

Hybrid Battery Modelling: Combining Physical Models with Recurrent Neural Networks

By

RUSSELL HAWKINS
BS UC Merced 2013 MS UC Davis 2016

THESIS

Submitted in partial satisfaction of the requirements for the degree of

MASTER OF SCIENCE

in

Mechanical and Aerospace Engineering

in the


OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:



Assistant Professor Xinfan Lin, Chair



Associate Professor Zhaodan Kong

fassadian

Professor Francis Assadian

Committee in Charge

2020

To my parents, whose love and support have made me who I am.

CONTENTS

List of Figures	v
List of Tables	vii
Abstract	viii
Acknowledgments	x
1 Introduction	1
1.1 Background on Lithium-Ion Batteries	1
1.2 Overview of Battery Modelling	3
1.2.1 Applications of Battery Models	3
1.2.2 Physics-based Battery Models	3
1.2.3 Data-Driven Battery Models	5
1.3 The Hybrid Approach	5
1.4 Thesis Organization	6
2 Physical Battery Models	8
2.1 Open Circuit Voltage Model	8
2.2 Pseudo-2-Dimensional Model	9
2.3 Single Particle Model	10
3 Recurrent Neural Network Models	14
3.1 Introduction to Recurrent Neural Networks	14
3.2 Gated Recurrent Unit Architecture	17
3.3 Implementing GRU in Keras	19
4 Hybrid Model Training and Testing Procedure	22
4.1 Training Procedure	22
4.1.1 Training Data Generation	22
4.1.2 GRU State Management	24
4.1.3 Learning Rate Scheduling	26

4.2	Testing Procedure	27
4.2.1	Testing Data Generation	27
4.2.2	Error Analysis and Accuracy Quantification	28
5	Results	29
5.1	Hybrid Open Circuit Voltage-GRU Model	30
5.1.1	Training Results	30
5.1.2	Drive Cycle Testing Results	31
5.1.3	Constant Current Testing Results	31
5.1.4	Accuracy/Data Quantity Tradeoff	33
5.2	Hybrid Single Particle-GRU Model	35
5.2.1	Training Results	35
5.2.2	Drive Cycle Testing Results	36
5.2.3	Constant Current Testing Results	37
5.2.4	Accuracy/Data Quantity Tradeoff	39
5.3	Results using Data from Physical Battery Experiments	39
5.3.1	Battery and Data Description	40
5.3.2	Training Results	40
5.3.3	Testing Results	41
5.4	Comparison to Results of Similar Studies	42
6	Conclusions and Future Directions	47
6.1	Summary	47
6.2	Directions for Improvement	48

LIST OF FIGURES

1.1	Schematic of a lithium-ion battery with cobalt oxide cathode [1].	2
1.2	Tradeoff between model accuracy and computational cost [2].	4
1.3	The basic modelling scheme.	6
2.1	Plot of OCV vs SOC for a LiFePO_4 battery.	9
2.2	Schematic of Single Particle Model, from [3]	10
3.1	Structure of a feedforward neural network [4].	14
3.2	Structure of an RNN [5].	16
3.3	Structure of an unrolled RNN [5].	16
3.4	Schematic of internal structure of GRU [5].	18
3.5	How Keras GRU produces predictions from sliding windows of inputs (with sliding window length of 3).	20
4.1	Example training current	23
4.2	Plots of (a) simulated battery voltage and OCV and (b) voltage error, under the training current shown in Figure 4.1. The data shown in (b) are what the GRU is trained to predict.	25
4.3	FUDS current profile used for testing, partially modified to prevent overdischarging.	27
5.1	Training performance of OCV-GRU model.	30
5.2	Training history of hybrid OCV-GRU model. Note that this MSE was computed during training before prediction rescaling, unlike the RMSE reported in the text.	31
5.3	FUDS testing results for hybrid OCV-GRU model.	32
5.4	Cumulative distribution function of absolute value of error in FUDS data prediction. The horizontal red line is at 0.9, and its intersection with CDF indicates the 90th percentile absolute error.	32

5.5	Testing results for constant current input of magnitude 1C, 3C, and 5C, where 1C = 26 amps for this simulated battery.	33
5.6	Plot of root mean-squared-error versus length of training data.	34
5.7	Training performance of hybrid SPM-GRU model.	35
5.8	Mean square error vs epoch number for hybrid SPM-GRU model. Note that this MSE was computed during training before prediction rescaling, unlike the RMSE reported in the text.	36
5.9	FUDS testing results for hybrid SPM-GRU model.	37
5.10	Cumulative distribution function of absolute value of error in FUDS data prediction for SPM error model.	37
5.11	Testing results for constant current input of magnitude 1C, 3C, and 5C, where 1C = 26 amps for this simulated battery.	38
5.12	Plot of root mean-squared-error versus length of training data for the SPM error model.	39
5.13	Plot of UAC and CS drive cycle currents (y-axis units are amps).	41
5.14	Training performance on UAC cycle data.	41
5.15	Prediction performance on CS cycle testing data.	42
5.16	CDF of errors on testing data.	43
6.1	The hybrid modelling scheme.	47

LIST OF TABLES

5.1	RMSE for drive cycle tests.	44
5.2	Table of rmse for constant current discharge tests.	45
5.3	Model sizes and training data quantities for Zhao, Moura, and our models. Model size refers to number of trainable parameters.	45

ABSTRACT

Hybrid Battery Modelling: Combining Physical Models with Recurrent Neural Networks

Accurate and efficient battery modelling is essential to effective battery management. Traditionally, this meant developing physics based models that include the many relevant electrochemical transport processes at play within a battery. As model fidelity improves, these models rapidly become computationally expensive, making real time modelling, say in an electric vehicle application, difficult if not impossible. Furthermore, these models usually involve dozens of physical parameters, many of which have complicated nonlinear effects on the output, and therefore require elaborate experimental procedures to identify properly. Another approach is to forgo physical modelling and rely entirely on machine learning techniques to fit a model from battery input/output data. Recurrent Neural Networks (RNNs) are a machine learning model often used, as they have a universality property that allows them to model any dynamical system, at least in principle. While this approach can achieve desirable accuracy, the models used usually have several thousand parameters, making them computationally intensive, and require large amounts of training data. In this thesis, I investigate a modelling scheme that combines both approaches, i.e. a small, efficient physics-based model in conjunction with a compact RNN model, attempting to create a model more efficient and accurate than either approach alone. This hybrid modelling scheme works by training an RNN on the prediction error of the simplified physical model. To generate a prediction, input data are fed into the physical model and RNN model in parallel, then the prediction of the RNN, which is the mismatch between the physical model and actual battery output (voltage), is added to the prediction of the physical model, yielding a more accurate prediction. Our hypothesis is that there is a balance between the accuracy of the physical model and the size of the RNN model to predict its error, that as the physical model becomes more accurate and computationally intensive, the RNN has to do less "work" to accurately predict its error, and can therefore have fewer parameters. This suggests there may be a "sweet spot" where a hybrid model

consisting of a relatively inaccurate and simplified physical model in conjunction with a compact RNN to predict its error can achieve sufficient accuracy while being overall easier and faster to train and more computationally efficient than either an all physics or all machine learning approach. To test this, I present results on accuracy and training requirements for two hybrid models, namely one with a simple static open-circuit-voltage function of the battery state-of-charge, and one with a relatively detailed model referred to as the Single Particle Model. In conjunction with each of these is an RNN of a particular architecture known as a Gated Recurrent Unit, which is an architecture designed to achieve a long memory of past input with relatively few parameters. Both hybrid models are tested first with a highly detailed computational battery model known as the Pseudo-Two-Dimensional model as a surrogate for a physical battery. We then test the open-circuit-voltage model on data from actual battery experiments. Analyses of accuracy and training complexity are also presented. Finally, the overall effectiveness of this hybrid approach is discussed, and potential future research directions are provided.

ACKNOWLEDGMENTS

I would like to express gratitude to all who supported me in this effort. First, to Professor Xinfan Lin, whose open-mindedness, patience, and generosity made this possible. Thank you for all of your guidance and support. Next, to all of the friends I've made during my time at UC Davis, who kept me sane while I rode the roller coaster of graduate school. I've met so many wonderful people whose friendship will be with me for life, and I will always be grateful for this formative period of my life. Finally I want to thank my family, my Father, Mother and Sister, for their untiring support. You have shaped me more than anyone else, and for that I thank you.

Chapter 1

Introduction

Lithium-ion batteries have become the rechargeable battery of choice due to their high energy density, lack of memory effect, and low self-discharge rate [6]. While the most obvious impact of lithium-ion batteries to date may be their ubiquity in portable electronics, their most significant impact will potentially be in transportation electrification. According to the Environmental Protection Agency (EPA), 28% of greenhouse gas emissions in the United States come from transportation [7], making this sector a prime target for emissions reduction. The electrification process is well underway, with demand for lithium-ion batteries in transportation projected to increase by an order of magnitude over the next decade [8]. Safe and efficient utilization of lithium-ion batteries in electric vehicles requires effective battery management systems, which rely on adequate battery models. In this thesis, an accurate and computationally efficient modeling architecture is explored, which is suitable for real-time control applications.

1.1 Background on Lithium-Ion Batteries

Batteries provide a means of storing electrochemical energy. Within a discharging battery, oxidation reactions occur at the negative electrode (anode), resulting in an excess of negative charge, while at the positive electrode (cathode) reduction reactions occur, resulting in an excess of positive charge. The excess negative charge in the anode is allowed to flow to the cathode by passing through the attached load, and the circuit is completed by a corresponding flow of positive ions from the negative electrode to the positive electrode

through an electrolyte. It is the alleviation of the charge imbalance between the electrodes that drives the useful work that batteries deliver. A schematic of the lithium-ion battery

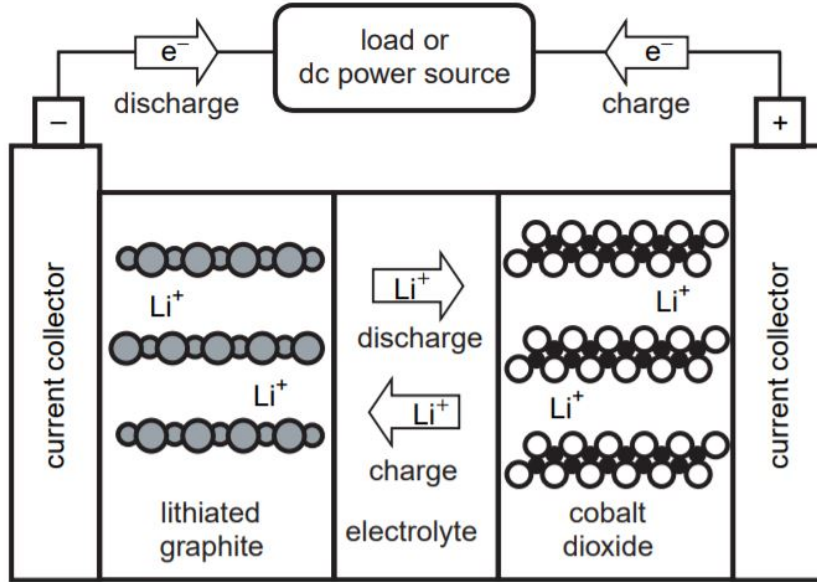
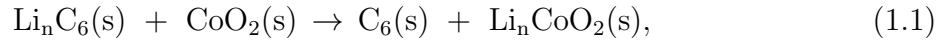


Figure 1.1: Schematic of a lithium-ion battery with cobalt oxide cathode [1].

is shown in figure 1.1. The anode material is typically graphite and the cathode generally consists of a lithium metal oxide. The overall reaction during discharging, for the chemistry in figure 1.1, is given by



which represents the intercalation of lithium with cathode material and deintercalation in anode. The above reactions in both anode and cathode are reversible, allowing for a lithium-ion battery to be recharged.

In addition to anode and cathode, other elements are necessary for a functioning battery. The electrolyte must have a solvent that allows for the rapid transport of the lithium ions, typically an organic liquid. A separator is required to ensure the insulation of electrodes against each other. Furthermore, the anode and cathode active materials are generally in the form of microscopic particles, which must be held together by a conductive binding material. The electrodes are attached to conductive current collectors, typically copper or aluminum, which are connected to the external load.

1.2 Overview of Battery Modelling

1.2.1 Applications of Battery Models

Battery modelling is useful for a number of reasons. The first one is for battery design [9]. Having an accurate model of a battery allows for the outcome of design decisions to be predicted without the cost of building a physical battery. Outcomes one would want to influence include increased energy density and lifespan among others, which can be optimized by changing battery architecture and material composition and properties based on the operating conditions [10].

Another useful function of battery models is in estimation of important internal states that cannot be directly measured but need to be closely monitored during operation, e.g. state of charge (SOC) and state of health (SOH) [3]. Specifically, SOC reflects the amount of charge left in a battery, and SOH, while difficult to define exactly as it corresponds to no single physical property, captures the extent of a battery’s functional degradation, e.g. change in the resistance or capacity over time. Model-based strategy has been a common practice for SOC and SOH estimation and widely studied in literature [11, 12, 13, 14, 15, 16].

Finally, battery models are useful for real-time control. For example, energy management [17, 18, 19], charging control [10, 20, 21, 22] and thermal management [23, 24] have all been often approached using model-based strategies. Such functions are necessary for the optimal utilization of batteries.

There are many different ways to model a battery [3]. The most basic distinction is between the physical battery models based on first principles and the data-driven models learned from data collected from real battery experiments. These approaches will be discussed in the following sections.

1.2.2 Physics-based Battery Models

Within the realm of physics-based battery models, the major categories include empirical models and electrochemical models. Empirical models capture the macroscopic physical behavior of a battery without referring to the electrochemical details occurring in an actual

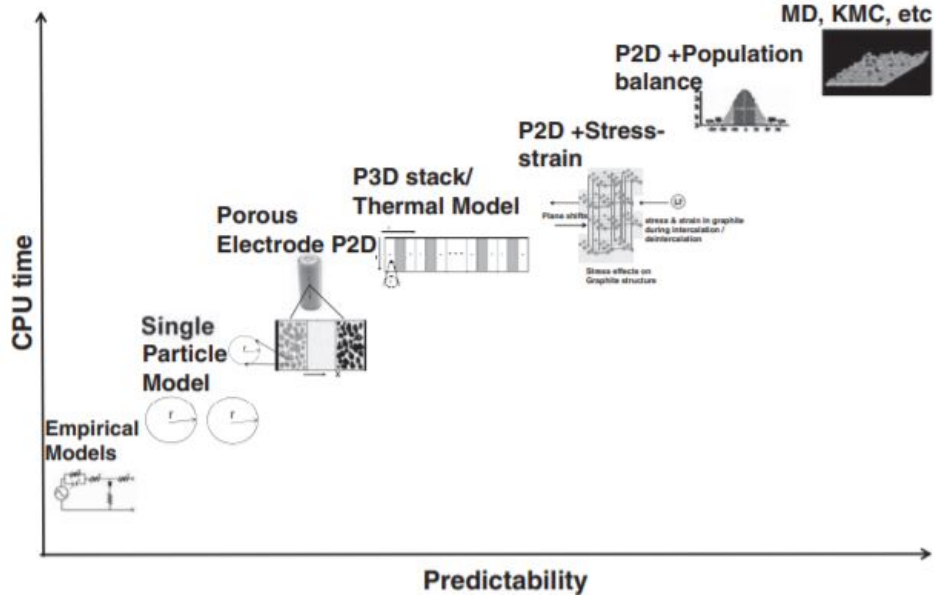


Figure 1.2: Tradeoff between model accuracy and computational cost [2].

battery. The most common form of empirical models, and the most common battery model used in battery management systems of any type [25], is the Equivalent Circuit Model (ECM) [26, 27]. These models work by fitting the parameters of an equivalent circuit, typically consisting of a open-circuit voltage (OCV) source and several resistor-capacitor pairs, to data collected from battery experiments. In more complex versions, the circuit parameters are taken as a function of SOC, and possibly temperature or even current. These models are quite computationally efficient, but their limited accuracy can somewhat restrict the application.

Electrochemical models, on the other hand, dive into the details of the processes occurring inside batteries. The most popular one is the Pseudo-Two-Dimensional Model (P2D), which was developed by Doyle, Fuller, and Newman [28]. The model captures the evolution of the electrode lithium concentration, the electrolyte lithium concentration, the electrode potential, and the electrolyte potential, each of which is governed by a partial differential equation (PDE). Current is taken as input, and the model output is the voltage. The P2D model will be discussed in more details in Chapter 2, as it is used as a surrogate in this work for a physical battery in initial simulation testings. The P2D model is quite computationally intensive, requiring the solution of 4 PDEs that are nonlinearly

coupled, and hence it is not yet widely used in real-time applications. Furthermore, the parameters of the P2D model are quite difficult to identify due to the large number of them and the complicated effects they have on the measurable model output (voltage) [29].

A more computationally efficient simplification of the P2D model is the Single Particle Model (SPM) [30]. This model is derived by assuming that the volume current density is constant along the electrode thickness. Each electrode is then modeled with a spherical diffusion equation, which can be solved in multiple ways [31, 32, 33]. The full equations of the SPM will be laid out in Chapter 2. The accuracy of SPM is limited to lower discharge rates compared with the P2D model. Extensions of SPM to include electrolyte dynamics [34, 35] and thermal effects [36, 37] have also been investigated. While more efficient than P2D, SPM is still more computationally intensive than ECM with a number of parameters that are challenging to identify.

1.2.3 Data-Driven Battery Models

Data-driven battery models are generic black-box input-output models that obtain all of their predictive power from being trained on battery experiment data [25]. In their purest form, there is no battery physics built into the structure of the model. Such models have been trained to capture the battery current input-voltage output relationship [38], estimate SOC [39, 40, 41] and SOH [42, 43], and fault diagnostics [44, 45]. Specific model architectures used include support vector machines [46, 47], fuzzy logic circuits [48, 49], feedforward neural networks [50, 51], and recurrent neural networks [38, 52]. Data-driven models have the advantage of being easy to train in the sense of not requiring elaborate experimental procedures, but do tend to require large amount of training data. Furthermore, these models often include thousands of parameters [38], which are computationally expensive for real time applications.

1.3 The Hybrid Approach

The approach pursued in this thesis integrates physical models with data-driven models, with the goal of creating a combined model that is accurate, efficient, and easy to train.

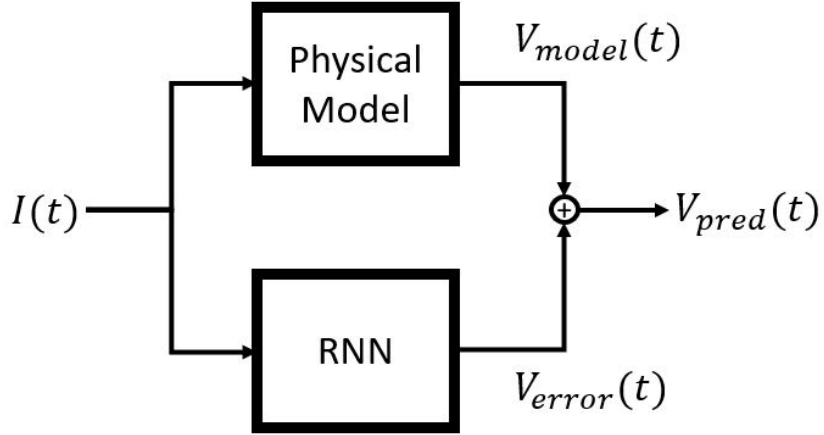


Figure 1.3: The basic modelling scheme.

The model architecture was first proposed in [53], under which a physics-based model and a Recurrent Neural Network (RNN) run in parallel, each taking current as input, as shown in figure 1.3. The RNN is trained to predict the error between the physical model and the true battery voltage, and hence the combination of two models produces a voltage prediction more accurate than that from the physical model alone. With this architecture, a relatively simple, computationally efficient, and easily identifiable physical model can be supplemented by a relatively small RNN, which has reasonable training data requirements. This thesis explores the accuracy and training data requirements for hybrid models with (1) Open Circuit Voltage (OCV) + RNN and (2) SPM + RNN, which are trained and validated using data derived from P2D battery model simulation and actual experiments.

1.4 Thesis Organization

Chapters 2 and 3 cover more background necessary to understand the hybrid model in detail. Chapter 2 discusses the relevant physical battery models in depth, and Chapter 3 introduces the basics of RNNs and the particular RNN architecture used in this study. Chapter 4 then explains in detail the procedures used to train and test the hybrid models, as well as how model accuracy is quantified. Chapters 5 presents results, first from simulation and then from actual battery experiments, as well as a comparison between

our results and those of some similar studies. Chapter 6 concludes with a summary and discussion of where improvements could be made in this study and what future directions seem fruitful.

Chapter 2

Physical Battery Models

The modelling scheme described in this thesis involves a combination of physical battery models and data-driven models. In this chapter, the details of two physical models that will be implemented in the hybrid models, namely the Open Circuit Voltage (OCV) model and the Single Particle Model (SPM), are discussed. In addition, the Pseudo-2-D (P2D) model, which is used as a surrogate for the physical battery in part of the research is also described.

2.1 Open Circuit Voltage Model

The Open Circuit Voltage (OCV) is the equilibrium voltage across the battery terminals, as determined by the thermodynamic and electrochemical states of the battery. As reactions at the anode and cathode proceed during discharge, this voltage drops. The progress of these reactions is tracked by SOC, and for the model used here OCV is taken to be a static function of SOC, which is computed by integrating the input current, an approach known as "Coulomb counting." [3] An example OCV function, i.e. the one for an LiFePO_4 -graphite battery that will be studied in this thesis, is shown in Figure 2.1.

The OCV model itself is quite limited in accuracy as it neglects the voltage dynamics due to reaction kinetics at the electrode surfaces as well as ion diffusion in the electrolyte and electrodes. These effects cause deviation of battery voltage from OCV even under moderate currents. Therefore, identifying the OCV function involves charging and discharging a battery at very low current, typically around $C/20$, to emulate the equilibrium

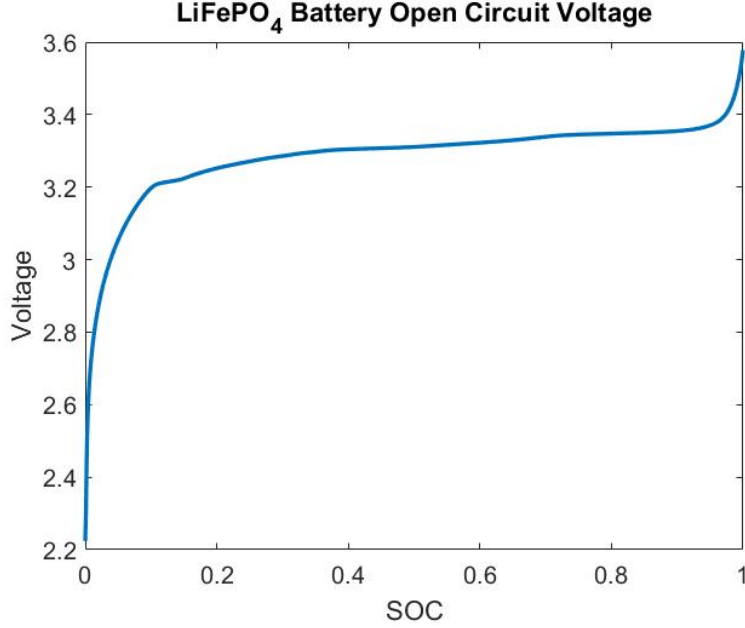


Figure 2.1: Plot of OCV vs SOC for a LiFePO_4 battery.

condition, where 1 C is defined as the current required to fully discharge the battery in one hour [26]. The voltage is recorded during the charge/discharge, and the measured voltage can either be used to fit a continuous function or directly as a look-up table. The OCV function is essentially an empirical model with moderate battery physics.

2.2 Pseudo-2-Dimensional Model

The Pseudo-2-Dimensional (P2D) model is often regarded as a full-order electrochemical model for batteries, though as Figure 1.2 shows, there are several layers of underlying physical processes that are abstracted away even in this model.

The P2D model is based on the porous electrode theory, and assumes that the anode and cathode consist of many microscopic spherical electrode particles immersed in electrolyte [28]. The model captures the spatiotemporal evolution of four sets of states, each governed by a partial differential equation (PDE) derived from a conservation principle and transport law. The first set of states are lithium ion concentrations in the electrodes, c_s , for which ion conservation and Fick's law of diffusion yield a spherical diffusion equation. The second one is lithium ion concentrations in electrolyte, c_e , for which similar conservation and transport dynamics lead to a one dimensional diffusion equation. The

third one is electrode potential, ϕ_s , for which conservation of charge and Ohm's law give a first order PDE, and the fourth one is the electrolyte potential ϕ_e , for which again conservation of charge and Ohm's law give a first order PDE. These PDEs need to be solved in the domains of the electrodes and separator, with appropriate boundary conditions. Additionally, there are intercalation/deintercalation reactions at the electrode surface, modeled via the Butler-Volmer equation, which determine the ion current density j^{Li} at the surface of each electrode particle. The model input is the current, which affects the boundary conditions, and the output is the battery voltage, which is the potential difference between two terminals.

The four PDEs of the P2D model are nonlinearly coupled, and therefore must be solved simultaneously by an iterative procedure. This is quite computationally intensive, rendering the P2D model unfit for most real-time applications. In addition, the P2D model is challenging to identify, as it contains over 30 parameters, each affecting the model output in a complicated way [29].

2.3 Single Particle Model

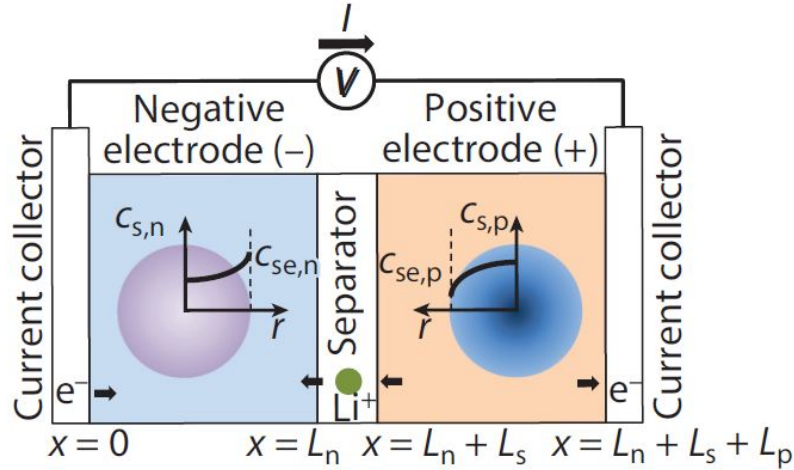


Figure 2.2: Schematic of Single Particle Model, from [3]

The Single Particle Model (SPM) is a computationally practical simplification of the P2D model [30], a schematic of which is shown in Figure 2.2. The model is derived by assuming that the current density j^{Li} is constant along the electrode thickness, so that it

can be directly computed as the volume average of I at each electrode

$$j_i^{\text{Li}} = \frac{I}{A_e \delta_i}, \quad (2.1)$$

where the subscript i indicates the electrode ($i = p$ for positive and $i = n$ for negative, A_e is the electrode area, and δ_i is the electrode thickness.

The diffusion equation governing lithium concentration c_s in each electrode particle is

$$\frac{\partial c_{s,i}}{\partial t} = \frac{D_{s,i}}{r^2} \frac{\partial}{\partial r} \left(r^2 \frac{\partial c_{s,i}}{\partial r} \right) \quad (2.2)$$

with the Von Neumann boundary conditions

$$\left. \frac{\partial c_{s,i}}{\partial r} \right|_{r=0} = 0, \quad \left. \frac{\partial c_{s,i}}{\partial r} \right|_{r=R_{s,i}} = \frac{j_i^{\text{Li}}}{a_{s,i} F}, \quad (2.3)$$

where r is the radial coordinate for each spherical particle, $D_{s,i}$ is the diffusion coefficient, $R_{s,i}$ is the particle radius, $a_{s,i}$ is the active particle surface area per unit volume, and F is Faraday's constant. This PDE can be solved in a number of ways. The approach taken here is to transform into the Laplace domain and then, using the boundary conditions, determine analytically the transfer function between I and $c_{s,i}$. The obtained transfer function is transcendental, and Pade approximation is then applied for simplification, which uses a finite order rational function as approximation. The coefficients in the numerator and denominator of the polynomial transfer function are determined by "moment matching", which involves matching the derivatives of the transfer function to s up to a certain order at $s = 0$. Here we adopt a 3rd order Pade approximation of the transcendental transfer function. This procedure is laid out in detail in [33]. Based on the obtained rational transfer function, state-space equations can be derived to calculate $c_{s,i}$ at the surface of each electrode particle, referred to as $c_{se,i}$, over time.

Lithium diffusion in the electrolyte is described by

$$\epsilon_{e,i} \frac{\partial c_e}{\partial t} = D_{e,i}^{\text{eff}} \frac{\partial^2 c_e}{\partial x^2} + (1 - t_+^0) \frac{j_i^{\text{Li}}}{F} \quad (2.4)$$

where $\epsilon_{e,i}$ is the porosity, $D_{e,i}^{\text{eff}}$ is the effective electrolyte phase diffusion coefficient, and t_+^0 is the lithium ion transference number [54]. A similar procedure is applied to solve this

equation with a 1st order Pade approximation, which is justified by the relatively slow electrolyte diffusion. Details of this calculation can be found in [55, 33].

The lithium ion concentrations will affect the battery terminal voltage, which is computed as

$$V = \phi_{s,p} - \phi_{s,n} - R_f I \quad (2.5)$$

where $\phi_{s,i}$ are the electrode potentials and R_f is a lumped resistance due to electrical contacts, current collectors, and wiring. The electrode potentials $\phi_{s,i}$ are given by

$$\phi_{s,i} = \eta_i + \phi_{e,i} + U(c_{se}, i), \quad (2.6)$$

where $U(c_{se}, i)$ is the open circuit potential as a function of the previously computed c_{se} , $\phi_{e,i}$ is the electrolyte potential, and η_i is the kinetic overpotential that drives the intercalation reaction at a finite rate. The kinetic overpotential η_i can be determined by inverting the Butler-Volmer equation

$$\eta_i = \frac{RT}{\alpha_{ac} F} \sinh^{-1} \left(\frac{j_i^{\text{Li}}}{2 a_{s,i} j_{0,i}} \right), \quad (2.7)$$

where R is the ideal gas constant, T is the temperature in Kelvin, α_{ac} is the reaction charge transfer coefficient, and $j_{0,i}$ is the concentration-dependent reference exchange current density. $j_{0,i}$ is given by

$$j_{0,i} = k_i \sqrt{c_e^0 (c_{smax,i} - c_{s,i}(R_i, t)) c_{s,i}(R_i, t)} \quad (2.8)$$

where k_i is the reaction rate, c_e^0 is the electrolyte concentration, $c_{smax,i}$ is the maximum concentration at the surface of each electrode, and $c_{s,i}(R_i, t)$ is the actual concentration at the surface of each electrode at time t . The concentration-dependence of $j_{0,i}$ gives rise to η_i 's dependence on both $c_{se,i}$ and $c_{e,i}$. The electrode potential $\phi_{s,i}$ is computed by integrating the following Ohm's law

$$\frac{\partial \phi_e}{\partial x} = -\frac{i_e(t)}{\sigma_e} + \frac{2RT(1 - t_0^+)}{F} (1 + \beta) \frac{\partial \ln(c_e)}{\partial x}, \quad (2.9)$$

where σ_e is the electrolyte conductivity and β is the activity coefficient [54]. According to the assumptions underlying SPM, each of these terms is at least piecewise constant

across the electrode and separator regions, allowing the equation to be easily integrated to yield ϕ_e .

The approximations underlying SPM become less accurate at high charge/discharge current rates, which provide the motivation to supplement SPM with a data-driven model to improve accuracy in [53]. In chapter 5, it will be shown that combining SPM with a Gated-Recurrent-Unit model achieves better accuracy with comparable model size. In the following chapter, Recurrent Neural Networks and the GRU architecture will be introduced first.

Chapter 3

Recurrent Neural Network Models

The data-driven component of the hybrid modelling scheme described in this thesis consists of a Recurrent Neural Network (RNN) with a Gated Recurrent Unit (GRU) architecture. This chapter explains the details of RNNs and GRUs, including working principles and training process.

3.1 Introduction to Recurrent Neural Networks

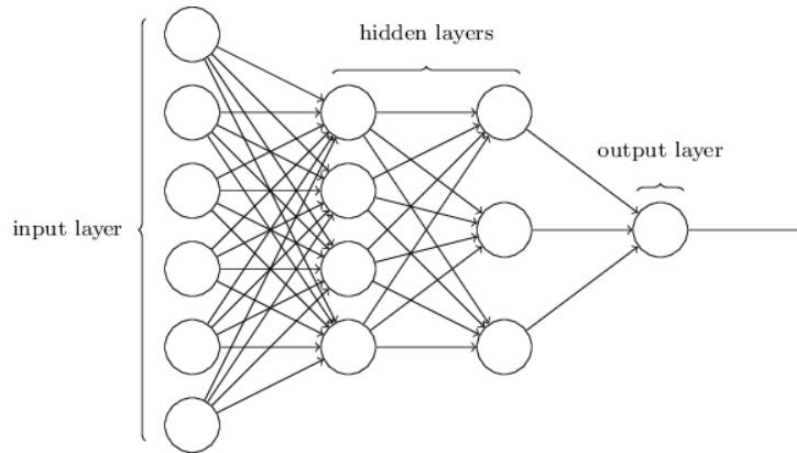


Figure 3.1: Structure of a feedforward neural network [4].

Recurrent Neural Networks are a type of Artificial Neural Network (ANN). Fundamentally, ANNs can be considered as complicated multivariable nonlinear functions. Their output is computed by passing the inputs through several layers of neurons/nodes. At each node, a weighted sum of inputs from the preceding layer is passed through a nonlin-

ear function, referred to as an activation function. The output of the activation function is passed on to nodes in the next layer, where the process is repeated, until finally the output layer is reached [4]. Mathematically, the calculation at each node is given by

$$y_i = f \left(\sum_j W_{ij} y_j + b_i \right) \quad (3.1)$$

where y_i is the output of the i^{th} node, $f(\cdot)$ is the activation function, y_j are the outputs of all the neurons in the preceding layer, W_{ij} are the weights for those outputs, and b_i is a constant bias associated with node i . The activation function is generally a monotonically increasing function with a range from -1 to 1, often a sigmoid or tanh function. Such behavior and structure of the nodes are inspired by biological neurons, and the connections between nodes are best represented as edges connecting them to form a network, as illustrated in Figure 3.1.

The key to making an ANN functional is to have the right weights W_{ij} . Remarkably, it has been shown that, given non-polynomial activation function and an unlimited number of nodes, an ANN can approximate *any* continuous function of its inputs to any desired degree of accuracy [56]. Thus any problem that can be formulated as evaluating a function, such as classifying written digits or spoken words, or translating from one language to another, can in principle be solved by an ANN, given appropriate weights [4].

Finding the appropriate weights for a neural network is known as training. The training process works by presenting a network with a data set of known inputs and outputs. The difference between the true outputs and those generated by the network, i.e. the prediction error, is passed through a cost function, which is typically the square function for a continuous prediction problem. The gradients of the cost function with respect to the network weights W_{ij} are then computed based on backpropagation [57], which is used to update the weights in the direction of decreasing cost. This process is repeated until a set of weights that deliver sufficient accuracy have been found.

ANNs as so described are static functions of their inputs. Each evaluation of the network is independent of the past inputs and outputs, and the network lacks *memory*. A Recurrent Neural Network is an ANN with memory, which works by having the output(s)

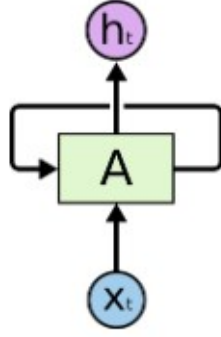


Figure 3.2: Structure of an RNN [5].

of the network at a given time instant fed back to the network as input(s) at the next instant. Note that a notion of time is inherent here, with the inputs coming at discrete time intervals. The edges connecting the output(s) of the network back to the inputs are called recurrent connections, as illustrated in Figure 3.2.

Just as ANNs have the ability to approximate any continuous function, RNNs can, in principle, approximate any continuous dynamical system [58]. This means that given a sufficient number of nodes and the correct set of weights, an RNN can be used to model any dynamic physical process. Again, the difficulty in practice is finding the correct weights. The training process for RNN is different, because the aforementioned standard training process for feedforward ANNs, particularly the backpropagation, does not consider recurrent connections in the network. The most common solution is a simple trick that effectively eliminates the recurrent connections in the network, which is to unroll/unfold the RNN over time by making multiple copies of the network and feeding the output of each copy to the next copy, as shown in Figure 3.3. The result is a feedforward

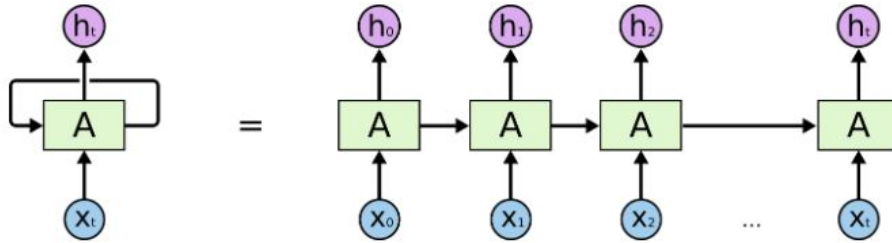


Figure 3.3: Structure of an unrolled RNN [5].

network with many repeated sub-networks of the same structures and weights but no recurrent connections, taking inputs over multiple time steps simultaneously. A network in this configuration can be trained based on the standard methods, e.g. backpropagation, by using a sliding window of inputs and outputs as training samples. This procedure is called Backpropagation Through Time (BPTT) [59, 60], and it is generally regarded as the most practical way to train RNNs [61].

3.2 Gated Recurrent Unit Architecture

All RNNs have memory, and given sufficient number of nodes, can in principle model any dynamical system with any length of memory. In practice, however, it is difficult to train RNNs with memory that lasts for more than a few dozen time steps [62]. This is principally due to the fact that the error gradients on network weights tend to decay exponentially as one goes/unfold further backward from the present moment, which is commonly known as the Vanishing Gradient Problem [63]. This is a substantial limitation for many RNN applications, as well as in our case due to the fact that batteries typically have critical dynamic time constants on the order of hundreds of seconds, while typical sampling/prediction frequency is in the order of 1 Hz, meaning that any useful RNN will need to have a memory capacity of hundreds of time steps.

The Vanishing Gradient Problem has primarily been addressed by designing more elaborate recurrent network architectures to have longer memory. A classic example is the Long Short Term Memory (LSTM) network [64]. An LSTM has a set of "gate" layers that exert enhanced control over the memory of the network, allowing it to have longer memory. A related architecture is the Gated Recurrent Unit (GRU) [65], which has fewer weights for a given number of nodes, and has been shown to have comparable memory performance [66].

The internal structure of a GRU is shown in Figure 3.4. The symbol x_t represents exogenous input(s) at time t , and h_{t-1} represents the input(s) that is recurrent from the previous output. Split of black lines in the figure denotes a copying of data, while merge of two black lines indicates concatenation of signals. The \times and $+$ symbols represent

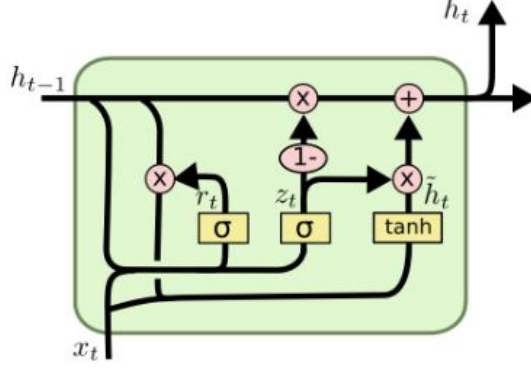


Figure 3.4: Schematic of internal structure of GRU [5].

element-wise multiplication and addition, respectively. Going from left to right, we first see the evaluation of r_t , which is known as the reset gate, given by

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t]) \quad (3.2)$$

where $\sigma(\cdot)$ is the sigmoid function, whose output range from 0 to 1, and W_r is the weight matrix for this gate. Then we have the evaluation of z_t , known as the update gate

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t]). \quad (3.3)$$

The meanings of these layers become apparent when we see how they are subsequently used. We next have the calculation of \tilde{h}_t , which is known as the candidate output

$$\tilde{h}_t = \tanh(W_h \cdot [r_t * h_{t-1}, x_t]), \quad (3.4)$$

where $*$ represents element-wise multiplication. The output of the reset gate r_t , which is between 0 and 1, determines how much of the previous state h_{t-1} is "reset". Finally, the output of the GRU is computed as

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t, \quad (3.5)$$

from which it is clear that the update layer z_t controls how the new output h_t is produced from a combination of the candidate output and the previous output.

The gated structure allows the GRU to overcome the vanishing gradient problem. Essentially, the gates control the flow of information between the input and memory. The network weights W_r and W_z learn how and when to change the internal state in response to input. This layer of control permits much longer memory of input.

3.3 Implementing GRU in Keras

All of the GRU model generation and training in this project was carried out using Keras [67], a popular machine learning Python package built on top of a library that handles basic mathematical operations in an optimized fashion. There are a number of library options available, and TensorFlow [68] was chosen for this project. Keras is designed for simplicity and code efficiency. To demonstrate this, code for creating the model used in this project is shown below.

```
1 vmodel = Sequential()  
2 vmodel.add(GRU(neurons, batch_input_shape = (batch_size, ...  
        train.shape[1], train.shape[2]), stateful = True))  
3 vmodel.add(Dense(1))  
4 vmodel.compile(loss='mean_squared_error')
```

This code builds a model out of multiple layers. The first layer is a GRU layer with a number of nodes equal to `neurons`. The output of the GRU layer, corresponding to h_t in figure 3.4, is a vector of length `neurons`. To convert this vector output into a single scalar, the predicted voltage error, another layer is needed. The single node `Dense` layer fills this need. It takes the vector output h_t and combines the elements in a weighted sum to produce the final scalar output. The model is then compiled such that the cost function used in training is the mean squared error. Training can be carried out with a single line of code,

```
1 history = vmodel.fit(train, Vd_train, epochs=1, ...  
        batch_size=batch_size, verbose=1, shuffle=False)
```

which performs a single epoch of training over the entire input/output training data set. The Adam optimizer was used, which is the default algorithm in Keras. All weights of the model `vmodel` are updated once, and the mean squared error for this epoch is passed to the `history` variable.

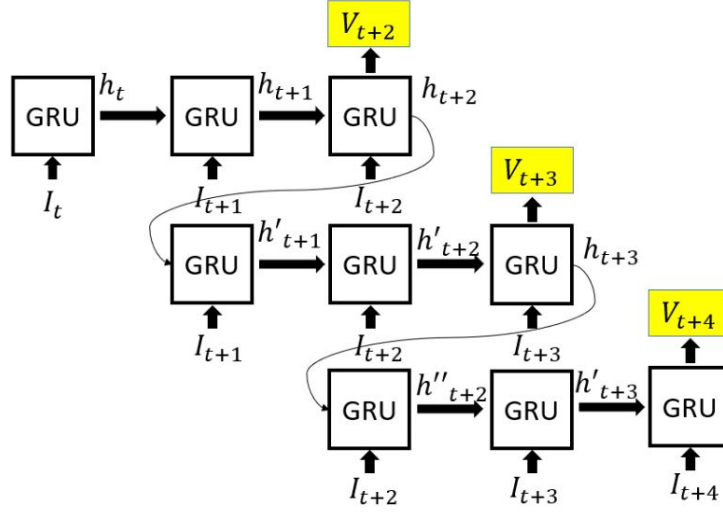


Figure 3.5: How Keras GRU produces predictions from sliding windows of inputs (with sliding window length of 3).

In order to carry out the BPTT training algorithm, the training data needs to be split into sliding windows of a fixed length. The length of the window corresponds to the number of times the model is "unfolded", as shown in figure 3.3. This sliding window structure also applies to model prediction. Testing data must also be split into sliding windows, and the GRU produces its output prediction by using all inputs in the slide window, as illustrated in Figure 3.5 for a case where the sliding window is of length 3. Here, the current input at various time instants/steps is indicated by I_t , and the internal state passed from one time step to the next is h_t . The inputs in an entire window are used to generate a single voltage prediction. The internal state is passed from one time step to the next as expected, and then passed from the end of one input window to the start of the next input window. The sliding window of inputs effectively acts as an additional form of memory, supplementing the internal memory of the GRU. In our case, adequate performance can be achieved with a window length of 5 time steps, which is deemed as acceptable.

In summary, this chapter explained the principles of how neural networks, recurrent neural networks, and Gated Recurrent Units work, including training procedures and some aspects of implementation within the Keras software framework. In the next chapter,

details for training GRUs to predict battery model errors and quantifying the performance will be presented.

Chapter 4

Hybrid Model Training and Testing Procedure

This chapter will explain in detail the training and testing of hybrid battery models. First, different aspects of the training process will be elaborated, e.g. how the training data are generated, how the internal states of the GRU are manipulated during training, and how the learning rate is adjusted during training. The specific training and testing procedures for the hybrid battery models are then discussed, including the data used, and how the accuracy of the model is assessed.

4.1 Training Procedure

4.1.1 Training Data Generation

Most of the results in this thesis are derived from battery simulation. The battery model used for simulation is the Pseudo-2-Dimensional (P2D) model, as described in section 2.2. The model solver is implemented in MATLAB, which also uses the CasADi package [69]. The P2D equations are discretized via Pade approximations and Finite Difference methods (FDM). Specifically, solid-state Lithium ion diffusion in both the anode and cathode is discretized by 3rd order Pade approximation at 76 spatially separated nodes, while diffusion in the separator is solved using FDM with 36 nodes. All other transport equations are solved entirely with FDM. A time step of 1 second was chosen for temporal discretization. Full model implementation details can be found in [70].

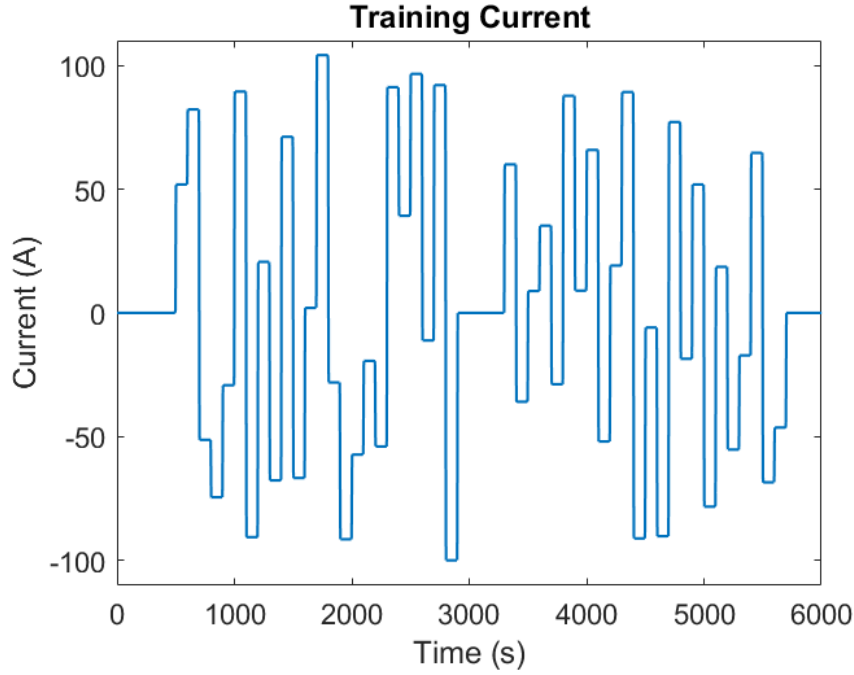


Figure 4.1: Example training current

With the model in place, the next step is to determine the current input to excite the battery and produce the training voltage data. A sequence of current pulses with random magnitudes is adopted here. Each pulse has a duration of 100 seconds, which is long enough for the battery internal dynamics to be at least partially exhibited in the voltage. The range of current magnitudes was chosen to exceed that of the current input used to test the model, as neural networks typically have difficulty extrapolating beyond the training data set. Experiments also showed that abrupt discontinuities in current input tend to induce larger errors in model output, and hence the training current was smoothed by a Lowell filter with a 20 second window. The mean of the input current was set to zero, so that the SOC was the same at the beginning and end of the current sequence. Finally, a zero-current phase of at least 500 seconds of was added to both the beginning and end of each training current sequence, for reasons that will be explained in the subsequent section. A representative training current sequence is shown in figure 4.1.

The output battery voltage profile is then generated in simulation based on the input sequence and the P2D model and treated as the ground-truth voltage data. Meanwhile, the voltage prediction by the physical battery model is also generated. For the Open

Circuit Voltage model, this simply involves computing the SOC throughout the training run by integrating the training current, and then evaluating the OCV as a function of SOC throughout the run. For the Single Particle Model, the training current is fed directly into the model.

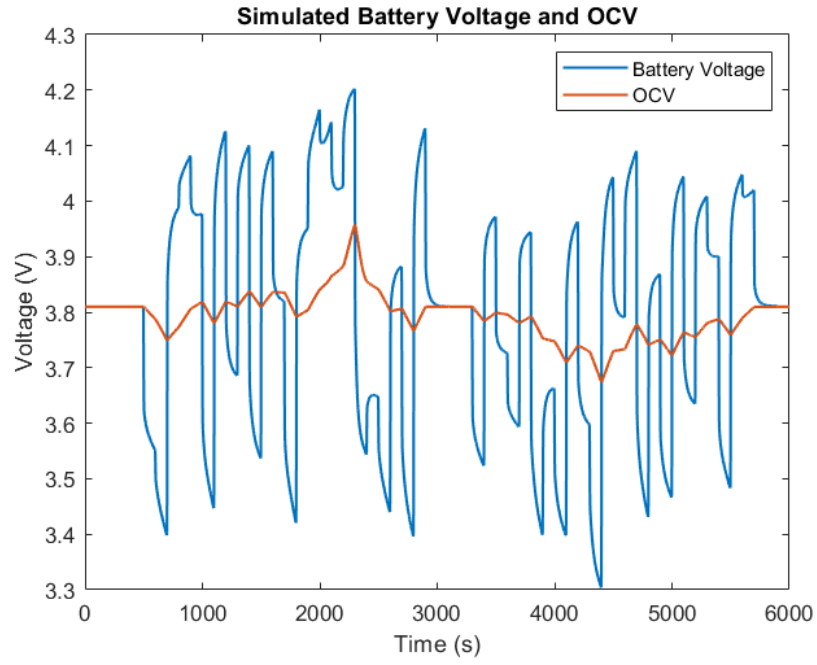
The physical model voltage predictions are then subtracted from the P2D simulation voltage data to obtain the voltage errors, which are the data for training GRU. Figure 4.2 illustrates this process for the OCV model case. Specifically, Figure 4.2a shows the simulated battery voltage and the OCV under the training current sequence in Figure 4.1, and Figure 4.2b shows the difference between the two, which are used for training GRU.

One final step before using the current and generated voltage error data to train the GRU is to scale the input and output data to the range between -1 and 1. This is a standard neural network practice, since the neural network activation functions are generally designed to saturate under inputs outside that range [71]. When using the trained RNN for prediction, it is necessary to scale down the current by the same factor used in training, and scale up the model voltage error prediction by the same factor used in training.

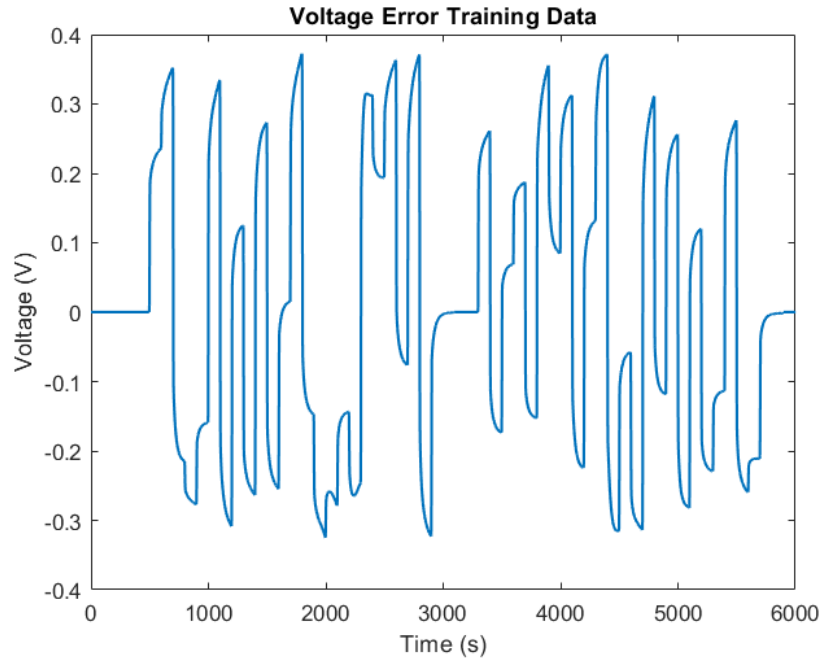
4.1.2 GRU State Management

A practical difficulty that goes along with using GRUs is the initialization of the internal state, which corresponds h_t in figure 3.4. The strategy used to overcome this difficulty was ensuring that the beginning and end of the current training sequence matched, so that the final state reached by the GRU at the end of the sequence matched what was needed to accurately predict the start of the sequence. This is the purpose of the stretch of zero current placed at the beginning and end of all training current sequences. At the end of the final stretch of zero input, the GRU is in a state such that zero input current results in zero voltage error prediction, which is just what is needed for the beginning stretch of zero input.

This same technique was applied to testing also, in that the testing current had a stretch of zero input appended to the start, thus ensuring that the model was, at the end of training, in a good initial state to start prediction on the testing data.



(a)



(b)

Figure 4.2: Plots of (a) simulated battery voltage and OCV and (b) voltage error, under the training current shown in Figure 4.1. The data shown in (b) are what the GRU is trained to predict.

4.1.3 Learning Rate Scheduling

It is a standard practice in neural networks to vary the learning rate during training, which is generally referred to as Learning Rate Scheduling [72, 73]. There are many elaborate approaches for such practice, and the basic idea is to reduce the learning rate, essentially the step size during gradient descent, over time, i.e. taking smaller steps as the model accuracy improves. In this work, the learning rate is manually reduced, usually by a factor of 4, after a number of epochs.

Training begins with the learning rate set to 0.001, which is the default value for the Adam optimizer. For the first 10 epochs, where one epoch corresponds to one complete pass through all of the training data, the internal states are reset to zero after each epoch. After 10 epochs, the model has learned the appropriate behavior in response to zero input, so that it becomes advantageous to not reset states between epochs. Typically 20 more epochs are carried out before reducing the learning rate to 0.00025, i.e. one quarter the default value. The model is then trained until the accuracy stops improving, typically in the range of 30-60 epochs total.

As mentioned previously, the training performed in this work is not entirely automated. This is because of the randomness and variability intrinsic to the process of training RNNs. The initial values of the weights are chosen randomly, causing some training runs to converge more rapidly than others, and some runs failing to converge at all. Furthermore, it was observed during training that even when the cost function, i.e. the mean squared error of the predictions on the training data, was rather flat, visual inspection of the training prediction performance showed improvement. This is probably due to fact that Adam optimizer incorporates several elaborations not present in simple gradient descent [74]. These observations made automating the termination of training, either through using a fixed number of epochs or training until a particular loss metric value is reached, undesirable, and hence training was terminated manually.

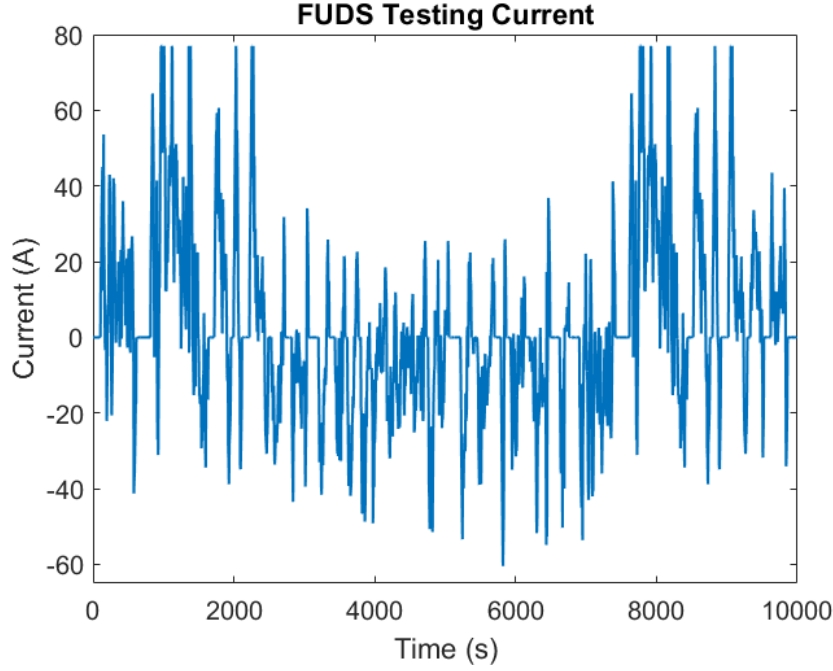


Figure 4.3: FUDS current profile used for testing, partially modified to prevent overdischarging.

4.2 Testing Procedure

4.2.1 Testing Data Generation

The models were tested primarily using the Federal Urban Driving Schedule (FUDS) current profile. The FUDS is a standardized drive cycle developed for vehicle testing that simulates a typical urban commute. It contains many abrupt starts and stops, and irregular, high frequency variation, effectively simulating a human driver. This cycle is distinct from the smoothed current steps used in training, giving an effective out-of-sample test for the GRU. The FUDS drive cycle has been converted to current values reflecting the power demanded of the battery, and includes negative current values, which are associated with charging by the regenerative braking system. The current was scaled such that its peak value is approximately 80 amps. With a time step of 1 second, it was found that the standard FUDS cycle would overdischarge the battery studied in this work, and hence a section, from around 2000 to 7000 seconds, was inverted in sign to avoid overdischarge. The FUDS cycle was applied to both the P2D simulator and battery physical model (OCV and SPM) to generate voltage simulation data, and the difference

between them are used to test the GRU component of the hybrid model. The testing has also been performed based on actual battery experiments, which replaces the P2D simulation voltage with the measured voltage data.

In addition to testing the performance of the model on drive-cycle input, we were also interested in seeing how the model performs on constant current input, such as is often experienced during battery charging. Accordingly, I simulated constant current discharging at rates of 1C, 3C, and 5C, where 1C is defined as the amount of current necessary to discharge a battery in 1 hour, which for this simulated battery is 26 amps. It should be noted that the current profiles used were not strictly constant, given the need to match initial conditions, as discussed in section 4.1.2, the current profiles each began with 100 seconds of zero input. Additionally, the step up to constant current was smoothed with a 20 second window Lowell filter, just like the steps in the training current profiles. Once the simulated battery voltages were generated, the physical model prediction was computed, then the difference calculated to give the V_{err} values that we wish to predict, just as with the other training and testing data.

4.2.2 Error Analysis and Accuracy Quantification

Once the GRU voltage error predictions on the FUDS input current are generated, the performance of the model is assessed. This was primarily done via two metrics. The first is the root mean squared error between the predicted and true voltage errors. This metric is closely related to the loss function used to train the model. The second metric is the 90th percentile absolute prediction error, which is the threshold where 90% of the absolute prediction errors fall within. It is computed by first evaluating the cumulative distribution function of the absolute errors, and then determining the value when that function is at 0.9. This value gives a sense of the upper bound of the model error, and reflects the worst-case model performance to certain extent. We will also compare our results with similar studies that used the same metrics.

Chapter 5

Results

This chapter demonstrates the predictive performance of the hybrid modelling scheme, starting with the results from training and testing using data from P2D model simulation. The results of training and testing are presented first for the configuration where the OCV model is in place as the physical model. Testing results for the FUDS drive cycle as well as constant current input are shown, followed by an analysis of the tradeoff between accuracy and training data quantity. The same analyses are then presented for the configuration where the SPM is the physical model. Finally, results of applying the hybrid modelling scheme to experiment data from an actual physical battery are shown. These results are then compared to some similar recent studies.

For each of these model variations, the performance of a single representative GRU model instance is shown. During the course of this study, hundreds of models have been trained to explore the effects of variations of model size and training data on predictive accuracy. The particular instances included here represent typical results using the procedures that produced the most accurate models. It is important to note that rapid iteration of training procedure is possible with this model, as the longest time required to train any of the models was on the order of 1 hour, carried out on a typical consumer laptop.

5.1 Hybrid Open Circuit Voltage-GRU Model

For the configuration with the OCV as the physical model, a GRU with 8 internal nodes was found to be the smallest functional model. A GRU of this size has a total of 272 weights. The timestep parameter, which corresponds to the length of the input window, as discussed in section 3.3, was set to 5. This was the smallest workable value, which was desirable as a smaller input window is more practical for real-time implementation.

5.1.1 Training Results

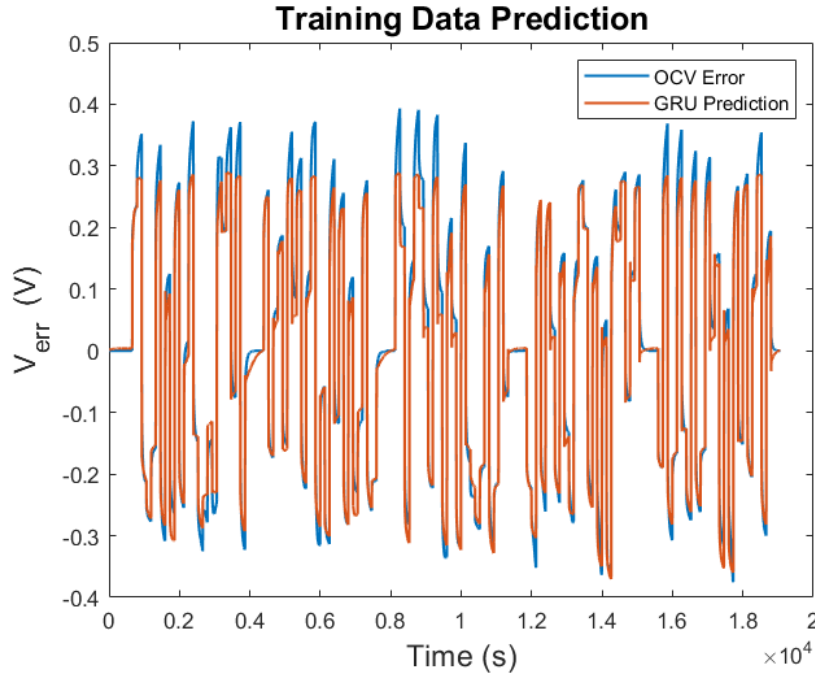


Figure 5.1: Training performance of OCV-GRU model.

Figure 5.1 shows the training results for the hybrid OCV-GRU model. For this particular model, a 19,300 second sequence of 100 second pulses was used for training. It is clear that the 8 neuron GRU with a 5 second input window is capable of capturing the dynamics of V_{err} , which is the difference between the true battery voltage and the OCV. The root mean-squared-error for this model on the training data is 29.3 mV.

Figure 5.2 shows the training history of this GRU. Only 30 epochs were necessary for this model which, for this quantity of training data, translates to approximately 30 minutes of training. While the mean squared error has converged to its final value by

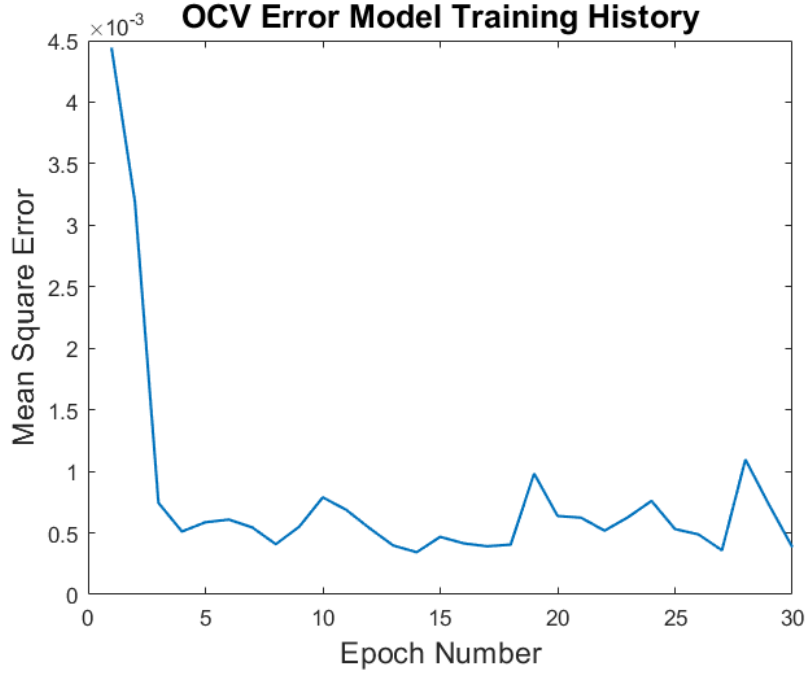


Figure 5.2: Training history of hybrid OCV-GRU model. Note that this MSE was computed during training before prediction rescaling, unlike the RMSE reported in the text.

epoch 10, experience was that there were still substantial improvements in predictive performance over the subsequent epochs. No variation in learning rate was done for this model, as the default value of 0.001 was maintained for all epochs.

5.1.2 Drive Cycle Testing Results

The prediction results for the FUDS drive-cycle test data are shown in Figure 5.3. The model can effectively capture the OCV error for this drive cycle, by correctly generalizing from a training data set that was of much lower frequency. The root mean-squared-error for the run was 21.8 mV. Figure 5.4 shows the cumulative distribution function (CDF) of the absolute value of the error over the course of the FUDS data prediction. From the intersection of the CDF with the red line we can see that 90% of predictions are within 36.2 mV of the true value.

5.1.3 Constant Current Testing Results

The results of constant current testing are shown in Figure 5.5. The root-mean-squared errors were 23.5 mV, 53.9 mV, and 155.9 mV for the 1C, 3C, and 5C tests, respectively.

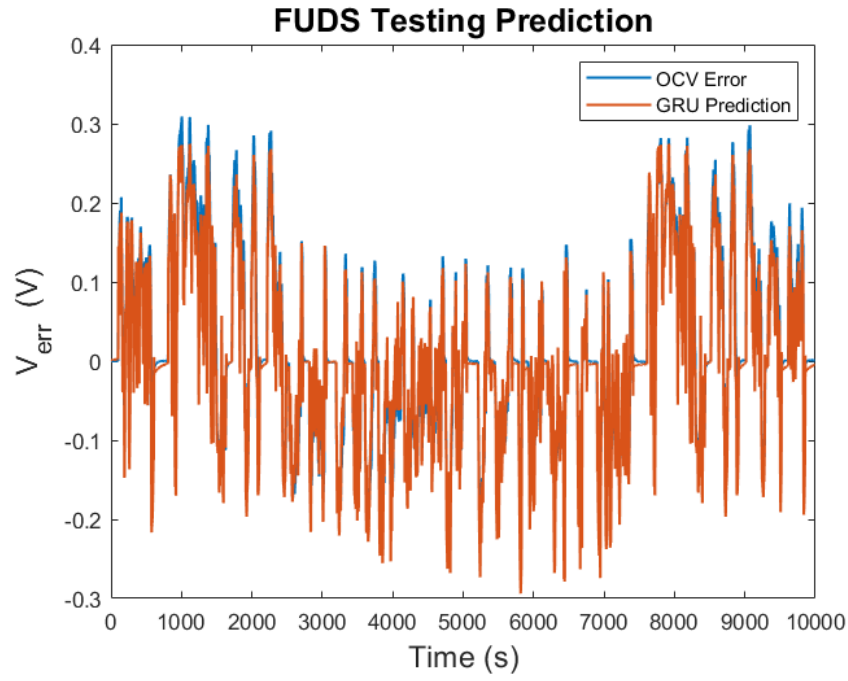


Figure 5.3: FUDS testing results for hybrid OCV-GRU model.

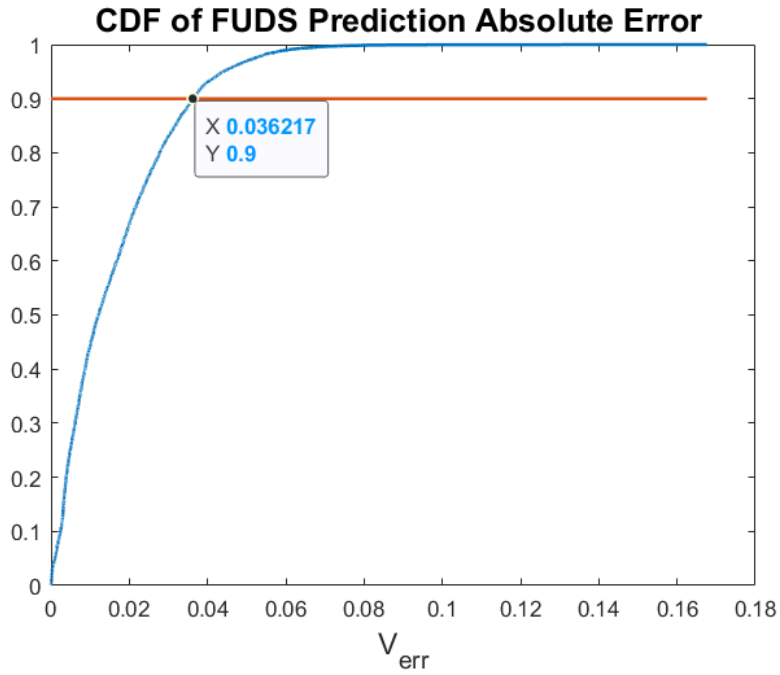


Figure 5.4: Cumulative distribution function of absolute value of error in FUDS data prediction. The horizontal red line is at 0.9, and its intersection with CDF indicates the 90th percentile absolute error.

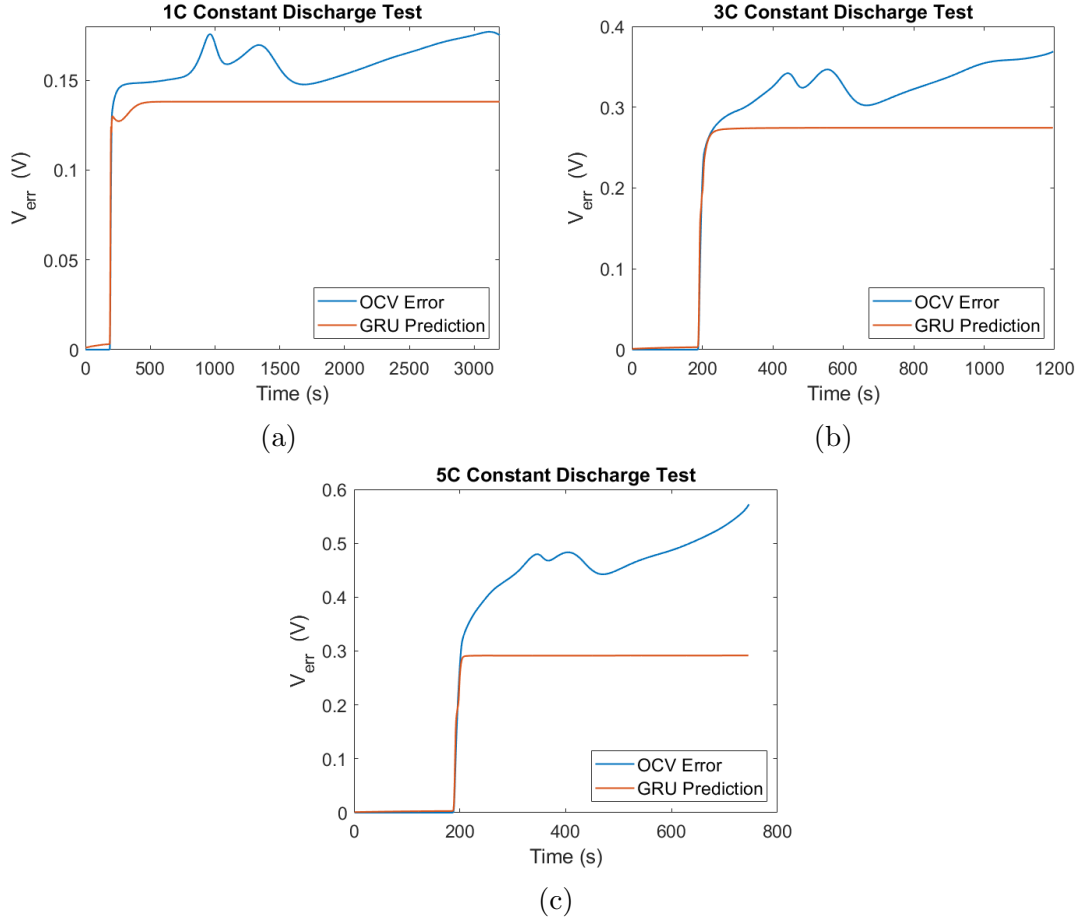


Figure 5.5: Testing results for constant current input of magnitude 1C, 3C, and 5C, where $1C = 26$ amps for this simulated battery.

It is clear that performance degrades as the input current magnitude increases, which is common of battery models due to the increasing complexity of battery dynamics at higher discharge rates. It is seen that the GRU output is close to the average value of V_{err} , but fails to capture the undulations over long timescale. This makes sense when considering that these features would be the most demanding of the GRU's memory, as they only appear after hundreds of time steps of unchanging input.

5.1.4 Accuracy/Data Quantity Tradeoff

After refining the process of model training, the dependence of model accuracy on the quantity of training data was evaluated. The goal was to determine a minimal training data set, which could still produce a well functioning model. To evaluate this, training

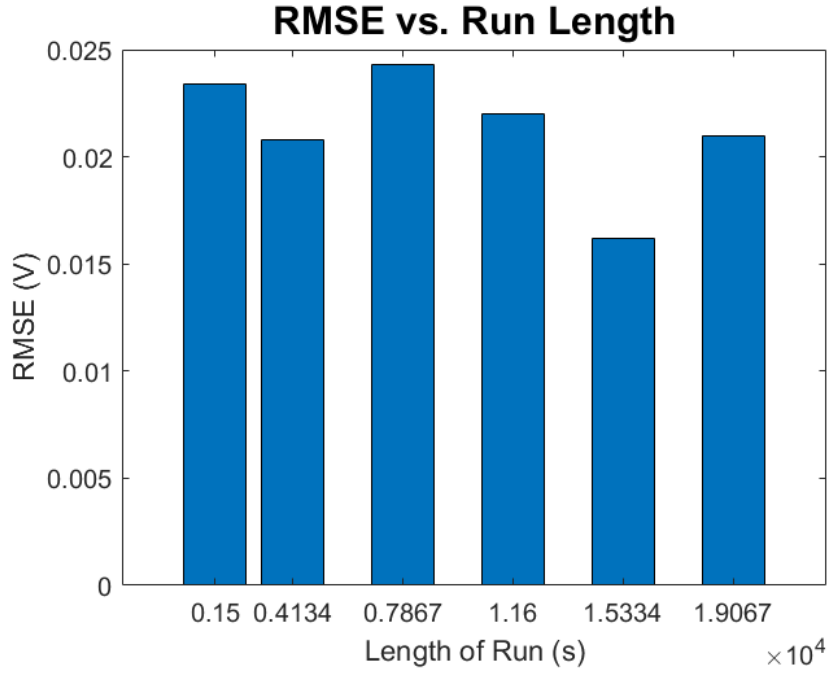


Figure 5.6: Plot of root mean-squared-error versus length of training data.

data sets of decreasing size were used to train models, whose accuracy was evaluated via root mean-squared-error of prediction on the FUDS testing data. Decline in accuracy as the quantity of training data decreased was expected.

The results are shown in Figure 5.6. It is seen that reducing the amount of training data by one order of magnitude did not substantially reduce the accuracy of the model. Over the range tested, the training data quantity is not well correlated with model accuracy. The smallest data set tested consisted of just 5 100 second pulses, with 500 seconds of zero input attached to the beginning and end, and the prediction accuracy was comparable to others. This result can be viewed positively as it could mean that much fewer training data are necessary, which saves time in generating data and training. However, it may also suggest that the model is "shallower" than thought. That is, maybe GRU is in fact learning a relatively simple model, which for the FUDS testing data is sufficient. Further investigation on this topic can be pursued in future.

5.2 Hybrid Single Particle-GRU Model

A configuration in which a Single Particle Model was used as the physical model was also tested. Given that the SPM is a much more accurate model for battery voltage dynamics, the magnitude of the model error that the GRU was trained to predict is substantially smaller than that of the hybrid OCV-GRU model. We hypothesized that a more accurate model only needs to be supplemented with a smaller and less powerful RNN. The results lend support to the hypothesis, as a 4 neuron GRU was found to be adequate, as opposed to the 8 neuron GRU used for the OCV error model, which translates to 84 network weights vs 272. A timestep value of 5 was used, as in the OCV case.

5.2.1 Training Results

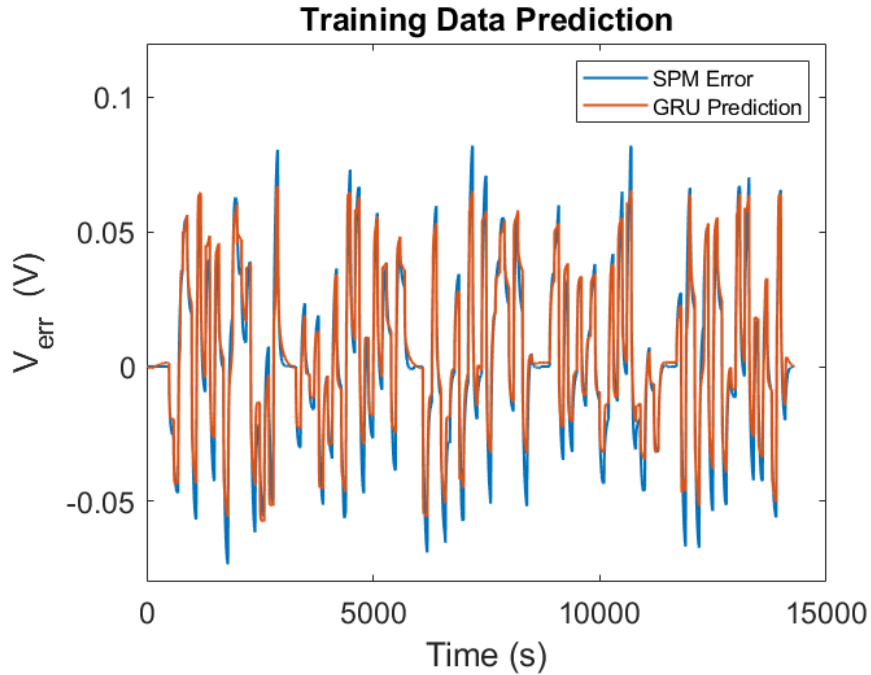


Figure 5.7: Training performance of hybrid SPM-GRU model.

The results of training for the SPM error model are shown in Figure 5.7. The smaller magnitude of the error, relative to the OCV model error, is evident here. It is clear that the model captures the error dynamics well. The root-mean-square-error was 7.7 mV for predictions on the training data set.

The training history of the model over 60 epochs is shown in Figure 5.8. It is seen that

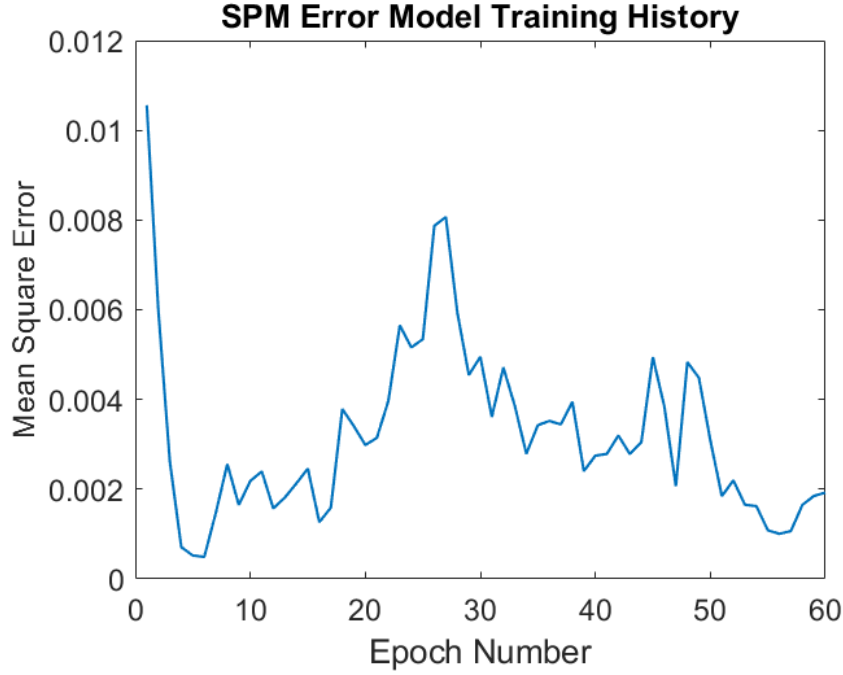


Figure 5.8: Mean square error vs epoch number for hybrid SPM-GRU model. Note that this MSE was computed during training before prediction rescaling, unlike the RMSE reported in the text.

the mean square error does not decrease monotonically, which was typical and expected given the use of momentum in the Adam optimizer. The last 20 epochs of training were done using a learning rate of 0.00025, which is one quarter of the default value.

5.2.2 Drive Cycle Testing Results

Figure 5.9 illustrates the performance of the GRU error model on the FUDS testing data set. It is clear that the model predicts the dynamics accurately, with a root mean square error of 2.4 mV. The most obvious discrepancies appear to be when the current input is zero. This was something observed consistently in model training, as learning the correct response to zero input appeared to be particularly difficult. The cumulative distribution function of the absolute error for the SPM error model is shown in Figure 5.10. The CDF intersection with the red line indicates the 90th percentile error, which in this case is 3.8 mV.

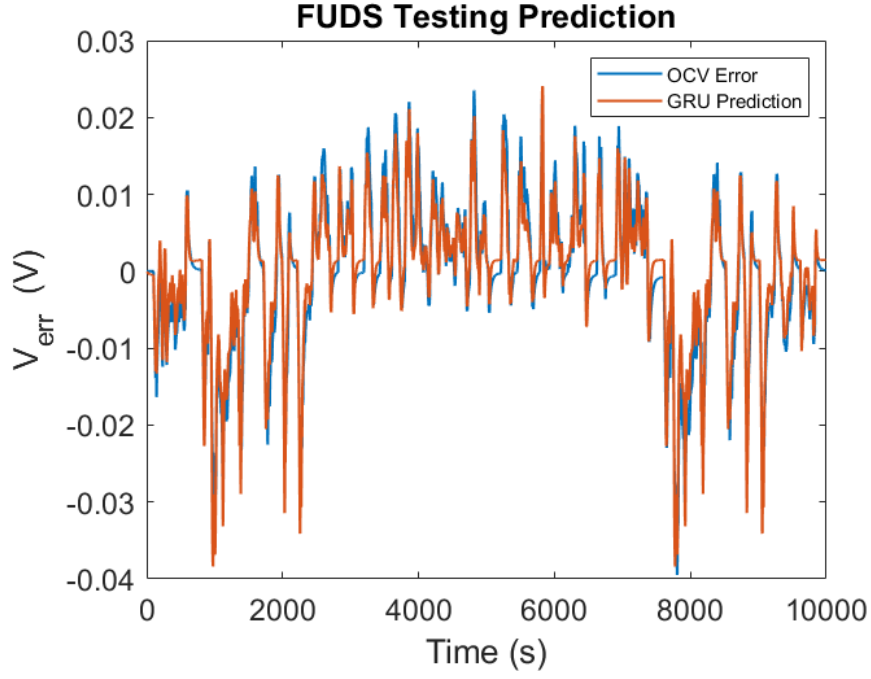


Figure 5.9: FUDS testing results for hybrid SPM-GRU model.

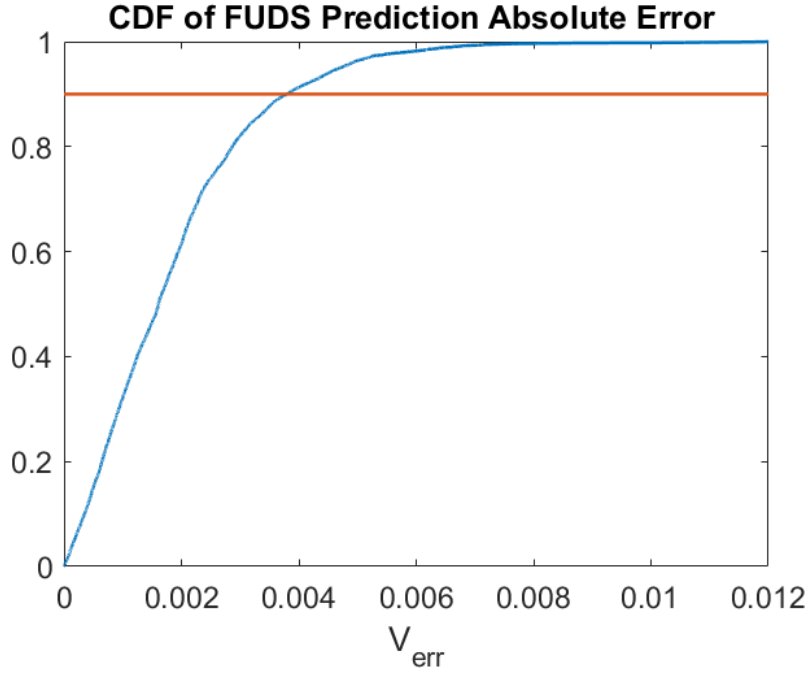


Figure 5.10: Cumulative distribution function of absolute value of error in FUDS data prediction for SPM error model.

5.2.3 Constant Current Testing Results

Constant current testing was carried out for the SPM error model as well, and the results are plotted in Figure 5.11. One note regarding these results is that the sign of the voltage

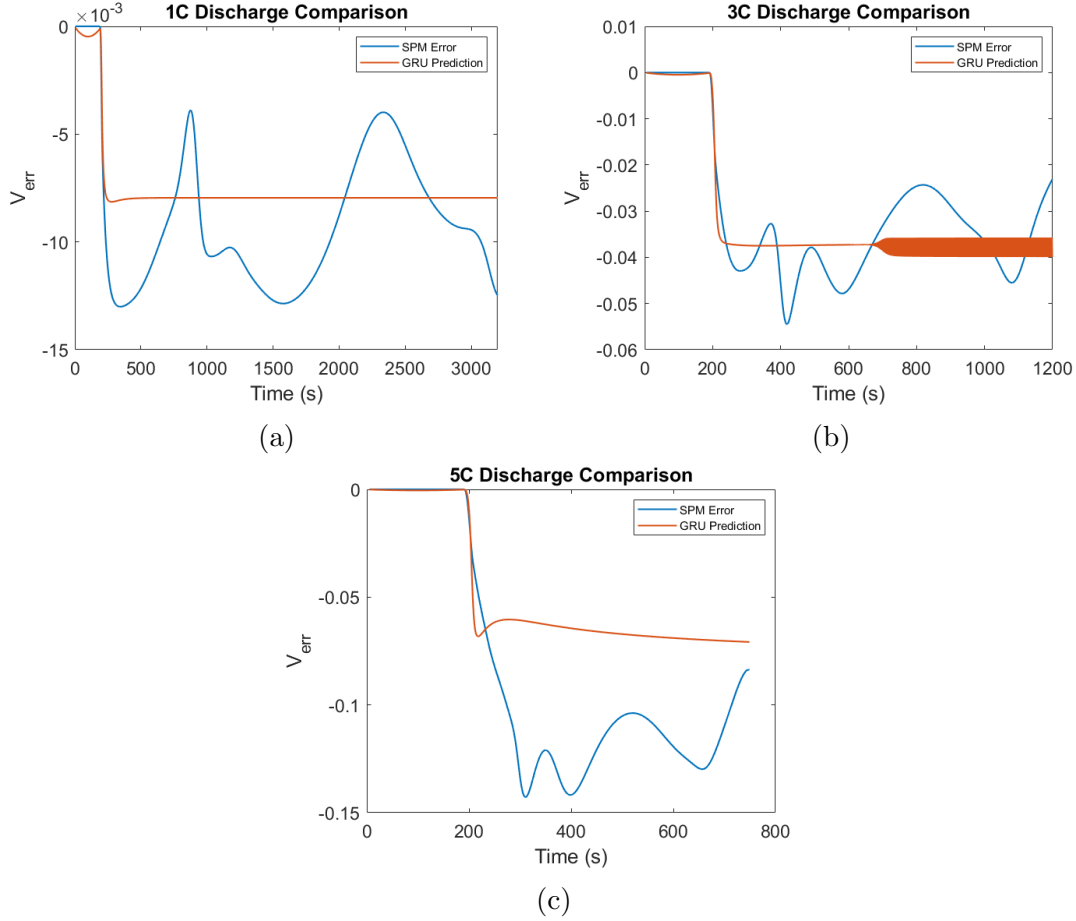


Figure 5.11: Testing results for constant current input of magnitude 1C, 3C, and 5C, where 1C = 26 amps for this simulated battery.

error in these experiments is negative, due to a sign inversion in the SPM analysis that did not occur in the OCV analysis. It is seen that, for the 1C and 3C test data at least, the model successfully captures the long-term trend albeit missing the short-term fluctuation. Overall, the accuracy decreases with higher discharge rate. There is an interesting feature in the 3C data. The thickening of the curve around 700 seconds is due to the fact that the model starts to oscillate between 2 values over consecutive time steps. This behavior was seen occasionally during constant current testing, which is certainly undesirable but not very detrimental to overall accuracy. The root mean square errors for the 1C, 3C, and 5C discharge tests were 3 mV, 7mV, and 44 mV respectively.

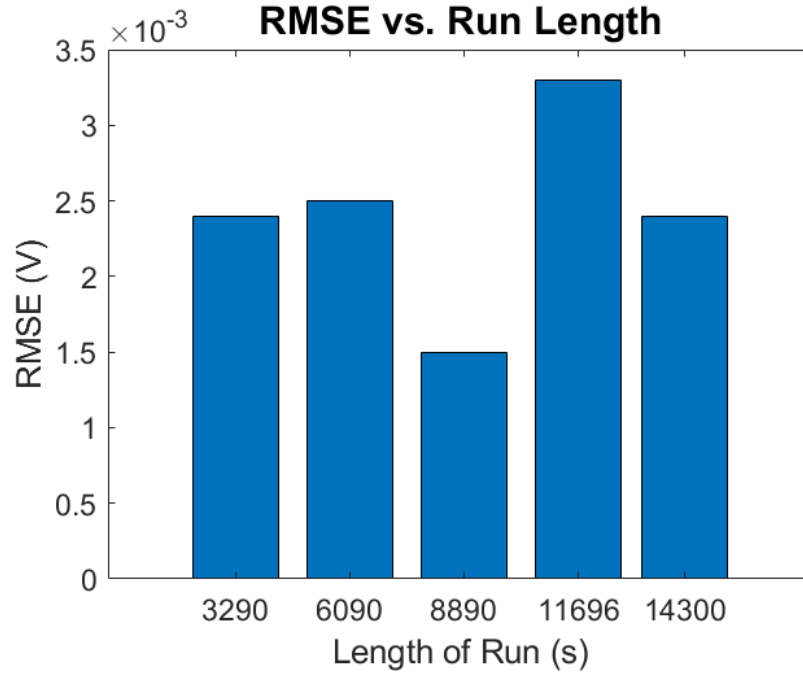


Figure 5.12: Plot of root mean-squared-error versus length of training data for the SPM error model.

5.2.4 Accuracy/Data Quantity Tradeoff

A test of model accuracy versus training data quantity was conducted for the SPM error model, as was done with the OCV error model. The results were similar, in that dramatically reducing the size of the data set did little to reduce the accuracy of the model during the FUDS drive cycle testing. Therefore the same concern arises, that the model may be learning a relatively simple relationship between inputs and outputs, which is effective for the drive cycle data but perhaps less effective on lower frequency data, such as the constant current discharge tests.

5.3 Results using Data from Physical Battery Experiments

One of the goals of this project is to apply the hybrid modeling methodology to a real physical battery. Unfortunately, this goal was hindered by the COVID-19 pandemic, which made it impossible for us to perform experiments on actual batteries with our battery testing equipment. Fortunately, there were data archived from a 2014 study [26]

available that was found to be sufficient to train the models. The results obtained are not ideal, due to lack of ability to design the training data, and instead training and testing had to be performed on two distinct existing drive cycle data sets. We could also only test the OCV error model, as we did not have a Single Particle Model identified for the battery.

A significant aspect of the 2014 study was the measurement of temperature data during battery discharge. The data indicate that the battery surface and internal temperatures increased substantially over the course of the drive cycles tested. Given that this is likely to affect the input current-terminal voltage relationship, we decided to incorporate temperature into the GRU error model. This was simple to implement using Keras, only requiring augmenting the model input. The temperature data were processed by normalizing (dividing by the maximum value) to the range between 0 and 1, and then dividing into sliding windows of size 5, which are similar procedures of processing the current input data. A GRU with 8 neurons was used.

5.3.1 Battery and Data Description

The data were collected from experiments on a 2.3 amp-hour A123 26650 LiFePO₄ graphite battery cell. The OCV was identified by charging and discharging slowly at a rate of C/20, which equals 0.115 amps. The drive-cycle current inputs used for training and testing are shown in Figure 5.13. The UAC (Urban Assault Cycle) and CS (Charge Sustaining) cycles are aggressive operating conditions, with peak discharge rates near 20 C. They were developed in the context of military electric vehicle simulation. The UAC cycle was used for training, and the CS cycle was used for testing. More details about the battery, experimental configuration, and drive cycles can be found in [26].

5.3.2 Training Results

The training results are shown in Figure 5.14. Clearly, the model is capable of fitting the dynamics of the OCV error with good accuracy. Visual inspection does not reveal any obvious patterns in discrepancy. The root mean square error for the training data set was 47 mV.

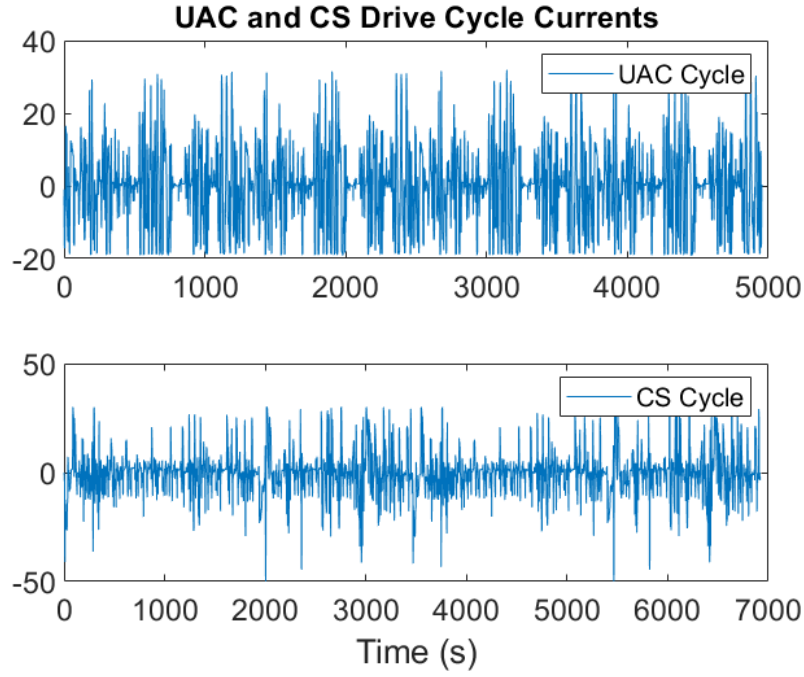


Figure 5.13: Plot of UAC and CS drive cycle currents (y-axis units are amps).

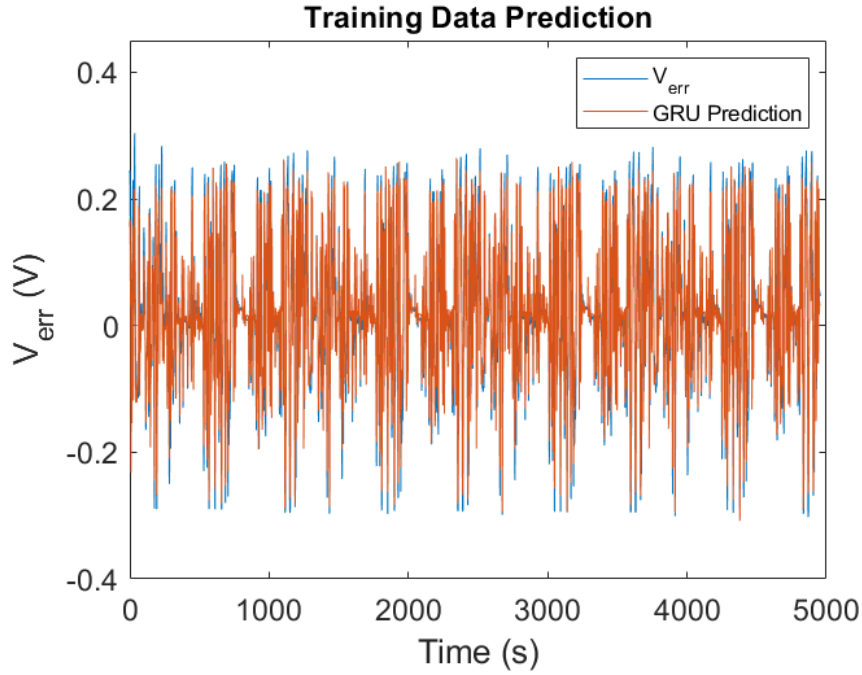


Figure 5.14: Training performance on UAC cycle data.

5.3.3 Testing Results

The model prediction performance on the testing data is shown in Figure 5.15. As with the training data, it is clear that the model is capturing the general trend of the battery

dynamics with largest discrepancies occurring at high output values. The root mean square error for the testing prediction was 58 mV, and the 90th percentile absolute error was 90mV, as can be seen in Figure 5.16. These error values are both substantially larger than those achieved with the simulated battery data, which is not surprising given the much more limited data availability for training.

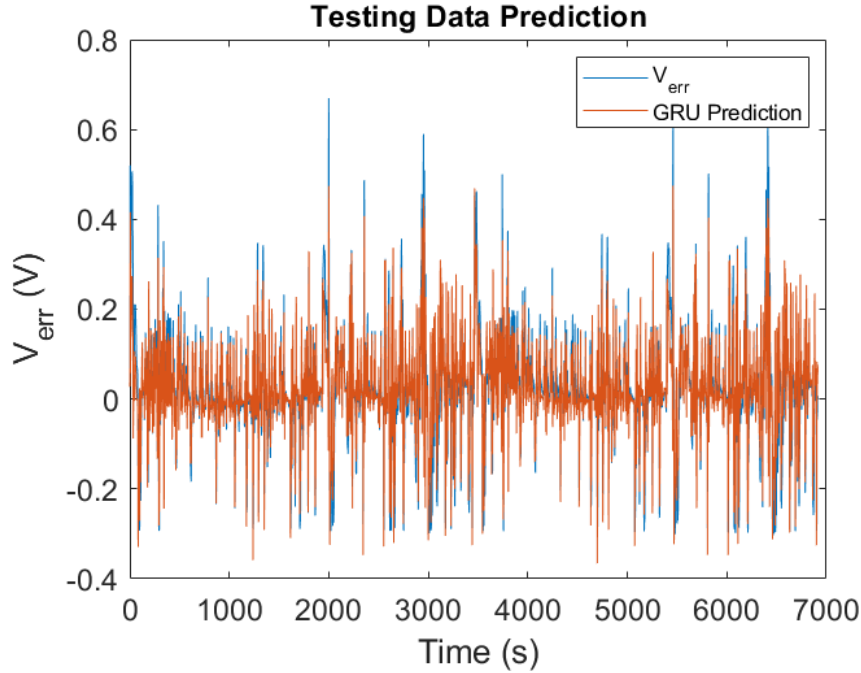


Figure 5.15: Prediction performance on CS cycle testing data.

5.4 Comparison to Results of Similar Studies

There are 2 recent studies to which the obtained results can be compared. The first, to be found in reference [38], will be referred to as the Zhao study, and the second, found in reference [53], will be referred to as the Moura study. The methodology of each study will first be summarized, followed by presentation of key data related to accuracy and model size that will be used to compare with our results. The comparisons will primarily be with the results derived from the simulation study, as the limitations of data on the experiment data study make the comparison not ideal. The physical battery results will be briefly compared with the results in [26], i.e. the Lin study, from which the training and testing data were taken.

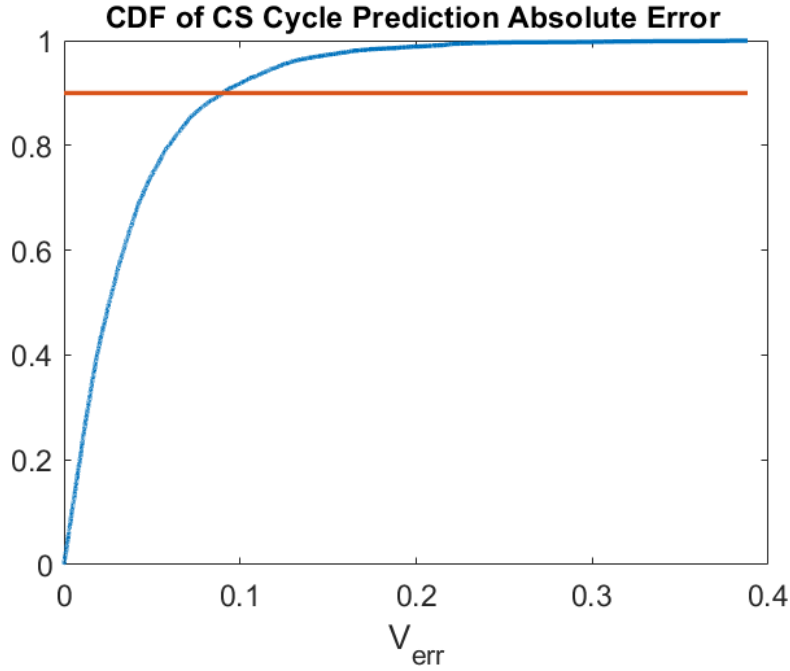


Figure 5.16: CDF of errors on testing data.

In the Zhao study, the authors created an RNN to model the current input-voltage output relationship of a Panasonic NCR 18650PF battery. The RNN architecture they chose had 2 GRU layers with 30 nodes each, combined with an input weighting layer. The model inputs are current, battery temperature, battery state-of-charge, and the output (voltage) of the model at the previous time step. This architecture leads to a model with over 6000 trainable parameters. Training data consisted of 4 different drive cycles, each carried out under 4 different temperatures. Each cycle was about 10,000 seconds long, so the total quantity of training data was approximately 160,000 seconds. The model was tested using a drive cycle consisting of random selections from the 16 cycles used during training.

The Moura study used a model architecture that was the inspiration for this work. It consisted of an SPM in conjunction with an RNN, with the RNN trained to predict the discrepancy between the SPM and the true battery voltage. The SPM used was more complicated than the one used here, with 60 states to model the lithium diffusion dynamics. The RNN used to model the SPM error was quite simple, i.e. a 4-node Elman network [75], with a total of 24 parameters. They used a P2D battery model for data

generation, similar to our simulation study. The model was trained using the Urban Dynamometer Driving Schedule (UDDS), a standard fuel economy test cycle. The UDDS was repeated twice, giving a total of about 3,000 seconds of training data. Training was carried out using the Real Time Recurrent Learning (RTRL) algorithm. The model was subsequently tested on 1C, 2C, and 5C constant current discharging simulation data.

Table 5.1 compares the root mean square error of the Zhao model with our models over drive cycle test data. It is seen that both the OCV and SPM error models in simulation study significantly outperform the Zhao model. However, the study based on experiment data, which is a fairer comparison given that the Zhao study used experiment data, does significantly worse. Granted, that was under the constraint of no capability to generate desired/design training data for the current experiment studies. Also noted that the error for the Zhao model cited here is under $T = 25\text{ }^{\circ}\text{C}$. They also reported errors at the 3 other temperatures tested, which were all greater. We report the $25\text{ }^{\circ}\text{C}$ result because it is the best and also closest to the average ambient temperature in experiment.

	Zhao Model	OCV Error Model (Simulation)	SPM Error Model (Simulation)	OCV Error Mode (Experiment)
RMSE	37 mV	21.8 mV	2.4 mV	58 mV

Table 5.1: RMSE for drive cycle tests.

Table 5.2 compares the constant current discharge test results for the Moura study and this work. Note that for the intermediate discharge rate, the Moura study used 2C and we used 3C. We choose to compare only the constant current results because the drive cycle in the Moura study were only used for training. It is seen that for all discharge rates, our OCV error model performs worse than the Moura model, while the SPM error model performs better.

These accuracy measures need to be combined with considerations on model size and training data quantity to obtain a comprehensive comparison of the usefulness of the models. Table 5.3 lists the model sizes, in terms of number of trainable parameters, and training data quantity used in the studies. The Zhao study used a much larger

Discharge Rate	Moura Model	OCV Error Model	SPM Error Model
1C	9.2 mV	23.5 mV	3.0 mV
2C/3C	18.5 mV	53.9 mV	7.2 mV
5C	66.5 mV	156 mV	43.8 mV

Table 5.2: Table of rmse for constant current discharge tests.

model, as well as much more training data than the other two. This makes sense given that it is relying entirely on a two layer GRU to model the battery, with no assistance from a physics-based model. Given the potential high accuracy of the OCV error model (demonstrated in simulation study) and the ease of measuring OCV, the hybrid OCV-GRU model provides a promising hybrid modeling architecture. For the SPM error model, the comparison is complicated by the substantial difficulty in identifying an SPM. Whether the enhanced accuracy is worth the greater identification difficulty is something that could only be determined in the context of a particular application.

	Zhao Model	Moura Model	OCV Error Model	SPM Error Model
Model Size	6000+	24	272	84
Training Data (s)	160,000	3,000	19,000	14,000

Table 5.3: Model sizes and training data quantities for Zhao, Moura, and our models. Model size refers to number of trainable parameters.

The Moura model is much more similar to our models in terms of size and data requirements, though that model requires about 1/4 of the trainable parameters. Training data requirements are comparable, considering that our training data/accuracy trade-off exploration showed that our models could be trained with considerably fewer data. Our SPM error model has greater accuracy, at the cost of larger model size, however it is still rather compact. We expect run-time computational cost to be similar. The OCV model has lower accuracy, but OCV is much easier to identify than the SPM. Again, it would take additional considerations from the demands of a particular application to determine

whether the decreased accuracy of the OCV error model is worth the trade-off for ease of identification.

Finally, the lumped-parameter electrothermal model developed in the Lin study [26] achieves an rmse of 20 mV on its drive cycle test. This is approximately 3 times better than our OCV error model achieved. However, again there is a considerable advantage in ease of model identification with the OCV error model.

Chapter 6

Conclusions and Future Directions

6.1 Summary

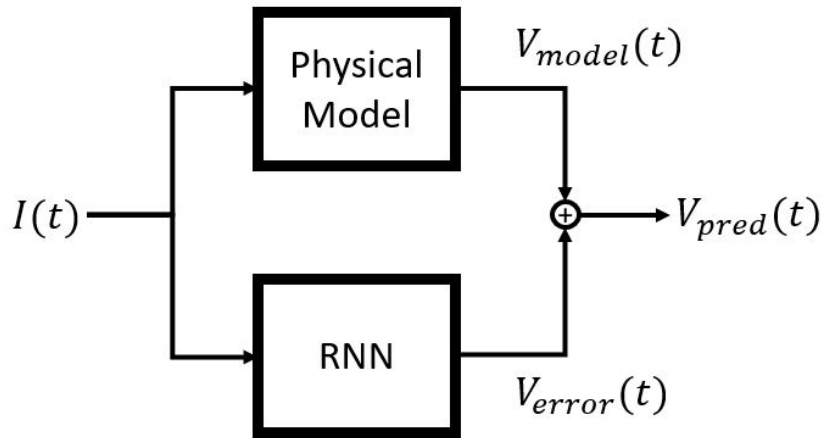


Figure 6.1: The hybrid modelling scheme.

This thesis described the development of a hybrid model for predicting the voltage response of a battery to current input. The model is a combination of a physical model and a recurrent neural network, with the RNN trained to predict the discrepancy between the physical model and the true battery voltage, or the error of the physical model, as illustrated in Figure 6.1. RNN with a Gated Recurrent Unit was used, which is an architecture designed to achieve long memory with relatively few parameters. Results were presented using a simple Open Circuit Voltage function as well as a Single Particle model as the physical battery model, with a Pseudo-2-Dimensional battery model adopted

for data generation. Additionally, the performance of a hybrid OCV-RNN model trained with data from actual physical battery experiment data was also discussed.

One hypothesis at the outset of this study was that a more sophisticated physical model would require a smaller RNN error model to achieve a given level of accuracy. The hypothesis is supported by the result that in the SPM case a substantially smaller RNN was effective than in the OCV case.

Overall, it was clear for each model variation that the GRU was able to capture the dynamics of the physical model error. For the SPM error model, achieved accuracy was better than that of a similar study [53], while the OCV error model was less accurate. In terms of ease of model identification, the OCV error model is definitely superior, since the SPM model identification requires much more elaborated efforts.

A recurring theme in these comparisons is the tradeoff between ease of model identification and accuracy. Both are significant, however only the context of a particular application can determine whether a given tradeoff between the two is worthwhile. This makes it difficult to claim in a general sense which model is "better". However, our results certainly indicated that for a battery modelling application where accuracy is important, this hybrid approach is worthy of consideration.

6.2 Directions for Improvement

There are a number of ways in which this study could be improved. One would be to rework the sliding window input architecture of the GRU. While we decided that the 5 second input duration was small enough to be practical, it would be yet more practical if current could be fed in one time step at a time, which is more convenient in practice and puts less demand on the internal memory of the RNN.

Another area that could be improved is the training data. Many variations of training data have been used with the attempt of improving prediction accuracy, particularly for the constant current testing cases, but no dramatic improvement had been found so far over the initial training data consisting of smoothed random 100 second pulses. The space of possible input data is enormous, and there are definitely certain ones that could yield

better results.

Finally, another obvious aspect for improvement is the training and testing data used for the physical battery experiment study. We only had data archived to create what would be considered a preliminary model, however the model was far from ideal due to the lack of data design capability. Ideally we would want to create and use new training and testing data, more akin to those used in the simulation study to improve and better validate the results.

REFERENCES

- [1] R. S. Treptow, “Lithium batteries: a practical application of chemical principles,” *Journal of chemical education*, vol. 80, no. 9, p. 1015, 2003.
- [2] V. Ramadesigan, P. W. Northrop, S. De, S. Santhanagopalan, R. D. Braatz, and V. R. Subramanian, “Modeling and simulation of lithium-ion batteries from a systems engineering perspective,” *Journal of the electrochemical society*, vol. 159, no. 3, p. R31, 2012.
- [3] X. Lin, Y. Kim, S. Mohan, J. B. Siegel, and A. G. Stefanopoulou, “Modeling and estimation for advanced battery management,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 2, pp. 393–426, 2019.
- [4] M. Nielsen, “Neural networks and deep learning.” <http://neuralnetworksanddeeplearning.com/chap1.html>, December 2019. (Accessed on 07/20/2020).
- [5] C. Olah, “Understanding lstm networks – colah’s blog.” <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>, August 2015. (Accessed on 07/15/2020).
- [6] “Advantages & limitations of the lithium-ion battery - battery university.” https://batteryuniversity.com/learn/archive/is_lithium_ion_the_ideal_battery. (Accessed on 06/30/2020).
- [7] “Sources of greenhouse gas emissions — greenhouse gas (ghg) emissions — us epa.” <https://www.epa.gov/ghgemissions/sources-greenhouse-gas-emissions>. (Accessed on 07/02/2020).
- [8] “Global battery demand by application — statista.” <https://www.statista.com/statistics/1103218/global-battery-demand-forecast/>. (Accessed on 07/02/2020).
- [9] C. D. Rahn and C.-Y. Wang, *Battery systems engineering*. John Wiley & Sons, 2013.
- [10] S. Jangra, C. H. Chung, Q. Lai, and X. Lin, “Optimal maintenance of electric vehicle battery system through overnight home charging,” in *Dynamic Systems and Control Conference*, vol. 59155, p. V002T22A001, American Society of Mechanical Engineers, 2019.
- [11] G. L. Plett, “Extended kalman filtering for battery management systems of lipb-based hev battery packs: Part 3. state and parameter estimation,” *Journal of Power sources*, vol. 134, no. 2, pp. 277–292, 2004.
- [12] P. Singh, S. Kaneria, J. Broadhead, X. Wang, and J. Burdick, “Fuzzy logic estimation of soh of 125ah vrla batteries,” in *INTELEC 2004. 26th Annual International Telecommunications Energy Conference*, pp. 524–531, IEEE, 2004.

- [13] S. Ebbesen, P. Elbert, and L. Guzzella, “Battery state-of-health perceptive energy management for hybrid electric vehicles,” *IEEE Transactions on Vehicular technology*, vol. 61, no. 7, pp. 2893–2900, 2012.
- [14] X. Lin, A. G. Stefanopoulou, Y. Li, and R. D. Anderson, “State of charge imbalance estimation for battery strings under reduced voltage sensing,” *IEEE Transactions on Control Systems Technology*, vol. 23, no. 3, pp. 1052–1062, 2014.
- [15] X. Lin, H. E. Perez, J. B. Siegel, A. G. Stefanopoulou, Y. Li, R. D. Anderson, Y. Ding, and M. P. Castanier, “Online parameterization of lumped thermal dynamics in cylindrical lithium ion batteries for core temperature estimation and health monitoring,” *IEEE Transactions on Control Systems Technology*, vol. 21, no. 5, pp. 1745–1755, 2012.
- [16] X. Li, Z. Wang, L. Zhang, C. Zou, and D. D. Dorrell, “State-of-health estimation for li-ion batteries by combining the incremental capacity analysis method with grey relational analysis,” *Journal of Power Sources*, vol. 410, pp. 106–114, 2019.
- [17] C. Musardo, G. Rizzoni, Y. Guezennec, and B. Staccia, “A-ecms: An adaptive algorithm for hybrid electric vehicle energy management,” *European Journal of Control*, vol. 11, no. 4-5, pp. 509–524, 2005.
- [18] G. Ripaccioli, A. Bemporad, F. Assadian, C. Dextreit, S. Di Cairano, and I. V. Kolmanovsky, “Hybrid modeling, identification, and predictive control: An application to hybrid electric vehicle energy management,” in *International Workshop on Hybrid Systems: Computation and Control*, pp. 321–335, Springer, 2009.
- [19] X. Li, L. Xu, J. Hua, X. Lin, J. Li, and M. Ouyang, “Power management strategy for vehicular-applied hybrid fuel cell/battery power system,” *Journal of Power Sources*, vol. 191, no. 2, pp. 542–549, 2009.
- [20] C.-H. Chung, S. Jangra, Q. Lai, and X. Lin, “Optimization of electric vehicle charging for battery maintenance and degradation management,” *IEEE Transactions on Transportation Electrification*, 2020.
- [21] R. Klein, N. A. Chaturvedi, J. Christensen, J. Ahmed, R. Findeisen, and A. Kojic, “Optimal charging strategies in lithium-ion battery,” in *Proceedings of the 2011 american Control Conference*, pp. 382–387, IEEE, 2011.
- [22] H. E. Perez, X. Hu, S. Dey, and S. J. Moura, “Optimal charging of li-ion batteries with coupled electro-thermal-aging dynamics,” *IEEE Transactions on Vehicular Technology*, vol. 66, no. 9, pp. 7761–7770, 2017.
- [23] Q. Wang, B. Jiang, B. Li, and Y. Yan, “A critical review of thermal management models and solutions of lithium-ion batteries for the development of pure electric vehicles,” *Renewable and Sustainable Energy Reviews*, vol. 64, pp. 106–128, 2016.

- [24] G. Karimi and X. Li, “Thermal management of lithium-ion batteries for electric vehicles,” *International Journal of Energy Research*, vol. 37, no. 1, pp. 13–24, 2013.
- [25] M.-F. Ng, J. Zhao, Q. Yan, G. J. Conduit, and Z. W. Seh, “Predicting the state of charge and health of batteries using data-driven machine learning,” *Nature Machine Intelligence*, pp. 1–10, 2020.
- [26] X. Lin, H. E. Perez, S. Mohan, J. B. Siegel, A. G. Stefanopoulou, Y. Ding, and M. P. Castanier, “A lumped-parameter electro-thermal model for cylindrical batteries,” *Journal of Power Sources*, vol. 257, pp. 1–11, 2014.
- [27] Y. Hu, S. Yurkovich, Y. Guezennec, and B. Yurkovich, “Electro-thermal battery model identification for automotive applications,” *Journal of Power Sources*, vol. 196, no. 1, pp. 449–457, 2011.
- [28] M. Doyle, T. F. Fuller, and J. Newman, “Modeling of galvanostatic charge and discharge of the lithium/polymer/insertion cell,” *Journal of the Electrochemical society*, vol. 140, no. 6, p. 1526, 1993.
- [29] J. C. Forman, S. J. Moura, J. L. Stein, and H. K. Fathy, “Genetic identification and fisher identifiability analysis of the doyle–fuller–newman model from experimental cycling of a lifepo4 cell,” *Journal of Power Sources*, vol. 210, pp. 263–275, 2012.
- [30] B. S. Haran, B. N. Popov, and R. E. White, “Determination of the hydrogen diffusion coefficient in metal hydrides by impedance spectroscopy,” *Journal of Power Sources*, vol. 75, no. 1, pp. 56–63, 1998.
- [31] D. Di Domenico, A. Stefanopoulou, and G. Fiengo, “Lithium-ion battery state of charge and critical surface charge estimation using an electrochemical model-based extended kalman filter,” *Journal of dynamic systems, measurement, and control*, vol. 132, no. 6, 2010.
- [32] J. C. Forman, S. Bashash, J. L. Stein, and H. K. Fathy, “Reduction of an electrochemistry-based li-ion battery model via quasi-linearization and pade approximation,” *Journal of the Electrochemical Society*, vol. 158, no. 2, p. A93, 2010.
- [33] Q. Lai, S. Jangra, H. J. Ahn, G. Kim, W. T. Joe, and X. Lin, “Analytical derivation and analysis of parameter sensitivity for battery electrochemical dynamics,” *Journal of Power Sources*, vol. 472, p. 228338, 2020.
- [34] S. J. Moura, F. B. Argomedeo, R. Klein, A. Mirtabatabaei, and M. Krstic, “Battery state estimation for a single particle model with electrolyte dynamics,” *IEEE Transactions on Control Systems Technology*, vol. 25, no. 2, pp. 453–468, 2016.
- [35] J. Li, N. Lotfi, R. G. Landers, and J. Park, “A single particle model for lithium-ion batteries with electrolyte and stress-enhanced diffusion physics,” *Journal of The Electrochemical Society*, vol. 164, no. 4, p. A874, 2017.

- [36] H. E. Perez, X. Hu, and S. J. Moura, "Optimal charging of batteries via a single particle model with electrolyte and thermal dynamics," in *2016 American Control Conference (ACC)*, pp. 4000–4005, IEEE, 2016.
- [37] T. R. Tanim, C. D. Rahn, and C.-Y. Wang, "A temperature dependent, single particle, lithium ion cell model including electrolyte diffusion," *Journal of Dynamic Systems, Measurement, and Control*, vol. 137, no. 1, 2015.
- [38] R. Zhao, P. J. Kollmeyer, R. D. Lorenz, and T. M. Jahns, "A compact methodology via a recurrent neural network for accurate equivalent circuit type modeling of lithium-ion batteries," *IEEE Transactions on Industry Applications*, vol. 55, no. 2, pp. 1922–1931, 2018.
- [39] L. Lu, X. Han, J. Li, J. Hua, and M. Ouyang, "A review on the key issues for lithium-ion battery management in electric vehicles," *Journal of power sources*, vol. 226, pp. 272–288, 2013.
- [40] E. Chemali, P. J. Kollmeyer, M. Preindl, and A. Emadi, "State-of-charge estimation of li-ion batteries using deep neural networks: A machine learning approach," *Journal of Power Sources*, vol. 400, pp. 242–255, 2018.
- [41] D. Jiménez-Bermejo, J. Fraile-Ardanuy, S. Castaño-Solis, J. Merino, and R. Álvaro-Hermana, "Using dynamic neural networks for battery state of charge estimation in electric vehicles," *Procedia computer science*, vol. 130, pp. 533–540, 2018.
- [42] A. Nuhic, T. Terzimehic, T. Soczka-Guth, M. Buchholz, and K. Dietmayer, "Health diagnosis and remaining useful life prognostics of lithium-ion batteries using data-driven methods," *Journal of power sources*, vol. 239, pp. 680–688, 2013.
- [43] J. Wu, Y. Wang, X. Zhang, and Z. Chen, "A novel state of health estimation method of li-ion battery using group method of data handling," *Journal of Power Sources*, vol. 327, pp. 457–464, 2016.
- [44] R. Yang, R. Xiong, S. Ma, and X. Lin, "Characterization of external short circuit faults in electric vehicle li-ion battery packs and prediction using artificial neural networks," *Applied Energy*, vol. 260, p. 114253, 2020.
- [45] R. Xiong, Y. Pan, W. Shen, H. Li, and F. Sun, "Lithium-ion battery aging mechanisms and diagnosis method for automotive applications: Recent advances and perspectives," *Renewable and Sustainable Energy Reviews*, vol. 131, p. 110048, 2020.
- [46] S. S. Mansouri, P. Karvelis, G. Georgoulas, and G. Nikolakopoulos, "Remaining useful battery life prediction for uavs based on machine learning," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 4727–4732, 2017.
- [47] J. Á. Antón, P. G. Nieto, F. de Cos Juez, F. S. Lasheras, M. G. Vega, and M. R. Gutiérrez, "Battery state-of-charge estimator using the svm technique," *Applied Mathematical Modelling*, vol. 37, no. 9, pp. 6244–6253, 2013.

- [48] I.-H. Li, W.-Y. Wang, S.-F. Su, and Y.-S. Lee, “A merged fuzzy neural network and its applications in battery state-of-charge estimation,” *IEEE Transactions on Energy Conversion*, vol. 22, no. 3, pp. 697–708, 2007.
- [49] A. Zenati, P. Desprez, and H. Razik, “Estimation of the soc and the soh of li-ion batteries, by combining impedance measurements with the fuzzy logic inference,” in *IECON 2010-36th Annual Conference on IEEE Industrial Electronics Society*, pp. 1773–1778, IEEE, 2010.
- [50] P. Khumprom and N. Yodo, “A data-driven predictive prognostic model for lithium-ion batteries based on a deep learning algorithm,” *Energies*, vol. 12, no. 4, p. 660, 2019.
- [51] X. Hu, S. E. Li, and Y. Yang, “Advanced machine learning approach for lithium-ion battery state estimation in electric vehicles,” *IEEE Transactions on Transportation electrification*, vol. 2, no. 2, pp. 140–149, 2015.
- [52] F. Yang, W. Li, C. Li, and Q. Miao, “State-of-charge estimation of lithium-ion batteries based on gated recurrent neural network,” *Energy*, vol. 175, pp. 66–75, 2019.
- [53] S. Park, D. Zhang, and S. Moura, “Hybrid electrochemical modeling with recurrent neural networks for li-ion batteries,” in *2017 American Control Conference (ACC)*, pp. 3777–3782, IEEE, 2017.
- [54] J. Newman and K. E. Thomas-Alyea, *Electrochemical systems*. John Wiley & Sons, 2012.
- [55] J. Marcicki, M. Canova, A. T. Conlisk, and G. Rizzoni, “Design and parametrization analysis of a reduced-order electrochemical model of graphite/lifepo4 cells for soc/soh estimation,” *Journal of Power Sources*, vol. 237, pp. 310–324, 2013.
- [56] G. Cybenko, “Approximation by superpositions of a sigmoidal function,” *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.
- [57] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning internal representations by error propagation,” tech. rep., California Univ San Diego La Jolla Inst for Cognitive Science, 1985.
- [58] A. M. Schäfer and H.-G. Zimmermann, “Recurrent neural networks are universal approximators,” *International journal of neural systems*, vol. 17, no. 04, pp. 253–263, 2007.
- [59] R. J. Williams and J. Peng, “An efficient gradient-based algorithm for on-line training of recurrent network trajectories,” *Neural computation*, vol. 2, no. 4, pp. 490–501, 1990.
- [60] J. Brownlee, “A gentle introduction to backpropagation through time.” <https://machinelearningmastery.com/gentle-introduction-backpropagation-time/>, August 2019. (Accessed on 07/21/2020).

- [61] I. Sutskever, *Training recurrent neural networks*. University of Toronto Toronto, Canada, 2013.
- [62] Y. Bengio, P. Simard, and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult,” *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.
- [63] R. Pascanu, T. Mikolov, and Y. Bengio, “On the difficulty of training recurrent neural networks,” in *International conference on machine learning*, pp. 1310–1318, 2013.
- [64] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [65] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using rnn encoder-decoder for statistical machine translation,” *arXiv preprint arXiv:1406.1078*, 2014.
- [66] D. Britz, A. Goldie, M.-T. Luong, and Q. Le, “Massive exploration of neural machine translation architectures,” *arXiv preprint arXiv:1703.03906*, 2017.
- [67] F. Chollet *et al.*, “Keras.” <https://keras.io>, 2015.
- [68] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015. Software available from tensorflow.org.
- [69] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, “CasADi – A software framework for nonlinear optimization and optimal control,” *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.
- [70] S. Jangra, “A framework for lithium-ion battery simulation and sensitivity analysis,” Master’s thesis, University of California, Davis, 2019.
- [71] J. Brownlee, “How to prepare your data for machine learning in python with scikit-learn.” <https://machinelearningmastery.com/prepare-data-machine-learning-python-scikit-learn/>, December 20129. (Accessed on 08/19/2020).
- [72] J. Brownlee, “How to configure the learning rate when training deep learning neural networks.” <https://machinelearningmastery.com/learning-rate-for-deep-learning-neural-networks/>, August 2019. (Accessed on 08/24/2020).

- [73] J. Brownlee, “Using learning rate schedules for deep learning models in python with keras.” <https://machinelearningmastery.com/using-learning-rate-schedules-deep-learning-models-python-keras/>, September 2019. (Accessed on 08/24/2020).
- [74] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [75] J. L. Elman, “Finding structure in time,” *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.