Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Hannah Hansen

CNN-based Remaining Useful Life Prediction of Lithium Ion Batteries Using Fast-Rate Incremental Capacity

Master's thesis in Cybernetics and Robotics
Supervisor: Professor Sebastien Gros
Co-supervisor: Dr. Eibar Flores & Dr. Michael Gerhardt
June 2023

NTNU
Norwegian University of
Science and Technology

Hannah Hansen

CNN-based Remaining Useful Life Prediction of Lithium Ion Batteries Using Fast-Rate Incremental Capacity

Master's thesis in Cybernetics and Robotics
Supervisor: Professor Sebastien Gros
Co-supervisor: Dr. Eibar Flores & Dr. Michael Gerhardt
June 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Abstract

Lithium-ion batteries (LIBs) are essential in transitioning to clean and renewable energy, serving as energy storage and power sources in transportation. However, battery degradation poses a significant challenge to their durability and performance. Due to the inherent complexity of the degradation process, machine learning methods have emerged as a promising modeling approach for SOH estimation and lifetime prediction due to their ability to model complex processes solely from data. Accurate estimation of the state of health (SOH) is essential to ensuring safe and effective operation, potentially revealing new insights into the underlying degradation process.

In this work, we propose a shallow 1D convolutional neural network (CNN) architecture to leverage information in fast-rate incremental capacity (IC) curves for predicting the remaining useful life (RUL) of LIBs, both in units of cycles and normalized to the total lifetime. Additionally, we look into the decision-making process of the CNN by identifying voltage regions and input patterns that correlate with reduced RUL.

The CNN demonstrates a root mean squared error (RMSE) and a mean absolute deviation (MAD) of 171 cycles and 131 cycles, respectively, when predicting RUL. For normalized RUL, the predictive performance improves, achieving an RMSE and MAD of 0.075 and 0.057, respectively. Despite the shallow architecture and limited input features, the CNN demonstrates the ability to connect features in the IC curves to remaining life. Furthermore, The CNN identifies consistent patterns indicating cell degradation, particularly related to IC peak reductions in voltage regions associated with graphite staging. The patterns can be attributed to reduced accessible capacity in this voltage window. Furthermore, the appearance of peaks at lower voltages may also indicate decreased RUL.

Overall, this work demonstrates the effectiveness of CNN-processed fast-rate IC difference curves for lifetime predictions of LIBs. The findings contribute to a better understanding of the CNN's decision-making process and provide insights into degradation patterns in LIBs.

Sammendrag

Lithium-ione-batterier spiller en sentral rolle i det grønne skiftet gjennom bruk i energilagring og elektrifisering av transportsektoren. Det er midlertidig utfordringer knyttet til forringelse av lithium-ione-batterier, noe som medfører redusert ytelse og levetid. Dette er en høyst kompleks prosess, noe som gjør maskinlæringsmodeller spesielt egnet for å estimere helsetilstand og levetid til batterier. Nøyaktig estimering og modellering av batteriets helsetilstand er avgjørende for sikker og effektiv drift, samtidig som det åpner for økt innsikt i underliggende årsaker til batteriforringelse.

Dette arbeidet presenterer et enkelt 1D konvolusjonelt nevralt nettverk (CNN) for prediksjon av gjenværende levetid, både i antall ladesykler og normalisert mot total levetid, basert på informasjon fra inkrementell kapasitet (IC) under hurtigladning. Gjennom analyse av beslutningsprosessen til CNN-et undersøkes hvilke spenningsområder og inngangssignaler som modellen relaterer til redusert gjenværende levetid.

CNN-et oppnår en RMSE og MAD på henholdsvis 171 og 130 sykler. Ved prediksjon av normalisert gjenværende levetid, er RMSE og MAD på henholdsvis 0,075 og 0,057. Til tross for en enkel arkitektur og begrenset inngangssignal, demonstrerer modellen tydelig evnen til å identifisere relevante sammenhenger mellom IC kurver og levetid. Videre undersøkelse tyder på at CNN-et assosierer redusert levetid med IC-reduksjon i spenningsregionen forbundet med interkalering av anoden. Mønstrene kan dermed tilskrives redusert tilgjengelig kapasitet i dette spenningsvinduet. I tillegg ser også signal ved lavere spenninger ut til indikere redusert RUL.

Samlet sett demonstrere arbeidet hvordan CNN-prosesserte IC kurver fra hurtiglading kan benyttes for levetidsestimering. Funnene bidrar til en bedre forståelse av beslutningsprosessen og gir innsikt i forringelsesprosessen.

Preface

This thesis concludes my Master's degree in Cybernetics and Robotics at the Norwegian University of Science and Technology (NTNU).

I would like to thank my supervisor Professor Sebastien Gros from the Department of Engineering Cybernetics at NTNU for helpful discussions and constructive feedback. I am also deeply grateful for the guidance and support provided by my supervisors at SINTEF, Dr.Eibar Flores and Dr. Michael Gerhardt, throughout the work with this thesis. Their expertise has been invaluable in introducing me to the field of battery technology and helped me shape the course of my thesis. Working alongside such competent individuals has been a privilege. I would like to extend a special appreciation to Eibar for his encouragement and unwavering support at every step of this process.

I would also like to thank my significant other, Alexander, for being a constant source of strength and motivation, and my dear friend Kristine, whose steadfast companionship has enriched my time in Trondheim with joy, laughter, and countless engaging discussions. Thank you to Kjersti, for all her help and for keeping my stomach full. And, of course, Mamma og Pappa, for their love and support, and for inspiring me to choose this path.

Trondheim, June 2023
Hannah Hansen

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

BMS	Battery management system
CNN	Convolution neural network
DNN	Deep neural network
EOL	End of life
GAN	Generative adversarial networks
GHG	Greenhouse gasses
IC	Incremental capacity
LAM	Loss of active material
LLI	Loss of lithium inventory
LIB	Lithium-ion battery
LTSM	Long short-term memory
MAD	Mean absolute divitation
ML	Machine learning
MLP	Multilayer perceptron
MSE	Mean squared error
ReLU	Rectified linear unit
RMSE	Root mean squared urror
SOH	State of health

1 Introduction

1.1 Background and Motivation

The ongoing climate change is one of the greatest challenges facing humanity. Global warming has been indisputably caused by human activities, primarily due to the release of greenhouse gases (GHG) [1]. Failing to implement substantial measures to reverse these trends might have catastrophic consequences. For example, the rising temperatures threaten life on land and in the sea [2]. Ice melting contributes to rising sea levels, posing a direct threat to coastal and island communities. As temperatures continue to rise, we can expect an increase in extreme weather events, resulting in more frequent and intense storms, extended droughts, and more wildfires. These climate changes not only endanger human lives and livelihoods but also threaten ecosystems and biodiversity. Additionally, climate change is our most significant health threat [3]. Air pollution has a direct and adverse effect on our health, and extreme weather, forced displacement, and increased hunger resulting from climate change also pose severe health challenges. Environmental factors are estimated to contribute to the loss of 13 million lives every year [2]. Fossil fuels contribute to approximately 75% of the global GHG emission, of which the transport sector accounts for one-third [2]. Mitigating climate change requires a substantial reduction in GHG emissions.

Batteries play an essential role in enabling the transition away from fossil-based energy sources. Renewable energy sources such as wind and solar power are inherently time-varying and intermittent and rely on batteries for efficient energy storage. A battery converts chemical energy to electric energy with high efficiency and without GHG emissions [4]. The portable nature of batteries makes them suitable as the power source for electric vehicles. Lithium-ion batteries (LIBs) are the most prominent choice for portable energy sources due to their high energy density and long cycle life [5].

However, the widespread adoption of LIBs faces challenges in terms of performance, cost-effectiveness, and durability. Over time and with usage, the battery degrades, reducing its ability to store energy and meet power demands. Eventually, this degradation results in the battery reaching its end of life (EOL) [6]. Enhancing the durability of LIBs contributes to maximizing their utilization and improves their competitiveness against the combustion engine. Most batteries get thrown away after they have reached the EOL. However, there is a growing interest in exploring second-life applications for battery reuse, such as using retired batteries from electric vehicles for energy storage purposes [7]. Enabling second-life applications contributes to lower costs and meeting the increasing battery demand. By extending the useful life of batteries we can address sustainability concerns associated with the extraction of raw materials, manufacturing of LIBs, and

disposal of LIB waste.

In this context, the state of health (SOH) estimation and EOL estimation of the battery is essential. SOH is commonly determined by measuring the loss of charge capacity over time, while EOL is defined when the capacity decreases below a specific threshold. Monitoring SOH and estimating the remaining useful life (RUL) enables safe and efficient control and facilitates proper maintenance of the battery, and is of particular importance in battery management systems (BMSs) [8]. Accurate lifetime predictions in the early stages of battery life may also accelerate the development of battery technology by simplifying the validation processes [9]. Additionally, EOL estimation is important for ensuring usefulness in a potential second-life application [10]. Lifetime predictions and SOH estimations facilitate optimal operation, second-life applications, and more efficient development of batteries.

SOH and EOL can be estimated with models describing the life of a battery cell. Modeling of cell lifetime can broadly be divided into two approached. Physics-based modeling aim to simulate the system using physical principles and equations. In the context of battery degradation, this requires understanding the underlying electrochemical phenomena, degradation mechanisms, and system parameters. Physics-based models often consider the effect of the growth of solid electrolyte interphase (SEI) [11, 12, 13], and particle cracking [12, 14], as these mechanisms are among the most dominant in degradation of LIBs [6]. Because such models rely on physical phenomena, they also provide valuable insight into the process. Due to their descriptive nature, physics-based models have the advantage of being generalizable. However, as degradation is a complicated and highly intercorrelated process, developing accurate physics-based models is challenging and even impossible when the physics and chemistry of such degradation processes are unknown.

Data-driven modeling, on the other hand, does not require explicit knowledge about the underlying system. Instead, the approach relies on statistical techniques and computational algorithms to learn the relationship from available data. This system-agnostic property makes data-driven modeling versatile and particularly powerful when dealing with complex and non-linear systems. Machine learning (ML) techniques have been widely adapted for lifetime predictions of batteries, including algorithms such as Gaussian processes [15, 16], state vector machines [17, 18] and neural networks [19, 20, 21, 22]. ML models often require extensive datasets and can be computationally heavy. While such models have demonstrated excellent predictive performance, their interpretability may be limited, making it challenging to extract meaningful insights about the internal degradation processes.

One way to improve interpretability is to learn from explainable battery signals. Incremental capacity (IC) analysis, which involves analyzing the slope of the voltage vs. capacity curve during standard cycling procedures, offers valuable insights into the underlying electrochemical processes at the electrode level. The IC curves, which represent the capacity change as a function of voltage, serve as an indicator of the battery state. IC analysis can also be used for degradation diagnostics [23]. Furthermore, implementation of IC analysis is feasible in BMSs, enabling in situ monitoring and diagnostics

[24]. However, IC analysis typically involves slow diagnostics cycling for obtaining the IC curve [25, 26], which is inconvenient for onboard BMS applications. The use of fast-rate IC curves mitigates the need for time-consuming diagnostic cycling. This study aims to leverage the electrochemical data in fast-rate IC curves for lifetime predictions.

The IC signal resembles electrochemical spectra by exhibiting features like peaks and valleys. The convolutional neural network (CNN) is a promising approach for processing this kind of spatial data, as evident by their application in spectroscopy analysis [27, 28, 29]. CNNs have also demonstrated excellent performance on tasks such as image classification and object detection due to their ability to extract features in spatially structured data [30].

CNN is often referred to as a "black-box" model due to the lack of interpretability of the inner workings of the model. Understanding the model's decision-making process may be challenging, limiting what we can learn about the underlying system being modeled and introducing skepticism to ML predictions. Understanding how a model arrives at its conclusions is crucial for justifying the use of such models, and ensuring their safety and reliability. Explainability is essential for reducing human bias in models and meeting legal and ethical requirements. Incorporating explainability into ML models enhances transparency and accountability and allows us to learn from the models. An interpretable model learning from explainable electrochemical signals would render accurate and understandable predictions.

1.2 Contribution and Research Objectives

1.2.1 Contribution

Data-driven modeling of lithium-ion batteries is a widely used strategy for battery lifetime prediction and SOH monitoring [25, 26]. Although these areas have been extensively researched in the literature, the focus tends to be on predictive performance. Consequently, this often results in deep and complex machine-learning models with intricate input features. While such models might exhibit high accuracy, they are usually black-box approaches providing limited insight into the underlying processes. Improving the understanding of degradation and the decision-making process of the model is crucial for advancing the knowledge in this field and justifying the use of machine learning models. This work proposes a strategy for utilizing a simple CNN for processing electrochemical data in the form of fast-rate IC curves for lifetime predictions and further aims to establish a link between CNN behavior and degradation observables.

1.2.2 Objectives

To guide this study, a set of research objectives are formulated. In addition, research questions leading to the research objective are presented.

Primary Objective: The primary objective of this work is to design a simple CNN for processing electrochemical data for lifetime predictions and establishing a link between

CNN behavior and degradation observables.

Secondary Objectives:

- Leverage information embedded in fast-charging incremental capacity curves for predicting remaining useful life
- Design a CNN architecture that balances prediction accuracy and interpretability
- Identify regions and patterns in the input that are important for RUL predictions
- Explore the connections between the identified regions and patterns in the IC curves and known degradation processes in lithium-ion batteries

Based on the objectives, we formulate the following research questions:

- What is the utility of using IC curves for RUL predictions?
- How does the proposed strategy compare in terms of predictive performance to deep and complex machine learning models?
- What regions and patterns in the input do the CNN correlate with low RUL?
- What insights into the underlying processes and degradation observables can be gained through the analysis of the CNN?

1.3 Structure of the Thesis

The thesis is organized into five chapters. Chapter 1 is the introduction, where the background for carrying out this study is motivated. Objectives and related research questions are also stated. Relevant theory is presented in Chapter 2. Section 2.1 contains a brief introduction to the working principles of batteries, degradation, and IC analysis, and Section 2.2 covers theoretical and practical aspects of deep neural networks and CNNs in particular. Next, the method is presented in Chapter 3. This chapter includes the preparation of data and input features, architecture design, and the training and validation process. The results are presented and discussed in Chapter 4. Lastly, the conclusion and suggestions for future work follow in Chapter 5.

2 Theory

This chapter provides an introduction to the relevant theory. We begin by discussing the basics of batteries, including working principles, degradation mechanisms, and IC analysis. We then proceed to the principles of deep neural networks and CNNs. Section 2.2 is partly based on the theory from a prior self-written report *Risk-based Convolutional Perception Models for Collision Avoidance in Autonomous Marine Surface Vessels using Deep Reinforcement Learning.*

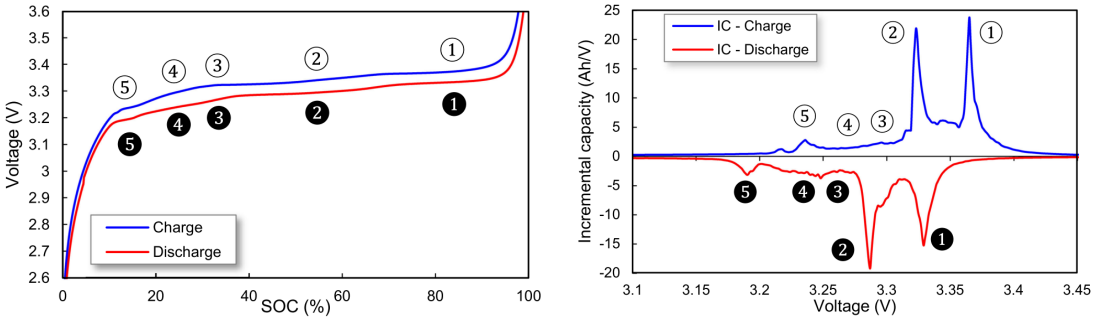
2.1 Batteries

2.1.1 Battery Working Principles

A rechargeable Li-ion battery consists of one or more electrochemical cells converting chemical energy into electrical energy through the reversible process of (de)intercalation of Li^+ into the electrodes [4, 31]. The cell comprises two electrodes, namely the anode and the cathode, which are separated by an electrolyte. During charge, the anode is lithiated by cations released from the cathode. The intercalation of Li^+ into the anode is a non-spontaneous reaction and is driven by an external voltage. The electrolyte transfers the ions while the electrons are forced outside the cell through an electric circuit. This process stores energy in the cell. Discharge is the reverse process of spontaneous intercalation of Li^+ back into the cathode. The electrons flow back through the external circuit, allowing the stored energy in the battery to be converted into useful work or power.

Graphite is the dominating choice of active anode material for commercial LIBs [32]. Graphite consists of graphene layers arranged in a stacked structure. The intercalation of ions into the graphite host lattice takes place in a layer-wise fashion, known as the graphite staging mechanism. This mechanism results in a periodic repetition of intercalated layers, with each stage denoting the number of graphene layers between intercalated layers, such that stage n has n layers of graphene separating each intercalated layer [32]. In LIBs, graphite intercalation follows stages 1L, 4, 3, 2L, 2, and 1, where stage 1 is the maximum lithiated structure. The letter L refers to an intermediate stage where the lithium ions are not perfectly ordered in the layers [32].

Lithium iron phosphate (LFP), $LiFePO_4$, is a common cathode active material in a wide range of applications, as it has excellent cycle life, high safety, low cost, and fast charging times [5]. While (de-)intercalation of graphite occurs at multiple stages at different voltages, the $FePO_4$–$LiFePO_4$ phase transformation of the LFP occurs around 3.5V vs. Li and appears as a single, broad plateau in the voltage discharge curve [31]. Figure 2.1a



(a) Voltage curves

(b) IC curves

Figure 2.1: Voltage curve and IC curve of a graphite||LFP cell at cycled at $C/25$. The numbers indicate the lithium staging phenomena on the graphite anode. Reprinted from [24]

[24] displays the voltage curve for the charge and discharge cycle for a graphite||LFP cell. The staging phenomena are observable as intermediate changes in the slope and are labeled from 1-5, where 1 corresponds to the transition between stages 1 and 2.

2.1.2 Degradation Modes

The degradation of LIBs is observable through a decrease in their ability to store energy and meet power demands [6]. The degradation of the Li-ion cell results of one or more degradation modes being present. The degradation modes are caused by various degradation mechanisms, which refer to undesirable physical and chemical phenomena occurring within the cell.

There are three commonly reported degradation modes [6]: Loss of lithium Inventory (LLI), loss of active material of the negative electrode ($\mathrm{LAM_{NE}}$), and loss of active material of the positive electrode ($\mathrm{LAM_{PE}}$). LLI refers to the reduction in the available amount of active lithium in a cell. Lithium ions can be consumed by parasitic reactions, such as SEI growth, lithium plating, and electrolyte decomposition, or trapped inside the active material. As the amount of available lithium decrease, the capacity of the cell gradually fades over time. LAM involves a reduction in the active material on the electrodes that are available for electrochemical activity. This can be caused by electrode particle cracking from operational stress and loss of electric contact. Additionally, lithium plating can block the active sites in the negative electrode, and structural disordering of the positive electrode contributes to $\mathrm{LAM_{PE}}$.

The degradation mechanisms, eventually leading to an observable effect on cell performance, are highly interconnected. For instance, lithium plating not only consumes lithium ions, leading to LLI, but also contributes to $\mathrm{LAM_{NE}}$ by blocking active sites. Particle cracking leads to LAM for both electrodes but also triggers lithium-consuming side reactions by exposing electrode material to the electrolyte. The degradation process is intricate and complex, involving multiple interrelated mechanisms. Therefore, ML methods have emerged as a promising approach for degradation modeling.

2.1.3 Incremental Capacity Analysis

The incremental capacity (IC), also known as differential capacity, provides a useful representation of the charge and discharge processes in a cell. The incremental capacity describes the change in capacity, ΔQ, over a fixed voltage step ΔV, $\frac{\Delta Q}{\Delta V}$. Peaks in the IC curves indicate regions where the capacity changes rapidly with respect to voltage. Each peak is characterized by its position, intensity, and shape, and is a result of the electrochemical reactions occurring in the positive and negative electrodes [33].

Figure 2.1b depicts the IC curves for an LPF cell for charge (blue) and discharge (red) at a C-rate of C/25. The C-rate represents the current at which the cell is charged or discharged relative to its nominal capacity, such that charging at 1C means that the battery is charged from 0% to 100% in one hour. The staging plateaus in the voltage curve in fig. 2.1a lead to an increased change in capacity per unit of voltage, resulting in the appearance of peaks when the capacity curve is differentiated. These peaks can be attributed to the graphite staging mechanism in the anode, as the intercalation into the LFP cathode occurs as a single broad plateau around 3.5V vs. Li [23]. Each peak is labeled based on the associated staging phenomenon.

IC analysis is recognized as a powerful technique for battery monitoring and battery diagnostics, as the peaks of the IC curve provide information about the electrochemical processes occurring within the electrodes [24]. Changes in the electrochemical reactions affect the position, shape, and intensity of the IC peaks, and the evolution of these peaks gives insight into the degradation of the cell.

The degradation modes have specific signatures in the IC curves, enabling diagnostics of degradation modes from IC analysis [23]. For instance, LLI primarily affects peaks 1 and 5 as numbered in fig. 2.1b. LLI manifests in the discharge IC curve as intensity reduction of peak 1 and shift of peak 5 to higher voltages. A thorough analysis of how other degradation modes manifest in the IC curves can be found in [23].

In addition to being considered an advanced technique to identify degradation modes in battery cells, IC analysis also has the advantage of being an in-situ technique [24]. The IC curves can be obtained and analyzed without removing the battery from its original place or disrupting the battery's normal operation too much. This makes this technique feasible in on-board battery management systems (BMSs). Battery prognosis facilitates effective decision-making, maintenance planning, and performance optimization, making such implementations valuable.

One of the main limitations of IC analysis is that it requires IC curves obtained from low current rates. The link between the IC peaks and degradation modes in LFP cells is documented for low-rate cycling [23, 24, 33]. IC analysis presents a more challenging task under higher cycling rates because the peaks merge together due to the convolution of kinetics and open-circuit behavior [9]. This effect makes it more challenging to relate IC peaks to specific processes, making identifying degradation modes difficult. As demonstrated in [33], the peaks that are clearly visible in fig. 2.1b combine into a single broad peak when cycled at 1C. The shape and position of the curve change considerably, and assigning peaks to specific processes is not trivial. However, the IC curve is

still a product of the underlying electrochemical processes on the electrodes. Although diagnostics may not be straightforward, the IC curve might still provide useful insight into the state of the cell under fast cycling as well.

2.2 Deep Neural Networks

Deep neural networks (DNNs) have shown excellent performance in applications like computer vision, speech recognition and natural language processing, in some cases exceeding human performance [34]. The key to the usefulness of DNNs is their ability to extract features from high dimensional data, identifying patterns and underlying structures.

DNNs are a type of machine learning approach that generally falls under the category of supervised learning. In supervised learning, the model learns to make predictions based on a set of labeled target values, that is, the desired model behavior is known by the designer. This characteristic distinguishes supervised learning from the two other paradigms of ML methods, namely unsupervised learning and reinforcement learning. Unsupervised learning involves learning underlying structures and patterns in an unlabeled dataset. In reinforcement learning, the model interacts with an environment. The desired behavior is not explicitly defined, and the model learns from trial and error based on feedback from an overarching objective.

2.2.1 Artificial Neurons

The basic unit of a neural network is the artificial neuron. The artificial neuron is designed to mimic the behavior of a biological neuron. Each neuron receives input from a number of other neurons and produces a single output. The neuron calculates the weighted sum of its inputs and passes it through a non-linear activation function to produce the output. This is analogous to the activation of a biological neuron. A biological neuron may also get inputs from several neighboring neurons, and passes the signal forward on the network of neurons only if the combined input signals are strong enough. In mathematical terms, the output a_k of the kth neuron is

$$a_k = \sigma(z) = \sigma\left(\sum_{i=0}^{n} w_{ki}x_i + b_k\right), \tag{2.2.1}$$

given inputs x_i with associated weights w_{ki}, bias term b_k, and activation function σ. The current standard activation function σ is the rectified linear unit (ReLU) activation function,

$$\sigma(z) = \begin{cases} z, & z > 0 \\ 0, & \text{otherwise,} \end{cases} \tag{2.2.2}$$

or variations of this function [35]. The neurons are structured in layers connecting the input layer to the output layer. A neural network consisting of multiple such hidden layers is called a deep neural network. This concept is illustrated in fig. 2.2, displaying

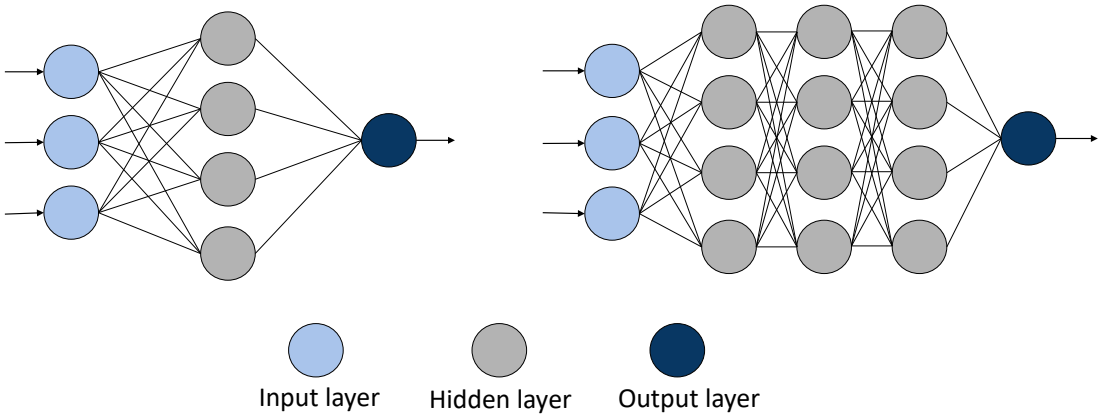


Figure 2.2: Illustration of an artificial neural network with one hidden layer (left) and a deep neural network with multiple hidden layers (right).

examples of fully connected feed-forward neural networks.The neurons are connected in a feed-forward fashion, propagating the information forward through the network.

2.2.2 Neural Networks as Universal Approximators

The DNN can be formulated as a mapping $\mathcal{F}$

$$\hat{y} = \mathcal{F}(X; \theta), \tag{2.2.3}$$

transforming the input X to some output $\hat{y}$, given parameters θ consisting of the weights and the biases. In fact, neural networks are universal approximators. A feed-forward neural network can approximate any measurable function to the desired accuracy given a sufficient number of hidden units [36]. This property makes DNNs well-suited for modeling complex, non-linear systems. However, there is no guarantee of successfully training them. Inadequate data, insufficient numbers of hidden units, poor learning algorithms, or the lack of a deterministic relationship between input and output can limit the performance of the DNN [36].

2.2.3 Backpropagation and Gradient Descent

Training of a neural network is done by minimizing the error between the labeled target y and the predictions $\hat{y}$ of the DNN. The predictions are performed by doing a forward pass, that is, propagating the input through the network. The performance is then evaluated with a loss function that compares the predictions with the target. The choice of loss function depends on the problem at hand. A typical loss function in regression

problems is the mean squared error (MSE)

$$\mathcal{L}(y, \hat{y}) = \frac{1}{2N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2, \tag{2.2.4}$$

where N is the number of samples. y_i is the true value and $\hat{y}_i$ is the corresponding predicted value.

The process of updating the model parameters is called backpropagation. By differentiating the loss with respect to the learnable parameters, we can determine how to modify the weights and biases to reduce the loss and thus improve performance. For the sake of simplicity, let us consider a neuron in layer k with output $a_k = \sigma(z_k)$ where $z = w_k a_{k-1}$, omitting the bias term and assuming a single input a_{k-1}. Differentiating the loss with respect to the weights w_k gives

$$\frac{\partial \mathcal{L}}{\partial w_k} = \frac{\partial \mathcal{L}}{\partial a_k} \frac{\partial a_k}{\partial z_k} \frac{\partial z_k}{\partial w_k} \triangleq \delta_k \frac{\partial z_k}{\partial w_k}, \tag{2.2.5}$$

such that δ_k is the derivative of the loss with respect to z_k. Assuming that k is the last layer and considering a single prediction, we can find the derivative of $\frac{\partial \mathcal{L}}{\partial a_k}$ directly from (2.2.4). With the definitions of a_k and z_k, we can rewrite (2.2.5) as

$$\frac{\partial \mathcal{L}}{\partial w_k} = (y - a_k)\sigma'(z_k)a_{k-1}. \tag{2.2.6}$$

This is straightforward to calculate, as a_{k-1}, a_k, and z_k are given by the forward pass, and y is the labeled target. If we now consider the neuron to be in a hidden layer, a_k will be the input of neuron $k+1$. Following the same notation and logic as in equation (2.2.5) and (2.2.6), the derivative becomes

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial w_k} &= \frac{\partial \mathcal{L}}{\partial a_k} \frac{\partial a_k}{\partial z_k} \frac{\partial z_k}{\partial w_k} \\ &= \frac{\partial \mathcal{L}}{\partial a_{k+1}} \frac{\partial a_{k+1}}{\partial z_{k+1}} \frac{\partial z_{k+1}}{\partial a_k} \frac{\partial a_k}{\partial z_k} \frac{\partial z_k}{\partial w_k} \\ &= \delta_{k+1} w_{k+1} \sigma'(z_k) a_{k-1} \\ &= \delta_k a_{k-1} \end{aligned} \tag{2.2.7}$$

Thus, the weight update is dependent on the local gradient $\frac{\partial z_k}{\partial w_k} = a_{k-1}$, which is given from the forward pass, and the upstream gradient $\frac{\partial \mathcal{L}}{\partial z_k} = \delta_k$, which is calculated in a recursive manner from the previous step of the backpropagation. The weights are updated sequentially, where the gradients flow from output to input, passing information backward in the structure.

A neural network usually consists of multiple hidden layers with multiple neurons per layer. However, the notation is only slightly more complicated than in the simple toy example above. Let l index one of a total of L layers. w_{jk}^l is the weight from neuron k in layer $l-1$ to neuron j in layer l, and b_j^l is the corresponding bias term. The

backpropagation can be summed up by the following equations

$$\delta^L = \nabla_{a^L}\mathcal{L} \odot \sigma'(z^L) \tag{2.2.8a}$$

$$\delta^l = ((w^{l+1})^\top \delta^{l+1} \odot \sigma'(z^1) \tag{2.2.8b}$$

$$\frac{\partial \mathcal{L}}{\partial b_j^l} = \delta_j^l \tag{2.2.8c}$$

$$\frac{\partial \mathcal{L}}{\partial w_{jk}^l} = \delta_j^l a_k^{l-1} \tag{2.2.8d}$$

where ∇ is the del operator and $\odot$ is the elementwise multiplication operator. The reader is referred to [37] for the complete derivation of the equations. Similarly to the toy example, the full model utilizes the local and upstream gradients. The upstream gradients propagate backward in the network, providing a simple and elegant procedure to calculate the individual gradients at every neuron.

Using gradient descent with learning rate λ, the updated parameters θ become

$$\theta \leftarrow \theta - \lambda \nabla_\theta \mathcal{L}, \tag{2.2.9}$$

there $\nabla_\theta \mathcal{L}$ is the gradient of the loss function with respect to all parameters.

There are three main approaches when performing gradient descent. The first is batch gradient descent, or simply gradient descent, where the entire dataset is considered to calculate the gradient. This can be computationally expensive, especially for large datasets. However, because the algorithm uses the entire dataset, it is often more accurate than the other gradient descent algorithms. The stochastic gradient descent, on the other hand, uses only a single sample at each iteration. This makes the algorithm more effective but might be less accurate and noisier. The standard approach in DNN training is the mini-batch gradient descent. With this approach, a subset of the training data is used at each iteration. This creates a middle ground between the two extremes, balancing computationally efficiency and accuracy. There are several extensions of the gradient descent algorithm. The Adaptive Moment Estimation (Adam) optimizer [38] is a widely used optimizer for training DNNs and has demonstrated good performance in a variety of DNN tasks. The regularization strategy of the Adam algorithm has been further improved in the AdamW optimizer [39]. For additional examples, the reader is referred to [40].

2.2.4 Convolutional Neural Networks

The composition of the hidden layers and how they are connected give rise to different types of DNNs. Convolutional neural networks (CNN) are considered the most important DNN architecture for tasks such as image recognition, object detection, and classification due to their ability to extract features from spatially structured data [30].

The CNN is a feed-forward network, where the main component of a CNN is the convolutional layer. CNNs have sparse connectivity, unlike the fully connected layers, where

each node is connected to every node in the neighboring layers with individual weights. The nodes in a convolutional layer are organized in small kernels or filters. The kernel is slid over the input, the output being the convolution between the kernel and the input at each location. The kernel weights are shared over the input, meaning that the weights of a kernel are the same at every location. This characteristic makes the convolution a local operation. The kernel learns to encode spatial structures in the input data. As a kernel is slid along the input, it will look for a specific feature at different locations of the input. This can be low-level features such as edges or corners. With multiple kernels at each layer, several patterns might be identified. As the input is passed through multiple layers of such filters, the abstraction level of the features increases. This enables the model to identify complex objects and structures.

The kernel size, stride, padding, and number of channels define a convolutional layer. The kernel size determines the area of the input participating in each convolutional operation. The stride is the number of elements by which the filter moves over the input at a time. A stride of size $n > 1$ reduces the output size, as the convolutional operation is performed at every n-th input element. Padding is a technique that adds additional elements, typically zeros, at the borders of the input before performing the convolution operation. Using a padding of size $(K-1)/2$, where K is the kernel size, a stride of 1 implies that the output size equals the input size. When a convolutional layer consists of multiple kernels, these are commonly referred to as channels.

Pooling layers are typically used in between convolutional layers in a CNN. They are used to downsample the input, and make the network more invariant to small translations or deformations in the input [40]. One common type of pooling operation is max pooling, which takes the maximum value within a neighborhood of the input and passes it to the next layer. If the value is shifted due to small translations but still within the neighborhood, the output might be left unchanged. The max pooling operation allows the network to retain only the most important information from each region of the input while discarding less important details.

Another common component in a CNN and other types of DNNs is batch normalization [41]. The batch normalization standardizes the activations within each mini-batch using the transformation

$$\bar{x_i} = \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}, \tag{2.2.10}$$

where ϵ is a small number introduced for numerical stability and $\mu_{\mathcal{B}}$ and $\sigma_{\mathcal{B}}$ are the mean and standard deviation of the batch, respectively. This normalization reduces the internal covariance shift, which refers to the change in the distribution of activations as the model parameters update during training. Reducing this covariance shift leads to more stable parameter growth, more generalizable models, and faster convergence during training [41].

A CNN typically consists of repeating blocks of convolutional layers, pooling layers, and batch normalization in various combinations. This part of a CNN is referred to as the feature extractor as it transforms the input into a feature-based representation. This

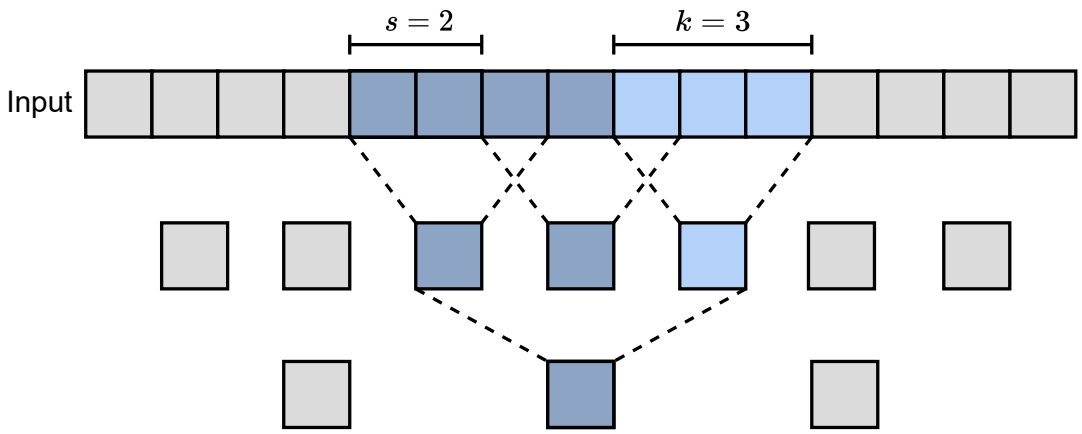


Figure 2.3: Illustration of receptive fields in the example of a CNN with two 1D convolutional layers with kernels of size 3, strides of 2, and no padding. Note how the receptive field widens as information propagates through the network.

representation is called an embedding and encapsulates the learned features and encodes the essential characteristics of the input. Each of the output channels of the feature extractor produces a feature vector, that together composes the embedding vector. In addition to the feature extractor, the CNN typically has a module that transforms the embedding vector into the final output through a set of fully connected layers. In the case of regression, this module is commonly known as the regression head.

Receptive Fields

The receptive field is the part of the input space visible for a particular node in a layer. Only the input within the receptive field influences the activation of a node. Consequently, the node is oblivious to the rest of the input. In the first layer, each node perceives the local region covered by the first kernel. As we move to deeper layers, each node incorporates the receptive fields of the previous layer's nodes. As the information propagates through the network, the receptive field of each neuron expands, allowing it to capture increasingly larger spatial features in the input data. Thus, the receptive field is crucial in understanding the network's ability to capture and process information.

Figure 2.3 illustrate this through a simple example of a CNN with two 1D convolutional layers with kernel size 3, stride 2, and no padding. While each node in the first layer only observes three input elements as indicated by the light blue color, a node in the second layer perceives all the seven input elements marked in blue.

The receptive fields at layer l can be characterized by the width r^l, the center position c^l of the first left receptive field, and the spacing j^l between the center positions of two adjacent receptive fields. The receptive field at each layer can be calculated iteratively

from the first to the last layer using the equations [42]

$$j^{l+1} = sj^l \tag{2.2.11a}$$
$$r^{l+1} = r^l + (k-1)j^l \tag{2.2.11b}$$
$$c^{l+1} = c^l + j^l\left(\frac{k-1}{2} - p\right), \tag{2.2.11c}$$

where s, k and p are the stride, kernel size and padding at layer $l+1$. For the initial layer we have $j^0 = 1$, $r^0 = k$ and $c^0 = \frac{k-1}{2} - p$. Note that these equations apply to both convolutional layers and max pooling layers.

2.2.5 Other Types of DNNs

The repeating layers give the DNNs a modular and flexible structure, enabling various configurations. In addition to the CNN, there exist a variety of types of DNNs.

A network consisting of fully connected feed-forward layers is also known as a multilayer perceptron (MLP). As illustrated in fig. 2.2, each neuron in a layer is connected to each neuron in the neighboring layers. This quickly grows to a huge number of parameters, making them computationally expensive. However, it is common to apply MLPs in combination with other types of networks, like the regression head in a CNN.

Recurrent neural networks are suitable for sequential data with temporal behaviors. Recurrent neural networks are characterized by their ability to store information about past states and integrate this knowledge into future predictions. Long short-term memory (LSTM) is one the most common kind of recurrent structure [34]

Autoencoders [43] is a type of feed-forward network that is tailored to compress and reconstruct data. An autoencoder is composed of an encoder that transforms the input to a lower-dimensional representation, and a decoder that uses this representation to reproduce the original input. Unlike the rest of the methods described, this is an unsupervised learning approach as there is no labeled data. By minimizing the difference between the reconstructed input and the actual input, the model learns how to most effectively encode the relevant information of the original input.

Generative Adversarial Networks [44] (GAN) generate synthetic data samples resembling the samples in the training data. A GAN consists of a generator network, which generates synthetic samples from random noise, and a discriminator network, which classifies whether a given sample is from the training data or generated by the generator. As the GAN is trained, the generator aims to deceive the discriminator by generating increasingly realistic samples that resemble the training data. GANs are used for image generators and text generators, exhibiting remarkable realism.

2.2.6 Model Validation

To assess the performance and generalization capability of a trained model, it is essential to validate the model using reliable techniques. Cross-validation [45] is a popular strategy

for model validation in machine learning. The main idea behind cross-validation is to divide the data into subsets, where one subset, often referred to as the validation set, is used to evaluate the model while the remaining data is used for training. This procedure is repeated several times using different validation sets, and the performance is averaged over the models. By training and validate over different combinations of the dataset, the method gives a more accurate description of the robustness and ability to generalize to unseen data. This approach is particularly useful for model selection, as it provides a reliable reference point for a fair comparison between different model architectures.

The k-fold cross-validation is one of the most common variants [46]. The data is divided into k equal-sized groups called folds. The model is trained using $k-1$ folds as training data, and evaluated on the remaining fold. This procedure is repeated k times with a new fold as the validation set, such that the model is evaluated once on each fold in the dataset. The model is re-initialized and trained from scratch each time. The performance is the average performance on the k validation sets.

The stratified k-fold cross-validation is a closely related variant. Each fold is chosen such that the distribution within each fold is similar to the distribution of the whole dataset. This variant is especially valuable when dealing with unevenly distributed categorical data. By applying stratified k-fold cross-validation, we can mitigate the risk of biased model performance and ensure that the model's training encompasses a representative range of categorical data instances during each cross-validation iteration.

The choice of k is a trade-off between bias, variance, and computational cost. Using a small $k-$value introduces more bias to the error estimate [47, 48]. For large datasets, however, the benefit from increasing the number of folds reduces, while the additional computational cost increases significantly. Values of $k = 5$ and $k = 10$ are commonly applied in the literature to strike a balance between these factors [47, 48].

Overfitting

Overfitting is a common problem in machine learning and refers to the case where the model is trained too well on the training data, such that the model memorize noise and randomness in the data. Rather than learning the underlying patterns that represent the complete data distribution, it becomes tailored for the specific samples in the training set. This might result in excellent performance on the training set but poor performance on new, unseen data. Several factors can contribute to overfitting, such as an overly complex model, excessive training duration, or a validation set that does not adequately represent the data distribution on which the model was trained. Cross-validation is a valuable tool to prevent overfitting. By comparing the model's performance on both the training data and the validation data, we can identify indications of overfitting and take appropriate measures to address this issue.

3 Method

3.1 Dataset

Developing accurate ML models generally requires large datasets. However, collecting cycling data from experimental or real-life operations is both time-consuming and expensive. Fortunately, several publicly available datasets related to Lithium-Ion Batteries (LIB) have been published. Some examples include datasets from NASA [49], Toyota Research Institute [9, 50] and CALCHE [51], which have been used in several studies on data-driven battery degradation modeling. Access to high-quality data across a range of chemistries, charging profiles, and conditions support the development of more accurate and sophisticated models.

When choosing a dataset, it is important to consider the problem at hand. The task of modeling how a specific battery behaves under different protocols requires different data than investigating how different batteries behave under the same protocol. The processes we want the ML model to learn must be observable in the dataset. This work aims to train a CNN for remaining useful life (RUL) prediction and explore the decision-making process of the model relates to degradation observables. To avoid the ML model learning "obvious" differences, we will use as similar cells as possible, both in terms of cycling protocols and chemistry. The reason for this can be illustrated with an example of two batteries of different chemistries with very different expected lifetimes. If the type of battery can be identified from the input, we might get a model that learns how to distinguish the chemistries from each other instead of identifying signs of degradation. Using data from the same chemistry leaves the variations between chemistries out of the equation and encourages the model to look for other features to decide the RUL. The choice of using similar charging profiles follows the same line of reasoning. By keeping the external conditions similar, the model must look for the changes caused by degradation mechanisms.

The dataset used in this work was generated by Severson et al. [9] and includes fast-charging cycle data from 135 commercial LFP/graphite cells with a nominal capacity of 1.1A h. The cells are cycled under comparable cycling protocols in a temperature-controlled environment of 30°C until reaching the EOL. The lifetimes in the dataset range from 150 to 2300 cycles and includes almost 97000 cycles.

The charging policies utilize different C-rates applied during various intervals of the charging. All cells are charged from 80% to 100% SOC at 1C. Between 0% and 80% SOC, the cells are charged using a one-step or two-step policy. For the one-step policies, the cell is charged at a constant C-rate over the whole 0% to 80% SOC interval. In the

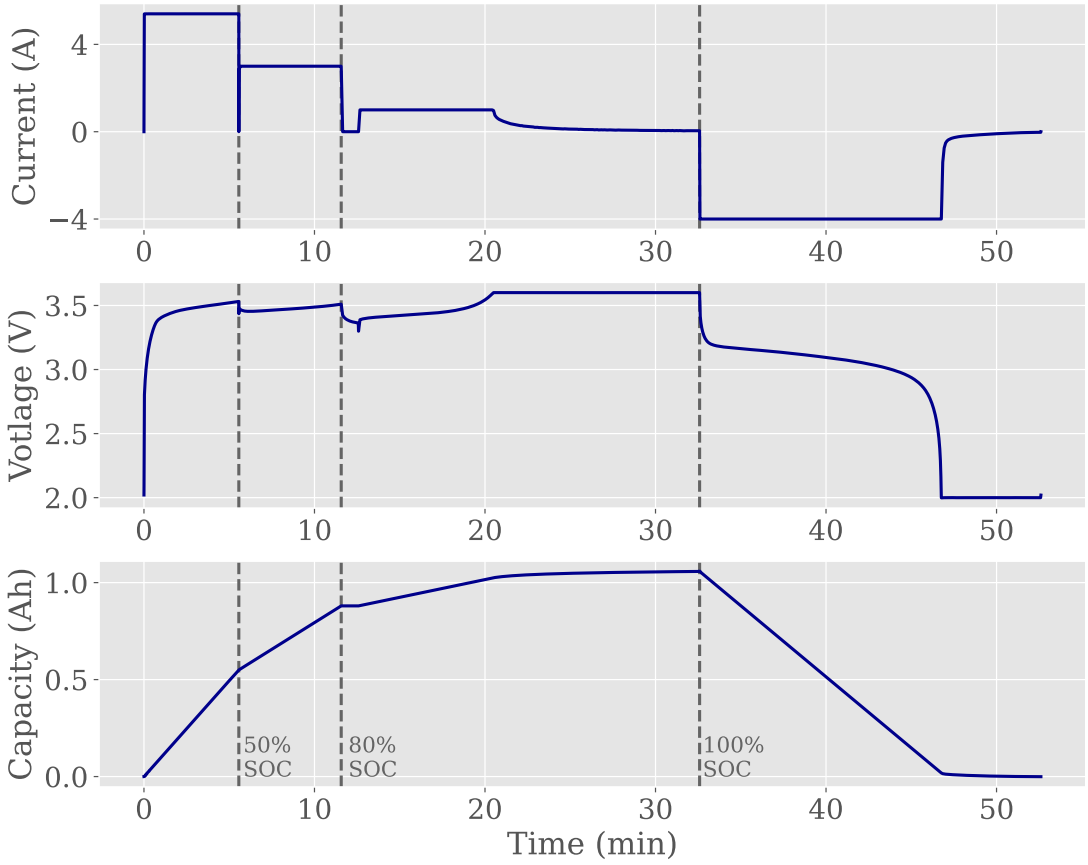


Figure 3.1: Example a cycling protocol from the dataset.

case of a two-step policy, the cell is charged with a constant C-rate to a certain SOC and then charged with a different C-rate until reaching 80% SOC. The values of the C-rates range from 3.6C to 8C, and the SOC at which the new C-rate is applied varies between 2% and 71%. An example of a two-step policy is displayed in fig. 3.1. A cut-off current of C/50 is also applied at the end of the charge and discharge cycle, and the cell rests at 80% SOC and after discharge. The cells are cycled in three different batches with slightly different conditions. For more details, the reader is referred to [9]. There are 72 different cycling protocols in total.

Several features relating to the state and performance of a battery are measured throughout the life of the cells. These features include current, voltage, internal resistance, temperature, and capacities, which are monitored during each cycle. In addition, the incremental capacity (IC) for the discharge cycle is also included in the dataset. The capacity vs. voltage curves and the corresponding IC curves are represented as functions of voltage and are evaluated at 1000 equally separated points in the voltage window from 2.0V to 3.5V.

3.1.1 Data Preparation

The initial pre-processing of the data follows the same procedure as in the original study [9]. The pre-processing includes removing cells with unexpectedly high measurement noise, along with cells that do not reach 80% of nominal capacity. One cell deviates notably from the rest of the cells and is removed from the dataset. Note that this cell was not removed in the original study, but was identified as a potential defect and later excluded from parts of the study. Their supplementary GitHub repository [52] provides the code for loading and pre-processing of the dataset that was also used in this work. After these adjustments, the dataset contains cycling data from 124 cells.

Standardization of the input data is a common practice as it accelerates the training [41]. However, in this work, no standardization or normalization of the input is performed. The reason behind this decision lies in the intention of preserving the inherent physical representation of the input. Introducing an additional transformation would complicate the interpretation of the data.

The original dataset includes much information, including metadata for each experiment, measurements per cycle, and time-series data for each cycle. The whole dataset is structured as a multi-level nested dictionary. Although this is a very efficient structure for such a complex dataset, it is inconvenient when we only use a small part of the complete dataset. Therefore, the relevant parts of the dataset were extracted and reorganized to a more suitable structure. Originally, the time series measurement from a cycle is stored as individual 1D time series in a dictionary. Instead, the cycle time series of a particular property are stacked on top of each other to form a 2D matrix of size $n_i \times m$, where n_i is the total number of cycles of cell i and m is the number of measurements per cycle for this property. With this representation, row j represents the time series measurements during cycle j, and the full matrix holds the time-series data over the whole cycle life. This 2D array is easier to manipulate and pass to the CNN model.

Measurement noise affects some of the IC curves. The possibility of applying smoothing algorithms was initially considered but ultimately not pursued due to time constraints and the potential complexity they could introduce to the model and the physical interpretation of the input. Instead, the level of noise in each curve was quantified by calculating the average absolute difference between adjacent points, drawing inspiration from [53]. We checked the sensitivity of our model on removing data above various noisiness thresholds and found no clear advantage. Therefore, we decided to proceed without removing any additional samples

3.1.2 Train-Test Split

The dataset is divided into a training set and a test set. To ensure that both the training and test sets are representative of the overall distribution of the data, the cells were shuffled before the splitting. Using a 70-30 split ratio, the training set and test set consist of 86 and 37 cells, respectively.

Figure 3.2 shows the distribution of cycle lives in the training set as blue bars, and the

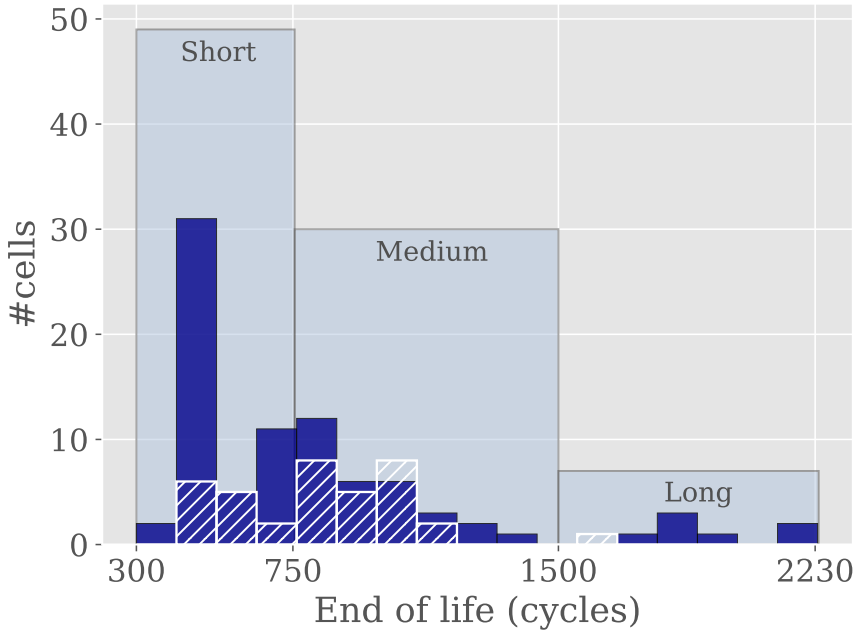


Figure 3.2: Distribution of cycle lives in the dataset. The dark blue bars indicate cells in the training set, and the light blue histogram divides the training set into groups of short, medium, and long-lived cells. The white bars represent cells in the test set.

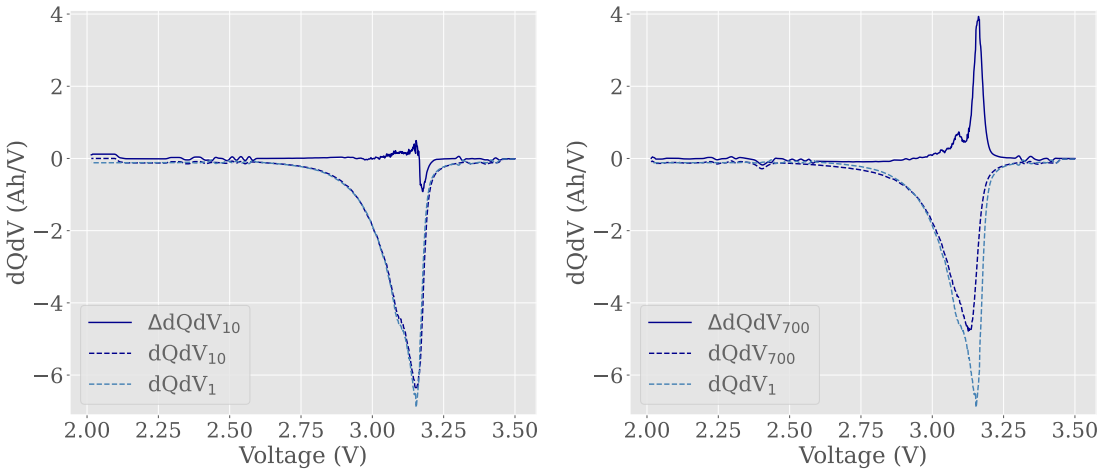
distribution of cycle lives in the test set in white. The training set is also divided into groups of short, medium, and long-lived cells. This categorization of cycle lives is done as a part of the model selection process, which will be further explained in section 3.5.2.

The training set is used to decide the model structure and determine the parameters of the model, while the test set is used to evaluate the performance of the model. Note that the test set, including the distribution of lifetimes within the test set, has not been seen throughout the design and training process. The test set was introduced only for the final performance reporting. In this way, we limit potential bias from the designer and ensure that the model is evaluated on completely unseen data.

3.2 Feature Engineering

For the purpose of predicting the RUL, we want to utilize electrochemical signals that hold information about the state of health of a battery. The IC curves are recognized for this purpose, as they provide insight into the electrochemical processes and degradation modes of the cell.

Furthermore, the IC curves also enable easier interpretation of the CNN model. The IC curve is expressed in terms of voltage and can be represented as a 1D vector where each element corresponds to the capacity change at a specific voltage. Consequently, the receptive field of the CNN model can be assigned to a region in the voltage window. The signal within this region can be related to the electrochemical process happening at



(a) ΔdQdV at cycle 10.

(b) ΔdQdV at cycle 700.

Figure 3.3: Two examples of ΔdQdV curves for a cell in the test set, together with the corresponding IC curves. The dashed light blue line is the initial IC curve, and the dark dashed line is the IC curve at the indicated subsequent cycle. The solid line is the difference ΔdQdV between the two IC curves.

this voltage.

As described in section 2.1.3, the degradation mainly manifests as changes in the IC peaks. Therefore, it is logical to consider the difference between the IC curve dQdV_n at cycle n, and the initial IC curve, dQdV_1, at the first cycle. Then we obtain the IC difference curve

$$\Delta\mathrm{dQdV}_n = \mathrm{dQdV}_n - \mathrm{dQdV}_1. \tag{3.2.1}$$

The dataset contains the IC curve for the discharge cycles evaluated at 1000 linearly separated points, which makes the subtraction between cycles straightforward. The $\Delta\mathrm{dQdV}_n$ for every cycle $n > 1$ is calculated by eq. (3.2.1). The resulting $\Delta\mathrm{dQdV}_n$ is a 1D vector of length 1000 and serves as the input of the CNN model. Figure 3.3 shows two examples of ΔdQdV curves. The dashed lines are the dQdV curves, and the difference ΔdQdV is shown as a solid line. Figure 3.3a displays ΔdQdV at cycle number 10. The small shift of dQdV_{10} is represented by an oscillation in $\Delta\mathrm{dQdV}_{10}$. $\Delta\mathrm{dQdV}_{700}$ for the same cell is shown in fig. 3.3b. After 700 cycles, the difference between the two IC discharge curves is more evident.

Another advantage of using the $\Delta\mathrm{dQdV}_n$ as input feature is the potential to adapt the method to other scenarios. By training on the differences, the CNN should learn to recognize changes in the IC curves rather than the form of the IC curve itself. While the shape of the IC curve may differ for other chemistries or conditions compared to the LFP cells in these experiments, degradation should still manifest as shifts and intensity reduction in the IC curve. The features learned from the differences might provide a more general representation of degradation that might also be relevant for IC curves with other cathode materials. Utilizing learned features from one model for a related task is known as transfer learning [40]. Evaluating the benefit of a transfer learning approach

is considered outside the scope of this work; however, the use of ΔdQdV_n curves aims to facilitate a more robust and versatile approach.

The target value is the RUL, which is the number of cycles until reaching EOL. EOL is conventionally defined as the cycle at which the cell reaches 80% of nominal capacity. Then, the remaining useful life RUL_n at the cycle number n is defined as

$$\text{RUL}_n = \text{EOL} - n. \tag{3.2.2}$$

The RUL at every cycle throughout the life of each cell is calculated according to this formula. RUL_n is in the unit of cycles.

The RUL can also be represented as a normalized value between 1 and 0 as

$$\overline{\text{RUL}}_n = \frac{\text{RUL}_n}{\text{EOL}} \tag{3.2.3}$$

$$= \frac{\text{EOL} - n}{\text{EOL}}. \tag{3.2.4}$$

The normalized RUL has a value of 1 at the first cycle and decreases linearly until reaching EOL at value 0.

The state of health (SOH) of the battery expresses the current condition of a battery in the form of a percentage [26]. It is typically measured by comparing the actual charge capacity with the initial charge capacity, with 100% representing the state at the start of the battery cycle life. The normalized RUL can be considered an alternative metric expressing the SOH, as it also provides a relative measure of degradation. Unlike the SOH, which can be defined in multiple ways, RUL has a straightforward and interpretable definition.

3.3 Regression Problem Formulation

This work aims to use a CNN to predict RUL given the IC difference at any cycle. This regression problem can be formulated as the mapping

$$y = \mathcal{F}(X; \theta), \tag{3.3.1}$$

from the input X to the output y, where the mapping $\mathcal{F}$ is a CNN parameterized by a set of parameters θ. The input X is the IC difference curve $\Delta\text{dQdV}_n \in \mathbb{R}^{1\times 1000}$ during discharge at cycle n, and the output y is the corresponding remaining useful life $\text{RUL}_n \in \mathbb{R}$. In addition to the initial cycle, the CNN uses information from a single discharge cycle to do lifetime predictions at an arbitrary point in the cycle life.

We may rewrite the regression problem as

$$\text{RUL}_n = \text{CNN}(\Delta\text{dQdV}_n; \theta), \tag{3.3.2}$$

The regression problem is solved by finding the parameterization θ that best approxi-

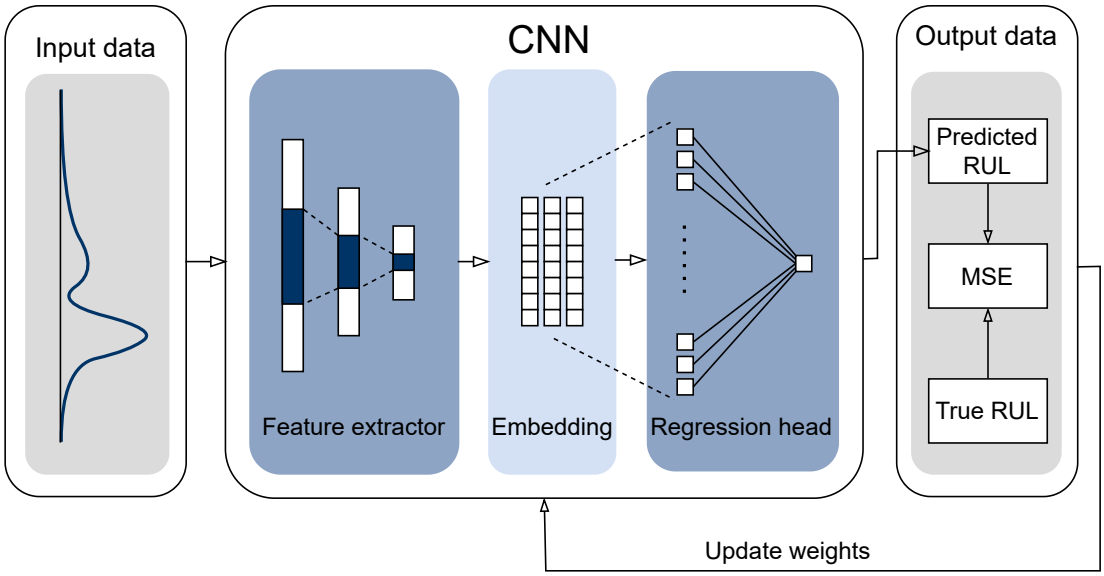


Figure 3.4: Diagram illustrating the key components of the proposed method. The CNN learns the mapping from the input, ΔdQdV, to the output, RUL, by iteratively updating its parameters such that the errors between the model outputs and the true values are minimized.

mates the mapping in eq. (3.3.2). This is obtained by minimizing the MSE (eq. (2.2.4)) between the outputs of the CNN and the true values.

The strategy is illustrated in fig. 3.4. The feature extractor utilizes a sequence of convolutional operations to extract features from the input curves and converts the input into an intermediate embedding representation. The regression head then predicts the RUL based on the embedding. By learning a mapping from ΔdQdV to RUL, the CNN learns patterns in the ΔdQdV curves that relate to cell degradation.

3.4 Model Architecture and Hyperparameters

Designing the model architecture and choosing hyperparameters are important elements in developing an accurate and efficient model. This includes multiple design choices, such as the depth of the model, the learning rate, and the size of each convolutional filter. Given the immense number of possible configurations, we need some guiding principles to limit the scope of the hyperparameter search.

Interpretability and simplicity: Part of the objective is to explore what the model has learned, implying that interpretability is a desired quality. Consequently, a simple model with a limited number of parameters is preferred. This means that the number of layers and the number of filters at each layer should be kept to a minimum. However, bigger models generally tend to perform better. The balance between accuracy and simplicity is an important aspect to consider.

Utilizing existing architectures and principles: Other successful CNN architectures and training principles is a natural starting point. CNNs have traditionally been used for 2D image processing, but their applications extended to other problems as well. A closely related application is CNNs for the classification of electrochemical spectra. The 1D spectra have similarities with the IC curves, making such CNN architectures a useful source of inspiration.

The final architecture is determined through a hyperparameter search. To limit the number of candidate architectures, certain parameters are fixed and others are chosen from the hyperparameter search. The reasoning behind the architecture design and parameter choices is described, setting up the parameter space used in the search.

3.4.1 Composition of Layers and Activation Function

The first thing to consider is the overall architecture, including the types of layers and the depth of the model. The main building block is the 1D convolutional layer. The authors of [54] compare 20 studies using DNNs for spectroscopy analysis. The CNN models described in the review have between 1 and 4 convolutional layers. With interpretability in mind, we consider 2 or 3 convolutional layers an appropriate network depth for this work. After conducting some initial experiments, we decided to use 3 convolutional layers. Models of this depth tended to perform better, aligning with the intuition that deeper models generally perform better. The number of layers also influences the receptive field of the embedding vector, and 3 layers tend to give a suitable width of the receptive field overall. Additionally, we apply batch normalization after each convolutional layer.

In addition to the convolutional layers, it is common practice to include a pooling layer. The pooling operation reduces the dimension of the feature map without introducing any learnable parameters and makes the intermediate representation approximately invariant to small local translations [40]. Degradation can be observed from shifts in the peaks of the IC curves, suggesting that the location of peaks might represent meaningful information. Thus, it is natural to consider the option of *not* including max pooling layers. This configuration is explored during the hyperparameter search, allowing for removing max pooling layers as one of the options.

We will use a simple regression head with one fully connected linear layer. With a single linear layer, the embedding vector is transformed to the scalar output through a single dot product with the linear weights and adding of the bias term. Having one layer facilitates for interpretability of the CNN, as this makes it straightforward to relate a weight in the linear layer to a specific region of the original input. The fully connected layer is a global operation where each node is connected to all input nodes. With a single output node, each weight is associated with one element in the embedding vector and the corresponding receptive field. By adding several linear layers, on the other hand, the complete embedding will influence the subsequent linear neurons. This regression head configuration is fixed during the hyperparameter search.

We use the ReLU function defined in (2.2.2) as the activation function. In recent years,

this has been the standard activation function in neural network architectures [35]. The activation function is applied after the batch normalization and as the final activation before the output layer. The output of the ReLU is non-negative and unbounded for positive values like the RUL, making it well-suited as the final activation.

With this setup, the feature extractor consists of three repeating blocks of convolution, batch normalization, max pool, and ReLU. In the case of not including max pool, the block consists of convolution, batch normalization, and ReLU. The regression head transforms the embedding vector to the final prediction using a single fully connected linear layer. Apart from the use of max pooling, the overall network structure is relatively constant over the hyperparameter search.

3.4.2 Layer-Specific Parameters

The next step is to define the kernel size, stride, padding, and the number of channels for the convolutional layers.

First, we determine suitable values for the kernel sizes. For CNN-based spectroscopy analysis, [29, 55] uses kernel sizes between 3 and 7, while [27, 28] also apply wider kernels of sizes 19 and 21. The CNN designed by [27] has a structure with reduced kernel size for deeper layers. In this work, we let the hyperparameter search determine the kernel size of the three layers. The first kernel size is assigned possible values of 7, 15, or 25. From visual inspection of the IC curves, it can be seen that the width of the peaks is roughly 25 points. The maximum kernel was chosen to reflect this signal. Following existing literature, the kernel sizes for the two last layers are selected as 5 or 7 during the search.

We then proceed to define the stride. The stride also broadens the receptive field. With only 3 convolutional layers in the model, using strides larger than one is an option to reduce the dimensionality faster and expand the receptive field. However, important information may be lost if the stride is too large. We choose the stride of the first kernel to be a hyperparameter taking values of approximately 1/3 or 1/5 of the kernel size. This corresponds to possible strides of $\{2, 3\}$, $\{3, 5\}$, or $\{5, 8\}$ for kernels of size 7, 15, and 25, respectively. For the kernels of size 5 or 7, the stride is 1 when max pooling is included and 2 if not. For simplicity, zero padding of size $(K - 1)/2$, where K is the kernel size, is used for all convolutional layers.

Each convolutional layer can have one or more channels. The number of channels reflects the maximum number of features each layer can identify. While more channels provide the ability to detect more features, it also implies a more complex and less interpretable model. As interpretability is considered an essential quality of the proposed model, a maximum number of 8 channels was initially selected. This number was later reduced to 4 channels due to the observed overfitting. To reduce the number of possible configurations, the number of channels is 2, 3, or 4 and is the same for all three layers.

The standard configuration of the max pooling operation is a kernel of size 2, stride of 2, and padding of 0. This means that the maximum value of each pair of adjacent input elements is passed through the max pooling layer, reducing the output size by half compared to the input size. This design will also be applied in this work whenever a max pooling layer is used.

The feature extractor's composition and parameters determine the size of the embedding. We will have one feature-based representation for each channel in the last convolutional layer, making the embedding space a 2D vector whose height is the number of output channels and the width is the length of the embedding vector. Generally, larger strides, more layers, and max pooling decrease the length of the embedding vector. When the embedding vector shrinks, the receptive field of each element broadens. Increasing kernel size also increases the receptive field. For easier interpretability, having a receptive field narrow enough to highlight signals in the input is convenient. However, many factors determine the size of the embedding vector and the corresponding receptive field of each element, making it impractical to impose the size of the receptive field or embedding vector directly. Instead, this is determined indirectly by the other design choices. The size of the embedding vector and receptive field has been considered when choosing other hyperparameters. No additional constraints are introduced to affect the size of the embedding vector.

3.4.3 Training Configuration

The MSE loss function will be applied in this work. The loss function quantifies the error between the target values and the model outputs. By minimizing the loss function, the parameters are adjusted to decrease the error. To solve this optimization problem, we will use the AdamW optimizer [39].

Another crucial training hyperparameter is the learning rate, which determines the step of each parameter update according to eq. (2.2.9). A very small learning rate can cause the algorithm to get stuck in local minima, while a large learning rate can lead to unstable training. The AdamW default learning rate of 0.001 serves as the starting point. The learning rate was initially defined in the range of $[1e{-}3, 1e{-}1]$ and left as a hyperparameter to tune. However, the learning rate turned out to be too large. The loss curves did not decrease over time, making it unclear if the model was learning over the epochs. Thus, the learning rate was reduced and later fixed at $1e{-}4$.

3.5 Training and Validation

Developing the model can be divided into two phases: Model selection where the final model architecture and hyperparameters are decided, and model optimization where the trainable parameters of the model are determined. Note that the training set is used for both model selection and model optimization. The test set is used for performance evaluation only after the final model is optimized.

3.5.1 Evaluation Metrics

The root-mean-squared error (RMSE) is used to evaluate the predictive performance. This is a standard evaluation metric for battery lifetime prediction, as well as regression problems in general. The RMSE of N samples is defined as

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}, \tag{3.5.1}$$

where y_i and $\hat{y}_i$ are the target and predicted values, respectively. The lifetime is measured in number of cycles, giving the RMSE in units of cycles.

Because the residuals are squared in the RMSE, the RMSE puts more weight on large residuals. The mean absolute deviation (MAD), on the other hand, emphasizes all residuals equally. This might be seen as a more intuitive metric and is often included in addition to the RMSE. The MAD is defined as

$$MAD = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|, \tag{3.5.2}$$

and represents the average absolute prediction error.

3.5.2 Model Selection with Stratified k-fold Cross-Validation

To evaluate the performance of the models generated through the hyperparameter search and determine the best model, we employed the cross-validation strategy. Ordinary k-fold was initially applied but was later substituted with the stratified k-fold algorithm. This change was motivated by the observation of significantly higher validation loss compared to the training loss, indicating overfitting. The degree of overfitting also varied between the individual folds, indicating a poorer ability to generalize on parts of the training data. Therefore, it was speculated that uneven distribution of the long-lived cells might lead to poor validation losses in some folds. As evident from fig. 3.2, few cells have cycle life over 1500 cycles. If a fold ends up with a large proportion of long-lived cells, the model might have trouble generalizing when trained on the remaining data. The cells were categorized as short-, medium-, and long-lived cells according to fig. 3.2. During the stratified cross-validation, the distribution between these three categories in each fold are roughly the same as in the complete training set.

The training set at hand includes close to 70 000 samples, making the computational cost a factor to consider. Thus, we consider $k = 5$ to be a reasonable trade-off between the bias and the computational cost of the procedure, in accordance with common practice [47, 48].

The CNN takes the IC curve from an arbitrary cycle as input and predicts the corresponding RUL. The CNN does not consider the time dependencies between the cycles, meaning the inputs can be presented in random order. However, we need to be careful with how the folds are chosen. IC curves from the same cell close in time are strongly

correlated. If IC curves from the same cell end up in two different folds, samples in the validation set will be correlated with samples in the training set. Then the validation set is not completely unseen for the model, giving an overly optimistic evaluation of the model's ability to generalize to new data. Consequently, it is necessary to include all samples from a cell within the same fold, which is why the folds were distributed based on cell life, and not the RUL values of individual samples.

Each architecture proposed by the hyperparameter search algorithm has to undergo a complete stratified 5-fold cross-validation scheme, where each model is trained for 35 epochs. To ensure consistency, the reported metric at each fold is the average of the five epochs with the lowest RMSE. This ensures that the metric reflects consistently low RMSE and not just a single "lucky" epoch.

3.5.3 Model Optimization

After the best model architecture has been identified from the cross-validation hyperparameter search, the final model can be trained. The final model is trained using the complete training set. The number of epochs is determined by a qualitative analysis of the cross-validation loss where the aspects of minimum loss and minimal risk of overfitting are balanced.

3.6 Implementation

The task described in this chapter was implemented using Python 3.9 and various open-source Python libraries.

For processing and computations on the dataset, we utilized the NumPy library [56]. NumPy is a widely used library for numerical computations in Python, providing efficient data structures and functions for array manipulation.

For the deep learning aspect of the implementation, we employed PyTorch [57], a popular machine learning framework specifically designed for deep learning tasks. PyTorch serves as the underlying framework for building and training our CNN, providing a comprehensive set of tools and functions for neural networks.

Scikit-learn [58] is another useful Python library for machine learning tasks. The library provides tools for cross-validation and model evaluation used in this implementation.

The hyperparameter search was implemented using Optuna [59], which is a hyperparameter optimization tool for machine learning frameworks. Optuna was integrated on top of the Pytorch training pipeline, enabling an efficient procedure for proposing, training, and comparing model architectures.

Plotting of the data was done using Matplotlib [60], a popular data visualization library in Python.

4 Results and Discussions

4.1 Model Architecture

The hyperparameter search using stratified 5-fold cross-validation was performed for 30 different configurations to find the most suitable model architecture in the hyperparameter space. The most successful model architecture is presented in table 4.1. The CNN has a repeating structure of a convolutional layer, batch normalization, ReLU, and a max pooling layer. This is repeated three times. The regression head of the CNN consists of a single fully connected linear layer. With repeating blocks of convolution, activation, and max pooling, the overall model structure is similar to the traditional 2D-CNN architecture [30].

The first convolutional layer has a kernel of size 7 and a stride of 2, and the two remaining layers have kernels of size 5 and strides of 1. All convolutional layers have 3 channels. This results in an embedding vector of size 62×3, or 186×1 when reshaped to the column vector that is passed to the linear layer. The architecture is illustrated in fig. 4.1. The model has a total of 325 trainable parameters. Even though the CNN technically qualifies as a deep neural network, this architecture is comparatively shallow when compared to other state-of-the-art models. For instance, the auto-CNN-LSTM model by Ren et al. [61] consists of a CNN module, an LSTM module, an autoencoder, and a seven-layer regression head. The CNN alone has around 54 000 parameters. The CNN model by Strange & Reis [62] has over 200 000 parameters, and the dilated CNN by Hong et al. [63] includes 2M parameters.

Table 4.1: Architecture of the final CNN. BN+ReLU+MP denotes the sequential layers of batch normalization (BN), ReLU and max pooling (MP). The kernel size and stride for these entries refer to the max pooling layer.

Module	**Layer**	**Kernel size/ #weights**	**Stride**	**Channels**	**Output size**
Feature extractor	Conv1D	7	2	3	500×3
	BN+ReLU+MP	2	2	-	250×3
	Conv1D	5	1	3	250×3
	BN+ReLU+MP	2	2	-	125×3
	Conv1D	5	1	3	125×3
	BN+ReLU+MP	2	2	-	62×3
Embedding vector	Flatten	-	-	-	186
Regression head	Linear Layer	186	-	-	1
	ReLU	-	-	-	1

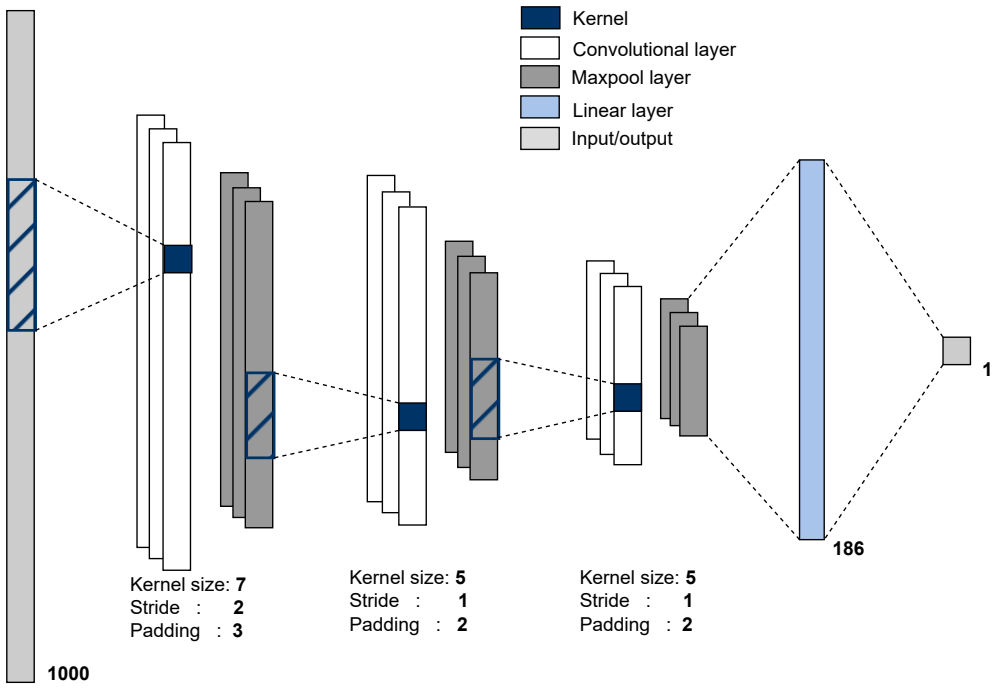


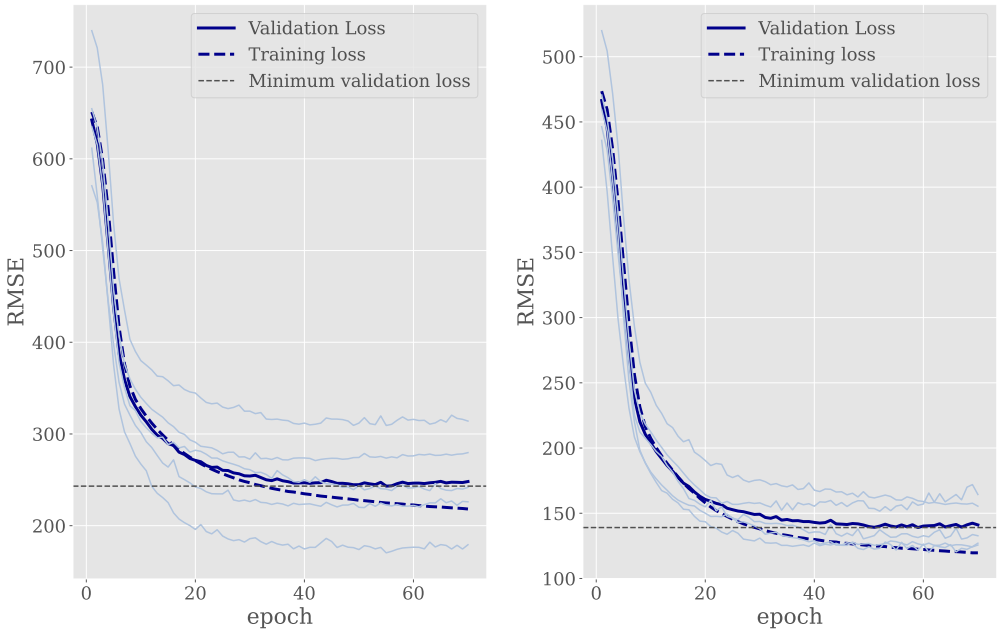
Figure 4.1: Illustration of the final CNN model architecture.

Given the architecture in table 4.1, the receptive field of each element in the embedding vector can be calculated according to eq. (2.2.11c). This results in a receptive field of size $r = 69$ with a spacing of $j = 16$ between two adjacent receptive fields.

The cross-validation loss over 70 epochs of the CNN is shown in fig. 4.2a. The solid dark-blue line is the validation loss averaged over all folds, and the dashed blue line is the average training loss. The light blue curves show the validation loss for the individual folds. The gray line marks the minimum average validation loss achieved during the 70 epochs. The loss curves exhibit the desired behavior: The loss reduces at the beginning and stabilizes at later epochs, indicating that the model is progressively learning relevant patterns for RUL prediction.

The cross-validation loss can also serve as a guide for selecting a suitable number of epochs for training the final model. When choosing the epoch number, we want to minimize both the RMSE and the risk of overfitting, which are often conflicting. Figure 4.2a shows that the average validation loss stabilizes around 40 epochs and reaches its minimum at epoch 54. The training loss, on the other hand, continues to decrease at a steady rate after 40 epochs. The training and validation losses are roughly the same up to epoch 25, where the model starts to overfit slightly. Nonetheless, we gain accuracy by training for more epochs. The RMSE does not reduce notably after 40 epochs. Thus, epoch number 40 appears to be a suitable balance between minimizing RMSE and reducing overfitting.

The CNN appears to be sensitive to the input data. As observed from fig. 4.2a, the minimum RMSE for the individual folds varies from 175 to 310 cycles. This variation indicates that some cells are deviating from the overall data distribution, making the



(a) Complete training set.

(b) Short and medium-lived cells.

Figure 4.2: 5-fold cross-validation loss for the CNN trained on (a) the whole training set and (b) short- and medium-lived cells in the training set, i. e with EOL below 1500 cycles. The solid dark line is the validation loss averaged over the folds, while the lighter solid lines are the validation loss of the individual folds. The dashed line is the average training loss.

RUL of such cells challenging to predict if they appear in the validation set. This demonstrates the importance of performing cross-validation instead of a single training-validation split. By evaluating on all parts of the training data, we remove the chance of just being "lucky" in how the training data is split. The cross-validation loss reflects the performance over the complete distribution of data.

The cross-validation loss excluding cells with cycle lives above 1500 cycles is displayed in fig. 4.2b, displaying a notable reduction in RMSE and improved generalizability. This result further suggests that the cells with longer cycle lives contribute significantly to the RMSE and large variation between the folds.

4.2 Predictive Performance

4.2.1 RUL Prediction

With the model architecture and training parameters decided, the final model is trained on the complete training set for 40 epochs. The CNN is then evaluated on the unseen test set.

The RMSE and MAD of the test set are presented in table 4.2. Despite the shallow model architecture, the CNN achieves an RMSE of 171 cycles and MAD of 131 cycles. It is worth noting that the model relies solely on the information derived from the difference

Table 4.2: Performance metrics for the proposed CNN for RUL prediction on the test set, together with the performance of a linear regression model. The models below the dashed line are references to other work on the same dataset. The inputs include incremental capacity (IC), time series of voltage (V), current (I) and temperature (T), and state of health indicator (SOH) in the form of per cycle capacity.

	RMSE	MAD	Input	Cycles used
CNN (this work)	171	131	IC	2 discharge cycles
Linear regression	502	225	IC	2 discharge cycles
Dilated CNN [63]	76	-	V, I, T	3-4 cycles
CNN [62]	99	59	V, I	1 cycle
LSTM [22]	106	-	SOH	100 cycles

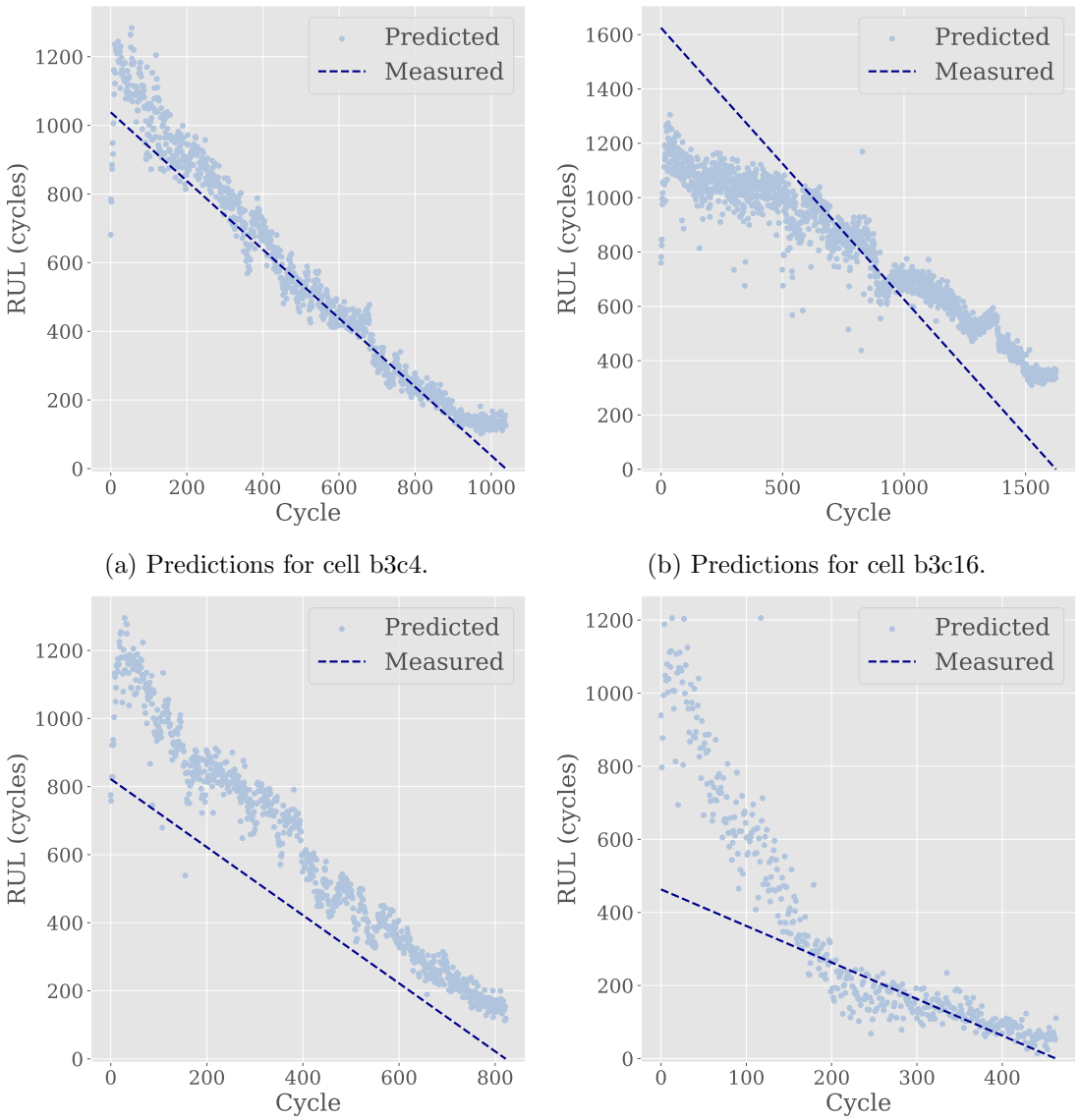
between the IC curve of two discharge cycles to make predictions at any location in the life cycle.

As a baseline, a linear regression model was trained using the same training data. This model yields an RMSE of 502 cycles and a MAD of 225 cycles, which is significantly higher than our CNN. The reduction in performance indicates that the input-output relationship is indeed non-linear and justifies the use of a non-linear approach like a CNN. Furthermore, the linear regression model generated several predictions considerably outside the range of reasonable target values. Such outliers were not seen among the predictions of the CNN. The CNN clearly outperforms the linear regression model in terms of accuracy and robustness.

The performance of other CNN-based models using the same dataset is also displayed in table 4.2. These models have significantly better predictive performance, as expected, given their significantly larger number of parameters. However, it is worth noting that the performance gains of the larger models do not scale proportionally with their number of parameters. Going from 325 parameters in our model to a dilated CNN with 2 million parameters reduces the RMSE by only 100 cycles. Hence, it is possible that the RUL prediction problem can be addressed with smaller neural networks, like the one we propose. In addition, with our small architecture, we can also explore the mechanism guiding the predictions. It should also be noted that the problem at hand poses a difficult task. Our CNN is presented with data from the discharging only, which happens at $4C$ for every cell. The lifetime is strongly correlated with the cycling protocol [64]. Hence, some authors include the time series during both charge and discharge [62, 63]. In contrast, our network does not get input from the charge cycle, but we expect it to learn the influence that the protocols have on the IC curves. Indeed, despite the simplistic architecture and limited input features, the CNN exhibits the ability to identify patterns relating IC curves to the remaining useful life and demonstrates reasonable accuracy.

Interestingly, the test RMSE is better than the RMSE from the cross-validation. The distribution of the test set can explain this. The splitting into training and test set was done randomly, without any specifications to influence the outcome. This was done to remove potential bias from knowledge of the test set. By chance, the test set contains

few long-lived cells. As discussed, samples with high RUL are associated with more challenging predictions and larger error terms. This is exemplified by fig. 4.2b, where removing long-lived cells significantly improves the performance. The average validation loss here is about 140 cycles. This modified training set better reflects the distribution of the test set, and we see that these two results are more aligned. When excluding long-lived cells, the training and test RMSE are very similar.



(a) Predictions for cell b3c4.

(b) Predictions for cell b3c16.

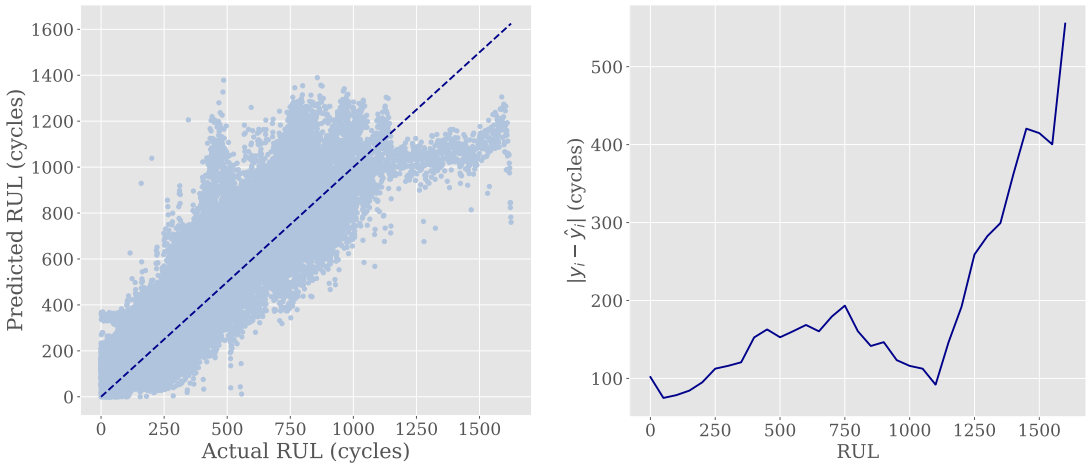
(c) Predictions for cell b3c8.

(d) Predictions for cell b2c27.

Figure 4.3: CNN predictions of RUL for four cells in the test set.

The predicted RUL for a selection of cells is shown in fig. 4.3. The dark blue lines represent the true cycle life, which decreases linearly until reaching the EOL. The light blue points indicate the predictions by the CNN. Particularly for the cell in fig. 4.3a, the predictions coincide well with the true values. For the cell in fig. 4.3c, the predictions exhibit an offset from the true values, but decrease linearly over the cycle life with approximately the same rate as the true RUL. Figure 4.3d depicts a short-lived cell. In this case, the predictions display a decreasing, but more non-linear behavior, which is also observed in other cells with short lives. Figure 4.3b shows the predictions for the cell with the highest EOL in the test set. The predictions also show a declining trend over the lifetime, although with significant errors observed at both the early and late stages of the cell's life. Overall, the predicted RUL clearly exhibits a decreasing trend throughout the cycle life. The CNN does seem to capture features in the IC difference associated with degradation and leverage this information to estimate RUL.

As can be observed in fig. 4.3, the deviations seem to be larger at the beginning of the cell life. This behavior is consistent for the other cells in the test set. During the early stages of the cycle life, signs of degradation may be limited, and the changes in dQdV_n curves compared to the initial dQdV_1 smaller. The input data at the initial cycles is expected to contain less information and therefore be more challenging to predict accurately. From fig. 4.3, it is evident that the CNN predicts roughly the same value for the early cycles across all four cells. This suggests that the patterns indicating the specific lifespan of 500 cycles or 2000 cycles at early cycles are not distinctly captured by the model, and it instead predicts some intermediate value. However, the results indicate that the model recognizes the early stage of the cycle life and incorporates this information into its predictions.



(a) Predicted versus actual RUL for the complete test set.

(b) The mean average deviation as a function of the target value.

Figure 4.4: CNN performance for RUL prediction on the complete test set.

The predicted values vs. the actual values for the whole test set are displayed in fig. 4.4a. The model tends to overestimate the remaining life for target values below 1000 cycles and consistently underestimates the RUL when the target value is above 1150 cycles.

The early cycles of the long-lived cells are more challenging to predict, as indicated by fig. 4.4b, showing that larger errors are associated with higher target values. In addition to limited signals in early cycles, this might be a consequence of the uneven distribution of data. With training samples spanning the region from 0 to 2230 cycles, there is a wide range of possible target values. However, there are considerably fewer samples of high target values (see fig. 3.2). Specifically, only eleven out of 86 cells have an EOL over 1150 cycles, meaning that a small subset contributes with samples above this value. As the target value decreases, the number of samples accumulates. More precisely, there are 62 000 training samples with RUL below 1150 and 5900 above this level. Given the distribution of the data, we can expect to observe diminishing performance for high target values.

Figure 4.5 displays the RUL predictions for two cells in the test set, together with the average cell temperature at each cycle. The predictions in the left plot align well with the true values, except for the interval from cycle 350 to cycle number 520. Notably, this region coincides with a rise in the average cell temperature. A similar pattern can be observed in the right plot of fig. 4.5. The results indicate a positive correlation between cell temperature and predicted RUL, where increased temperature leads to increased RUL predictions. This implies that the cell temperature influences the shape of the IC curve in a way that the CNN associates with a longer remaining life. Given the observed intercorrelation between IC curves and temperature, including additional features might be necessary to enable the CNN to account for dependencies with correlated factors.

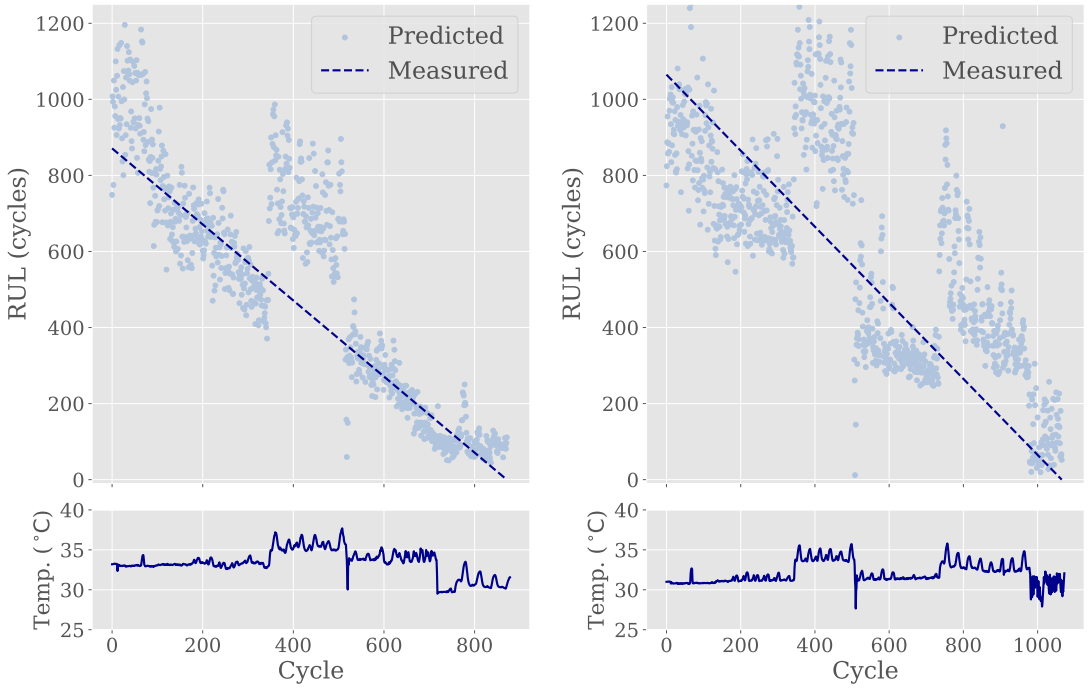


Figure 4.5: Predictions by the CNN for two of the cells in the test set together with the average cell temperature. The predicted RUL seems to correlate positively with the cell temperature.

4.2.2 Normalized RUL Prediction

Considering the discussion in section 4.2.1, it is evident that the model has the ability to qualitatively distinguish between dQdV curves from early, middle, or late stages of the cycle life. This observation suggests that the ΔdQdV curves are well-suited for predicting a normalized version of the RUL within this framework. Therefore, the model was retrained to predict the normalized RUL as defined in eq. (3.2.4).

A new hyperparameter search was conducted using the RUL normalized to the EOL as the target, and over the same parameter space as described in section 3.4. The resulting model architecture did give a small advantage over the model presented in table 4.1 in terms of RMSE, but ended up being less interpretable due to wide receptive fields. As interpretability was considered more important than the modest improvement in accuracy, we decided to continue using the original CNN architecture in table 4.1.

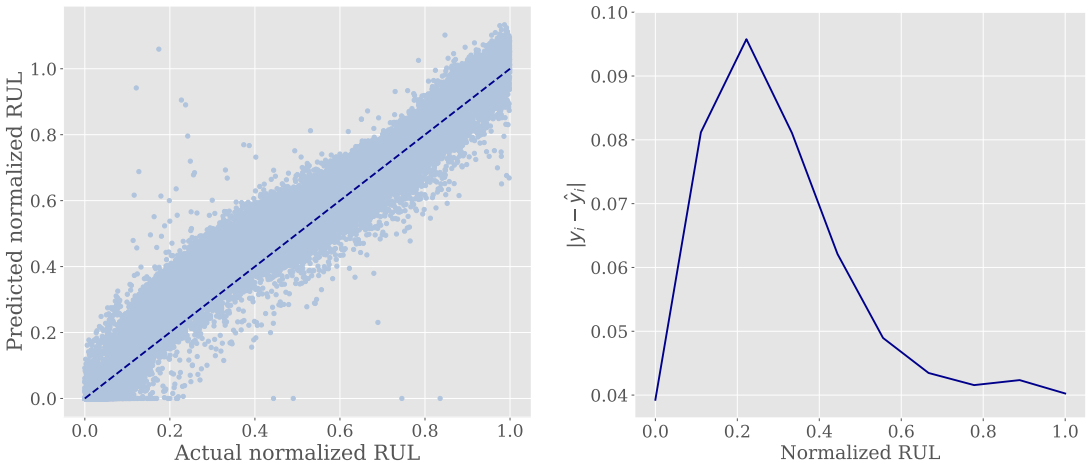
Table 4.3: Performance metrics for the proposed CNN for normalized RUL prediction along with the performance of a linear regression model. Note that the Auto-CNN-LSTM uses the NASA dataset [49].

	RMSE	MAD	Input	Cycles used
CNN	0.075	0.057	IC	2 discharge cycles
Linear regression	0.27	0.093	IC	2 discharge cycles
Auto-CNN-LSTM[61]	0.051	-	21 features	14 cycles

The RMSE and MAD of the CNN trained for normalized RUL prediction are 0.075 and 0.057, respectively. The corresponding metrics for the baseline linear regression model are 0.27 and 0.093, which again demonstrate that a non-linear model is suitable. The numbers are presented in table 4.3, together with the Auto-CNN-LSTM model by Ren et al. [61]. Ren et al. use 21 manually extracted features from charge and discharge voltage and current curves. The feature vectors for 14 adjacent cycles are stacked to an input feature map of size 14×21. Note that they use data from the NASA PcoE dataset [49]. Our model performs slightly worse but uses considerably fewer parameters and simpler input features.

Figure 4.6a displays the predicted normalized RUL against the actual normalized RUL, where a value of 1 corresponds to the first cycle of a cell, and a value of 0 corresponds to the EOL. The predictions are more evenly distributed around the line indicating the true value as compared to fig. 4.4a. The point cloud also exhibits a wave-like shape. At the point right before the degradation starts to accelerate, the model may not anticipate the subsequent rapid decline, which could explain the observed overestimation between 0.1 and 0.4.

Figure 4.7 shows the predictions for four cells in the test set, which correspond to the cells presented in fig. 4.3. The predictions are considerably more consistent with the true values and exhibit greater similarity across the four cells. The CNN does not have the ability to incorporate information about previous cycles into the prediction of the current cycle, so the predictions are only based on information embedded in IC curve from the current discharge cycle.



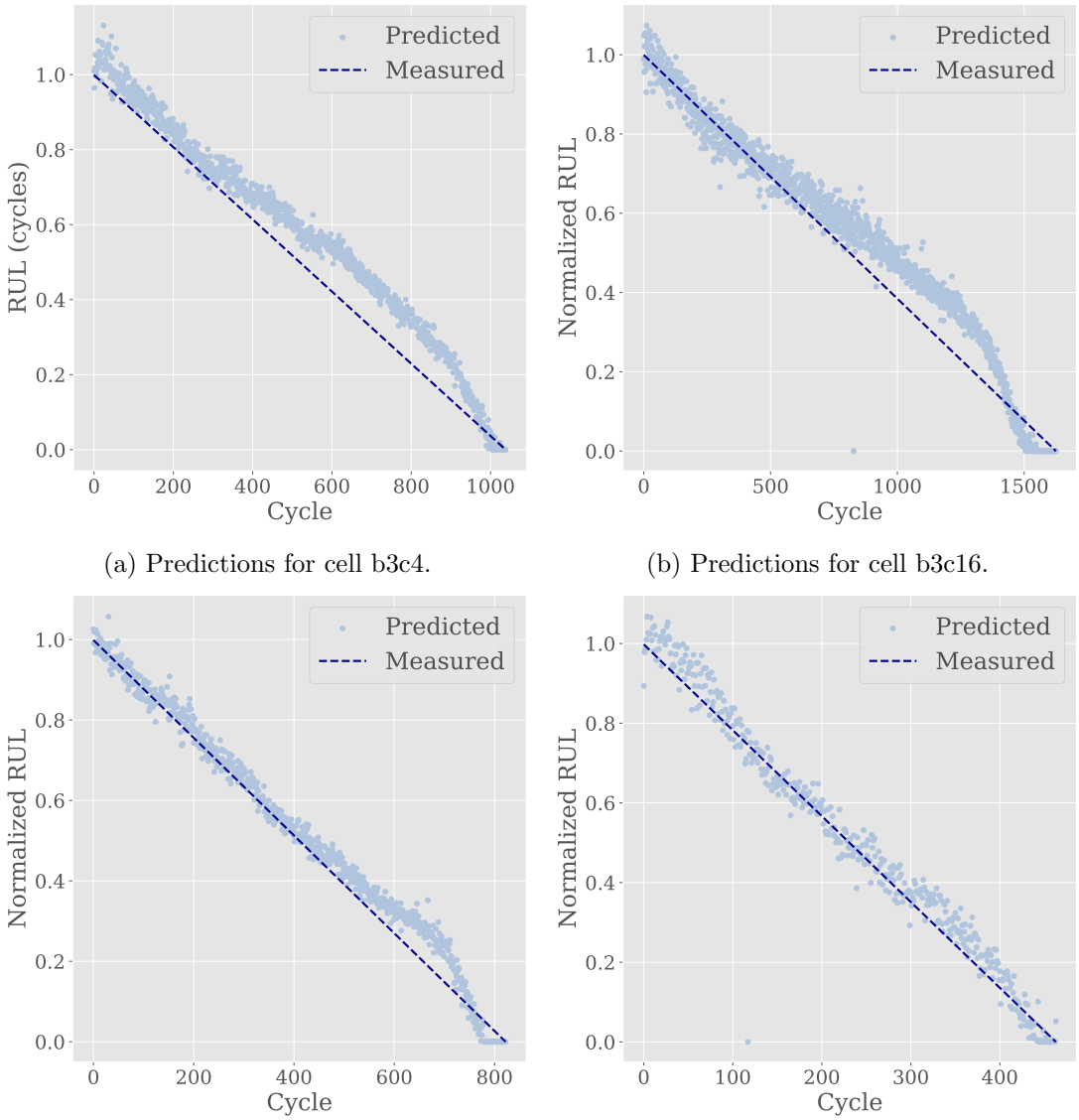
(a) Predicted versus actual normalized RUL for the complete test set.

(b) The mean average deviation as a function of the target value.

Figure 4.6: CNN performance for normalized RUL prediction on the complete test set.

Using normalized RUL significantly improves the predictions, particularly for the long-lived cell and in the beginning of cycle lives. This can be attributed to the characteristics of the normalization process. In the case of RUL given in cycles, early IC curves map to very different values depending on the EOL. When the RUL is normalized, on the other hand, the first cycle consistently maps to 1 independent of the EOL. This also reduces the effect of large errors at long cycle lives overly influencing the model. Additionally, the normalization appears to incentivize the model to put more emphasis on small values. While only one of the cells in fig. 4.3 have predictions of zero, all of the cells in fig. 4.7 now generate predictions of zero. In addition, normalizing RUL gives a more uniform distribution of target values, further contributing to the improved consistency of the model performance.

The results presented in fig. 4.6 support the hypothesis that the ΔdQdV curves contain sufficient information to be utilized for the prediction of normalized RUL. The CNN exhibits the ability to identify and relate patterns in the input to the relative location within the cycle life. However, when it comes to RUL predictions in units of cycles, the results suggest that additional augmentations such as additional features or a more sophisticated model might be necessary to improve performance.



(a) Predictions for cell b3c4.

(b) Predictions for cell b3c16.

(c) Predictions for cell b3c8.

(d) Predictions for cell b2c27.

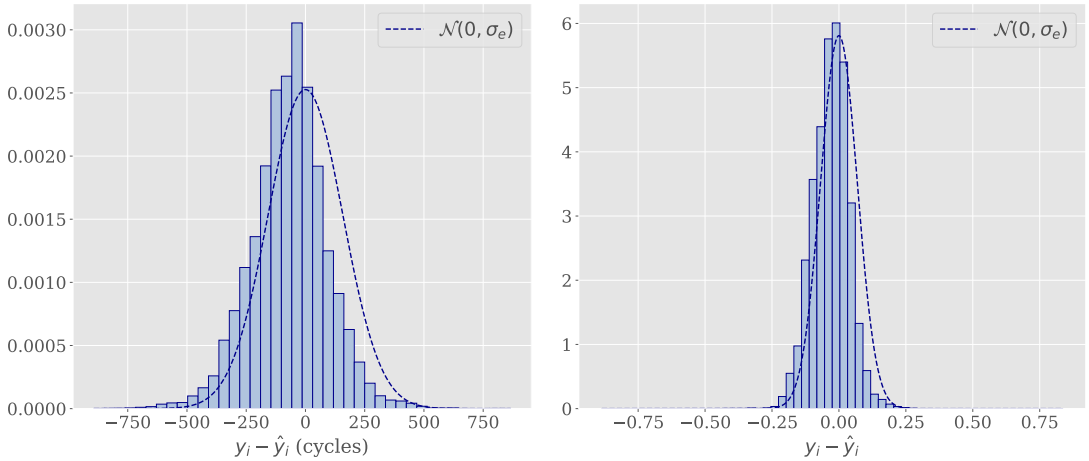
Figure 4.7: CNN predictions of normalized RUL for four cells in the test set.

4.2.3 Error Analysis

The task of mapping ΔdQdV to RUL utilizing a CNN is a supervised learning approach. By minimizing the MSE, we obtain a parameterization of the CNN that best fit the observed data. Essentially, this is maximum likelihood estimation under the assumption that the error is Gaussian and independently distributed [65].

Figure 4.8a shows the distribution of the error $y_i - \hat{y}_i$ between true value y_i and corresponding predicted value $\hat{y}_i$ in the test set for RUL predictions in units of cycles. The mean error is -66.7 cycles and the standard deviation σ_e is 157 cycles. For comparison, the Gaussian distribution with zero mean and standard deviation σ_e is included in the figure. The error distribution is close to symmetric about its mean value and resembles the bell shape of the Gaussian distribution. The distribution is slightly more clustered around zero and shifted towards negative values, which is consistent with the observed tendency to overestimate. However, the shape is very similar to the Gaussian distribution. The corresponding distribution in the case of normalized RUL predictions is displayed in fig. 4.8b. The distribution has a mean of -0.03 and standard deviation σ_e of 0.069 and exhibits an even better approximation of the Gaussian. The distribution is also narrower, indicating reduced variance.

In both cases, the model error can be considered close to Gaussian. Previous results have shown that the error is not independently distributed, but dependent on the target value. However, this is expected and can be considered acceptable behavior. From a statistical perspective, the results in fig. 4.8 provide justification for using the MSE loss function. The previous discussion also points to potential improvements by exploring alternative loss functions for optimizing the CNN.



(a) Normalized error distribution for RUL predictions.

(b) Normalized error distribution for normalized RUL predictions.

Figure 4.8: Normalized histograms showing the distribution of prediction errors $y_i - \hat{y}_i$ for every pair of target and predicted value $\{y, \hat{y}_i\}$ in the test set. The dashed line is the zero-mean Gaussian with standard deviation σ_e, where σ_e is the standard deviation of the error distribution. The error distributions are bell-shaped and close to symmetric around the mean.

The MSE loss function uses the squared error term, implying that larger errors are emphasized more. As the errors are generally larger at high RUL values, i. e. early in long-lived cell lives, these samples might disproportionally influence the weight update. If the model is 1% off in predicting a RUL of 2000, the MSE is 20^2. If the model is 1% off in predicting a RUL of 200, the MSE is 4. This means that the model might focus on improving the performance on very high RULs at the cost of performance closer to the EOL.

Few samples for high RULs can be a reason why the CNN consistently predicts lower RULs for targets above 1000 cycles. There might be too few samples to give accurate predictions in high regions. The fact at the large RUL are considerably fewer in number downweighs the accumulated contribution from these large errors, but the error terms might be large enough to nudge the model toward higher predictions overall. This could explain some of the reasons why the model tends to overestimate RULs below 1000 cycles. With such a wide spread of values, the model might be more incentivized to give up accuracy for very low RULs to gain some accuracy for higher RULs. We can observe from fig. 4.4a that the model seems reluctant to predict low values.

There exist other loss functions that weigh errors differently. The MAD loss function uses the absolute error and will emphasize all errors equally. More advanced loss functions as the Huber loss uses MSE loss for small errors and MAD loss for larger errors. The loss function can also be customized to meet more specific objectives. If accuracy toward the end of life is more important than in the beginning, this could be incorporated by penalizing errors toward the end of life more.

For this work, the MSE is considered a suitable option. The goal in terms of accuracy is to obtain a model good enough to make reasonable statements about the connection between CNN predictions and degradation observables. The focus has been on overall good performance. Without any additional constraints on what part of the predictions is most important, MSE is a reasonable choice. However, a different loss function might be more suitable in a more practical application, particularly for the prediction of RUL in units of cycles. Some additional objectives should be formulated to guide the design and evaluate if one loss function is better than another. This is considered outside of the scope of this work but is encouraged in a more practical application.

4.3 Looking Into the Black Box

The CNN, like many ML models, is a black-box approach. The processes and mechanisms governing the decision-making process within the model are not visible to the outside world. The inner workings of the model do not need to be understood for utilizing such models. This is one of the strengths of black-box approaches; however, it also poses challenges regarding transparency. Explainability analysis can help build trustworthy and accountable models and reveal potential errors and biases. Improving the understanding of the models might also uncover new and valuable insights into the underlying process being modeled, contributing to the knowledge of the field.

To get a better understanding of the aspects guiding the predictions of the model, we utilize the embedding vector (see fig. 3.4) and the weight of the linear layer. Given the consistent and improved performance of the CNN for normalized RUL predictions, we continue with this model for the following analysis.

4.3.1 Which Voltage Regions Are Important?

Exploring the weights of the linear layer is a natural starting point to identify which parts of the input contribute to decreasing the predicted RUL. Each of the three output channels of the feature extractor produces a feature vector, that together composes the embedding vector. The output of the CNN is the dot product of the linear layer weights and the embedding vector. Since the ReLU function is applied at every layer, all elements of the embedding vector will be positive. Consequently, a negative weight in the linear layer will contribute to decreasing the predicted normalized RUL. This correlation suggests that the receptive field of high negative weights contains signals indicating degradation.

First, we identify the negative weights in the linear layer. The negative weights are visualized in fig. 4.9a. A darker blue color indicates more negative weight. All non-negative weights are shown in white. The weights are stacked by channel, and the horizontal axis corresponds to the positions in the feature vector. Most of the negative weights are associated with channel 3. Furthermore, the weights are summed over the channels to get a better picture of the most negative regions overall. The sum of the weights is displayed in fig. 4.9b, using the same color bar. Most of the negative weights are located between nodes 19 and 51, with a higher density observed at the ends of this range. The hatched areas in the figure indicate the five positions along the embedding vector with the most negative weights combined. The corresponding receptive fields are displayed in fig. 4.9c. Two examples of ΔdQdV input curves are shown for reference. Negative weights assigned to these receptive fields indicate that the CNN has learned to associate certain features at these locations with reduced remaining life.

As described in section 2.1.3, the shape of the IC curves results from the electrochemical processes on the electrodes. Specifically, the peaks correspond to the graphite staging. At low currents, the peaks can be assigned to specific staging phases, but at high currents, the peaks broaden and merge together. Additionally, the IC peaks shift to lower voltages at higher currents due to increased over-potential in the cell. These effects are clearly visible in the IC curves for the graphite||LFP cell in this dataset. As depicted by fig. 3.3, the IC curve appears as a single broad peak from 2.85V to 3.25V, with a small shoulder appearing at around 3.09V. Considering the discussion in section 2.1.3, graphite staging is expected to occur in this region; however, it is infeasible to distinguish the individual stages.

Receptive fields A, B, and E cover the region of the IC curve that can be assigned to graphite staging. As the cell approaches the end of life, we expect to observe changes in intensity, location, and width of the peak. Such changes will appear in the ΔdQdV curves. The negative weights perceiving these areas suggest that the CNN identifies

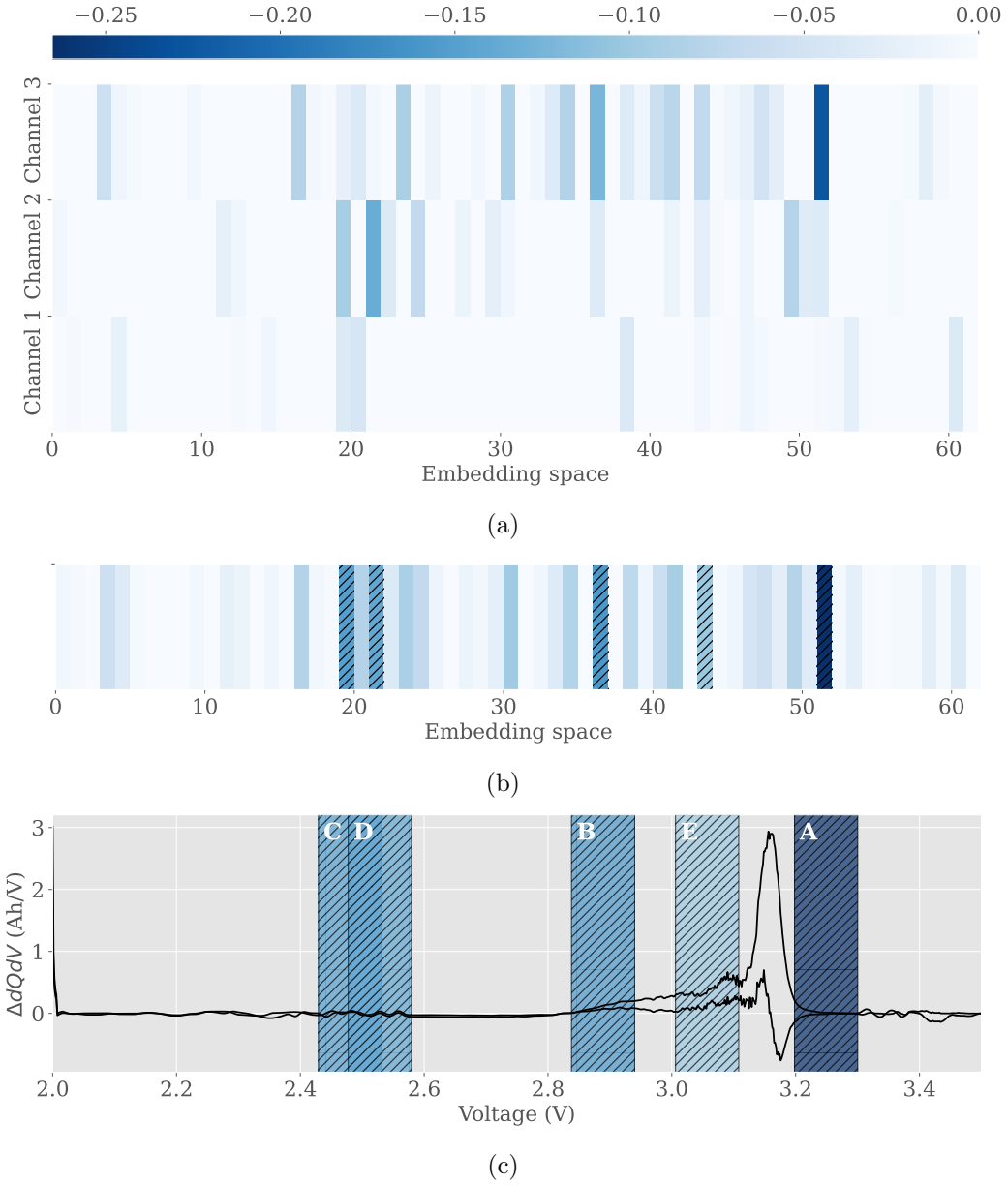


Figure 4.9: Visualization of the weights in the linear layer. (a) Heatmap of the negative weights of the linear layer. Darker blue indicates more negative weights and all non-negative weights are shown in white. (b) Weights summed over the three channels, with the same color bar. The hatched area indicates the five positions with the most negative weights combined. (c) The receptive field corresponding to the nodes with the most negative weights combined. They are labeled as A to E from most negative to least negative associated weights. Two examples of ΔdQdVs are included for reference.

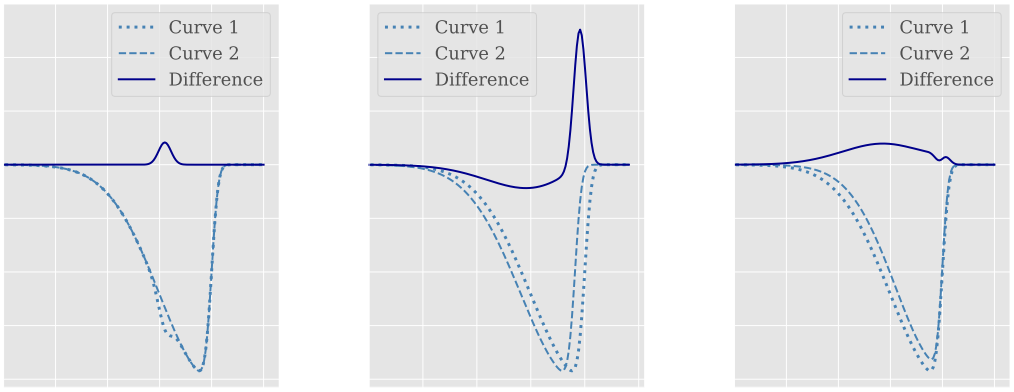
signals relating to the staging mechanisms and interprets these as signs of degradation.

High negative weights are also associated with the receptive fields C and D, covering the area from 2.43V to 2.53V. This voltage region is associated with SEI-formation and electrolyte reduction [66], suggesting that the network has identified patterns related to SEI stability that influence the EOL.

4.3.2 Which Patterns Are Important?

The receptive fields of the negative nodes indicate where the CNN picks up signs of an aging cell. However, it does not say anything about which patterns the CNN recognized. To understand this, we need to look at the embedding vector activations. For a negative linear node to reduce the output value, the corresponding activation must be non-zero. The higher the activation, the more the output decreases.

To find this, the embeddings for the training set were obtained by passing all training samples through the CNN and extracting the intermediate embedding representation. Then, the 20 curves yielding the maximum activations at the nodes of interest were identified. These input curves represent the patterns that give the maximum activation at the most negative nodes.



(a) Peak disappearing (b) Left shift (c) Right shift of left tail

Figure 4.10: Toy plots visualizing how shifts and intensity reductions manifest when subtracting two curves.

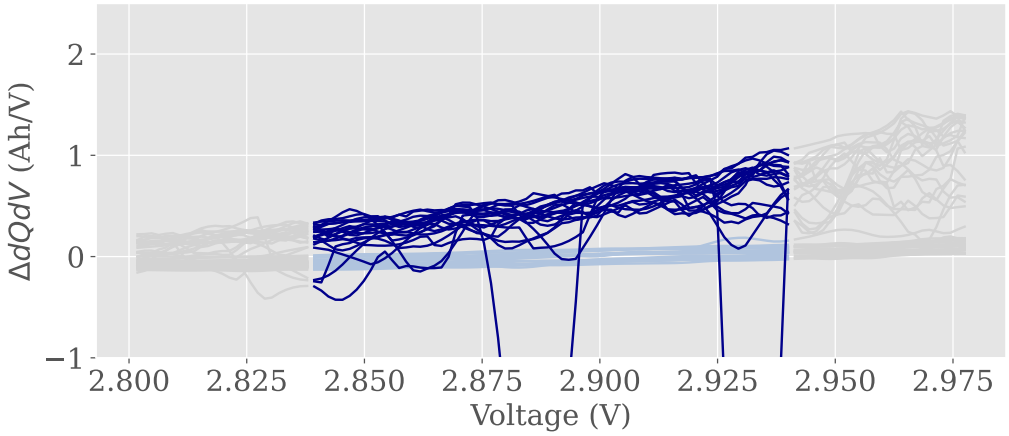
Before exploring the patterns appearing, let us take a step back and clarify how the shape of the ΔdQdV curves relates to changes in the IC curves. A decrease in the magnitude of a peak in a specific IC curve will appear as a positive peak in our ΔdQdV, as displayed in fig. 4.10a. The width and intensity reflect the original shape of the peak. As evident from the toy example in fig. 4.10c, a shift may also appear as a peak in the difference curve. A slight shift in a steep curve results in a high-intensity peak, as illustrated in fig. 4.10b. A shift in the gentle slope results in a broad and low peak. Figure 4.10c exemplifies how a reduction of the most prominent peak locally appears as a shift of the

gentle slope, resulting in the same plateau-like shape. Because the slope is shifted to higher voltages as opposed to the example in fig. 4.10b, the ΔdQdV peak is positive. A combination of these patterns may simultaneously be present in the ΔdQdV curves.

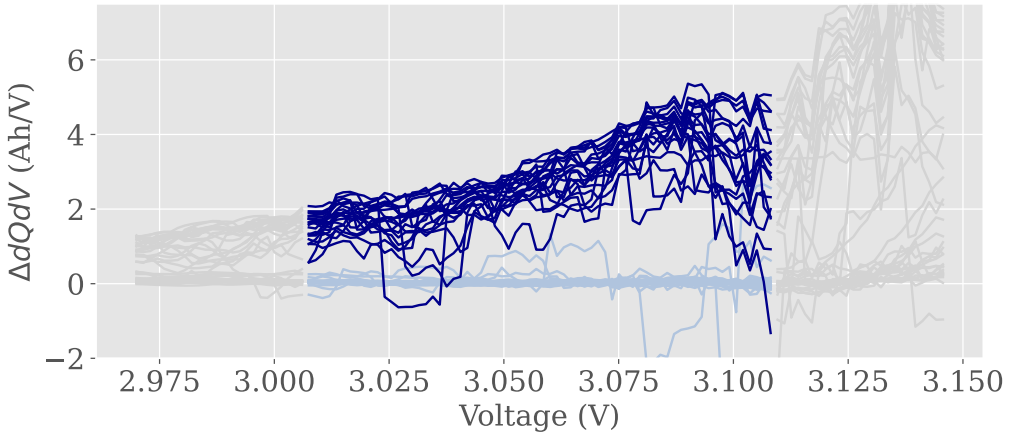
The 20 inputs yielding maximum activations in receptive field B at channel 3 are visualized as dark blue lines in fig. 4.11a. The light blue are examples of input curves that give zero activation. These are included to help distinguish which input patterns are triggering the activation. The gray area represents the input outside of the receptive field. This is not visible for the node but is included for easier interpretation. The receptive field of the node covers the interval from 2.84V to 2.94V. The activating inputs clearly differ from the non-activation examples and are characterized by being positive, relatively flat, and increasing toward higher voltages. In accordance with fig. 4.10c, this shape of the ΔdQdV originates from a shift of the gentle IC slope to higher voltages. The IC peak generally tends to shift towards lower voltages, as illustrated in fig. 3.3b. Thus, the shape in the activating patterns can be attributed to the narrowing of the IC peak. A narrower peak means less capacity available in this voltage window. Considering that the peaks in this voltage region can be assigned to graphite staging, the signal that the CNN picks up seems to correspond with the anode's reduced ability to accept lithium ions at this voltage.

Figure 4.11b shows the most activating inputs in receptive field E at channel 3, covering 3.01V to 3.11V. The activating inputs exhibit the same characteristics as described above. This receptive field also includes the region where the shoulder peak of the initial IC curve typically appears. Looking at the maximum activations, a small bump is visible around 3.09V. This can be interpreted as a peak reduction. This pattern does not appear in the zero-activation examples. The CNN does not necessarily recognize the shape of the bump itself, but rather the increased value of the ΔdQdV. The CNN seems to identify the positive plateau in the ΔdQdV originating from the change in slope, in addition to the intensity reduction or disappearance of the shoulder peak, indicating that the capacity is less accessible at the corresponding staging phase, at least at that particular voltage.

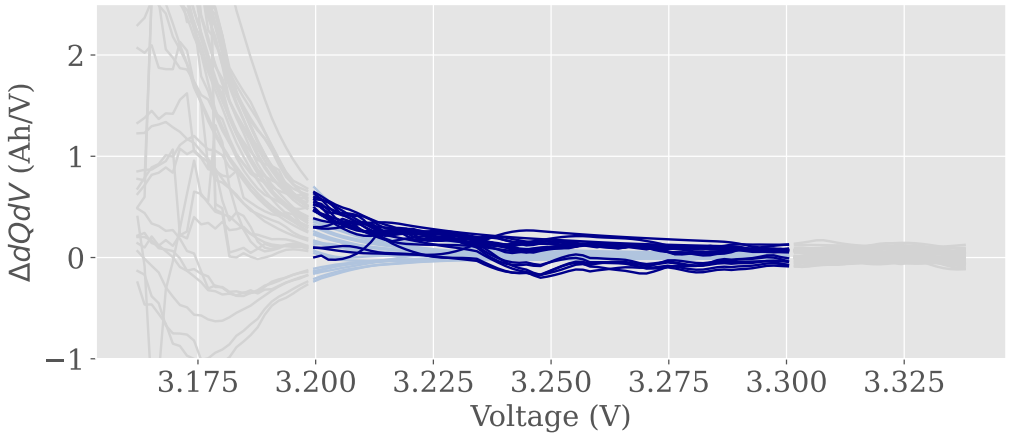
Receptive field A from 3.20V to 3.30V corresponds to the most negative linear weights combined, and the weight at channel 3 is the most negative individual weight overall. The 20 inputs giving maximum activation at this node are displayed in fig. 4.11c. While the two previous activation plots have shown a clear difference between maximum activating patterns and non-activating patterns, the distinction is not as obvious in this case. The dark blue curves have a flat shape close to zero and partially overlap with the non-activating patterns. The receptive field covers the region just at the beginning of the ΔdQdV peak corresponding to the shift of the IC peak (see fig. 4.10b). However, some of the curves with zero activation also include this shift. The activating patterns seem to have slightly higher values at the beginning of the peak, and in the right half of the window, the non-activating patterns are more tightly centered on zero. The fact that the maximum and minimum activation inputs appear very similar could indicate the variations between ΔdQdVs in this region are small. The high weight assigned to this node can be a way to put more emphasis on a weak signal.



(a) Node 36

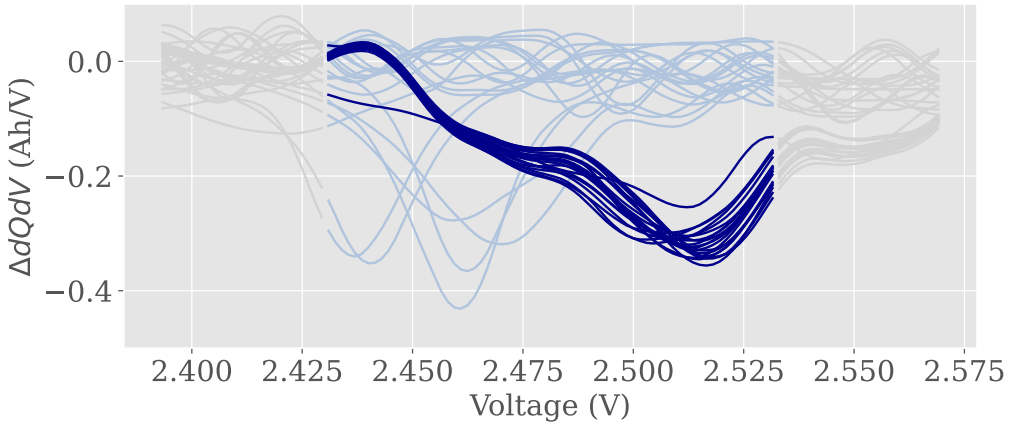


(b) Node 43

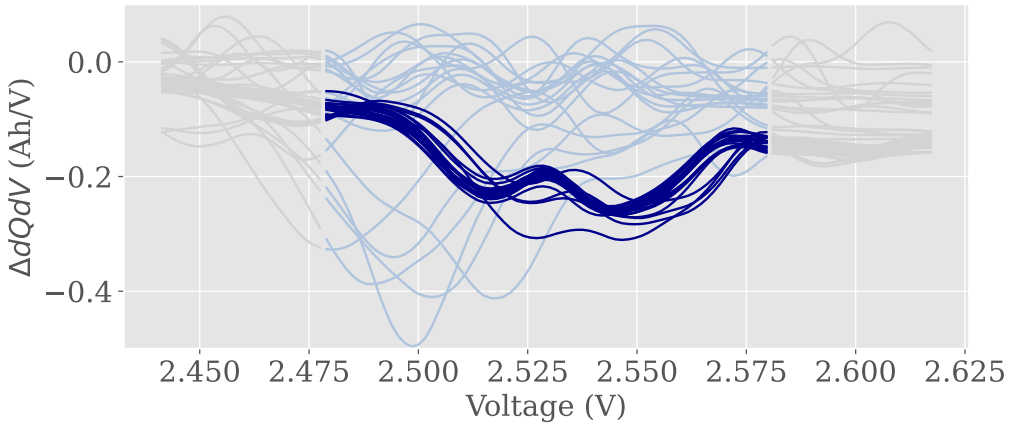


(c) Node 51

Figure 4.11: Maximum activating inputs of three nodes at channel 3. The light blue lines are examples of non-activation inputs.



(a) Node 19



(b) Node 21

Figure 4.12: Maximum activating inputs of two nodes at channel 2. The light blue lines are examples of non-activation inputs.

For receptive fields C and D, the inputs giving maximum activation at channel 2 are displayed in fig. 4.12a and fig. 4.12b, respectively. The receptive fields of these two nodes are overlapping and cover voltages between 2.43V to 3.30V. In both plots, the activations seem to be related to negative peaks. A negative peak in the ΔdQdV curve corresponds to increased intensity or appearance of a new peak relative to the initial IC curve.

As evident in both fig. 4.12a and fig. 4.12b, the position and shape of the peak within the receptive field seem to influence the activation. fig. 4.12a, displays peaks appearing around 2.45V, however, these pattern results in zero activation. The same is observable in fig. 4.12b, where the non-activating peaks also have a significantly higher intensity than the activating patterns. This could imply that a negative peak must be located in the middle or right part of the receptive field to be observable for a node at channel 2. The activation patterns also exhibit a more oscillating behavior than the non-activating peaks, suggesting that this might also be a feature that the CNN identifies.

The CNN exhibits the ability to find consistent patterns to distinguish between cycles early and late in life. The regions that seem most important for the CNN to predict low RUL are those associated with graphite staging. Input from 2.43V to 3.30V also seem to contribute toward lower RUL. We can observe degradation from IC curves under fast cycling, demonstrating the potential to also use fast-charging IC curves for RUL prediction and SOH monitoring, representing more realistic operation conditions compared to conventional slow-charging IC curves.

4.4 Lessons Learned

Looking back at the project, it is important to reflect on alternative approaches that could have been taken and acknowledge the limitations of the results.

Given more time, a more sophisticated approach to noise handling could have been pursued. The dataset contains some IC curves significantly influenced by measurement noise. The strategy for handling noisy curves involved a naive approach where samples with some noise above a threshold were removed. As no improvement was observed when removing outliers at various thresholds, it was decided not to remove any additional samples than those deemed noisy by [9]. However, this decision was made using a preliminary model before the final model was obtained. It is possible that the strategy could have given a different outcome using the final model and other adjustments introduced during the process. Employing a smoothing algorithm such as bin smoothing, simple moving average, or local weighted regression as a part of the pre-processing or integrated with the ML model could have been beneficial.

We also decided not to normalize or standardize the input data in order to do minimal alternations to the data. Such a transformation adds a layer of processing that would make it more challenging to relate the patterns observed by the CNN to the corresponding physical observations.

These strategies related to pre-processing of input data could have been explored more in-depth. The main reasons for not doing this were the time constraints and the intention of preserving the physical meaning of the input. However, this might have limited the performance in terms of accuracy. These design choices could be reconsidered for future applications where accuracy is of higher priority.

A limitation of the model is the high RMSE compared to the current state-of-the-art, particularly when predicting RUL in cycles. This should be considered when leveraging the CNN's behavior to establish connections between IC curves and battery degradation. Acknowledging that less accurate predictions can introduce uncertainty into the arguments presented in section 4.3 is important.

A related limitation is an implicit assumption that non-linear features are presented within a region of 69 input elements. The model's output is a linear transformation of the embedding vector (with ReLU activation), implying that the model assumes a close-to-linear mapping between the embedding and the RUL. Non-linear relationships are primarily captured by the feature extractor. The feature extractor can identify non-linear patterns within its receptive fields, which cover an area of 69 consecutive input elements. A deeper model with a broader receptive field could be employed to capture non-linear relationships across a wider portion of the input. This might improve accuracy but compromise interpretability. Again, the objective of simplicity has guided the design of the CNN, but it is an aspect to be aware of.

We utilized the IC curves from the dataset directly. Computing an IC curve involves a numerical differentiation of the voltage profile. The specific method used for differentiating the discharge capacity curve is unknown, which introduces the possibility of numerical or interpolation effects influencing the IC curves and subsequently being interpreted by the CNN. We cannot completely discard the possibility that the differentiation algorithm introduces certain artifacts and that these are what we observed in fig. 4.12 considering their smooth behavior. However, the patterns in the specific voltage region still correlate strongly with decreasing RUL. We expect differentiation artifacts to introduce random noise in the inputs and drive accuracy down, but it is unlikely such patterns would be as correlated to RUL as it is observed in our results. Even if certain patterns result from the differentiation algorithm, an underlying factor still drives these patterns to change as the cell ages. If some patterns that the CNN associates with decreased RUL result from the differentiation, it suggests that degradation plays a role in the emergence of these artifacts.

5 Conclusion and Further Work

5.1 Conclusion

In this work, we have exploited ML models for lifetime predictions of LIB. Specifically, we have proposed a shallow 1D-CNN model utilizing IC difference curves between any discharge cycle and the initial discharge cycle to predict RUL, both normalized and in units of cycles. Then, we analyzed the inner workings of the model to gain insight into underlying processes guiding the decisions of the model. The proposed CNN architecture has a shallow structure, aiming to balance accuracy and interpretability. Despite the simplistic structure and limited input features, the CNN achieves accuracy comparable to related models.

The proposed CNN achieves good accuracy when predicting normalized RUL and exhibits the ability to connect spatial features of the ΔdQdV curves to the relative location within the cycle life. This suggests that the IC curves encapsulate sufficient information about the state of the cell to enable accurate predictions of normalized RUL predictions. The CNN demonstrates reasonable accuracy when predicting RUL in units of cycles, but falls short of state-of-the-art performance, particularly for long-lived cells. This can be attributed to the wide range of target values and uneven distribution of training data, in combination with the simple modeling approach. However, the model clearly demonstrates the ability to identify and establish patterns in ΔdQdV curves related to RUL.

The results emphasize the usefulness of CNN-processed ΔdQdV curves for lifetime predictions. While the performance in normalized RUL predictions is promising, the results indicate that relying solely on ΔdQdV discharge curves may not be sufficient for accurate RUL predictions. Further enhancements to the model or the inclusion of additional features should be considered to improve prediction accuracy and robustness. It is worth highlighting that the model was not designed to achieve the highest predictive performance possible but also to enable interpretation of the decision-making process of the CNN. This emphasis resulted in a simpler model compared to other deep learning approaches. Nevertheless, the CNN demonstrates satisfactory performance.

Furthermore, we have explored which part of the input the CNN correlates with low RUL. The voltage window corresponding to graphite staging emerges as one of the important regions. The CNN identifies patterns related to reduced accessible capacity in this voltage window. The CNN also correlates low remaining life with the presence of peaks in the IC curves at voltages between 2.43V and 2.53V. While the origin of these peaks is not certain, the findings suggest that they may be related to some underlying degradation process. Overall, the CNN identifies consistent patterns indicating degradation.

5.2 Further Work

The scope of this work can be further expanded to address some remaining challenges that have been identified, and also to better tailor it for both online battery monitoring and degradation diagnostics. To start with, conducting a sensitivity analysis can provide insight into the reliability of the model and further examine the validity of the findings, particularly the consistency of the identified patterns and regions. This would involve systematically training and evaluating the CNN with various initialization or small model adjustments. If the patterns and input regions that correlate with low RUL remain consistent across different initializations or model adjustments, it would strengthen the validity of our conclusions. Performing a sensitivity analysis might be useful for assessing the robustness of the CNN performance and the credibility of the insights derived from the model.

There are several avenues to explore in order to enhance the explainability of the CNN. While this work has used the weights of the linear layer as a starting point for investigating the predictions of the CNN, focusing on the activations within the CNN could be an alternative approach. By exploring the activations at different layers, it may be possible to gain further insight into the decision-making process of the model, and also get a more comprehensive picture of the mechanisms influencing the predictions. Additionally, there exist other explainability techniques such as LIME, SHAP, and layerwise relevance propagation that could provide complementary information. Improving the understanding of the ML models for lifetime predictions and SOH estimation is crucial for ensuring reliable and transparent models and may also reveal new and valuable insight into the underlying degradation processes.

One of the main limitations of onboard IC analysis for SOH estimation in BMS is the requirement of IC curves from low current rates [25, 26], typically C/25. Diagnostics involves stopping the normal operation of the battery system to perform check-up cycles at low rates, which is undesirable due to the outage. Instead, this work demonstrates the potential of utilizing IC curves obtained from fast discharging at 4C for RUL prediction. While the proposed strategy may not be directly applicable in a BMS, this work demonstrates the potential of fast-rate IC curves for battery monitoring. Adjustments are necessary to make the approach sufficiently accurate to be feasible in a practical application. Such adjustments could include employing a deeper model, incorporating more complex ML components, and using additional input features. For instance, we have seen that the predictions correlate with the internal cell temperature, and adding this feature might help the model account for this effect. Additional measurements could enable the model to better capture the relationship between correlating factors and further improve accuracy. The proposed model has quite a simple architecture, and it is reasonable to assume that a more complex model fed with richer input data will achieve better predictive performance. Enhancements such as adding layers to increase the receptive field or incorporating recurrent layers to capture time dependencies between cycles can further improve the model's performance.

Although the strategy uses fast-rate IC curves, the cycling protocols are not representative of the real-life operation of an electric vehicle. To further assess the utility of

this kind of fast-rate IC for lifetime predictions, the approach should be evaluated on more realistic driving cycles. There are also challenges related to measurement noise and performing the online numerical differentiation for obtaining the IC curves [25, 26] that need to be addressed. Further research efforts focused on how degradation modes manifest in IC curves under high current rates have the potential to uncover new insights and enhance the utility of fast-rate IC curves.

Another interesting path to explore is to extend this strategy to other cathode materials using a transfer learning approach. In transfer learning, knowledge gained from a model trained on one task is applied to a different but related task. Transfer learning might accelerate the training on a new chemistry by initializing the model with parameters from a model pre-trained on another chemistry. This CNN might be suited to generalize to other chemistries, for instance, by freezing the parameters of the feature extractor and only updating the linear layer. Successful transfer learning assumes that features learned by the original model are general for both the original dataset and the new task [40]. Our CNN is trained on the difference between IC curves, potentially giving the CNN the ability to extract more general features relating to peak reductions and shifts in the IC curve. Even if the shape of the IC curve might vary between different chemistries and conditions, changes in the IC curves are still expected to occur as the cell degrades. Furthermore, less data is needed to adapt the technique to other chemistries. If a transfer learning approach can be applied successfully, the new dataset can be significantly smaller than the original dataset [67]. This is particularly advantageous considering the time-consuming and costly nature of obtaining battery cycling aging data.

In conclusion, there are several directions for further investigation and improvement. By pursuing these research areas, we can advance the understanding and application of CNN-processed fast-rate IC curves in the context of battery degradation and lifetime predictions.

Bibliography

[1] IPCC. Summary for policymakers. In: Climate Change 2023: Synthesis Report. A Report of the Intergovernmental Panel on Climate Change. Contribution of Working Groups I, II and III to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change [Core Writing Team, H. Lee and J. Romero (eds.)], 2023. (in press).

[2] UN. Causes and Effects of Climate Change. https://www.un.org/en/climatechange/science/causes-effects-climate-change, 2023. Accessed: 2023-05-27.

[3] World Health Organization. COP24 Special Report: Health and Climate Change, 2018. Licence: CC BY-NC-SA 3.0 IGO.

[4] John B. Goodenough and Youngsik Kim. Challenges for Rechargeable Li Batteries. *Chemistry of Materials*, 22(3):587–603, 2010.

[5] Beth E. Murdock, Kathryn E. Toghill, and Nuria Tapia-Ruiz. A Perspective on the Sustainability of Cathode Materials Used in Lithium-Ion Batteries. *Advanced Energy Materials*, 11(39), 2021.

[6] Christoph R. Birkl, Matthew R. Roberts, Euan McTurk, Peter G. Bruce, and David A. Howey. Degradation diagnostics for lithium ion cells. *Journal of Power Sources*, 341:373–386, 2017.

[7] Mohammad Shahjalal, Probir Kumar Roy, Tamanna Shams, Ashley Fly, Jahedul Islam Chowdhury, Md. Rishad Ahmed, and Kailong Liu. A review on second-life of li-ion batteries: Prospects, challenges, and issues. *Energy*, 241:122881, 2022.

[8] Wladislaw Waag, Christian Fleischer, and Dirk Uwe Sauer. Critical Review of the Methods for Monitoring of Lithium-ion Batteries in Electric and Hybrid Vehicles. *Journal of Power Sources*, 258:321–339, 2014.

[9] Kristen A. Severson, Peter M. Attia, Norman Jin, Nicholas Perkins, Benben Jiang, Zi Yang, Michael H. Chen, Muratahan Aykol, Patrick K. Herring, Dimitrios Fraggedakis, Martin Z. Bazant, Stephen J. Harris, William C. Chueh, and Richard D. Braatz. Data-driven prediction of battery cycle life before capacity degradation. *Nature Energy*, 4(5):383–391, May 2019.

[10] Gavin Harper, Roberto Sommerville, Emma Kendrick, Laura Driscoll, Peter Slater, Rustam Stolkin, Allan Walton, Paul Christensen, Oliver Heidrich, Simon Lambert, Andrew Abbott, Karl Ryder, Linda Gaines, and Paul Anderson. Recycling Lithium-Ion Batteries From Electric Vehicles. *Nature*, 575(7781):75–86, 11 2019.

[11] E. Prada, D. Di Domenico, Y. Creff, J. Bernard, V. Sauvant-Moynot, and F. Huet. A Simplified Electrochemical and Thermal Aging Model of LiFePO4-Graphite Li-ion Batteries: Power and Capacity Fade Simulations. *Journal of The Electrochemical Society*, 160(4):A616, feb 2013.

[12] J. Li, K. Adewuyi, N. Lotfi, R.G. Landers, and J. Park. A Single Particle Model With Chemical/Mechanical Degradation Physics for Lithium Ion Battery State of Health (SOH) Estimation. *Applied Energy*, 212:1178–1190, 2018.

[13] Rutooj Deshpande, Mark Verbrugge, Yang-Tse Cheng, John Wang, and Ping Liu. Battery Cycle Life Prediction with Coupled Chemical Degradation and Fatigue Mechanics. *Journal of The Electrochemical Society*, 159(10):A1730, aug 2012.

[14] Kenji Takahashi and Venkat Srinivasan. Examination of Graphite Particle Cracking as a Failure Mode in Lithium-Ion Batteries: A Model-Experimental Study. *Journal of The Electrochemical Society*, 162(4):A635, jan 2015.

[15] Samuel Greenbank and David Howey. Automated Feature Extraction and Selection for Data-Driven Models of Rapid Battery Capacity Fade and End of Life. *IEEE Transactions on Industrial Informatics*, 18(5):2965–2973, 2022.

[16] Robert R. Richardson, Michael A. Osborne, and David A. Howey. Battery health prediction under generalized conditions using a Gaussian process transition model. *Journal of Energy Storage*, 23:320–328, 2019.

[17] Meru A. Patil, Piyush Tagade, Krishnan S. Hariharan, Subramanya M. Kolake, Taewon Song, Taejung Yeo, and Seokgwang Doo. A novel multistage support vector machine based approach for Li ion battery remaining useful life estimation. *Applied Energy*, 159:285–297, 2015.

[18] Jingwen Wei, Guangzhong Dong, and Zonghai Chen. Remaining Useful Life Prediction and State of Health Diagnosis for Lithium-Ion Batteries Using Particle Filter and Support Vector Regression. *IEEE Transactions on Industrial Electronics*, 65(7):5634–5643, 2018.

[19] Datong Liu, Wei Xie, Haitao Liao, and Yu Peng. An Integrated Probabilistic Approach to Lithium-Ion Battery Remaining Useful Life Estimation. *IEEE Transactions on Instrumentation and Measurement*, 64(3):660–670, 2015.

[20] Xiaoyu Li, Lei Zhang, Zhenpo Wang, and Peng Dong. Remaining useful life prediction for lithium-ion batteries based on a hybrid model combining the long short-term memory and elman neural networks. *Journal of Energy Storage*, 21:510–518, 2019.

[21] Jiantao Qu, Feng Liu, Yuxiang Ma, and Jiaming Fan. A Neural-Network-Based Method for RUL Prediction and SOH Monitoring of Lithium-Ion Battery. *IEEE Access*, 7:87178–87191, 2019.

[22] Laura Hannemose Rieger, Eibar Flores, Kristian Frellesen Nielsen, Poul Norby, Elixabete Ayerbe, Ole Winther, Tejs Vegge, and Arghya Bhowmik. Uncertainty-aware and explainable machine learning for early prediction of battery degradation trajectory. *Digital Discovery*, 2:112–122, 2023.

[23] Matthieu Dubarry, Cyril Truchot, and Bor Yann Liaw. Synthesize battery degradation modes via a diagnostic and prognostic model. *Journal of Power Sources*, 219:204–216, 2012.

[24] David Anseán, Víctor Manuel García, Manuela González, Cecilio Blanco-Viejo, Juan Carlos Viera, Yoana Fernández Pulido, and Luciano Sánchez. Lithium-Ion Battery Degradation Indicators Via Incremental Capacity Analysis. *IEEE Transactions on Industry Applications*, 55(3):2992–3002, 2019.

[25] Rui Xiong, Linlin Li, and Jinpeng Tian. Towards a Smarter Battery Management System: A Critical Review on Battery State of Health Monitoring Methods. *Journal of Power Sources*, 405:18–29, 2018.

[26] Sunil K. Pradhan and Basab Chakraborty. Battery Management Strategies: An Essential Review for Battery State of Health Monitoring Techniques. *Journal of Energy Storage*, 51:104427, 2022.

[27] Jinchao Liu, Margarita Osadchy, Lorna Ashton, Michael Foster, Christopher J. Solomon, and Stuart J. Gibson. Deep convolutional neural networks for Raman spectrum recognition: a unified solution. *Analyst*, 142:4067–4074, 2017.

[28] Salim Malek, Farid Melgani, and Yakoub Bazi. One-dimensional convolutional neural networks for spectroscopic signal regression. *Journal of Chemometrics*, 32(5), 2018.

[29] Xiaolei Zhang, Tao Lin, Jinfan Xu, Xuan Luo, and Yibin Ying. DeepSpectra: An end-to-end deep learning approach for quantitative spectral analysis. *Analytica Chimica Acta*, 1058:48–57, 2019.

[30] Asifullah Khan, Anabia Sohail, Umme Zahoora, and Aqsa Saeed Qureshi. A survey of the Recent Architectures of Deep Convolutional Neural Networks. *Artificial Intelligence Review*, 53(8):5455–5516, Dec 2020.

[31] John B. Goodenough and Kyu-Sung Park. The Li-Ion Rechargeable Battery: A Perspective. *Journal of the American Chemical Society*, 135(4):1167–1176, 2013.

[32] Jakob Asenbauer, Tobias Eisenmann, Matthias Kuenzel, Arefeh Kazzazi, Zhen Chen, and Dominic Bresser. The success story of graphite as a lithium-ion anode material – fundamentals, remaining challenges, and recent developments including silicon (oxide) composites. *Sustainable Energy Fuels*, 4:5387–5416, 2020.

[33] Matthieu Dubarry and Bor Yann Liaw. Identify capacity fading mechanism in a commercial LiFePO4 cell. *Journal of Power Sources*, 194(1):541–549, 2009.

[34] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S. Emer. Efficient Processing of Deep Neural Networks: A Tutorial and Survey. *Proceedings of the IEEE*, 105(12):2295–2329, 2017.

[35] Andrea Apicella, Francesco Donnarumma, Francesco Isgrò, and Roberto Prevete. A survey on modern trainable activation functions. *Neural Networks*, 138:14–32, 2021.

[36] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer Feedforward Networks are Universal Approximators. *Neural Networks*, 2(5):359–366, 1989.

[37] Michael A. Nielsen. Neural Networks and Deep Learning. http://neuralnetworksanddeeplearning.com/index.html, 2015. Accessed: 2022-13-12.

[38] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. arXiv:1412.6980.

[39] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization, 2019. arXiv:1711.05101.

[40] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2017.

[41] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *CoRR*, abs/1502.03167, 2015.

[42] Dang Ha The Hien. A guide to receptive field arithmetic for Convolutional Neural Networks. https://blog.mlreview.com/a-guide-to-receptive-field-arithmetic-for-convolutional-neural-networks-e0f514068807, 2017. Accessed: 2023-05-21.

[43] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders, 2020. arXiv:2003.05991.

[44] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks, 2014. arXiv:1406.2661.

[45] M. Stone. Cross-Validatory Choice and Assessment of Statistical Predictions. *Journal of the Royal Statistical Society: Series B (Methodological)*, 36(2):111–133, 1974.

[46] Tadayoshi Fushiki. Estimation of prediction error by using K-fold cross-validation. *Statistics and Computing*, 21(2):137–146, 2011.

[47] Juan D. Rodriguez, Aritz Perez, and Jose A. Lozano. Sensitivity analysis of k-fold cross validation in prediction error estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(3):569–575, 2010.

[48] Bruce G. Marcot and Anca M. Hanea. What is an optimal value of k in k-fold cross-validation in discrete bayesian network analysis? *Computational Statistics*, 36:2009–2031, 2020.

[49] Bhaskar Saha and Kai Goebel. Battery Data Set. NASA Prognostics Data Repository, 2007.

[50] Peter M. Attia, Aditya Grover, Norman Jin, Kristen A. Severson, Todor M. Markov, Yang-Hung Liao, Michael H. Chen, Bryan Cheong, Nicholas Perkins, Zi Yang, Patrick K. Herring, Muratahan Aykol, Stephen J. Harris, Richard D. Braatz, Stefano Ermon, and William C. Chueh. Closed-loop optimization of fast-charging protocols for batteries with machine learning. *Nature*, 578(7795):397–402, Feb 2020.

[51] CALCHE. Battery Data. https://calce.umd.edu/battery-data, 2016. Accessed: 2023-05-21.

[52] Kristen A. Severson, Peter M. Attia, Patrick K. Herring, Richard D. Braatz, and Martin Hwasser. Data-Driven Prediction of Battery Cycle Life before Capacity Degradation, 2019. Accessed: 2023-04-07.

[53] Paul H. C. Eilers. A perfect smoother. *Analytical Chemistry*, 75(14):3631–3636, 2003.

[54] Jie Yang, Jinfan Xu, Xiaolei Zhang, Chiyu Wu, Tao Lin, and Yibin Ying. Deep learning for vibrational spectral analysis: Recent progress and a practical guide. *Analytica Chimica Acta*, 1081:6–17, 2019.

[55] Lanfa Liu, Min Ji, and Manfred Buchroithner. Transfer Learning for Soil Spectroscopy Based on Convolutional Neural Networks and Its Application in Soil Clay Content Mapping Using Hyperspectral Imagery. *Sensors*, 18(9):3169, 2018.

[56] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[57] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library, 2019. arXiv:1912.01703.

[58] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[59] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework, 2019. arXiv:1907.10902.

[60] J. D. Hunter. Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[61] Lei Ren, Jiabao Dong, Xiaokang Wang, Zihao Meng, Li Zhao, and M. Jamal Deen. A Data-Driven Auto-CNN-LSTM Prediction Model for Lithium-Ion Battery Remaining Useful Life. *IEEE Transactions on Industrial Informatics*, 17(5):3478–3487, 2021.

[62] Calum Strange and Gonçalo dos Reis. Prediction of future capacity and internal resistance of li-ion cells from one cycle of input data. *Energy and AI*, 5:100097, 2021.

[63] Joonki Hong, Dongheon Lee, Eui-Rim Jeong, and Yung Yi. Towards the swift prediction of the remaining useful life of lithium-ion batteries with end-to-end deep learning. *Applied Energy*, 278:115646, 2020.

[64] Soo Seok Choi and Hong S Lim. Factors that affect cycle-life and possible degradation mechanisms of a li-ion cell based on licoo2. *Journal of Power Sources*, 111(1):130–136, 2002.

[65] Li Chai, Jun Du, Qing-Feng Liu, and Chin-Hui Lee. Using Generalized Gaussian Distributions to Improve Regression Error Modeling for Deep Learning-Based Speech Enhancement. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(12):1919–1931, 2019.

[66] Seong Jin An, Jianlin Li, Claus Daniel, Debasish Mohanty, Shrikant Nagpure, and David L. Wood. The state of understanding of the lithium-ion-battery graphite solid electrolyte interphase (sei) and its relationship to formation cycling. *Carbon*, 105:52–76, 2016.

[67] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks?, 2014. arXiv:1411.1792 [cs.LG].

NTNU

Norwegian University of
Science and Technology