REPUBLIC OF TURKEY
YILDIZ TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

PHYSICS GUIDED NEURAL NETWORK BASED STATE-OF-CHARGE ESTIMATOR FOR LITHIUM-ION BATTERIES

Fedi SALHI

MASTER OF SCIENCE THESIS
Department of Mechatronics Engineering
Program of Mechatroncis Engineering

Supervisor
Prof. Dr. Erhan AKDOĞAN

July, 2022

REPUBLIC OF TURKEY
YILDIZ TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

PHYSICS GUIDED NEURAL NETWORK BASED STATE-OF-CHARGE ESTIMATOR FOR LITHIUM-ION BATTERIES

A thesis submitted by Fedi SALHI in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE** is approved by the committee on 08.07.2022 in Department of Mechatronics Engineering, Program of Mechatroncis Engineering.

Prof. Dr. Erhan AKDOĞAN
Yildiz Technical University
Supervisor

Approved By the Examining Committee

Prof. Dr. Erhan AKDOĞAN, Supervisor
Yildiz Technical University _____

Prof. Dr. Songül VARLI, Member
Yildiz Technical University _____

Dr. Öğr. Üye. Gazi AKGÜN, Member
Marmara University _____

I hereby declare that I have obtained the required legal permissions during data collection and exploitation procedures, that I have made the in-text citations and cited the references properly, that I haven't falsified and/or fabricated research data and results of the study and that I have abided by the principles of the scientific research and ethics during my Thesis Study under the title of Physics Guided Neural Network Based State-of-Charge Estimator for Lithium-Ion Batteries supervised by my supervisor, Prof. Dr. Erhan AKDOĞAN. In the case of a discovery of false statement, I am to acknowledge any legal consequence.

Fedi SALHI

Signature

Dedicated to my family
and my best friend

ACKNOWLEDGEMENTS

I would like to thank my teacher and supervisor, *Prof. Dr. Erhan AKDOĞAN* for his patience and valuable support. I have gained a plethora of knowledge working with him over the years.

I also acknowledge the financial support from the *Republic of Türkiye* through the *Türkiye Scholarships* program dedicated for outstanding international students and researchers from all over the world.

I am grateful that I have been a part of *Yildiz Technical University*, one the most prestigious universities in Turkey, that provides an excellent atmosphere for learning and growth.

My Family deserves all my respect and gratitude, for the unconditional loving support. They have always been a source of motivation reminding me of what it is important in life.

Fedi SALHI

TABLE OF CONTENTS

LIST OF SYMBOLS vii

LIST OF ABBREVIATIONS ix

LIST OF FIGURES xii

LIST OF TABLES xiv

ABSTRACT xv

ÖZET xvii

1 INTRODUCTION 1

1.1 Literature Review . . . 3

1.1.1 Physics Guided Neural Networks . . . 3

1.1.2 State-of-the-art in State of Charge Estimation For Lithium-Ion Batteries . . . 10

1.2 Objective of the Thesis . . . 20

1.3 Original Contribution . . . 20

1.4 Method Synopsis . . . 21

1.5 Thesis Organization . . . 22

2 THEORETICAL BACKGROUND 23

2.1 Overview of the Convolutional Neural Networks Building Blocks . . . 23

2.1.1 Convolutional Layer . . . 25

2.1.2 Pooling Layer . . . 28

2.1.3 Fully-Connected Layer . . . 30

2.1.4 Batch Normalization Layer . . . 31

2.2 Cost Function . . . 33

2.3 Adam Optimizer . . . 34

3 SOC ESTIMATION USING PHYSICS GUIDED CONVOLUTIONAL NEURAL NETWORK 37

3.1 Panasonic 18650PF Lithium-ion Battery Data . . . 37

3.2 PGCNN and N-PGCNN Models Structure 43
3.3 Training Algorithm for PGCNN and N-PGCNN Models 47
3.4 Accuracy and Physics Consistency Test Results 54
3.4.1 Accuracy Evaluation Using MAPE Metric 54
3.4.2 Physics Consistency Evaluation Using PCI Metric 58

4 RESULTS AND DISCUSSION **60**

REFERENCES **62**

PUBLICATIONS FROM THE THESIS **68**

LIST OF SYMBOLS

$Q_{available}$	Available Capacity
BN_i	Batch Normalization Output
C_n	Battery Capacity
Q	Battery Rated Capacity
ΔSOC	Change in State-of-Charge
CNN	Convolutional Neural Network
η_i	Coulombic Efficiency
I	Electric Current
$i(t)$	Electric Current at Instant t
E_{Phy_0}	Energy Vector Corresponding to IN_{Phy_0}
$E_{Phy_{100}}$	Energy Vector Corresponding to $IN_{Phy_{100}}$
E	Expected Value
IN_{Phy_0}	First Input Tensor for Physical Loss Regularization
m_t	First Moment of Cost Function
ϵ	Gradient Approximation Error
g_t	Gradient of Cost Function
$CNN(X)$	Inference of X Input Tensor
V_i	Input Vector
η	Learning Rate
L_{data}	Loss Calculated from the Model Prediction
μ_i	Mean Value
α	Momentum
μ_{mov}	Moving Average Mean

β_1	Moving Average Smoothness Parameter 1
β_2	Moving Average Smoothness Parameter 2
σ_{mov}	Moving Average Standard Deviation
$\hat{V}_i$	Normalized V Vector
m_n	n^{th} Moment of a Random Value
L_{Phy}	Physical Loss
λ	Physical Regularization Rate
X	Random Value
Q_{rated}	Rated Capacity
γ	Scaling Factor
$IN_{Phy_{100}}$	Second Input Tensor for Physical Loss Regularization
v_t	Second Moment of Cost Function
β	Shifting Factor
σ_i	Standard Deviation Value
SOC_{target}	State-of-Charge Ground Truth
SOC_{pred}	State-of-Charge Prediction
SOC_{Phy_0}	State-of-Charge Predictions Inferred from IN_{Phy_0}
$SOC_{Phy_{100}}$	State-of-Charge Predictions Inferred from $IN_{Phy_{100}}$
$SOC(t)$	State-of-charge at Instant t
T	Temperature
L_{total}	Total Loss
P	Trainable Parameters of the CNN Model
$P_{updated}$	Trainable Parameters Updated After Back Propagation
B_{train}	Train Batch
BG_{train}	Train Batch Generator
k	Training Iteration Index
U	Voltage
V_{Phy0}	Voltage Vector Corresponding to IN_{Phy_0}
$V_{Phy_{100}}$	Voltage Vector Corresponding to $IN_{Phy_{100}}$

LIST OF ABBREVIATIONS

ANFIS	Adaptive Neuro-Fuzzy Inference System
ANNBM	Artificial Neural Network Method
ANN	Artificial Neural Networks
BMS	Battery Management System
CCM	Coulomb Counting Method
CFD	Computational Fluid Dynamics
CNC	Computer Numerical Control
CNN	Convolutional Neural Network
CV	Constant Voltage
DCRNN	Diffusion Convolutional Recurrent Neural Network
DD	Data driven
DNN	Deep Neural Network
DOF	Degree of Freedom
DR-RNN	Deep Residual Recurrent Neural Networks
ECM	Equivalent Circuit Model
EIM	Electrochemical Impedance Model
EIS	Electrochemical Impedance Spectroscopy
EM	Electrochemical Model
FL	Fuzzy Logic
GAN	Generative Adversarial Network
GRU	Gated Recurrent Unit
HWFET	Highway Fuel Economy Test
ISBM	Impedance Spectroscopy Based Method

LSSVM	Least Square Support Vector Machine
LSTM	Long Short-Term Memory
Li-Ion	Lithium-Ion
LiFeMnPO4	Lithium Iron Manganese Phosphate
MAE	Mean Absolute Error
MAE	Mean Average Error
MAPE	Mean Absolute Percentage Error
MBM	Model Based Method
MB	Model Based
MLP	Multi-Layer Perceptron
MSE	Mean Squared Error
N-PGCNN	Non-Physics Guided Convolutional Neural Network
NN	Neural Network
NiCd	Nicked-Cadmium
NiMH	Nickel-Metal Hydride
OCVM	Open Circuit Voltage Method
OCV	Open Circuit Voltage
OM	Other Methods
PCI	Physics Consistency Index
PF	Power Flow
PGCNN	Physics Guided Convolutional Neural Network
PGML	Physics Guided Machine Learning
PGNN	Physics Guided Neural Network
PI	Proportion Integration
PPgNN	Probabilistic Physics Guided Neural Network
PhyCNN	Physics Guided Convolutional Neural Network
RELU	Rectified Linear Unit
RGB	Red Green Blue
RMSE	Root Mean Squared Error

RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
SOC	State-of-Charge
SOH	State-of-Health
SOL	State-of-Life
SPE	Steps Per Epoch
SVM	Support Vector Machine
SVR	Support Vector Regression
UBT	Universal Battery Tester
UDSS	Urban Dynamometer Driving Schedule
2-D	2 Dimensional

LIST OF FIGURES

Figure 1.1 Classification of black box modeling methods [4] 3
Figure 1.2 Physics guided neural networks: A combination between physics and data knowledge [5] (modified) 5
Figure 1.3 Classification of physics guided data driven methods [6] (modified) 6
Figure 1.4 Classification of SOC estimation methods 11
Figure 1.5 Basic structure for model based SOC estimation 13
Figure 1.6 Commonly used algorithms for model based SOC estimation 13
Figure 1.7 The simplest neural network architecture for SOC estimation . . . 15
Figure 1.8 The proposed SOC estimation method that combines physical and data knowledge 21
Figure 2.1 The analogy between image inputs and time series inputs for 2-D CNN architecture 24
Figure 2.2 Illustration of the first 3 steps of the convolution operation 26
Figure 2.3 Local connection of the input tensor to the neurons of the convolutional layer 27
Figure 2.4 Illustration of the first 3 steps of the max pooling operation 29
Figure 2.5 The connection between the pooling layer output tensor with the fully connected layers 30
Figure 2.6 Summary of the main building blocks of CNN architecture 31
Figure 2.7 Batch normalization input tensor 32
Figure 3.1 DIGATRON MCT-ME cell tester [79] 38
Figure 3.2 Battery test procedure 39
Figure 3.3 A 5000-sample random chunk from the train data set 41
Figure 3.4 A 5000-sample random chunk from the validation data set 42
Figure 3.5 A 5000-sample random chunk from the test data set 43
Figure 3.6 Visualization of the proposed CNN model structure 46
Figure 3.7 Train algorithm for PGCNN and N-PGCNN models 47
Figure 3.8 Combined loss function implementation in Python programming language 51
Figure 3.9 Visualization of the combined loss function in TensorBoard environment 52

Figure 3.10 IN_{Phy_0} and $IN_{Phy_{100}}$ input tensors used for physical loss regularization 53
Figure 3.11 SOC predictions and absolute error of N-PGNN model ($\lambda = 0$, time window = 100) with random test data 55
Figure 3.12 SOC predictions and absolute error of PGNN model ($\lambda = 0.4$, time window = 100) with random test data 56
Figure 3.13 SOC predictions and absolute error of N-PGNN model ($\lambda = 0$, time window = 25) with random test data 56
Figure 3.14 SOC predictions and absolute error of PGNN model ($\lambda = 0.1$, time window = 25) with random test data 57

LIST OF TABLES

Table 1.1 Literature examples for model based SOC estimation algorithms [26] 14
Table 1.2 Comparison of the most commonly used SOC estimation methods . 19
Table 2.1 The most commonly used loss functions in regression and classification problems 34
Table 3.1 NCR18650PF Lithium-ion battery specifications 38
Table 3.2 Test equipment specifications 38
Table 3.3 Drive cycles used for battery discharging 40
Table 3.4 Sizes of train, validation, and test data sets 40
Table 3.5 Structure of the CNN model 45
Table 3.6 Symbols and abbreviations used in algorithm 1 48
Table 3.7 N-PGCNN and PGCNN models MAPE test results 58
Table 3.8 N-PGCNN and PGCNN models PCI test results 59

ABSTRACT

Physics Guided Neural Network Based State-of-Charge Estimator for Lithium-Ion Batteries

Fedi SALHI

Department of Mechatronics Engineering
Master of Science Thesis

Supervisor: Prof. Dr. Erhan AKDOĞAN

In light of the growing demands of reliable substitutions to vulnerable conventional energy sources, electrical energy manifests as a potent surrogate that has started to prevail in a variety of industries and applications. The effectiveness and usability of this energy is highly associated with the development of sophisticated storing systems, a topic that has been increasingly attracting the attention of scientists and engineers.

Among all technologies, batteries are the most preponderant solutions for electrical energy storage, in the form of electro-chemical potential energy. Particularly, Lithium-ion batteries, being characterized by a high energy density and a low self-discharge rate, are considered one of the most effective solutions for this purpose. However, Lithium-ion cells necessitate more complex management systems, in order to ensure the operation safety and maximize the cycle life. The state-of-charge represents a crucial parameter estimated by Battery Management Systems used in energy and power calculations. Hence, an accurate state-of-charge estimation is indispensable to prolonging the battery life and preventing hazardous usage conditions.

Considerable effort has been put into the development and enhancement of state-of-charge estimation techniques. Model based methods are the most dominant estimators in industrial applications. Data driven algorithms, on the other hand, are gaining more and more attention among researchers due to the fast-paced development of hardware platforms which is unlocking the potential of these algorithms that require a relatively high computational power. Nevertheless, black box modeling is usually confined to the data scope, struggles under unseen conditions

and it is difficult to deliver a general model. Additionally, data driven estimators are prone to be inconsistent with physics laws, which make their deployment in real life applications more risky.

In this work, we leverage the latest advancements in machine learning to develop a novel technique to train a Physics Guided Convolutional Neural Network based state-of-charge estimator for Lithium-ion batteries. In a nutshell, the methodology consists of introducing physical constraints during the train phase by designing a cost function that combines both physical and data losses, based on an energy approach. The proposed algorithm is validated using open data of Panasonic 18650PF Lithium-ion battery [1], by training and testing physics guided and non-physics guided convolutional neural network models, referred to as PGCNN and N-PGCNN models, respectively.

Results showed that the proposed methodology outperformed the classic paradigm in both accuracy and physics consistency based on Mean Absolute Percentage Error and Physics Consistency Index metrics, for large and small time windows, which validates the proposed approach as a surrogate method for neural network based state-of-charge estimation.

Keywords: Physics guided neural network, state-of-charge estimation, Lithium-Ion batteries, data driven modeling

ÖZET

Lithium-Ion Piller İçin Fizik Destekli Sinir Ağı Tabanlı Şarj Durumu Tahmincisi

Fedi SALHI

Mekatronik Mühendisliği Anabilim Dalı
Yüksek Lisans Tezi

Danışman: Prof. Dr. Erhan AKDOĞAN

Konvansiyonel enerji kaynaklarının güvenilir yedeklerine artan talepleri göz önünde bulundurunca, çeşitli endüstri ve uygulamalarda hızlıca yaygınlaşan elektrik enerjisi, bir alternatif olarak kendini göstermektedir. Elektrik enerjisinin verimliliği ve kullanılabilirliği, bilim insanları ve mühendislerin ilgisini artarak çeken gelişmiş depolama sistemlerinin geliştirilmesi ile son derece ilişkilidir

Tüm teknolojiler arasında, enerjiyi elektro-kimyasal potansiyel enerji olarak depolayan piller, elektrik enerjisi depolamak için en etkili çözümdür. Özellikle, yüksek enerji yoğunluğu ve düşük özboşalım oranları özelliklerine sahip Lithium-ion (Li-ion) pilleri, bu görev için en etkili çözüm olarak görülmektedir. Ancak, Li-ion pillerinin çevrim ömrü uzatmak ve sistem güvenliği sağlamak için diğer pil tiplerine göre daha karışık yönetim sistemlerine ihtiyaç duymaktadır. Batarya Yönetim Sistemi'nin (BMS) tarafından tahmin edilen ve enerji ve güç hesaplamalarında kullanılan *şarj durumu (state-of-charge, SOC)* önemli bir parametredir. Doğru bir şarj durumu tahmini pil ömrünü uzatmak ve tehlikeli kullanım durumlarını önlemek için son derecede önemlidir.

SOC tahmin tekniklerinin geliştirilmesi ve iyileştirilmesi için önemli bir çaba sarf edilmiştir. Endüstriyel kullanımlarda model tabanlı metodlar en çok kullanılan olarak görülmektedir. Bunun yanı sıra, yüksek sayısal güç gerektiren algoritmaların tüm potansiyeline ulaşımını sağlayan donanım platformlarının hızlıca gelişmesiyle, veri güdümlü algoritmalar gün geçtikçe daha çok dikkat çekmektedir, özellikle araştırmacıların tarafından. Ancak, kara kutu modelleme yöntemleri genellikle veri

kapsamı ile sınırlıdır, öngörülemez durumlarda zorlanır ve bu metotları kullanarak genel bir modelin geliştirilmesi oldukça zordur. Üstelik, veri güdümlü tahminlerin fizik yasalarına göre tutarsiz olmaya yatkın olmaları gerçek uygulamalarda kullanılması daha riskli yapmaktadır.

Bu çalışmada, Li-ion pilleri için yeni bir fizik destekli yapay sinir ağı tabanlı şarj durumu tahmin algoritması geliştirmek amacıyla, makine öğrenmesindeki en son gelişmelerden yararlanılmıştır. Bu yöntemde, kısaca, bir enerji yaklaşımına dayanarak, eğitim aşamasında fiziksel kısıtlamalar sunmayı sağlayan, fiziksel ve veri hataları birleştiren bir maliyet fonksiyonu tasarlanmıştır. Önerilen algoritma, fizik güdümlü (PGCNN) ve fizik güdümlü olmayan (N-PGCNN) evrişimsel yapay sinir ağları, Panasonic 18650PF Li-ion pilinin açık verileri [1] kullanılarak, eğiterek ve test ederek doğrulanmıştır.

Sonuçlar, önerilen metodolojinin, ortalama mutlak yüzde hata (MAPE) ve fizik tutarlılık indeksi (PCI) metrikleri kullanılarak, büyük ve küçük zaman pencereleri için, hem doğruluk hem de fizik tutarlılığı açısından klasik paradigmadan daha iyi olduğunu göstermiştir. Bu, önerilen yaklaşımı yapay sinir ağları tabanlı şarj durumu tahmini için yeni bir yöntem olarak doğrular.

Anahtar Kelimeler: Fizik gündümlü yapay sinir ağı, şarj durumu tahmini, Lithium-Ion pil, veri tabanlı modelleme

1
INTRODUCTION

Batteries are powering every aspect of our modern life, from cell phones and computers, to electrical vehicles and drones. Battery technologies have been evolving over the years to cater the critical requirements of different industries such as military applications, where the battery quality is a main concern due to the high energy consumption of algorithms running the devices used in these applications [2].

Batteries are also becoming a major player in the renewable energy revolution with the creation of a new generation of reliable, efficient, and sustainable hybrid renewable power plants that are able to overcome intermittent wind and solar energy. In fact, the significant advancement in battery technologies in recent years has offered more flexibility in harnessing and storing renewable energy, which makes it more and more cost-efficient. Grid-storage is also another example where batteries are being actively used. These systems store excess generated energy for later use during rush hours. In addition, batteries are used for grid backup in power distribution stations to prevent power outage during maintenance or in case of system issues. Among all technologies, Lithium-Ion (Li-Ion) batteries are considered the most effective, thanks to their high energy density, low self-discharge rate, and long lifetime.

The notable presence of batteries in several areas required the development of sophisticated electronic systems and algorithms to ensure a faultless functioning of these systems. A Battery Management System (BMS) is an electronic device that provides a safe operation of rechargeable batteries. The primary functions of a BMS are the protection of both the application user and the battery pack itself by monitoring the battery state, detecting unsafe conditions, and preventing undesired outputs and hazardous conditions. Another important function of BMS is the optimization of the battery performance and service life by specifying the most convenient energy and power limits that maximize the battery's life. In order to fulfill these requirements, BMS perform some basic functionalities that can be classified into these categories, as stated in [3]:

(a) Measurements and high-voltage control
A BMS measures each cell's voltage, temperature, and the battery current.It also controls the contactors to connect or isolate the application load, and the thermal management systems.

(b) Protection
It contains electronics and algorithms to protect the battery user and battery-pack against hazardous conditions including overcharge, overdischarge, overcurrent, short circuits and high internal temperatures.

(c) Application interface
It communicates with the application and provides it with the available energy, power limits, and other information related to the battery pack. It also records unusual errors and events in a permanent memory.

(d) Performance management
It estimates the state-of-charge (SOC) of each cell, conducts energy and power calculations and balances cells in the battery pack. The method discussed in this thesis is related to the SOC estimation function, that must be fulfilled by every BMS.

(e) Diagnostics
It estimates the state-of-health (SOH) and the state-of-life (SOL) of the battery pack.

BMSs are often required only by systems with a specific level of complexity. Simple battery powered systems may not necessitate a sophisticated BMS. However, all applications using Li-ion batteries require a BMS to maintain a margin of safety and prevent overcharging and other risky conditions. BMSs are crucial for many battery powered applications but usually need a certain computational power that depends on the algorithms being implemented withing the management system. Thus, a reasonable time and effort should be spent on choosing and optimizing the best solution according to the system requirements.

The work represented in this thesis belongs yo the *Performance management* function in which a novel physics guided convolutional neural network (PGCNN) based Li-ion SOC estimation method is proposed. In a nutshell, the methodology consists of introducing physics constraints during the train phase by designing a hybrid cost function that combines both physical and data losses, based on an energy approach. Although the proposed method is demonstrated using a convolutional neural network (CNN) architecture and validated with open data of Panasonic 18650PF Li-ion battery,

the approach is also applicable for other types of batteries and different neural network (NN) architectures.

1.1 Literature Review

1.1.1 Physics Guided Neural Networks

1.1.1.1 Black Box Modeling Paradigm

Artificial Neural Networks (ANN) have been attracting the attention of scientific researchers for several years especially with the continuous improvement of hardware capabilities that allowed the design and implementation of more sophisticated models that demand a significant computational power, particularly, for the training phase.
Black box (BB) modelling is a method that uses empirical data to extract the input-output relationship for a specific system. This paradigm can be implemented using several data science models. According to [4], we distinguish two categories of BB models: linear and nonlinear forms, as shown in Figure 1.1.

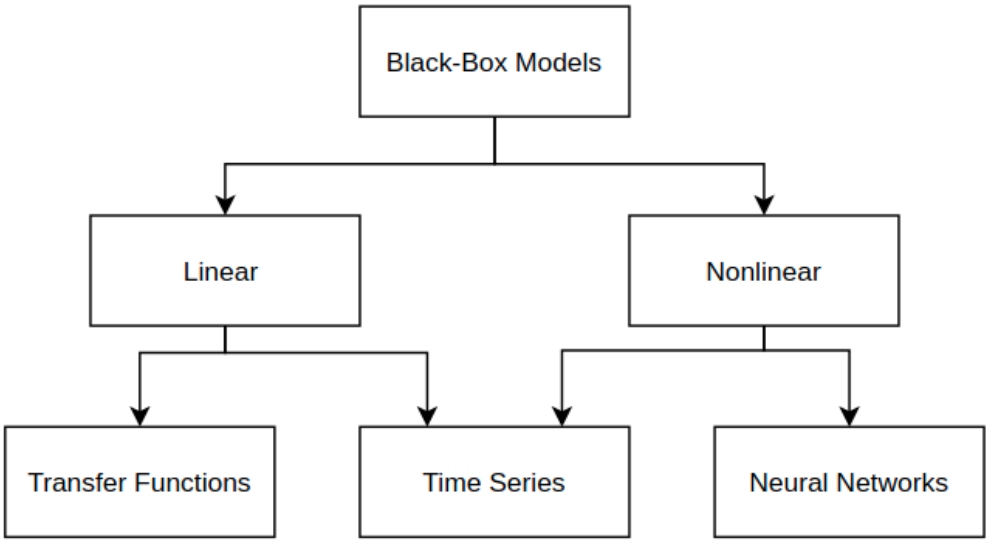


Figure 1.1 Classification of black box modeling methods [4]

Transfer functions and time series are the most commonly used linear BB modelling methods. The parameters of such models can be identified using multiple techniques, and in particular least squares algorithms. On the other hand, NNs are the most preponderant forms of nonlinear BB modeling, alongside time series.

1.1.1.2 Consolidation of Black Box Paradigm with the Integration of Physics Laws into the Modeling Process

Although this modeling paradigm has been achieving a huge success in solving real world prediction and classification problems, it does not come without limitations. A major disadvantage of data driven (DD) models, that are not aware of the scientific principles driving the system being modeled because of their dependency to the available labeled data, which makes the model only usable for specific problems under certain conditions and do not generalize beyond the scope of the used data. Additionally, black-box models can deliver inconsistent results with the well-verified physics laws [5].

Thus, BB lack of physics guidance can limits the reliability and robustness of the model, the problem that started to be solved by introducing a new algorithm for NNs that involves physics laws into the learning process. These kind of models are often referred to as physics guided neural networks (PGNN) as coined by Arka Daw et al. in [5] for the first time, in which they presented a novel framework that can be used in scientific problems combining both physics based models with neural networks. The general idea is to introduce a new method that utilizes scientific theory with empirical data, to produce a more reliable model consistent with physics laws, as illustrated in Figure 1.2. This method was validated in the same paper by implementing and testing an application of lake temperature estimation.

1.1.1.3 Objectives of Physics Guided Data Driven Modeling

Using physics guided DD modeling to solve science and engineering problems has been a major area of interest among researchers, especially for the last few years. Rui et. al provided a thorough review of state-of-the-art physics guided deep learning methods [6]. The study adduced the main motivations behind the integration of physics laws into DD models e.g machine learning architectures. One of the reasons is to find alternative models to replace or complement the classic methods of simulation which are generally computationally expensive and need a high level of expertise. The traditional simulation methodologies are mainly based on solving differential equations that describe physical laws governing the system being simulated. For instance, Han et. al have applied physics guided deep learning to Computational Fluid Dynamics (CFD) simulations [6]. The work presented by X. I. A. Yang et. al that consists of using PGNN for large-eddy-simulation wall modeling [7], is also another example of using this paradigm to accelerate data simulation.

There is no doubt that the consistency with science is one crucial objective that PGNNs aim at. Irregardless of all evaluating parameters, a model that does not provide

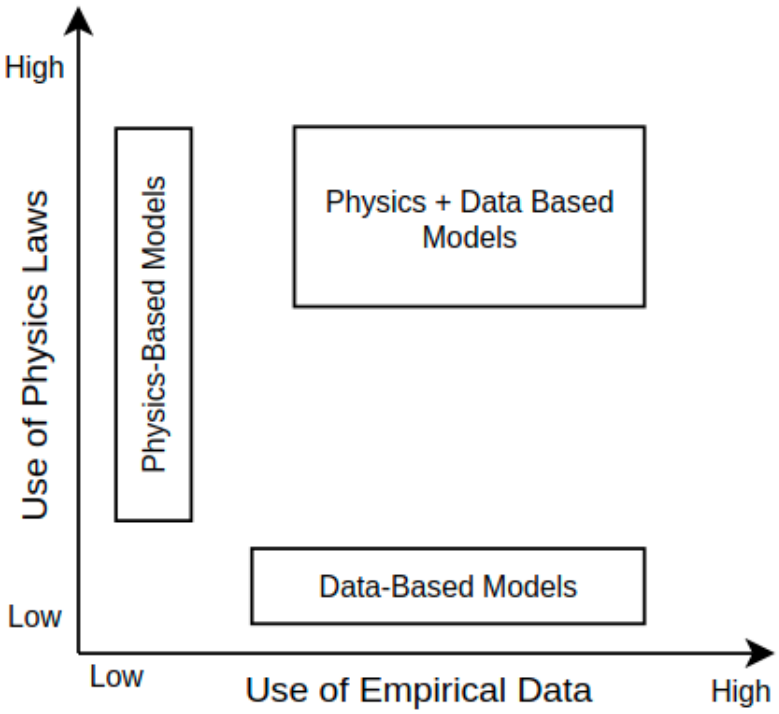


Figure 1.2 Physics guided neural networks: A combination between physics and data knowledge [5] (modified)

physics/science consistent predictions is considered unreliable and can not be used in critical contexts. Hence, the inclusion of physical principles in the BB architecture will prevent the violation of physics logic and will generate a more solid and trusty models. This objective was illustrated in several works including [8] in which the researchers have added physical constraints to CNN architecture for incompressible turbulent flow modeling. Jin-Long et. al [9] included physics consistency to Generative Adversarial Networks (GANs) for chaotic dynamical systems modeling by enforcing statistical constraints to the model.

DD models are generally confined to the information extracted from the data under certain conditions and do not exhibit generic solutions. In other words, BB models often struggle with unseen scenarios that are not represented by the available data. This limitation can be minimized by using larger data during the training phase, that encapsulates all possible situations. However, this may not be feasible or may require strenuous effort to realize. Guiding DD models, paricularly NN, with physics knowledge will improve their generalizability. This can be explained by the invariability of physics laws for different applications and domains, which offers a guidance in features extraction. Namely, Rui et. al [10] have proposed a method to improve the generalization of CNNs by incorporating different symmetries into the model. The suggested methodology has been validated for a variety of physical

dynamics including Rayleigh Bénard convection and ocean currents and temperatures.

On the whole, assisting DD models with physics laws aims at offering a more generic architectures usable under different conditions and consistent with physics laws to reduce the complexity of the conventional numeric methods used in different engineering problems such as simulation. Bear in mind that physics guided DD models also have other advantages, they can be used for different objectives, and are not necessarily limited to three aims previously discussed, which we consider the most important.

1.1.1.4 Different Methods of Integrating Physics Knowledge Into Data Driven Models

Significant effort has been put by researchers on figuring out the best way to integrate physics knowledge into DD models. As a result, several methods has been proposed and validated for a wide variety of engineering and science problems. [6] has proposed a categorization of physics guided deep learning methods. However, we consider that this classification is also valid for all BB modeling methods as well. The different approaches of combining physics principles with data extracted knowledge are shown in Figure 1.3. It is worth mentioning that the learning algorithm proposed in this

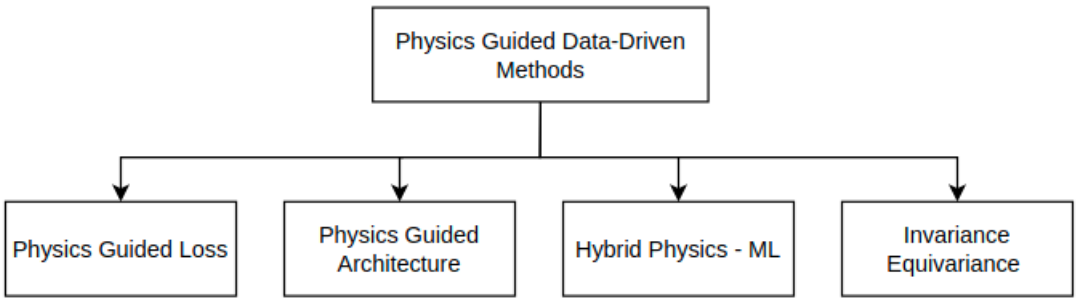


Figure 1.3 Classification of physics guided data driven methods [6] (modified)

work fits into the first category, physics guided loss. The chief idea of all architectures included in this category is to adjust the loss function during the training phase, based on the level of physics consistency in the model predictions. This demands the extension of the total cost function by adding a physical loss that should be zero for physics consistent predictions, but greater than zero for physics inconsistent predictions. This will regulate the total loss function depending on the model outputs, whether they are physics consistent or not. We will go through these categories in the following sub-sections.

Physics Guided Loss Models

This is a straightforward and uncomplicated approach to incorporate physics knowledge into DD models. The major idea is to design a cost function consisting of two part: Data loss and physical loss. The optimization of a well designed cost function during the training process will necessarily decrease the gap between predictions and ground truth, while increasing the physics consistency of the model according to the chosen physics constraints.

Physics Guided Architecture

This method consists of adding new neurons or layers to the original non-guided model. The flexibility and modularity nature of NNs facilitates the extension of models to include physics constraints into the learning process. Turbulent-Flow Net [11] is an example of architectures that fit into this category.

Hybrid Physics Machine Learning Models

This method aims to combine physics models with machine learning architectures in some possible ways. For instance, DeepGLEAM [12] uses a combination of pure physical simulation model called GLEAM with a Diffusion Convolutional Recurrent Neural Network (DCRNN) [13] to predict COVID-19 death count.

Invariant and Equivalent Models

Noether's theorem [14], that relates conserved quantities with symmetries (e.g. spatial translations, rotation invariance, and time translations are related to linear momentum, angular momentum, and energy conservation, respectively), has inspired researchers, namely Rui et. al [10], to use symmetric deep learning models to conserve the associated quantities and subsequently the consistency of predictions with physics principles.

1.1.1.5 Examples of Physics Guided Data Driven Models

This paradigm of combining information extracted from data with well-defined physics laws have been used then in several researches in an effort to solve a variety of engineering problems. This section includes recent studies that have employed, in different ways, a physical guidance to get more satisfactory results comparing to purely DD models.

Particularly, [6] has tried to tackle the issue of tool wear prediction in manufacturing process by suggesting a physics guided Gated Recurrent Unit (GRU), (a type of recurrent neural networks) model to predict the tool wear in high-speed CNC machine. In the training phase, the proposed model has taken into consideration the physical knowledge that the tool wear degree should increase when the tool cutting number increases. Results showed that the model outperformed bi-directional GRU, CNN,

Long Short-Term Memory (LSTM), and Support Vector Regression (SVR) in both accuracy, measured with Root Mean Squared Error (RMSE) and Mean Average Error (MAE), and physical consistency.

The idea of utilizing physics to enhance DD models has been also extended to other types of NNs. Ruiyang et al. [15] have used Physics Guided Convolutional Neural Network (PhyCNN) to model the seismic response of buildings. The proposed model uses the ground acceleration and predicts the state space variables including displacement, velocity, and restoring force. The CNN architecture has been physically guided by including physical loss, calculated from modeling the dependency between output features, in the total loss function which is used to update the model parameters i.e. weights and biases. The performance of this novel model has been demonstrated using numerical and experimental examples. Results showed that the proposed learning algorithm delivered a model capable of predicting structures response without the need of physics based methods and outperformed the conventional non-guided NNs.

Power Flow (PF) analysis is also another engineering area where researchers have tried to leverage the latest advancement in machine learning methods by using PGNNs. Xinyue et al. [16] has proposed a PGNN that enables PF analysis under non-deterministic system parameters. In this work, researchers have proposed three kind of PF solvers characterized by different physics knowledge of power systems. This work suggests that the proposed models are not confined to PF, but they can be extended to solve other power system problems including probabilistic load flow analysis and security constraints, thanks to their high speed in PF calculations. The same research has mentioned that the models should be periodically updated in order to adapt with the evolving nature of power grids.

The integration of physics knowledge into DD machine learning models manifested in different forms in the literature, especially in recent academic works. For instance, Suraj et. al [17] have introduced a modular physics guided machine learning framework (PGML) that differs from the other models previously presented, in the way that physics laws do not directly affect the total loss function, i.e. the loss function does not include the physical loss, but it is solely based on the DD loss. This new approach interjects the predictions of a physical model alongside other parameters (needed by the physical model) in the an n^{th} hidden layer as new inputs. In other words, the latent variables of the n^{th} hidden layer are augmented by the physical model predictions and parameters. The suggested methods has been applied to a computational fluid dynamics problem, in which the proposed model that predicts the lift force, was augmented by physical parameters of the flow, including Reynolds

number and angle of attack, and the prediction from the Hess-Smith panel method i.e. lift and pressure drag coefficients. The researches concluded that the inclusion of physical information into machine learning methods improves the model accuracy and decreases the uncertainty.

Another method of guiding NNs using physics laws for structural dynamics simulation, proposed by Yang et. al [18], consists of combining two modules: physics based layers and DD layers. The first layer includes residual blocks that indicate the consistency of predictions with the physical formulations by calculating the difference between the inputs and the outputs of the physics equations. During the training phase, the residual function is minimized using Deep Residual Recurrent Neural Networks (DR-RNN) [19] architecture to increase the conformity of the model predictions with physics laws. The output of the physical model, which is the system state vector, may not be observable. For this reason, an Multi-layer Perceptron (MLP) based second layer is used to map the relationship between the observable output and the hidden system state. The methodology has been validated using a variety of structural systems such as three-degree-of-freedom (DOF) and a nonlinear spring-mass-damper dynamics. The different case studies showed that the proposed model has several advantages comparing to purely DD machine learning methods including a high generalization ability, reduced training cost, as well as the capability of modeling systems with partially known physics.

Embedding physics knowledge into classical machine learning methods have been also proven to be effective in overcoming the limitations of BB models for fatigue analysis. Jie et. al [20] have proposed a Probabilistic Physics Guided Neural Network (PPgNN) for fatigue S-N curve estimation. S-N curve, also referred to as Wöhler curve, is the method that is often used to describe the fatigue of materials. This curve describes the relationship between stress amplitude and the number of cycles to failure [21]. Typically, the curvature of the fatigue curve should decrease as the stress decreases i.e. the life of materials should be longer when applying lower levels of stress, and shows asymptotic behavior near the fatigue limit or very long life for materials without a specific fatigue limit [22]. Additionally, the standard deviation of the fatigue life usually increases with lower strains. These two considerations have been used to integrate physical knowledge into the suggested model by imposing relevant constraints on its parameters including weights, biases, or both. Different fatigue data from metallic and composite materials have been used to validate the designed model. Results showed that the PPgNN architecture was able to accurately predict the mean and the standard deviation of materials fatigue. This study inferred that a flexible inclusion of physics laws is crucial for DD fatigue analysis.

1.1.2 State-of-the-art in State of Charge Estimation For Lithium-Ion Batteries

1.1.2.1 Definition of State-of-Charge

State-of-charge, often referred to as SOC, is a crucial parameter that signifies the available charge of a battery, [23] and simply defined by the ratio of the available electric charge $Q_{available}$ to the maximum amount of charge Q_{rated} that can be stored in the battery [24], as shown in Equation 1.1

$$SOC(t) = \frac{Q_{available}(t)}{Q_{rated}} \tag{1.1}$$

The SOC is not a measurable physical quantity and despite its uncomplicated definition, it is challenging to provide an accurate estimation of this parameter due to the non-linear nature of the battery characteristics and the complexity of the electro-chemical reactions [23]. The estimation is typically based on voltage, current, and internal temperature measurements [25]. An accurate SOC estimation is crucial for the remaining energy and available power calculations from one side, and maximizing the life of the battery from another side. Hence, tremendous work has been done over the years by researchers to provide more convenient solutions for SOC estimation. Over the years, many methods have been proposed to deliver a valid SOC estimation. In this section we will go through these different methods, with a focus of DD methods since they are more relevant to this thesis. Dickson et. al [26] have reviewed the recent SOC estimation methods with a focus on model based and DD approaches. These methods are summarized in Figure 1.4. Note that there are also other hybrid methods that combine estimators from different categories, which are not mentioned in the Figure. We will go through all of these approaches in next sections.

1.1.2.2 Lithium-Ion Batteries

Lii-ion batteries offer several advantages compared to other electrochemical batteries. Li-ion cells have a higher nominal voltage, typically about 3.7 V. Other types of cells such as Nickel-Metal Hydride (NiMH) and Nicked-Cadmium (NiCd) operate at lower voltages of around 1.2 V, which means that lower number of Li-ion cells is required for applications that need a high supply voltage [3]. Li-ion cells also have a higher energy density and specific energy, which means that for a specific capacity they are smaller and lighter in comparison with other batteries using different chemistries. Li-ion cells are characterized by a lower self-discharge rate than other types of cells. NiMH and NiCd cells are prone to lose up to 5% of their charge daily when they are not connected to any load. In contrast, Li-ion are capable to maintain most of their charge for months while they are not being used [27]. However, Li-ion cells require more

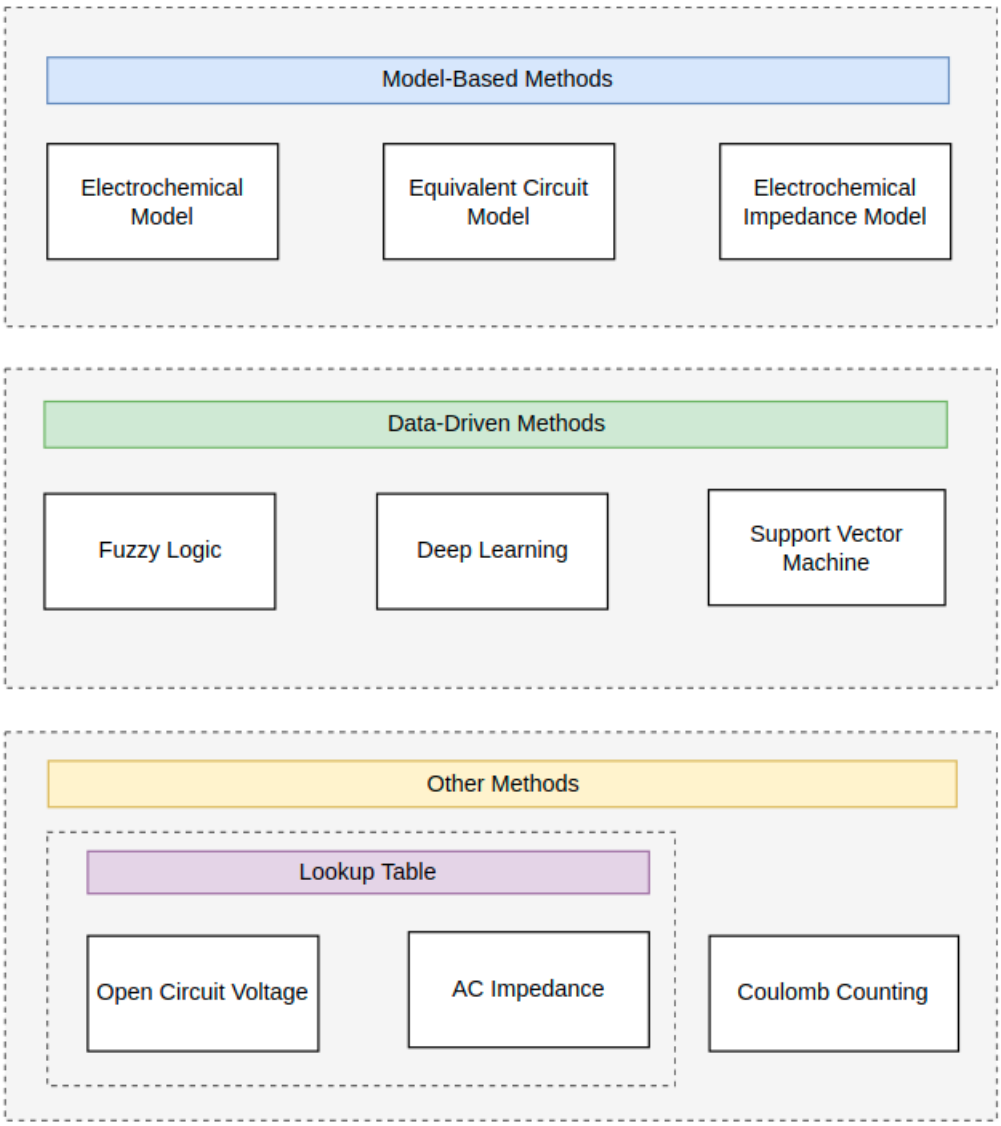


Figure 1.4 Classification of SOC estimation methods

sophisticated management systems and have more complex manufacturing processes, which increases their cost compared to other solutions. In this work, data related to this type of batteries is used to validate the proposed method.

1.1.2.3 Model Based State-of-Charge Estimation

This approach, also known as white box modeling, is considered to be a valid classic paradigm to solve engineering problems [28] that requires a set of physical and mathematical rules that describe the system behaviour or the process to be modeled [29].

In the literature, we distinguish different methodologies used for model based SOC estimation. These methods generally use either the electrical or chemical behaviours of the cell or a combination of both. They are generally based on the following models:

(a) Electrochemical Model (EM)
EM takes into consideration the porous electrode theory, explained in details in [30].

(b) Equivalent Circuit Model (ECM)
ECM is based on the electrical properties of the battery cell, represented by the combination of various electrical elements. There are several electrical models that has been proposed in the literature including the Thevenin model, the Rint model, and the RC model [31]. Obviously, the ECM requires to be parameterized in order to include the cell behaviour that changes depending on the battery type and chemicals of the cell [32].

(c) Electrochemical Impedance Mode (EIM)
EIM is similar to the EC model, but with inclusion of Z_{arc} and Warburg element [33].

The model based SOC estimation approach often involves state estimation algorithms and adaptive filters into the estimation process. Figure 1.5 shows a basic way to combine battery models with estimators, where T is internal temperature of the battery, I is the current, U_{model} is the battery voltage computed from the model, U is the measured battery voltage, and SOC is the estimated state-of-charge.

The most commonly used estimators are summarized in Figure 1.6. Table 1.1 represents these methods with some examples from the literature.

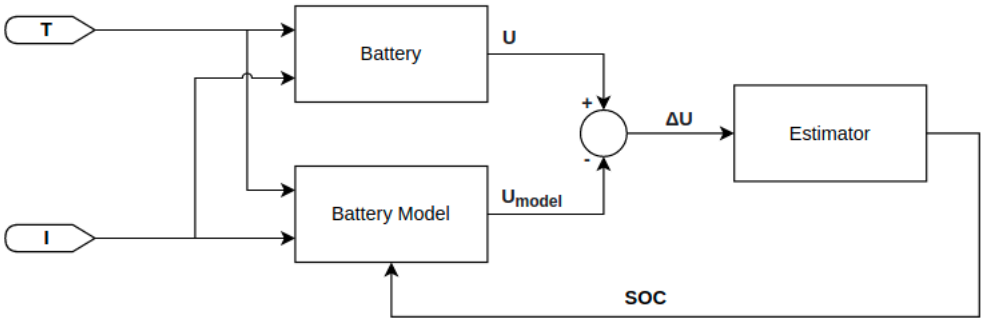


Figure 1.5 Basic structure for model based SOC estimation

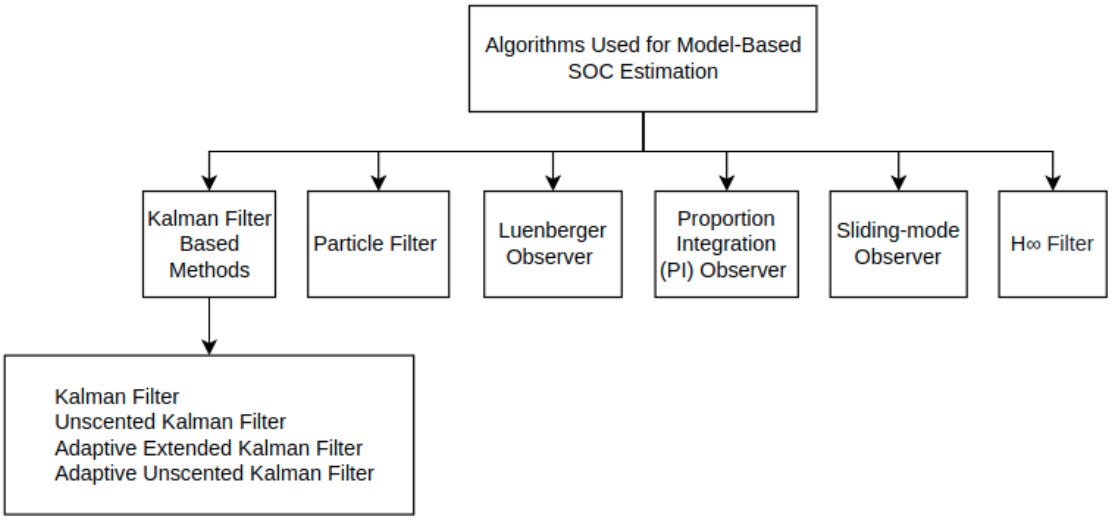


Figure 1.6 Commonly used algorithms for model based SOC estimation

Table 1.1 Literature examples for model based SOC estimation algorithms [26]

Estimation Algorithms	Literature Examples
Unscented Kalman Filter	[24]
Adaptive Extended Kalman Filter	[34]
Kalman Filter Considering Model Error	[35]
Adaptive Unscented Kalman Filter	[36]
Particle Filter	[37], [38], [39]
$H\infty$ Filter	[40], [41], [42]
Luenberger Observer	[43], [44]
Proportion Integration (PI) Observer	[45], [46]
Sliding-mode Observer	[47], [48]

1.1.2.4 Data Driven SOC Estimation

DD SOC estimation paradigms avoid the necessity for a deep understanding of the internals to model a specific system. This methodology does not require domain expertise since the formulation of the mathematical and physical laws governing the system behaviour are often not involved in the modeling process. This technique is accompanied with simplifications of the modeling procedure, and reduction of the complexity usually associated with model based SOC estimation that often necessitate the construction and the verification of robust set of rules describing the system.

The simplicity of DD modeling consists in mapping the inputs (typically the measured internal temperature, current, and voltage) with the output (SOC) using a relationship that does not involve battery models and estimators or observers. A variety of DD models have been used for SOC estimation. In this section we will go through the most common and recent structures that dominate the literature for DD SOC estimation, including NNs and its varieties, Support Vector Machine (SVM), and Fuzzy Logic (FL).

Neural Networks and Variants Based Methods:
The simplest form of NN for SOC estimation is shown in Figure 1.7. This is the general architecture of NN based SOC estimation [23]. The model is composed of 3 layers, the first of which is the input layer consisting of 3 neurons for measured battery internal temperature, current and voltage, a hidden layer, and an output layer for SOC prediction. Note that the inputs represented in 1.7 are not always the same, they can vary depending on the modeling approach.

This basic structure has been developed and enhanced by researchers to deliver more accurate predictions. For instance, Shijie et. al [49] split the classic 1-vector input structure into three vectors corresponding to three modes: Idle mode, charging mode, and discharging mode. These modes are defined based on the measured current value. During the training phase, only one of these vectors is activated at the same time (e.g.

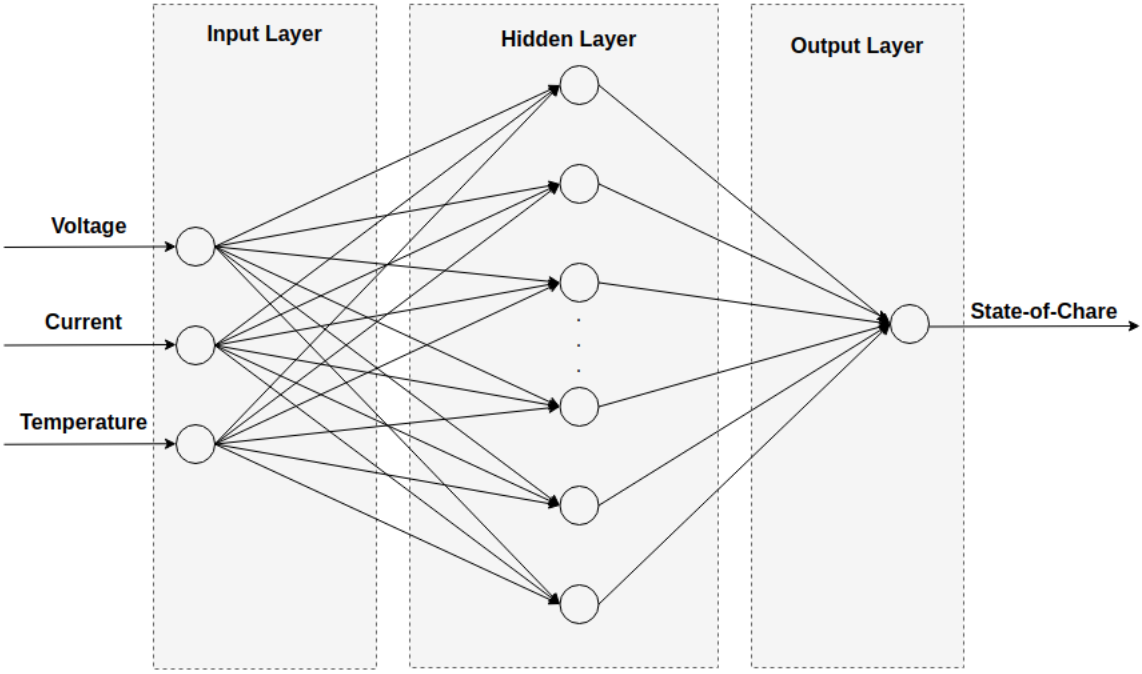


Figure 1.7 The simplest neural network architecture for SOC estimation

if the battery is in charging mode, the two vectors corresponding to the idle and the discharging modes are set to zero). The input vector consists of different features extracted from measurement data. The proposed model has been trained using US06 drive cycle and validated with a pulse test. Results showed that the suggested model outperformed the classical NN structure by achieving a 3.8% average error in SOC predictions.

The feature extraction step is often required only for structures with one hidden layer. Deep models, constructed with an aggregation of hidden layers, naturally have the ability to extract features from raw measurements. Although the term "deep" is not a unanimous term among researchers, we can consider the models with more that two hidden layers as Deep Neural Networks (DNNs). Ephren et. al [50] used a Deep FeedForward Neural Network to directly map the battery measurements to the SOC. The model was trained and tested for different ambient temperatures between $-20°C$ and $25°C$. The proposed model achieved a MAE lower than 2.17%.

Besides NNs, other deep learning architectures have also taken place in the literature among the DD Li-ion battery SOC estimation models. Fangfang et. al [51] have used LSTM structure to model Li-ion battery dynamics and predict the SOC from current, voltage, and temperature measurements. The model was trained using different drive cycles data and evaluated according to various criteria including accuracy, computation time, and behaviour under unknown initial states. Results showed

that the proposed model delivered better prediction accuracy (2% and 1% accuracy with RMS and MAE, respectively) comparing to unscented Kalman filter, with a fast convergence to the true SOC, in case of inaccurate initial SOC.

GRU is also another type of NNs with the ability to catch the time dependency of model predictions, which can increase the output accuracy for time series inputs. In [52], Bin et. al have used a GRU model for SOC estimation. An Ensemble Optimizer, based on Nadam and AdaMAx algorithms, was used to find the optimal parameters that minimize the cost function. The suggested model provided a reduced training duration with increased prediction accuracy.

Support Vector Machine Based Methods:
SVM is a machine learning algorithm used for regression and classification problems. This algorithm separates data belonging to different classes using a hyperplane in multi-dimensional space. The separation is optimal when the distance between the hyperplane and the closest data point from any class is maximized [53]. In [54], J.N et al. have proposed an optimized SVM algorithm for SOC estimation based on double search optimization process. The model was validated using ADVISOR simulation data. The tested model generated more accurate predictions compared to a NN based baseline predictor in spite of its simplicity. In [55], Juan et al. have also used the SVM approach to estimate the SOC of a lithium iron manganese phosphate ($LiFeMnPO_4$) battery cell, based on current, voltage, and temperature measurements. Results showed that the proposed model maintained a high accuracy for constant or variable current charge/discharge profile. On the other hand, other works have suggested the combination of SVM with Kalman filter to improve the model reliability. For instance, Jinhao et al. [56] have proposed an algorithm for SOC estimation based on the combination of an adaptive Unscented Kalman Filter with Least Square Support Vector Machine (LSSVM) model. According to the authors, an accurate battery model with limited training data has been built thanks to the applied moving window method. The effectiveness of this technique has been validated by both experimental and simulation data.

Fuzzy Logic Based Methods:
Contrary to Boolean logic, FL is an approach that adds other degrees between false and true. According to this logic, statements have a concept of partial truth. The degree of truth is represented by a value between 0 and 1 [57]. FL integrates different steps into the decision process. These steps, explained in [58], mainly includes fuzzification, inference, and defuzzification.

Although FL does not dominate the DD SOC estimation methods, the literature shows

that some effort has been put into the development of this technique for Lithium cells SOC evaluation. [59] is an example of work where the authors have suggested an SOC estimation method based on fuzzy least square support vector machine for lithium iron phosphate (LiFePO4) battery. Meftahu et al. [60] have designed an SOC monitoring device for golf cart, to investigate the effect of temperature on SOC estimation. The system was based on a FL algorithm that uses the current, voltage, and temperature measurements of charge and discharge cycles. The researchers inferred that the temperature affects the SOC value from 10 to 20%.

Other varieties of FL have been also introduced by the literature for SOC estimation. Adaptive Neuro-Fuzzy Inference System (ANFIS) is combination of NNs with FL. Zahid et al. [61] achieved an SOC estimation error below 1% using this architecture with a model integrating six inputs including actual power loss, available and requested power, cooling air temperature, battery thermal factor temperature and current.

1.1.2.5 Other Methods

Besides model based and DD paradigms, the literature shows that other methods have also been used for Li-ion SOC estimation. These methods are generally characterized by their simplicity, but often not used in applications requiring an accurate SOC estimation.

Lookup Table Methods:
Generally, lookup table methods are preferred for systems with limited resources since they do not require a high computational power. These techniques are based on either Open Circuit Voltage (OCV) or Electrochemical Impedance Spectroscopy (EIS) measurements. EIS is more difficult to measure online and differs according to the battery type and the working conditions [62], which explains the relatively limited usage of these methods compared to the OCV methods.

The OCV SOC estimation method uses the OCV-SOC relationship for the estimation process. However, an accurate determination of this relationship is often impossible, due to the difference in operation conditions in real life applications [63]. Hence, a significant effort has been spent into the enhancement of OCV-SOC methods taking into considerations other conditions including temperature and cell aging. For instance, In [64], the researchers have proposed an aging correction technique that tracks and corrects the OCV. The method has been implemented on a low-cost 8-bit micro-controller. Yinjiao et al. [65] considered the incorporation of temperature into the OCV-SOC inference process by proposing a temperature based model in which the parameters were estimated using an unscented Kalman filter.

Coulomb Counting Method:

The Coulomb counting method (CCM), also referred to as ampere-hour integral method, is considered an efficient method to estimate the SOC of a cell [66]. According to CCM, SOC is defined as:

$$SOC(t) = SOC(0) - \int_0^t \frac{\eta_i . i(t)}{C_n} dt \tag{1.2}$$

where $SOC(t)$ is the SOC value at instant t, $SOC(0)$ is the initial SOC value, η_i is the Coulombic efficiency, C_n is the battery capacity and $i(t)$ is the current value at instant t, which is negative and positive during charging and discharging, respectively. CCM obviously uses the integral of the charging and discharging current over a t period of time to estimate the SOC, which makes it a straightforward and an easy-to-implement method. Nevertheless, CCM introduces some challenges due to the impact of operation conditions on the battery capacity, which makes it more difficult to be precisely determined. In particular, a low temperature decreases the available capacity and affects the Coulombic efficiency, which influences the accuracy of SOC estimations [67]. A significant work has been conducted to improve the CCM such as [68], [69], and [70], which propose some solutions to overcome its limitations.

1.1.2.6 Comparative Summary of SOC Estimation Approaches

To wrap up, model-based and DD are the most dominants techniques for SOC estimation, although they are not the only methods existing in the literature. Table 1.2 summarizes these techniques and shows the advantages and disadvantages of each category.

Table 1.2 Comparison of the most commonly used SOC estimation methods

SOC Estimation Methods			
Category	Examples	Advantages	Disadvantages
Data-Driven Methods	- Deep Neural Networks (DNN) - Convolutional Neural Networks (CNN) - Long Short-Term Memory (LSTM) - Gated Recurrent Unit (GRU) - Support Vector Machine (SVM) - Fuzzy Logic (FL)	- Do not require a high level of battery knowledge - Models are easy to train and deploy	- Wide range of training data is required - Predictions can be inconsistent with physics laws - Some algorithms need sophisticated hardware for the training phase
Model-Based Methods	- Electrochemical Model (EM) - Equivalent Circuit Model (ECM) - Electrochemical Impedance Model (EIM)	- An accurate initial SOC measurement is not not required - Accurate estimation - Robust and reliable	- Computationally expensive - Require and accurate model with a convenient estimator/observer - The physics of the battery should be known - Require domain expertise
Other Methods	- Coulomb Counting Method - Lookup Table Methods	- Low computational requirements - Easy to implement	- An accurate $SOC(t_0)$ value is needed (initial value) - Temperature and battery aging can affect the OCV measurements - SOC Error accumulation due to inexact sensor measurements - Challenging for online measurements

1.2 Objective of the Thesis

In this study, a novel approach of SOC estimation for Lithium-ion batteries is proposed. The main concept is to design and train a Neural Network with physics constraints based on an energy approach. This method, based on the incorporation of physics knowledge into the learning process, improves the classic paradigm of DD SOC estimation in terms of physics consistency and accuracy. This is realized through the design of a hybrid cost function that combines physical and data losses. A publicly available data of Panasonic 18650PF Lithium-ion battery [1] is used to validate the new approach by training and testing different CNN models, in this work referred to as PGCNN (Physics-Guided Convolutional Neural Network) and N-PGCNN (Non-Physics-Guided Convolutional Neural Network). Models are compared based on accuracy and physics consistency using Mean Absolute Percentage Error (MAPE) and Physics Consistency Index (PCI) metrics. The proposed methodology aims at providing a new learning method to enhance the classic DD NN based SOC estimation techniques.

1.3 Original Contribution

As represented in Section 1.1, over the years, several methods has been proposed in the literature to provide accurate SOC estimation solutions for Lithuim-ion batteries. These methods are either:

(a) Model-Based (MB)
MB methods, which are generally based on EM, ECM, and EIM battery model, require an accurate model with a convenient estimator or observer. MB techniques do not require an accurate initial SOC estimation, and considered to be robust and reliable. However, the physics of the battery should be known, which necessitates a domain expertise.

(b) Data Driven (DD)
DD methods, including NN, LSTM, GRU, SVM, and FL, do not require a high level of battery knowledge, in addition to the facility of training and deployment. However, these methods are not as accurate as MB methods and lack physics consistency.

(c) Other Methods (OM)
OM, including Coulomb Counting and Lookup Tables, have several challenges such as the accumulation of SOC error due to inaccurate sensors and the difficulty of online measurements.

This work uses a novel approach for Lithium-ion SOC estimation that contributes to the DD category. This contribution does not only improve the model accuracy, but also enhances the physics consistency in SOC predictions. This work introduces the idea of utilizing hybrid cost functions for SOC estimation to the literature. As we know, cost functions are generally designed to represent the gap between the model predictions and ground truth, and optimized during the learning phase. The hybrid cost function proposed in this work combines physics and data knowledge (Figure 1.8). The proposed technique is demonstrated using CNN architecture. However, it can be also applied to other NN models for other types of batteries.

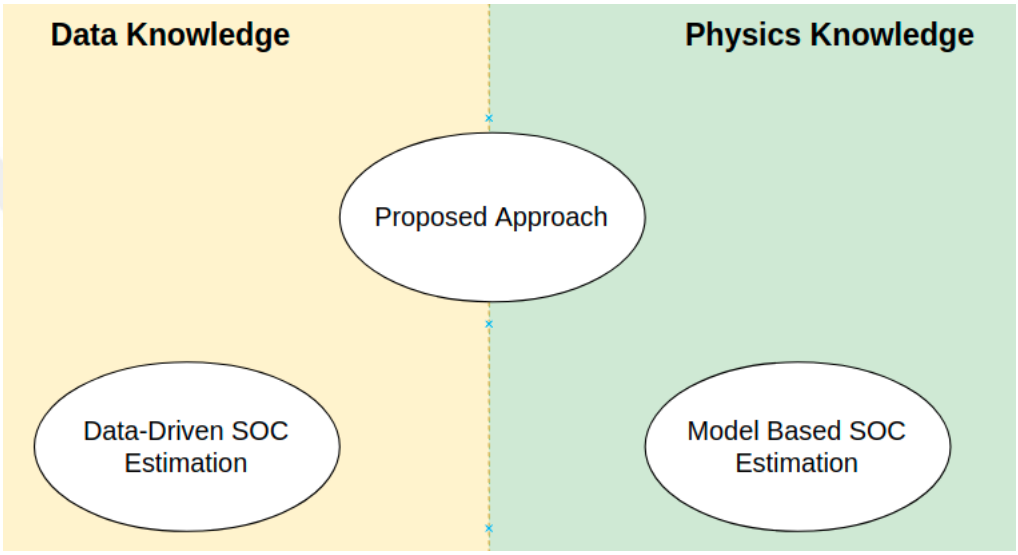


Figure 1.8 The proposed SOC estimation method that combines physical and data knowledge

1.4 Method Synopsis

The main concept in this work is to embed physics knowledge into the learning process of a DD SOC estimation model. This is achieved through designing a custom hybrid cost function that combines physics and data losses, to deliver a more reliable model with a higher consistency. A model is said to be physics consistent, if the predictions do not contradict physics laws. In our case, during a driving cycle in which the available energy in the battery cell should decrease, if the available energy computed from the predicted SOC is increasing over time, the model is considered to be physics inconsistent. In this case, a physical regularization is added to the cost function to guide the model towards higher physics consistency and accuracy during the training phase. Although the proposed method is implemented using a CNN architecture and validated with Li-ion battery cell data, it is also applicable for different NN architectures and other types of batteries.

1.5 Thesis Organization

The first chapter in this thesis provides a thorough review of PGNNs with several examples from the literature (Section 1.1.1) and summarizes the state-of-the-art in SOC estimation for Li-ion batteries (Section 1.1.2). It also highlights the objective of the thesis (Section 1.2) and its original contribution (Section 1.3), and briefly explains the suggested method (Section 1.4).

The second chapter provides the theoretical background of this work, including CNN building blocks (2.1), cost functions (Section 2.2), and Adam optimizer (Section 2.3).

The third chapter explains the proposed method in details by highlighting the structure (Section 3.2) and the training algorithm (Section 3.3) for the PGCNN and N-PGCNN models. The chapter finishes by comparing the the models in term of MAPE (Section 3.4.1) and PCI (Section 3.4.2).

The last chapter (Section 4) in this work provides general conclusions and suggests future works related this topic.

2
THEORETICAL BACKGROUND

2.1 Overview of the Convolutional Neural Networks Building Blocks

CNN has been chosen to demonstrate the proposed paradigm for SOC estimation. This architecture of NNs is widely used in image based deep learning applications including object detection and image classification. However, in this work, we exploit the properties of CNN to make the proposed model able to learn the relation between input features from one side, and the relation between time steps from the other side. In this section we will go through the basic operations of the CNN architecture, and we will explain how this model is used in this work.

As mentioned in [71], the idea behind CNNs is not so different from the classic NNs. Each of them is constructed from several neurons that have trainable parameters: weights and biases. These neurons perform a dot product of the received inputs with the weights followed by an activation function that adds non-linearity to the model. The difference between the model predictions and the targets is expressed by a *cost function* (also referred to a*sloss function*). The major goal of the training phase is to find a set of parameters (i.e. weights and biases) that minimizes this cost function to an acceptable value.

The structure of CNNs is suitable for 3-dimensional images as the model inputs. These dimensions are the image height, width, and depth. The depth of the image is 3 (red, green, and blue channels) for RGB images, and 1 for gray scale and binary images. In our case, the model inputs are not images, but contiguous multiple time series, as shown in 2.1. These time series, including chamber and cell temperatures, voltage, and current, form an $n \times m$ matrix in which n and m are the time steps and the number of time series, respectively. Hence, the input of our model has the same dimension of gray scale and binary images and can be used with a 2-Dimensional CNN (2-D CNN) architecture. CNNs consists of a sequence of layers that perform specific operations on input tensors without reducing their dimensions. Hence, The inputs

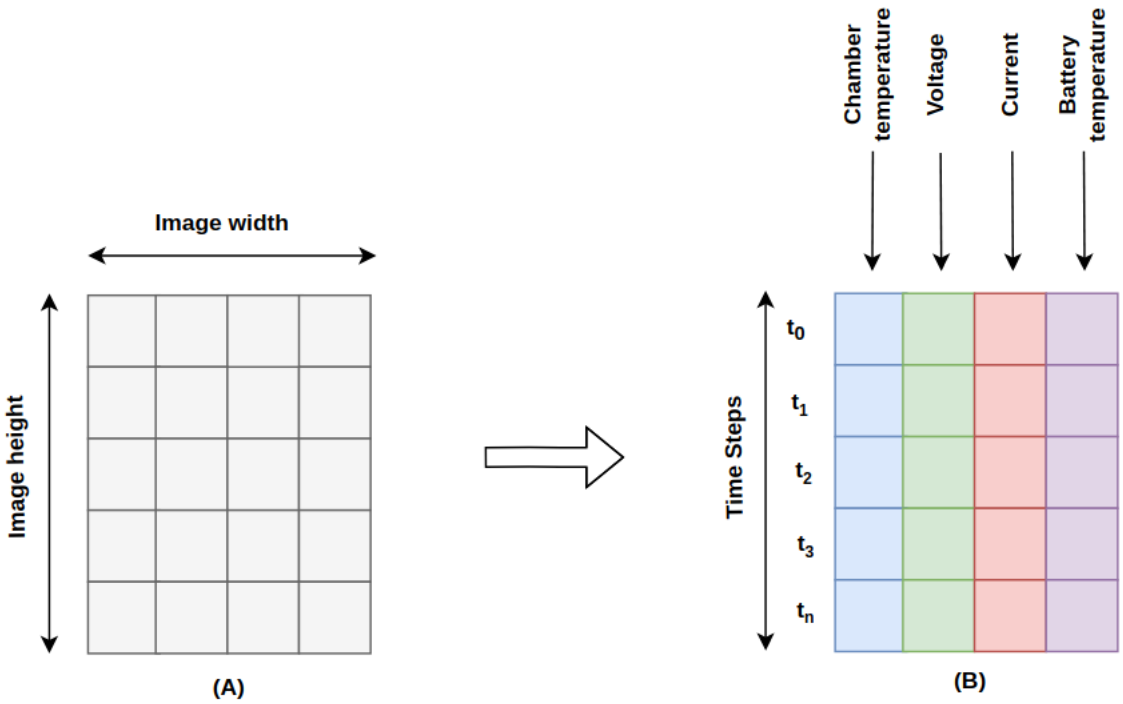


Figure 2.1 The analogy between image inputs and time series inputs for 2-D CNN architecture

(A) is an input image (gray-scale/binary) for 2-D CNN architecture. (B) is the input of our 2-D CNN model. It has the same dimensions of a the gray-scale/binary image (A) and consists of 4 contiguous time series representing chamber and battery temperatures, voltage, and current data. In this work, different time steps were utilized to train and test the CNN model.

of our model, which are 2-D tensors, will have the same dimension after applying the CNN operations.The building blocks of CNN architecture are *Convolutional Layer*, *Pooling Layer*, and *Fully-Connected Layer*.

2.1.1 Convolutional Layer

This layer consists of several filters (also refereed to as kernels) with trainable parameters. These filters have smaller height and width than the input tensor, but the same depth, which is 1 for 2-D inputs, as it is the case for the input tensors used in our model (Figure 2.1). Every filter is slid along the width and height axes while a dot product of the filter and the correspondent region of the input tensor is performed, the operation that produces *feature maps* (also referred to as *activation maps*) that represents the patterns extracted by every filter. These patterns are spacial for image inputs. However, in for our model, they represent the temporal relationship between different time steps from one side, and the relationship between the sensors measurements from the other side. Figure 2.2 shows an illustration of the first 3 steps of the convolution operation of a 5×4 input tensor (5 time steps and 4 features) with a 3×3 filter. Note that the time steps and and filters are reduced in the figure for simplification purposes. In the first step, the dot product of the filter and the region of input tensor covered by the filter is calculated. Then the kernel is translated by 1 stride and the next item of the feature map is calculated in the same way. We can intuitively think of this process as encoding items of the input tensor in term of the neighbor items. After 6 steps, all elements of the feature map are calculated and the size of the input tensor is reduced to 3×2 (the red region of the feature map). To conserve the size of the input tensor, *padding* operation is often performed and the edges of the output tensor are filled with zeros (represented by P in Figure 2.2). Note that same process is repeated with different filters and each one produce a different feature map. These features maps are stacked together to be fed to the next layer of the CNN. ***Receptive Field:*** Connecting all neurons in the input tensor to the neurons of the convolutional layer is not efficient. Instead, only a local region, which is the region covered by the filter, is connected to each neuron in this layer through a number of weights (i.e. the filter items) and 1 bias. For instance, for a 100×4 input tensor and a 3×3 filter, only 9 weights plus 1 bias are needed to connect the the input tensor to each neuron in the convolutional layer (Figure 2.3). We refer to layers using local connectivity as *Locally-Connected Layers*.

Size of The Convolutional Layer Output Tensor: The size of the output tensor is defined by the number of neurons and their spatial placement which is determined by three parameters: the *number of filters*, *stride*, and *zero-padding*:

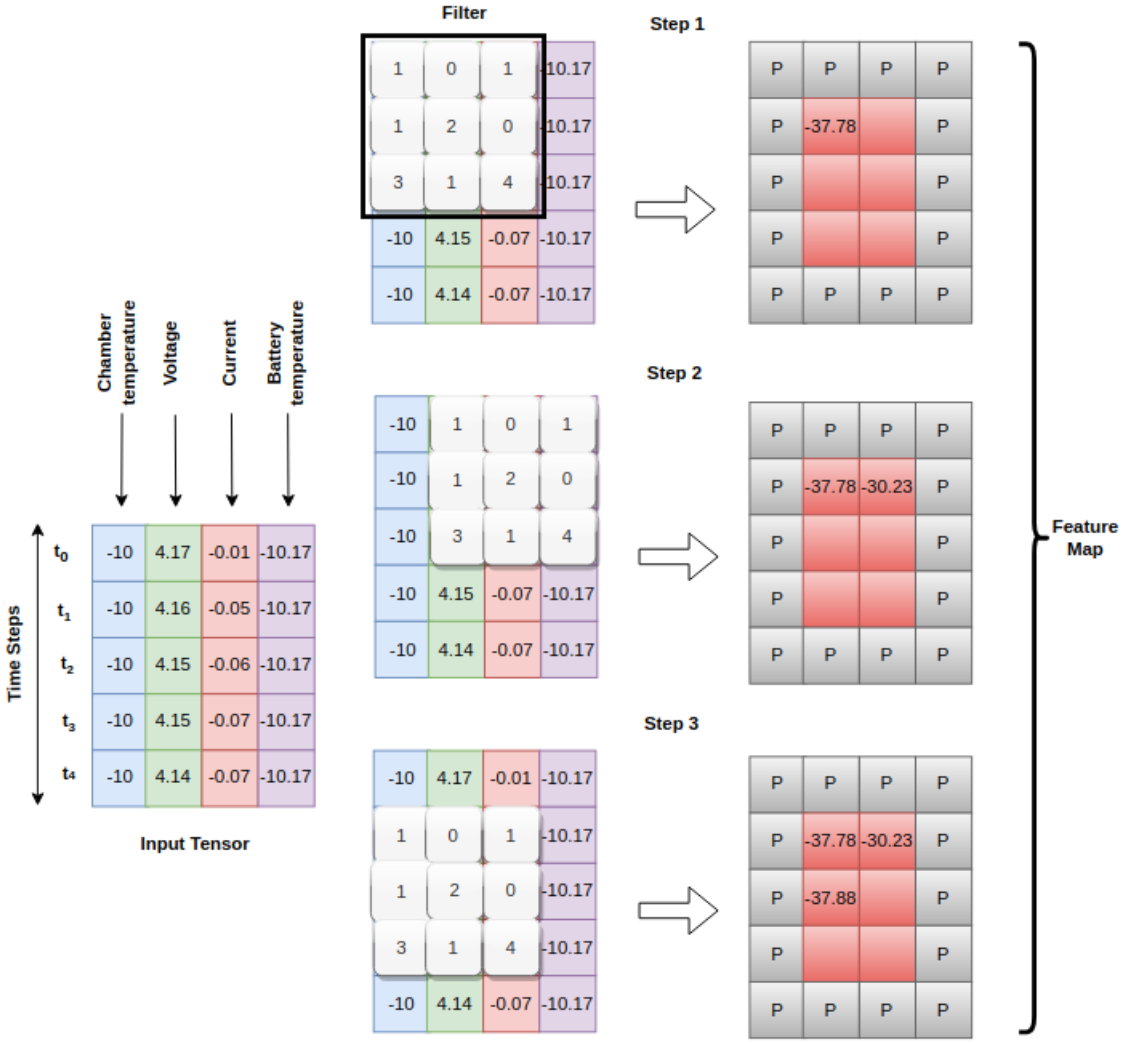


Figure 2.2 Illustration of the first 3 steps of the convolution operation

The first 3 steps of the convolution process performed by a single filter. Every item of the feature map is the dot product of the kernel (filter) and and the region of the input tensor covered by the kernel. In this example only the weights (items of the filter) are represented and the bias is neglected to simplify the figure. Notice that *Padding* is used to preserve the size of the input tensor.

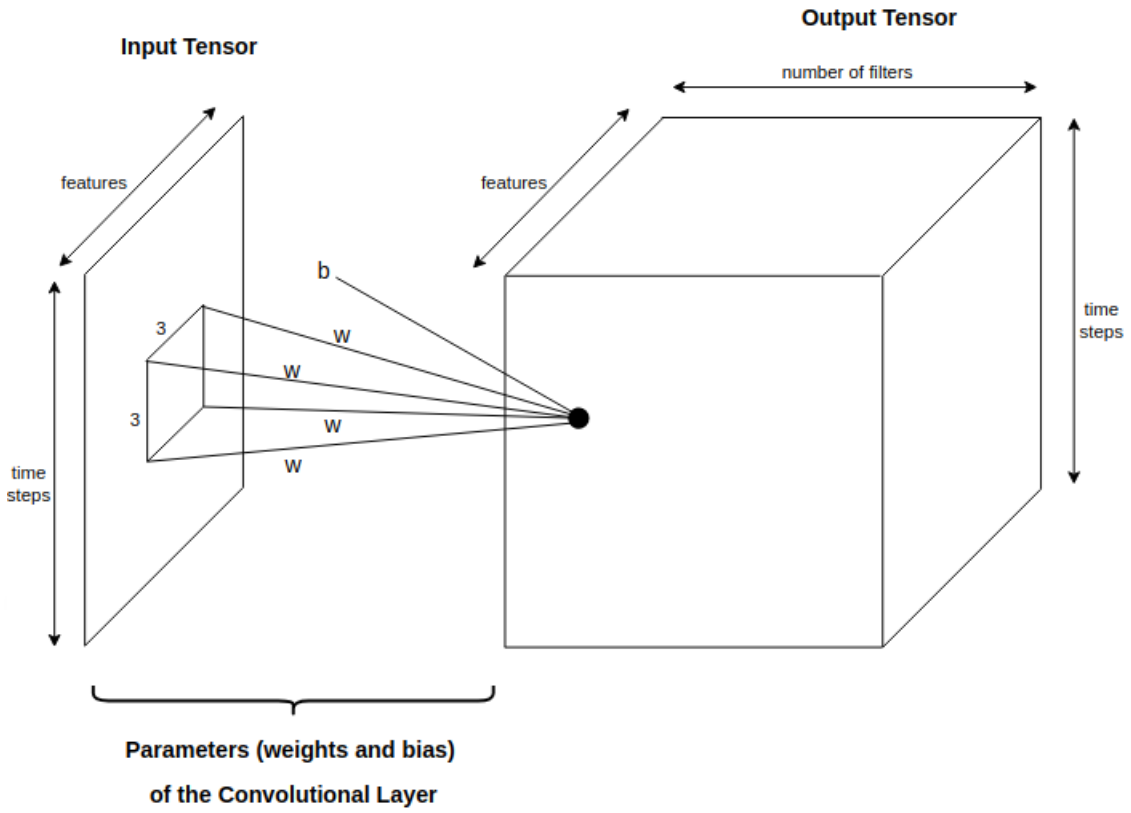


Figure 2.3 Local connection of the input tensor to the neurons of the convolutional layer

The input tensor is locally connected to the neurons of the convolutional layer through weights and bias. The depth of the output tensor is equal to the number of filters. For instance, if we use five 3×3 filters, the total number of parameters would be $3 \times 3 \times 5$ weights $+ 5$ biases $= 50$ trainable parameters. Note that the size of the output tensor (height and width) is not changed because of the zero-padding operation.

Number of filters: Determines the depth of output tensor. For instance if the layer has 5 filters, the depth of the output tensor would be 5.

Stride: The step in which the filters move across the width and the height of the input tensor. In the example shown in Figure 2.2 the stride is one. Bigger strides will generate smaller output sizes.

Zero-padding: Using this parameter we can preserve the size of the input by adding zeros to the borders of the output.

Sharing Parameters Between Neurons in the Same Depth Slices: A depth slice is referred to a set of neurons at the same depth of the convolutional output tensor. When using 5 filters, we can define 5 depth slices containing different neurons with the same x and y coordinates but at different depth levels. In practice, all neurons in the same depth slice can share the same weights and biases, in order to decrease the total number of parameters for more efficient training. The idea behind sharing parameters is that if a feature should activate a neuron with (x_1, y_1) coordinates, it probably should activate its neighbour at (x_2, y_2). Thus, there is no need to relearn this feature. Of course, this is true for some patterns such as vertical or horizontal lines, but will not necessary lead to good results for centered features such as human faces where different features are expected to be learned by different neurons in the same region.

2.1.2 Pooling Layer

Convolutional Layers are often followed by *Pooling Layers* in the CNN architecture. While the convolutional layer is used to extract patterns from input tensors, the main function of the pooling layer is to reduce the input size, and hence reducing the trainable parameters and the complexity of the model which can also help preventing overfitting. It is worth saying that this layer reduces the size of the depth slices independently. In other words, the depth of the input tensor, which equal to the number of filters of the previous convolutional layer, is conserved and only the width and the height of the depth slices are reduced. The size reduction is usually realized by applying the *Max* function. As shown in Figure 2.4, non-trainable filters are applied to the depth slices and only the maximum item in the covered region is taken. The figure shows a 2×2 filter applied to a 5×4 feature map and the first 3 steps of the max pooling process.

Other Types of Pooling: Although *Max Pooling* is the most commonly used Pooling function, other types of pooling also exists, including *Min Pooling*, *Average Pooling*, and *L2-Norm Pooling*.

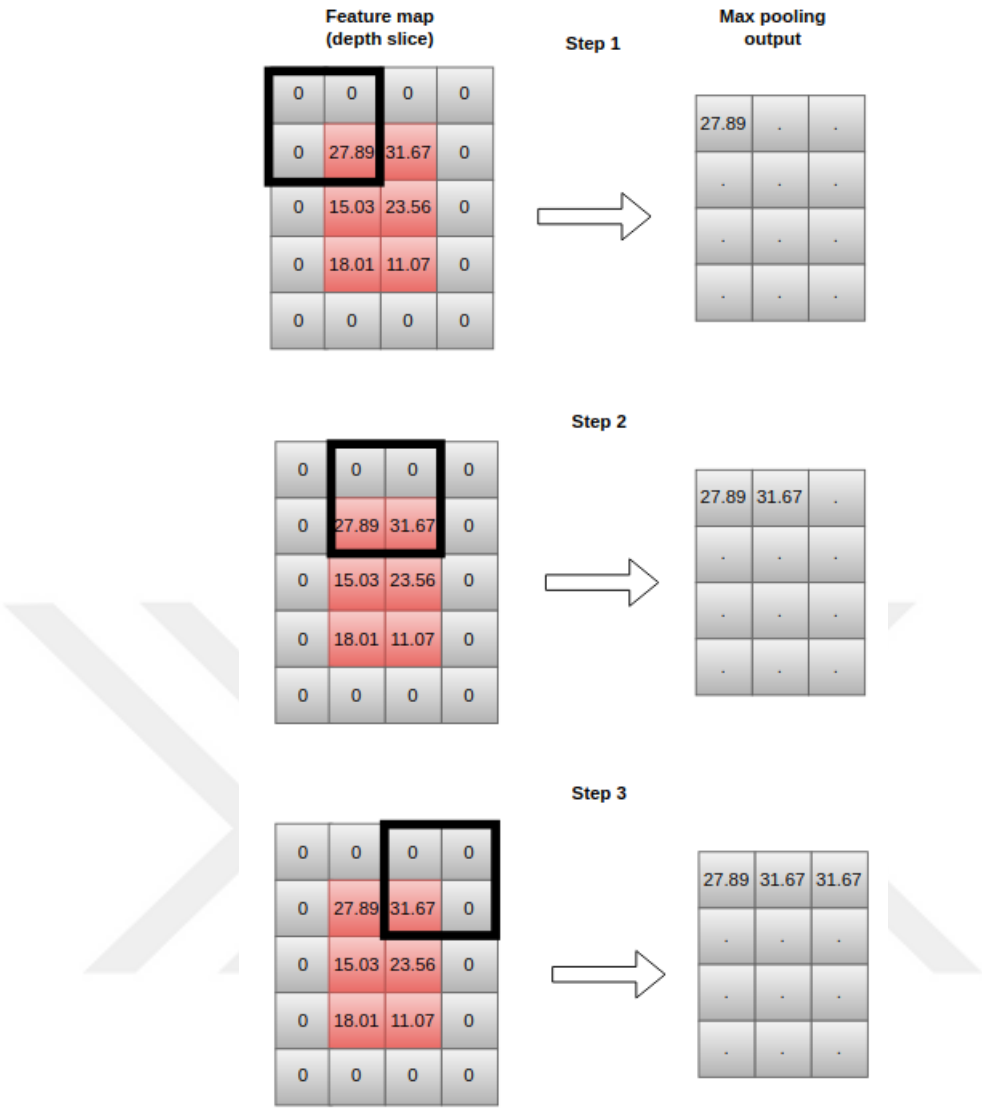


Figure 2.4 Illustration of the first 3 steps of the max pooling operation

The figure shows an application of the Max Pooling function to a 5×4 feature map using a 2×2 filter and a stride equal to 1. In every step, the maximum number of the region covered by the filter is taken, then the kernel is translated by 1 stride, and so on. The feature map is one depth slice of the output of the convolutional layer. Remember that using n number of filters in the convolutional layer will generate n depth slices. Since the purpose of this function is to reduce the size of the input, no zero-padding is needed.

Min Pooling: Every stride, the minimum value of the numbers covered by the filter is taken.

Average Pooling: Instead of taking the maximum or minimum value of the region covered by the filter, the average of all values is calculated.

L2-Norm Pooling: This method uses the *L2-Norm* (also known as *Euclidean distance*) of the feature map region covered by the filter.

2.1.3 Fully-Connected Layer

The *Fully-Connected Layer* is the type of layers commonly used in classic NNs. Each neuron of this layer is connected to all activations in the previous layer, as shown in Figure 2.5. In CNN architecture, *Pooling Layers* are generally followed by one or more fully-connected layers, the last of which represents the output layer and generates the prediction of the model. Figure 2.6 summarizes the main steps of CNN operations. In practice, we usually use more than one convolutional, pooling, and fully-connected layers. However, the figure is only a simplification of the CNN architecture. The details of the structure of the model used in this thesis is given in next sections.

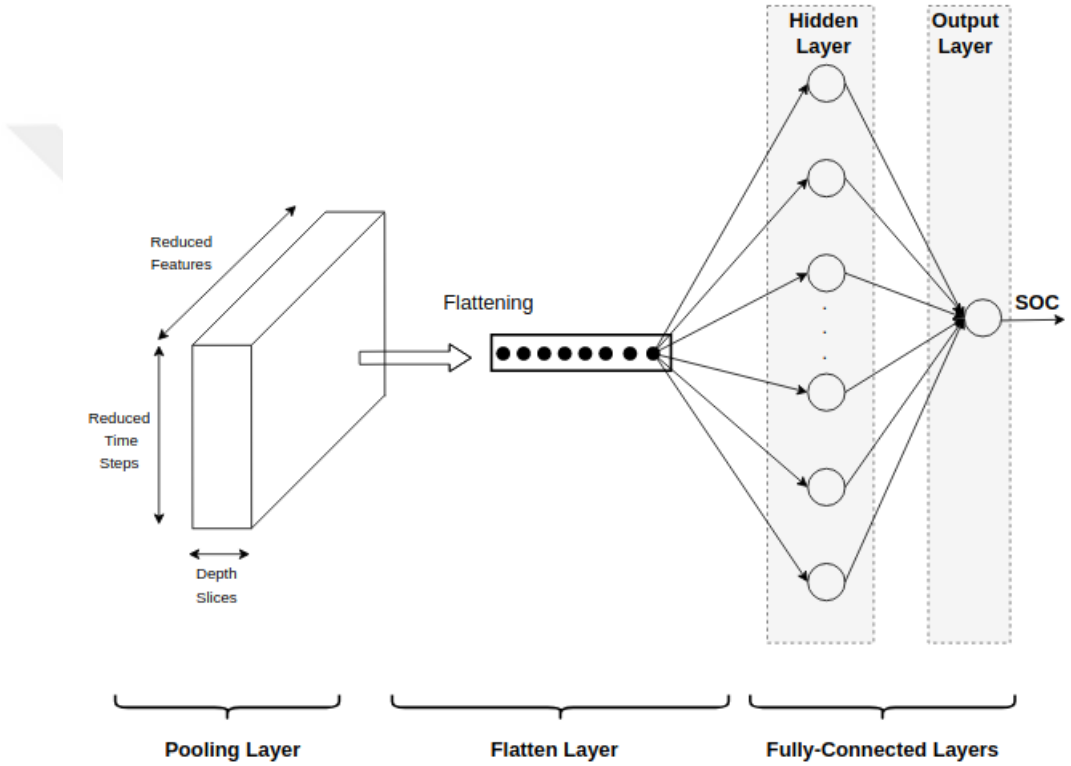


Figure 2.5 The connection between the pooling layer output tensor with the fully connected layers

Each neuron of the fully-connect layers is connected to all activations of the pooling layer. The flatten layer is referred to a simple shape conversion operation. For instance, if the input tensor has a shape $15 \times 3 \times 5$ (15, 3, and 5 are the reduced height and width of the feature maps, and the number of filters used in the previous convolutional layer (depth slice), respectively), after the flattening operation the output will have a shape of 225×1 ($255 = 15 \times 3 \times 5$).

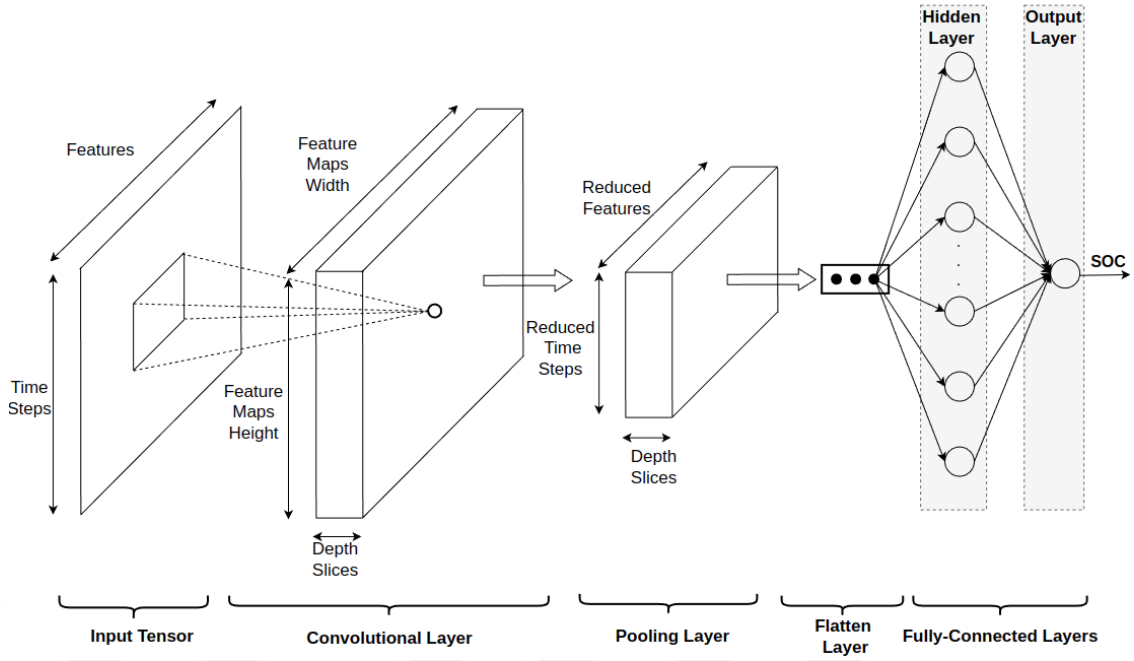


Figure 2.6 Summary of the main building blocks of CNN architecture

This figure is simplification of the CNN model used for this thesis. The input tensor used in this work is two-dimensional as explained before, the width and the height represent the sensors measurements (voltage, current, chamber and cell temperatures) and the time steps, respectively. The model output is the state-of-charge estimation.

2.1.4 Batch Normalization Layer

It worth mentioning that *Batch Normalization*[72] can be useful (but not necessary) for CNN architectures to accelerate the learning process and deliver a faster convergence and stable training. Batch Normalization consists of changing the data distribution to have $\mu = 0$ and $\sigma = 1$, with μ and σ are the distribution mean and standard deviation, respectively. The data points are normalized using the following formula, where V_i denotes the vector to be normalized, $\hat{V}_i$ is the normalized vector, and μ_i and σ_i are the vector mean and standard deviation.

$$\hat{V}_i = \frac{V_i - \mu_i}{\sigma_i} \tag{2.1}$$

The dimension of the vector V_i depends on the shape of the input data. In our case, we use 2-D training data ($time\ steps \times sensors\ measurements$) and one extra dimension (depth slice) will be added after the convolutional layer. Therefore, the input shape of a batch normalization layer situated after a convolutional layer will have a shape of $batch size \times time steps \times sensors measurements \times depth slices$ where $batch\ size$ is the number of training samples per batch. Hence, performing batch normalization according to the last axis of the input data, means that the V_i vector

will be two-dimensional. Figure 2.7 shows an example of batch normalization layer input tensor.

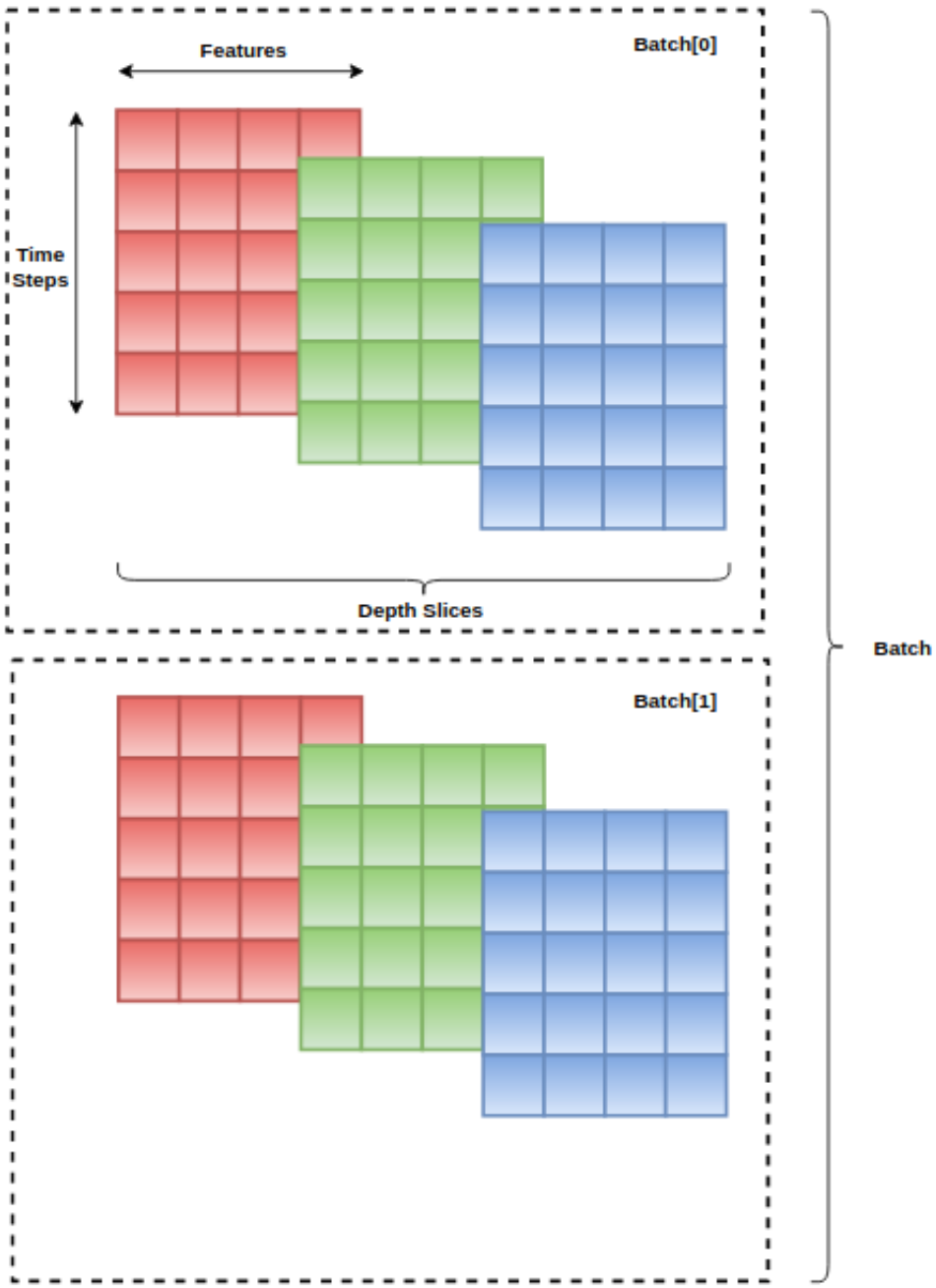


Figure 2.7 Batch normalization input tensor

The figure shows an example of a 2-batch input tensor for natch normalization Layer. The tensor has 4 axis: batches (axis 0), times steps (axis 1), features (axis 2), and depth slices (axis 3). The shape of the represented tensor is $(2, 5, 4, 3)$.

Similarly to other layers, the batch normalization layer also has trainable parameters. These parameters are utilized to scale and shift the data distribution, denoted as γ and β, respectively. The scaling and shifting operations are performed by multiplying every item of the normalized data by γ factor and adding to it an offset of β. These

factors grant learning the best mean and standard deviation for the input distribution during the training phase for optimal model predictions. The output BN_i of the Batch Normalization Layer is given by:

$$BN_i = \gamma \odot \hat{V}_i + \beta \tag{2.2}$$

Where $\hat{V}_i$ is the normalized input vector, and γ and β are the scaling and shifting factors, respectively.

During the inference phase, when the input batch contains only one item, the *moving average mean* μ_{mov_i} and the *moving average standard deviation* σ_{mov_i} are used to calculate to the normalized vector, which are calculated during the training phase giving the following formulas:

$$\mu_{mov_i} = \alpha\mu_{mov_i} + (1-\alpha)\mu_i \tag{2.3}$$

$$\sigma_{mov_i} = \alpha\sigma_{mov_i} + (1-\alpha)\mu_i \tag{2.4}$$

Where α denotes the momentum value. α is a hyperparmater (i.e. fixed parameter that does not change during the training phase) that determines the change speed of the moving average.

2.2 Cost Function

Cost functions measure the error between the model predictions and the ground truth. They evaluate how well the NN models the dataset [73]. Cost functions, also referred to as *loss functions* or *objective functions*, depend on of the model parameters and they are optimized during the learning phase. The optimization process consists of finding the best model parameters that minimize the difference between the model predictions and the targeted outputs to a satisfactory error value.

Generally, we distinguish three categories of cost functions depending on the type of the problem being solved by the NN: *Regression Loss Functions*, *Binary Classification Loss Functions*, and *Categorical Classification Loss Functions*.

(a) Regression Loss Functions
This type of loss functions is, obviously, used for *regression* problems. Regression models predict a continuous value such as the velocity of a car or the pitch angle of an aircraft. In this work, we use this type of loss function, since we are predicting the SOC of a battery cell, which is a continuous value.

(b) Binary Classification Loss Functions
In binary classification problems, the output can only be either 1 or 0. An example of this kind of problems would be the detection of a disease in a patient. The diagnosis result could be either positive of negative. For these type of problems we use the *binary classification loss function* type.

(c) Categorical Classification Loss Functions
As the name indicates, categorical (also referred to multi-class) classification problems classify an input tensor into at least two classes. A simple example would be the classification of a person photo into male of female categories. This is considered as an extension of the binary classification problem. For this type of problems we use the *categorical classification loss function* type.

Table 2.1 summarizes the most commonly used loss functions for regression and classification problems.

Table 2.1 The most commonly used loss functions in regression and classification problems

Regression	Mean Squared Error (MSE)	$L = \frac{1}{n}\sum_{i=0}^{n}(y_i - \hat{y}_i)^2$
	Mean Absolute Error (MAE)	$L = \frac{1}{n}\sum_{i=0}^{n}\lvert y_i - \hat{y}_i\rvert$
	Mean Absolute Percentage Error (MAPE)	$L = \frac{1}{n}\sum_{i=0}^{n}\left\lvert\frac{y_i - \hat{y}_i}{y_i}\right\rvert \times 100$
	Mean Squared Logarithmic Error (MSLE)	$L = \frac{1}{n}\sum_{i=0}^{n}(log(y_i+1) - log(\hat{y}_i+1))^2$
Classification	Binary Crossentropy	$L = -\frac{1}{c}\sum_{i=1}^{c} y_i.log\hat{y}_i + (1-y_i).log(1-\hat{y}_i)$
	Categorical Crossentropy	$L = -\sum_{i=0}^{c} y_i.log\hat{y}_i$

2.3 Adam Optimizer

The combined cost function, used in this work and explained in next chapter, is optimized using the *Adam* algorithm, which is an efficient gradient-based optimization method for stochastic functions that only requires first-order gradients with little memory requirements, as stated in the original paper [74]. In contrast with *Stochastic Gradient Descent (SGD)*, Adam uses an adaptive learning rate i.e. different learning

rates for different parameters [75]. The name *Adam* is derived from its working mechanism, which consists of estimating the first and second moments of gradient to calculate the learning rate corresponding to each parameter in the NN. The n^{th} moment of a random variable is defined in Equation 2.5, where m_n, E, and X^n denote the n^{th} moment, the expected value, and the random variable to the power of n:

$$m_n = E[X^n] \tag{2.5}$$

During forward propagation, the cost function is computed using a random batch of data. Hence, the gradient of the cost function can be considered as a random variable. Adam algorithm calculates the first and second moments of the gradient, which are the average and the uncentered variance, using moving averages, as indicated in Equation 2.6 in which the first and second moments, and the gradients of the cost function are denoted as m_t, v_t, g_t, respectively:

$$\begin{cases} m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \\ v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \end{cases} \tag{2.6}$$

β_1 and β_2 are two hyper-parameters that define the smoothness of the moving averages. These parameters have default values of 0.9 and 0.999 respectively, in the open-source library *Keras* [76] implementation that we are using in this work. The moving averages in Equation 2.6 can be extended and rewritten as:

$$\begin{cases} m_t = (1-\beta_1) \sum_{i=0}^{t} \beta_1^{t-i} g_i \\ v_t = (1-\beta_2) \sum_{i=0}^{t} \beta_2^{t-i} g_i^2 \end{cases} \tag{2.7}$$

Thus, the expected value of m_t can be formulated as in Equation 2.8 where ϵ is the gradient approximation error.

$$\begin{aligned} E[m_t] &= E\Big[(1-\beta_1) \sum_{i=0}^{t} \beta_1^{t-i} g_i\Big] \\ &= E[g_t](1-\beta_1) \sum_{i=0}^{t} \beta_1^{t-i} + \epsilon \\ &= E[g_t](1-\beta_1^t) + \epsilon \end{aligned} \tag{2.8}$$

With the same analogy, the expected value of the second moment can be formulated as:

$$\begin{aligned} E[v_t] &= E\Big[(1-\beta_2)\sum_{i=0}^{t}\beta_2^{t-i}g_i^2\Big] \\ &= E[g_t^2](1-\beta_2)\sum_{i=0}^{t}\beta_2^{t-i} + \epsilon \\ &= E[g_t^2](1-\beta_2^t) + \epsilon \end{aligned} \tag{2.9}$$

After a bias correction, the first and second moments estimations can be written as:

$$\begin{cases} \hat{m}_t = \frac{m_t}{1-\beta_1^t} \\ \hat{v}_t = \frac{v_t}{1-\beta_2^t} \end{cases} \tag{2.10}$$

The NN parameters are updated based on these estimations, according to the the learning rule of Adam optimization algorithm:

$$\hat{w}_t = w_{t-1} - \eta\frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon} \tag{2.11}$$

Note that η is the learning rate, and ϵ is a small value is added to the denominator to avoid dividing by zero.

3
SOC ESTIMATION USING PHYSICS GUIDED CONVOLUTIONAL NEURAL NETWORK

3.1 Panasonic 18650PF Lithium-ion Battery Data

Panasonic 18650PF Li-ion battery data [1] has been used to train and test the N-PGCNN and PGCNN models. This data set was mainly collected for research purposes and it was intended to be used to test and compare NN and Kalman Filter state-of-charge estimation algorithms or to develop battery models.

As stated in the original description, the dataset consists of different tests, all of which were conducted using a new $2.9Ah$ Panasonic 18650PF cell in a $0.226cm^3$ ($8ft^3$) thermal chamber with a $25A$, $18V$ Universal Battery Tester (UBT) and other test equipment manufactured by Digatron.

UBT's are mainly used for cycling, capacity and internal resistance testing of automotive batteries. However, these test equipment can be used with other types of batteries as well. The UBT conducts charging and discharging cycles specified by the test schedule. It can be either controlled with a battery manager or BTS-600 PC software. Whereas the latter is generally used for research and development purposes, the former is utilized for industrial testing and quality assurance [77]. Some advanced testers such as the Lithium Cell Tester MCT Series of Diagtron, shown in Figure 3.1, offer up to 128 charge/discharge circuits per cabinets and support several options including current range switching and paralleling of circuits, fast mode switching, and additional input channels via CAN-Bus [78].

The data set was collected using a Panasoic NCR18650PF Li-ion battery cell. This type of batteries is characterized by a high energy and power density, and can be used in power assisted bicycles, 2-way radios, medical devices and robotics, according to [80]. The specifications of this battery and the test equipment [81] are summarized in Table 3.1 and Table 3.2, respectively.



Figure 3.1 DIGATRON MCT-ME cell tester [79]

Table 3.1 NCR18650PF Lithium-ion battery specifications

Max. Capacity	$2900mAh$
Nominal Voltage	$3.6V$
Min / Max Voltage	$2.5V$ / $4.2V$
Charge Temperature	0 to +45°C
Discharge Temperature	−20 to +60°C
Storage Temperature	−20 to +50°C
Energy Density	$577Wh/l$
Weight (max.)	$48.0g$

Table 3.2 Test equipment specifications

Cell Cycler Manufacturer	Digatron
Test Channel	$25A$, $0-18V$ Channel
Voltage / Current Accuracy	±0.1%
Data Acquisition Rate	$10Hz$
Thermal Chamber	Cincinatti Sub Zero ZP-8
Thermal Chamber Size	$0.226cm^3$ $(8ft^3)$
Thermal Chamber Temperature Accuracy	±0.5°C

As mentioned in [81], the data was generated by performing several drive cycles under different ambient temperatures.The same paper has also explained the test procedure, which is summarized in Figure 3.2. The test procedure starts by setting the temperature of the thermal chamber to 25°C and waiting for 3 hours until the

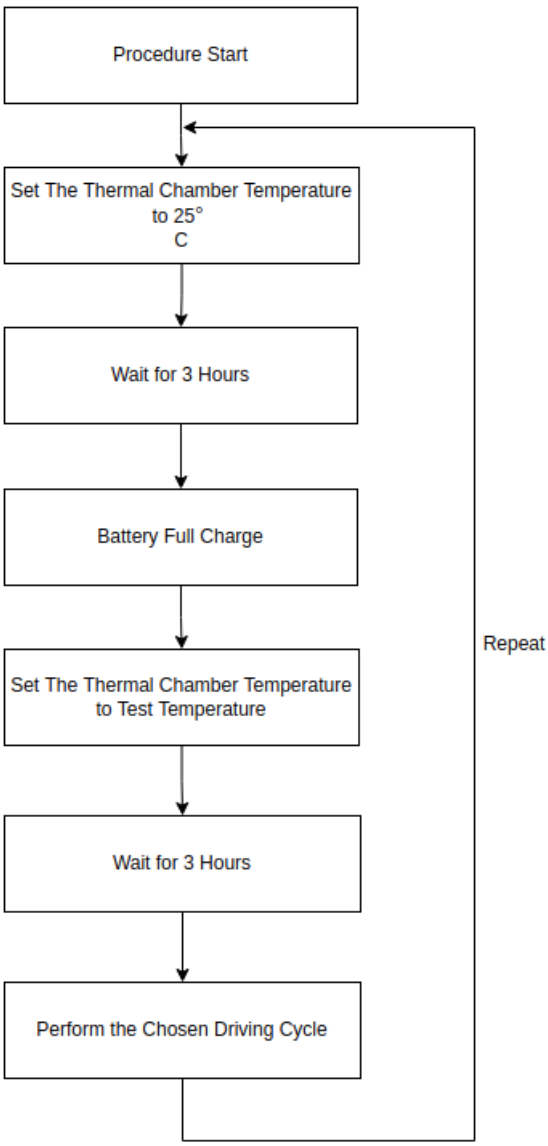


Figure 3.2 Battery test procedure

battery internal temperature reaches 25°C. Then, Constant-Current Constant-Voltage (CC/CV) method is used to fully charge the battery, during which a constant current charge rate of $2.9A$ ($1C$) is applied until reaching the maximum battery voltage, followed by applying a constant voltage of $4.2V$ (the battery maximum voltage), which is in turn terminated when the current decreased to a cut off value set to $50mA$. After the charging phase, the thermal chamber is set to the test temperature. As the time of writing, this data set contains drive cycles data recorded for 5 test temperatures: -20°C, -10°C, 0°C, 10°C, and 25°C. The charge phase is followed by discharging the battery using one of the drive cycles given in Table 3.3 [82].The last drive cycle (cycle 9), contains some additional dynamics useful for training and validation NNs. The Data set also includes some other tests which are not used for the purpose of this work.

Table 3.3 Drive cycles used for battery discharging

Cycles 1-4	Random mix of US06, HWFET, UDDS, and LA92
Cycle 5	US06: High Acceleration Aggressive Driving
Cycle 6	HWFET: Highway Fuel Economy Driving Schedule
Cycle 7	UDDS: Urban Dynamometer Driving Schedule
Cycle 8	LA92: Class 3 Heavy-Duty vehicles
Cycle 9	Neural Network (NN): Combination of US06 and LA92,

Note that these cycles includes regenerative braking power for test temperatures equal or greater than 10°C. This is the temperature in which this type of batteries can be charged with $1C$ rate. Below this temperature (i.e. between 0°C and 10°C, the battery should be charged with $0.25C$ rate. The minimum charging temperature is 0°C), as mentioned in the datasheet [80]. During generative breaking, the energy is transferred into the cell which is being charged.

A 5000-sample random chunks from the train, validation, and test data sets are shown are shown in Figures 3.3, 3.4, and 3.5, respectively. The top 4 graphs in each figure represent the model features including chamber and battery temperatures, voltage, and current. The last graph shows the SOC targets corresponding to these features. The sizes of train, validation, and test data sets used in this work are given in Table 3.4.

Table 3.4 Sizes of train, validation, and test data sets

Data Set	Number of Samples	Ratio of Total Available Data (%)
Train	266000	70
Validation	76000	20
Test	38000	10

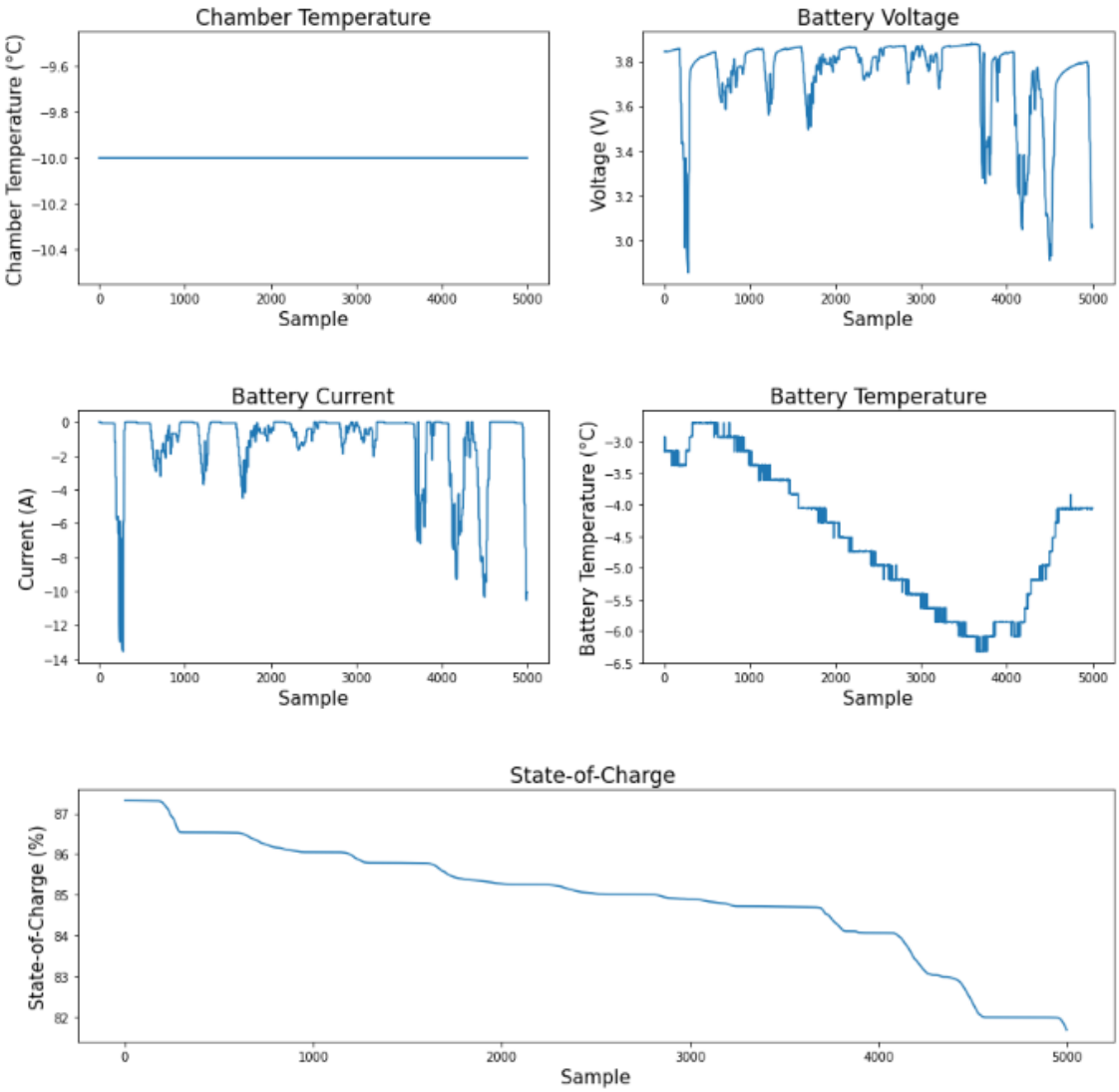


Figure 3.3 A 5000-sample random chunk from the train data set

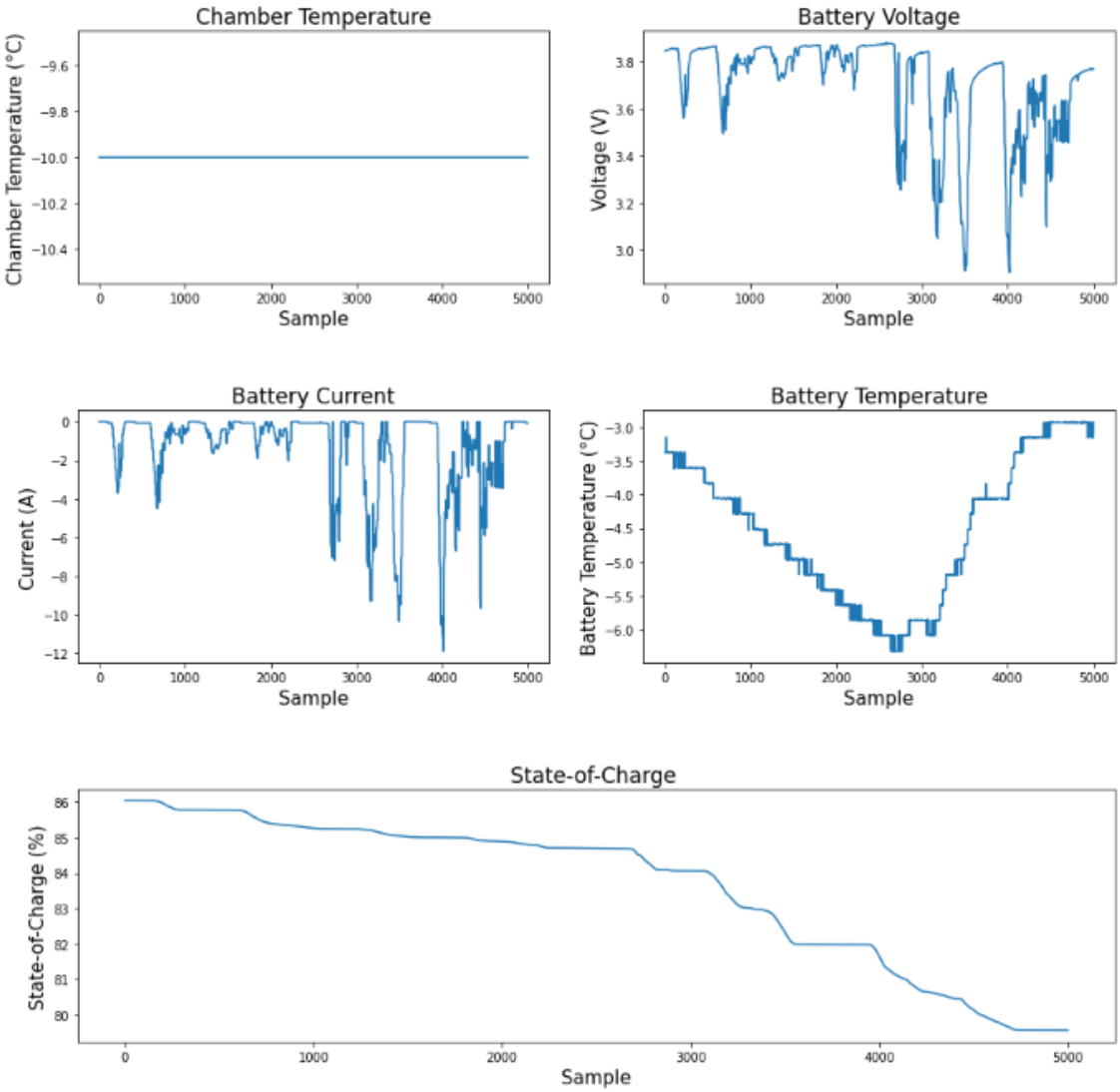


Figure 3.4 A 5000-sample random chunk from the validation data set

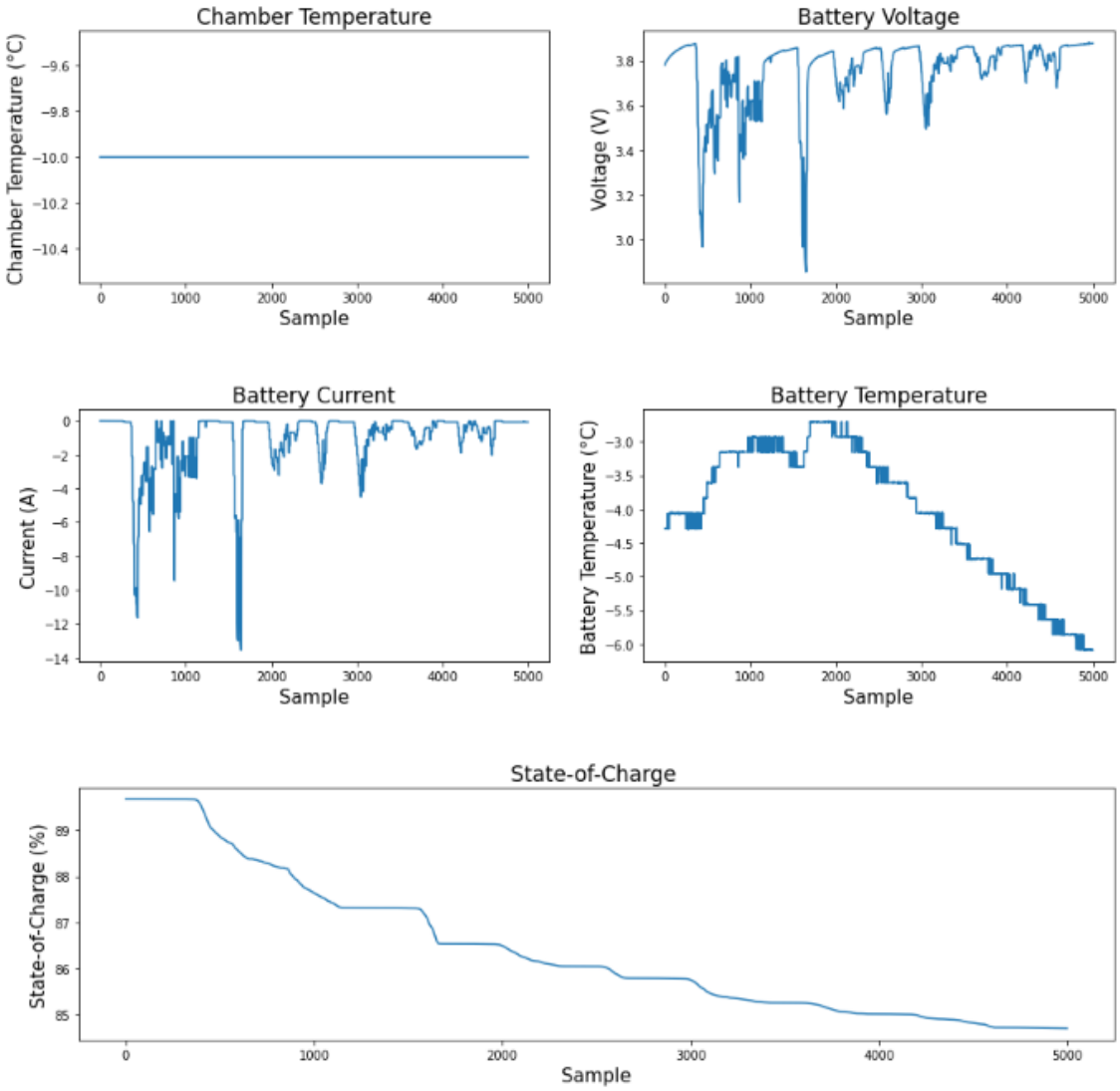


Figure 3.5 A 5000-sample random chunk from the test data set

3.2 PGCNN and N-PGCNN Models Structure

The structure of the CNN model used in this work is given in Table 3.5. The model consists of 9 layers, including *2-D Convolutional Layers*, *2-D Max Pooling Layers*, *Batch Normalization Layers*, and *Fully-Connected Layers*. The Parameters of each layer were tuned in order to get the best training and test results. These Parameters are:

Number of filters: The number of kernels used for features extraction. n number of filters will generate n feature maps for the convolutional layer output.

Filter size: The size of feature extraction kernel in form of $(height, width)$

Stride: The pace in which the kernels move, with an $n \times m$ stride the kernels will move n steps along the height and m steps along the width at each step.

Zero-Padding: Maintaining the same size of the input by adding zeros to the border of the output. This is useful to conserve the size of the convolutional layer input tensor.

Normalization axis: Determines which dimension of the input data will be normalized.

Momentum: A hyperparameter used in the calculation of the moving mean and standard deviation averages (see Equations 2.3 and 2.4). This parameter determines the changing speed of the moving averages.

Epsilon: Small float added to the standard deviation in Equation **??** to prevent the division by zero.

Centring: Shifting the data by α trainable factor (see Equation 2.2).

Scaling: Scaling the data by γ trainable factor (see Equation 2.2).

Number of units: The number of neurons in the layer.

Figure 3.6 shows a visualization of the proposed model in *TensorBoard*, a tool offered by TensorFlow [83] machine learning framework.

Table 3.5 Structure of the CNN model

Layer	Layer Type	Activation	Parameters
Layer 1	2-D Convolutional	Leaky ReLU	Num. of filters: 10 Filter size: 21×4 Stride: 1×1 Zero-Padding
Layer 2	2-D Max Pooling	No Activation	Filter size: 2×2 No Padding
Layer 3	Batch Normalization	No Activation	Norm. axis: Depth axis Momentum: 0.99 Epsilon: 0.001 Centering Scaling
Layer 4	2-D Convolutional	Leaky ReLU	Num. of filters: 5 Filter size: 7×4 Stride: 1×1 Zero-Padding
Layer 5	2-D Max Pooling	No Activation	Kernel size: 2×2 No Padding
Layer 6	Batch Normalization	No Activation	Normalization axis: Depth axis Momentum: 0.99 Epsilon: 0.001 Centering Scaling
Layer 7	Flatten	No Activation	No Parameters
Layer 8	Fully-Connected	ReLU	Num. of units: 15
Layer 9	Fully-Connected	Sigmoid	Num. of units: 1

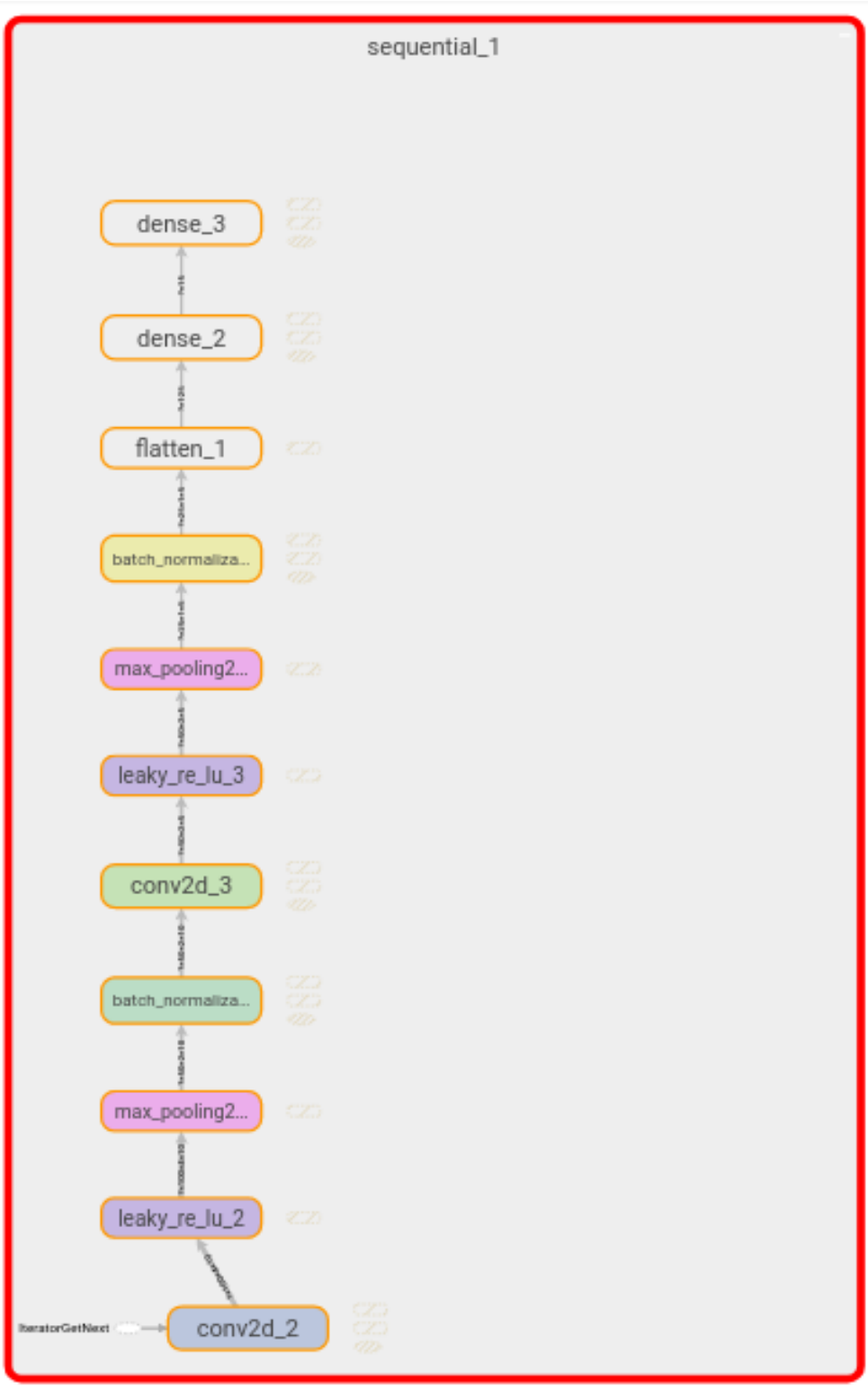


Figure 3.6 Visualization of the proposed CNN model structure

The graph shows the building blocks of the proposed CNN model. The data flow from the bottom (the first convolutional layer, *conv2d_2*) to the top (the last fully-connected layer, *dense_3*). Note that the activation functions of the convolutional layers are represented as separate layers in the graph.

3.3 Training Algorithm for PGCNN and N-PGCNN Models

The CNN model presented in the previous section is trained in a way that introduces physics knowledge into the learning process. As discussed before, NNs are generally confined to the data they are exposed to, during the training phase and do not necessarily deliver physics consistent predictions. To overcome this limitation, we have designed a hybrid loss function that guides the CNN during the learning phase in a way that increases the physics consistency. The simplified training algorithm for the PGCNN is given in Figure 3.7. The symbols and abbreviations used in the algorithm are explained in Table 3.6. Note that ⊙ represents the element-wise product operator.

Algorithm 1 Train physics guided convolutional neural network

$k, s = 0$

while $k < EPOCHS$ **do**

  for $s = 0$ to SPE **do**

    $B_{train} \leftarrow BG_{train}()$

    $SOC_{pred} \leftarrow CNN\left(B_{train}\right)$

    $L_{data} \leftarrow MAPE\left(SOC_{target}, SOC_{pred}\right)$

    $SOC_{Phy_0} \leftarrow CNN\left(IN_{Phy_0}\right)$

    $SOC_{Phy_{100}} \leftarrow CNN\left(IN_{Phy_{100}}\right)$

    $E_{Phy_0} \leftarrow Q.\Delta SOC \odot V_{Phy_0}$

    $E_{Phy_{100}} \leftarrow Q.\Delta SOC \odot V_{Phy_{100}}$

    $L_{Phy} \leftarrow \lambda.Mean\left(ReLU\left(E_{Phy_{100}} - E_{Phy_0}\right)\right)$

    $L_{total} \leftarrow L_{data} + L_{Phy}$

    $P \leftarrow P_{updated}$

  end for

  $k \leftarrow k + 1$

end while

Figure 3.7 Train algorithm for PGCNN and N-PGCNN models

In the training algorithm, the total loss L_{total} consists of two parts: the *physical loss* L_{Phy} and *data loss* L_{data}. In this work, the "data loss" term is used to describe the difference between the ground truth and the model predictions, without including any physical regularization. This difference is expressed as MAPE, as shown in Equation 3.1, in which the SOC ground truth and predictions as referred to as SOC_{target} and SOC_{pred}, respectively.

Table 3.6 Symbols and abbreviations used in algorithm 1

B_{train}	Train Batch
BG_{train}	Train Batch Generator
CNN	Convolutional Neural Network
$CNN(X)$	Inference of X input tensor
$EPOCHS$	Number of training iterations
E_{Phy_0}	Energy vector corresponding to IN_{Phy_0}
$E_{Phy_{100}}$	Energy vector corresponding to $IN_{Phy_{100}}$
IN_{Phy_0}	First input tensor for physical loss regularization
$IN_{Phy_{100}}$	Second input tensor for physical loss regularization
k	Training iteration index
L_{data}	Loss calculated from the model prediction
L_{Phy}	Physical loss
L_{total}	Total loss
$MAPE$	Mean Absolute Percentage Error
P	Trainable parameters of the CNN model
$P_{updated}$	Trainable parameters updated after back propagation
Q	Battery rated capacity
SOC_{pred}	State-of-Charge prediction
SOC_{Phy_0}	State-of-Charge predictions inferred from IN_{Phy_0}
$SOC_{Phy_{100}}$	State-of-Charge predictions inferred from $IN_{Phy_{100}}$
SOC_{target}	State-of-Charge ground truth
SPE	Steps Per Epochs, the number of batches in each epoch (iteration)
V_{Phy0}	Voltage vector corresponding to IN_{Phy_0}
$V_{Phy_{100}}$	Voltage vector corresponding to $IN_{Phy_{100}}$
ΔSOC	The change in State-of-Charge
λ	Physical regularization rate

$$L_d = \frac{1}{n}\sum_{i=0}^{n}\left|\frac{SOC_{target_i} - SOC_{pred_i}}{SOC_{target_i}}\right| \times 100 \tag{3.1}$$

Generally speaking, cost functions in NNs are based on the difference between the model predictions and the ground truth, which can be expressed in various ways depending on the problem solved by the model and its output nature. Some of the most commonly used loss functions are given in Table 2.1, where n, c, y, and $\hat{y}$ represent the number of samples, the number of classes, the ground truth and the model prediction, respectively. Note that in this thesis the *cost* and *loss* terms are used interchangeably and by them, we mean exactly the same thing.

In this work, we introduce a physical regularization to the cost function to include physical knowledge into the learning process of the CNN. The total loss is described in Equation 3.2.

$$L_{total} = \underbrace{MAPE\left(SOC_{target}, SOC_{pred}\right)}_{\text{data loss } (L_d)} + \underbrace{\lambda.Mean\left(ReLU\left(E_{Phy_{100}} - E_{Phy_0}\right)\right)}_{\text{physical loss } (L_{Phy})} \quad (3.2)$$

E_{Phy_0} and $E_{Phy_{100}}$ are two energy vectors, calculated from the model predictions SOC_{Phy_0} and $SOC_{Phy_{100}}$ which are the inference results of two physical regularization input tensors IN_{Phy_0} and $IN_{Phy_{100}}$, respectively, as shown in Equations 3.3, 3.4, 3.5, and 3.6.

$$SOC_{Phy_0} = CNN\left(IN_{Phy_0}\right) \quad (3.3)$$

$$SOC_{Phy_{100}} = CNN\left(IN_{Phy_{100}}\right) \quad (3.4)$$

$$E_{Phy_0} \approx Q.\Delta SOC_{Phy_0} \odot V_{Phy_0} \quad (3.5)$$

$$E_{Phy_{100}} \approx Q.\Delta SOCPhy_{100} \odot V_{Phy_{100}} \quad (3.6)$$

Equations 3.5 and 3.6 give an approximation of the battery available energy, where Q is rated capacity scalar value, V_{Phy_0} and $V_{Phy_{1000}}$ are the voltage vectors of the input tensors IN_{Phy_0} and $IN_{Phy_{100}}$, respectively. The exact energy formula is given in Equation 3.7, where OCV is the open circuit voltage.

$$E(t) = Q.\int_{SOC_{min}}^{SOC_t} OCV(\epsilon)\, d\epsilon \approx Q.V.\Delta SOC \quad (3.7)$$

The input tensors IN_{Phy_0} and $IN_{Phy_{100}}$ are two 2-D input vectors extracted from the train data set and each of them consists of 10000 rows (time steps). These vectors are identical, but shifted by 100 time steps i.e. $IN_{Phy_0}[k+100] = IN_{Phy_{100}}[k] \quad \forall k \in [0, 9900[$ as shown in Figure 3.10. Given the fact that the data belong to discharge cycle, the condition 3.8 should be verified, as the battery remain energy decreases with time:

$$E_{Phy_{100}}[k] < E_{Phy_0}[k] \quad \forall k \in [0, 9899] \quad (3.8)$$

From Equations 3.5 and 3.6,

$$Q.\Big(SOC_{Phy_{100}}[k]-SOC_{min}\Big).V_{Phy_{100}}[k] < Q.\Big(SOC_{Phy_0}[k]-SOC_{min}\Big).V_{Phy_0}[k] \quad \forall k \in [0,9899] \tag{3.9}$$

Supposing that the minimum SOC value is zero ($SOC_{min} = 0$), by substituting the SOC estimations in 3.9 by their expressions 3.3 and 3.4, we obtain the condition that should be verified for the CNN inferences denoted by $CNN\big(IN_{Phy_0}\big)$ and $CNN\big(IN_{Phy_{100}}\big)$:

$$Q.\Big(CNN\big(IN_{Phy_{100}}\big)[k]\Big).V_{Phy_{100}}[k] < Q.\Big(CNN\big(IN_{Phy_0}\big)[k]\Big).V_{Phy_0}[k] \quad \forall k \in [0,9899] \tag{3.10}$$

As we all know, the *ReLU* function, is defined by:

$$ReLU(x) = \begin{cases} x & \forall x > 0 \\ 0 & \forall x \leq 0 \end{cases} \tag{3.11}$$

The *ReLU* function in 3.2 returns a non-zero value for SOC predictions that violate the condition 3.10 i.e. when $E_{Phy_{100}}[k] > E_{Phy_0}[k]\ \forall k \in [0,9899]$. In this case, physical loss, regularized by λ parameter, is added to the total loss function to minimize physics inconsistency in the CNN inferences. Implementation of the combined loss function in *Python* programming language is given in Figure 3.8, and Figure 3.10 shows a visualization of the loss function from the model TensorBoard graph.

```
1
2  from keras import backend as K
3  import numpy as np
4  from keras.losses import mean_absolute_percentage_error
5
6  def combined_loss(parameters):
7      """
8      Computes the combined loss
9      Args:
10         parameters:    list containing the energy difference
11                        and the physical regularization constant
12                        lambda. energy_difference
13                        = compute_energy(CNN(IN_Phy100)) -
14                          compute_energy(CNN(IN_Phy0))
15     Returns:
16         combined loss = data loss + physical loss
17     """
18     energy_difference, lam = parameters
19     energy_difference_np   = np.array(energy_difference,
20                                       dtype=np.float32 )
21
22     def loss(y_true, y_pred):
23         return mean_absolute_percentage_error(y_true, y_pred) + \
24         lam * K.mean(K.relu(energy_difference_np))
25
26         return loss
```

Figure 3.8 Combined loss function implementation in Python programming language

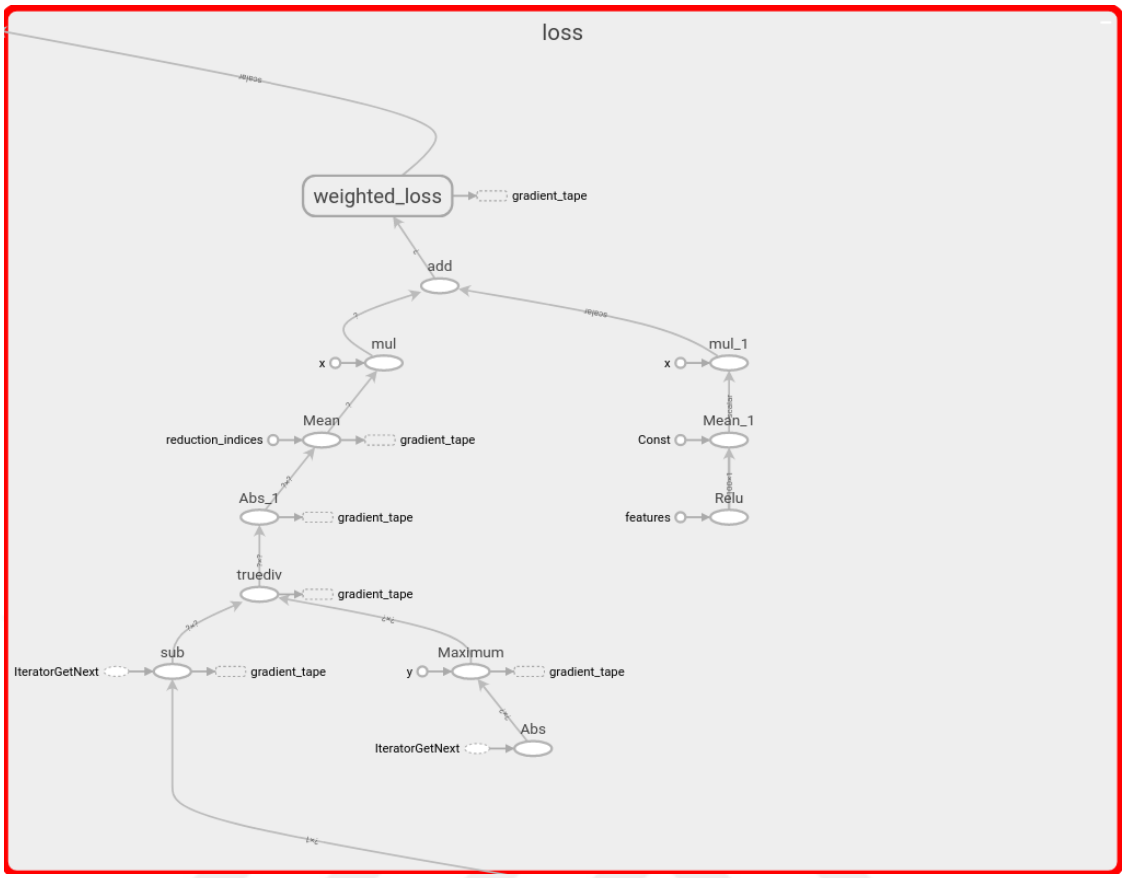


Figure 3.9 Visualization of the combined loss function in TensorBoard environment

The right branch of the loss function is the physical loss, and the left branch is the data loss. The two losses are added and the output is transferred to the model optimizer. To avoid dividing by zero, a small epsilon is added to the denominator of the MAPE function.

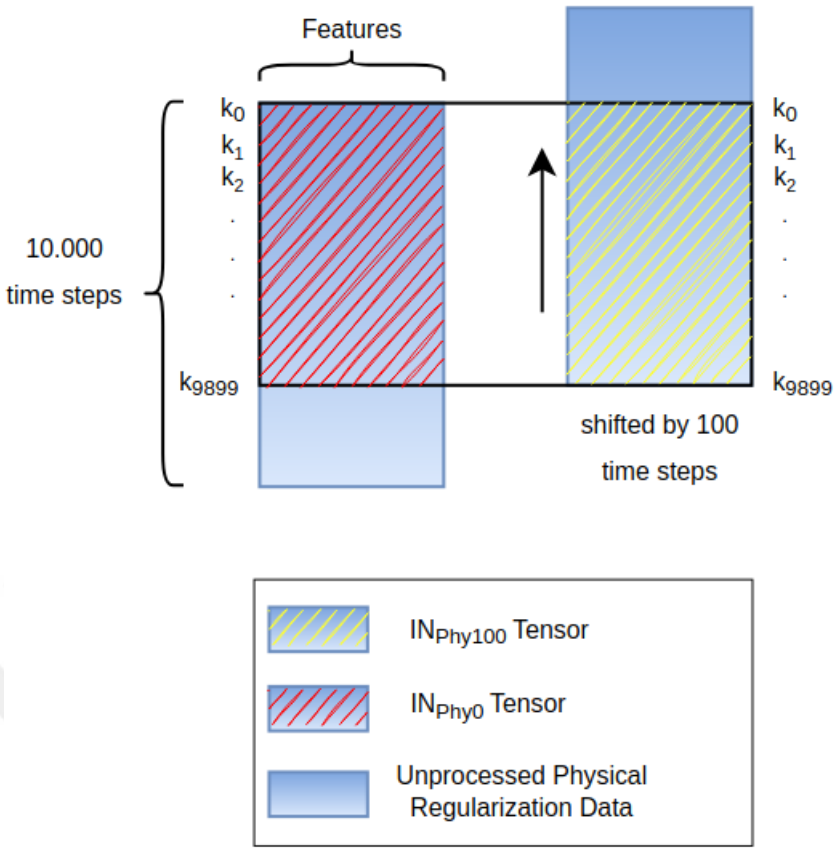


Figure 3.10 IN_{Phy_0} and $IN_{Phy_{100}}$ input tensors used for physical loss regularization

k indicates the time step in both tensors. The 2^{nd} tensor $IN_{Phy_{100}}$ is identical to the 1^{st} tensor IN_{Phy_0}, but with a 100-time-step shift. This means that $IN_{Phy_0}[k+100] = IN_{Phy_{100}}[k] \quad \forall k \in [0, 9899]$

3.4 Accuracy and Physics Consistency Test Results

PGCNN ($\lambda \neq 0$) and N-PGCNN ($\lambda = 0$) models were tested with different time windows and λ values. As stated before, the time window corresponds to the number of time steps i.e. data rows in the input tensors. λ rate regulates the amount of physical loss added to total cost function. The models were mainly evaluated on two metrics: MAPE and Physics Consistency Index (PCI). As discussed in the previous sections, the first metric, MAPE, expresses the rate of error to the ground truth and gives an idea on how significant the predictions errors are compared to the targets. In addition, the second metric, referred to in this work as PCI, was used as an indication of the physics consistency in the baseline and proposed models.

The procedure of the evaluation tests consists of feeding the tested model with 100 input batches, each contains 1000 input tensors with n time steps, where n is the number of samples in the time window. The models were tested with *large* and *small* time windows, containing 100 and 25 time steps, respectively. The test batches were randomly chosen from the test data set, which is separate from the train and validation data, and unused during the training phase, which is a necessary condition to conduct an accurate and unbiased test. Then, the MAPE and PCI results were recorded for each test to compare the accuracy and the physics consistency of each model.

Note that different λ values (0.1, 0.2, 0.4, 0.8, 1, 2, 5, and 10) were tested to figure out the most convenient physical regularization rate that delivers the best result. The λ value for the N-PGCNN models is zero, since there is no physical loss involved in the N-PGCNN cost function.

3.4.1 Accuracy Evaluation Using MAPE Metric

The MAPE is computed for each model using Equation 3.12, in which the *total inputs* is equal to *number of batches* $\times$ *batch size*, and SOC_{target_i} and SOC_{pred_i} denote the ground truth and SOC prediction for the i^{th} input, respectively.

$$MAPE(\%) = \frac{1}{\text{total inputs}} \sum_{i=0}^{\text{total inputs}} \left| \frac{SOC_{target_i} - SOC_{pred_i}}{SOC_{target_i}} \right| \times 100 \tag{3.12}$$

The MAPE metric, obviously, informs about the predictions accuracy. Lower prediction error to ground truth rate indicates higher accuracy, and vice versa. Generally speaking, other metrics can be also used to evaluate the accuracy of the NNs, however, in this work, we have chosen the $MAPE$ metric because it is convenient to the range of the SOC outputs.

The recorded MAPE results for the N-PGCNN and the PGCNN models are given in Table 3.7. Figures 3.11, 3.12, 3.13, and 3.14 show the SOC predictions and absolute errors for the different N-PGCNN and PGCNN models with the best λ values (λ rates that deliver the lowest MAPE),for random test data.

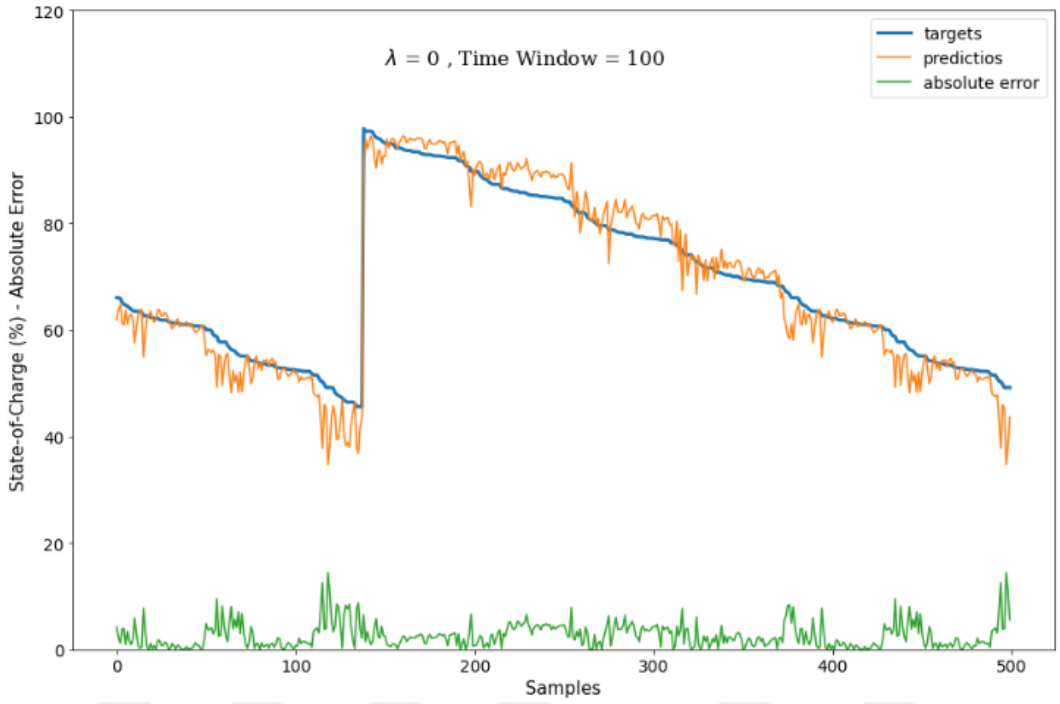


Figure 3.11 SOC predictions and absolute error of N-PGNN model ($\lambda = 0$, time window $= 100$) with random test data

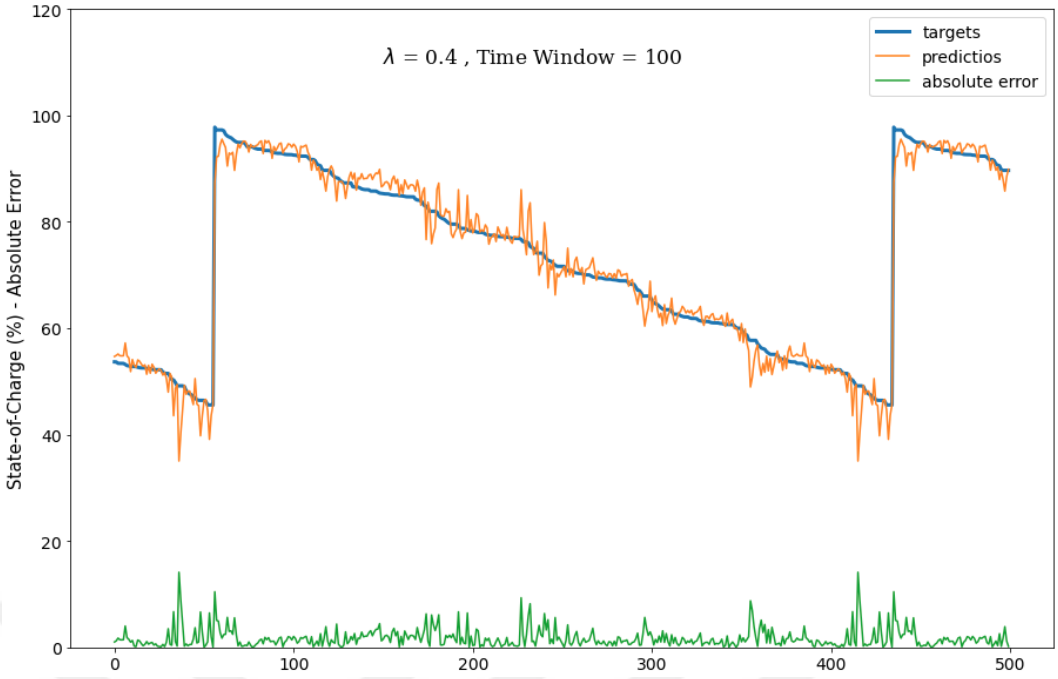


Figure 3.12 SOC predictions and absolute error of PGNN model ($\lambda = 0.4$, time window = 100) with random test data

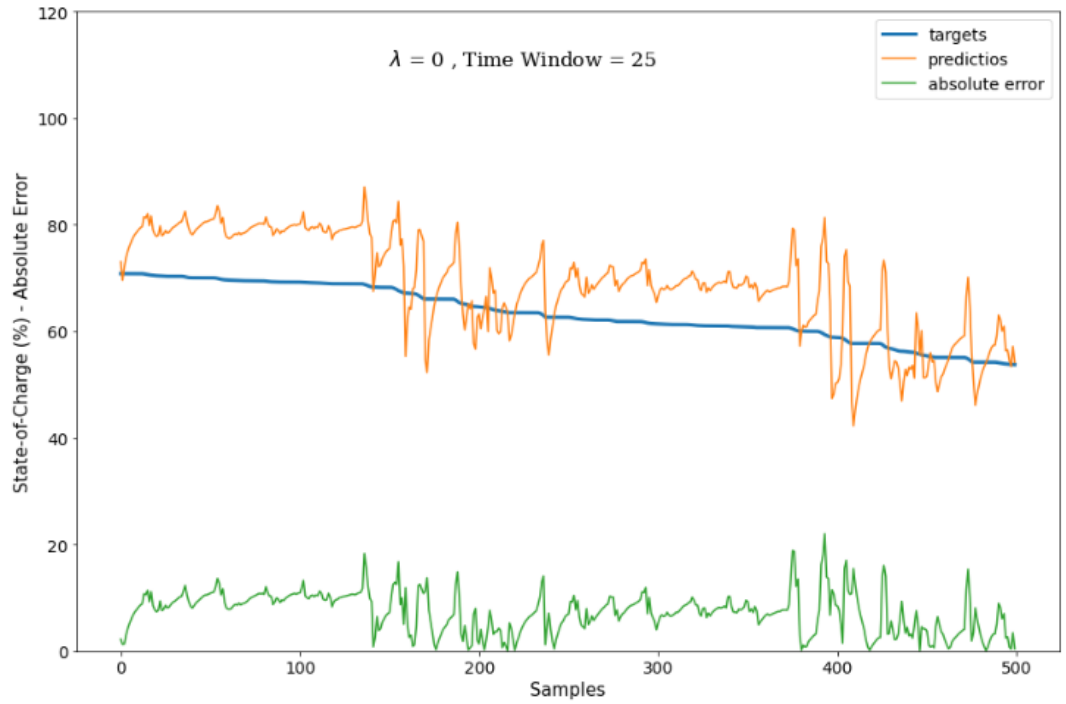


Figure 3.13 SOC predictions and absolute error of N-PGNN model ($\lambda = 0$, time window = 25) with random test data

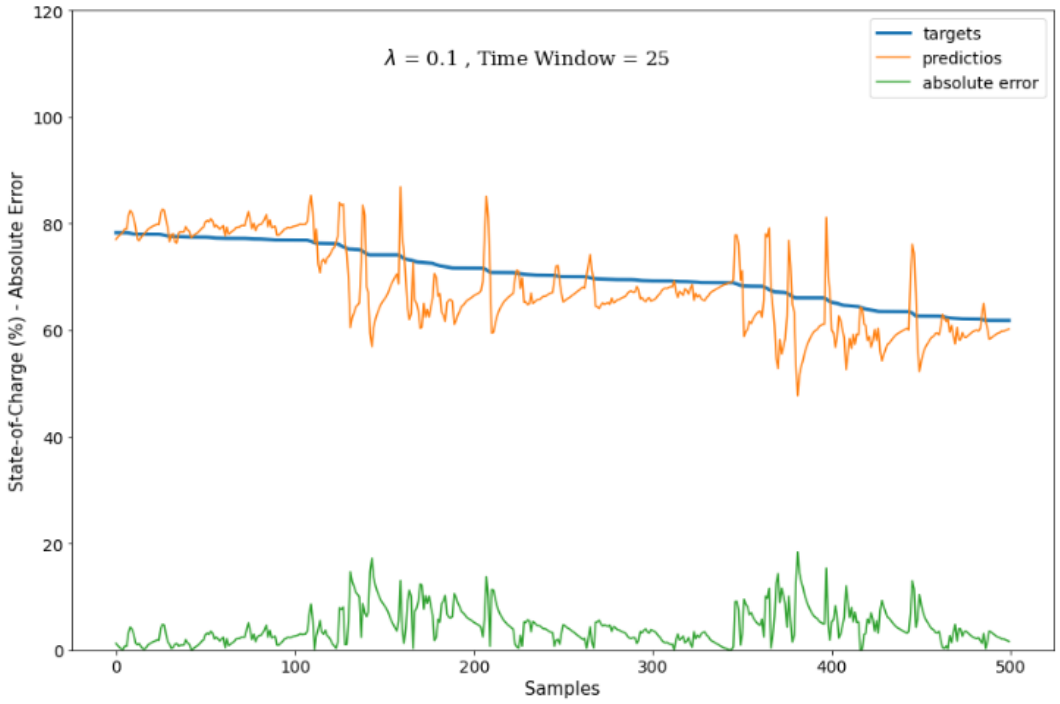


Figure 3.14 SOC predictions and absolute error of PGNN model ($\lambda = 0.1$, time window $= 25$) with random test data

As seen from Figure 3.11 and Figure 3.12, a higher prediction accuracy was achieved for the PGCNN model compared to the N-PGCNN with the large time window. This enhancement in the prediction accuracy was also achieved with small time window, as shown in Figure 3.13 and Figure 3.14.

Table 3.7 shows the MAPE test results for the N-PGCNN and PGCNN models. Overall, models with larger time window achieved lower MAPE results which means that increasing the input tensors time steps enhanced the performance of all models regardless of the introduction of physical knowledge into the learning process. The N-PGCNN model with large time window (100 time steps) scored a 4.395% lower MAPE compared with the small time-window (25 time-steps) N-PGNN model. This improvement was also recorded for all PGCNN models, with a 5.779% decrease in the MAPE, for the best PGCNN model ($\lambda = 0.4$).

The comparison of N-PGCNN and PGCNN models with the same time window shows that the regularization of the cost function with a physical loss led to better results in the model accuracy. For the large time window, the MAPE decreased from 3.990% to 2.547% on the test data set, for the regularization rate $\lambda = 0.4$. This enhancement was also observed with the small time window, from 8.385% to 5.399% MAPE, with $\lambda = 0.1$.

Table 3.7 N-PGCNN and PGCNN models MAPE test results

Models	Time Window (Time Steps)	λ	MAPE (%)
N-PGCNN	100	0	3.990
	25	0	8.385
PGCNN	100	0.1	2.805
		0.2	2.610
		0.4	2.547
		0.8	3.027
		1	3.018
		2	2.851
		5	4.227
		10	7.419
	25	0.1	5.399
		0.2	7.630
		0.4	8.326
		0.8	14.418
		1	23.380
		2	14.451
		5	41.606
		10	15.692

3.4.2 Physics Consistency Evaluation Using PCI Metric

In this work, *PCI* was used to evaluate the physics consistency of tested models. Generally, models that deliver SOC predictions that lead to increased energy in the battery during a discharge cycle are considered non consistent with physics and vice versa. The PCI (Equation 3.13) is used to quantify this statement and compare the physics consistency for the N-PGCNN and PGCNN models.

$$PCI = \frac{1}{\text{total batches}} \sum_{i=0}^{\text{total batches}} L_{Phy_i} \tag{3.13}$$

L_{Phy_i} in Equation 3.13 is the physical loss corresponding to the i^{th} input batch and defined by:

$$L_{Phy_i} = \frac{1}{\text{batch size}} \sum_{k=0}^{\text{batch size}} ReLU\big(E_{Phy_{100}}[k] - E_{Phy_0}[k]\big) \tag{3.14}$$

$E_{Phy_{100}}$ and E_{Phy_0} in Equation 3.14, are energy vectors computed from $SOC_{Phy_{100}}$ and SOC_{Phy_0} model predictions, respectively.

From Equations 3.13 and 3.14 we can infer that higher PCI value corresponds to lower physics consistency and vice versa. Table 3.8 provides a comparison of tested models

in term of PCI.

Table 3.8 N-PGCNN and PGCNN models PCI test results

Models	Time Window (Time Steps)	λ	PCI
N-PGCNN	100	0	0.133
	25	0	0.156
PGCNN	100	0.1	0.298
		0.2	0.088
		0.4	0.056
		0.8	0.225
		1	0.18
		2	0.178
		5	0.007
		10	0.027
	25	0.1	0.180
		0.2	0.061
		0.4	0.259
		0.8	0.047
		1	0.046
		2	0.105
		5	0.215
		10	0.181

As the results indicate, the proposed methodology achieved an improvement in the physics consistency translated in lower PCI results. The best 100-time-step PGCNN model ($\lambda = 5$) scored 0.007 PCI, compared to 0.133 for the N-PGCNN model with the same time window. In addition, the PCI metric shows the improvement in the physics consistency for the small time window models, from 0.156 to 0.046 for the N-PGCNN and and the best PGCNN ($\lambda = 1$) models, respectively.

4
RESULTS AND DISCUSSION

In this study, a novel technique for SOC estimation is proposed. The proposed approach enhances the DD NN based SOC estimation methods. These methods are explained in Section 1.1.2.4.

The main concept is to design and train a CNN model with physics constraints based on an energy approach for Li-ion batteries. Physics knowledge is incorporated into the learning process of the model by designing a hybrid cost function that combines physical and data losses, to improve the classic NN based DD paradigm in both accuracy and physics consistency.

A publicly available data of Panasonic 18650PF Lithium-ion battery is utilized to validate the new approach by training and testing physics guided and non-physics guided models, in this work, referred to as PGCNN and N-PGCNN, respectively, with different time windows. The common CNN architecture that was used for the proposed (PGCNN) and the baseline (N-PGCNN) models is given in Section 3.2. The incorporation of physics knowledge during the learning process is explained in Section 3.3.

The evaluation of the models was conducted based on MAPE and PCI metrics. Results, represented in 3.4, showed that the PGCNN outperformed the N-PGCNN in both accuracy and physics consistency for large and small time windows.

The comparison of N-PGCNN and PGCNN models with the same time window shows that the regularization of the cost function with a physical loss led to better results in both accuracy and physics consistency. Different physical regularization rates λ were used to train different PGCNN models. The best MAPE and PCI results were obtained with different models. However, the best PGCNN models that enhanced both accuracy and physics consistency at the same time is obtained with $\lambda = 0.4$ and $\lambda = 0.2$, for the large and small time windows, respectively.

With these models, for the large time window, the MAPE decreased from 3.990% to

2.547% on the test data set, with a regularization rate $\lambda = 0.4$. This enhancement was also observed with the small time window, from 8.385% to 7.630% MAPE, with $\lambda = 0.2$.

The proposed methodology also achieved an improvement in the physics consistency translated in lower PCI results. The chosen 100-time-step PGCNN model ($\lambda = 0.4$)scored 0.056 in PCI, compared to 0.133 for the N-PGCNN model with the same time window. In addition, the PCI metric shows and improvement in the physics consistency for the small-time-window from 0.156 to 0.061 for the N-PGCNN and chosen PGCNN ($\lambda = 0.2$)models, respectively. These results validate the proposed approach as a surrogate paradigm for NN based DD SOC estimation.

This work proves that DD methods, for instance NNs, can deliver better results when used together with mathematical and physical approaches depending on the problem being solved by these models. The combination of data and physics knowledge can be realized in different ways, in particular customizing the loss function, as shown in this work.

Future works may include the application of the proposed methodology to different DD architectures, and the deployment of these models to real-time battery management systems.

REFERENCES

[1] P. Kollmeyer, *Panasonic 18650pf li-ion battery data*, Jun. 2018. DOI: 10.17632/wykht8y7tg.1. [Online]. Available: https://data.mendeley.com/datasets/wykht8y7tg/1.

[2] Z. C. Nxumalo, P. Tarwireyi, M. O. Adigun, "Lithium-based batteries in tactical military applications: A review," in *2015 International Conference on Computer, Communications, and Control Technology (I4CT)*, IEEE, 2015, pp. 575–579.

[3] G. L. Plett, *Battery management systems, Volume II: Equivalent-circuit methods*. Artech House, 2015.

[4] P. Zhang, *Advanced industrial control technology*. William Andrew, 2010.

[5] A. Karpatne, W. Watkins, J. Read, V. Kumar, "Physics-guided neural networks (pgnn): An application in lake temperature modeling," *arXiv preprint arXiv:1710.11431*, 2017.

[6] J. Wang, Y. Li, R. Zhao, R. X. Gao, "Physics guided neural network for machining tool wear prediction," *Journal of Manufacturing Systems*, vol. 57, pp. 298–310, 2020.

[7] X. Yang, S. Zafar, J.-X. Wang, H. Xiao, "Predictive large-eddy-simulation wall modeling via physics-informed neural networks," *Physical Review Fluids*, vol. 4, no. 3, p. 034 602, 2019.

[8] K. Kashinath, P. Marcus, *et al.*, "Enforcing physical constraints in cnns through differentiable pde layer," in *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020.

[9] J.-L. Wu, K. Kashinath, A. Albert, D. Chirila, H. Xiao, *et al.*, "Enforcing statistical constraints in generative adversarial networks for modeling chaotic dynamical systems," *Journal of Computational Physics*, vol. 406, p. 109 209, 2020.

[10] R. Wang, R. Walters, R. Yu, "Incorporating symmetry into deep dynamics models for improved generalization," *arXiv preprint arXiv:2002.03061*, 2020.

[11] R. Wang, K. Kashinath, M. Mustafa, A. Albert, R. Yu, "Towards physics-informed deep learning for turbulent flow prediction," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 1457–1466.

[12] D. Wu *et al.*, "Deepgleam: A hybrid mechanistic and deep learning model for covid-19 forecasting," *arXiv preprint arXiv:2102.06684*, 2021.

[13] Y. Li, R. Yu, C. Shahabi, Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," *arXiv preprint arXiv:1707.01926*, 2017.

[14] A. Cogliati, "The noether theorems. invariance and conservation laws in the twentieth century by yvette kosmann-schwarzbach. translated by bertram e. schwarzbach. dordrecht, heidelberg, london, new york (springer). 2011. isbn 978-0-387-87868-6," 2012.

[15] R. Zhang, Y. Liu, H. Sun, "Physics-guided convolutional neural network (phycnn) for data-driven seismic response modeling," *Engineering Structures*, vol. 215, p. 110 704, 2020.

[16] X. Hu, H. Hu, S. Verma, Z.-L. Zhang, "Physics-guided deep neural networks for power flow analysis," *IEEE Transactions on Power Systems*, vol. 36, no. 3, pp. 2082–2092, 2020.

[17] S. Pawar, O. San, B. Aksoylu, A. Rasheed, T. Kvamsdal, "Physics guided machine learning using simplified theories," *Physics of Fluids*, vol. 33, no. 1, p. 011 701, 2021.

[18] Y. Yu, H. Yao, Y. Liu, "Structural dynamics simulation using a novel physics-guided machine learning method," *Engineering Applications of Artificial Intelligence*, vol. 96, p. 103 947, 2020.

[19] J. N. Kani, A. H. Elsheikh, "Dr-rnn: A deep residual recurrent neural network for model reduction," *arXiv preprint arXiv:1709.00939*, 2017.

[20] J. Chen, Y. Liu, "Probabilistic physics-guided machine learning for fatigue data analysis," *Expert Systems with Applications*, vol. 168, p. 114 316, 2021.

[21] F. Engineering. "Description of a s-n curve." (), [Online]. Available: `https://www.fatec-engineering.com/2018/02/20/description-of-a-s-n-curve/` (visited on 01/28/2022).

[22] F. G. Pascual, W. Q. Meeker, "Estimating fatigue curves with the random fatigue-limit model," *Technometrics*, vol. 41, no. 4, pp. 277–289, 1999.

[23] M. A. Hannan, M. S. H. Lipu, A. Hussain, M. H. Saad, A. Ayob, "Neural network approach for estimating state of charge of lithium-ion battery using backtracking search algorithm," *Ieee Access*, vol. 6, pp. 10 069–10 079, 2018.

[24] X. Wu, X. Li, J. Du, "State of charge estimation of lithium-ion batteries over wide temperature range using unscented kalman filter," *Ieee Access*, vol. 6, pp. 41 993–42 003, 2018.

[25] M. H. Lipu, A. Hussain, M. Saad, A. Ayob, M. Hannan, "Improved recurrent narx neural network model for state of charge estimation of lithium-ion battery using pso algorithm," in *2018 IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE)*, IEEE, 2018, pp. 354–359.

[26] D. N. How, M. Hannan, M. H. Lipu, P. J. Ker, "State of charge estimation for lithium-ion batteries using model-based and data-driven methods: A review," *Ieee Access*, vol. 7, pp. 136 116–136 136, 2019.

[27] G. L. Plett, *Battery management systems, Volume I: Battery modeling*. Artech House, 2015.

[28] S. Yin, X. Li, H. Gao, O. Kaynak, "Data-based techniques focused on modern industry: An overview," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 1, pp. 657–667, 2014.

[29] S. Yin, S. X. Ding, X. Xie, H. Luo, "A review on basic data-driven approaches for industrial process monitoring," *IEEE Transactions on Industrial Electronics*, vol. 61, no. 11, pp. 6418–6428, 2014.

[30] M. Doyle, T. F. Fuller, J. Newman, "Modeling of galvanostatic charge and discharge of the lithium/polymer/insertion cell," *Journal of the Electrochemical society*, vol. 140, no. 6, p. 1526, 1993.

[31] H. He, R. Xiong, J. Fan, "Evaluation of lithium-ion battery equivalent circuit models for state of charge estimation by an experimental approach," *energies*, vol. 4, no. 4, pp. 582–598, 2011.

[32] Y. Hu, S. Yurkovich, Y. Guezennec, B. Yurkovich, "Electro-thermal battery model identification for automotive applications," *Journal of Power Sources*, vol. 196, no. 1, pp. 449–457, 2011.

[33] J. Xu, C. C. Mi, B. Cao, J. Cao, "A new method to estimate the state of charge of lithium-ion batteries based on the battery impedance model," *Journal of power sources*, vol. 233, pp. 277–284, 2013.

[34] Z. Chen, X. Li, J. Shen, W. Yan, R. Xiao, "A novel state of charge estimation algorithm for lithium-ion battery packs of electric vehicles," *Energies*, vol. 9, no. 9, p. 710, 2016.

[35] W. Wang, J. Mu, "State of charge estimation for lithium-ion battery in electric vehicle based on kalman filter considering model error," *Ieee Access*, vol. 7, pp. 29 223–29 235, 2019.

[36] S. Peng, C. Chen, H. Shi, Z. Yao, "State of charge estimation of battery energy storage systems based on adaptive unscented kalman filter with a noise statistics estimator," *Ieee Access*, vol. 5, pp. 13 202–13 212, 2017.

[37] J. Wang, L. Zhang, J. Mao, J. Zhou, D. Xu, "Fractional order equivalent circuit model and soc estimation of supercapacitors for use in hess," *IEEE Access*, vol. 7, pp. 52 565–52 572, 2019.

[38] D. Zhou, K. Zhang, A. Ravey, F. Gao, A. Miraoui, "Online estimation of lithium polymer batteries state-of-charge using particle filter-based data fusion with multimodels approach," *IEEE Transactions on Industry Applications*, vol. 52, no. 3, pp. 2582–2595, 2016.

[39] S. Schwunk, N. Armbruster, S. Straub, J. Kehl, M. Vetter, "Particle filter for state of charge and state of health estimation for lithium–iron phosphate batteries," *Journal of Power Sources*, vol. 239, pp. 705–710, 2013.

[40] Q. Yu, R. Xiong, C. Lin, "Online estimation of state-of-charge based on the h infinity and unscented kalman filters for lithium ion batteries," *Energy Procedia*, vol. 105, pp. 2791–2796, 2017.

[41] C. Chen, R. Xiong, W. Shen, "A lithium-ion battery-in-the-loop approach to test and validate multiscale dual h infinity filters for state-of-charge and capacity estimation," *IEEE Transactions on power Electronics*, vol. 33, no. 1, pp. 332–342, 2017.

[42] C. Lin, H. Mu, R. Xiong, W. Shen, "A novel multi-model probability battery state of charge estimation approach for electric vehicles using h-infinity algorithm," *Applied Energy*, vol. 166, pp. 76–83, 2016.

[43] B. Wang, Z. Liu, S. E. Li, S. J. Moura, H. Peng, "State-of-charge estimation for lithium-ion batteries based on a nonlinear fractional model," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 1, pp. 3–11, 2016.

[44] S. Dey, B. Ayalew, P. Pisu, "Nonlinear robust observers for state-of-charge estimation of lithium-ion cells based on a reduced electrochemical model," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 5, pp. 1935–1942, 2015.

[45] L. Zheng, L. Zhang, J. Zhu, G. Wang, J. Jiang, "Co-estimation of state-of-charge, capacity and resistance for lithium-ion batteries based on a high-fidelity electrochemical model," *Applied Energy*, vol. 180, pp. 424–434, 2016.

[46] J. Xu, C. C. Mi, B. Cao, J. Deng, Z. Chen, S. Li, "The state of charge estimation of lithium-ion batteries based on a proportional-integral observer," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 4, pp. 1614–1621, 2013.

[47] B. Xiong, J. Zhao, Y. Su, Z. Wei, M. Skyllas-Kazacos, "State of charge estimation of vanadium redox flow battery based on sliding mode observer and dynamic model including capacity fading factor," *IEEE Transactions on Sustainable Energy*, vol. 8, no. 4, pp. 1658–1667, 2017.

[48] J. Du, Z. Liu, Y. Wang, C. Wen, "An adaptive sliding mode observer for lithium-ion battery state of charge and state of health estimation in electric vehicles," *Control Engineering Practice*, vol. 54, pp. 81–90, 2016.

[49] S. Tong, J. H. Lacap, J. W. Park, "Battery state of charge estimation using a load-classifying neural network," *Journal of Energy Storage*, vol. 7, pp. 236–243, 2016.

[50] E. Chemali, P. J. Kollmeyer, M. Preindl, A. Emadi, "State-of-charge estimation of li-ion batteries using deep neural networks: A machine learning approach," *Journal of Power Sources*, vol. 400, pp. 242–255, 2018.

[51] F. Yang, X. Song, F. Xu, K.-L. Tsui, "State-of-charge estimation of lithium-ion batteries via long short-term memory network," *Ieee Access*, vol. 7, pp. 53 792–53 799, 2019.

[52] B. Xiao, Y. Liu, B. Xiao, "Accurate state-of-charge estimation approach for lithium-ion batteries by gated recurrent unit with ensemble optimizer," *Ieee Access*, vol. 7, pp. 54 192–54 202, 2019.

[53] T. Hansen, C.-J. Wang, "Support vector based battery state of charge estimator," *Journal of Power Sources*, vol. 141, no. 2, pp. 351–358, 2005.

[54] J. Hu *et al.*, "State-of-charge estimation for battery management system using optimized support vector machine for regression," *Journal of Power Sources*, vol. 269, pp. 682–693, 2014.

[55] J. C. Álvarez Antón, P. J. García Nieto, C. Blanco Viejo, J. A. Vilán Vilán, "Support vector machines used to estimate the battery state of charge," *IEEE Transactions on Power Electronics*, vol. 28, no. 12, pp. 5919–5926, 2013. DOI: 10.1109/TPEL.2013.2243918.

[56] J. Meng, G. Luo, F. Gao, "Lithium polymer battery state-of-charge estimation based on adaptive unscented kalman filter and support vector machine," *IEEE Transactions on Power Electronics*, vol. 31, no. 3, pp. 2226–2238, 2016. DOI: 10.1109/TPEL.2015.2439578.

[57] V. Novák, I. Perfilieva, J. Močkoř, "Functional systems in fuzzy logic theories," in *Mathematical Principles of Fuzzy Logic*, Springer, 1999, pp. 179–222.

[58] A. J. Salkind, C. Fennie, P. Singh, T. Atwater, D. E. Reisner, "Determination of state-of-charge and state-of-health of batteries by fuzzy logic methodology," *Journal of Power sources*, vol. 80, no. 1-2, pp. 293–300, 1999.

[59] H. Sheng, J. Xiao, "Electric vehicle state of charge estimation: Nonlinear correlation and fuzzy support vector machine," *Journal of Power sources*, vol. 281, pp. 131–137, 2015.

[60] M. Anwar *et al.*, "State of charge monitoring system of electric vehicle using fuzzy logic," in *2018 International Conference on Sustainable Energy Engineering and Application (ICSEEA)*, 2018, pp. 34–38. DOI: 10.1109/ICSEEA.2018.8627131.

[61] T. Zahid, K. Xu, W. Li, C. Li, H. Li, "State of charge estimation for electric vehicle power battery using advanced machine learning algorithm under diversified drive cycles," *Energy*, vol. 162, pp. 871–882, 2018.

[62] A. Densmore, M. Hanif, "Determining battery soc using electrochemical impedance spectroscopy and the extreme learning machine," in *2015 IEEE 2nd International Future Energy Electronics Conference (IFEEC)*, IEEE, 2015, pp. 1–7.

[63] R. Xiong, Q. Yu, *et al.*, "Open circuit voltage and state of charge online estimation for lithium ion batteries," *Energy Procedia*, vol. 142, pp. 1902–1907, 2017.

[64] M.-W. Cheng, Y.-S. Lee, M. Liu, C.-C. Sun, "State-of-charge estimation with aging effect and correction for lithium-ion battery," *IET Electrical Systems in Transportation*, vol. 5, no. 2, pp. 70–76, 2015.

[65] Y. Xing, W. He, M. Pecht, K. L. Tsui, "State of charge estimation of lithium-ion batteries using the open-circuit voltage at various ambient temperatures," *Applied Energy*, vol. 113, pp. 106–115, 2014.

[66] K. Kaushik, D. Sureka, H. Rao, "Review of battery state-of-charge estimation algorithms," in *Advances in Renewable Energy and Electric Vehicles*, Springer, 2022, pp. 375–387.

[67] F. Feng, R. Lu, C. Zhu, "A combined state of charge estimation method for lithium-ion batteries used in a wide ambient temperature range," *Energies*, vol. 7, no. 5, pp. 3004–3032, 2014.

[68] J. Xie, J. Ma, K. Bai, "Enhanced coulomb counting method for state-of-charge estimation of lithium-ion batteries based on peukert's law and coulombic efficiency," *Journal of Power Electronics*, vol. 18, no. 3, pp. 910–922, 2018.

[69] L. He, D. Guo, "An improved coulomb counting approach based on numerical iteration for soc estimation with real-time error correction ability," *IEEE Access*, vol. 7, pp. 74 274–74 282, 2019.

[70] I. Baccouche, S. Jemmali, A. Mlayah, B. Manai, N. E. B. Amara, "Implementation of an improved coulomb-counting algorithm based on a piecewise soc-ocv relationship for soc estimation of li-ionbattery," *arXiv preprint arXiv:1803.10654*, 2018.

[71] S. U. CS231. "Cs232n convolutional neural networks for visual recognition." (), [Online]. Available: https://cs231n.github.io/convolutional-networks/ (visited on 02/13/2022).

[72] S. Ioffe, C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, PMLR, 2015, pp. 448–456.

[73] M. Z. Mulla. "Cost, activation, loss function|| neural network|| deep learning. what are these?" (), [Online]. Available: https://medium.com/@zeeshanmulla/cost-activation-loss-function-neural-network-deep-learning-what-are-these-91167825a4de (visited on 05/20/2022).

[74] D. P. Kingma, J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[75] V. Bushaev. "Adam — latest trends in deep learning optimization." (), [Online]. Available: https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c (visited on 03/06/2022).

[76] F. Chollet *et al.* "Keras." (2015), [Online]. Available: https://github.com/fchollet/keras.

[77] M. H. Vishkasougheh, "Characterization of a li-ion battery based stand-alone a-si photovoltaic system," M.S. thesis, Istanbul Sehir University, 2014, p. 32. [Online]. Available: https://acikbilim.yok.gov.tr/bitstream/handle/20.500.12812/631656/yokAcikBilim_10032035.pdf?sequence=-1&isAllowed=y.

[78] *Lithium cell tester mct series*, Digatron. [Online]. Available: https://www.digatron.com/Portals/38/Images/documents/PRODUKTBLATT_MCT_EN.PDF?ver=2019-06-12-091043-723&timestamp=1561618385876 (visited on 01/15/2022).

[79] Digatron. "Digatron mct-me." (), [Online]. Available: https://www.digatron.com/en-us/Products/Laboratory/Cycler/Cell-Tester/MCT-ME (visited on 01/15/2022).

[80] *Lithium cell tester mct series*, Panasonic. [Online]. Available: https://actec.dk/media/documents/70FC46554038.pdf (visited on 01/15/2022).

[81] E. Chemali, P. J. Kollmeyer, M. Preindl, R. Ahmed, A. Emadi, "Long short-term memory networks for accurate state-of-charge estimation of li-ion batteries," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 8, pp. 6730–6739, 2017.

[82] U. S. E. P. Agency. "Graphic review of driving schedules." (), [Online]. Available: https://www.epa.gov/vehicle-and-fuel-emissions-testing/dynamometer-drive-schedules (visited on 01/16/2022).

[83] Martín Abadi *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: https://www.tensorflow.org/.

PUBLICATIONS FROM THE THESIS

Conference Papers

1. Fedi Salhi, Erhan Akdogan, "Physics Guided Neural Network Based State-of-Charge Estimator for Lithium-Ion Batteries", 1^{st} International Congress on Modern Sciences, Tashkent (May 2022)