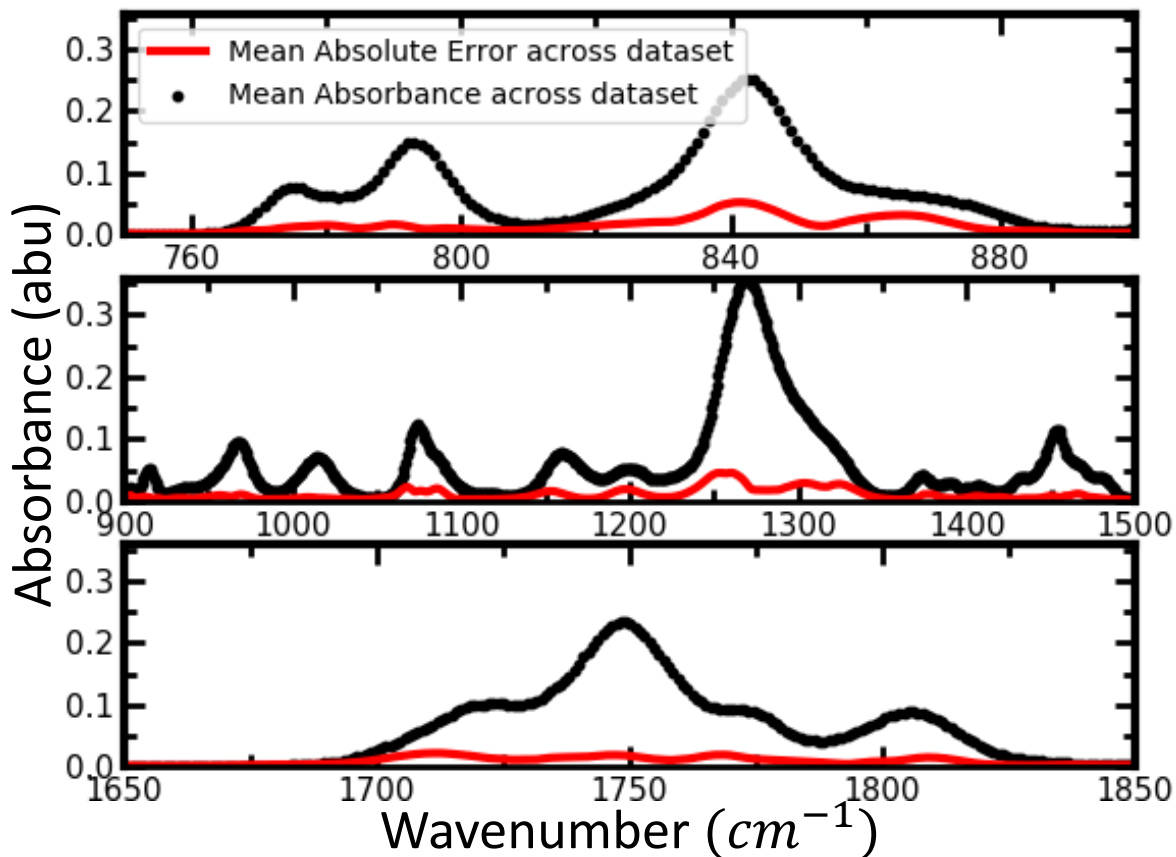


Figure 4.4 Evaluation of the Constant-VM model's cross-validation reconstruction performance with 30% of the data which was held-out of the calibration and only used for testing. In black, the mean absorbance spectrum over the whole dataset is shown, while in red, the mean absolute error of reconstruction (measured in the same units) across the held-out data is shown.

Figure 4.5 shows an evaluation of the physical plausibility of the vibration modes of the Linear-VM model. Each component of the electrolyte produces a partial spectrum, revealing the vibrational modes and these partial spectra produced by the model are shown in dashed colored lines. After calibration over the full dataset (i.e. 0% held-out), the model is used to reconstruct the spectra of the calibration set. By looking at the contribution from each electrolyte component, we can see if the vibration modes make sense. The total reconstruction tracks the measured spectrum as expected. Yet, the individual contributions for each component are smooth and the DEC vibration modes appear more or less constant over most of the spectrum. The components behavior here is representative across the calibration set. Note that despite providing plausible components, there is no guarantee that these exactly reflect the individual vibrational modes, but rather they are a sign of the model's health.

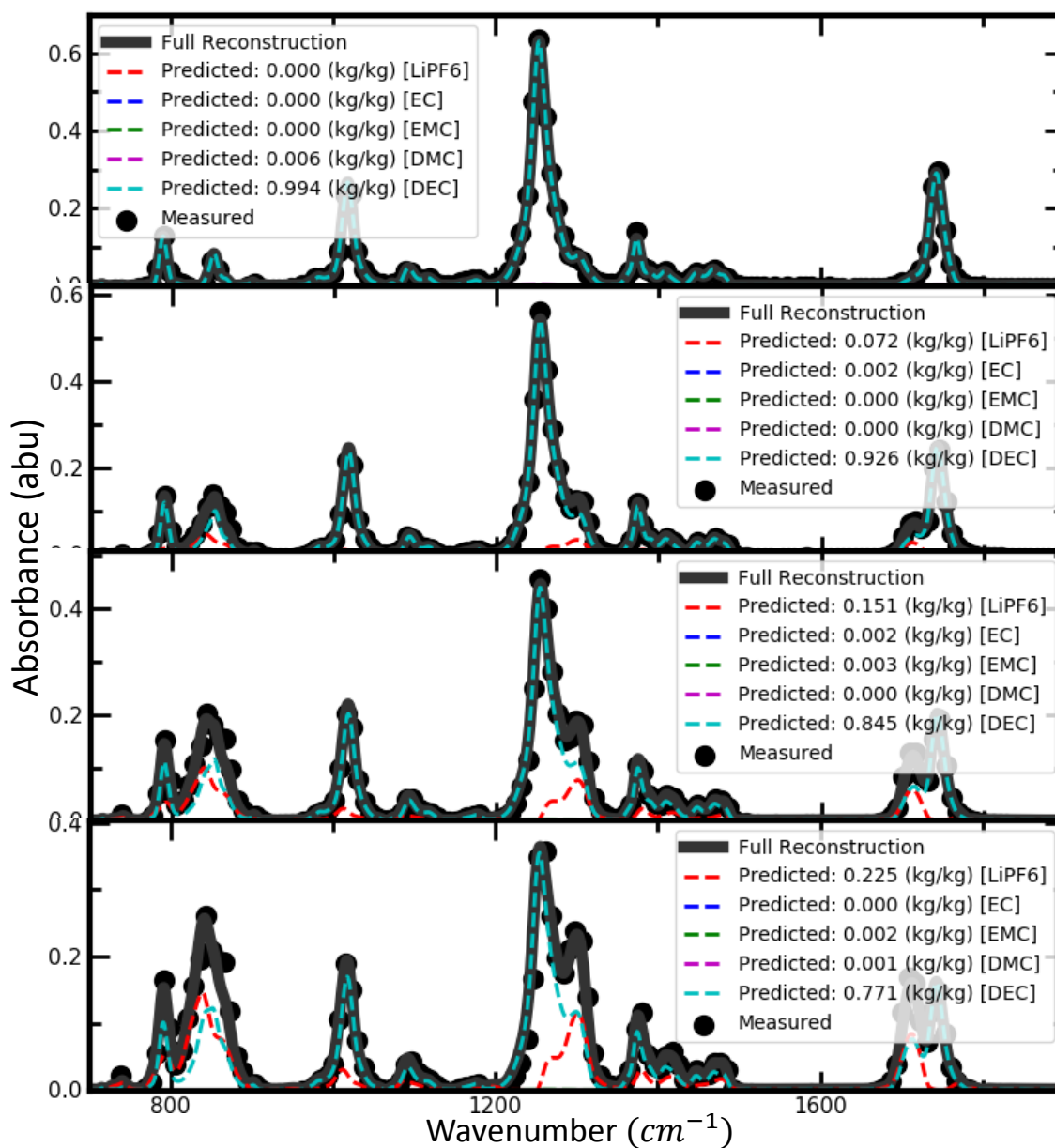


Figure 4.5 Evaluation of the physical plausibility of the vibration modes of the Linear-VM model.

Each component of the electrolyte produces a partial spectrum, revealing the “vibrational modes” and these partial spectra produced by the model are shown in dashed colored lines. The four panels show mixtures of DEC and LiPF6, and the predicted amounts are accurate up to 0.01kg/kg. The measured data has been down-sampled for visibility.

In order to assess how the trained model varies across various runs of the cross validation, Figure 4.6 and Figure 4.7 show the learned parameters of X and A . Indeed, both figures show the mean value of each parameters as well as the standard deviation as error bars. Figure 4.6 shows that the parameters of X do not strongly vary from trial to trial when 30% of the data is held-out, and that X does indeed vary smoothly with respect to wavenumber.

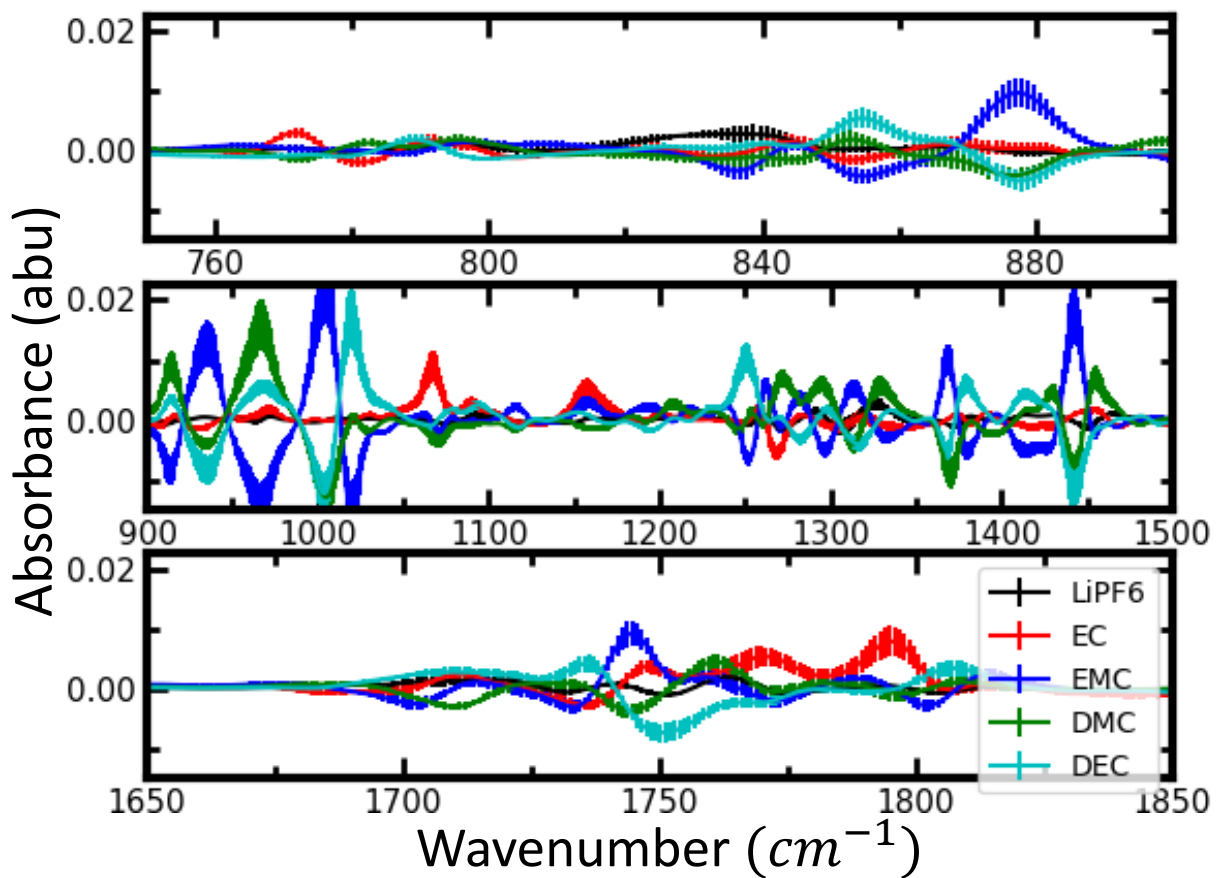


Figure 4.6 Parameters of X during the cross-validation with 30% of the data which was held-out of the calibration and only used for testing. Each color corresponds to the slice through X where the concentration index is set to a given component. The solid line corresponds to the average across different trials while the error bars correspond to the standard deviation.

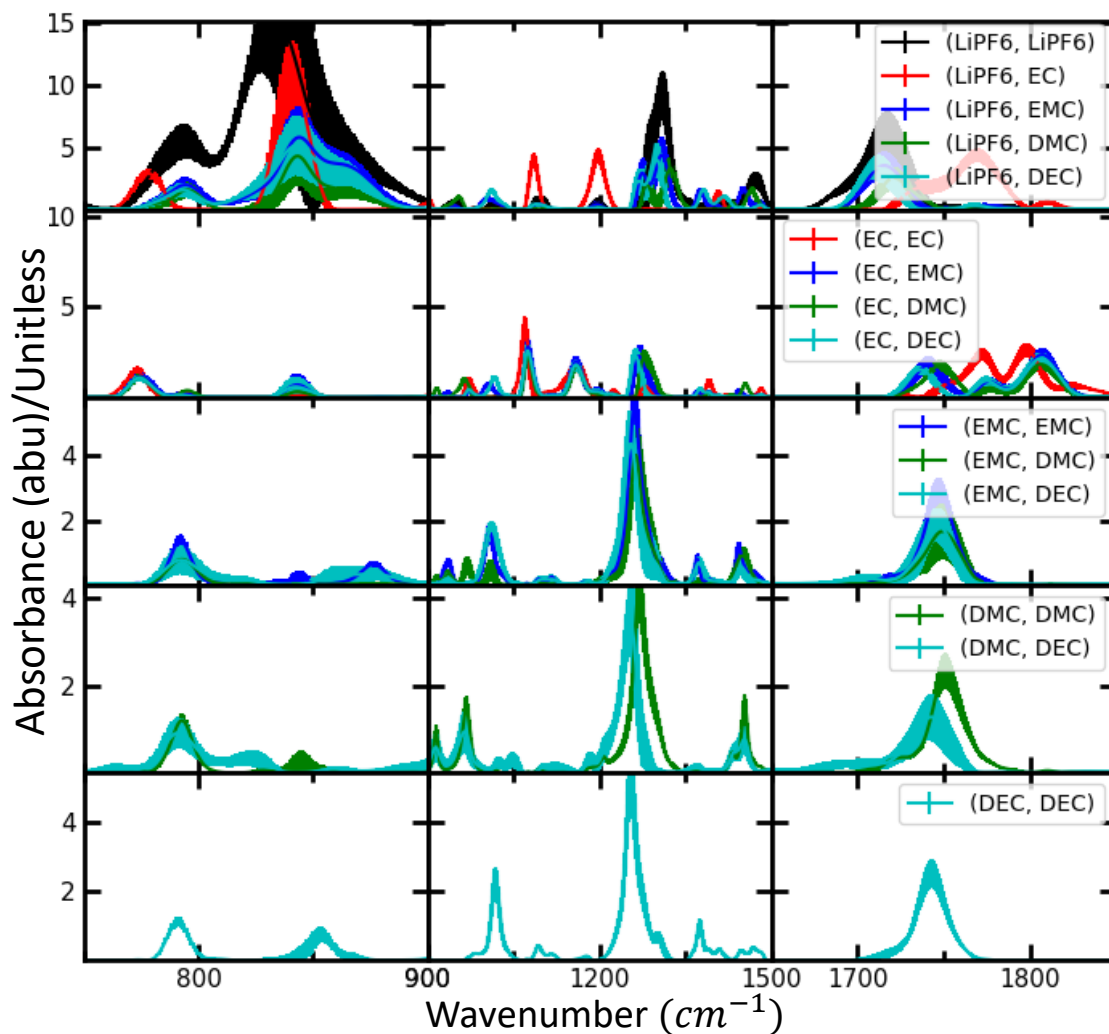


Figure 4.7 Parameters of \mathbf{A} during the cross-validation with 30% of the data which was held-out of the calibration and only used for testing. Each color corresponds to the slice through \mathbf{A} where the concentration indices are set to the given components. The solid line corresponds to the average across different trials while the error bars correspond to the standard deviation.

Figure 4.7 shows that the parameters of \mathbf{A} do vary more from run to run. More specifically, the variation is mostly localized in ranges of wavenumbers where the reconstruction error is larger.

Since \mathbf{A} has been chosen with many elements 0, we can visualize each distinct pair of indices j, k as a function of wavenumber.

In summary:

- In terms of predicting the mass ratios the two proposed models are virtually identical in their performance. (As a percentage of the maximum value within the dataset, the mass ratio prediction errors for LiPF_6 and the linear carbonates is around 1 percent, while the ethylene carbonate error is between 2 and 3 percent.) Figure 4.1 shows how the prediction performance depends on held-out percentage, while Figure 4.2 shows the exact predictions for 30% held-out sets.
- However, in terms of the reconstruction of the absorbance spectra, Linear-VM is clearly superior, with total absolute error around 10 percent of the total measured absorbance, which can be compared to Constant-VM with total absolute error around 20 percent of the total measured absorbance (Figure 4.3 and Figure 4.4 suggest that the main difference occurs in the high wavenumber region, but the error is worse across all wavenumbers for Constant-VM). Note that even the predicted spectra from the Linear-VM do not agree perfectly with the measurement, but they still correspond to a useful reconstruction allowing the user to identify samples far outside of the LiPF_6 , EC, EMC, DMC, DEC system. It seems plausible that a more sophisticated model could improve the reconstruction further, but in cases of complicated models, false positives become a worry (i.e. samples outside the space of the components in the training set which can nevertheless be well reconstructed by using extreme values of the concentrations). Figure 4.1 shows the quantitative picture, as a function of held-out percentage, while Figure 4.3 and Figure 4.4

show a qualitative analysis of typical reconstructions from the Linear-VM and Constant-VM models for the 30% held-out sets.

- The reconstruction can even be decomposed by electrolyte components to give some idea of the underlying features of the spectra (i.e. the vibrational modes), with plausible shapes. This might be worth future work in cases where the vibrational modes themselves are of interest. As the model is, it is hard to guarantee that the vibrational modes extractable from calibration on a limited dataset are unique. Figure 4.5 shows a sequence of samples starting from pure DEC, with increasing amounts of LiPF₆, where the partial spectra due to each component are shown together with the reconstruction. As can be seen, the vibration modes of DEC and LiPF₆ are recognizable throughout, while changing slightly in shape in addition to being scaled by the predicted concentrations. Furthermore, the individual contributions to the absorbance spectrum from each molecule type are smooth with respect to wavenumber (note that without the derivative penalty discussed in Methods section, we can instead get noisy contributions from each components that perfectly cancel out to give a smooth reconstruction).

4.4 Conclusions

In this Chapter we showed how the concentration of electrolyte components in lithium-ion cells can be determined using Fourier transform infrared spectroscopy, Beer's law, and machine learning. A physically grounded model was used. We have also open-sourced a carefully prepared dataset and the code to replicate and extend to different electrolyte mixtures. With this new model, a prediction error of around 1-2% for the LiPF₆-to-total mass ratio, as well as all the linear carbonates, and around 2-3% for ethylene carbonate is achieved. Furthermore, this model allows

for a useful reconstruction of the FTIR spectrum of an unknown sample. This allows the user to detect samples significantly different from those in the dataset (e.g. due to a bad measurement, to a significant amount of a different molecule, or to a significant change in apparatus). Furthermore, the model is data efficient such that a model for mixtures of 5 components can be calibrated well with less than 50 carefully prepared samples. Therefore, it should be possible to easily extend this work to other systems.

This work refines and generalizes our previous work⁹³ and improves the physical underpinnings of the model. The composition of unknown electrolyte samples, with a specified set of components, can be well determined using inexpensive and rapid measurements with attenuated total reflectance fourier transform infrared spectroscopy.

The code is available at https://github.com/Samuel-Buteau/Electrolyte_Analysis_FTIR, with all the documentation in the README.md file.

4.5 Acknowledgements

This work was supported financially by the Natural Sciences and Engineering Research Council of Canada (NSERC), and Tesla Motors. Sam Buteau acknowledges scholarship support from NSERC. Furthermore, Sam Buteau would like to thank Prof. Yoshua Bengio and the Mila community for useful discussion groups and seminars.

Chapter 5 A Database for Lithium-ion Data

Previous chapters have discussed big ideas and theoretical insights to ensure the robustness of machine learning models in the context of lithium-ion cell research, yet the underlying *data processing system* may well determine the success or failure of projects based on such models.

Obviously, models developed based on data are not a closed system since their performance is directly influenced by the data they are fed, and how they are used. Similarly, these models may influence research decisions down the line, leading to different data being collected and the models being used differently.

To plan for success, it is helpful to look at the whole system (the lab) filled with researchers, equipment, computer programs, etc.. The lab produces various data, develops tools to analyse this data, and the whole process is very hard to characterize since research direction is influenced by an evolving understanding which in turn depends on the previous research as well as the inputs from other labs or industrial partners. Yet some features are simple to understand.

Just like a big machine, there can be friction, effort, and frustration associated with processing data, storing it, retrieving it, understanding it, developing tools for it, etc.. The ease with which these operations are performed, henceforth called *data processing efficiency*, is the subject of this chapter. **Unless explicitly noted, this Chapter is a documentation of the features already implemented and already in active use.**

5.1 Motivation for a Data Processing System

Now that the importance of data processing efficiency has been described generally, let us dive into the specifics of the lithium-ion research.

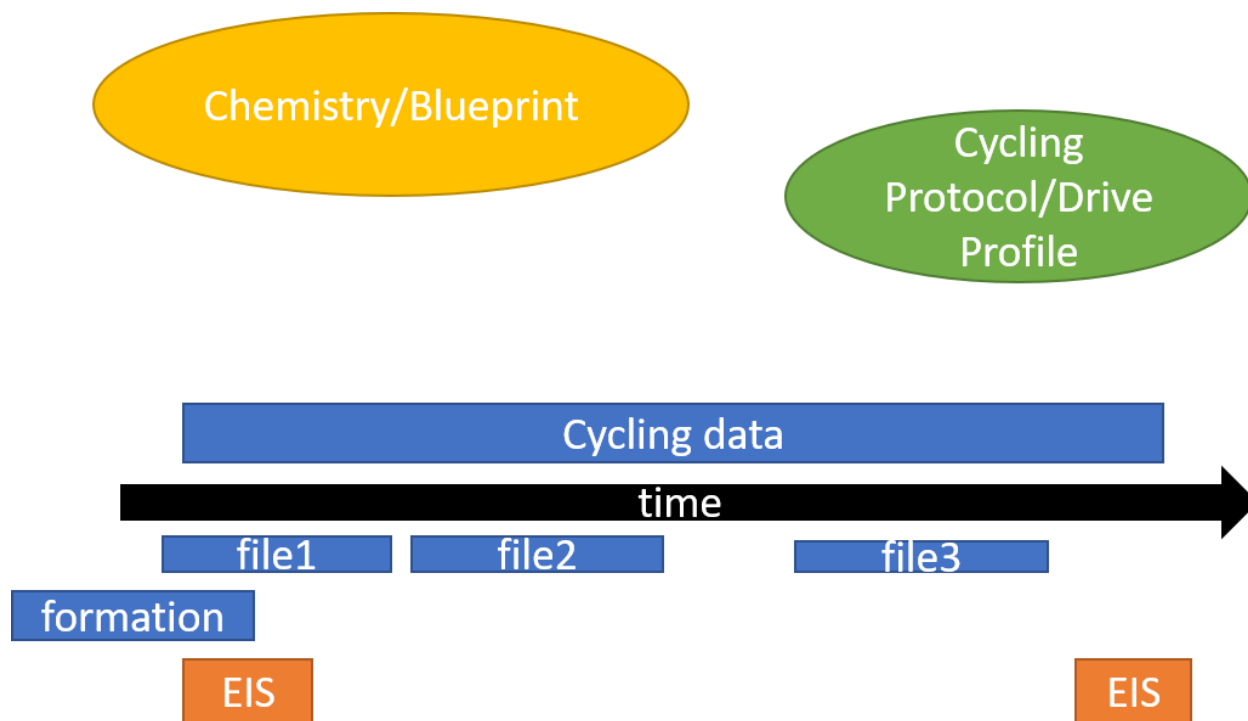


Figure 5.1 Schematic of the various modalities of battery data. The rounded bubbles contain some important *metadata* which have their own structure. The black arrow indicates the temporal relationship between various experimental observations. The cycling data is spread across three files in this case. There are some other types of observations, such as the formation data and the electrochemical impedance spectra taken at the beginning and end of cycling, respectively.

Figure 5.1 shows a schematic of some typical experimental observations for a single cell. First, there is the metadata associated with the cell itself (Chemistry/Blueprint, henceforth called *Cell Metadata*), including the geometry of the cell, the composition of the electrolyte, the electrodes, etc.. Second, there is the metadata of how the cell was cycled, typically allowing for each cycle to select a charge current, discharge current, limit upper and lower voltages, etc.. Third, the cycling data itself, which contains the precious information about the capacity of the cell, the impedance

of the cell in the form of polarization in the voltage curve between charge and discharge, and how these characteristics evolve through time (see Section 5.3.5 for a more in-depth discussion of the structure of cycling data). The cycling data may be split across multiple files and each file may encode the information in a different format. Indeed, even a single manufacturer (e.g. Neware) can produce the data in many different formats and variants, totalling more than 20 different commonly used variations. This in itself is a significant friction factor for the analysis of the data and in practice required substantial effort and iteration to manage.

There are typically also many different types of observations such as formation, impedance spectroscopy (see Chapter 2), but also some chemical composition measurements (see Chapter 4), some pressure measurements, and so on. **Though the data processing system already can recognize files of these various modalities and was designed to be easily extendable, it only fully supports cycling data as of the writing of this thesis.**

Unless explicitly noted, this Chapter is a documentation of the features already implemented and already in active use.

Other difficulties exist:

- Cycling experiments are ongoing for weeks, months, or years, with the experimental equipment updating the files once per day. Furthermore, the output of these experiments must be studied in real time in order to prioritize limited resources and guide future experiments.
- Given all the files produced by the various experiments, maintaining knowledge of what each file contains (i.e. which cell it measures, under what condition, at what point in the lifecycle of the cell, henceforth called *Valid File Metadata*) is quite challenging. This is a

major issue when the work of many students must be combined, especially once students have left the lab. At one level, this problem comes down to each student creating their own *ad-hoc* system to keep track of their experiments, but on a deeper level, it takes more information to situate an experiment in the complete context of tens of students across many years than it takes to situate an experiment in the limited context of a small study. Without special care, the incentives of individual students are pushing them to only keep track of the limited information relevant to their current project in their own format specific to that project. There is also a perverse correlation, which is obvious in the data the author studied, that the most prolific producers of experiments are typically the least compliant to any systematic way of entering the metadata. (It makes sense that larger projects benefit most from developing their own special-purpose formats). Without considering the human factor of using a given system, the “best” of designs will fail.

- When coming up with a common format which needs to encapsulate all the various activities in a lab studying lithium-ion batteries, it becomes clear that lithium-ion cells are complicated.
- For any software solution to the various issues of data processing, no matter how elegant the solution is, in order to be used, it must interface with the large unstructured legacy systems (i.e. handwritten notes or similar).
- Entering all the information manually would be prohibitively tedious and time-consuming, yet any automation scheme will run into the human factor: at any point, in any stage of the data processing, incorrect data exists and will continue to exist and be generated.

- The absence of a good solution to these problems not only plagues individual students in a given lab, but also forces different labs and companies to spend years “inventing the same wheel”.

Given these issues, creating large reliable datasets (on the scale of 10 000 to 100 000 cells) for the development of various models and analysis tools is not feasible without a suitable data processing system.

However, the emphasis in this chapter will be on how such a system can enhance the operation of a lab even without considering the downstream effects on model creation. In order to be useful at all, such a system must become an integral part of the lab’s operation. The curation and maintenance of very large-scale diverse datasets is a by-product here.

5.2 Requirements and Desired Properties of a Data

Processing System

Before diving into the implementation of the data processing system (henceforth called the *Universal Battery Database*), the various requirements must be discussed.

In overview,

- Information must be accessible by any lab member, even years after the experiment was completed, and hopefully also from other labs.
- Finding a given experiment among hundreds of thousands of experiments must be easy, without prior knowledge of what experiments do or do not exist.
- Tracking given experiments across years and comparing them should be easy.
- Visualizing the data should be automatic.

- Extracting different types of cycles (slow or fast discharge, different depths of discharge, etc.) should be integral and automatic.
- Producing and maintaining simple CSV files for further analysis should be automatic.
- Producing and maintaining datasets friendly for machine learning models should be automatic.
- Cells should be described systematically.
- Information should be curated automatically.
- Easy corrigibility at every stage through user interface.
- Flexibility to grow and maintainability.

Unless explicitly noted, this Chapter is a documentation of the features already implemented and already in active use.

5.3 Design of the System and Fundamentals of its Workings

5.3.1 Lithium-ion Cell Ontology (Structure of the Cell Metadata)

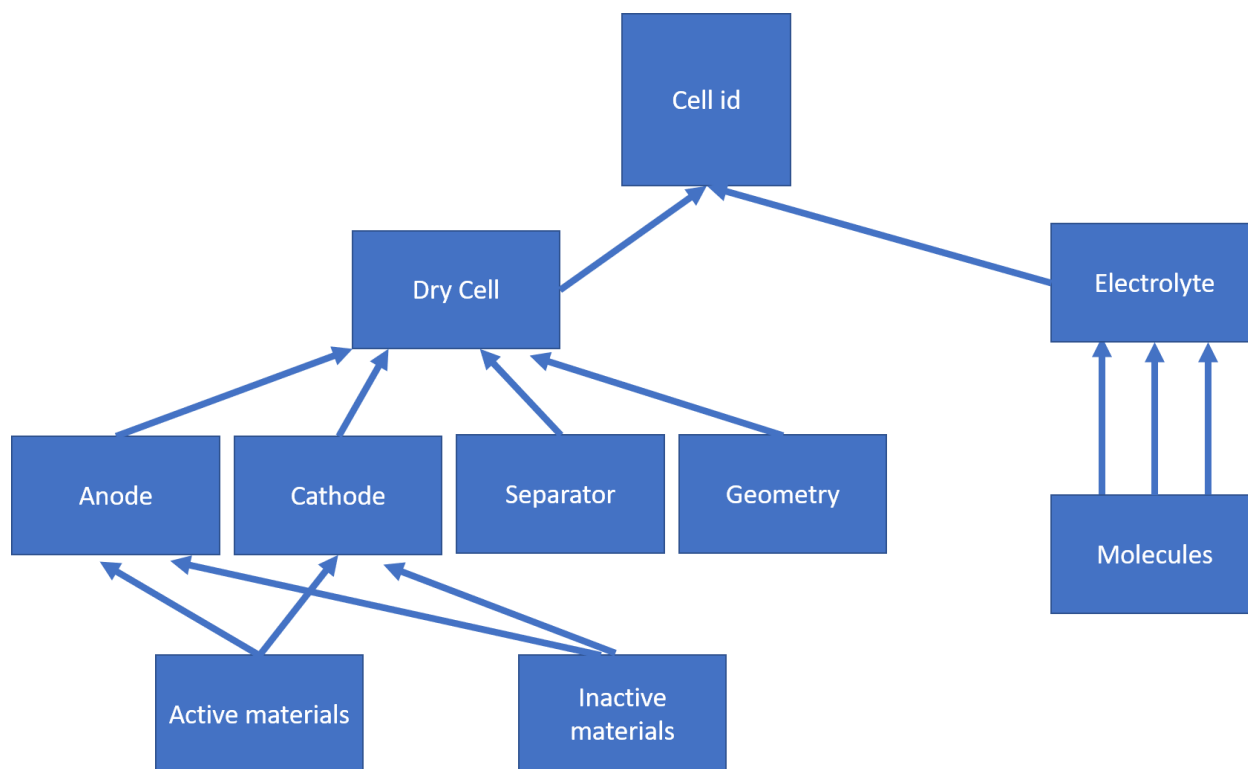


Figure 5.2 A hierarchical breakdown of a cell into components, which are themselves made of components. At the top level, a cell “has” a dry cell as well as an electrolyte. The electrolyte is itself a weighted combination of molecules, and the dry cell has a cathode, an anode, a separator, and some geometric information. Similarly, the anode and cathode are each a weighted combination of various active materials and inactive materials.

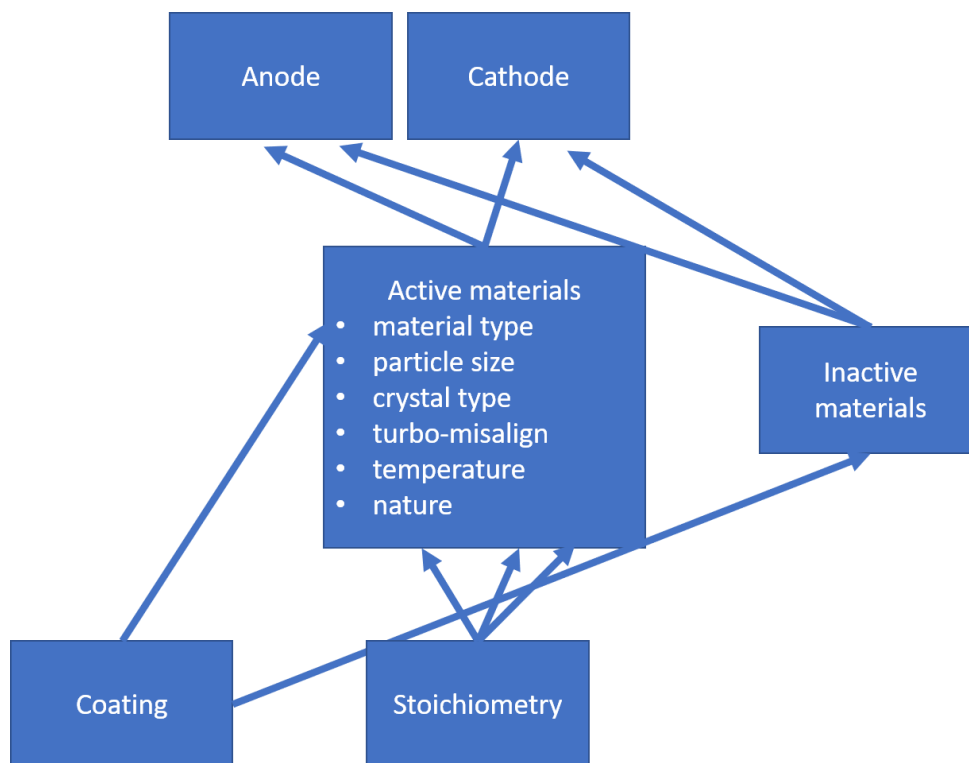


Figure 5.3 A more detailed breakdown of the components of the cell into subcomponents. As in Figure 5.2, the anode, cathode, and separator are each a weighted combination of various active materials and inactive materials. Each active material has various attributes such as material type, particle size, crystal type, etc.. Additionally, they can have stoichiometry information represented as a weighted combination of atoms, and an optional coating.

Figure 5.2 and Figure 5.3 illustrate the structured description of lithium-ion cells. Most important for searchability, the electrolyte is a weighted combination of molecules. Ideally, a simple weight ratio would be enough to describe all electrolytes in a standard way. Unfortunately, various conventions and ways of making electrolytes were used in the past. Typically, molecules are used either as *salts*, *solvents*, or *additives*. Furthermore,

- salt ratios should be given in molal units (mols of salt per kilogram of solvents), but are sometimes given in molar units (moles of salt per liter of solution)
- solvent ratios are given as weight ratios such that the sum of all solvents is 100 percent
- additive ratios are normally given as weight percent of the complete electrolyte.

Anodes, cathodes, and separators are always given as weight combinations such that the weight ratios of active materials sum to 100 percent and the inactive materials are given as weight percent of total material. Also, each molecule is unique, but the same molecule could be used as a salt in one context and an additive in another context (similarly with a solvent). Therefore, each molecule has a default type, but each electrolyte can override the type of its molecules.

Another important feature of this hierarchy is that each node has an “unstructured” field where information can be written as *text*. This is important since for every part of the hierarchy, there could be cells which do not “fit the mold”. For instance, “anode-free” cells do not really have active and inactive materials in their anode. Their anode is entered in the system as an anode with “anode-free” in the unstructured field, and an empty set for the combination of materials. Similarly, some cells are proprietary and so the details are not known. It is possible to distinguish different *proprietary* dry cells and electrolytes by simply filling the unstructured field with an appropriately unique name.

Finally, for each node in the hierarchy, we can have a “blueprint” or a specific “lot”. Concretely, imagine the recipe for an electrolyte (the blueprint), and different people may produce an actual electrolyte following the same recipe at different times and get different results (each would be different **electrolyte lots** of the same **electrolyte**). In order to accommodate the fact that often the specific lot is known but sometimes it isn’t, and different labs might not keep track of the lots at all whereas other labs might assign lots to everything, the system allows lots and blueprints to be

used more or less interchangeably. Concretely, the hierarchies are made of lots, but for each blueprint, there is a “default” lot which only has the blueprint information.

5.3.2 Obtaining Unique Names at Scale

In order to stay sane, name uniqueness properties must be both automatically enforced and well-defined.

Entities exist at every level of the hierarchy, and every entity has a name (e.g. an anode is an entity, a molecule is an entity, a dry cell lot is an entity). The desired properties are as follows:

1. The name of an entity is a subset of the information contained within the entity.
2. If two entities have different names, the entities must be truly different.
3. No two distinct entities have the same name.
4. To call two names different, they must be *semantically* different (must never have e.g. merely a different spelling for something, or a different ordering of elements).

The solution to this problem is simple: instead of maintaining names for each entity, we maintain a list of Boolean variables for each entity. The name of an entity is defined recursively on the hierarchy (i.e. molecules have names, then electrolyte names are defined by assembling the names of the molecules and the amounts, etc..).

The Boolean variables simply state whether a given piece of information should be part of the name or not (e.g. a dry cell has a Boolean for the cathode, a Boolean for the anode, and a Boolean for the separator, etc.).

Then, whenever we modify the set of entities, we ensure that

1. The entities without their Boolean variables are unique.

2. The *masked entities* without their Boolean variables are unique.

Masked entities are obtained by replacing each sub-component associated with a Boolean variable by a special value when the Boolean is False. For instance, if the anode Boolean is true, the anode is just the entity ID, but if the anode Boolean is False, the anode is replaced by a special ID.

In case that was confusing, here is a more mathematical treatment:

Entities contain many fields, and each field could either be a basic type: a string, a number, an entity ID which is itself only a positive integer, a set of entity IDs, or they could be a flagged type, which is a Boolean and a basic type.

On entities, we first define two projection operators:

1. Object Projection. The object projection maps *tuples* (i.e. finite sequences) of either basic types or flagged types to tuples of basic types in the natural way. Namely, for each component of the tuple, a basic type maps to itself, but a flagged type maps to the contained basic type.
2. Name Projection. The name projection maps tuples of either basic types or flagged types to tuples of either basic types or a special value denoted with the symbol \perp (an upside-down “T”). Namely, for each component of the tuple, a basic type maps to itself, but a flagged type maps to the contained basic type if the contained Boolean type is True but otherwise it maps to a special value denoted by the symbol \perp .

For each projection, we can define a notion of equivalence by saying that two entities are equivalent according to the projection if the two outputs of the projection are equal.

Then, when adding or modifying an entity, we simply check that no other entity is equivalent to it either according to the object projection or the name projection.

Intuitively speaking, the task of maintaining good unique names then boils down to selecting which aspects of an entity should be included in the name in order to distinguish it from other entities at a glance. It is always safe to set all the Booleans to True (meaning that the name will contain all the details), but this has the disadvantage of making very long names which are hard to read. Ideally, there are a few key properties that make an entity stand out, and those can be made visible.

Finally, when converting entities and their Booleans to a string, it must be ensured that, no matter the contents of the entities:

1. If two entities are not equivalent according to the name projection, then their strings will be distinct.
2. Strings are a function of **masked entities**.

5.3.3 Linking Experimental Files to the Appropriate Cells

There are at least three almost independent subsystems to the Universal Battery Database; the first dealing with what cells are (i.e. what electrolyte is in them, the *cell metadata*), the second dealing with what files are (i.e. what experiment is contained in a given file, the *file metadata*), and the third dealing with the data content of the files (i.e. what are the results inside of a given file, the *cycling data*). It is important to understand that the only safe assumption about files is that each filename is an address on the file system at which experimental equipment will write periodically.

Figure 5.4 shows the relationship between filenames, cycling data, valid file metadata, and cell IDs. Cell IDs exist in cell metadata (results of searches, see Section 5.3.1) and in valid file metadata. The link between a Cell ID and the cycling data **has to go through the valid file**

metadata. Furthermore, many files can have the same Cell ID contained within their valid file metadata. The cycling data of a single file contains information for each cycle and this information is indexed by the relative cycle number (each file begins at cycle 0) and the absolute time. Yet, the cycling data of a Cell ID (the cycling data from each file gathered together), is indexed by the absolute cycle number and time differences between a given cycle and the first cycle. As can be seen, valid file metadata for cycling files contains the Cell ID, the start cycle, which is used to offset the relative cycles contained within the file, the start date, as well as an ID for the creator of the experiment (Char ID), and finally some data about the experimental conditions such as the temperature the cell was operated at and the maximum voltage the cell was submitted to (Upper Cut-off Voltage).

The fundamental unit which always exists is the filename (viewed as an address on a file system). No matter what, if a file exists on the file system, the filename will enter the database.

Then, depending on whether the filename has encoded the valid file metadata information in an understandable format, this information will be decoded and stored automatically as a Valid File Metadata object. The *file metadata* subsystem is written to treat many different kinds of data, coming from many different equipment, and therefore, this Valid File Metadata object also contains information about the type of experiment (in the case of interest, “Cycling”) and the equipment used (in the case of interest, *Neware*⁹⁹).

If the valid file metadata information is not understood from the filename, no Valid File Metadata object is created for that filename. However, it is simple and quick to manually enter this information in the Universal Battery Database (see Section 5.4.4), as well as correct this information if it was somehow incorrect (for instance if a typo led the filename to mention the wrong Cell ID).

Therefore, the system only loads the actual cycling data (reads the content of the file) if the filename has a Valid File Metadata object associated to it. This automatic loading occurs at a fixed time every night, but it is possible to request a reading of the contents of a file through the Universal Battery Database (see Section 5.4.2).

The connection between filename and the Cell ID metadata (see Section 5.3.1) need not be maintained separately. Every time the search system looks for the data associated with a given Cell ID, it simply gets all the Valid File Metadata object containing the right Cell ID. If we are visualizing the data for a given cell, the system will find all the appropriate filenames as before, and the filenames for which some file content has been imported into the system will show up together in a unified view of the cycling data (called *Valid Cycling Data* for that cell, see Section 5.4.2).

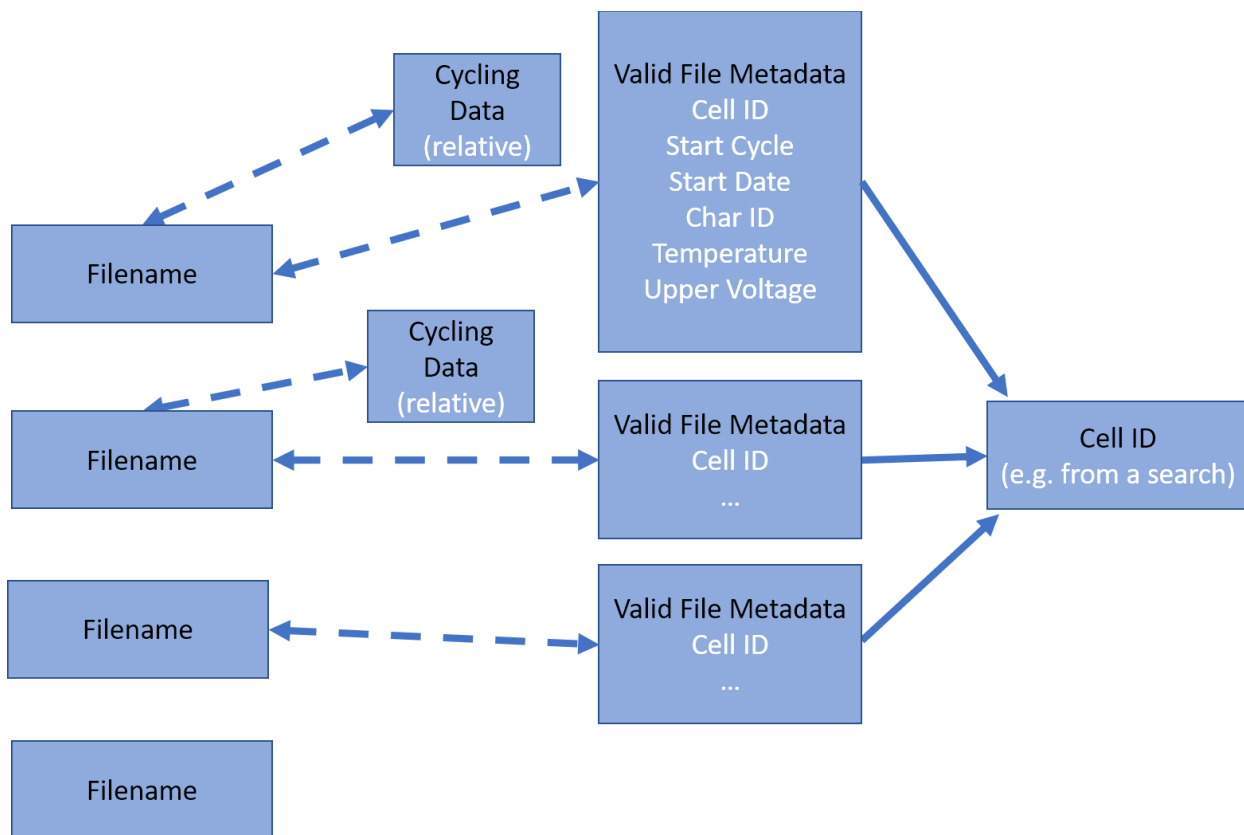


Figure 5.4 Illustration of the connection between filenames, cycling data, valid file metadata, and cell IDs. The first two filenames have valid file metadata associated with them as well as cycling data. The third file has valid file metadata associated to it, but no cycling data. It is impossible to have cycling data associated to a filename without valid file metadata. The fourth file does not have valid file metadata.

5.3.4 Searches Based on the Ontology of Lithium-ion Cells

When an entity is made of a combination of many entities at a lower level of the hierarchy, there can be many fewer sub-entities than the top level entity (i.e. if there are five possible widths, four possible heights, and three possible lengths, then there are sixty possible rectangular prisms). A prime example are the electrolyte entities (a few thousand unique electrolytes) which are a

combination of potentially many molecules (less than a hundred unique molecules, with only ten or so being commonly used). If the search mechanism cannot understand this hierarchy, searching for electrolytes is inconvenient. On the other hand, if the search mechanism allows to search electrolytes in terms of their molecules (and the amounts of each molecules), this suddenly becomes a very convenient tool to navigate thousands of electrolytes, and by the same logic, hundreds of thousands of cells.

On the other hand, very sparsely populated hierarchies can have the opposite effect where there are approximately the same number of top-level entities as their constituents. For instance, anodes are combinations of active and inactive materials, so we would expect that the number of distinct anodes is much greater than the number of distinct materials. However, almost every distinct anode uses a distinct active material. In such cases, forcing the structure of the ontology into the search mechanism has less clear benefits. There are no search mechanisms which will be optimal for all use cases and all data distributions. However, for the Universal Battery Database project, the search mechanism tries to strike a balance by allowing various searches for:

- electrolyte based on molecules
- dry cell based on anode, cathode, separators, and dry cell lots
- dry cell lots, which in the lab's case are physical boxes of dry cells which are supposedly identically manufactured, based on same inputs as for dry cell search
- wet cells based on electrolyte search, dry cell search, experimental conditions, filenames, and a higher level of organization called *datasets*. (the search can take into account the wet cells defined in the system, or the wet cells which appear in valid experimental files)
- distinct electrolytes based on wet cell search (i.e. which electrolytes have been tried with this cathode?)

- distinct dry cells based on wet cell search (i.e. which dry cells have been tried with X solvent combination under Y experimental condition)

The search mechanism can:

- simply list the names of the entities (see Section 5.4.2)
- allow the creation of groups of wet cells called *datasets* (see Sections 5.3.6 and 5.4.3)
- visualize the valid cycling data associated with the cell IDs (see Section 5.4.2)
- allow reimporting of missing data and various mechanisms to “fix” problems with the data (see Section 5.4.5)

It should be clear that the real measure of success of a search mechanism is whether the users can find what they are looking for and how easy it is to perform their searches. Therefore, a different search mechanism would have to be implemented for a laboratory dealing with several thousand cathodes and only a few electrolytes. Hopefully, many of the challenging types of search required for different labs already have analogue components within the current system.

Without discussing in detail the implementation of searches (also known as Queries) in the Structured Query Language (SQL) based database⁹⁶, it is insightful to discuss electrolyte searches in particular at the mathematical level.

For simplicity, assume that each molecule can be represented by a unique molecule ID. Then, electrolytes can be thought of as:

1. An unstructured field called *notes*.
2. A Boolean field called *proprietary*.
3. A list of triplets, where each triplet is made of:
 - a. An integer representing a molecule ID

- b. Either a number representing the *weight* of the given molecule or a special token representing an *unknown* weight.
- c. Either an overridden type (solvent, salt, or additive) or a special token representing the *default type* for the molecule

Then, each search has to produce a list of such electrolytes. A list of some common searches follows as a motivation. In order to remove unnecessary intellectual property from the discussion, assume that A, B, C, D, E, F, G, H, ... stand for actual molecules in the system.

- Electrolytes with 2 percent additive A, 1 percent additive B
- Electrolytes with C
- Electrolytes with only D as an additive
- Electrolytes with D as an additive, potentially A, but no other additives
- Electrolytes without any C
- Electrolytes with 30 percent solvent F, and salt G
- Electrolytes with around 2.3 (+/- 0.2) molal of salt H and only E and F as solvents
- Proprietary electrolytes
- Electrolytes with “company 5” in the notes
- Electrolytes with “1% H + 10%E” in the notes

Based on such examples, what are searches? Coming up with a restrictive definition of what a search is allows design of a **robust** search mechanism. Coming up with a definition which includes all the searches one might like to do ensures the search mechanism is **useful**.

Therefore, we define an electrolyte search as follows:

1. A triplet of Boolean fields called “complete salt”, “complete solvent”, “complete additive” or collectively “completes”
2. A real-valued positive number called “relative tolerance”
3. A Boolean field called “proprietary”
4. A triplet of unstructured fields called “notes 1”, “notes 2”, and “notes 3” or collectively “notes”
5. A list of quadruplets, with each quadruplet containing:
 - a. A molecule ID
 - b. A requirement type, either *mandatory*, *allowed*, or *prohibited*
 - c. A real-valued positive number called *amount* which could alternatively be set to an optional value denoted by the symbol \perp
 - d. A real-valued positive number called *tolerance*, could alternatively be set to a special value denoted by the symbol \perp

As an example, Table 5 illustrates the purpose of the various components of a *search* by giving examples of how to represent various informal searches into the restricted domain.

Now the question is how to map these *searches* to a set of *electrolytes*. Beside suggesting interested parties to explore the code itself for the fully general search case, a few more details follow. In addition, see Section 5.4.2 for various examples in the user interface.

1. For each electrolyte, we can go through the list of tuples, and filter out those electrolytes which contain a prohibited molecule.
2. If one of the “completes” is set to True (without loss of generality, consider “complete additive” set to True), we can go through the list of tuples in each electrolytes, and for every molecule either marked as *additive* or marked as *default* with the default type of the

molecule being additive, we can check if the molecule is contained within the set of molecules in the search marked either as *mandatory* or as *allowed*.

3. We can go through the list of quadruplets in the search marked as “mandatory” and then for every electrolyte, verify the presence of the molecule (if no amount was specified) or verify that the molecule is present and the amount falls within the range proscribed in the search (if an amount was specified).

Search in words	Completes (default is all false)	relative tolerance (default is 5 percent)	Proprietary (default is false)	notes	quadruplets
2 percent additive A 1 percent additive B					(A, mandatory, 2, \perp) (B, mandatory, 1, \perp)
with C					(C, mandatory, \perp , \perp)
with only D as an additive	complete additive: True				(D, mandatory, \perp , \perp)
with D as an additive, potentially A, but no other additives	complete additive: True				(D, mandatory, \perp , \perp) (A, allowed, \perp , \perp)
without any C					(C, prohibited, \perp , \perp)
with 30 percent solvent F, and salt G					(F, mandatory, 30, \perp) (G, mandatory, \perp , \perp)
with 2.3 (+/- 0.2) molal of salt H only E and F as solvents	complete solvent: True				(H, mandatory, 2.3, 0.2) (F, allowed, \perp , \perp) (E, allowed, \perp , \perp)
Proprietary			Proprietary: True		
with "company 5" in the notes				Notes 1: "company 5"	

Table 5: A few examples of how various informal searches can be represented as restricted formal searches.

5.3.5 Data Processing and Grouping the Cycles Automatically

Lithium-ion cycling data has a repeating structure where every *cycle* the cell must undergo charge and discharge. More generally, a cycle will contain a sequence simple steps, where each step is typically controlled via a drive profile. For instance,

1. Charge the cell with a constant current of X milliamperes until the voltage across the terminal reaches Y volts.
2. Discharge the cell with a constant current of X until the voltage reaches Y.
3. Charge at a constant current of X until voltage reaches Y, but then hold a constant voltage of Y until the current drops below Z.
4. Rest for W minutes (open circuit).
5. Hold a constant voltage for W minutes.

Each step is made of a sequence of observations, typically of the voltage, the current, and the time. By integrating the current multiplied by differences in time, we get the capacity, and by integrating the current multiplied by the voltage multiplied by the differences in time, we get the energy.

The choice of drive profile has two main impacts:

1. The observed results for that step strongly depend on the drive profile for that step. For instance, at higher currents, there is typically less capacity, and more polarization in the voltage between charge and discharge (i.e. drive profile determines what information is extracted).
2. The state of the cell, both in terms of reversible short-term changes to the expected results for the next few steps and in terms of the irreversible long-term degradation of the cell, is affected a little bit every step, based on the precise drive profile (i.e. drive profile determines degradation).

The first point is important since one can get useful information by periodically adding a few “check-up” cycles which by either using a low or high current, or by setting different bounds on the voltages, will reveal different information about the cell.

The second point is mostly important in aggregate. For instance, there will be differences in aging if every cycle uses a very high current, in comparison to a different experiment where every cycle uses a very low current.

Due to the presence of rare “check-up” cycles, there is a very large imbalance between observations for different drive profiles within a single experiment. This both makes it tricky to separate these cycles by hand (for humans), but also necessary to separate these cycles properly for various algorithms (otherwise, models will have trouble with the information in the rare cycles since they are overwhelmed with the more common cycles).

If every cycle had a precise, exact, intended drive profile, which only had a few possible values for each cell, then it would be trivial to group cycles (i.e. they would effectively be already grouped in the raw data).

However, for many data formats and many chargers, the intended drive profile is not included, the currents are approximate and rarely exactly constant, and generally things are messy.

We therefore identify five attributes that every charge or discharge possesses and apply an approximate clustering algorithm to generate a small number of meaningful “groups” across these attributes.

The details of the clustering methods are not critical, but the goals are that:

1. Every cycle’s charge/discharge must belong to at least one group.
2. The number of groups should be as small as possible.
3. The five attributes should be as uniform within a given group as possible. Namely, the standard deviation of each attribute should be as small as possible when taken within a group.

Now, given a clustering method which optimizes these three objectives, the attributes themselves must be chosen such that the groups thus obtained would explain as much of the variance in observations between groups as possible.

Intuitively, we want attributes which capture the most glaring variations between observations from cycle to cycle.

In case the mathematical details are of interest, a step typically contains a set of tuples containing voltage, capacity, current, and time which can be abstracted to a function of voltage, capacity and current versus time through some form of interpolation. Then, for nearby cycles, such functions may be compared to each other by the integral of the squared sum of their differences.

To summarize, the attributes must be chosen in order to minimize those distances for cycles in a given group.

Technically, more is needed to really describe what we want (for instance, the attributes which must explain the variance are the **average attributes** of a given group; they can't change as a function of cycle number; and we are interested in all possible drive profiles). However, the intuition given can hopefully explain the choice of attributes used, even though there may be better attributes yet to be discovered.

The attributes chosen are:

1. **Constant Rate.** The current during the “constant current” portion of the given step. (For ease of comparison, the current is normalized by the theoretical capacity of a cell.)
2. **End Rate.** The current at the end of a given step. (Again, normalized to the theoretical capacity of the cell). This will differ from the Constant Rate in cases where e.g. the

charge has a constant current portion followed by a voltage hold until the current falls below a certain threshold.

3. **Previous End Rate.** The End Rate of the previous step. (For instance, the previous step of a discharge is a charge and vice-versa.)
4. **End Voltage.** The voltage at which the given step terminated.
5. **Previous End Voltage.** The End Voltage of the previous step.

These attributes can distinguish between slow charge and fast charge, similarly with discharge. They can also distinguish between different cycles which may cycle across a restricted voltage range. However, they do not capture short-term effects where having a slow cycle at a given point will affect a handful of subsequent cycles^{100,101}.

Yet, this ability to isolate various types of cycles designed to study different aspects of the cell is quite useful for visualization purposes, as well as for balancing the data prevalence in the context of developing a model.

See Table 6 through Table 8 for some examples of *filters* applied to the five attributes.

5.3.6 Creating and Annotating Datasets

The search mechanisms allows one to find individual cells. Yet, for the purpose of a given project, one is usually interested in a fixed subset of cells.

Furthermore, when the set of cells is small (for instance, ~10 cells), then finding meaningful names for each cells is much easier and the names can be much shorter. For instance, if there are five cells which are identical but for the fact that they respectively incorporate 1%, 2%, 3%, 4%, and 5% of a given additive in their electrolytes, then it may be much easier to simply name them “1%”,

“2%”, “3%”, “4%”, and “5%” in the context of this dataset, even if such a name would be completely inadequate in a complete database of tens of thousands of cells. Indeed, it might still be useful to give them such names, even if they were not completely identical in other respects, as long as they were the best controlled cells which could be assembled to compare the given additive’s impact.

Finally, in the context of a given project, the kinds of cycles which are of interest for a given cell will usually be restricted. Indeed, it would be rare to require the five attributes mentioned in Section 5.3.5 to describe a group of cycles in this context. As an example, if the End Voltage and Previous End Voltage attributes are always the same for every discharge, but the Constant Rate is different, then it could be easier to give a special name to a subset of cycles, such as “1C” to denote a discharge at a constant current equal to the theoretical capacity of a given cell divided by one hour. Furthermore, the name could be kept consistent across different cells of the dataset, even if some cells were in fact cycled at “0.87C” while others were cycled at “1.12C”, for the purpose of visual clarity.

With those ideas in mind, the design of the “dataset” system is as follows:

1. Creating a dataset is as easy as finding a name for the dataset that has not already been used.
2. The output of a cell search can be added to any dataset (it is not possible to add a cell to a dataset multiple times).
3. Any cell may be renamed in the context of any dataset. This name does not affect anything outside of the given dataset. The “dataset specific name” of a cell must be unique within the dataset.

4. For a given cell, we can define a number of “filters”, each of which requires a different name. These filters can put range constraints on any of the five attributes for the charge or the discharge of a cycle. Namely, either any value is accepted for a given attribute, or an upper and lower limit is provided. These filters each represent the subset of all the cycles for the given cell which satisfy all the constraints on their attributes.
5. When defining filters, one can define the same filter for any subset of cells in the dataset.

Consider a couple of examples.

First, imagine a dataset of various cells where most cycles are charging and discharging over a restricted voltage range (could vary cell to cell) at a given rate (could vary cell to cell). Then, approximately every 500 hours, a “check-up” cycle is inserted: the cells are cycled over a larger range of voltages (from 3.2 to 4.2 volts for some cells; from 2.8 to 3.9 volts for others) at a rate of charge and discharge of $1/20$ hours⁻¹ (often referred as C/20, said “Sea over twenty”). Then, the goal is to design a filter which only captures the “check-up” cycle, and a filter which excludes the “check-up” cycle.

In words, the check-up filter would be “C/20 constant rate for charge and discharge; charge starting at 3.2 and ending at 4.2 volts; discharge starting at 4.2 and ending at 3.2 volts”. Depending on the data in the set, the tolerance on these quantities should be set appropriately, but to a first approximation, “C/20” would be 0.05, so we could accept a range of 0.03 to 0.07. For the voltage, we could tolerate up to 0.02 volts. In this scenario, Table 6 shows two plausible filters which could be applied to the whole dataset, though only one of them would be active for any given cell.

Furthermore, consider the cases where the cells with “check-up” cycle between 3.2 and 4.2 volts have their restricted ranges between 3.2 and some voltage lower than 4 volts (could vary cell to

cell), while the cells with check-up cycle between 2.8 and 3.9 have their restricted ranges between some voltage higher than 2.9 volts and 3.9 volts. In this scenario, Table 7 shows two plausible filters which could be applied to the whole dataset, though only one of them would be active for any given cell, and would essentially exclude the check-up cycles, which may be useful for some analysis.

Second, imagine a different dataset of various cells where most cycles are charging and discharging at a given rate (could vary cell to cell, either 1C or C/3 charge and discharge, but consistent for a given cell). Then, approximately every 100 cycles, a “Rate map” set of cycles is inserted: the cells are charged at 1C, discharged at C/20, then charged and discharged at C/20, then charged at 1C and discharged progressively faster (C/2, 1C, 2C, 3C). The goal is to design filters for each of the Rate map cycles and which does not include the regular cycles. In this scenario, Table 8 shows an example of such filters.

Name:	Attribute	Polarity	Minimum	Maximum
Check-up High V	Constant Rate	Charge	0.03	0.07
	Constant Rate	Discharge	0.03	0.07
	End Voltage	Charge	4.18	4.22
	Previous End Voltage	Charge	3.18	3.22
	Previous End Voltage	Discharge	4.18	4.22
	End Voltage	Discharge	3.18	3.22
Check-up Low V	Constant Rate	Charge	0.03	0.07
	Constant Rate	Discharge	0.03	0.07
	End Voltage	Charge	3.88	3.92
	Previous End Voltage	Charge	2.78	2.82
	Previous End Voltage	Discharge	3.88	3.92
	End Voltage	Discharge	2.78	2.82

Table 6 Examples of dataset filters to isolate “Check-up” cycles.

Name:	Attribute	Polarity	Minimum	Maximum
No Check-up High V	End Voltage	Charge		4.0
	Previous End Voltage	Charge	3.18	3.22
No Check-up Low V	End Voltage	Charge	3.88	3.92
	Previous End Voltage	Charge	2.9	

Table 7 Examples of dataset filters to get the various cycles except the “Check-up” cycles.

Name:	Attribute	Polarity	Minimum	Maximum
1C C/20	Constant Rate	Charge	0.85	1.5
	Constant Rate	Discharge	0.03	0.1
C/20 C/20	Constant Rate	Charge	0.03	0.1
	Constant Rate	Discharge	0.03	0.1
C/2	Constant Rate	Discharge	0.3	0.7
1C (Check-up)	Constant Rate	Discharge	0.85	1.5
	Previous Constant Rate	Charge	0.3	0.7
2C	Constant Rate	Discharge	1.5	2.4
3C	Constant Rate	Discharge	2.4	4

Table 8 Examples of dataset filters to get the various cycles in a “Rate Map”.

5.3.7 Outputting Experimental Data in a Simple Format (for Humans)

The observations contained in cycling data in their raw form are pretty complex in structure and this acts as a constant source of friction for analysis and visualization tasks. Among the complications, the data for a given cell may exist **across multiple files**, it is **updated** regularly, and there are **many types of cycles** useful for studying different aspects of the performance of lithium-ion cells. The observations for a single cycle are also quite complex in structure and difficult to easily compare from cycle to cycle. Cycles are split across various steps, each containing some set of observations (voltage, current, capacity, time), but the voltages are not typically “aligned” from cycle to cycle, with the number of observations per cycle rarely constant from cycle to cycle, and the time given is sometimes unreliable. This data structure is difficult to work with. Therefore, *summary quantities* are typically used, such as the *total discharge capacity*, the *average voltages of charge*, the *average voltage of discharge*, their differences called *delta V*, etc..

To the summary quantities can be added “index quantities” such as the cycle number and the time difference between the first cycle and a given cycle (called the *cumulative time*). Then, successive cycles will produce the rows of a matrix, with the columns corresponding each to a specific summary quantity, or a specific index quantity.

This format can be produced for any cell and any filter, and it can easily be formatted as a Comma-Separated Values (CSV) file, which can be manipulated with various programs such as Microsoft Excel, and can be read in by almost all graphing programs worth the name¹⁰².

In practice, when outputting the data,

1. every dataset becomes a **folder**,
2. every cell becomes a **subfolder**, and
3. every filter becomes a **CSV file**.

Every CSV file has the same columns in the same order, corresponding to the various summary quantities and index quantities.

The same general idea may be applied for more special-purpose uses. For instance, the differential capacity as a function of voltage (also known as dQ/dV versus V) can be useful to look at for cycles with a relatively low current (such as $C/20$).

Then, by having a filter which selects only the appropriate cycles, the data could either be presented in a single CSV file with columns Cycle Number, dQ/dV , V , or it could be presented as a multitude of files in a *sub* subfolder with only dQ/dV and V as columns but with the Cycle Number as the filename. For instance, a file named **105.csv** would exist in a sub subfolder named **DifferentialCapacityVersusVoltage**.

The general idea is to *unroll* nested data structures as matrices (CSV files) within a hierarchy (the filesystem).

5.3.8 Outputting Data in a Complete Format (for Machine Learning Algorithms)

In order for a machine learning algorithm to be easily applied to a dataset, the format should:

1. provide as much information as possible since different models may require different levels of details
2. Store things compactly but in a way which can quickly be loaded
3. Be easily translatable into uniform tensors as much as possible
4. No customization, good automation

In the context of producing an output suitable for machine learning algorithms, a process which we call “compiling a dataset”, there are two difficulties:

1. Describing the cell metadata
2. Describing the cycling data (including the relevant valid file metadata)

The cell metadata is represented by assigning a unique integer to every unique entity.

The cycling data is represented as groups containing sequences of cycles. Each cycle contains some metadata, including temperature, as well as some conditions like the Constant Current. Furthermore, each cycle contains a sequence of voltage and capacity pairs. As future work, it may be worth representing time along with voltage and capacity.

For most users of the database, this feature will not be useful, so we do not go into all the details, but we refer interested readers to the code for all the details.

5.4 User Manual and Frequently Asked Questions

Here are listed the most common usages, organized by subsections corresponding to tasks one might use the Universal Battery Database for. In each case, the most frequently encountered confusions are discussed in more details.

Since the database contains sensitive content, some of the specific values have been obscured by a grey “sensitive content” sign. In cases where the general structure of the content is important, a grey box containing a **fake** content with the appropriate structure was superimposed on the image.

5.4.1 Access to the Universal Battery Database

The code is freely and openly available at <https://www.github.com/Samuel-Buteau/universal-battery-database> and the system can be deployed in various ways.

Figure 5.5 shows the way to interact with the software within a lab network. From any computer within the lab “intranet”, visiting a given URL from a browser will connect the user to the software. By following the hyperlinks at the top, the user can visit the various subsystems, again from the browser of a computer on the local network of the lab (in this configuration, the IP address of the server must be visible to the other computers on the lab network).

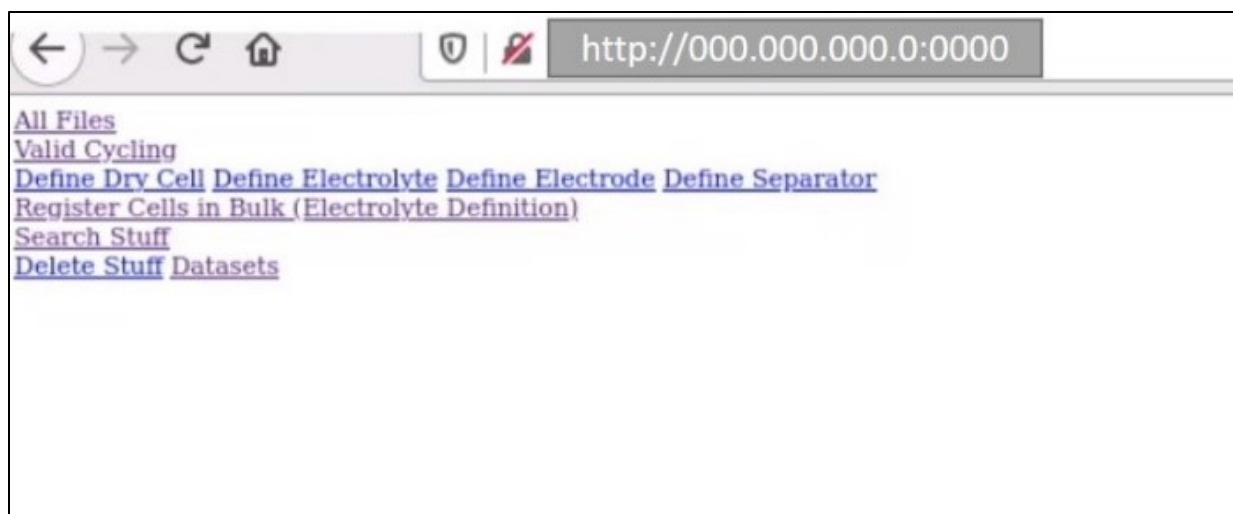


Figure 5.5 The front page of the software accessed from a computer within the lab’s network. The front page shows only the various hyperlinks to leading to the various subsystems of the software. These hyperlinks are present at the top of every page within the software.

In the absence of any internet connection, the software may still be set up and used from a browser on the computer hosting the software (also known as the “server”).

Finally, it would be possible to allow connections from all across the internet, but in such a scenario, an authentication mechanism should be deployed and maintained, in order to minimize theft of intellectual property. However, setting up a secure system which can hold valuable intellectual property with a perpetually available surface of attack from across the world is beyond the scope of this work. Security is easy to do **poorly**, and all the details potentially matter a great deal. While it would be beneficial to have a robust security-oriented design for the Universal Battery Database project, it is left as future work. Also worth noting that a good security design for the Universal Battery Database could still be insecure when deployed side-by-side with the various legacy systems keeping the lab running.

5.4.2 Searching the Database

There are various levels at which searches may be performed, corresponding to various levels in the ontology of lithium-ion cells.

Figure 5.6 shows an overview of the search page, as it would show up when clicking on the hyperlink **Search Stuff**. There are four main sections to the search page. At the top, the results for the wet cell searches and for the valid cycling data searches will show up, then there is a table subdivided into the three remaining sections: the valid cycling data section, the electrolyte section, and the dry cell section. Each section will be explored further below.

Figure 5.7 shows the structure of the possible searches in the form of a *search tree*. The rounded nodes on this diagram can be searched *for* (for instance, we can search *for* some electrolytes), whereas the square nodes are what the searches are *based on* (for instance, we can search for some electrolytes *based on* their substructure). Generally, Figure 5.7 depicts a *tree*. An arrow connecting node A to node B indicates that node A is a *child* of node B. There are two types of searches: *simple* and *powerful*. Simple searches are performed based on the square *descendants* of the node. Powerful searches, on the other hand, can follow arrows in both directions, essentially projecting a search (simple or powerful) for the direct parent of a node (e.g. wet cell) onto the node itself (e.g. electrolyte).

The projection of a search for a parent onto a child node is defined as follows: given the set of results for the parent node (e.g. wet cell), extract the values for the child node (e.g. electrolyte) of each element of the set of results (e.g. the electrolyte of each wet cell), and list the set of *distinct* values for the child node (e.g. the distinct set of electrolytes). Note that this projection can be based

on a simple search (e.g. a wet cell search based on electrolyte substructure and dry cell substructure) or on a powerful search (e.g. a valid cycling data search projected onto the wet cell node).

The following subsections explore various possibilities, but to clarify the general principle, here follows a list of three different searches for electrolytes:

1. A simple search for electrolytes based on some additive molecules being present at certain percentages (electrolyte substructure).
2. A powerful search for electrolytes based on the same additive molecules, as well as the dry cell box ID. This is interpreted as a simple search on wet cells projected onto electrolytes.
3. A powerful search for electrolytes based on the same additive molecules, the same dry cell box ID, and the temperature of cycling (experimental conditions). This is interpreted as a simple search on valid cycling data projected onto wet cells (a powerful search on wet cells) which is itself projected onto electrolytes.

Note that the first search is completely independent of which dry cells might exist or which wet cells might be registered, or which valid file metadata may be defined. However, the second search does depend on which wet cells are registered, and the third search depends on which wet cells and which valid cycling data exist within the system.

Furthermore, for every square node in Figure 5.7, we can define a *trivial* input as the input which will allow every possible variations of the node to match in the search (e.g. if everything is left blank in the electrolyte substructure node, then a search on electrolytes will return all existing electrolytes). These trivial inputs are a useful tool to find and fix problems in the database.

For any given simple search, we can define one or more *corresponding* powerful searches, as powerful searches which have *exactly the same nontrivial inputs as the simple search*. For instance, the following three searches are *corresponding searches*:

1. A simple search for electrolytes based on some additive molecules being present at certain percentages (electrolyte substructure).
2. A powerful search for electrolytes based on the same additive molecules, as well as a trivial input for the dry cell box ID, dry cell substructure, dataset, and cell ID.
3. A powerful search for electrolytes based on the same additive molecules, the same trivial inputs as before, and trivial inputs for the dataset, cell ID, experimental conditions, and file substructure.

If all data were present and properly entered in the database, then corresponding searches most likely should all return the same results. In other words, there should not be unused electrolytes which never appear in a wet cell. Similarly, if a wet cell is registered into the system and this cell is or has been cycled, it would be expected that it should appear in a valid cycling data.

Therefore, if these searches do not produce the same results, the most likely explanation is that a wet cell has been registered improperly, or not at all, or that the experimental data for a given cell is named incorrectly or the format of the file is corrupted.

In a perfect world, every contributor to the database would ensure correct data entry, but in practice, it is necessary from time to time to fix inconsistencies. Yet, because of the persistent nature of the database, incorrect data need only be corrected once, and so the quality and reliability of the powerful searches should only increase with usage.

Search Stuff

Wet Cells

Display Page: 1 Show Visuals: Preview Wet Cell Preview Validated Cycling Data

Keyword in Filename	Value to search			Electrolyte Display Page: 1 Preview Electrolyte Complete salt: <input type="checkbox"/> Complete solvent: <input type="checkbox"/> Complete additive: <input type="checkbox"/> Relative tolerance: 5.0 <small>the default tolerance in percentage</small> Proprietary flag: <input type="checkbox"/> Notes1: <input type="text"/> Notes2: <input type="text"/> Notes3: <input type="text"/> Only wet cell electrolytes: <input type="checkbox"/>	Dry Cell Display Page: 1 Box id1: <input type="checkbox"/> Box id2: <input type="checkbox"/> Box id3: <input type="checkbox"/> Box id4: <input type="checkbox"/> Box id5: <input type="checkbox"/> Notes: <input type="text"/> Relative tolerance: 5.0 Proprietary: <input type="checkbox"/> Geometry category: <input type="checkbox"/> Geometry category exclude missing: <input type="checkbox"/> Cathode: <input type="checkbox"/> Cathode exclude missing: <input type="checkbox"/> Anode: <input type="checkbox"/> Anode exclude missing: <input type="checkbox"/>																																											
Keyword in Filename																																																
Keyword in Filename																																																
Keyword in Root																																																
Keyword in Root																																																
Keyword in Root																																																
	Exact Match	Minimum	Maximum																																													
Char ID																																																
Cell ID																																																
Upper Cutoff Voltage																																																
Temperature																																																
Date																																																
Dataset																																																
Limit other searches: <input type="checkbox"/>				<table border="1"> <thead> <tr> <th>Molecule</th> <th>Must have?</th> <th>Amount</th> <th>Tolerance</th> </tr> </thead> <tbody> <tr><td>-----</td><td>mandatory</td><td></td><td></td></tr> <tr><td>-----</td><td>mandatory</td><td></td><td></td></tr> <tr><td>-----</td><td>mandatory</td><td></td><td></td></tr> <tr><td>-----</td><td>mandatory</td><td></td><td></td></tr> <tr><td>-----</td><td>mandatory</td><td></td><td></td></tr> <tr><td>-----</td><td>mandatory</td><td></td><td></td></tr> <tr><td>-----</td><td>mandatory</td><td></td><td></td></tr> <tr><td>-----</td><td>mandatory</td><td></td><td></td></tr> <tr><td>-----</td><td>mandatory</td><td></td><td></td></tr> <tr><td>-----</td><td>mandatory</td><td></td><td></td></tr> </tbody> </table>	Molecule	Must have?	Amount	Tolerance	-----	mandatory			-----	mandatory			-----	mandatory			-----	mandatory			-----	mandatory			-----	mandatory			-----	mandatory			-----	mandatory			-----	mandatory			-----	mandatory		
Molecule	Must have?	Amount	Tolerance																																													
-----	mandatory																																															
-----	mandatory																																															
-----	mandatory																																															
-----	mandatory																																															
-----	mandatory																																															
-----	mandatory																																															
-----	mandatory																																															
-----	mandatory																																															
-----	mandatory																																															
-----	mandatory																																															

Figure 5.6 Overview of the search page, as it would show up when clicking on the hyperlink **Search Stuff**. The searches are arranged hierarchically according to the search tree of Figure 5.7. At the top, the results for the Wet Cell searches and for the Valid Cycling Data searches will show up, then there is a table subdivided into the three remaining sections. From left to right, the valid cycling data section, the electrolyte section, and the dry cell section. The results of electrolyte searches and dry cell searches will show up in their respective sections.

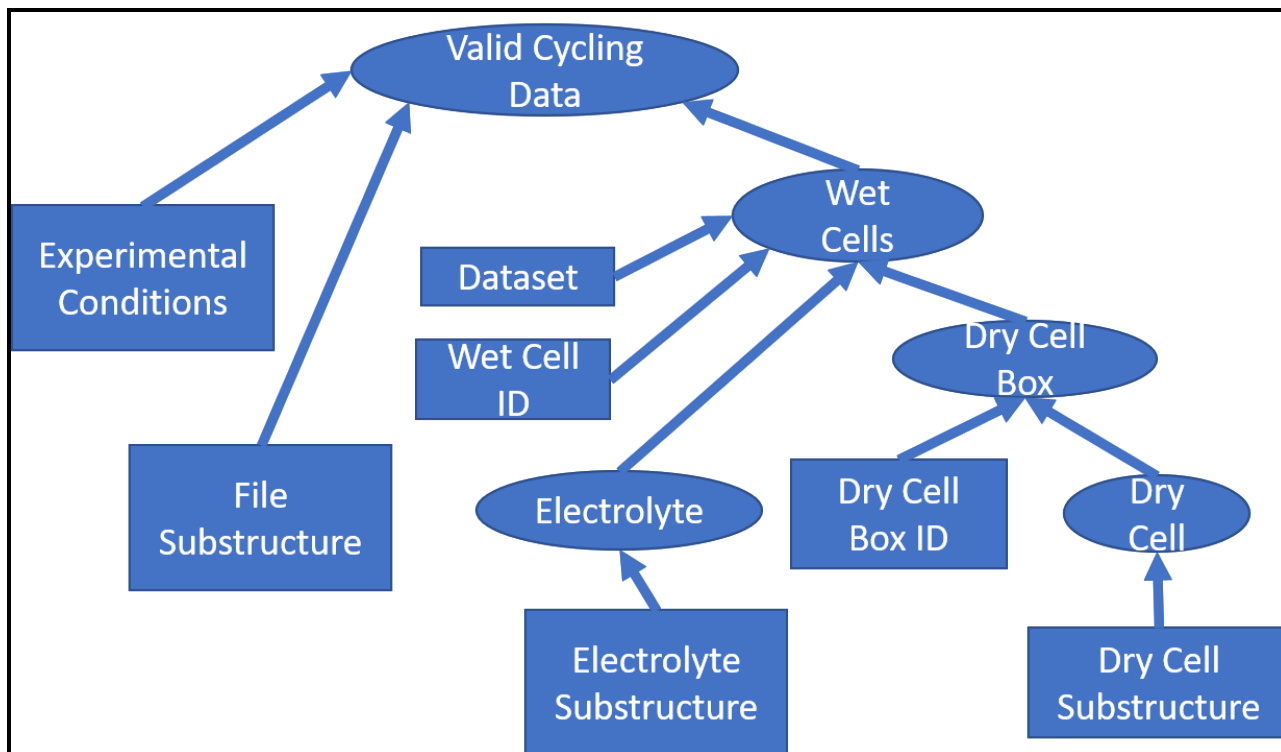


Figure 5.7 Illustration of the structure of the possible searches, called the “search tree”. The rounded nodes on this diagram can be searched *for* (for instance, we can search *for* some electrolytes), whereas the square nodes are what the searches are *based on* (for instance, we can search for some electrolytes *based on* their substructure). Generally, this figure depicts a *tree* and an arrow connecting node A to node B indicates that node A is a *child* of node B. There are two types of searches: *simple* and *powerful*. Simple searches are performed based on the square descendants of a node. Powerful searches, on the other hand, can follow arrows in both directions, essentially projecting a search (simple or powerful) for the direct parent of a node (e.g. wet cell) onto the node itself (e.g. electrolyte).

The following sub sections explore various examples of searches in more detail.

Simple Electrolyte Search

Figure 5.8 shows a zoomed-in view of the electrolyte section of the search page, with an example input for the electrolyte substructure node corresponding to the first entry in Table 5, as it would appear just before the search. Note that dark grey boxes are masking actual molecules and instead use abstract letters. This input corresponds to an electrolyte with 2 of A and 1 of B. For instance, if A and B were used as additives, it would mean 2 percent of total weight of electrolyte is made of molecule A and one percent made from B (the default tolerance is 5 percent, so the actual ranges of acceptable total weight ratios are 1.90 to 2.10 and 0.95 to 1.05 respectively). To perform a search for electrolytes, simply click on the **Preview Electrolyte** button. To perform a simple search, make sure that **Only wet cell electrolytes** is unchecked. To perform a powerful search, check the **Only wet cell electrolytes** box. To use a simple search for wet cells in the powerful search for electrolytes, **Limit other searches** should be left unchecked. To use a simple search for valid cycling data in the powerful search for electrolytes, **Limit other searches** should be checked together with **Only wet cell electrolytes**. Table 9 illustrates these options.

Figure 5.9 shows the results of the simple electrolyte search. The electrolytes shown are fictitious but the format is actual.

1. Solvents are listed in decreasing order of weight percent (normalized to 100% for the total solvents).
2. Salts are listed in decreasing order of molality.
3. Additives are listed in decreasing order of weight percent of total electrolyte. In this particular search, there are many results.

In order to keep response times short and to display effectively the list, a page number must be provided with the search, and only a certain number of electrolytes will be displayed on each page.

This page mechanism is only active at the node searched for. In other words, when searching for a wet cell, the page number of the electrolyte search does not matter; all electrolytes in all pages will be used in the wet cell search.

Electrolyte				Dry C
Display Page: 1 <input type="button" value="Preview Electrolyte"/>				Display I
Complete salt:	<input type="checkbox"/>			Box id
Complete solvent:	<input type="checkbox"/>			Box id
Complete additive:	<input type="checkbox"/>			Box id
Relative tolerance:	5.0 <input type="button" value="↑"/> <input type="button" value="↓"/>	the default tolerance in percentage.		Box id
Proprietary flag:	<input type="checkbox"/>			Box id
Notes1:	<input type="text"/>			Note
Notes2:	<input type="text"/>			Relati
Notes3:	<input type="text"/>			toleran
Only wet cell electrolytes:	<input type="checkbox"/>			Proprie
Molecule	Must have?	Amount	Tolerance	Geome
A	mandatory <input type="button" value="v"/>	2 <input type="button" value="↑"/> <input type="button" value="↓"/>	<input type="button" value="↑"/> <input type="button" value="↓"/>	catego
B	mandatory <input type="button" value="v"/>	1 <input type="button" value="↑"/> <input type="button" value="↓"/>	<input type="button" value="↑"/> <input type="button" value="↓"/>	Geome
----	<input type="button" value="v"/> mandatory <input type="button" value="v"/>	<input type="button" value="↑"/> <input type="button" value="↓"/>	<input type="button" value="↑"/> <input type="button" value="↓"/>	catego
----	<input type="button" value="v"/> mandatory <input type="button" value="v"/>	<input type="button" value="↑"/> <input type="button" value="↓"/>	<input type="button" value="↑"/> <input type="button" value="↓"/>	exclu
----	<input type="button" value="v"/> mandatory <input type="button" value="v"/>	<input type="button" value="↑"/> <input type="button" value="↓"/>	<input type="button" value="↑"/> <input type="button" value="↓"/>	missin
----	<input type="button" value="v"/> mandatory <input type="button" value="v"/>	<input type="button" value="↑"/> <input type="button" value="↓"/>	<input type="button" value="↑"/> <input type="button" value="↓"/>	Catho
----	<input type="button" value="v"/> mandatory <input type="button" value="v"/>	<input type="button" value="↑"/> <input type="button" value="↓"/>	<input type="button" value="↑"/> <input type="button" value="↓"/>	
----	<input type="button" value="v"/> mandatory <input type="button" value="v"/>	<input type="button" value="↑"/> <input type="button" value="↓"/>	<input type="button" value="↑"/> <input type="button" value="↓"/>	
----	<input type="button" value="v"/> mandatory <input type="button" value="v"/>	<input type="button" value="↑"/> <input type="button" value="↓"/>	<input type="button" value="↑"/> <input type="button" value="↓"/>	
----	<input type="button" value="v"/> mandatory <input type="button" value="v"/>	<input type="button" value="↑"/> <input type="button" value="↓"/>	<input type="button" value="↑"/> <input type="button" value="↓"/>	

Figure 5.8 Zoomed-in view of the electrolyte section of the search page, with an example input for the electrolyte substructure node corresponding to the first entry in Table 5, as it would appear just before the search. This input corresponds to an electrolyte with 2 of A and 1 of B.

Electrolyte

Display Page: Showing page 1/4.

Preview Electrolytes			
85%D+15%C + 0.5mPepper + ADDITIVES:(2%A+1%B)			
D + 0.3mPepper + ADDITIVES:(2%A+1%B)			
D + 0.1mSalt + ADDITIVES:(2%A+1%B+1%C)			
50%D+50%E + 0.1mSalt + ADDITIVES:(2%A+1%B+1%C)			
50%D+50%E + 0.1mSalt + ADDITIVES:(2%A+1%B)			
50%D+25%E+25%F + 0.1mSalt + ADDITIVES:(2%A+1%B)			
50%D+50%F + 0.2mSalt + ADDITIVES:(2%A+1%B)			
50%D+50%F + 0.2mSalt + ADDITIVES:(2%A+1%B+0.5C)			
50%D+25%E+25%F + 0.1mSalt + ADDITIVES:(5%C+2%A+1%B)			

Complete salt:	<input type="checkbox"/>
Complete solvent:	<input type="checkbox"/>
Complete additive:	<input type="checkbox"/>
Relative tolerance:	<input type="text" value="5.0"/> the default tolerance in percentage.
Proprietary flag:	<input type="checkbox"/>
Notes1:	<input type="text"/>
Notes2:	<input type="text"/>
Notes3:	<input type="text"/>
Only wet cell electrolytes:	<input type="checkbox"/>

Molecule	Must have?	Amount	Tolerance
A	mandatory <input type="text" value="v"/>	2 <input type="text" value=""/>	<input type="text" value=""/>
B	mandatory <input type="text" value="v"/>	1 <input type="text" value=""/>	<input type="text" value=""/>
----	mandatory <input type="text" value="v"/>	<input type="text" value=""/>	<input type="text" value=""/>

Dry Cell

Display Page: Showing page 1/4.

Box id	<input type="text"/>
Box id	<input type="text"/>
Box id	<input type="text"/>
Box id	<input type="text"/>
Box id	<input type="text"/>
Notes	<input type="text"/>
Relative tolerance	<input type="text"/>
Proprietary flag	<input type="checkbox"/>
Geometry category	<input type="text"/>
Geometry category excluded	<input type="checkbox"/>
Missing	<input type="checkbox"/>
Cathode	<input type="text"/>
Cathode excluded	<input type="checkbox"/>
Missing	<input type="checkbox"/>
Anode	<input type="text"/>
Anode excluded	<input type="checkbox"/>
Missing	<input type="checkbox"/>
Separator	<input type="text"/>

Figure 5.9 Results of a simple search for electrolytes based on the input of Figure 5.8.

Simple Dry Cell Search

Figure 5.10 shows a zoomed-in view of the dry cell section of the search page, with an example input for the dry cell substructure, as it would appear just before the search. This input corresponds

to a dry cell with either a *pouch*, *coin*, or *missing* geometry category, as well as the cathode D5 E4 F[LMO] and anode [Stone, ART]. These names mean that the active material for the cathode is a Lithium Metal Oxide (LMO) with stoichiometry ratios for atoms *D*, *E*, *F* of 5, 4, 1 without any specified inactive materials, and the anode's active material is of type *Stone* and has attribute *artificial*. The searches are performed on subsets of all possible cathodes, anodes, and separators. **By holding down the *Ctrl* key and clicking on the various choices, it is possible to select exactly the subset of cathodes which would match the search.** Selected entries appear blue and unselected entries appear with a white background. To perform a search for dry cells, simply click on the ***Preview Dry Cell*** button, similarly with a search for dry cell lots (also known as dry cell boxes), simply click on the ***Preview Box IDs*** button. Table 9 gives the options to control the type of search. To perform a simple search, make sure that ***Only wet cell dry cells*** is unchecked. To perform a powerful search, check the ***Only wet cell dry cells*** box. To use a simple search for wet cells in the powerful search for dry cells, ***Limit other searches*** should be left unchecked. To use a simple search for valid cycling data in the powerful search for dry cells, ***Limit other searches*** should be checked together with ***Only wet cell dry cells***.

Figure 5.11 and Figure 5.12 show the results of simple searches for dry cells and dry cell boxes respectively. Figure 5.13 reproduces the same result as Figure 5.11 by using the dry cell box ID input node.

X= Electrolyte Y=Dry Cell/Dry Cell Box	X=Only wet cell electrolytes Y=Only wet cell dry cell	Limit other searches
Simple Search		
Powerful Search (projected from Wet Cell search)	✓	
Powerful Search (projected from Valid Cycling Data)	✓	✓

Table 9 The option combinations to trigger a simple search or a powerful search on electrolytes or dry cells.

Dry Cell	
Display Page:	1 <input type="button" value="Preview Dry Cell"/> <input type="button" value="Preview Box IDs"/>
Box id1:	<input type="text"/>
Box id2:	<input type="text"/>
Box id3:	<input type="text"/>
Box id4:	<input type="text"/>
Box id5:	<input type="text"/>
Notes:	<input type="text"/>
Relative tolerance:	5.0 <input type="text"/> the default tolerance in percentage.
Proprietary:	<input type="checkbox"/>
Geometry category:	<input type="text" value="pouch"/> <input type="text" value="cylinder"/> <input type="text" value="stack"/> <input type="text" value="coin"/>
Geometry category exclude missing:	<input type="checkbox"/>
Cathode:	<input type="text" value="A[LMO] + INACTIVES:(1%B)"/> <input type="text" value="D5 E4 F[LMO]"/> <input type="text" value="D4 E3 F3[LMO, COAT=H]"/> <input type="text" value="D4 E3 F3[LMO, COAT=G]"/> <input type="text" value="D E F[LMO, CRYSTAL=Single]"/>
Cathode exclude missing:	<input checked="" type="checkbox"/>
Anode:	<input type="text" value="[Stone, NAT]"/> <input type="text" value="[Stone, ART]"/>
Anode exclude missing:	<input checked="" type="checkbox"/>
Separator:	<input type="text" value="SENSITIVE CONTENT"/> <input type="text" value="SENSITIVE CONTENT"/> <input type="text" value="SENSITIVE CONTENT"/>
Separator exclude missing:	<input type="checkbox"/>
Only wet cell dry cells:	<input type="checkbox"/>

Figure 5.10 Zoomed-in view of the dry cell section of the search page, with an example input for the dry cell substructure, as it would appear just before the search. This input corresponds to a dry cell either in with a pouch, coin, or missing geometry category, as well as the cathode D5 E4 F[LMO] and anode [Stone, ART]. Any separator as well as a missing separator would also be acceptable.

Dry Cell	
Display Page:	1 <input type="button" value="Preview Dry Cell"/> <input type="button" value="Preview Box IDs"/> Showing page 1/1.
Preview Dry Cells	
[CATH= D5 E4 F[LMO],ANOD=[Stone, ART]]	
Box id1:	<input type="text"/>
Box id2:	<input type="text"/>
Box id3:	<input type="text"/>
Box id4:	<input type="text"/>
Box id5:	<input type="text"/>
Notes:	<input type="text"/>
Relative tolerance:	5.0 <input type="button" value="v"/> the default tolerance in percentage.
Proprietary:	<input type="checkbox"/>
Geometry category:	<input type="text" value="pouch"/> <input type="text" value="cylinder"/> <input type="text" value="stack"/> <input type="text" value="coin"/>
Geometry category exclude missing:	<input type="checkbox"/>
Cathode:	<div style="border: 1px solid gray; padding: 5px;"> <p>A[LMO] + INACTIVES:(1%B)</p> <p style="background-color: #0070C0; color: white; text-align: center;">D5 E4 F[LMO]</p> <p>D4 E3 F3[LMO, COAT=H]</p> <p>D4 E3 F3[LMO, COAT=G]</p> <p>D E F[LMO, CRYSTAL=Single]</p> </div>
Cathode exclude missing:	<input checked="" type="checkbox"/>
Anode:	<div style="border: 1px solid gray; padding: 5px;"> <p>[Stone, NAT]</p> <p style="background-color: #0070C0; color: white; text-align: center;">[Stone, ART]</p> </div>

Figure 5.11 Results of a simple search for dry cells based on the input of Figure 5.10.

Simple Dry Cell Box Search

Dry Cell						
Display Page:	1 <input type="button" value="Preview Dry Cell"/> <input type="button" value="Preview Box IDs"/> Showing page 1/1.					
Preview Box IDs						
50000([CATH=D5 E4 F[LMO],ANOD=[Stone, ART]])						
Box id1:	<input type="text"/>					
Box id2:	<input type="text"/>					
Box id3:	<input type="text"/>					
Box id4:	<input type="text"/>					
Box id5:	<input type="text"/>					
Notes:	<input type="text"/>					
Relative tolerance:	5.0 <input type="button" value="the default tolerance in percentage."/> <input type="button" value="the default tolerance in percentage."/>					
Proprietary:	<input type="checkbox"/>					
Geometry category:	<input type="button" value="pouch"/> <input type="button" value="cylinder"/> <input type="button" value="stack"/> <input type="button" value="coin"/>					
Geometry category exclude missing:	<input type="checkbox"/>					
Cathode:	<table border="1"> <tr><td>A[LMO] + INACTIVES:(1%B)</td></tr> <tr><td>D5 E4 F[LMO]</td></tr> <tr><td>D4 E3 F3[LMO, COAT=H]</td></tr> <tr><td>D4 E3 F3[LMO, COAT=G]</td></tr> <tr><td>D E F[LMO, CRYSTAL=Single]</td></tr> </table>	A[LMO] + INACTIVES:(1%B)	D5 E4 F[LMO]	D4 E3 F3[LMO, COAT=H]	D4 E3 F3[LMO, COAT=G]	D E F[LMO, CRYSTAL=Single]
A[LMO] + INACTIVES:(1%B)						
D5 E4 F[LMO]						
D4 E3 F3[LMO, COAT=H]						
D4 E3 F3[LMO, COAT=G]						
D E F[LMO, CRYSTAL=Single]						
Cathode exclude missing:	<input checked="" type="checkbox"/>					
Anode:	<table border="1"> <tr><td>[Stone, NAT]</td></tr> <tr><td>[Stone, ART]</td></tr> </table>	[Stone, NAT]	[Stone, ART]			
[Stone, NAT]						
[Stone, ART]						
Anode:						

Figure 5.12 Results of a simple search for dry cell boxes based on the input of Figure 5.10.

Powerful Dry Cell Search using a Simple Dry Cell Box Search

Dry Cell	
Display Page:	1 <input type="button" value="Preview Dry Cell"/> <input type="button" value="Preview Box IDs"/> Showing page 1/1.
Preview Dry Cells	
[CATH= D5 E4 F[LMO],ANOD=[Stone, ART]]	
Box id1:	50000 <input type="text"/>
Box id2:	<input type="text"/>
Box id3:	<input type="text"/>
Box id4:	<input type="text"/>
Box id5:	<input type="text"/>
Notes:	<input type="text"/>
Relative tolerance:	5.0 <input type="text"/> the default tolerance in percentage.
Proprietary:	<input type="checkbox"/>
Geometry category:	<input type="text" value="pouch"/> <input type="text" value="cylinder"/> <input type="text" value="stack"/> <input type="text" value="coin"/>
Geometry category exclude missing:	<input type="checkbox"/>
Cathode:	A[LMO] + INACTIVES:(1%B) D5 E4 F[LMO] D4 E3 F3[LMO, COAT=H] D4 E3 F3[LMO, COAT=G] D E F[LMO, CRYSTAL=Single]
Cathode exclude missing:	<input type="checkbox"/>
Anode:	[Stone, NAT] [Stone, ART]
Anode	

Figure 5.13 Results of a powerful search for dry cells based on the dry cell box ID input node.

The results are equivalent to Figure 5.11 despite having a trivial dry cell substructure input.

Simple Wet Cell Search

Figure 5.14 shows how simple searches for wet cells can combine elements of electrolyte searches and dry cell boxes searches, with Figure 5.15 showing the results.

Wet Cells

Display Page: 1 Show Visuals: Preview Wet Cell Preview Validated Cycling Data

Keyword in Value to search		
Keyword in Filename		
Keyword in Filename		
Keyword in Filename		
Keyword in Root		
Keyword in Root		
Keyword in Root		
Exact Match		
Char ID	Minimum	Maximum
Cell ID		
Upper Cutoff Voltage		
Temperature		
Date		
Dataset		

Limit other searches:

Electrolyte

Display Page: 1 Preview Electrolyte

Complete salt:	<input type="checkbox"/>
Complete solvent:	<input type="checkbox"/>
Complete additive:	<input type="checkbox"/>
Relative tolerance:	5.0 <input type="text"/>
the default tolerance in percentage.	
Proprietary flag:	<input type="checkbox"/>
Notes1:	
Notes2:	
Notes3:	
Only wet cell electrolytes:	<input type="checkbox"/>

Molecule	Must have?	Amount	Tolerance
A	mandatory <input type="checkbox"/>	2	
B	mandatory <input type="checkbox"/>	1	
	mandatory <input type="checkbox"/>		
	mandatory <input type="checkbox"/>		
	mandatory <input type="checkbox"/>		
	mandatory <input type="checkbox"/>		

Dry Cell

Display Page: 1

Box id1:	50000
Box id2:	
Box id3:	
Box id4:	
Box id5:	
Notes:	
Relative tolerance:	5.0
the default tolerance:	
Proprietary:	<input type="checkbox"/>
Geometry category:	pouch
Geometry category:	cylinder
Geometry category:	stack
Geometry category:	coin
Geometry category exclude missing:	<input type="checkbox"/>

Figure 5.14 View of the search page, with an example input for the dry cell box ID and the electrolyte substructure similar to Figure 5.13 and Figure 5.8 respectively.

Wet Cells

Display Page: 1 Show Visuals: Preview Wet Cell Preview Validated Cycling Data

Showing page 1/1. Target Dataset: ----- Register to Dataset

Wet cells:

CELL ID: 10001, ELECTROLYTE: 85%D+15%C + 0.5mPepper + ADDITIVES:(2%A+1%B), DRY CELL: 50000([CATH= D5 E4 F[LMO],ANOD=[Stone, ART]])

CELL ID: 10011, ELECTROLYTE: 50%D+50%E + 0.1mSalt + ADDITIVES:(2%A+1%B+1%C), DRY CELL: 50000([CATH= D5 E4 F[LMO],ANOD=[Stone, ART]])

CELL ID: 10101, ELECTROLYTE: 50%D+25%E+25%F + 0.1mSalt + ADDITIVES:(5%C+2%A+1%B), DRY CELL: 50000([CATH= D5 E4 F[LMO],ANOD=[Stone, AR

Keyword in Filename	Value to search			Electrolyte Display Page: 1 Preview Electrolyte Complete salt: <input type="checkbox"/> Complete solvent: <input type="checkbox"/> Complete additive: <input type="checkbox"/> Relative tolerance: 5.0 the default tolerance in percentage. Proprietary flag: <input type="checkbox"/> Notes1: <input type="text"/> Notes2: <input type="text"/> Notes3: <input type="text"/> Only wet cell electrolytes: <input type="checkbox"/>	Dry Cell Display Page: 1 Box id1: 50000 Box id2: <input type="text"/> Box id3: <input type="text"/> Box id4: <input type="text"/> Box id5: <input type="text"/> Notes: <input type="text"/> Relative tolerance: 5.0 the def Proprietary: <input type="checkbox"/> Geometry category: pouch cylinder stack coin Geometry: <input type="text"/>
Keyword in Filename	Exact Match	Minimum	Maximum		
Keyword in Root	Char ID	Cell ID			
Keyword in Root	Upper Cutoff Voltage				
Keyword in Root	Temperature				
	Molecule	Must have?	Amount		
	A	mandatory	2		
	B	mandatory	1		

Figure 5.15 Results of a simple search for wet cells based on the inputs of Figure 5.14.

Dataset Creation from Wet Cell Search

Figure 5.16 shows how to add cells to a dataset once results of a search for wet cells have been produced.

Wet Cells

Display Page: 1 Show Visuals: Preview Wet Cell Preview Validated Cycling Data

Showing page 1/1. Target Dataset: Dataset 1 Register to Dataset

CELL ID: 10001, EL	%D+15%C + 0.5mPepper + ADDITIVES:(2%A+1%B), DRY CELL: 50000([CATH= D5 E4 F[LMO],ANOD=[Stone, ART]])
CELL ID: 10011, EL	%D+50%E + 0.1mSalt + ADDITIVES:(2%A+1%B+1%C), DRY CELL: 50000([CATH= D5 E4 F[LMO],ANOD=[Stone, ART]])
CELL ID: 10101, ELECTROLYTE: 50%D+25%E+25%F + 0.1mSalt + ADDITIVES:(5%C+2%A+1%B), DRY CELL: 50000([CATH= D5 E4 F[LMO],ANOD=[Stone, AR	

Wet cells:

Keyword in Filename	Value to search			Electrolyte Display Page: 1 Preview Electrolyte Complete salt: <input type="checkbox"/> Complete solvent: <input type="checkbox"/> Complete additive: <input type="checkbox"/> Relative tolerance: 5.0 (the default tolerance in percentage) Proprietary flag: <input type="checkbox"/> Notes1: Notes2: Notes3: Only wet cell electrolytes: <input type="checkbox"/>	Dry Cell Display Page: 1 Box id1: 50000 Box id2: Box id3: Box id4: Box id5: Notes: Relative tolerance: 5.0 (the def Proprietary: <input type="checkbox"/> Geometry category: pouch cylinder stack coin Geometry:												
Keyword in Filename																	
Keyword in Filename																	
Keyword in Root																	
Keyword in Root																	
Keyword in Root																	
Keyword in Root																	
Char ID	Exact Match	Minimum	Maximum														
Cell ID																	
Upper Cutoff Voltage																	
Temperature																	
<table border="1"> <thead> <tr> <th>Molecule</th> <th>Must have?</th> <th>Amount</th> <th>Tolerance</th> </tr> </thead> <tbody> <tr> <td>A</td> <td>mandatory</td> <td>2</td> <td></td> </tr> <tr> <td>B</td> <td>mandatory</td> <td>1</td> <td></td> </tr> </tbody> </table>				Molecule	Must have?	Amount	Tolerance	A	mandatory	2		B	mandatory	1			
Molecule	Must have?	Amount	Tolerance														
A	mandatory	2															
B	mandatory	1															

Figure 5.16 Zoomed-in view of adding some cells to a dataset, from the outputs of the search in Figure 5.15. In this case, clicking on “Register to Dataset” would ensure that cells 10001 and 10101 would be part of dataset “Testing”.

Simple Valid Cycling Data Search

Generally, a simple search for a given node performed based on an *input node* not directly connected to the *searched for node* will rely on the existence of all the nodes on the path connecting the *input node* to the *searched for node*. However, there are two exceptions to this rule. First, even if a dry cell box ID does not exist for a given dry cell, the system could still find a wet cell based

on the dry cell substructure if a wet cell exists with the “default” dry cell box ID (see Section 5.3.1).

The other exception is the case where a valid cycling data exists, with the corresponding valid file metadata mentioning a cell ID, but no wet cell registered with that cell ID. In the case of a simple search for the valid cycling data based only on the *given cell ID*, the system would still return the valid cycling data. In this way, a search for wet cells based on cell ID will return existing registered wet cells even if valid cycling data is missing and vice versa.

Figure 17 shows a search for valid cycling data based on experimental conditions and filename substructure. Figure 18 shows the result.

Wet Cells

Display Page: 1 Show Visuals: Preview Wet Cell

	Value to search		
Keyword in Filename	<input type="text"/>		
Keyword in Filename	<input type="text"/>		
Keyword in Filename	<input type="text"/>		
Keyword in Root	<input type="text"/>		
Keyword in Root	<input type="text"/>		
Keyword in Root	<input type="text"/>		
	Exact Match	Minimum	Maximum
Char ID	sb <input type="text"/>	<input type="text"/>	<input type="text"/>
Cell ID	<input type="text"/>	<input type="text"/>	<input type="text"/>
Upper Cutoff Voltage	<input type="text"/>	<input type="text"/>	<input type="text"/>
Temperature	30 <input type="text"/>	<input type="text"/>	<input type="text"/>
Date	-----	-----	-----
Dataset	-----	-----	-----

Limit other searches:

Figure 5.17 A search for valid cycling data based on experimental conditions and filename substructure.

Wet Cells

Display Page: 1 Preview Wet Cell

Showing page 1/1.

Call ID: 10001 Exclude Number of Active Files: 1 Number of Imported But Deprecated Files: 0 Number which Needs Importing: 1 **First Active File**: Sb_CYC_10001_nw_C2_41V_30C_1C1C_170201.b

Trigger Re-Importing

	Value to search		Minimum	Maximum
Keyword in Filename				
Keyword in Filename				
Keyword in Filename				
Keyword in Root				
Keyword in Root				
Keyword in Root				
Char ID	Sb	Exact Match		
Cell ID				
Upper Cutoff Voltage				
Temperature	30			
Date				
Dataset				

Limit other searches:

Electrolyte

Display Page: 1

Complete salt:

Complete solvent:

Complete additive:

Relative tolerance: 5.0 the default tolerance in percent

Proprietary flag:

Notes1:

Notes2:

Notes3:

Only wet cell electrolytes:

Molecule:

Figure 5.18 The results of the search from Figure 5.17.

Wet Cells

Display Page: 1 | Show Visuals: Preview Wet Cell | Preview Validated Cycling Data

Showing page 1/1

Cell ID	Exclude	Number of Active Files	Number of Imported But Deprecated Files	Number which Needs Importing	First Active File
10001	<input checked="" type="checkbox"/>	1	0	1	Sb_CVC_10001_nw_c2_41V_30C_1C1C_170

Trigger Re-Importing

Value to search		Minimum		Maximum	
Keyword in Filename					
Keyword in Filename					
Keyword in Filename					
Keyword in Root					
Keyword in Root					
Keyword in Root					
Char ID	sb				
Cell ID					
Upper Cutoff Voltage					
Temperature	30				
Date					
Dataset					

Initial other searches:

Electrolyte

Display Page: 1 | Preview Electrolyte

Complete salt:	<input type="checkbox"/>
Complete solvent:	<input type="checkbox"/>
Complete additive:	<input type="checkbox"/>
Relative tolerance:	5.0
Proprietary flag:	<input type="checkbox"/> the default tolerance in p
Note1:	
Note2:	
Note3:	
Only wet cell electrolytes:	<input type="checkbox"/>
Molecule	
Must	mandat

Figure 5.19 The results of the search from Figure 5.17 with the “Show Visuals” option checked.

Figure 5.19 shows the visualization of the results of the search from Figure 17. This visualization is obtained by checking the “Show visuals” box. In this case, the cell experienced two groups of discharge. In grey are the cycles with slow discharge and in yellow are the cycles with faster discharge. The legend has been masked. This cell only has a single file active, and the visualization shows a colored bar coextensive with the cycles within that file (also masked).

Valid Cycling Data Re-importing

Figure 5.20 shows how to force the system to reimport the data contained in the files with valid file metadata associated with the given cell IDs. This data is automatically updated every day for all files with valid file metadata.

Wet Cells

Display Page: 1 | Show Visuals: Preview Wet Cell | Preview Validated Cycling Data

Showing page 1/1.

Cell ID	Exclude	Number of Active Files	Number of Imported But Deprecated Files	Number which Needs Importing	First Act
10001	<input type="checkbox"/>	1	0	1	Sb_CYC_10001_nW_c2_41V_30C_1C1C_1702

Trigger Re-Importing

Value to search

Keyword in Filename	Keyword in Filename	Keyword in Filename	Keyword in Root	Keyword in Root	Keyword in Root	Exact Match	Minimum	Maximum
Char ID	sb							
Call ID								
Upper Cutoff Voltage								
Temperature	30							
Date								
Dataset								

Electrolyte

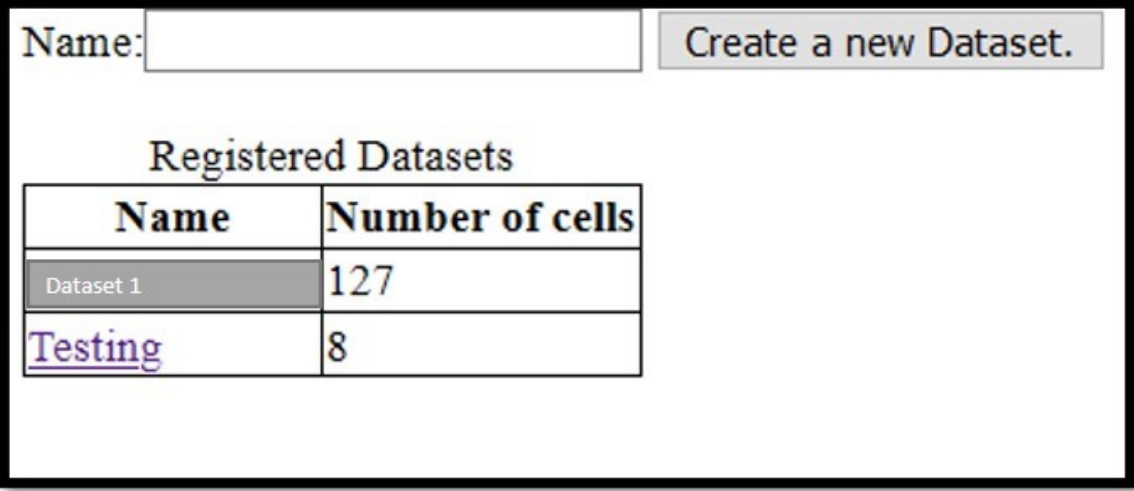
Display Page: 1 | Preview Elect

Complete salt:	<input type="checkbox"/>
Complete solvent:	<input type="checkbox"/>
Complete additive:	<input type="checkbox"/>
Relative tolerance:	5.0 <input type="text"/>
Proprietary flag:	<input type="checkbox"/>
Notes1:	<input type="text"/>
Notes2:	<input type="text"/>
Notes3:	<input type="text"/>
Only wet cell electrolytes:	<input type="checkbox"/>
Molecule	mandarin

Figure 5.20 The steps required to force the system to reimport the files associated with given cells. Starting with Figure 5.19 (or Figure 5.18), first uncheck all the exclude boxes for the cells to be reimported, then click on the “Trigger Re-Importing” button.

5.4.3 Datasets

By following the “Datasets” hyperlink at the top of any page, one is taken to the overview of all datasets. Figure 5.21 shows the overview of all datasets. This is where one can create a new dataset or click on an existing dataset to view it.



The screenshot shows a web interface for managing datasets. At the top, there is a text input field labeled "Name:" and a button labeled "Create a new Dataset.". Below this, the heading "Registered Datasets" is centered. Underneath the heading is a table with two columns: "Name" and "Number of cells". The table contains three rows: a header row, a row for "Dataset 1" with 127 cells, and a row for "Testing" with 8 cells. The "Testing" name is underlined and blue, indicating it is a hyperlink.

Name	Number of cells
Dataset 1	127
Testing	8

Figure 5.21 The overview of all datasets.

Create

a)

Name: Create a new Dataset.

Registered Datasets

Name	Number of cells
Dataset 1	127
Testing	8

b)

Name: Create a new Dataset.

Registered Datasets

Name	Number of cells
Dataset 1	127
Testing	8

c)

Create a new Dataset.

SUCCESS: created dataset with name Testing version 2

Registered Datasets

Name	Number of cells
Dataset 1	127
Testing	8
Testing version 2	0

Figure 5.22 Creating a new dataset step by step. a) start at the overview. b) enter a new name a click on “Create a new Dataset”, c) the new dataset shows up in the overview and a message of success appears.

Figure 5.22 shows the creation of a new dataset, simply by entering a new name. After the dataset is created it will be empty. To add Wet Cells to the dataset, see Section 5.4.2 (Dataset Creation from Wet Cell Search).

View

Dataset: Testing (Visualize)				
Page number: 1 Per page: 10 List New Page				
Showing page 1/1.				
Cell	Specific Name	Default Position	Default Color	Filters
CELL ID: 10001, ELECTROLYTE: 85%D+15%C + 0.5mPepper + ADDITIVES:(2%A+1%B), DRY CELL: 50000([CATH= D5 E4 F[LMO],ANOD=[Stone, ART]])	No specific name	(1,1)		All pos=(1,1)

Figure 5.23 The listing of an example dataset.

To view a dataset, click on it on the overview of all datasets. Then the wet cells along with their description is visible. Figure 5.23 shows the listing of an example dataset.

Annotate

By clicking on “View”, one is taken to the edit page for that dataset. Figure 5.24 shows the default view of the edit page. Figure 5.25 shows how to change the name of a cell in the context of the dataset. Figure 5.26 shows how the edit page changes with the new name.

Dataset: Testing (List)

Specific Names Change Specific Name Specific Name: Wet Cell: 10001 Name: <input type="text"/> Remove from Dataset: <input type="checkbox"/> Reset defaults: <input type="checkbox"/> Plot Settings: Color: <input type="color"/> Grid Position: (1, 1)		Filters Change Filter Filter: Wet Cell: 10001 10101 20001 Name: <input type="text"/> Remove from Cell: <input type="checkbox"/> Plot Settings: Color: <input type="color"/> Grid Position: (1, 1)	
Dataset Preview and Immediate Export Plot Dataset Export Dataset		Rule Match None: <input type="checkbox"/> Charge: <input type="checkbox"/> Attribute: (Min, Max) Constant Rate (This Step): <input type="text"/> <input type="text"/> End Rate (This Step): <input type="text"/> <input type="text"/> End Rate (Previous Step): <input type="text"/> <input type="text"/> End Voltage (This Step): <input type="text"/> <input type="text"/> End Voltage (Previous Step): <input type="text"/> <input type="text"/>	
Individual Cells Preview Display Page: 1 Cells per Page: 25 Cells per Row: 5 Plot Cels			

Figure 5.24 The default edit page for an example dataset.

Dataset: Testing (List)

Specific Names Change Specific Name Specific Name: Wet Cell: 10001 Name: TOM-5 Remove from Dataset: <input type="checkbox"/> Reset defaults: <input type="checkbox"/> Plot Settings: Color: <input type="color"/> Grid Position: (1, 1)		Filters Change Filter Filter: Wet Cell: 10001 10101 20001 Name: <input type="text"/> Remove from Cell: <input type="checkbox"/> Plot Settings: Color: <input type="color"/> Grid Position: (1, 1)	
Dataset Preview and Immediate Export Plot Dataset Export Dataset		Rule Match None: <input type="checkbox"/> Charge: <input type="checkbox"/> Attribute: (Min, Max) Constant Rate (This Step): <input type="text"/> <input type="text"/> End Rate (This Step): <input type="text"/> <input type="text"/> End Rate (Previous Step): <input type="text"/> <input type="text"/> End Voltage (This Step): <input type="text"/> <input type="text"/> End Voltage (Previous Step): <input type="text"/> <input type="text"/>	
Individual Cells Preview Display Page: 1 Cells per Page: 25 Cells per Row: 5 Plot Cels			

Figure 5.25 How to give a specific name to a cell in the context of a dataset. First, type the name in the box circled in red. Finally, click “Change Specific Name”.

Dataset: Testing (List)

Specific Names Change Specific Name Specific Name: Wet Cell: 10001: (TOM-5) Name: <input type="text"/> Remove from Dataset: <input type="checkbox"/> Reset defaults: <input type="checkbox"/>		Filters Change Filter Filter: 10001: (TOM-5) Wet Cell: 10101 Remove from Cell: <input type="checkbox"/>																	
Plot Settings Color: <input type="text"/> (1 <input type="text"/> , 1 <input type="text"/>) Grid Position: <input type="text"/>		Plot Settings Use Cell Defaults: <input type="checkbox"/> Use Cell Defaults: <input type="checkbox"/> Color: <input type="text"/> (1 <input type="text"/> , 1 <input type="text"/>)																	
Dataset Preview and Immediate Export Plot Dataset Export Dataset		Rule <table border="1"> <thead> <tr> <th colspan="2">Charge</th> </tr> </thead> <tbody> <tr> <td>Match None:</td> <td><input type="checkbox"/></td> </tr> <tr> <td>Attribute</td> <td>(Min, Max)</td> </tr> <tr> <td>Constant Rate (This Step)</td> <td><input type="text"/> (<input type="text"/>)</td> </tr> <tr> <td>End Rate (This Step)</td> <td><input type="text"/> (<input type="text"/>)</td> </tr> <tr> <td>End Rate (Previous Step)</td> <td><input type="text"/> (<input type="text"/>)</td> </tr> <tr> <td>End Voltage (This Step)</td> <td><input type="text"/> (<input type="text"/>)</td> </tr> <tr> <td>End Voltage (Previous Step)</td> <td><input type="text"/> (<input type="text"/>)</td> </tr> </tbody> </table>		Charge		Match None:	<input type="checkbox"/>	Attribute	(Min, Max)	Constant Rate (This Step)	<input type="text"/> (<input type="text"/>)	End Rate (This Step)	<input type="text"/> (<input type="text"/>)	End Rate (Previous Step)	<input type="text"/> (<input type="text"/>)	End Voltage (This Step)	<input type="text"/> (<input type="text"/>)	End Voltage (Previous Step)	<input type="text"/> (<input type="text"/>)
Charge																			
Match None:	<input type="checkbox"/>																		
Attribute	(Min, Max)																		
Constant Rate (This Step)	<input type="text"/> (<input type="text"/>)																		
End Rate (This Step)	<input type="text"/> (<input type="text"/>)																		
End Rate (Previous Step)	<input type="text"/> (<input type="text"/>)																		
End Voltage (This Step)	<input type="text"/> (<input type="text"/>)																		
End Voltage (Previous Step)	<input type="text"/> (<input type="text"/>)																		
Individual Cells Preview Display Page: 1 <input type="text"/> Cells per Page: 25 <input type="text"/> Cells per Row: 5 <input type="text"/> Plot Cels																			

Figure 5.26 The edit page after having set a specific name for cell “10001”. Now the specific name is visible in the dropdown menus.

Analyse

By setting filters, it is possible to separate different cycles into different data tables to be outputted. Figure 5.27 shows how to set the most basic filter which encompasses all cycles for all cells.

Dataset: Testing (List)

Specific Names		Filters	
Change Specific Name		Change Filter	
Specific Name	Wet Cell: 10001: (TOM-5) Name: <input type="text"/>	Filter	Wet Cell: 10001: (TOM-5) 10101 Name: Al
Remove from Dataset: <input type="checkbox"/> Reset defaults: <input type="checkbox"/>		Remove from Cell: <input type="checkbox"/>	
Plot Settings	Color <input type="text"/> (1 <input type="text"/>) 1 <input type="text"/>	Plot Settings	Color <input type="text"/> (1 <input type="text"/>) 1 <input type="text"/>
Dataset Preview and Immediate Export		Rule	
Plot Dataset		Charge	
Export Dataset		Match None: <input type="checkbox"/> <input type="checkbox"/>	
Individual Cells Preview		Attribute (Min, Max)	
Display Page: 1 <input type="text"/>		Constant Rate (This Step) <input type="text"/> <input type="text"/> <input type="text"/>	
Cells per Page: 25 <input type="text"/>		End Rate (This Step) <input type="text"/> <input type="text"/> <input type="text"/>	
Cells per Row: 5 <input type="text"/>		End Rate (Previous Step) <input type="text"/> <input type="text"/> <input type="text"/>	
Plot Cels		End Voltage (This Step) <input type="text"/> <input type="text"/> <input type="text"/>	
		End Voltage (Previous Step) <input type="text"/> <input type="text"/> <input type="text"/>	

Figure 5.27 How to set the most inclusive of all filters. When no cell is selected, a filter is created for every cell in the dataset. When no rules are given, the filter will match every valid cycle for a given cell. First, enter a name for the filter in the box circled in red, then click on “Change Filter”.

Output

The filters are used to produce CSV files every night, but in case the data is needed more urgently, it is possible to request an immediate output. Figure 5.28 shows how to request an immediate output of the CSV files based on the filters defined for a given dataset.

Dataset: Testing (List)

Specific Names Change Specific Name Specific Name: Wet Cell: 10001: (TOM-5) Name: <input type="text"/> Remove from Dataset: <input type="checkbox"/> Reset defaults: <input type="checkbox"/>		Filters Change Filter Filter: 10001: (TOM-5) Wet Cell: 10101 Remove from Cell: <input type="checkbox"/> Name: <input type="text"/>																	
Plot Settings Color: <input type="text"/> (1 <input type="text"/>) Grid Position: <input type="text"/> (1 <input type="text"/>)		Plot Settings Use Cell Defaults: <input type="checkbox"/> Use Cell Defaults: <input type="checkbox"/> Color: <input type="text"/> (1 <input type="text"/>) Grid Position: <input type="text"/> (1 <input type="text"/>)																	
Dataset Preview and Immediate Export Plot Dataset Export Dataset		Rule <table border="1"> <thead> <tr> <th colspan="2">Charge</th> </tr> </thead> <tbody> <tr> <td>Match None:</td> <td><input type="checkbox"/></td> </tr> <tr> <td>Attribute</td> <td>(Min, Max)</td> </tr> <tr> <td>Constant Rate (This Step)</td> <td><input type="text"/> (<input type="text"/>)</td> </tr> <tr> <td>End Rate (This Step)</td> <td><input type="text"/> (<input type="text"/>)</td> </tr> <tr> <td>End Rate (Previous Step)</td> <td><input type="text"/> (<input type="text"/>)</td> </tr> <tr> <td>End Voltage (This Step)</td> <td><input type="text"/> (<input type="text"/>)</td> </tr> <tr> <td>End Voltage (Previous Step)</td> <td><input type="text"/> (<input type="text"/>)</td> </tr> </tbody> </table>		Charge		Match None:	<input type="checkbox"/>	Attribute	(Min, Max)	Constant Rate (This Step)	<input type="text"/> (<input type="text"/>)	End Rate (This Step)	<input type="text"/> (<input type="text"/>)	End Rate (Previous Step)	<input type="text"/> (<input type="text"/>)	End Voltage (This Step)	<input type="text"/> (<input type="text"/>)	End Voltage (Previous Step)	<input type="text"/> (<input type="text"/>)
Charge																			
Match None:	<input type="checkbox"/>																		
Attribute	(Min, Max)																		
Constant Rate (This Step)	<input type="text"/> (<input type="text"/>)																		
End Rate (This Step)	<input type="text"/> (<input type="text"/>)																		
End Rate (Previous Step)	<input type="text"/> (<input type="text"/>)																		
End Voltage (This Step)	<input type="text"/> (<input type="text"/>)																		
End Voltage (Previous Step)	<input type="text"/> (<input type="text"/>)																		
Individual Cells Preview Display Page: 1 <input type="text"/> Cells per Page: 25 <input type="text"/> Cells per Row: 5 <input type="text"/> Plot Cels																			

Figure 5.28 How to output the dataset immediately to CSV format. Simply click the “Export Dataset” button.

5.4.4 Registering Cells

To register cells, first one must follow the “Register Cells in Bulk (Electrolyte Definition)” hyperlink. Figure 5.29 and Figure 5.30 show the two parts of the process. First, one selects either a range of Cell IDs or a number of Cell IDs, then one selects the appropriate box whence the cells came (each box must be registered separately), then one selects a set of molecules, and chooses both the default amount and whether to use the molecules in an unconventional way. When clicking on “Change Defaults”, an array of values is generated with the defaults prefilled. All these values can be changed, and finally, when clicking on “Register These Cells,” the corresponding Wet Cells will be created. Note however that unless the “Override Existing Registration” option is selected, only Cell IDs which did not already have a Wet Cell will be

created. When the option is selected, the previous Wet Cells are also replaced by newly created ones.

Figure 5.29 Registering Cells (part 1). a) specify a range of Cell IDs, b) select a box, c) choose a set of molecules and default amounts, d) override how a molecule should be used, e) click on “Change Defaults.”

Cell ID	Molecules	Default Amounts	Types
40322	A, E, H, C, D	70, 25, 5, 1.5	Use Default, salt
40323	A, E, H, C, D	70, 25, 5, 1.5	Use Default, salt
40324	A, E, H, C, D	70, 25, 5, 1.5	Use Default, salt
40325	A, E, H, C, D	70, 25, 5, 1.5	Use Default, salt
40326	A, E, H, C, D	70, 25, 5, 1.5	Use Default, salt
40327	A, E, H, C, D	70, 25, 5, 1.5	Use Default, salt
40328	A, E, H, C, D	70, 25, 5, 1.5	Use Default, salt
40329	A, E, H, C, D	70, 25, 5, 1.5	Use Default, salt
40330	A, E, H, C, D	70, 25, 5, 1.5	Use Default, salt

Figure 5.30 Registering Cells (part 2). A table similar to an Excel document is prefilled with the defaults.

5.4.5 Fixing Bad File Metadata and Bad Cycling Data

To access the existing files, whether they have valid file metadata or not, one must follow the “All Files” hyperlink at the top. One is then taken to the page shown in Figure 5.31, which shows how to fix bad file metadata. After checking the “only invalid files” option and clicking on “Search database and fix errors”, one will see a set of rows, each corresponding to a filename (on the left) and the information deduced automatically (on the right). This information can be overridden if inaccurate or entered if missing. To impact the desired files, remove the “Exclude” option for all the desired files, make sure that the metadata has either been corrected, entered properly, or that the file was marked as “Deprecated” which ignores that file henceforth and does not require valid metadata. As a final step, press the “Make Changes” button all the way at the bottom of the screen.

All Files

	Use in search?	Value to search
Keyword in Filename	<input type="checkbox"/>	
Keyword in Filename	<input type="checkbox"/>	
Keyword in Filename	<input type="checkbox"/>	
Keyword in Root	<input type="checkbox"/>	
Keyword in Root	<input type="checkbox"/>	
Keyword in Root	<input type="checkbox"/>	

only valid files	<input type="checkbox"/>
only invalid files	<input checked="" type="checkbox"/>
only deprecated files	<input type="checkbox"/>
only non-deprecated files	<input type="checkbox"/>
Select Experiment Type	neware (cycling) ▾
Display files with a different experiment type	<input type="checkbox"/>

Display Page: 1 Showing page 1/68.

Filename	Exclude	Deprecate	CharID	Cell Id	Start Cycle	Upper Cutoff Voltage	Temperature
Ben_CYC_23200_879812_stitched.txt	<input type="checkbox"/>	<input checked="" type="checkbox"/>	ben	23200	0	4.0	
BLT_CYC_200212_NW_c0_40V_Vague_Description_121.33.12.1_160008_1_1_stitched.txt	<input checked="" type="checkbox"/>	<input type="checkbox"/>	blt	200212	0	4.0	

Figure 5.31 Fixing File Metadata. The “Exclude” box must be unchecked for the system to process a given row. Here the “Deprecate” box has been checked for the first file, which will remove that file from the rest of the database. Both files show up after a search for “only invalid files” because they have some missing data, including “Temperature”.

In order to remove bad cycles, first find a small visual of the desired cell, such as in Figure 5.19, and click on the small image. The image is a hyperlink to the page shown in

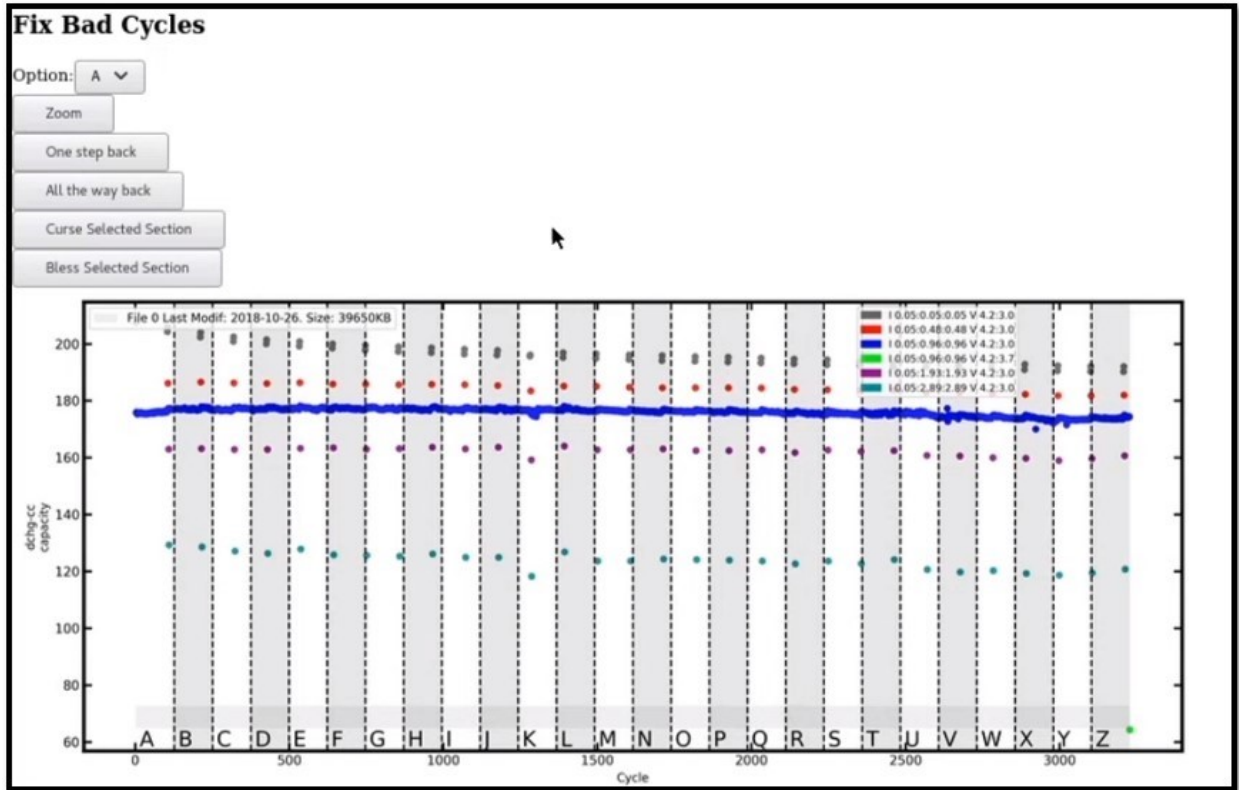


Figure 5.32 The “Fix Bad Cycles” page.

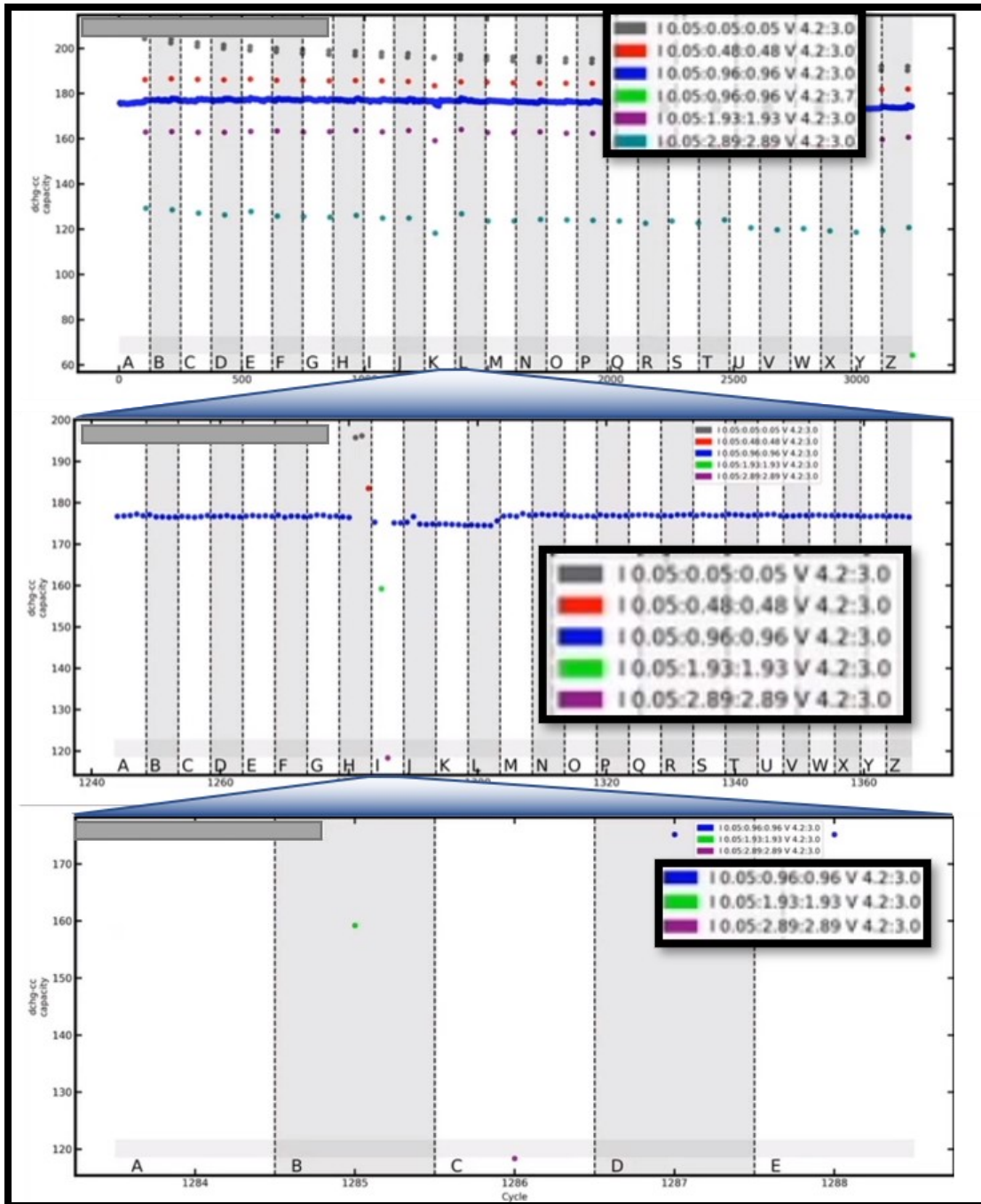


Figure 5.33 Zooming into a given cycle number region. The top panel represents all the cycles. Then, the middle panel only shows the cycles contained within region K in the top panel. Similarly, the bottom panel only shows the cycles within region I in the middle panel. Zoomed-in legends have been added for readability.

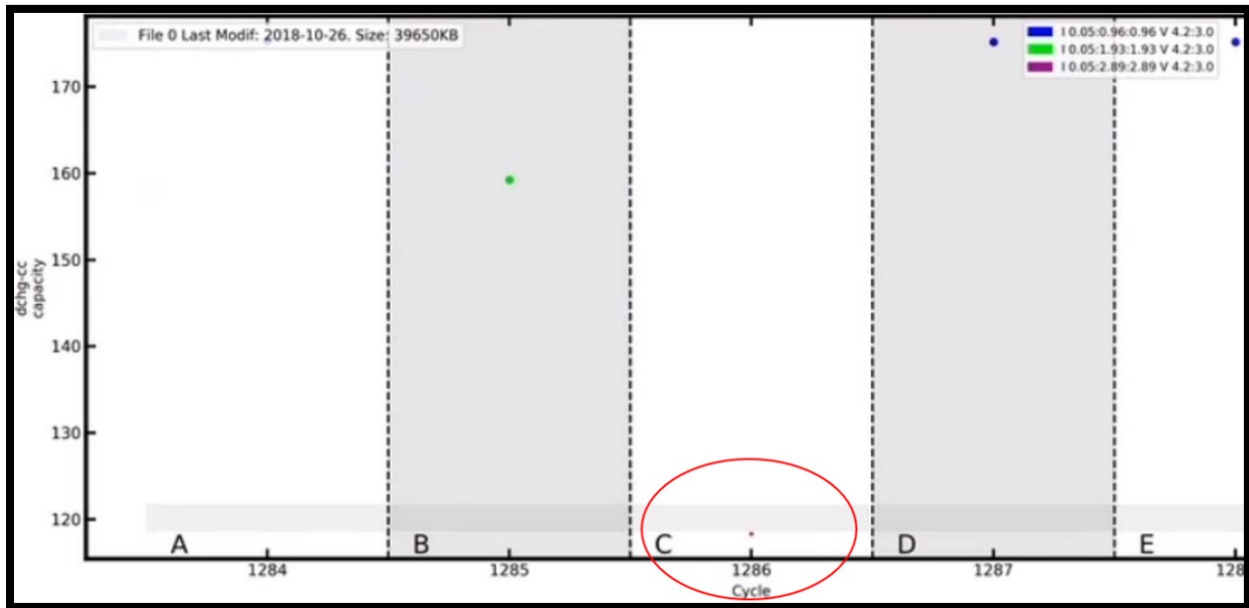


Figure 5.34 View of a cycle after “cursing” region C in Figure 5.33. Inactive cycles are represented visually by smaller dots. The first datapoint is obscured by the top left legend but is visible under the “2” in “2018” on the website itself.

5.5 Future Work

In order for the Universal Battery Database to remain useful, it must be maintained, adapted, and improved as the needs and desired uses change. To perform these operations in the code requires a basic level of proficiency with the tools (Python, Django, HTML, SQL) and it can be hard to find good starting points which are relatively simple but which can improve one’s skills. Therefore, the author has assembled a basic set of improvements which should be accessible to a beginner.

5.5.1 Forms to Formsets

In Django and HTML, the interaction with the software is quite basic and essentially consists of the server sending partially filled *forms* (i.e. the software equivalent of a bureaucratic paper forms,

but this is a technical term found in Django's documentation¹⁰³), and the user completing them, sending them back to the server, and the server reacting to the input.

In cases where information must be specified for a set of entities (i.e. a set of cells in a dataset), it is quite inconvenient for the user to fill a form for each separate entities, but it is a bit simpler to write the software to support the simple case. To help with easier user interaction, there is a concept called *formsets* which simply handles the transition from a single form to a set of identical forms of variable length. With these the idea is to partially fill an array of forms with information, send the array to the user and let the user interact with the formset essentially as they would in a program like *Excel*.

The future work is as follows: look through the website and identify tasks that are currently done with forms but would be faster and nicer to user as formsets, then rewrite the code to implement the change.

This is a nice first future work since there are many examples of both forms and formsets in the website and by studying a few examples, the implementation becomes clear. Furthermore, these two ways of providing interactivity are some of the most documented, often appearing in the first steps of any Django tutorial.

A good first case could be the form to give a specific name to a cell in the context of a specific dataset.

5.5.2 Static Database Information

Most of the work on the Universal Battery Database has been focussed on users actively interacting with the database, since this was the trickiest aspect to implement. However, now that the interaction aspect is in a tolerable state, a lot of value can be added with a lot less work, simply by

displaying data from the database in key places. This is the simplest form of interaction since the server need only perform a query and format the output as HTML instead of worrying about interaction, and is among the first features discussed in the official Django tutorial.

Here are a few examples:

- The database contains a set of molecules which are displayed by an acronym. The database has a description field (and a SMILES¹⁰⁴ code) which can be optionally provided, and it keeps track of which molecule is by default a solvent, a salt, or an additive. It is trivial to create a static page which tabulates all the molecules with their important attributes, and it could provide value when a user is unsure if their “new” molecule has been defined before with a different name. The same goes for electrodes, and other entities, but the set of molecules is always going to be of manageable size and fast to query, so it is the natural place to start.
- The database contains a set of acceptable experiments, equipment, and file formats, which could be displayed as a table, though this would require some care. This would add value by removing the need to maintain a separate rule sheet outside the database. If the rules displayed by the database look right, the database would enforce them. If the separate rule sheet looks right, it has no impact on the database.
- Each valid cycling data has different groups of cycles. When looking at a cell, it would be simple and fast to get that information and display it. Furthermore, in the case of a dataset, it is often useful to know which groups of cycles exist when defining filters.

5.5.3 Anomaly Warnings and Troubleshooting Help

The majority of issues users encounter are quite similar. For instance,

- Visualizing a dataset with cells that do not have valid cycling data.
- Filenames that do not specify the start cycle when it is 0.
- Valid cycling data for a Cell ID exists, but there is no Wet Cell defined for that cell (the cell was not registered).
- An electrolyte does not show up in a search because the correct values of salt, solvent, and additives have not been extracted and are all written as text in the “notes”.

Many of these issues can be proactively detected, while some require the knowledge that there is an issue.

For the proactively identifiable issues, the idea would be to write simple programs which periodically search the database for patterns which resemble typical problems, and lists them so that they can be displayed statically on a “potential problems” page.

For the issues which require knowledge of a problem, a “troubleshoot” page could be created, with different options such as looking for different types of issues on a given Cell ID, Wet Cell, dataset, etc..

5.5.4 Outputting Data in a Complete Format through the User Interface

There is a command line program which takes as input a list of Cell IDs and constructs compiled data in a format suitable for modelling and machine learning (see Section 5.3.8). Yet, there is already a system for organizing Cell IDs, namely the dataset system. A quite simple improvement over the current system is to allow one to select a subset of all datasets in the overview of all datasets page, and click a button called “Export for Modelling”. Behind the scenes, all that would be required would be to query to database for all the wet cells belonging to at least one of the selected datasets, and call the function used in the command line export

program with that list of wet cells (their Cell IDs). This would add value by making it convenient to carefully curate specific datasets to test specific aspects of a model, as well as easily amass a large amount of data meeting quality and diversity requirements.

5.6 Acknowledgements

This work was supported financially by the Natural Sciences and Engineering Research Council of Canada (NSERC), and Tesla Motors.

Chapter 6 Conclusion and Outlook

This thesis aimed at presenting insights into the data-driven development of models and tools (i.e. machine learning) in the context of lithium-ion cell research.

Chapter 1 recapitulated concepts in machine learning. Chapter 2 illustrated how a deep mathematical understanding of an applied problem could be sought. Chapter 3 then illustrated how such understanding could be leveraged to create a robust solution with machine learning techniques. Chapter 4 illustrated how the failures of a machine learning system can be made benign and its workings interpretable when the problem setting does not admit of a perfect solution. Finally, Chapter 5 illustrated how data processing systems could be built to benefit the daily operation of a laboratory and increase the scale of feasible machine learning projects.

Whenever possible, the author sought to transmit the intuition acquired during his exploration of the topic. This was done in the text by capturing arguments and reasoning about machine learning system performances at the level of rigor the author used to guide design decisions. Such arguments represent near-immediate intuitions to guide exploration and therefore are not validated and corrected as thoroughly as proofs would be since their use can tolerate a degree of inaccuracy.

As a reminder of the principle used to develop such intuition: **when modifying a machine learning design and running a numerical experiment, seek not to increase the performance metrics at all cost; instead, seek to be the least confused by the differences in performance between various alternative designs, and when a change in performance is confusing, strive to understand it. In the author's experience, not only does this lead to much simpler and bug-**

free code, but it also is the most powerful way to improve one's intuition on the relevant problems one is tasked to solve.

The set of insights for the application of machine learning techniques to relevant problems in lithium-ion research and science more broadly is much larger than a single thesis and it evolves as people gain more experience and reflect more on the topic. Therefore, this whole thesis was only a preliminary introduction to this set of insights. Hopefully, the future work sections will be useful to the next travellers on this path.

Furthermore, the applications discussed were those that seemed most relevant and achievable at the time the author studied in the lab. Namely, Chapter 2 and Chapter 3 focus on electrochemical impedance spectrometry, Chapter 4 focusses on fourier transform infrared spectrometry, and Chapter 5 focussed on a database for long-term cycling data and structural description of lithium-ion cells. As time changes, the most promising applications of machine learning will change. Indeed, it is the hope of the author that the systems discussed in Chapter 5 will enable ambitious studies of the forecast of lithium-ion cell degradation to be completed.

Chapter 3 presented a general paradigm to automate the fitting of empirical data to physical models, namely to determine an inverse model parametrized with a deep neural network by directly minimizing the mean squared error of the reconstructed empirical data, with a successful application to EC model fitting of impedance spectra of lithium-ion cells (a failure rate of less than 1% and good fit quality on two large and diverse datasets with a single inverse model and using ADAM to finetune the EC parameters). Crucially, this method does not require knowledge of the true EC parameters corresponding to the empirical data, allowing the use of generated data, as well as any available impedance spectra to train the inverse model. This makes the method easy to implement, as well as being flexible.

This application allowed us to illustrate the process by which deeper understanding of the underlying application domain may be leveraged to produce robust machine learning solutions.

In Chapter 4 we showed how the concentration of electrolyte components in lithium-ion cells can be determined using Fourier transform infrared spectroscopy, Beer's law, and machine learning. A physically grounded model was used. We have also open-sourced a carefully prepared dataset and the code to replicate and extend to different electrolyte mixtures. With this new model, a prediction error of around 1-2% for the LiPF₆-to-total mass ratio, as well as all the linear carbonates, and around 2-3% for ethylene carbonate is achieved. Furthermore, this model allows for a useful reconstruction of the FTIR spectrum of an unknown sample. This allows the user to detect samples significantly different from those in the dataset (e.g. due to a bad measurement, to a significant amount of a different molecule, or to a significant change in apparatus). Furthermore, the model is data efficient such that a model for mixtures of 5 components can be calibrated well with less than 50 carefully prepared samples. Therefore, it should be possible to easily extend this work to other systems.

This work refines and generalizes our previous work⁹³ and improves the physical underpinnings of the model. The composition of unknown electrolyte samples, with a specified set of components, can be well determined using inexpensive and rapid measurements with attenuated total reflectance fourier transform infrared spectroscopy.

The code is available at https://github.com/Samuel-Buteau/Electrolyte_Analysis_FTIR, with all the documentation in the README.md file.

In Chapter 5 we documented a working battery database deployed on more than 20000 long-term cycling experiments. We described a structure of cell metadata allowing intricate searches and

unique naming at scale. We described how experiments spread across multiple files could be reliably gathered, searched, visualized, organized in datasets, etc.. We also explained how the search was implemented, how it could be used, how new cells could be added to the system, and how inaccurate data could be fixed within the system.

Bibliography

1. Nelson K. STUDIES OF THE EFFECTS OF HIGH VOLTAGE ON THE PERFORMANCE AND IMPEDANCE OF LITHIUM-ION BATTERIES. Published online December 11, 2017. Accessed January 25, 2019. <https://DalSpace.library.dal.ca/handle/10222/73493>
2. Ellis L. Probing the Causes and Effects of Parasitic Reactions in Lithium-Ion Cells. Published online 2018. Accessed August 24, 2020. <http://hdl.handle.net/10222/74117>
3. Li H. STUDIES OF Ni-RICH POSITIVE ELECTRODE MATERIALS FOR LITHIUM ION BATTERIES. Published online 2020. Accessed August 24, 2020. <http://hdl.handle.net/10222/77688>
4. Weber R. Advanced Materials for Lithium Batteries. Published online 2020. Accessed August 24, 2020. <http://hdl.handle.net/10222/78832>
5. Glazier S. ISOTHERMAL MICROCALORIMETRY AS A TOOL TO PROBE PARASITIC REACTIONS IN LITHIUM-ION CELLS. Published online 2019. Accessed August 24, 2020. <http://hdl.handle.net/10222/76690>
6. Louli AJ. Probing the Reversible and Irreversible Volume Expansion Observed in Li-Ion Pouch Cells. Published online 2017. Accessed August 24, 2020. <http://hdl.handle.net/10222/73460>
7. Goodfellow I, Bengio Y, Courville A. *Deep Learning*. MIT Press; 2016. Accessed December 19, 2018. <https://www.deeplearningbook.org/>
8. Nielsen MA. Neural Networks and Deep Learning. Published online 2015. Accessed August 21, 2020. <http://neuralnetworksanddeeplearning.com>
9. Dive into Deep Learning — Dive into Deep Learning 0.15.1 documentation. Accessed December 28, 2020. <http://d2l.ai/>
10. Deep Learning with PyTorch: A 60 Minute Blitz — PyTorch Tutorials 1.7.1 documentation. Accessed December 28, 2020. https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html
11. Loss Functions — ML Glossary documentation. Accessed December 28, 2020. https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html
12. Rosasco L, Vito ED, Caponnetto A, Piana M, Verri A. Are Loss Functions All the Same? *Neural Comput.* 2004;16(5):1063-1076. doi:10.1162/089976604773135104
13. Clyde M, Cetinkaya-Rundel M, Rundel C, Banks D, Chai C, Huang L. *Chapter 3 Losses and Decision Making | An Introduction to Bayesian Thinking*. Accessed July 27, 2020. <https://statswithr.github.io/book/losses-and-decision-making.html#loss-functions>

14. Kaplan W. *Advanced Calculus*. Subsequent edition. Addison-Wesley; 1991.
15. Griewank A. Complexity of gradients, Jacobians, and Hessians. In: Floudas CA, Pardalos PM, eds. *Encyclopedia of Optimization*. Springer US; 2009:425-435. doi:10.1007/978-0-387-74759-0_78
16. An Intuitive Introduction to the Hessian for Deep Learning Practitioners | Machine Learning Explained. Published February 2, 2018. Accessed December 28, 2020. <https://mlexplained.com/2018/02/02/an-introduction-to-second-order-optimization-for-deep-learning-practitioners-basic-math-for-deep-learning-part-1/>
17. Bottou L, Curtis FE, Nocedal J. Optimization Methods for Large-Scale Machine Learning. *ArXiv160604838 Cs Math Stat*. Published online February 8, 2018. Accessed December 28, 2020. <http://arxiv.org/abs/1606.04838>
18. Kiefer J, Wolfowitz J. Stochastic Estimation of the Maximum of a Regression Function. *Ann Math Stat*. 1952;23(3):462-466. doi:10.1214/aoms/1177729392
19. Robbins H. A Stochastic Approximation Method. Published online 2007. doi:10.1214/AOMS/1177729586
20. 5 Regression Loss Functions All Machine Learners Should Know | by Prince Grover | Heartbeat. Accessed December 28, 2020. <https://heartbeat.fritz.ai/5-regression-loss-functions-all-machine-learners-should-know-4fb140e9d4b0>
21. Brownlee J. How to Control the Stability of Training Neural Networks With the Batch Size. *Machine Learning Mastery*. Published January 20, 2019. Accessed December 28, 2020. <https://machinelearningmastery.com/how-to-control-the-speed-and-stability-of-training-neural-networks-with-gradient-descent-batch-size/>
22. Kingma DP, Ba J. Adam: A Method for Stochastic Optimization. *ArXiv14126980 Cs*. Published online December 22, 2014. Accessed January 12, 2019. <http://arxiv.org/abs/1412.6980>
23. IPRally blog: Recent improvements to the Adam optimizer. Accessed December 28, 2020. <https://www.iprally.com/news/recent-improvements-to-the-adam-optimizer>
24. Currying - HaskellWiki. Accessed December 28, 2020. <https://wiki.haskell.org/Currying>
25. Czarnecki WM, Osindero S, Jaderberg M, Świrszcz G, Pascanu R. Sobolev Training for Neural Networks. *ArXiv170604859 Cs*. Published online July 26, 2017. Accessed July 27, 2020. <http://arxiv.org/abs/1706.04859>
26. Automatic differentiation. In: *Wikipedia*. ; 2020. Accessed December 28, 2020. https://en.wikipedia.org/w/index.php?title=Automatic_differentiation&oldid=995938170

27. 4. Fully Connected Deep Networks - TensorFlow for Deep Learning [Book]. Accessed December 28, 2020. <https://www.oreilly.com/library/view/tensorflow-for-deep/9781491980446/ch04.html>
28. Nwankpa C, Ijomah W, Gachagan A, Marshall S. Activation Functions: Comparison of trends in Practice and Research for Deep Learning. *ArXiv181103378 Cs*. Published online November 8, 2018. Accessed December 28, 2020. <http://arxiv.org/abs/1811.03378>
29. Rectifier (neural networks). In: *Wikipedia*. ; 2020. Accessed December 28, 2020. [https://en.wikipedia.org/w/index.php?title=Rectifier_\(neural_networks\)&oldid=992119472](https://en.wikipedia.org/w/index.php?title=Rectifier_(neural_networks)&oldid=992119472)
30. Saha S. A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way. Medium. Published December 17, 2018. Accessed December 28, 2020. <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>
31. Brown TB, Mann B, Ryder N, et al. Language Models are Few-Shot Learners. *ArXiv200514165 Cs*. Published online July 22, 2020. Accessed August 14, 2020. <http://arxiv.org/abs/2005.14165>
32. Vaswani A, Shazeer N, Parmar N, et al. Attention Is All You Need. *ArXiv170603762 Cs*. Published online December 5, 2017. Accessed July 27, 2020. <http://arxiv.org/abs/1706.03762>
33. Hu D. An Introductory Survey on Attention Mechanisms in NLP Problems. *ArXiv181105544 Cs Stat*. Published online November 12, 2018. Accessed January 12, 2019. <http://arxiv.org/abs/1811.05544>
34. Wolpert DH. The Lack of A Priori Distinctions Between Learning Algorithms. *Neural Comput*. 1996;8(7):1341-1390. doi:10.1162/neco.1996.8.7.1341
35. Petibon R, Aiken CP, Sinha NN, et al. Study of Electrolyte Additives Using Electrochemical Impedance Spectroscopy on Symmetric Cells. *J Electrochem Soc*. 2013;160(1):A117-A124. doi:10.1149/2.005302jes
36. Orazem ME, Tribollet B, Wiley Online Library (Online service). *Electrochemical Impedance Spectroscopy*. Wiley; 2008. Accessed October 27, 2017. <http://onlinelibrary.wiley.com/book/10.1002/9780470381588>
37. Equivalent Circuits - an overview | ScienceDirect Topics. Accessed December 28, 2020. <https://www.sciencedirect.com/topics/engineering/equivalent-circuits>
38. Buteau S, Dahn DC, Dahn JR. Explicit Conversion between Different Equivalent Circuit Models for Electrochemical Impedance Analysis of Lithium-Ion Cells. *J Electrochem Soc*. 2018;165(2):A228-A234. doi:10.1149/2.0841802jes

39. Buteau S, Dahn JR. Analysis of Thousands of Electrochemical Impedance Spectra of Lithium-Ion Cells through a Machine Learning Inverse Model. *J Electrochem Soc.* 2019;166(8):A1611. doi:10.1149/2.1051908jes
40. Lasia A. Modeling of Experimental Data. In: *Electrochemical Impedance Spectroscopy and Its Applications*. Springer, New York, NY; 2014:301-321. doi:10.1007/978-1-4614-8933-7_14
41. Nelson KJ, d'Eon GL, Wright ATB, Ma L, Xia J, Dahn JR. Studies of the Effect of High Voltage on the Impedance and Cycling Performance of Li[Ni_{0.4}Mn_{0.4}Co_{0.2}]O₂/Graphite Lithium-Ion Pouch Cells. *J Electrochem Soc.* 2015;162(6):A1046-A1054. doi:10.1149/2.0831506jes
42. Dai H, Jiang B, Wei X. Impedance Characterization and Modeling of Lithium-Ion Batteries Considering the Internal Temperature Gradient. *Energies.* 2018;11(1):1-18. Accessed December 19, 2018. <https://ideas.repec.org/a/gam/jeners/v11y2018i1p220-d127410.html>
43. Amey S. 7 - Constrained optimization. Accessed December 28, 2020. <https://economics.uwo.ca/math/resources/calculus-multivariable-functions/7-partial-derivatives-constrained-optimization/content/>
44. Verhulst P-F. Notice sur la loi que la population poursuit dans son accroissement. *Corresp Mathématique Phys.* 1838;10:113-121.
45. Pandey S, Kumar D, Parkash O, Pandey L. Equivalent circuit models using CPE for impedance spectroscopy of electronic ceramics. *Integr Ferroelectr.* 2017;183(1):141-162. doi:10.1080/10584587.2017.1376984
46. The Reparameterization Trick. Accessed December 28, 2020. <http://gregorygundersen.com/blog/2018/04/29/reparameterization/>
47. Dion F, Lasia A. The use of regularization methods in the deconvolution of underlying distributions in electrochemical processes. *J Electroanal Chem.* 1999;475:28-37. doi:10.1016/S0022-0728(99)00334-4
48. Weisstein EW. Hyperbolic Secant. Accessed December 28, 2020. <https://mathworld.wolfram.com/HyperbolicSecant.html>
49. Ng AY. Feature Selection, L1 vs. L2 Regularization, and Rotational Invariance. In: *Proceedings of the Twenty-First International Conference on Machine Learning*. ICML '04. ACM; 2004:78-. doi:10.1145/1015330.1015435
50. Zoltowski P. The power of reparameterization of measurement models in electrochemical impedance spectroscopy. *J Electroanal Chem.* 1997;424(1):173-178. doi:10.1016/S0022-0728(96)04928-5

51. Hooten MB, Leeds WB, Fiechter J, Wikle CK. Assessing First-Order Emulator Inference for Physical Parameters in Nonlinear Mechanistic Models. *J Agric Biol Environ Stat.* 2011;16(4):475-494. Accessed December 13, 2018. <https://www.jstor.org/stable/23238828>
52. Rahimi A, Sapp J, Xu J, Bajorski P, Horacek M, Wang L. Examining the Impact of Prior Models in Transmural Electrophysiological Imaging: A Hierarchical Multiple-Model Bayesian Approach. *IEEE Trans Med Imaging.* 2016;35(1):229-243. doi:10.1109/TMI.2015.2464315
53. Busuioc D. Circuit Model Parameter Extraction and Optimization for Microwave Filters. Published online 2002. Accessed December 13, 2018. <https://uwspace.uwaterloo.ca/handle/10012/804>
54. Jiang J, Lin Z, Ju Q, Ma Z, Zheng C, Wang Z. Electrochemical Impedance Spectra for Lithium-ion Battery Ageing Considering the Rate of Discharge Ability. *Energy Procedia.* 2017;105:844-849. doi:10.1016/j.egypro.2017.03.399
55. Stević Z, Vujasinović MR, Radunović M. Estimation of Parameters Obtained by Electrochemical Impedance Spectroscopy on Systems Containing High Capacities. *Sensors.* 2009;9(9):7365-7373. doi:10.3390/s90907365
56. Buteau S. Finding Hadamard and (epsilon,delta)-Quasi-Hadamard Matrices with Optimization Techniques. Published online 2016. doi:10.20381/ruor-5427
57. Seif G. The 5 Clustering Algorithms Data Scientists Need to Know. Medium. Published December 13, 2020. Accessed December 28, 2020. <https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68>
58. Clustering | Types Of Clustering | Clustering Applications. Accessed December 28, 2020. <https://www.analyticsvidhya.com/blog/2016/11/an-introduction-to-clustering-and-different-methods-of-clustering/>
59. Kawaguchi K, Bengio Y. Depth with nonlinearity creates no bad local minima in ResNets. *Neural Netw.* 2019;118:167-174. doi:10.1016/j.neunet.2019.06.009
60. Kawaguchi K, Huang J. Gradient Descent Finds Global Minima for Generalizable Deep Neural Networks of Practical Sizes. *ArXiv190802419 Cs Math Stat.* Published online June 16, 2020. Accessed August 14, 2020. <http://arxiv.org/abs/1908.02419>
61. The Simple Math of Evolution - LessWrong. Accessed December 28, 2020. <https://www.lesswrong.com/s/MH2b8NfWv22dBtrs8>
62. Bode plot. In: *Wikipedia.* ; 2020. Accessed December 28, 2020. https://en.wikipedia.org/w/index.php?title=Bode_plot&oldid=994123816
63. Prior probability. In: *Wikipedia.* ; 2020. Accessed December 28, 2020. https://en.wikipedia.org/w/index.php?title=Prior_probability&oldid=995025735

64. He K, Zhang X, Ren S, Sun J. Deep Residual Learning for Image Recognition. *ArXiv151203385 Cs*. Published online December 10, 2015. Accessed January 1, 2019. <http://arxiv.org/abs/1512.03385>
65. Bayesian inference. In: *Wikipedia.* ; 2020. Accessed December 28, 2020. https://en.wikipedia.org/w/index.php?title=Bayesian_inference&oldid=995551877
66. Brownlee J. Why Initialize a Neural Network with Random Weights? Machine Learning Mastery. Published July 31, 2018. Accessed December 28, 2020. <https://machinelearningmastery.com/why-initialize-a-neural-network-with-random-weights/>
67. Complexity Penalties in Statistical Learning - LessWrong. Accessed December 28, 2020. <https://www.lesswrong.com/posts/boBSTYL3K4KSbh4ec/complexity-penalties-in-statistical-learning>
68. tf.stop_gradient | TensorFlow Core v2.4.0. Accessed December 28, 2020. https://www.tensorflow.org/api_docs/python/tf/stop_gradient
69. Kendall A, Gal Y, Cipolla R. Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics. In: ; 2018:7482-7491. Accessed December 28, 2020. https://openaccess.thecvf.com/content_cvpr_2018/html/Kendall_Multi-Task_Learning_Using_CVPR_2018_paper.html
70. Sarkar T. Synthetic data generation — a must-have skill for new data scientists. Medium. Published July 18, 2019. Accessed December 28, 2020. <https://towardsdatascience.com/synthetic-data-generation-a-must-have-skill-for-new-data-scientists-915896c0c1ae>
71. Iwana BK, Uchida S. An Empirical Survey of Data Augmentation for Time Series Classification with Neural Networks. *ArXiv200715951 Cs Stat*. Published online July 31, 2020. Accessed December 28, 2020. <http://arxiv.org/abs/2007.15951>
72. Shorten C, Khoshgoftaar TM. A survey on Image Data Augmentation for Deep Learning. *J Big Data*. 2019;6(1):60. doi:10.1186/s40537-019-0197-0
73. Reed S, de Freitas N. Neural Programmer-Interpreters. *ArXiv151106279 Cs*. Published online February 29, 2016. Accessed December 28, 2020. <http://arxiv.org/abs/1511.06279>
74. Zhang Y, Yang Q. A Survey on Multi-Task Learning. *ArXiv170708114 Cs*. Published online July 26, 2018. Accessed December 28, 2020. <http://arxiv.org/abs/1707.08114>
75. Honchar A. Multitask learning: teach your AI more to make it better. Medium. Published December 2, 2018. Accessed December 28, 2020. <https://towardsdatascience.com/multitask-learning-teach-your-ai-more-to-make-it-better-dde116c2cd40>
76. Kocmi T. *Exploring Benefits of Transfer Learning in Neural Machine Translation.*; 2020.

77. Wang Z, Dai Z, Póczos B, Carbonell J. Characterizing and Avoiding Negative Transfer. In: ; 2019:11293-11302. Accessed December 28, 2020. https://openaccess.thecvf.com/content_CVPR_2019/html/Wang_Characterizing_and_Avoiding_Negative_Transfer_CVPR_2019_paper.html
78. Alake R. Understand Local Receptive Fields In Convolutional Neural Networks. Medium. Published June 12, 2020. Accessed December 28, 2020. <https://towardsdatascience.com/understand-local-receptive-fields-in-convolutional-neural-networks-f26d700be16c>
79. Understanding and implementing a fully convolutional network (FCN) | by Himanshu Rawlani | Towards Data Science. Accessed December 28, 2020. <https://towardsdatascience.com/implementing-a-fully-convolutional-network-fcn-in-tensorflow-2-3c46fb61de3b>
80. Piramanayagam S, Saber E, Schwartzkopf W, Koehler F. Supervised Classification of Multisensor Remotely Sensed Images Using a Deep Learning Framework. *Remote Sens.* 2018;10:1429. doi:10.3390/rs10091429
81. Ioffe S. Batch Renormalization: Towards Reducing Minibatch Dependence in Batch-Normalized Models. *ArXiv170203275 Cs*. Published online February 10, 2017. Accessed January 1, 2019. <http://arxiv.org/abs/1702.03275>
82. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J Mach Learn Res.* 2014;15:1929-1958. Accessed January 1, 2019. <http://jmlr.org/papers/v15/srivastava14a.html>
83. [Coding tutorial] Padding and masking sequence data - Sequence Modelling. Coursera. Accessed December 28, 2020. <https://www.coursera.org/lecture/customising-models-tensorflow2/coding-tutorial-padding-and-masking-sequence-data-4cbXR>
84. Brownlee J. A Gentle Introduction to Imbalanced Classification. Machine Learning Mastery. Published December 22, 2019. Accessed December 28, 2020. <https://machinelearningmastery.com/what-is-imbalanced-classification/>
85. Understanding Data Bias. Types and sources of data bias | by Prabhakar Krishnamurthy | Towards Data Science. Accessed December 28, 2020. <https://towardsdatascience.com/survey-d4f168791e57>
86. Rocca B. Handling imbalanced datasets in machine learning. Medium. Published March 30, 2019. Accessed December 28, 2020. <https://towardsdatascience.com/handling-imbalanced-datasets-in-machine-learning-7a0e84220f28>
87. Andrychowicz M, Denil M, Gomez S, et al. Learning to learn by gradient descent by gradient descent. *ArXiv160604474 Cs*. Published online November 30, 2016. Accessed December 28, 2020. <http://arxiv.org/abs/1606.04474>

88. Alammam J. The Illustrated Transformer. Accessed December 28, 2020. <http://jalammar.github.io/illustrated-transformer/>
89. Kazemnejad A. Transformer Architecture: The Positional Encoding. Published September 20, 2019. Accessed August 18, 2020. https://kazemnejad.com/blog/transformer_architecture_positional_encoding/
90. Ke G, He D, Liu T-Y. Rethinking Positional Encoding in Language Pre-training. *ArXiv200615595 Cs*. Published online July 8, 2020. Accessed December 28, 2020. <http://arxiv.org/abs/2006.15595>
91. Einops — a new style of deep learning code. Accessed February 19, 2021. <https://arogozhnikov.github.io/2018/12/06/einops.html>
92. Buteau S, Lee E, Young S, Hames S, Dahn JR. User-Friendly Freeware for Determining the Concentration of Electrolyte Components in Lithium-Ion Cells Using Fourier Transform Infrared Spectroscopy, Beer's Law, and Machine Learning. *J Electrochem Soc*. 2019;166(14):A3102. doi:10.1149/2.0151914jes
93. Ellis LD, Buteau S, Hames SG, Thompson LM, Hall DS, Dahn JR. A New Method for Determining the Concentration of Electrolyte Components in Lithium-Ion Cells, Using Fourier Transform Infrared Spectroscopy and Machine Learning. *J Electrochem Soc*. 2018;165(2):A256-A262. doi:10.1149/2.0861802jes
94. Sahore R, Dogan F, Bloom ID. Identification of Electrolyte-Soluble Organic Cross-Talk Species in a Lithium-Ion Battery via a Two-Compartment Cell. *Chem Mater*. 2019;31(8). doi:10.1021/acs.chemmater.9b00063
95. Thompson LM, Stone W, Eldesoky A, et al. Quantifying Changes to the Electrolyte and Negative Electrode in Aged NMC532/Graphite Lithium-Ion Cells. *J Electrochem Soc*. 2018;165(11):A2732-A2740. doi:10.1149/2.0721811jes
96. Bowman J, Emerson S, Darnovsky M. *The Practical SQL Handbook: Using SQL Variants*. 4th Edition. Addison-Wesley Professional
97. Griffiths PR, De Haseth JA. *Fourier Transform Infrared Spectrometry*. 2nd ed. Wiley; 1986.
98. Goodfellow IJ, Shlens J, Szegedy C. Explaining and Harnessing Adversarial Examples. *ArXiv14126572 Cs Stat*. Published online March 20, 2015. Accessed August 25, 2020. <http://arxiv.org/abs/1412.6572>
99. Du K. Neware BTS9000 – The most sophisticated tester ever. Published January 9, 2019. Accessed August 6, 2020. <https://newarebattery.com/neware-bts9000-the-most-sophisticated-tester-ever/>

100. Fath JP, Alsheimer L, Storch M, et al. The influence of the anode overhang effect on the capacity of lithium-ion cells – a 0D-modeling approach. *J Energy Storage*. 2020;29:101344. doi:10.1016/j.est.2020.101344
101. Gyenes B, Stevens DA, Chevrier VL, Dahn JR. Understanding Anomalous Behavior in Coulombic Efficiency Measurements on Li-Ion Batteries. *J Electrochem Soc*. 2014;162(3):A278. doi:10.1149/2.0191503jes
102. CSV, Comma Separated Values (RFC 4180). Published online November 27, 2012. Accessed August 3, 2020. <https://www.loc.gov/preservation/digital/formats/fdd/fdd000323.shtml>
103. Django Software Foundation. *Django*.; 2020. Accessed July 27, 2020. <https://djangoproject.com>
104. SMILES Tutorial | Research | US EPA. Accessed December 28, 2020. https://archive.epa.gov/med/med_archive_03/web/html/smiles.html