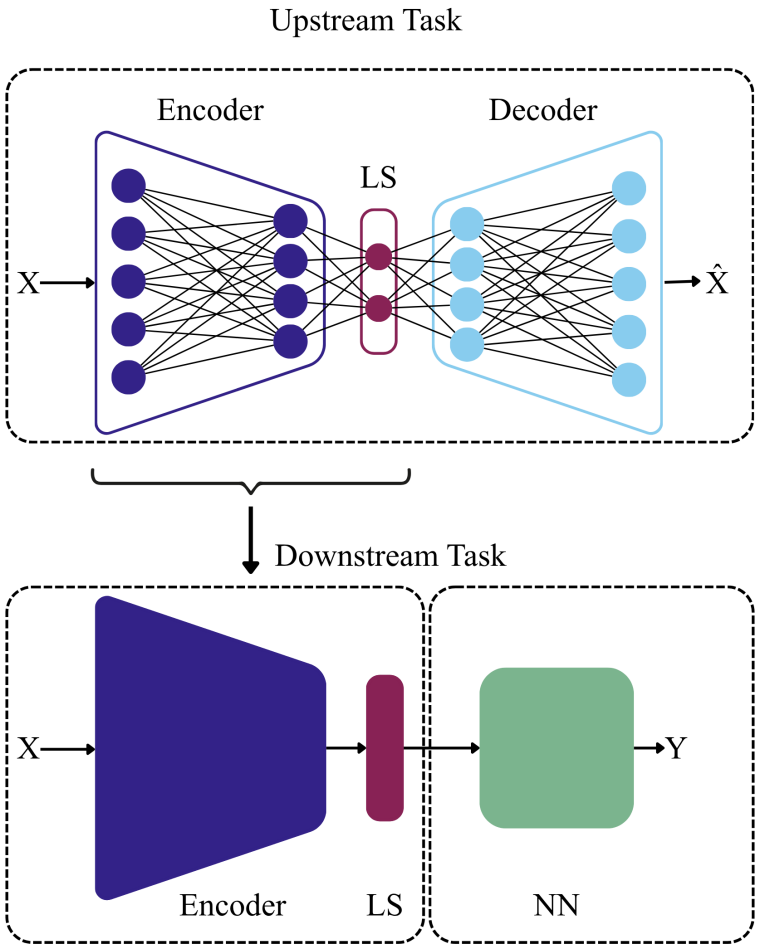
CHALMERS
UNIVERSITY OF TECHNOLOGY



Unlocking Unlabeled Battery Data

Semi-Supervised and Self-Supervised Deep Learning Approaches for SoH Estimation

Master's thesis in Complex Adaptive Systems

ELSA ARTE & ALVA LINDSTRÖM

DEPARTMENT OF PHYSICS
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

Master's thesis 2026

Unlocking Unlabeled Battery Data

Semi-Supervised and Self-Supervised Deep Learning Approaches for SoH Estimation

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Physics
Chalmers University of Technology
Gothenburg, Sweden 2026

Unlocking Unlabeled Battery Data
Semi-Supervised and Self-Supervised Deep Learning Approaches for SoH Estimation
ELSA ARTE & ALVA LINDSTRÖM

Supervisors: Christian Fleischer, Cognivity AI
Anders Hellman, Department of Physics
Henrik Klein Moberg, Department of Physics

Examiner: Anders Hellman, Department of Physics

Master's Thesis 2026
Department of Physics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Model of a hybrid architecture with an upstream and a downstream task.

Typeset in LaTeX
Gothenburg, Sweden 2026

Unlocking Unlabeled Battery Data
Semi-Supervised and Self-Supervised Deep Learning Approaches for SoH Estimation
ELSA ARTE & ALVA LINDSTRÖM
Department of Physics
Chalmers University of Technology

Abstract

The transport sector is expanding rapidly and with it the use of lithium-ion batteries. Accurately estimating a battery's State of Health (SoH) is crucial to ensure optimal usage and in turn optimize both the safety and the environmental impact of the battery. Traditional machine learning relies on labeled data, which is limited in the field of SoH estimation. Furthermore, the field holds a significant gap between laboratory and field data. This project aims to investigate how self-supervised and semi-supervised machine learning methods can be used for SoH estimation. This is done through upstream self-supervised learning of unlabeled data using autoencoders, and downstream supervised estimation of SoH. Furthermore, a multi-chemistry battery dataset containing both field and laboratory data was curated. To further bridge the gap between field and laboratory data a fine-tuning task was performed. The proposed semi-supervised hybrid architecture consists of a convolutional autoencoder combined with an LSTM head. This resulted in SoH predictions with an MAE of 0.0196 and an RMSE of 0.0313. The amount of labeled training data could be significantly reduced whilst maintaining accurate results. The model was successfully fine-tuned to produce solid early predictions for SoH on the field data. This project provides a machine learning pipeline which both lowers the need for labeled data and helps bridge the laboratory-field gap in SoH estimation. Future work with a focus on data engineering and acquiring higher quality field data could further build on these results.

Keywords: State of Health, Lithium-ion Batteries, Battery Degradation, Machine Learning, Self-supervised Learning, Semi-supervised Learning.

Acknowledgements

We would like to express our gratitude to everyone who has assisted us throughout this project. Thank you to the Cognivity AI team for your hospitality and for creating a collaborative working environment. An extra big thank you to Christian Fleischer and Anna Rohova for your invaluable expertise.

We would also like to thank our examiner Anders Hellman and our academic supervisor Henrik Klein Moberg for excellent support and guidance throughout the project.

Elsa Arte & Alva Lindström, Gothenburg, June 2026

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

AE	Autoencoder
CC	Constant Current
CEEMDAN	Complete Ensemble Empirical Mode Decomposition with Adaptive Noise
CNN	Convolutional Neural Network
DD	Dynamic Discharging
EoL	End of Life
EV	Electric Vehicle
GRU	Gated Recurrent Unit
IMF	Intrinsic Mode Function
LFP	Lithium Iron Phosphate
LiB	Lithium-ion Battery
LOWESS	Locally Weighted Scatterplot Smoothing
LSTM	Long-Short-Term-Memory
MLP	Multi Layer Perceptron
MSE	Mean Squared Error
NCA	Lithium Nickel Cobalt Aluminum Oxide
NMC	Lithium Nickel Manganese Cobalt Oxide
PINN	Physics Informed Neural Network
RC	Random Cycling
RNN	Recurrent Neural Network
RPT	Reference Performance Testing
RUL	Remaining Useful Life
SEI	Solid Electrolyte Interface
SoC	State of Charge
SoH	State of Health

Contents

List of Figures

List of Tables

1
Introduction

In the coming years, the electrification of the transport sector is expected to continue, causing the demand for Electric Vehicles (EVs) to grow with it. The World Energy Outlook from 2025 estimates the demand for electric vehicle batteries to more than triple by 2030, compared to 2024 [1]. Much due to their superior energy density, Lithium-ion Batteries (LiB) are the leading battery type in the EV market [2]. LiBs are also popular in a wide range of other sectors, including portable electronics and grid-scale renewable energy integration and storage [3].

Though electric vehicles are generally considered more environmentally friendly than vehicles with internal combustion engines, the production and disposal of lithium-ion batteries still affect the environment negatively [3]. Therefore, a battery's lifespan is crucial to its environmental impact and optimizing it is of utmost importance. Furthermore, a faulty or mishandled battery can, in the worst case, result in accidents and hence be a safety hazard [4]. This further emphasizes the need for proper battery handling and maintenance. In order for users to be able to perform their maintenance and general battery handling in an optimal way to ensure safety and effectiveness, reliable predictions of State of Health (SoH), State of Charge (SoC) and Remaining Useful Life (RUL) are key [5]. Batteries that have reached their end-of-life can still be used for second hand tasks such as energy storage power stations, further emphasizing the need for accurate SoH estimations [6].

Battery aging depends on many different factors, including (but not limited to) ambient temperature, charging profiles and structural differences [3]. Therefore, estimating a battery's health through SoC, SoH and RUL is not a simple task. Attempts to predict these metrics have been made before with varying results. Wang et al. [6] tackle the problem of SoH estimation by developing a Physics-Informed Neural Network (PINN) in order to combine the physical principles at play during battery degradation with machine learning. Furthermore, battery degradation data often has limited ground-truth labels for SoH and RUL. Yunhong et al. [7] therefore propose a self-supervised framework where an autoencoder is used for pretext learning by reconstructing partial current-voltage curves. For the downstream task, the pretrained encoder is then fine-tuned to estimate SoH using only a few labeled data points. Wang et al. [8] introduce a different self-supervised framework that instead uses charging capacity as weak labels. Zhang et al. [9] handle the limited availability of RUL labels by performing label propagation using KNN clustering to predict RUL.

There are plenty more studies that have been made on this topic. Though previous work shows great promise in estimating SoC, SoH and RUL, a majority of studies achieve their results using lab-generated data. This data is not fully representative of a real-world scenario since the batteries operate in a controlled environment and are available for measurements at all times. The data that can be acquired from actual EV-batteries in traffic is much more irregular and noisy. Furthermore, real-world data often lacks sufficient SoH and RUL labeling, making traditional machine learning difficult.

The novelty of this project lies in creating a semi-supervised pipeline for SoH prediction. The pipeline allows for a reduced labeling ratio for SoH prediction until End of Life (EoL). Futhermore, an early SoH prediction is performed using fine-tuning to enable the use of field and lab data interchangeably. The model is trained on a multi-chemistry dataset combining several different cycling protocols.

1.1 Aim

This project aims to evaluate how self-supervised and semi-supervised learning can be combined for State of Health (SoH) estimation. Specifically, this will be done by creating a pipeline consisting of dataset selection and data preprocessing, a self-supervised upstream task and a semi-supervised downstream task, as well as evaluation methods with a focus on stability and accuracy. The upstream and downstream architectures will be selected by evaluating combinations of different architectures. Furthermore, the model will be tested on both lab and field data to evaluate its applicability in a real-world scenario with noisy data and limited labeling.

1.2 Research Questions

To achieve this aim, the project will answer the following research questions:

1. How can self-supervised and semi-supervised learning methods be combined to create an SoH estimation pipeline for battery degradation data?
   (a) How does the proposed pipeline perform when using different upstream and downstream models, and which is the optimal combination?
   (b) How does the model perform when tested on field data compared to lab data, and how does it react to noise?
   (c) How much can the labeling ratio be minimized whilst still ensuring model performance when using the proposed pipeline?

2

Theory

In this chapter, theory relevant to the thesis is presented and explained.

2.1 Batteries

In this section, lithium-ion batteries and their degradation is explained.

2.1.1 Battery Chemistry

Figure 2.1 shows a simple model of a LiB during the discharge process. The battery is composed of a positive cathode, a negative anode, electrolytes and a separator. The main driving force of a lithium-ion battery is the exchange of lithium ions (Li^+), between the anode and the cathode. During discharge the anode releases Li^+ which then migrates through the battery to the cathode. The porous separator allows the Li^+ to pass smoothly whilst still isolating the two sides to prevent short circuits. At the same time, electrons travel through an external circuit from anode to cathode. The charging of a battery follows the same pattern but in reverse, following an external potential difference being placed on both anode and cathode [10].

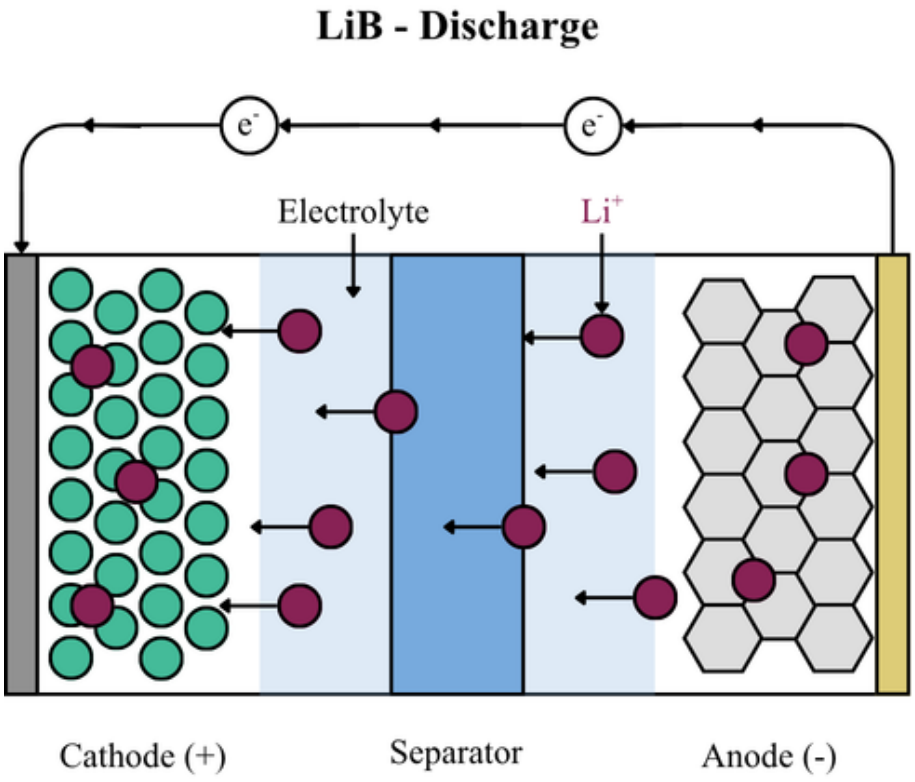


Figure 2.1: A model of a lithium-ion battery during the discharge process.

LiBs can have many different chemical compositions, which affect how the batteries behave, and therefore, what the degradation of the battery look like. The most important of which is the cathode material, which has a large influence on the battery energy density [11]. Therefore, lithium-ion batteries are often categorized by their cathode chemistry. Commonly used cathode chemistries include: Lithium Nickel Cobalt Aluminum Oxide (NCA), Lithium Nickel Manganese Cobalt Oxide (NMC) and Lithium Iron Phosphate (LFP). The anode material of a lithium-ion battery can also vary, but it is often graphite.

2.1.2 Battery Degradation

The aging of a LiB is largely due to irreversible chemical processes within the battery. These processes affect the battery's capacity and internal resistance through for example the growth of Solid Electrolyte Interface (SEI), lithium plating and the loss of active materials [12].

One main metric used when assessing battery degradation is SoH [13]. SoH represents the maximum amount of capacity that the battery is capable of, given its current state, compared to its nominal capacity. SoH is most commonly expressed in percentage form and is computed according to Equation 2.1.

$$SoH = \frac{C_{max}}{C_n} \tag{2.1}$$

C_{max} is the maximum available capacity in the battery and C_n is the nominal capacity of the battery given by the manufacturer when the battery is new. A battery is often considered to reach its end-of-life (EoL) when its SoH decreases to 80% of its nominal capacity [9].

Though it might appear straightforward, estimating a battery's SoH is a complex task. As explained above, there are multiple chemical mechanisms that determine how batteries age. To get an accurate measurement of the maximum available capacity of a battery, and thereby the SoH, one needs full charging and discharging cycles. However, this is something that rarely occurs in real world driving [12]. For EV-applications, a Battery Management System (BMS) is used for keeping track of the car battery and its degradation. The BMS optimizes battery performance by protecting the battery pack from excessive temperatures and damaging charging profiles. Though not always accurate, it also includes software intended to determine battery states such as SoH [13].

2.1.3 Battery Cycling Protocols

Since measuring the true SoH is time-consuming and expensive, battery degradation is often evaluated through lab testing by performing Constant Current (CC) discharge cycling. The battery is put through tests that mimic usage patterns so that the degradation of the battery can be studied. The cycling tests are combined with Reference Performance Testing (RPT) that measures the capacity of the battery between the cycles [14]. When performing CC-tests, the batteries are often charged from empty to full capacity repeatedly without rest. A driver of an electric vehicle would probably rather charge and discharge in a range, never reaching 0 and with long rest periods. Performing cycling tests that simulate this behavior is referred to as Dynamic Discharging (DD). Another method that can be used to simulate degradation is Random Cycling (RC). This was used in a study by Heydarzadeh et al. where the discharge current was decided based on battery cell-specific parameters and some factor of randomness [15].

Performing dynamic discharging affects the aging of the battery and creates patterns in a different way than what is normally found in traditional CC lab testing. In a study from 2024, Geslin et al. show that dynamic discharge enhances lifetime compared to constant charging [14]. This indicates that traditional lab tests give an unrealistic picture of battery degradation compared to field data or lab data obtained via dynamic charging and discharging with rest periods. This is caused by the fact that different driving and cycling behaviors trigger different degradation mechanisms. SEI growth, which is one of the main degradation modes, causes the volume of the negative electrode to expand during charging and discharging. This expansion degrades the battery. SEI growth is caused by for example high loads,

in particular a large depth of discharge. This explains why performing full charging cycles is more demanding on the battery than real-world driving/dynamic discharging with shorter charging and discharging cycles [16].

Another mechanism that could be affecting the lab-field gap is that real driving and dynamic discharging often include longer breaks for the batteries. This enforces cell relaxation which allows some of the capacity loss to be reversed, which is generally not seen in CC tests [16]. Since dynamic discharging enhances lifetime, the amount of time and therefore data collection resources needed to obtain battery data until EoL is much higher than for CC testing [12].

Many aging mechanisms of batteries have an impact on the battery's temperature generation, which is one of the reasons why temperature is strongly related to battery SoH. Temperature is also a parameter that is different in a lab environment compared to a real-world driving scenario. Other useful health indicators that are often used when modeling battery degradation include current and voltage data.

2.2 Machine Learning Architectures

In this section, relevant machine learning architectures are presented and explained.

2.2.1 Autoencoders

A popular self-supervised learning architecture is the autoencoder. Autoencoders consist of an encoder and a decoder with a latent space in between [17]. The basic structure can be seen in Figure 2.2. The encoder learns to represent the input data X in the lower-dimensional latent space, which is then reconstructed by the decoder as $\hat{X}$. The self-supervised aspect of the autoencoder lies within its ability to calculate loss based on the accuracy of the reconstruction $\hat{X}$, using the input X as the ground truth and fully bypassing the need for labeled data.

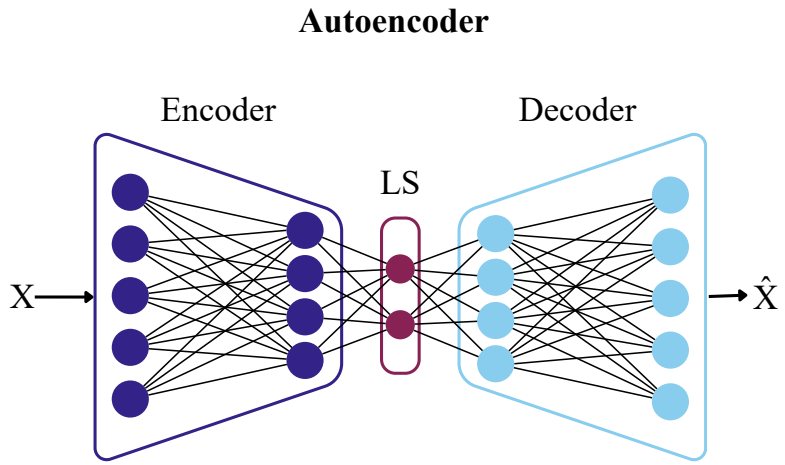


Figure 2.2: A model of an autoencoder with dense layers.

Autoencoders have drawn increasing attention in recent years, leading to an evolution in the area. There are now multiple different versions of autoencoders, including convolutional, variational and denoising autoencoders [17]. There are also autoencoders that use other discoveries in the machine learning field. The encoder and decoder layers can be interchanged with a wide variety of neural network structures, like recurrent layers found in Recurrent Neural Networks (RNNs) or Convolutional layers found in Convolutional Neural Networks (CNNs).

2.2.2 CNN

Convolutional neural networks (CNNs) consist of convolutional layers, pooling layers and fully connected layers. A simple CNN model structure can be seen in Figure 2.3. The pooling layers downsample the input efficiently by using kernels and different kernel operations. CNNs have been shown to be very effective when handling images but also other datasets that require models that can handle more spatial dependency than simpler neural networks [18].

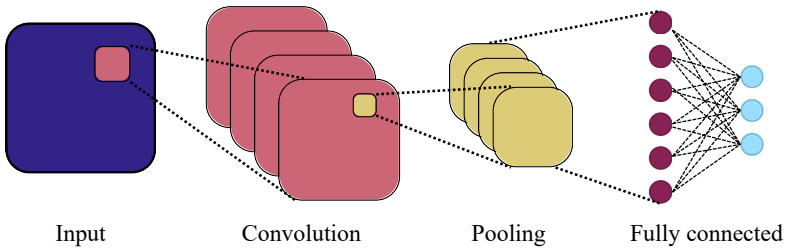


Figure 2.3: A model of the CNN architecture.

2.2.2.1 TCN

A Temporal Convolutional Network (TCN) uses the principals of a CNN but with added recurrence to account for the time aspect of the data. The model has been

used by Zhou et al for SoH predictions, proving its efficacy for battery degradation analysis. In contrast to the RNN, the TCN however uses convolutional kernels like a CNN would, making the prediction of local patterns more accurate than the RNN[19].

2.2.3 RNN

A recurrent neural network (RNN) is, as its name reveals, recurrent by nature. Besides the input x_t, each node at time t takes as additional input the output h_{t-1} from itself in the previous timestep $(t-1)$. This is done through so-called self-feedback loops, which allow the RNN to maintain a hidden state that functions as a form of short-term memory [20]. A simple model of the RNN architecture can be seen in Figure 2.4.

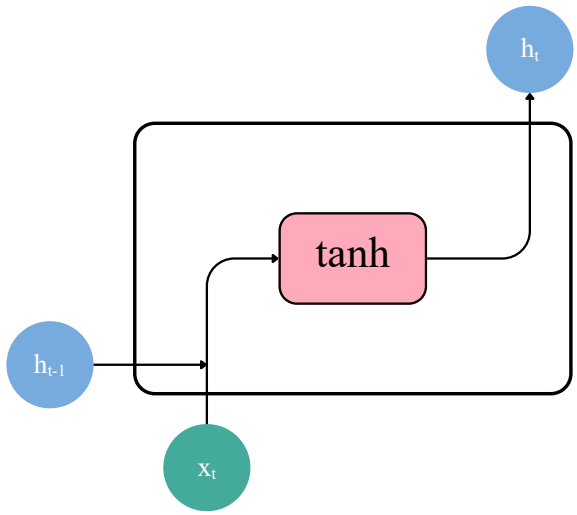


Figure 2.4: A model of the RNN architecture.

2.2.3.1 GRU

The gated recurrent unit (GRU) is an extension of the standard RNN. It introduces a gating mechanism as a tool to mitigate the vanishing gradient problem which commonly occurs with RNN models. The gating mechanism allows the GRU to control the flow of information through the unit [20]. A model of a simple GRU architecture can be seen in Figure 2.5.

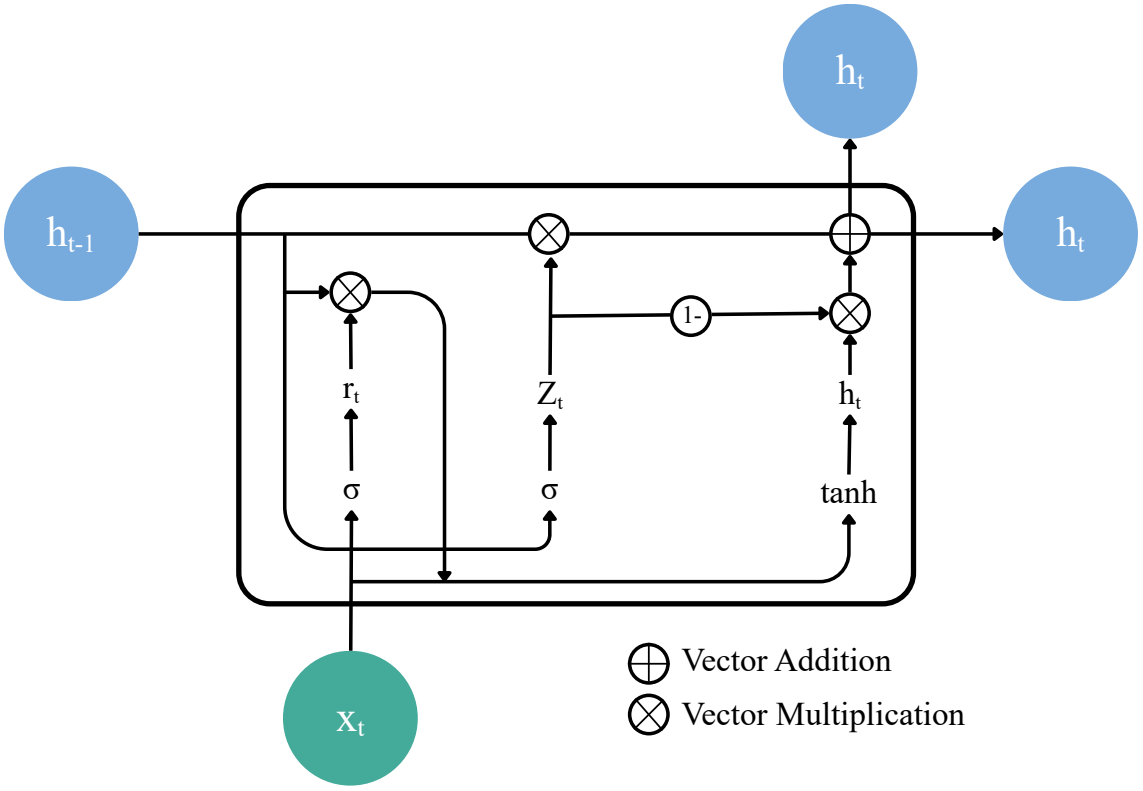


Figure 2.5: A model of the GRU architecture.

2.2.3.2 LSTM

Adding complexity to the recurrent models, the LSTM (long-short-term-memory), in turn, adds an internal state within the hidden layer architecture as well as a gating mechanism. These features mitigate the issues with exploding/vanishing gradients which are otherwise common in RNNs. Figure 2.6 shows a schematic diagram of the basic LSTM structure. The internal state C_t acts as a long-term memory state, containing information from the first timestep $t_0 = 0$ to the present. An LSTM has three gates: the forget gate, the input gate and the output gate. The forget gate regulates which information is to be forgotten from the previous internal state C_{t-1}, the input gate regulates the information that is kept from the previous h_{t-1} and the current input x_t. Lastly, the output gate is used to determine the external state h_t, based on the internal state C_t. Altogether, the gates control the flow of information through the LSTM cell.

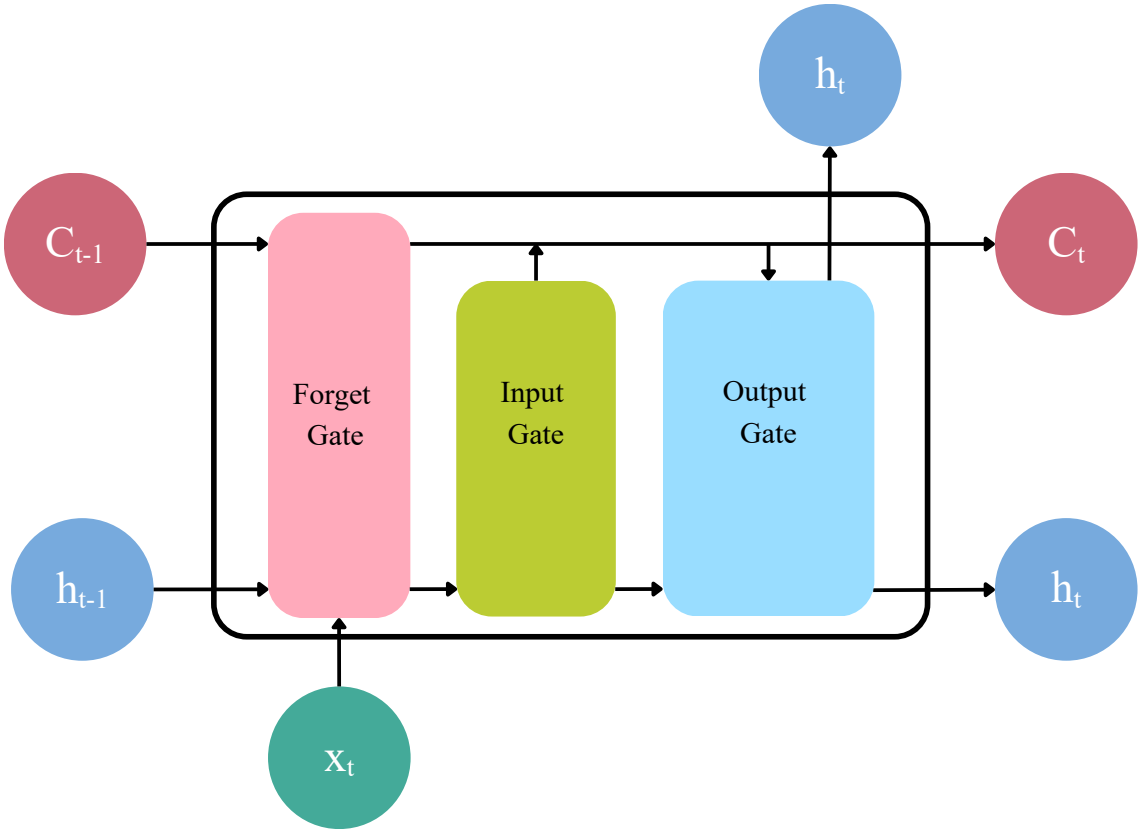


Figure 2.6: A model of the LSTM architecture.

2.2.4 Transformers

First introduced by Vaswani et al in 2017 [21], the Transformer is a neural network architecture that utilizes attention mechanisms to bypass the use of recurrence and convolution. Recurrent models do not allow parallelization during training which for longer sequences can become critical. Hence, a non-recurrent model might be preferred. Transformers have an encoder-decoder structure where the encoder maps an input to a continuous representation and the decoder generates an output given this representation. A model of a simple Transformer can be seen in Figure 2.7

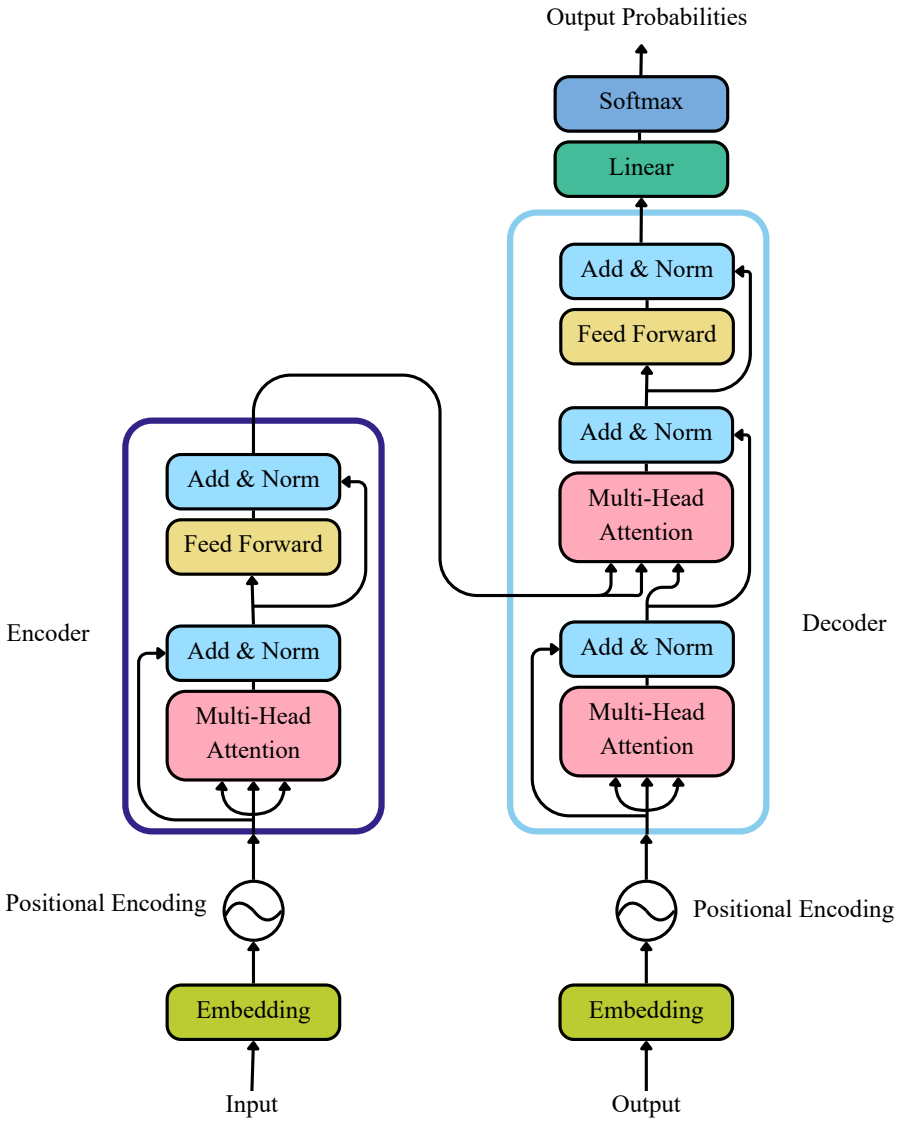


Figure 2.7: A model of the Transformer architecture.

The main feature of the Transformer, the attention, is a function mapping query, key and value to an output. The Transformer uses multi-head attention, meaning that it runs multiple attention mechanisms in parallel. Attention is used in the Transformer as self-attention in the encoder, self-attention in the decoder and attention in the decoder where queries come from the previous decoder layer, but the keys and values are from the encoder output. By using attention in this way, Transformers allow for both local and global dependency recognition [21].

Since the Transformer has no recurrence or convolution in its design, positional encoding is used to keep track of the order of a sequence. Furthermore, the Transformer is auto-regressive, meaning that each generated value uses previously generated values as additional input [21].

2.2.5 Hybrid Architectures

When creating a network architecture, one might benefit from combining different neural networks to achieve the desired results. This is often referred to as hybrid architectures or as performing upstream and downstream tasks. An example of this, combining an autoencoder with a neural network can be seen in Figure 2.8.

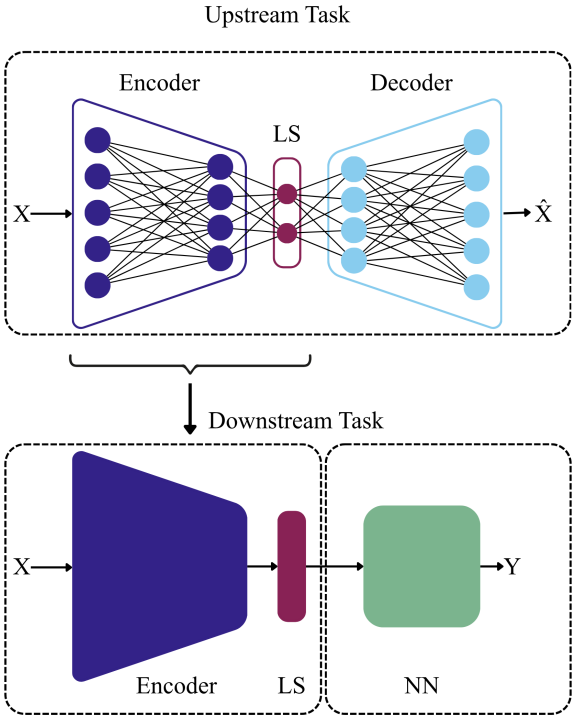


Figure 2.8: An example of a hybrid architecture with an upstream and a downstream task.

Zhang et al. [7] propose a self-supervised learning method using a sparsely labeled training set. The method consists of training an autoencoder to reproduce cycling data from batteries and then using the pretrained encoder combined with a simple prediction model to predict SoH. By doing this the encoder learns to efficiently represent the data and then a simple model can be effectively used with a small sample of training data. This way, a large unlabeled and a small labeled dataset can be utilized.

Another study using upstream and downstream tasks is Wang et al. [8], where the authors propose a method using Transformer encoders for SoH prediction. The method uses pretraining and finetuning of the Transformer encoder to improve accuracy. The pretraining adds noise to the data using random patches, which creates a pretext learning task. Using this method, the authors could lower the labeling ratio to 30%.

2.2.5.1 Pre-training and Fine-tuning

A commonly used method in machine learning and especially when creating hybrid architectures is performing fine-tuning on a pretrained model. This is usually executed by training a model on a large, broad dataset and then re-training the weights with a smaller learning rate on a smaller, more niche dataset. Another option is to re-train only a few layers, usually in the end of the model. The goal with performing fine-tuning is often to perform predictions with good accuracy while having less data that is labeled and of good quality. This is for example used in an article by Wang et al., where the model was pretrained on data with a large amount of unannotated data and fine-tuned on a smaller sample with annotations [8]. In Figure 2.8, where a hybrid architecture is shown, the autoencoder could be pretrained on a large, broad dataset and then in the downstream task, the encoder is fine-tuned using a smaller dataset.

3

Methods

In this chapter, methods used when performing the thesis work are presented and explained. The method for this project is a pipeline divided into dataset selection and preprocessing, an upstream task, a downstream task, a fine-tuning task and evaluation metrics.

3.1 Dataset Selection

The novelty of this project lies partly in using the available data in a more efficient way by creating a model that can be trained on a majority of the more accessible lab data and show good performance on the more realistic field data. The lab data used in this project comes from 9 different sources. These are shown in Table 3.1. Note that the "number of batteries"-column shows the total amount of batteries remaining after some initial preprocessing and quality checks. For example, for the field data, batteries that had no decrease in capacity were removed based on the assumption that this was due to faulty measurements. The lab datasets are referred to as NASA [22], MIT [23], XJTU [24], HUST [25], Tongji [26], Turku [15], Stanford [27], Geslin [14] and Berlin [28]. The datasets Berlin, Geslin and Turku, contained lab data that came from driving-simulated lab experiments. The field data was obtained from an article by Liu et al. from Chongqing University [12], and is referred to as the EV fleet data. The field data consisted of data from 300 EVs equipped with NMC batteries over three years. The battery pack contained 96 battery cells. In Table 3.1 the distribution of the lab and field data is displayed. The battery chemistry in the NASA dataset was not disclosed by the authors and therefore that field is empty.

Table 3.1: Description of used datasets.

Dataset	Number of batteries	Battery chemistry	Cycling protocol
NASA	17	-	CC
MIT	139	LFP	CC
XJTU	55	NMC	CC
HUST	77	LFP	CC
Tongji	73	NCA and NMC	CC
Stanford	204	NCA	CC
Turku	8	NCA, NMC and LFP	CC and RC
Geslin	92	NCA	CC and DD
Berlin	3	NMC	DD
EV fleet	183	NMC	Real driving

3.2 Data Preprocessing

In this section, the data preprocessing methods used in this project are explained. The input data consists of voltage, current and temperature measurements and the target data consists of SoH curves.

3.2.1 Input Data

The data was preprocessed for a few different purposes, but mainly to downsize it to even sizes and obtain a common format for all data. The input data consisted of voltage, current, and temperature data, from which the first 10 cycles were extracted. This was done since early battery behavior has been shown to relate to the capacity degradation patterns [29]. To be able to use the field data and the lab data interchangeably the data needed to be converted to a common format. The greatest misalignment between datasets was that the lab data mainly included time measured in cycles and not in even time steps, which the field data included. This resulted in artificial time steps being introduced for the lab data by linear interpolation of the existing time steps. To be able to extract 10 cycles from the field data, some conversion to cycles needed to be performed. This was done through extraction of equivalent full cycles (EFC). The EFCs were extracted by computing the cumulative capacity throughput. A sample result of this EFC extraction can be seen in Figure 3.1, where a lab data sample is compared to a field data sample.

The lab data is very regular, with evenly spaced cycles that reach the same amplitudes. The ten cycles are clearly distinguishable. The field data is more noisy and less regular, and the cycles are often not as easy to distinguish. The difference comes from the different cycling methods resulting in different patterns, but also from different preprocessing methods for the data collectors as well as different preprocessing methods within this thesis project. From the lab cycling data, 200 evenly spaced data points were extracted for the 10 cycles, resulting in a total of 2000 data

points each for the voltage, current and temperature features. The total 6000 data points became the common format for all input training data.

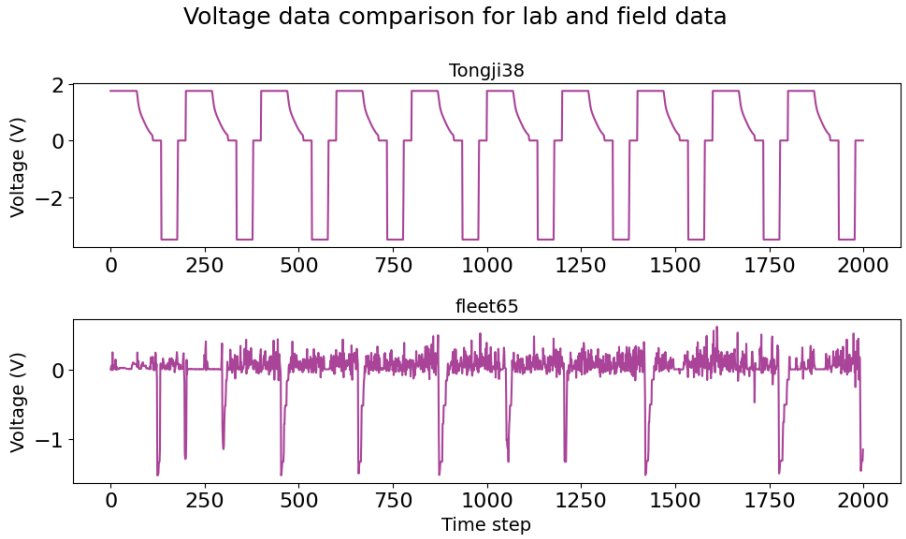


Figure 3.1: Examples of two voltage samples, Tongji38 originating from a lab dataset and fleet65 originating from the field dataset.

Some of the batteries were not used in the project due to poor data quality. This could e.g. be that a large chunk of the data was 0 or None, resulting in that the cycles could not be described accurately. The input data was normalized using mean and standard deviation calculations. This can be seen in equation 3.1, where μ denotes the mean and σ denotes the standard deviation.

$$X_{norm} = \frac{X - \mu}{\sigma} \tag{3.1}$$

For the field data, the data was measured over the battery pack and not individual cells. This is due to the field setting not allowing for measurements over individual cells, unlike the lab setting. Since the field battery cells were parallel connected, the current features were divided by the number of battery cells to account for the different scale compared to the lab data.

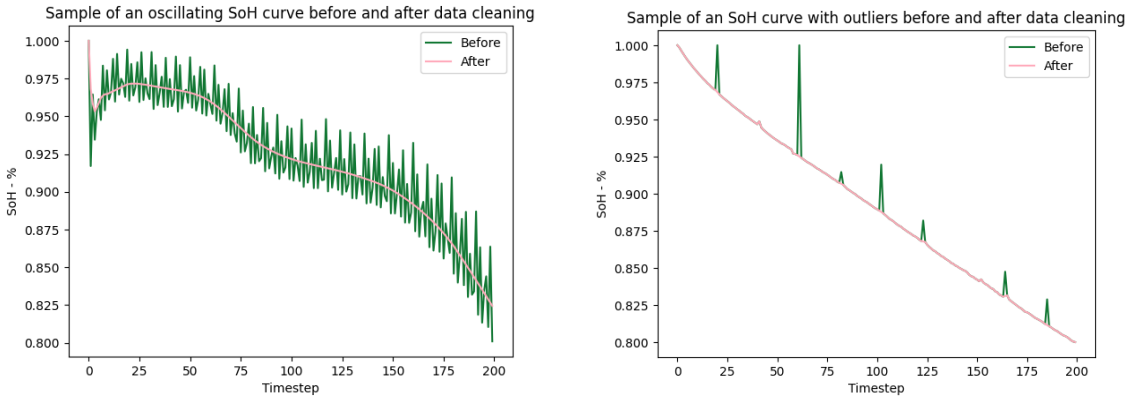
3.2.2 Targets

The SoH targets were based on capacity measurements from every available cycle. The capacity measurements in the lab data were converted to SoH measurements by division with initial capacity according to equation 2.1. The SoH curve was re-sampled to have 200 data points to obtain a uniform shape. To be able to give a temporal prediction an additional parameter referred to as the number of cycles was

extracted and predicted, so all of the SoH prediction models predicted a curve with 200 data points as well as the integer describing the total number of cycles so that the curve could be rescaled to the correct length. The number of cycles integer was also normalized using equation 3.1. The SoH value of the first cycle was considered as corresponding to the initial capacity and therefore equal to 1, and the full SoH curve was clipped to stay below 1.0, because of the limitation that batteries can not gain capacity and can therefore not take higher values than the initial capacity. The SoH curves were also discarded if the SoH value was equal to 1.0 for more than 25% of the number of cycles. This was regarded as a sign of unexpected degradation patterns, which were not wanted in the predictions.

3.2.2.1 Target Preprocessing

The SoH curves were also processed to remove oscillations and spikes that can be regarded as faulty measurements or outliers. To decide which curves contained oscillations and spikes, a small MLP was trained to classify the SoH curves into simple classes describing the behavior of the curves. The model could then identify the curves that showed, e.g. oscillating behavior or sudden spikes/drops. Based on this, the outliers could be removed through interpolation and the oscillations could be removed through Locally Weighted Scatterplot Smoothing (LOWESS). LOWESS is a linear regression method that effectively smoothens out the oscillating curves. This can be seen in Figure 3.2. The reason the noise was removed from the targets was that it could be linked to sensor errors and other noise, which is not something the models should predict. Generally, it is not feasible for a battery to gain capacity after degradation, so having targets with that behavior would have likely caused the models to give non-physically feasible predictions.



(a) Oscillating SoH Curve **(b)** SoH Curve with outliers

Figure 3.2: Samples showing how the cleaning of SoH curves was performed.

In the field data, the SoH values were extracted by analyzing the charging status and extracting segments where the vehicle is charging a significant amount of time. The SoH targets are therefore approximations, and required a bit more work and

preprocessing compared to the lab targets. The targets are also affected by the BMS system in the cars, which are in turn based on approximations by the car manufacturer. Because of this, the field data targets are less reliable. To remove some noise, Complete Ensemble Empirical Mode Decomposition with Adaptive Noise (CEEMDAN) denoising was performed on the field SoH curves. This has been shown to be a good method for processing SoH curves, since CEEMDAN effectively removes noise that was caused by signal errors [30]. CEEMDAN was applied to the field data since the SoH approximations were noisy. CEEMDAN was also applied to the Stanford dataset, since the SoH curves were regarded to be more noisy than the rest of the lab data, which did not need the smoothing to the same extent. The idea behind using CEEMDAN to have smooth targets was the same as for using LOWESS, the SoH should physically not oscillate, have sudden drops or spikes, so that is not something the model should be trained to predict.

CEEMDAN works by decomposing the curve into several Intrinsic Mode Functions (IMFs). Gaussian noise corresponding to the normal distribution of the signal is added. The residual of the signal and the IMF is repeatedly calculated and decomposed until no more decompositions are possible. Then the original signal can be expressed as a combination of IMFs and the residual. By comparing the Pearson correlation of the different components to the original SoH curves, the components with a small correlation, which correlates to noise can be removed [30].

3.2.3 Dataset Separation

For the purpose of the project, three training datasets were created, as well as one test set. During training, the training sets were further split into training and validation sets. The three sets are referred to as:

- The upstream training set
- The downstream training set
- The fine-tuning training set

and the distribution of data between these sets is described in Table 3.2. Worth noting here is that there is an overlap between all of the training sets but no overlap between the training sets and the test set.

Table 3.2: Description of the train and test set split.

Dataset	Number of batteries	Number of batteries in upstream set	Number of batteries in downstream set	Number of batteries in fine-tuning set	Number of batteries in test set
NASA	17	17	9	7	0
MIT	139	119	82	107	20
XJTU	55	47	42	1	0
HUST	77	77	77	77	0
Tongji	73	73	53	72	0
Stanford	204	204	192	187	0
Turku	8	0	0	0	8
Geslin	92	92	81	81	0
Berlin	3	0	0	0	3
EV fleet	183	168	0	65	15
Total	821	797	536	597	46

Since the aim of the thesis was to use a larger unlabeled dataset for upstream training, the upstream training set contained all training data that had passed quality filtering.

For the downstream training set, the batteries' EoL was considered to be when the battery reached 80% of its initial capacity. All of the data that was not in the test set that reached below 82.5 % SoH was added to the downstream training set. Since 80% SoH is a common threshold for EoL in the field, a significant amount of data was found to be cycled until about the 80% mark. To account for any inexact measuring, all batteries that reach below 82.5% were included. The SoH curves that reached between 80% and 82.5% SoH were kept as they were, while the SoH curves that reached below 80% were cut at the point they first reached 80%. The number of cycles until 80% or whenever else EoL was reached was saved as the number of cycles.

All of the data that was not in the test set and that reached below 95 % SoH was added to the fine-tuning training set. This training set was used for the separate early SoH prediction task, referred to as the fine-tuning task. The targets for this set were extracted at 95%, since the field data had a much slower degradation than the lab data. Some of the lab data was filtered out from the fine-tuning training set due to very fast degradation. A fast degradation often correlated to very few data points before 95% was reached, and this would have caused inexact data point estimations when rescaling to 200 steps.

In addition to the three training sets, a test set with unused data was used to evaluate the performance of the model. The test set consisted of the batteries from Turku, Berlin, 15 batteries from the EV Fleet data and 20 batteries from the MIT data. This data was separated from the rest of the data before extracting data for the training sets. The goal with the test set was to create a set that could benchmark

the performance across many different chemistries and cycling protocols. Turku and Berlin were selected due to that their size allows them to be removed entirely from the training data so that holdout testing can be performed. The test targets were not cut but kept as they were. This led to the test targets having uneven lengths. Therefore, SoH estimation on the test targets was a much more difficult task.

A description of what input data and targets were used in all of the datasets can be seen in Table 3.3.

Table 3.3: Description of the input data and targets in each dataset.

Dataset	Input data	Input data shape	Target data	Target data shape
The upstream training set	Voltage, current and temperature data	[2000 x 3]	Voltage, current and temperature data	[2000 x 3]
The downstream training set	Voltage, current and temperature data	[2000 x 3]	SoH degradation curve until 80 % SoH + number of cycles at 80% SoH	[200] + [1]
The fine-tuning training set	Voltage, current and temperature data	[2000 x 3]	SoH degradation curve until 95 % SoH + number of cycles at 95% SoH	[200] + [1]
The test set	Voltage, current and temperature data	[2000x3]	Full SoH degradation curve + number of cycles at last measurement	[Uneven lengths] + [1]

To make sure that there was no data leakage, the validation sets for all training tasks were selected to be a subset of the validation set in the upstream validation set. The upstream validation set was selected by a random split from the training data. All of the batteries were marked with a unique label, so e.g. the third battery imported from the MIT dataset received label MIT3. The split of the downstream set and the fine-tuning set was based on which batteries that were in the randomly selected upstream validation set. By doing this, there was no overlap between the upstream training set and the downstream validation set. The same goes for the upstream training set and the fine-tuning validation set. Lastly, there was no overlap between the downstream training set and the fine-tuning validation set, which

ensures that all validation tasks were performed on unseen batteries.

3.3 Upstream Task

The main purpose of the upstream task was to compress the data and extract the most important features from unlabeled data to optimize the input to the downstream model. For this task, several autoencoder (AE) structures were evaluated. These were built using the PyTorch library in Python. The autoencoders were trained to reconstruct temperature, voltage and current cycles. To ensure comparability, the autoencoders were built to have similar parameter counts as well as latent spaces of the same size. The models evaluated for this task were the following:

- MLP autoencoder
- CNN autoencoder
- RNN autoencoder
- GRU autoencoder
- LSTM autoencoder
- Transformer autoencoder

The MLP autoencoder, also referred to as the vanilla autoencoder, was chosen as a baseline model for its simplicity. This autoencoder is the simplest form of autoencoder with fully connected layers, similar to what can be found in an MLP. This autoencoder sees each timestep independently and hence lacks temporal dependency. Therefore, the CNN autoencoder was tested. The one-directional convolutional layers allow the CNN autoencoder to capture local patterns, and in more dimensions than the MLP autoencoder, making it better for images, where CNNs are mostly used [17]. This could also be useful when processing time series, since the dimensions of the data can be handled differently. This was shown by Obregon et al., who were able to estimate SoH in LiBs using a convolutional autoencoder combined with a deep neural network [31]. The CNN autoencoder is however still limited in its ability to see global trends.

To get a model with the ability to capture global trends three different recurrent autoencoders were tested. Recurrent models are often used for timeseries data. A simple RNN autoencoder uses recursion to handle larger patterns and longer sequences in the data better [32]. GRU and LSTM autoencoders use the same principles but with larger levels of complexity. LSTM autoencoders have shown to work well when handling large, complex datasets where the temporal dimension is important. By preserving the temporal dimension better, the average error can be lowered compared to an MLP autoencoder [33]. The intention behind trying all three models was to investigate how and if the added features of gates and memory from the GRU and LSTM effected the model performance compared to using a simpler RNN. A decision was made to not use teacher forcing in the decoders of the recurrent models, even though it improved recreation results significantly. The reason for this decision

was that this would allow the models to bypass the information in the latent space, only using the input x for its reconstruction, which in turn might lead to a less valuable latent space.

A Transformer autoencoder is an autoencoder consisting of Transformer modules. Transformers have been shown to be powerful tools when handling large amounts of data [21]. Therefore, it was of interest to investigate if this dataset is large and complex enough for a Transformer to show good performance in the autoencoder decompression task. A Transformer consists of an encoder and a decoder, just like an autoencoder. However, it does not by itself compress the data like an autoencoder. To enforce a latent space that could be extracted, a bottleneck was added between the encoder and the decoder.

The size of the latent space was kept to 128 for all models, except for the CNN autoencoder, which has a latent space of 125 to account for even convolutional steps. A latent space of 128 corresponds to about 6 % of the size of the input data with 2000 data points, which was found to be a good level of compression. Other parameters were optimized using Optuna within reasonable intervals and therefore varied between different models. The goal was to keep the number of trainable parameters within the same range so that the models would be comparable. However, some compromises to this were made.

3.4 Downstream Task

To continue the pipeline, the encoders from the upstream task were extracted and followed by a downstream model. Most of the encoder layers were frozen, whilst some of the last layers were set to trainable. This was done to have the encoders remember most of the information learned in the pretrained task but with some flexibility to adapt to the SoH prediction task and how the different downstream models behave. The number of trainable layers varied between the different encoders, but the goal was to keep around 20% of the encoder layers trainable.

The task for the downstream model was to predict SoH degradation from full SoH to EoL, which was considered to be at 80%. This was done by both predicting a projection of the SoH curve, as well as predicting the number of cycles the battery goes through until reaching EoL. The loss function used during training was an element-wise Mean Squared error (MSE), which is described in equation 3.2. The α parameter was set to 0.05, which was found to be a good compromise given that the task of predicting an accurate degradation curve was found to be a harder task than the number of cycles.

$$\mathcal{L} = \text{MSE}(\text{SoH}_{\text{target}}, \text{SoH}_{\text{prediction}}) + \alpha \ \text{MSE}\Big(\text{cycle}_{\text{target}}, \text{cycle}_{\text{prediction}}\Big) \tag{3.2}$$

During evaluation, the SoH prediction was de-normalized and scaled to the predicted

number of cycles, to give the SoH estimation temporal value. Since this resulted in SoH estimations and targets of varying length, the shortest curve in each target-prediction pair was padded with SoH values at 80% to represent EoL, in order to be able to calculate MAE and RMSE. The models evaluated for this task are listed below.

- MLP
- CNN
- TCN
- LSTM
- Transformer

The MLP was once again used as a benchmark for its simplicity. Since the CNN showed good accuracy for the upstream task, a CNN was also evaluated downstream. To further capture the temporal aspects of the SoH estimation task, the Temporal Convolutional Network (TCN), LSTM and Transformers were tested. The use of the TCN was inspired by Zhou et al. [19]. LSTMs and Transformers have been shown to have good results for different SoH prediction and battery degradation related tasks [30] [5].

All of the models were also evaluated on training with 50 % of the training data, to determine the stability with a lower labeling ratio.

After all combinations had been evaluated, the best performing architecture was chosen for further testing and to be used for the early SoH prediction task. This architecture is referred to as the final model.

To investigate how much the labeling ratio can be minimized during training, the final model was trained on a range of ratios spanning from 10% to 100 %. An acceptable performance degradation when downsizing the labeling ratio was set at a 25% decrease in RMSE. This was done over five different train/validation splits for all ratios to measure the stability.

Furthermore, the best performing model was exposed to different robustness test and it was also evaluated without the upstream autoencoder taking the multimodal voltage, current and temperature data directly as input. This was done to further investigate the effect of the hybrid architecture. Lastly, the final model was tested on the final test set as explained in section 3.2.3.

3.5 Early SoH Prediction Fine-tuning Task

To further bridge the gap between lab data and field data, a fine-tuning task was performed. The goal was to create a separate model that was able to do an early SoH prediction until 95% SoH. This task entails fine-tuning the hybrid architecture

on batteries where the SoH curves only reach 95%, a different task compared to the downstream model. As explained in section 3.2, the reason for this was to be able to predict the degradation of the field data as the field data rarely reach a SoH below 90% and therefore could not be used in the original downstream task. The fine-tuning was done by freezing most of the model but keeping the last layers of the encoder as well as the SoH head and cycle head trainable with the goal of using the main information learned from the full training task on a different target.

The fine-tuned model was then tested on the final test set, as explained in section 3.2.3, and compared to the original downstream model.

3.6 Evaluation Metrics

The different models were compared through different metrics. One of the metrics used was a Root Mean Squared Error (RMSE). RMSE was computed by calculating the square of the sum of the summed difference between the prediction $\hat{y}_i$ and the target y_i, divided by the number of samples. This can be seen in equation 3.3.

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}} \tag{3.3}$$

Another used metric was a Mean Absolute Error (MAE). The MAE was calculated by summing the absolute errors and dividing by the number of samples, according to equation 3.4.

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |\hat{y}_i - y_i| \tag{3.4}$$

The reason that both of these were used was that the RMSE gives a larger impact to predictions that result in a very large error since the error is squared. The MAE gives the error impact proportional to the size of the error. The idea being that the RMSE could point out any models that were not able to capture some aspect of the data at all while the MAE would give a good overall accuracy score.

For the downstream and fine-tuning task, both the MAE and the RMSE were calculated on the rescaled targets, so the prediction of the degradation was rescaled to match the prediction of number of cycles. If the prediction and the target were of different lengths, the EoL target was added to the end of the list until the target and prediction reached the same length. For example, if the downstream model predicted 890 cycles, the degradation prediction with length 200 was rescaled to 890 data points. The target in this example was 910 cycles, so the target prediction was scaled to 910 cycles and 20 times 0.8 was added to the end of the prediction, so both the rescaled target and the rescaled prediction reached a length of 910. In this way, an error could be calculated that reflected both the prediction on the number of cycles until EoL and the degradation curve.

The test set was created in a way that would show how the model behaves on both lab and field data, across several different cell chemistries. The model's performance

on the test set shows how the model behaves on completely unseen datasets and how well it adapts to a new domain. Since the unseen Turku and Berlin datasets did not have sufficiently many batteries reaching below 80 % SoH, MAE and RMSE values could not accurately be calculated for the downstream model performance on the test set. The early SoH prediction model, however, could be quantitatively evaluated since most test batteries reached below 95 % SoH.

4

Results

In the following section, the results from the upstream, downstream and the early SoH prediction fine-tuning task are presented.

4.1 Upstream Task

Table 4.1 and Figure 4.1 show the numeric results for the autoencoders in their task to reconstruct the voltage, current and temperature curves. The Transformer AE shows the best results, closely followed by the CNN AE. The worst results come from the MLP AE and the recurrent models. The recurrent models had to be made smaller than the other models due to overfitting issues. Scaling up the recurrent models to match the other models led to even worse results.

Table 4.1: Performance comparison of autoencoder models.

Autoencoder	**Trainable parameters**	**RMSE**	**MAE**
MLP AE	1 223 760	1.9314	0.5616
CNN AE	1 327 878	0.4438	0.1245
RNN AE	413 827	1.3503	0.4143
GRU AE	779 651	1.3726	0.4316
LSTM AE	186 374	1.3481	0.4064
Transformer AE	1 360 854	0.3721	0.07949

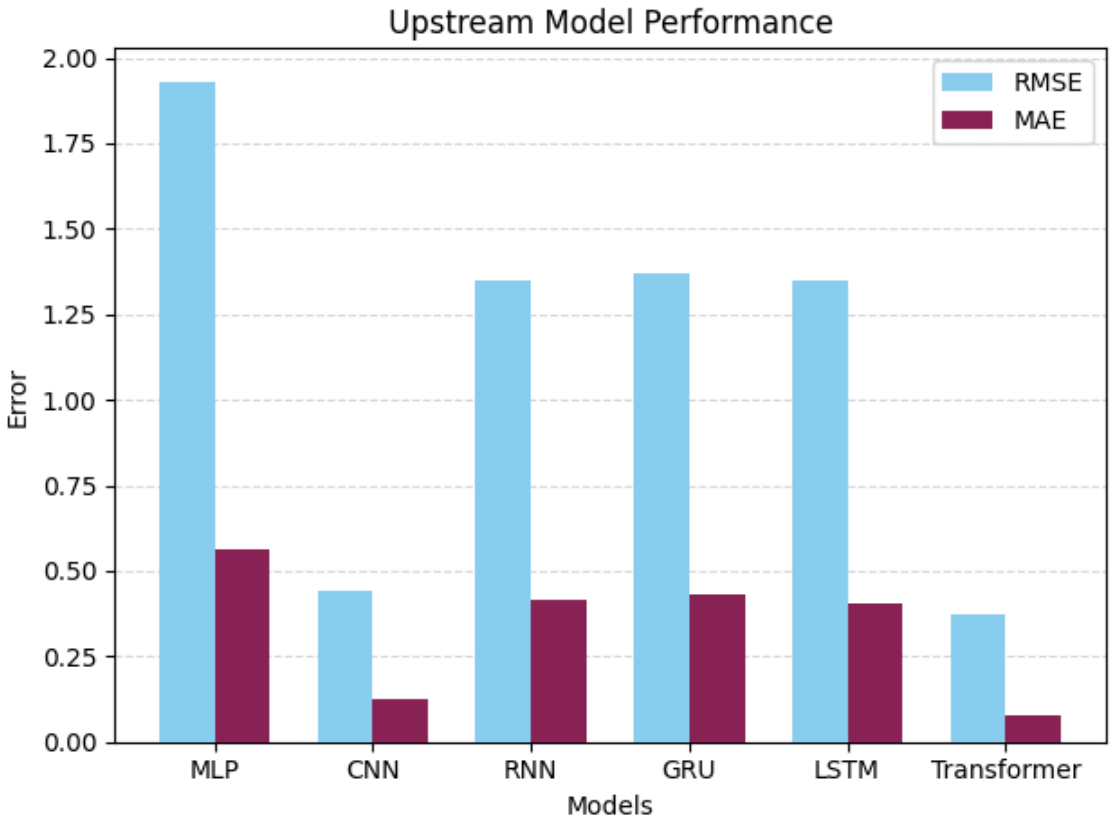


Figure 4.1: Performance comparison of upstream models.

Figure 4.2, 4.3 and 4.4 show the different models' performances on three different batteries. Batteries 10 and 25 are from the MIT and XJTU datasets, respectively. They have both been tested in a lab-setting. Battery 10 is very regular and battery 25 shows some irregularities. Battery 65 is from the EV fleet dataset, which means that it has been tested in the field, leading to even more irregularities. The MLP shows a solid performance on both batteries 10 and 25, but struggles greatly with reconstructing the data from battery 65. The same goes for the recurrent models. Even the Transformer, which has the best overall results, shows some errors in its reconstruction of battery 65.

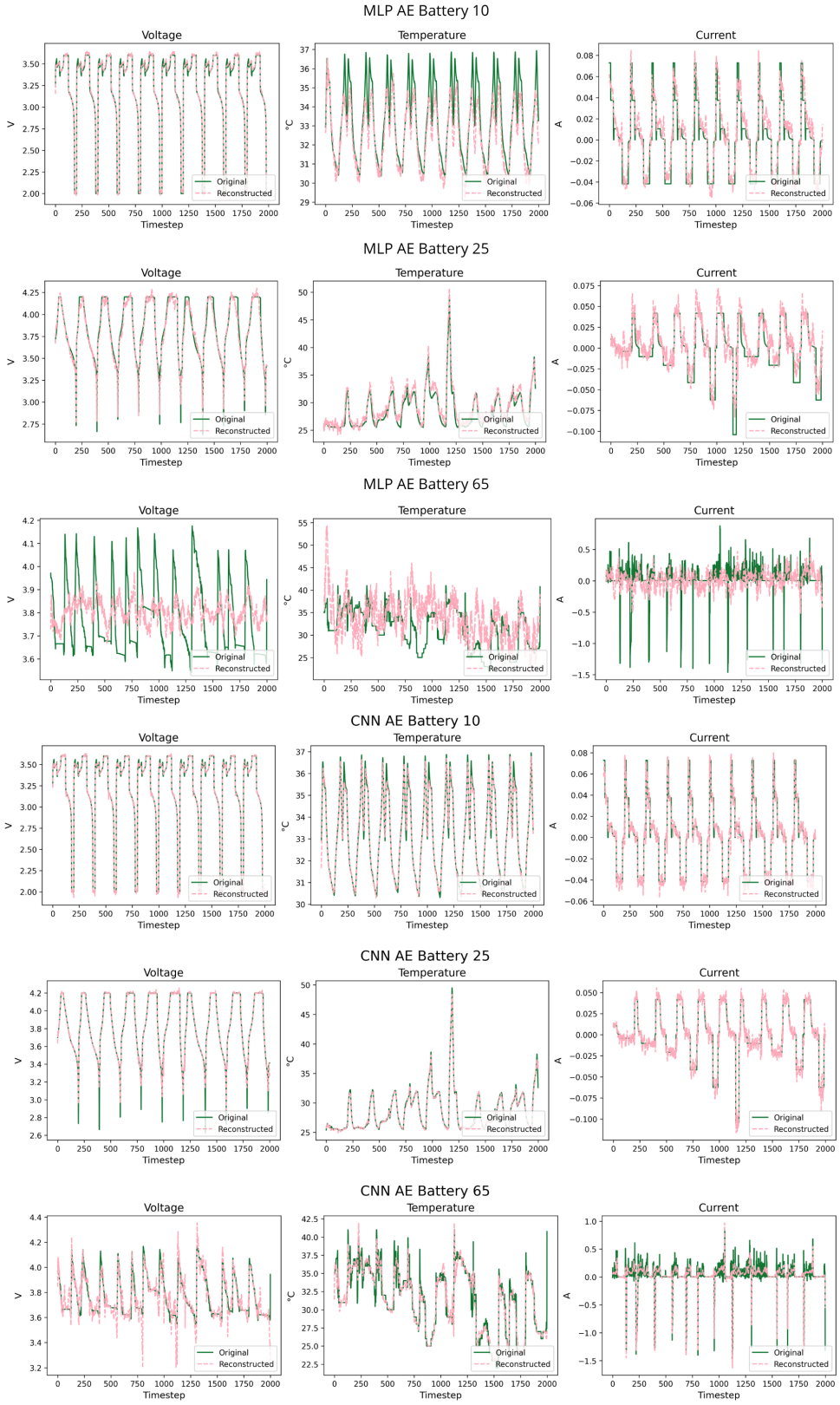


Figure 4.2: Results from the MLP and CNN autoencoder for three different batteries. Battery 10 originates from the MIT dataset and battery 25 from XJTU. Battery 65 is from the EV fleet dataset.

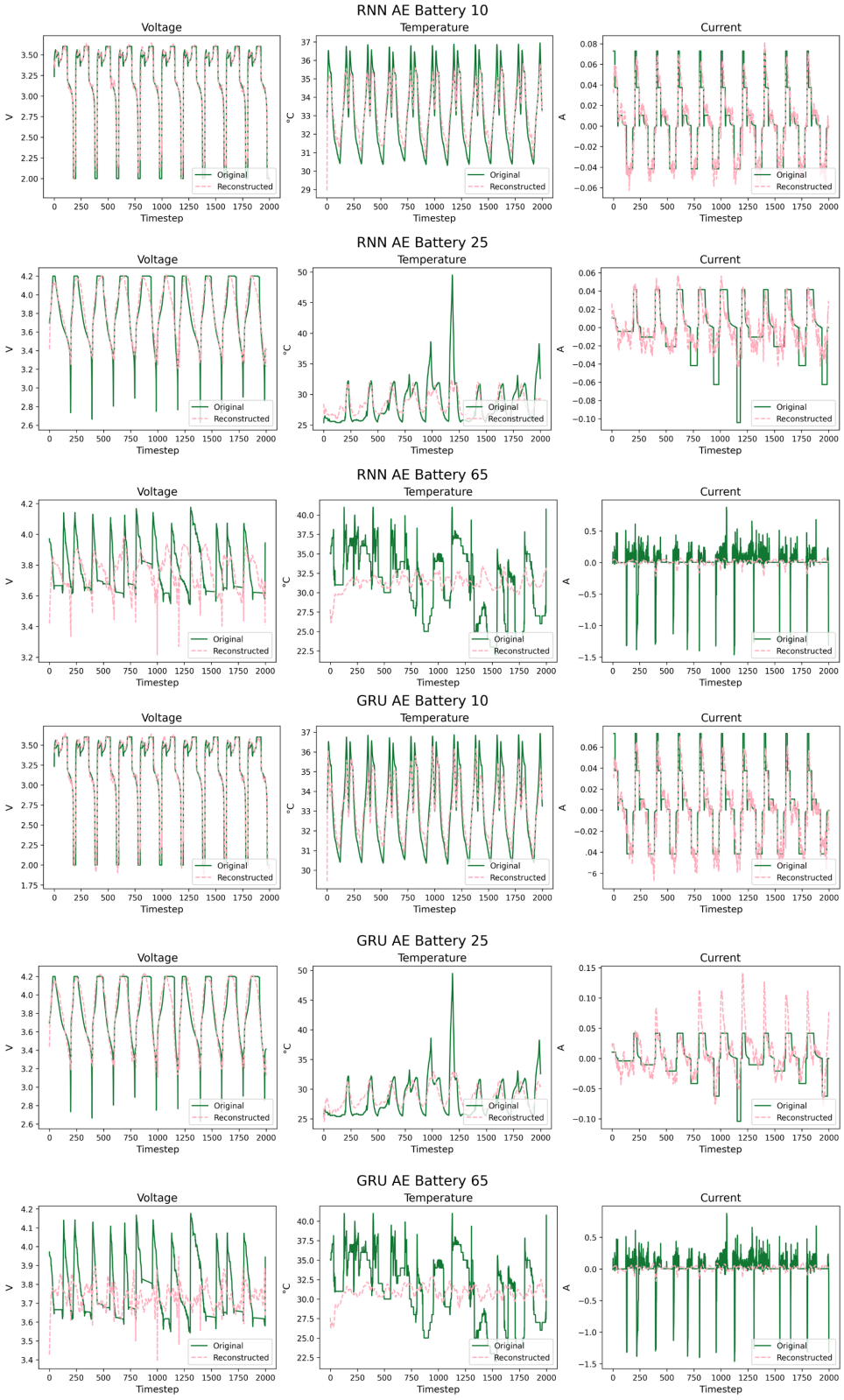


Figure 4.3: Results from the RNN and GRU autoencoder for three different batteries. Battery 10 originates from the MIT dataset and battery 25 from XJTU. Battery 65 is from the EV fleet dataset.

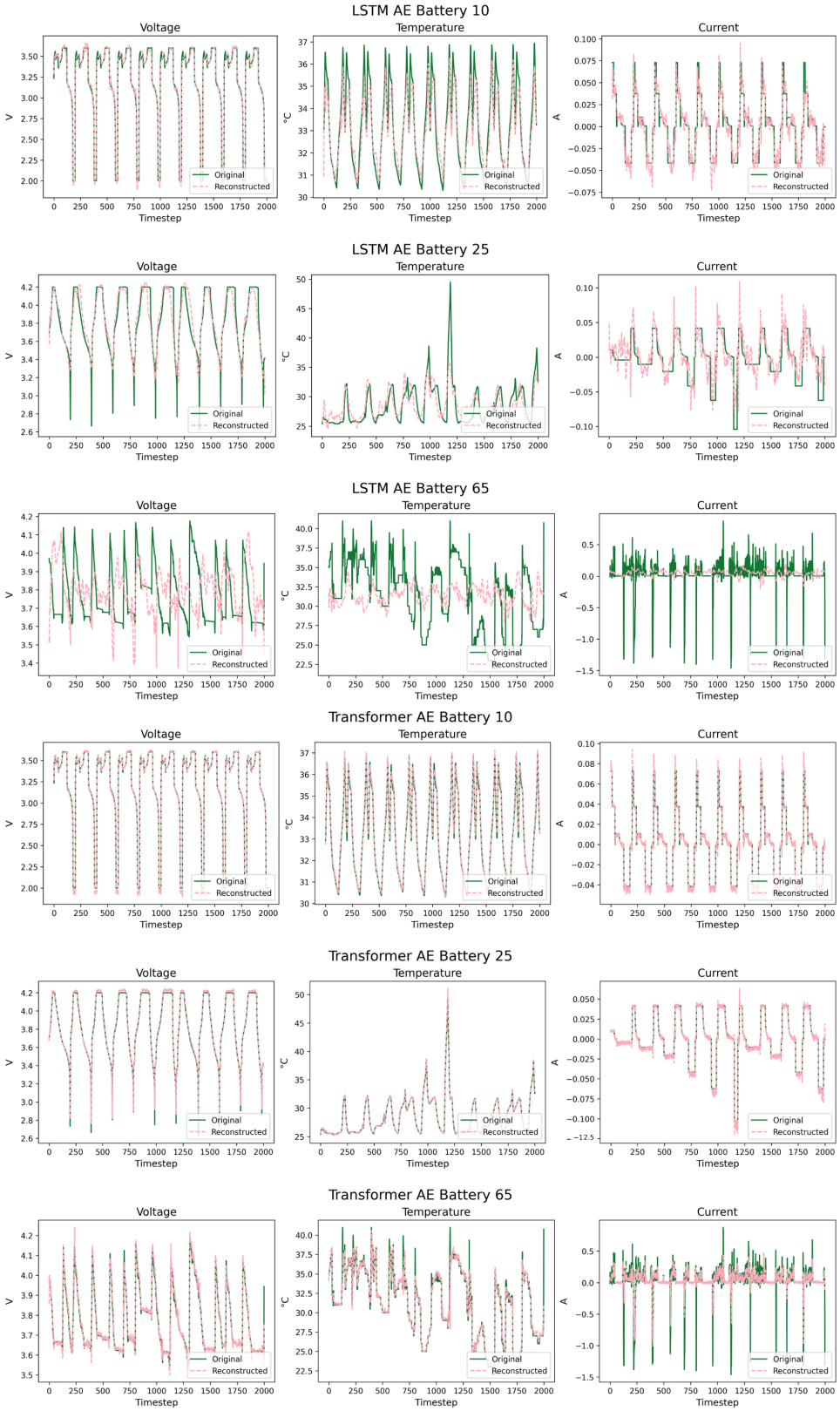


Figure 4.4: Results from the LSTM and Transformer autoencoder for three different batteries. Battery 10 originates from the MIT dataset and battery 25 from XJTU. Battery 65 is from the EV fleet dataset.

The encoders that were carried over to the downstream task were the Transformer encoder, the CNN encoder as well as the MLP encoder. The reason the MLP encoder was used despite its poor performance was that it could provide a benchmark for the other encoders. In addition to this, a more linear compression might give a more informative latent space than some of the other models that have more complicated decoders. A simpler model is often preferred so testing the MLP encoder also ensures that the pipeline is not overcomplicated.

4.2 Downstream Task

The performance of all different model combinations evaluated for the downstream task can be seen in Table 4.2, and Figure 4.5. The MLP downstream model has the worst performance. The other four models perform quite evenly, where the LSTM is slightly better than the rest. The best performing overall architecture is the upstream CNN AE combined with the downstream LSTM.

Table 4.2: Performance comparison of hybrid architectures.

Downstream model	**Autoencoder**	**Number of parameters**	**Trainable parameters**	**MAE**	**RMSE**
MLP	MLP	1235745	44881	0.03103	0.04237
MLP	CNN	1353303	31572	0.03400	0.04750
MLP	Transformer	1372839	268500	0.03694	0.04893
CNN	MLP	1509433	318569	0.03162	0.03923
CNN	CNN	1792335	470604	0.02382	0.03372
CNN	Transformer	1452767	348428	0.02841	0.03898
TCN	MLP	1564697	373833	0.02912	0.03714
TCN	CNN	1598415	276684	0.03364	0.04038
TCN	Transformer	1556703	452364	0.02645	0.03448
LSTM	MLP	1566137	375273	0.02326	0.03547
LSTM	CNN	2017839	696108	0.01963	0.03130
LSTM	Transformer	2867487	1763148	0.02187	0.03384
Transformer	MLP	8749337	7558473	0.02797	0.04115
Transformer	CNN	1649679	327948	0.03329	0.04357
Transformer	Transformer	1732383	628044	0.02446	0.03661

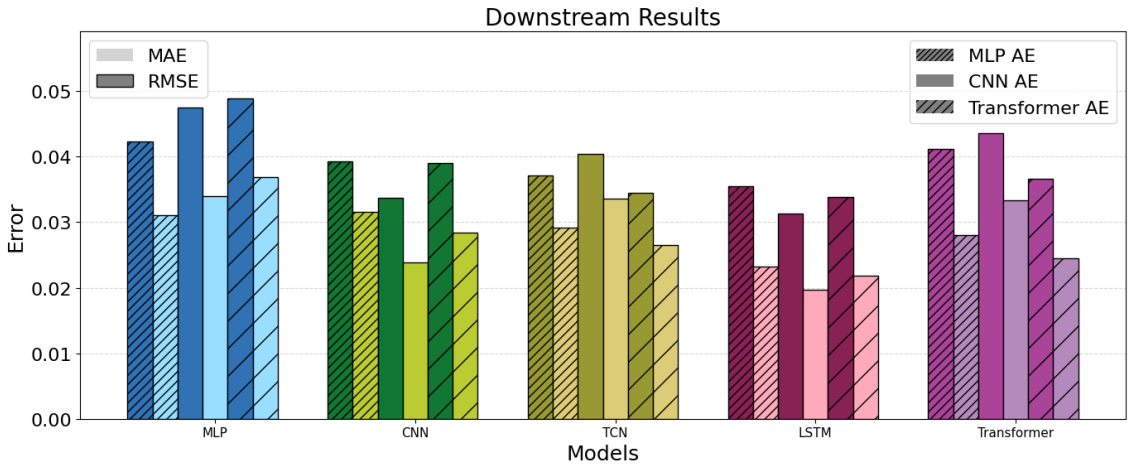
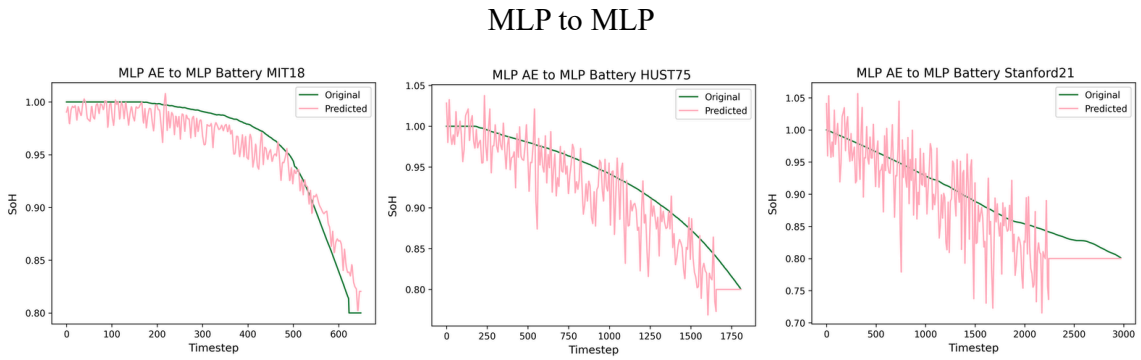


Figure 4.5: Performance comparison of hybrid architectures in the terms of RMSE and MAE.

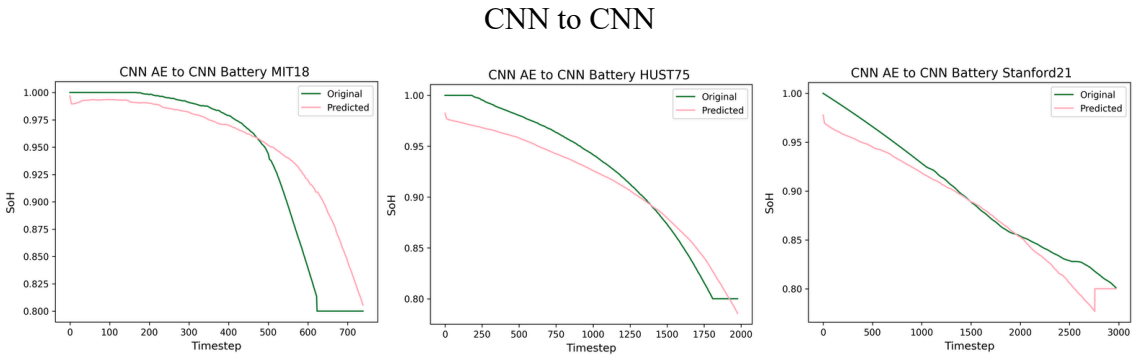
Figure 4.6 shows SoH reconstructions for three different architectures: the benchmark MLP to MLP, the CNN to CNN as well as the best performing CNN to LSTM. Note that the scales on the axes are not the same for the different batteries.

The MLP encoder consisted of three layers and therefore the last layer was set trainable, corresponding to 33,33% of the total number of layers. The CNN consisted of four trainable layers and therefore the last was set to trainable. This corresponds to 25% of the total number of layers. The Transformer encoder consists of two layers and a bottleneck to the latent space. Leaving one of the layers trainable would correspond to more than 50 % of the encoder being trainable and therefore only the bottleneck was set to be trainable.

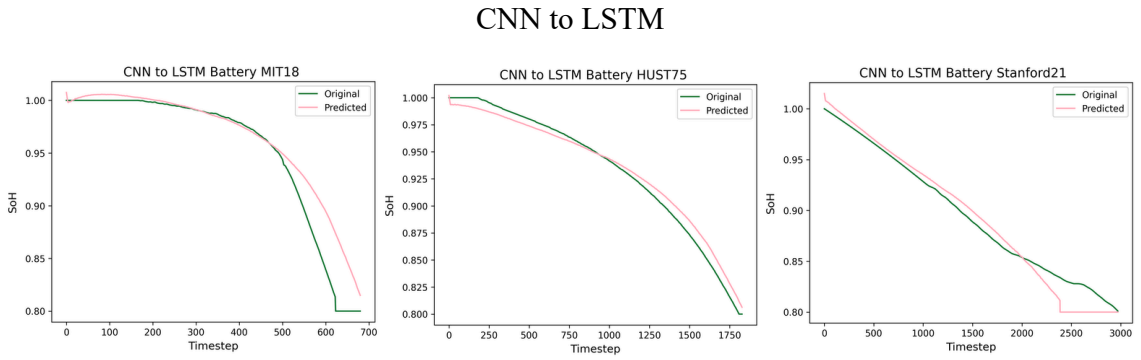
The RMSE and MAE of the number of cycle prediction is also recorded, and the results can be seen in Figure 4.7. The average MAE for the MLP autoencoder is 216, the average for the CNN autoencoder is 202.2 and for the Transformer autoencoder the average is 231. For the full models, the average MAEs are 240.3, 210, 207, 196 and 228 for the MLP, CNN, TCN, LSTM and Transformer, respectively. The largest overall MAE is produced by the Transformer autoencoder with the MLP head. The smallest overall MAE is produced by the CNN autoencoder to CNN head combination.



(a) MLP–MLP: RMSE = 0.04237, MAE = 0.03103.



(b) CNN–CNN: RMSE = 0.03372, MAE = 0.02382.



(c) CNN–LSTM: RMSE = 0.03130, MAE = 0.01962.

Figure 4.6: SoH estimations of three batteries for three different architectures. One battery is from the MIT dataset, one battery is from the HUST dataset and one battery is from the Stanford dataset.

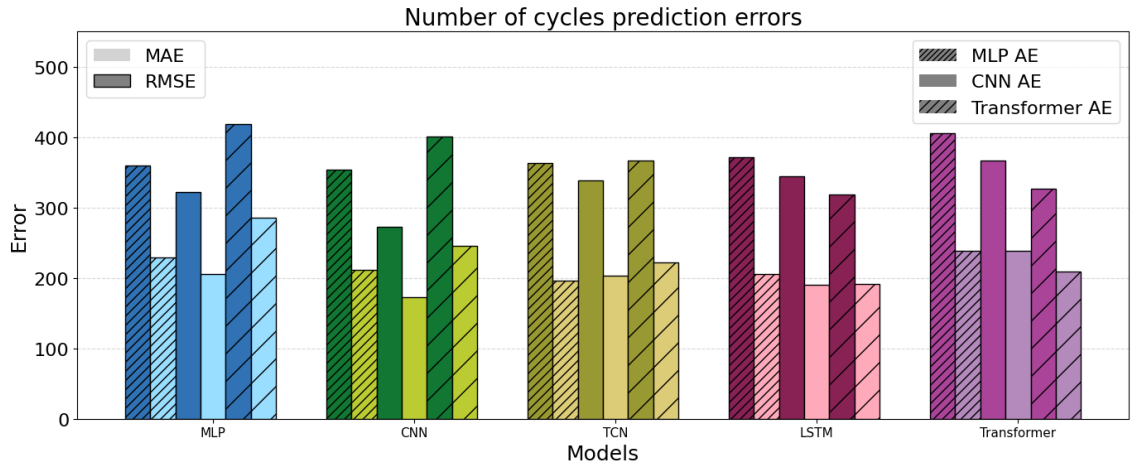


Figure 4.7: Performance comparison of hybrid architectures in the terms of number of cycles prediction error.

Table 4.3 and Figure 4.8 shows the performance of the models when trained on only 50% the data, corresponding to 214 batteries. The best performing architecture when trained on all training data is the CNN autoencoder to LSTM. The next best model was the Transformer autoencoder to LSTM. When training the model on just 50 % of the data, the Transformer autoencoder to TCN model outperforms the CNN autoencoder to LSTM.

Table 4.3: Performance of hybrid architectures when trained on 50 % of training data.

Downstream model	**Autoencoder**	**MAE**	**RMSE**
MLP	MLP	0.03706	0.05301
MLP	CNN	0.03793	0.05026
MLP	Transformer	0.04550	0.06108
CNN	MLP	0.03737	0.04491
CNN	CNN	0.03394	0.04530
CNN	Transformer	0.03111	0.04118
TCN	MLP	0.03658	0.04453
TCN	CNN	0.03224	0.04267
TCN	Transformer	0.02418	0.03682
LSTM	MLP	0.02811	0.04224
LSTM	CNN	0.02629	0.04024
LSTM	Transformer	0.03223	0.04288
Transformer	MLP	0.02860	0.04281
Transformer	CNN	0.03270	0.04210
Transformer	Transformer	0.03357	0.04548

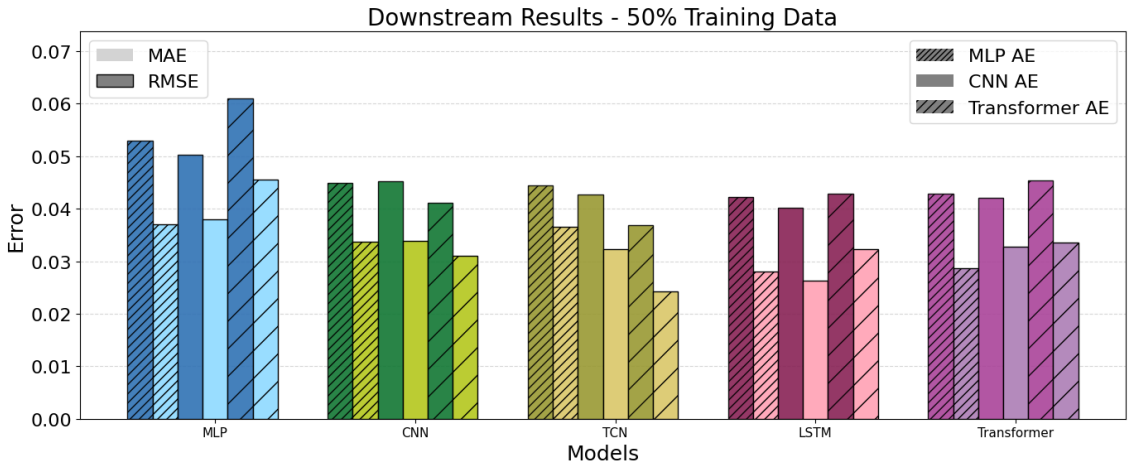


Figure 4.8: Performance comparison of hybrid architectures when trained on 50% the training data.

4.3 Final Model

Using the findings from the upstream and downstream task, a decision was made to use the CNN to LSTM model as the final model architecture in this project. Figure 4.9 shows a schematic model of this architecture.

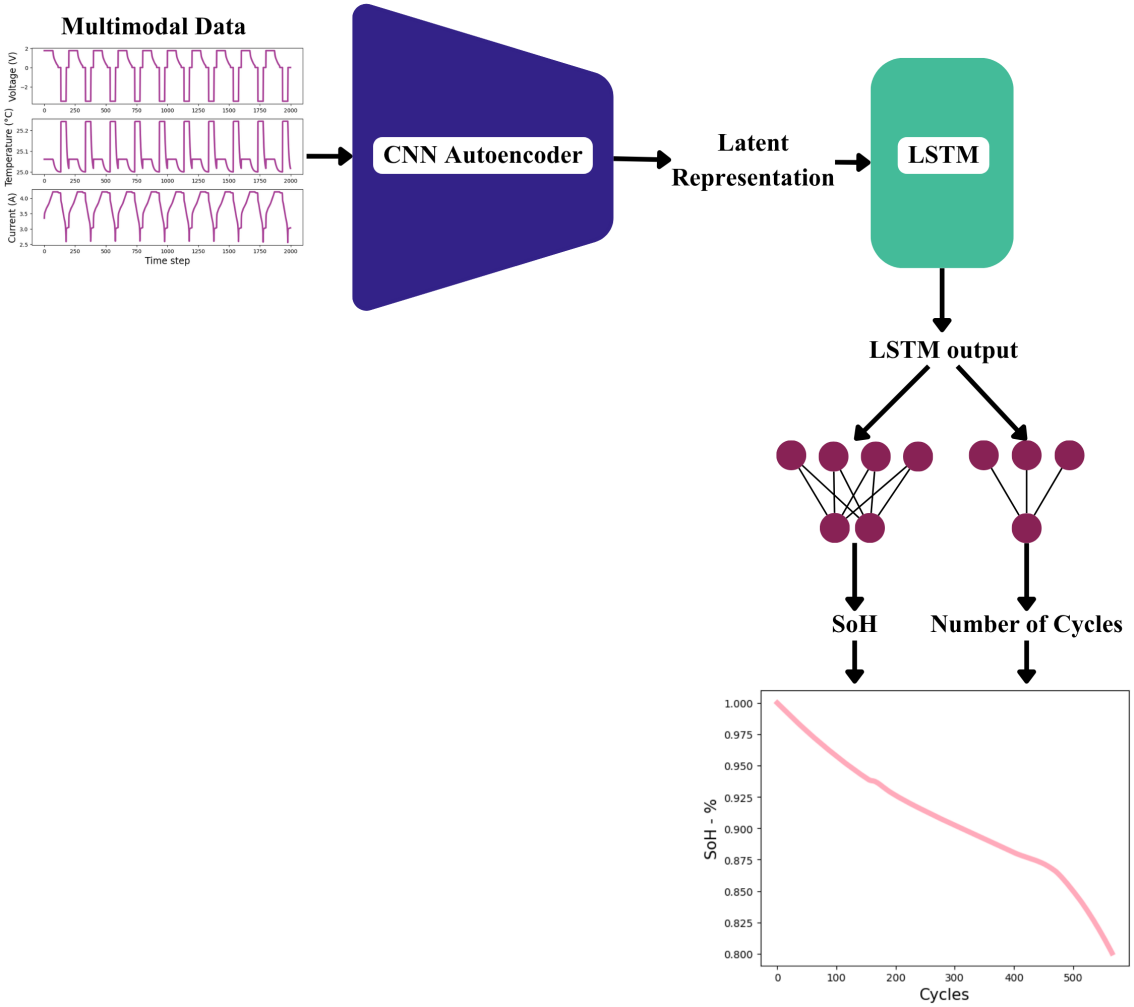


Figure 4.9: Schematic model of the final hybrid architecture consisting of the pretrained CNN encoder and the downstream LSTM model, which in turn consists of one LSTM layer and two MLP heads for SoH and number of cycles predictions.

Figure 4.10 shows the performance of the model over five different training / validation splits. This resulted in a mean RMSE of 0.03310 with a standard deviation of 0.001478 as well as a mean MAE of 0.02158 with a standard deviation of 0.00111. The different training / validations splits were selected so that the validation set was a subset of the upstream validation set, but the subset was different every iteration. All batteries not selected for the validation set were added to the training set.

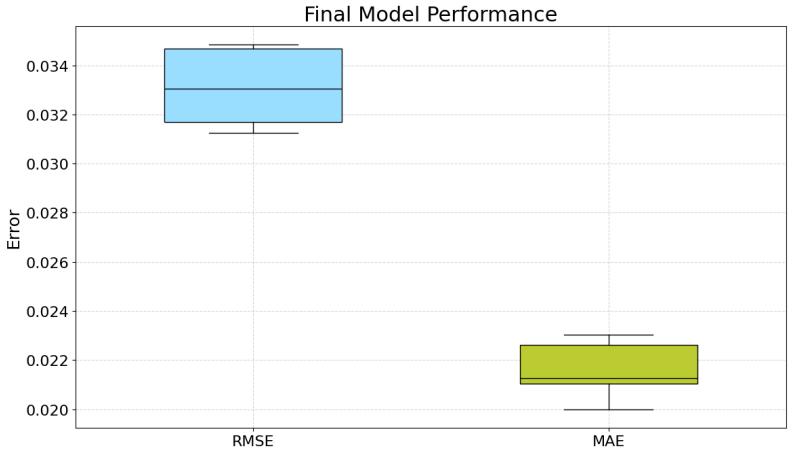
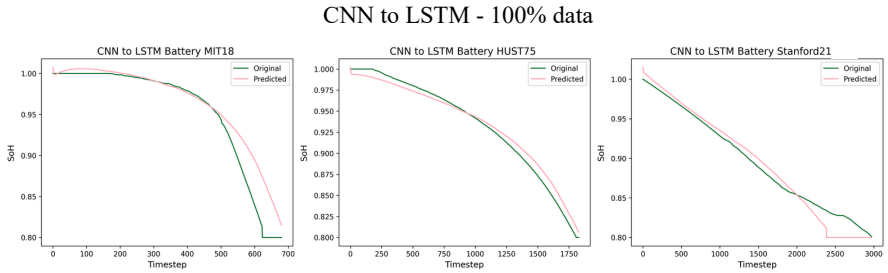


Figure 4.10: Model performance of the CNN to LSTM architecture on five different train/val splits.

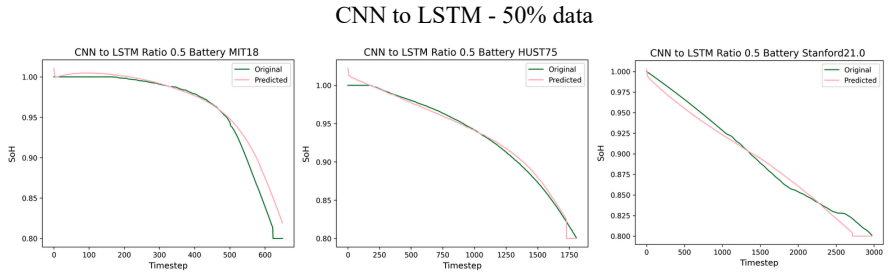
4.3.1 Lowering the Labeling Ratio

In the next step, the final model was trained on fractions of the training data spanning from 10% to 100% using five different training/validation splits. Different training/validations splits refer to the fact that the training and validation sets were chosen as different subsets of the full training and validation sets in different iterations.

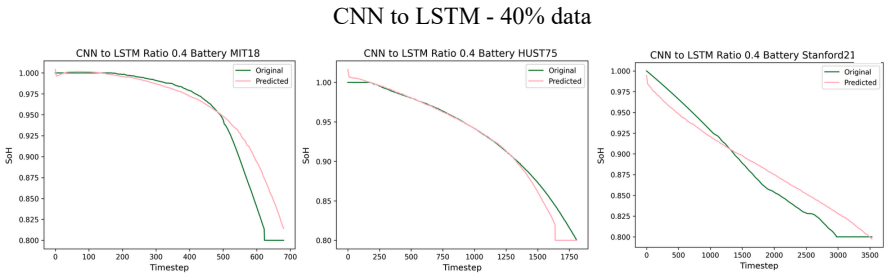
The results can be seen in Figure 4.12 and Table 4.4. An acceptable performance reduction was set at an RMSE increase by 25%. Figure 4.11 shows the corresponding SoH estimations and targets. Using only 50% of the training data gives a mean RMSE of 0.04047, which represents a 23% increase in RMSE compared to using 100% of the data. Looking at Figure 4.11a and 4.11c, we can see that this performance loss is not critical to the SoH estimations. Downsizing the training data further to 40% gives a mean RMSE of 0.04103%; a 24% increase from the original value. Figure 4.11c also shows slightly worse performance, specifically on battery 21 from Stanford, which has proven to be more difficult to estimate. The model performs similarly on the other batteries. When trained on only 30% of the data the model began producing estimations which are slightly less smooth but the overall trends are still followed nicely and the number of cycle error is considered acceptable. At a 20% labeling ratio, the model's estimations become oscillating, and the predetermined limit of a 25% increase in RMSE is surpassed. This shows that the model can safely be trained on 70% less labeled training data (corresponding to only 128 batteries) without significant performance reduction.



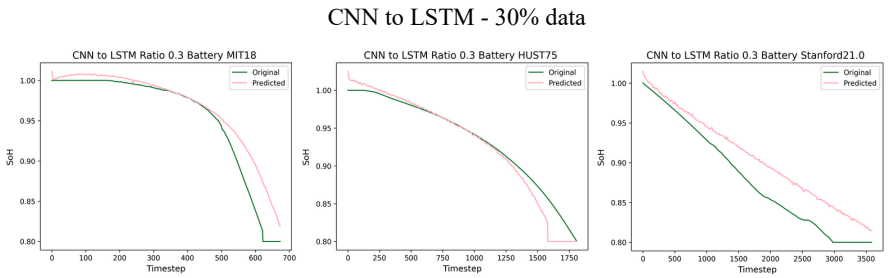
(a) CNN to LSTM trained on 100% of the training data.



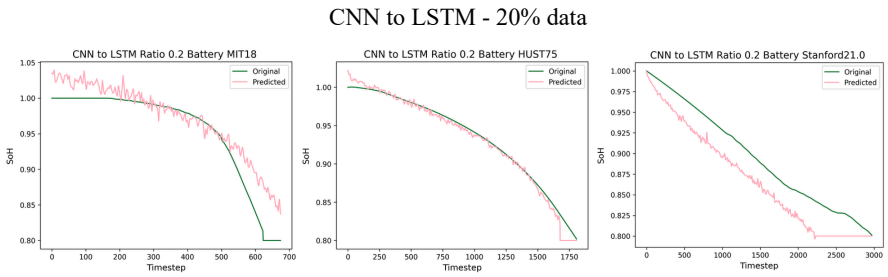
(b) CNN to LSTM trained on 50% of the training data.



(c) CNN to LSTM trained on 40% of the training data.



(d) CNN to LSTM trained on 30% of the training data.



(e) CNN to LSTM trained on 20% of the training data.

Figure 4.11: SoH estimations of three batteries for CNN to LSTM model, trained on different amounts of data.

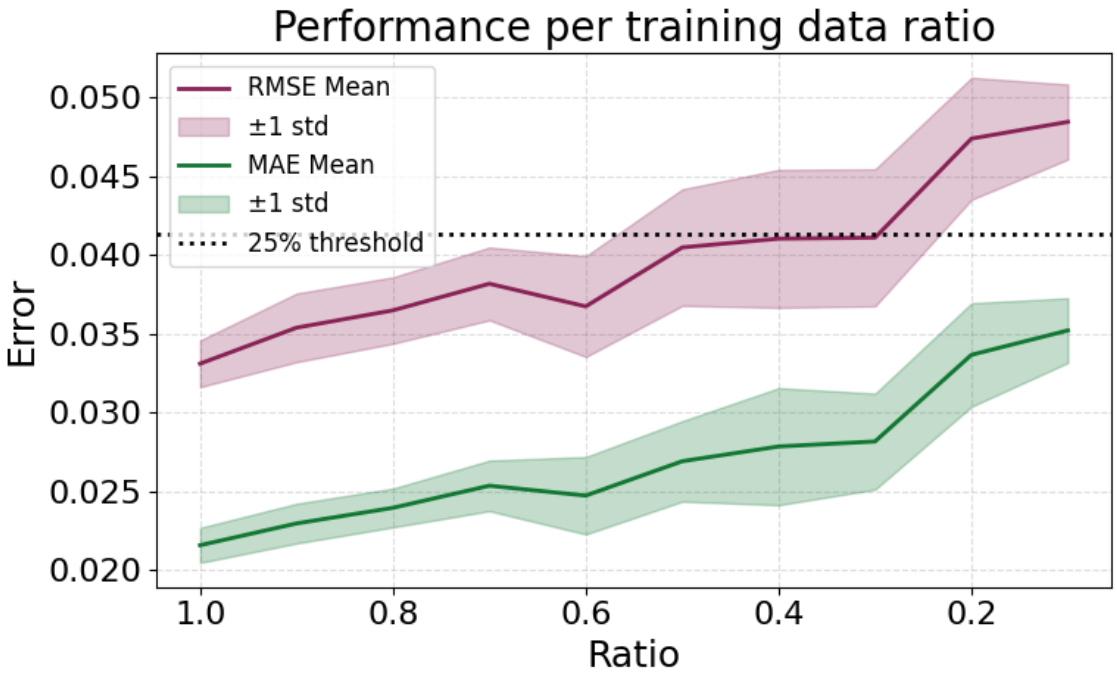


Figure 4.12: Model performance of the CNN to LSTM architecture when trained on different amounts of training data.

Table 4.4: Performance of the CNN to LSTM model when trained on different data fractions. The results show the average and standard deviation from five different training/validation splits.

Data Ratio	MAE	RMSE
100 %	0.02159 ± 0.001109	0.03311 ± 0.001479
90%	0.02297 ± 0.001259	0.03539 ± 0.002174
80%	0.02396 ± 0.001230	0.03649 ± 0.002107
70%	0.02536 ± 0.001602	0.03817 ± 0.002307
60%	0.02474 ± 0.002454	0.03674 ± 0.003194
50%	0.02692 ± 0.002552	0.04047 ± 0.003695
40%	0.02785 ± 0.003711	0.04103 ± 0.004372
30%	0.02817 ± 0.003044	0.04110 ± 0.004344
20%	0.03367 ± 0.003282	0.04738 ± 0.003867
10%	0.03521 ± 0.002046	0.04844 ± 0.002376

4.3.2 Model Robustness

In Figures 4.13 and 4.14 the results from different robustness tests can be seen. Figure 4.13 displays how the model performs when randomly selected data points are set to 0. The number of data points that were skewed corresponded to a certain percentage of the data. This was done to simulate how the model would react to a

faulty sensor.

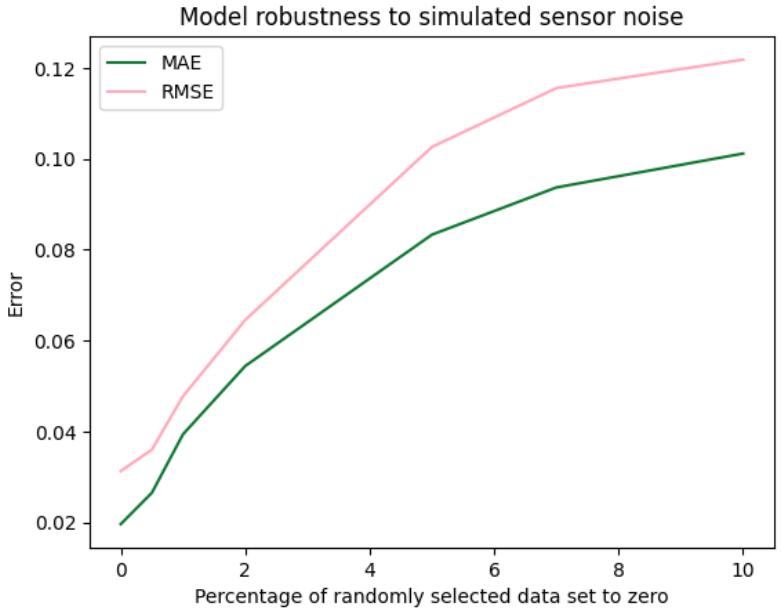


Figure 4.13: Model performance after replacing a certain percentage of the validation data with zeros.

In Figure 4.14 a Gaussian noise was added to one of the input features. The Gaussian noise had a standard deviation corresponding to the standard deviation of the feature. This was done to see which of the features is the most sensitive to noise.

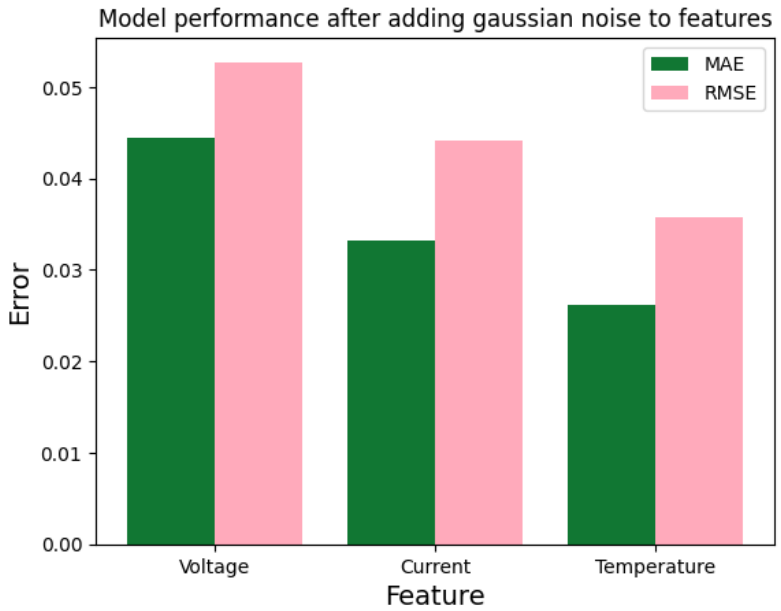


Figure 4.14: Model performance after gaussian noise with standard deviation equal to the standard deviation of the feature was added to the feature.

4.3.3 Comparison of the Model With and Without Autoencoder

Figure 4.15 shows a performance comparison between the CNN to LSTM hybrid architecture and the downstream LSTM model without the upstream autoencoder. The plots show the mean RMSE and MAE values over five different train/val splits. The hybrid architecture consistently performs better than the solo LSTM model. The difference in performance grows as the labeling ratio gets smaller. The solo LSTM model crosses the performance threshold already at a labeling ratio of 0.6.

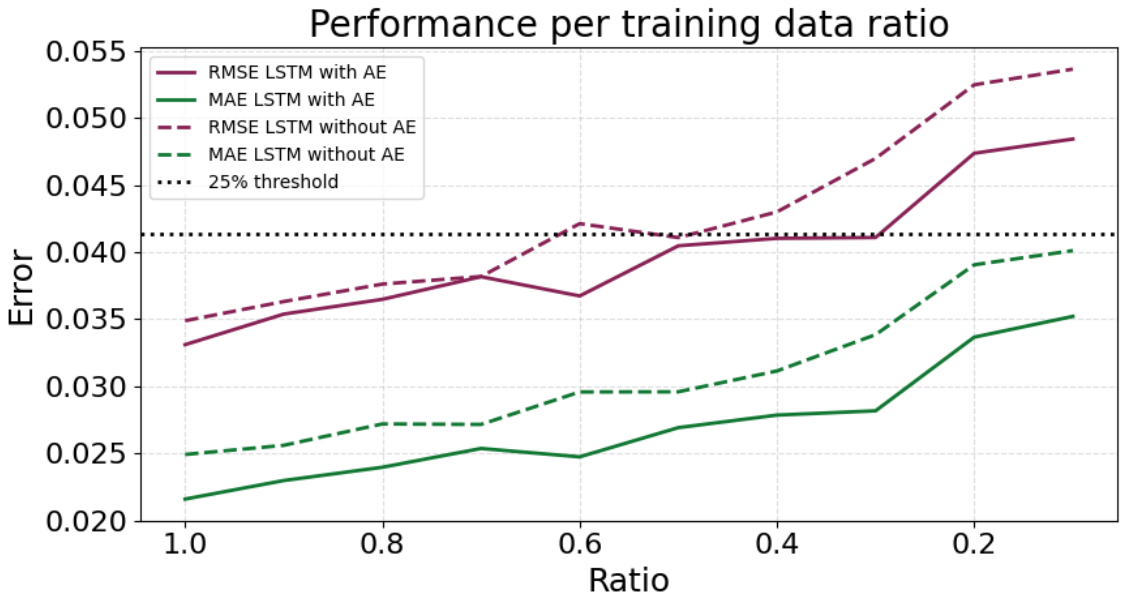


Figure 4.15: Comparison of the model performance with and without the upstream autoencoder when trained on different amounts of training data. The RMSE and MAE values represent the mean values over five different train / val splits.

4.4 Early SoH Prediction Fine-tuning Task

Fine-tuning the final CNN to LSTM model on data that only reaches 95 % SoH led to the following results. This training set, in contrast to the downstream training set, contained fleet data. Figure 4.16 shows SoH estimations for three batteries. The model performs well on the first battery, but struggles slightly more with the MIT shapes than it did in the downstream task. The model gives a worse prediction for the EV fleet batteries.

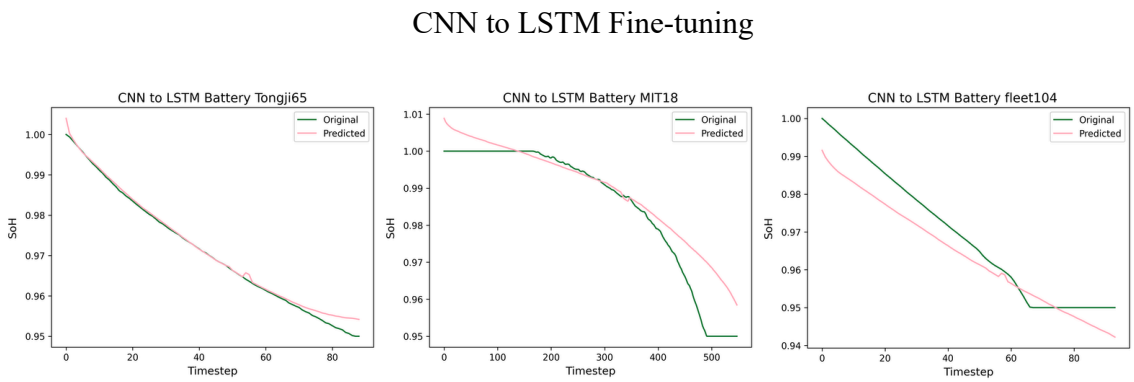


Figure 4.16: Examples of SoH estimations and targets from the early SoH prediction task.

Figure 4.17 shows the performance of the model on a different EV fleet battery. This battery shows unexpected degradation, very unlike the degradation from the lab-data. Looking at the figure it is clear that the model struggles to estimate this pattern.

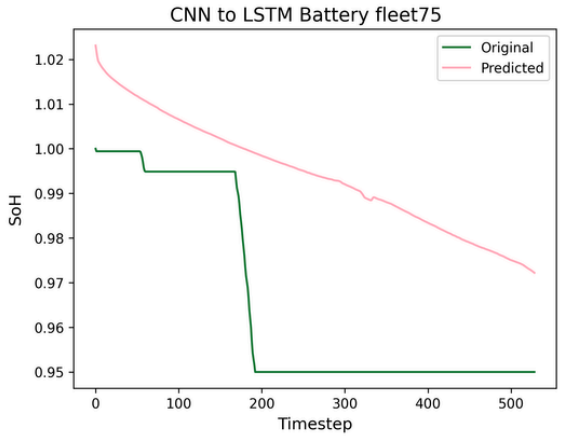


Figure 4.17: SoH estimation and target on an EV fleet battery with irregular degradation.

Table 4.5: Performance of the fine-tuned model when tested on different data. The results for the field and lab data are the average and standard deviation from five different train/validation splits.

Data	**MAE**	**RMSE**
Total	0.0076 ± 0.0003	0.01040 ± 0.0003
Field Data	0.0174 ± 0.0026	0.0210 ± 0.0027
Lab Data	0.0070 ± 0.0001	0.0094 ± 0.0002
Test Data	0.01220	0.01727

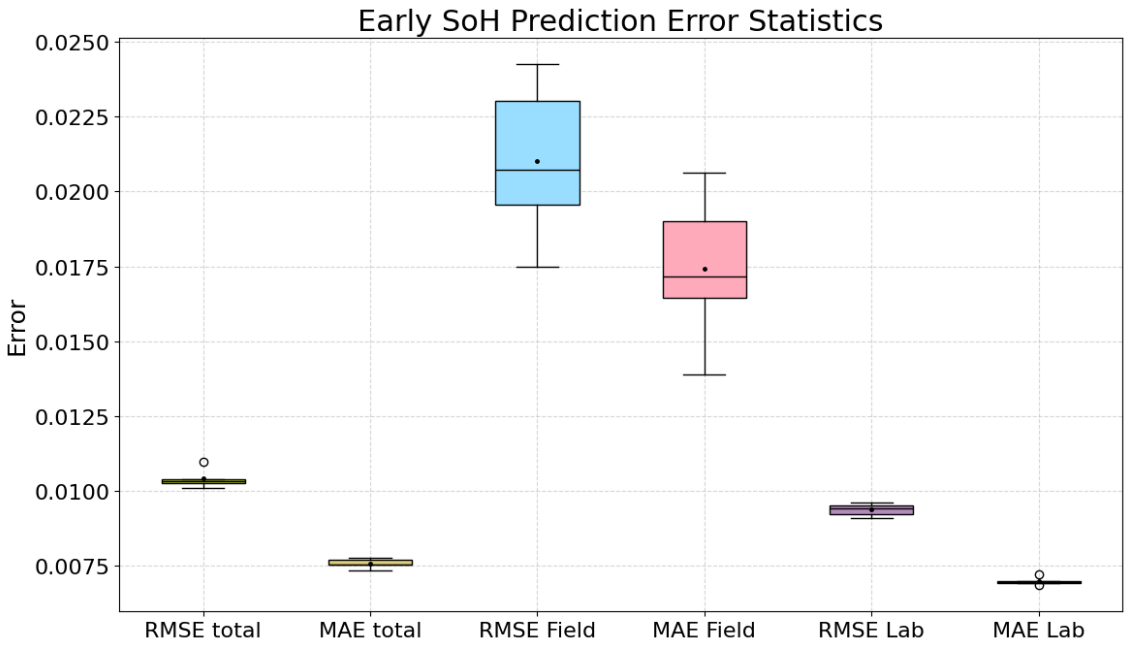


Figure 4.18: Prediction error statistics for lab and field data over five different train/test splits.

Table 4.5 and Figure 4.18 show the RMSE and MAE of the model when fine-tuned on the 95% SoH training set over five different train/test splits. It is clear that the model performs much better on lab data than field data, which is to be expected. Furthermore, the model is more stable on lab data. Worth noting is that shifting the target to 95% SoH instead of 80% skewed the error scale and hence these results are not to be compared with the results from the upstream task.

Figure 4.19 shows how the final models perform on three batteries from the test set before and after finetuning. One battery is from the MIT dataset, one battery is from the Turku dataset (which the model has never seen before), and one battery is from the EV fleet dataset. Both the fine-tuned and the non-fine-tuned (here also referred to as downstream) models follow the shape of the MIT curve well. The fine-tuned model only estimates the battery's SoH until around 95% and the downstream model predicts below 82.5%, which is in line with how the models were trained. The fine-tuned model performs much better on the unseen Turku dataset than the downstream model. The fine-tuned model also captures the fleet data much better than the downstream model. The downstream model does not predict degradation from 1.0 to 0.8 like it is trained to do.

In Table 4.5 the RMSE and MAE on the test set can be seen. The RMSE and MAE lie between the total and field data errors. This was calculated after cutting the targets of the test data batteries reaching lower than 95% SoH at 95 % and then comparing to the predictions. This was performed in the same way as for the validation data, so rescaling and achieving even lengths by padding with 0.95. Note that this does not correspond to what is seen in Figure 4.19, where the target is neither cut nor padded.

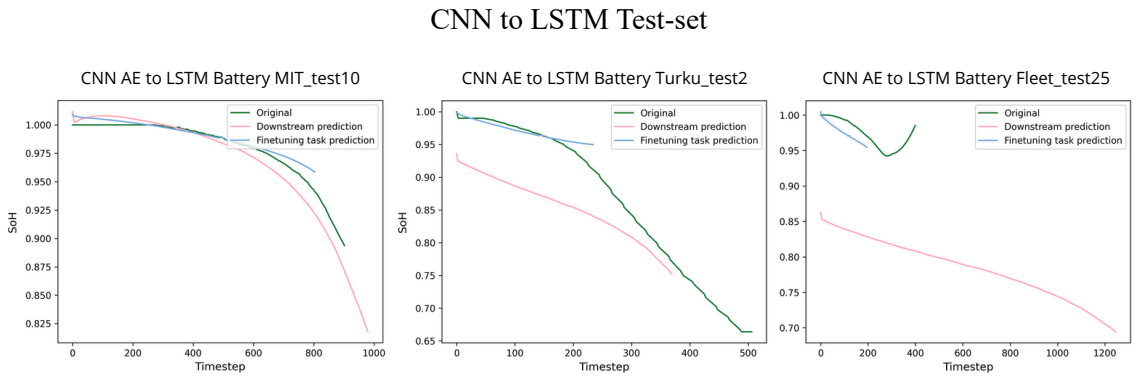


Figure 4.19: SoH estimations and targets for the model before and after the fine-tuning task tested on the test set.

5
Discussion

In this section, a discussion of the project will be presented as well as limitations and future work.

5.1 Discussion

In this section, the results will be discussed and analyzed.

5.1.1 Upstream Task

The autoencoders showing the best performance in the upstream task were the CNN autoencoder and the Transformer autoencoder. The CNN autoencoder was evaluated since it has been shown to be good at capturing local patterns in data. Since the input data consists of 10 cycles, the patterns that impact the degradation might be found in each cycle rather than the degradation over all 10 cycles. This could explain why the CNN performs well in this task.

The Transformer is overall a powerful architecture that performs well in tasks that are relatively complicated and when there are large amounts of data. While this is not entirely the case for this task, the large amount of data is probably enough for the Transformer autoencoder to perform well. The attention mechanism might also work especially well for the cyclic input data with both local and global patterns.

The three recurrent autoencoders in the upstream task show very similar results, which is likely because the model architectures are similar. The models were prone to overfitting and did not show as good results as the Transformer and convolutional autoencoders. This might be because the 2000 timesteps in the input data are too many, or have local and global patterns that are hard for the model to recognize. Another reason for the poor performance is that the model is fed voltage, temperature and current data simultaneously, which could be too much and too diverse data for the model to have optimal results. Due to the overfitting issues, the recurrent models needed to be kept at a smaller size than the other models, which were kept to very similar sizes.

Lastly, the MLP autoencoder has the worst performance among the tested archi-

tectures. This is likely a result of the model being simple and not capturing the cyclic patterns as well as some of the other models. This can be seen in Figures 4.2, 4.3 and 4.4 where the reconstructions are visualized. The MLP autoencoder shows very poor performance on battery 65, which is a field battery, but also poorer performance for the current and temperature reconstructions compared to the other models. The architecture is simply not as powerful as the other architectures.

All autoencoders tested perform worse on the field data than on the laboratory data. The reason for including the field data in the upstream task was to mitigate the final finetuning task. However, if the model was to be used solely on lab data, the performance on the lab data would be better since the field data is more difficult to reconstruct.

5.1.2 Downstream Task

The autoencoder's performance on the upstream task did not directly correspond to their performance on the downstream task. This shows that a strong reconstruction is not necessarily representative of a strong and informative latent space. This can be seen in the Transformer autoencoder, which showed the best upstream performance, but not the best results among all downstream models, as can be seen in Tables 4.1 and 4.2. The fact that the Transformer autoencoder does not show as good results in the downstream task as the upstream task could be due to a few different reasons. One of them is that the latent space is not as strong as in the MLP and the CNN. A Transformer is always a type of autoencoder, consisting of an encoder and a decoder, but without the decompression to the latent space that a vanilla autoencoder has. To enforce this, a bottleneck structure was added between the encoder and decoder to get the decompression to the latent space of size 128 that was wanted. Because of this, the risk is that even though the autoencoder is good at reconstructing, the bottleneck structure is not optimal for creating an informative latent space. This is also one of the reasons the MLP autoencoder was used as a benchmark, since the linear compression can show a more informative latent space than the much more complicated Transformer autoencoder could.

Worth noting is also that the best performing autoencoder is not the same for all downstream models. Both the MLP, CNN and Transformer autoencoders result in good accuracies depending on which downstream models they are combined with. This could be since they create the latent space in different ways that are easier or harder to convert to SoH series for different models. Also interesting is that combining an autoencoder with the same structure in the downstream model shows good performance for all three autoencoders, possibly due to similarities between the downstream model and the encoder. The best performing autoencoder for the downstream MLP is the MLP autoencoder and for the downstream CNN the CNN autoencoder performs the best. The same goes for the Transformer, where the Transformer autoencoder performs the best. This could be since the autoencoders create a latent space that is easier to capture by the downstream models. The information

might be compressed in a way that the downstream model is able to decode more easily than when the autoencoder has a different structure from the downstream model.

The results in Table 4.2 also show that the MAE and RMSE relate very well. The model combinations with low MAE also have a low RMSE which indicates that the models are trained to be overall good and do not produce any large outliers or encounter any input data that they are not able to handle at all.

In Figure 4.6, the results for the reconstruction of a sample of models are displayed. Apart from that the MLP to MLP model is very oscillating, what often sets the different models' performance apart is the number of cycle predictions. For many of the tested models, the shape of the degradation curve for the different lab samples was easy to predict. What sets the CNN to LSTM model apart is that the number of cycles prediction as well as the curve prediction being accurate.

One thing that could be causing the inaccurate cycle predictions is that the loss function during training is not optimal. The way of combining errors as in equation 3.2 is common in the field. And even though several α values were tested before deciding on 0.05, it is possible that some other α factor would give better results, or that there is some even more sophisticated way of combining the loss that was not discovered. In Figure 4.5, however, it can be seen that the cycle prediction errors correspond well to how the overall errors look. The lowest cycle prediction errors are found for the CNN autoencoders as well as for the LSTM models. The largest errors are found for the Transformer autoencoders and the MLP models, which correspond well to the overall results.

For the number of cycles prediction task, we can see that the difference between the MAE and RMSE is proportionally larger than for the full task MAE and RMSE. This indicates that there are outliers in the number of cycles impacting the results. This seems reasonable, since the batteries in the dataset have very varying numbers of cycles, from around one hundred to over four thousand, until EoL is reached. When rescaling the targets to the correct number of cycles, not that much penalty is added since the shorter curve is padded with 0.8, resulting in this inaccuracy being better captured in the number of cycles error. This can also be seen in Figure 4.6, where the scale on the x-axis varies a lot.

The training data downsizing task can be seen in Figure 4.12. The plots in Figure 4.11 show the performance of the model on different training data fractions. The plots show that the effect on the error is subtle up to a 70 % decrease where the cycle predictions and SoH curve reconstructions are slightly worse. When the lower percentages are reached, the curve starts slightly oscillating. This pattern can also be seen in Figure 4.6a, where the MLP autoencoder combined with the MLP downstream model gives similar oscillating results. The reason that the MLP to MLP model performs like this is probably that the model is not complex enough to learn all the information in the data. The final model, on the other hand, is probably too complex for the very small dataset that it is trained on in the final steps of the

downsizing task, which is why the oscillating behavior occurs. The wiggly SoH prediction is, apart from the high error, not preferred since it is physically not possible for the battery to gain capacity after it has degraded. Therefore, the conclusion is that the amount of labeled training data can be reduced to somewhere around 30% of the initial amount, while maintaining model performance.

5.1.3 Final Model

When testing the final model on the test set, the performance differs quite a lot based on which dataset the battery originates from. The model performs well on batteries belonging to MIT, which is reasonable since MIT is found in both the upstream and downstream training sets. The performance on the unseen Turku and Berlin datasets is worse, probably because the cycling protocol is DD instead of CC as in most of the downstream training data, but also because the datasets are completely unseen. The performance on the fleet dataset is poor, which is expected since field data is not found in the downstream training set and is relatively different from the lab data in the downstream training set. It is interesting that the model fails to predict a curve starting at 1 and ending at 0.8, which indicates that the input data in the lab and field datasets are very different. The performance on the test set creates a good benchmark to evaluate the performance of the model after fine-tuning. Furthermore, it should be noted that the test set is not clipped at any specific SoH, compared to the training set, which always ends at 80% to 82.5%. The reason that an accurate RMSE or MAE can not be calculated on the test set is that the field data and a lot of the Berlin and Turku data do not reach below 82.5%. This results in that either the error would have to be calculated on targets that are not comparable to the training targets, or that the error would be calculated on a subset of the data that does not reflect the distribution of the test set as a whole.

As seen in Figure 4.15, the performance of the final model is better than an LSTM without the pretrained encoder. This shows that the pipeline with the unsupervised upstream model adds value, not only by better performance but especially when trying to lower the labeling ratio. The 25% RMSE higher performance threshold is reached quicker without the encoder in the pipeline. This indicates that the proposed architecture does in fact allow for less labeled data to be used.

The robustness test in Figure 4.13 shows that the model is not especially robust to noise. Modifying only 1% of the data leads to an error about double as large as not modifying the data. 1% of the data corresponds to 20 data points of the 2000 total data points being compromised, which is not an unrealistic amount of error. There is likely already a bit of noise, irregularities and outliers in the data that were not effectively removed by the data preprocessing, resulting in the model probably already adapting to noise instead of the pure ground truth. When introducing additional noise, the model is not able to effectively handle this. When adding Gaussian noise to the different features, as seen in Figure 4.14, it can be shown that the error impacts the voltage feature the most. This could be an incentive to be even more

thorough when preprocessing the voltage data compared to the temperature and current data.

5.1.4 Early SoH Prediction Fine-tuning Task

Fine-tuning enabled the model to learn how to perform early SoH estimations of the EV fleet dataset, whilst still showing adequate results on the lab-data that it was trained on in the earlier tasks. The way the training datasets were designed with overlap means that during fine-tuning, the model sometimes got fed the same input data as in the downstream task but with a new target. This, in turn, means that the fine-tuned model learns to predict SoH until around 95% while the downstream model always predicts until around 80% SoH. This explains the differences seen in Figure 4.19. Though the fine-tuned model does not estimate the full target for any of the test batteries, it can still be considered a good estimation given the circumstances.

In Figure 4.19, the performance of the fine-tuned model on the Turku and field samples is much better than that of the non-fine-tuned downstream model. The performance on the field data is explained by the fact that the downstream model is not trained on the field data while the fine-tuned model is. This shows that fine-tuning is a way to be able to predict early degradation when full SoH data until EoL is unavailable. Even though the predictions on the field data were not optimal, the domain shift as seen in the right part of 4.19 can show that this method is a good start to fully bridge the gap between lab and field data. The fact that the fine-tuned model performs well on the Turku dataset could indicate that the RC protocol creates data more similar to the field data than the lab data, which highlights the need to create datasets containing diverse cycling protocols when creating models intended to work well on both lab and field data. This is also supported by the MAE and the RMSE on the test set, as seen in Table 4.5, which lies in between the MAE and RMSE of the full validation data and the field batteries in the validation data. This indicates that the model performs worse on unseen datasets than on seen datasets but better on the unseen datasets than on field data.

An alternative method would be to keep the downstream and fine-tuning training sets completely separate. This might then lead to a model with the ability to differentiate between the datasets and hence know when to predict which length SoH curve, giving better results on the test set. However, the purpose of the fine-tuning task was to bridge the gap between the two different data types, lab and field. It was not to train a model to know which batteries have which length of SoH curve, since this would be a manufactured feature for training purposes. Furthermore, keeping some lab data in the fine-tuning training set might have helped the model remember the main shapes of these SoH curves and hence prevented forgetting. Therefore, the choice to overlap the two training sets can be seen as sound.

Furthermore, the EV Fleet dataset has, as explained in Section 3.2, undergone ex-

tensive preprocessing in order to make it suitable for the task. However, the final SoH curves are still of varying quality. Figure 4.17 shows the estimation and target SoH curves of a battery from the EV fleet dataset. It is clear to see that this SoH curve does not follow the same trends as the others, and therefore the fine-tuned model struggles to estimate it correctly. In Table 4.5, we see that the fine-tuned model on average performs worse on the batteries from the EV fleet dataset. This is probably due to poor-quality field data. Having better field data might lead to better overall performance of the fine-tuned model, not only on the EV fleet batteries but also on the others since their training is affected by EV fleet data in the training set as well.

The choice to only use the first 10 cycles of voltage, current and temperature data as input was mainly based on studies that had shown good results on lab data. This data is of course periodic and hence it makes sense that the beginning could be used to represent the full sequence. However, the cars from the EV fleet dataset have real-life drivers who don't necessarily follow the same driving patterns throughout the car's life cycle. Therefore, ten charging cycles as input might be insufficient when estimating the SoH of the EV fleet batteries. This could, together with the quality issues explained above, explain why the model struggles more with this data than the lab data.

5.2 Limitations

Many of the limitations of this project are data-related. One such limitation is that the data preprocessing is quite extensive and uses a lot of cleaning methods, some more complicated than others. The most important features in the raw data could be in something that is filtered out as an outlier or is lost when reshaping the data. Of course, the data needs to be processed in multiple ways to get something useful and usable out of it, but one limitation is that due to the scope of the work, this is not done as thoroughly as it needs to be to gain optimal quality of data given the available datasets. This can also be seen in the results, where the robustness test displayed in Figure 4.13 shows a drastic increase in the error if only 1% of the data is noisy.

Another limitation lies in the fact that that the field data only comes from one source. Even though the number of batteries is sufficient, the data is always impacted by method choices from the data collector. In this case, when the data comes from EV's, the data is also impacted by the BMS. So if the BMS used when collecting this data has some flaws or errors, the model is largely impacted by this. This error would be mitigated by having several sources of data.

The field data also includes many batteries with unnatural degradation, some of which were not filtered out during preprocessing. For example, battery fleet75 which can be seen in Figure 4.17. Batteries like these have a strong impact on both error calculations and model training and hence better field data is very likely to strongly

improve the results of the finetuning task. Furthermore, the field data has very slow degradation. If field data that reached 80 % SoH could be used, the methodology of this study could be different since the 95 % SoH finetuning task would not be needed.

When training ML-models, apart from building the architecture, a lot of work lies in tuning the hyperparameters of the models. This was done by using the Optuna framework, which enables a structured way to keep track of the optimal parameters and try a lot of different combinations of parameters. The parameter selection is therefore also based on an element of randomness and a finite number of combinations is evaluated, which could result in the optimal combination never being evaluated. However, by using Optuna instead of manually varying the parameters, a lot more combinations can be evaluated in a shorter amount of time and the bias from manually deciding which parameters to try is removed.

5.3 Future Work

The AI/ML field is evolving rapidly, so the possibilities of future work and improvements are endless. Here are some potential areas of improvement that have been identified during the thesis work.

One feature that greatly affects the degradation of batteries is the battery chemistry. The goal for this thesis was to create a multi-chemistry dataset so that the model would be robust to different degradation patterns. Another way to approach this problem with degradation patterns being affected by the battery chemistry is to create separate models for different cell chemistries. By doing this, the model could be trained on just one chemistry, hopefully creating better predictions on that chemistry. However, this was not done due to the aim of the project being focused on a multi-chemistry dataset, but could be used to improve predictions in a different scenario.

Another possibility to adapt to this would be to e.g. use cell chemistry as an input to the model. The cell chemistries could be converted to labels that could be inserted into the latent space of the upstream autoencoder. By adding information about cell chemistry into the latent space the model could possibly separate different behaviors in the batteries caused by chemical and physical attributes of the batteries. This was experimented with but ended up not being used in the final pipeline, since it did not show large model performance improvement. However it is possible that using a slightly different approach to implement the cell chemistry label would improve the performance.

Even though the used dataset is large in comparison to the standard in the field, the dataset could always be larger, more diverse and contain even more field data. If there would exist a field dataset containing a large amount of batteries with degradation past 80%, the possibilities when building the models and bridging the gap between the field and lab data would be completely different.

Due to the scope of the project, the decision to only predict one degradation metric was made. The pipeline could probably be adjusted to instead predict Remaining Useful Life (RUL) or State-of-Charge (SoC) with good accuracy given there would be suitable data preprocessing pipelines available to extract useful input data and targets for those predictions.

Another possibility to further bridge the gap between field and lab data would be to reevaluate the early cycle prediction method. Using the first 10 cycles to predict SoH/EoL is common in the field, but the results supporting this were mainly evaluated on lab datasets. Since the field data has a slower degradation, having more input cycles or even working with different input like averages or standard deviations might be a more informative input for the models.

6
Conclusion

The aim of the project was to construct a pipeline containing a hybrid architecture to estimate State-of-Health (SoH) for batteries in electric vehicles. By combining a self-supervised upstream autoencoder with a supervised downstream model, the amount of labeled data could be lowered, achieving semi-supervised learning. Furthermore, the gap between the lab data and the field data was bridged by the proposed pipeline, contributing to the novelty of the project.

The comparison between different autoencoders and machine learning models showed that a convolutional autoencoder combined with an LSTM showed the best results, even though other combinations also showed similar performance. The performance of the autoencoder in the upstream task was not directly linked to the performance in the downstream task, indicating that the combination of models is more important to evaluate than the performance of the models by themselves.

To bridge the gap between lab and field data, a fine-tuning task for early SoH estimation was added. This resulted in a model that could perform sufficient prediction of the degradation of field data batteries. When using the proposed pipeline, the amount of labeled data could be significantly reduced. This could be especially important when working with field data, which is expensive to collect.

Even though the training data originated from many sources and contained many different battery cell chemistries, testing on a completely unseen data set is always a hard task. The difference between different data sets is large, which highlights the importance of using training data as representative as possible of the data that will be used in practice. The key to creating a well-performing and stable model is to focus on the data quality and data engineering, because if this is not sufficient, the models will never gain full potential.

Bibliography

[1] International Energy Agency. World Energy Outlook 2025. Technical report, 2025.

[2] Alessandro M. Ralls, Kaitlin Leong, Jennifer Clayton, Phillip Fuelling, Cody Mercer, Vincent Navarro, and Pradeep L. Menezes. The Role of Lithium-Ion Batteries in the Growing Trend of Electric Vehicles, 9 2023.

[3] Kitalu Ricin Ngoy, Valantine Takwa Lukong, Kelvin O. Yoro, John Beya Makambo, Nonso Christopher Chukwuati, Chinedu Ibegbulam, Orevaoghene Eterigho-Ikelegbe, Kingsley Ukoba, and Tien Chien Jen. Lithium-ion batteries and the future of sustainable energy: A comprehensive review, 11 2025.

[4] Zhongbao Wei, Zhongyi Quan, Jingda Wu, Yang Li, Josep Pou, and Hao Zhong. Deep Deterministic Policy Gradient-DRL Enabled Multiphysics-Constrained Fast Charging of Lithium-Ion Battery. *IEEE Transactions on Industrial Electronics*, 69(3):2588–2598, 3 2022.

[5] Wenbin Song, Di Wu, Weiming Shen, and Benoit Boulet. A Remaining Useful Life Prediction Method for Lithium-ion Battery Based on Temporal Transformer Network. In *Procedia Computer Science*, volume 217, pages 1830–1838. Elsevier B.V., 2022.

[6] Fujin Wang, Zhi Zhai, Zhibin Zhao, Yi Di, and Xuefeng Chen. Physics-informed neural network for lithium-ion battery degradation stable modeling and prognosis. *Nature Communications*, 15(1), 12 2024.

[7] Yunhong Che, Yusheng Zheng, Xin Sui, and Remus Teodorescu. Boosting battery state of health estimation based on self-supervised learning. *Journal of Energy Chemistry*, 84:335–346, 9 2023.

[8] Tianyu Wang, Zhongjing Ma, Suli Zou, Zhan Chen, and Peng Wang. Lithium-ion battery state-of-health estimation: A self-supervised framework incorporating weak labels. *Applied Energy*, 355:122332, 2 2024.

[9] Song Zhang, Yannan Li, Jinpeng Tian, Zhihong Man, Chi Yung Chung, and Weixiang Shen. Improving Battery Life Prediction With Unlabeled Data: Confidence-Weighted Semi-Supervised Learning With Label Propagation. *IEEE Transactions on Transportation Electrification*, 11(2):5938–5949, 2025.

[10] Bowen Zheng, Zhichao Deng, Zhenhao Luo, Shuoyuan Mao, Minggao Ouyang, Xuebing Han, Hewu Wang, Yalun Li, Yukun Sun, Depeng Wang, Yuebo Yuan, Liangxi He, Zhi Yang, and Yanlin Zhu. A comprehensive review of lithium-ion battery modelling research and prospects: in-depth analysis of current research and future directions. *Applied Energy*, 401, 12 2025.

[11] Pedro H. Camargos, Pedro H.J. dos Santos, Igor R. dos Santos, Gabriel S. Ribeiro, and Ricardo E. Caetano. Perspectives on Li-ion battery categories for electric vehicle applications: A review of state of the art, 10 2022.

[12] Hongao Liu, Chang Li, Xiaosong Hu, Jinwen Li, Kai Zhang, Yang Xie, Ranglei Wu, and Ziyou Song. Multi-modal framework for battery state of health evaluation using open-source electric vehicle data. *Nature Communications*, 16(1), 12 2025.

[13] Oscar E. Rojas and Muhammad A. Khan. A review on electrical and mechanical performance parameters in lithium-ion battery packs, 12 2022.

[14] Alexis Geslin, Le Xu, Devi Ganapathi, Kevin Moy, William C. Chueh, and Simona Onori. Dynamic cycling enhances battery lifetime. *Nature Energy*, 10(2):172–180, 2 2025.

[15] Mohsen Heydarzadeh, Teemu Toivola, Victor Vega-Garita, and Eero Immonen. Dataset of lithium-ion cell degradation under randomized current profiles for NMC, NCA, and LFP chemistries. *Data in Brief*, 60, 6 2025.

[16] Jakob Schneider, Thomas Kröger, Nikolaos Wassiliadis, Manuel Ank, Marcel Rogge, Jan Schöberl, Philipp Rosner, Cristina Grosu, Andreas Jossen, and Markus Lienkamp. Understanding lithium-ion battery degradation in vehicle applications: Insights from realistic and accelerated aging tests using Volkswagen ID.3 pouch cells. *Journal of Energy Storage*, 112, 3 2025.

[17] Pengzhi Li, Yan Pei, and Jianqiang Li. A comprehensive survey on design and application of autoencoder in deep learning, 5 2023.

[18] Keiron O'Shea and Ryan Nash. An Introduction to Convolutional Neural Networks. 12 2015.

[19] Danhua Zhou, Zhanying Li, Jiali Zhu, Haichuan Zhang, and Lin Hou. State of Health Monitoring and Remaining Useful Life Prediction of Lithium-Ion Batteries Based on Temporal Convolutional Network. *IEEE Access*, 8:53307–53320, 2020.

[20] Zhong kai Feng, Jing shuai Zhang, and Wen jing Niu. A state-of-the-art review of long short-term memory models with applications in hydrology and water resources, 12 2024.

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. 8 2023.

[22] Bhaskar Saha and Kai Goebel. NASA Battery Dataset, 2007.

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

[24] Fujin Wang. XJTU Battery Dataset, 2023.

[25] Ye Yuan, Guijun Ma, and Songpei Xu. The Dataset for: Real-time personalized health status prediction of lithium-ion batteries using deep transfer learning, 2022.

[26] Jiangong Zhu, Yixiu Wang, Yuan Huang, R. Bhushan Gopaluni, Yankai Cao, Michael Heere, Martin J. Mühlbauer, Liuda Mereacre, Haifeng Dai, Xinhua Liu,

Anatoliy Senyshyn, Xuezhe Wei, Michael Knapp, and Helmut Ehrenberg. Data-driven capacity estimation of commercial lithium-ion batteries from voltage relaxation , 2022.

[27] Bruis van Vlijmen, Vivek N. Lam, Patrick A. Asinger, Xiao Cui, Joachim Schaeffer, Alexis Geslin, Devi Ganapathi, Shijing Sun, Patrick K. Herring, Chirranjeevi Balaji Gopal, Natalie Geise, Haitao D. Deng, Henry L. Thaman, Stephen Dongmin Kang, Steven B. Torrisi, Amalie Trewartha, Abraham Anapolsky, Brian D. Storey, William E. Gent, Richard D. Braatz, and William C. Chueh. Aging matrix visualizes complexity of battery aging across hundreds of cycling protocols. *Energy and Environmental Science*, 18(13):6641–6654, 5 2025.

[28] Jiaqi Yao. Lithium-Ion Battery Drive Cycle Dataset, 8 2024.

[29] Jiahuan Lu, Rui Xiong, Jinpeng Tian, Chenxu Wang, Chia Wei Hsu, Nien Ti Tsou, Fengchun Sun, and Ju Li. Battery degradation prediction against uncertain future conditions with recurrent neural network enabled deep learning. *Energy Storage Materials*, 50:139–151, 9 2022.

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

[31] Josue Obregon, Yu Ri Han, Chang Won Ho, Devanadane Mouraliraman, Chang Woo Lee, and Jae Yoon Jung. Convolutional autoencoder-based SOH estimation of lithium-ion batteries using electrochemical impedance spectroscopy. *Journal of Energy Storage*, 60, 4 2023.

[32] Wennian Yu, Il Yong Kim, and Chris Mechefske. Analysis of different RNN autoencoder variants for time series classification and machine prognostics. *Mechanical Systems and Signal Processing*, 149, 2 2021.

[33] Fadhila Lachekhab, Messouada Benzaoui, Sid Ahmed Tadjer, Abdelkrim Bensmaine, and Hichem Hamma. LSTM-Autoencoder Deep Learning Model for Anomaly Detection in Electric Motor. *Energies*, 17(10), 5 2024.

DEPARTMENT OF PHYSICS
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY