Self-Supervised Masked Autoencoding Meets Federated Learning for Electric Vehicle Battery State-of-Health Estimation

Self-Supervised Masked Autoencoding Meets Federated Learning for Electric Vehicle Battery State-of-Health Estimation

By Mohanad Ismail,

A Thesis Submitted to the School of Graduate Studies in the Partial Fulfillment of the Requirements for the Degree Master of Applied Science

McMaster University

Master of Applied Science (2025)

Hamilton, Ontario (Department of Mechanical Engineering)

TITLE: Self-Supervised Masked Autoencoding Meets Federated Learning for Electric Vehicle Battery State-of-Health Estimation

AUTHOR: Mohanad Ismail (McMaster University)

SUPERVISOR: Dr. Ryan Ahmed

NUMBER OF PAGES: xiv, 137

Abstract

Electric Vehicles (EVs) live and die by their batteries. To keep drivers safe and confident in their vehicles, we need efficient, accurate, and private ways to track each battery's State-of-Health (SoH). *But*, EV labelled data is scarce, sharing raw data raises privacy flags, and big models strain on-board hardware. This thesis tackles all three problems through a two-step remedy in one shot.

1. **Learn data representations without needing labels:** Each car trains a small autoencoder to reconstruct its own collected sensor data after randomly hiding parts of the signal.

2. **Share knowledge, not data:** Instead of uploading the raw collected data, every car sends only its trained model parameters to a remote cloud server. The server aggregates parameters from all cars and sends the improved model back.

Four simple questions guide our work:

- Does this usage of unlabelled data improve the model's performance?
- How much of the signal should be hidden to get the best representation learning?
- What is the optimal strategy for incorporating the limited labelled data available into the model?
- Does this aggregation of separately trained models hurt accuracy compared with a fully centralized approach?

Our experiments show a 17% lower average Mean Absolute Error (MAE), with up to a 60% improvement in the best cases, when we make use of the available unlabelled data versus training exclusively on labelled data. Hiding 30-40% of signals strikes the

balance between challenge and clarity. Finally, aggregation of models on average stays within 0.05 Ah of centralized training, virtually no loss, with zero raw-data exposure.

This thesis incorporates cloud computing, Self-Supervised Learning (SSL), and Federated Learning (FL) to present a light, privacy-friendly pipeline for fleet-wide SoH estimation, evidence that unfrozen fine-tuning outshines frozen variants, the first systematic look at how masking ratio shapes battery time-series representation learning, and practical proof that sharing model weights instead of data keeps accuracy basically untouched and privacy intact.

Acknowledgements

بِسمِ اللَّهِ الرَّحمٰنِ الرَّحيمِ

وَمَا بِكُم مِّن نِّعْمَةٍ فَمِنَ ٱللَّهِ ۖ ثُمَّ إِذَا مَسَّكُمُ ٱلضُّرُّ فَإِلَيْهِ تَجْـَٔرُونَ ﴿٥٣﴾

سورة النحل - آية ٥٣

Whatever blessings you have are from Allah. Then whenever hardship touches you, to Him alone you cry for help . (53)

Qur'an - Surah An-Naḥl 16 : 53

All praise is for Allah—Lord of all worlds, the Most Compassionate, Most Merciful, and peace and blessings be upon our prophet Muhammad, his family, and his companions. Anything that I have ever, or will ever achieve is first and foremost due to Allah's guidance and support.

This will probably not be your average acknowledgement section, in that it will be longer than usual. But I believe this is necessary, seeing the immense amount of love and support I got from those around me during the time of my studies.

To my mom, you are the reason I am here today and writing down these words. You were always by my side from the moment I took my first breath, to taking my first steps, to not comprehending I have to be away from you for a few hours every day for school, to exhausting training and preparations for karate until my world championship, to the very, *very* long, consuming, and decisive final year of high school, to persevering through the difficult years of my undergrad, to the heavy weight of me moving my whole life to the other side of the planet, and now, to writing the few final words in my master's thesis. I owe every single thing I achieved to, after Allah Almighty, you always being there and having faith in me. These few words cannot express my gratitude towards having you as my mother. To many more achievements, with you by my side.

To my friends, I believe every one of you deserves to be named, and I hope I do not miss anyone, but forgive me if I do. I will go over each one of you, in alphabetical order:

Abdalla, thank you for being the friend always ready to support and cheer up everyone of us in our group. Our papers, *and sanity*, owe you so much.

Ahmed, my first friend in a new unlocked part of the world map, we have seen it all together. Thank you for your support during some of the days where I felt most lost and doubted my life decisions. These two years will not have been the same without you.

Andrea, you kept me sane and going throughout my whole second year of this degree. Most of the work and writing in this thesis was done during our work sessions, and you made sure to keep them fun and lighthearted. I am grateful for having such a fun and kind soul in my circle during these times. Thank you for constantly supporting me and brightening up my day.

My dear undergrad friends, Hazem H., Hisham, Karim, Marina, Mario, Mohamed E., Mohamed M., Mo'men, even though each one of us went their own way in life, that did not stop you from constantly checking up on me and seeing how life has been treating me. Your support has kept me going, and made me feel loved at times where I most needed it. I am lucky that I got to cross paths with every one of you. Thank you from the depth of my heart.

Hazem S., thank you for being the friend in our group we turn to when the going gets tough. I hope you finally manage to get your paper done before Sunday's deadline (does not matter which Sunday).

Mohab, the idea that this whole thesis is based on came up during one of our walks in the neighbourhood. I wanted to preface with this to show how much our time together means to me. You were there through the ups and downs, and you have seen my darker

sides, your support never changing, and your belief in me pushing me to become my best self every day. Your friendship is a true blessing that I will never take for granted. Thank you for the support, laughs, consoling, and encouragement. Thank you for everything.

To my family, thank you for your constant encouragement and prayers. Seeing your smiling faces on my phone screen kept me moving forward.

To my lab mates at the CMHT, thank you for every single time you stepped out of your way to help me understand a concept, figure out why something is not working, or simply cheering me up. You made the days pass faster and easier.

Last, but surely not least, I would like to dedicate my thanks and gratitude to my supervisor, Dr. Ryan Ahmed. His mentorship and guidance throughout the whole process before I even set foot in Canada has significantly shaped this piece of work that I present to you today.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context and Motivation

Electric Vehicles (EVs) are increasingly viewed as a cornerstone technology for achieving global decarbonization targets. Worldwide EV sales exceeded 14 million units in 2024 and are projected to reach one-third of all light-duty vehicle sales by 2030 [1]. Behind this momentum lies the battery pack, whose performance and safety degrade with age and misuse. Accurately tracking a battery's State-of-Health (SoH) is therefore pivotal. Under-estimating SoH inflates range anxiety and warranty costs, whereas over-estimating SoH risks thermal runaway, vehicle downtime, and reputational damage for manufacturers. Modern Battery Management Systems (BMSs) already embed sophisticated estimation algorithms, yet three structural challenges still limit their effectiveness:

1. **On-board computational limits:** Edge processors inside vehicles struggle to run larger, more accurate data-driven models in real time.

2. **Data scarcity and imbalance:** Only a small portion of field data is labelled with ground-truth SoH, and ageing patterns differ drastically across chemistries,

climates, and driving styles.

3. **Privacy and heterogeneity:** Centralizing raw data for model training is inhibited by commercial sensitivity, data-protection regulation, and non-uniform fleet sensor data.

1.2 Problem Statement

The above constraints create a gap between the theoretical performance of SoH algorithms demonstrated in controlled laboratory studies and the reliability required for mass-market EV deployment. Bridging this gap demands methods that (i) harvest knowledge from the vast amount of *unlabelled* field data, (ii) learn collaboratively without transferring proprietary datasets, and (iii) capitalize on scalable cloud resources while preserving real-time inference at the vehicle edge.

1.3 Research Gaps

Current literature addresses each challenge in isolation: cloud-based frameworks alleviate compute bottlenecks [2], Self-Supervised Learning (SSL) extracts representations from unlabelled cycles [3], and Federated Learning (FL) offers privacy-preserving model aggregation [4]. However, no prior work has *integrated* SSL and FL within an operationally viable cloud architecture for SoH estimation, nor has it systematically quantified how design choices such as the SSL masking ratio or fine-tuning strategy affect predictive accuracy and communication cost on real EV data.

1.4 Research Aim and Questions

Aim: This thesis aims to develop and evaluate an integrated SSL-FL framework that improves SoH estimation accuracy while respecting on-board compute limits and data-privacy constraints.

Research Questions (RQs).

RQ1 Does pre-training with masked auto-encoding on unlabelled drive cycles enhance SoH prediction compared with purely supervised baselines under limited labelled data?

RQ2 How does the choice of *masking ratio* in SSL influence representation quality and downstream SoH accuracy?

RQ3 How do different fine-tuning strategies for the SSL encoder (fully frozen, full unfreeze) affect model performance and training stability?

RQ4 Does FL aggregation for model parameters degrade performance compared with centralized training?

1.5 Hypotheses

H_0. SSL pre-training followed by FL aggregation provides no statistically significant improvement in SoH estimation over supervised centralized baselines.

H_1. SSL pre-training combined with FL aggregation *does* yield a significant improvement, with the effect size modulated by the masking ratio and fine-tuning strategy.

1.6 Contributions

1. **Framework:** First end-to-end pipeline that unifies masked-autoencoder SSL with the FedAvg method of FL for EV SoH estimation.

2. **Masking-ratio study:** Analysis of how varying the SSL masking ratio affects representation quality, training stability, and downstream accuracy on real-world drive-cycle data.

3. **Fine-tuning strategy study:** Empirical comparison of frozen and fully unfrozen encoders and guidelines on when each is advantageous.

4. **Cloud-edge prototype:** A cloud architecture that offloads heavy training while ensuring fast and efficient on-board inference and data privacy preservation.

1.7 Thesis Organization

- Chapter 2 reviews the evolution from internal combustion to electric mobility and existing SoH estimation methods, including their advantages and current drawbacks.

- Chapter 3 surveys cloud-computing for battery state estimation in literature and industry and highlights the shortcomings of current offerings.

- Chapter 4 analyses self-supervised learning techniques for time-series data and motivates masked auto-encoding, and examines federated learning foundations and prior SoH applications.

- Chapter 5 details the proposed SSL-FL framework, dataset, and model architectures.

- Chapter 6 reports experimental results and compares them to baselines, and discusses findings and limitations.
- Chapter 7 summarizes findings and presents avenues for future work.

1.8 Significance and Impact

By fusing elastic resources of the cloud, self-supervised representation learning, and the privacy-preserving nature of federated training, the proposed framework shows more accurate SoH estimates without breaching data sovereignty or overwhelming vehicle CPUs. In practical terms, this translates to longer battery lifetimes, safer operation, and reduced total cost of ownership.

Chapter 2

Literature Review

2.1 Introduction

This chapter starts by covering the transition from Internal Combustion Engines (ICEs) to Electric Vehicles (EVs). A historical review (Section 2.2) shows the motivation for the technological shift, followed by an assessment of the environmental and economic benefits and the persistent market barriers to large-scale EV adoption (Section 2.3). We then narrow the focus to lithium-ion battery ageing, introduce the two state variables that comprise battery management, *State-of-Charge (SoC)* and *State-of-Health (SoH)*, and explain why reliable in-field SoH estimation is both difficult and essential (Section 2.4). Finally, the same section surveys the three principal SoH estimation paradigms (direct, model-based, data-driven) and highlights the open problems that motivate this thesis: high on-board computational load, data scarcity, and privacy concerns.

2.2 Historical Context and Evolution of Automotive Propulsion

The ICE has long been one of the main pillars of modern transportation. Its origins trace back to the late 19th century, with main developments occurring around 1886 when pioneers such as Karl Benz and Nikolaus Otto introduced gasoline-powered vehicles and four-stroke engine designs, respectively [5], [6]. Over the subsequent century, ICE technology was refined to become the dominant propulsion system in automobiles, trucks, and machinery, facilitating rapid industrialization and global mobility through increased speeds and extended travel ranges.

The pros of ICE-based vehicles include mature and widespread fuel infrastructure, high energy density of fossil fuels, and economical production methods that have enabled mass-market adoption for over 100 years [5], [6]. However, the cons are significant. ICE vehicles have relatively low thermal efficiency, constrained to approximately 20% energy conversion efficiency [7], which translates into higher fuel consumption per kilometre. Furthermore, their reliance on fossil fuels has resulted in considerable environmental side effects, including emissions that significantly contribute to climate change [8], [9].

In response to these issues, there is a growing recognition of the need to transition toward a more efficient and greener alternative. EVs have demonstrated energy usage efficiencies nearing 60% from grid electricity to wheel power, compared to the approximate 20% efficiency of ICEs [7]. This transition is also supported by technological advancements in battery chemistry, as well as governmental mandates and policies such as Canada's planned phase-out of ICE sales by 2035 [10].

EVs possess a storied evolution that dates back to the mid-1800s. In fact, early EVs were among the fastest automobiles of their era, holding land speed records until around 1900 despite their inherent limitations in range [11]. During the late 19th century,

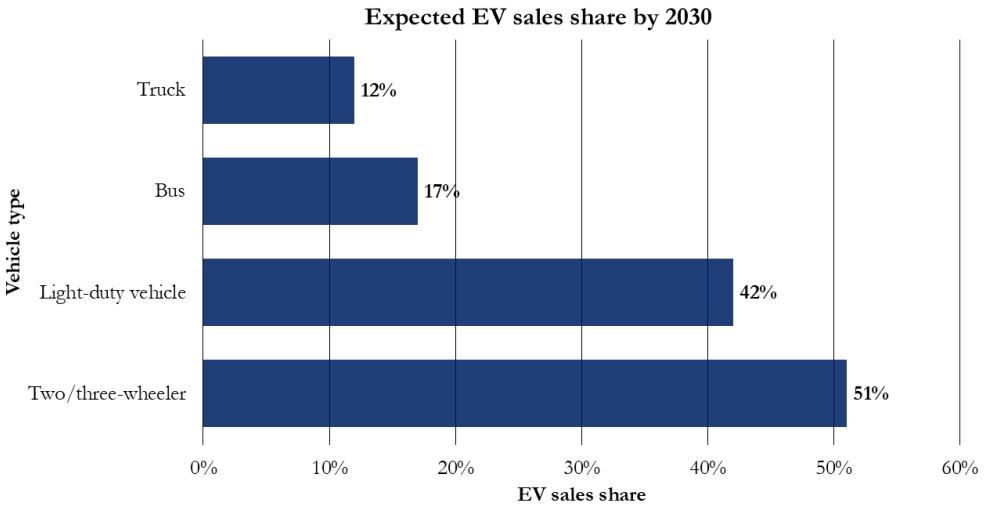

Figure 2.1: Share of EV sales by vehicle type by 2030

EVs enjoyed a period of significant appeal owing to their quiet and smooth operation. However, the advent and mass production of ICE vehicles in the early 20th century relegated EVs to niche status. Their performance and range being hindered by battery technologies of the time did not help either [12].

Renewed interest in electric mobility emerged in the late 20th century when the development of lithium-ion batteries in the 1990s enabled improvements in energy density and longevity [13]. From 2011 onward, the global EV market has experienced exponential growth. By the early 2020s, the state of the EV market had transformed dramatically. Global EV sales exceeded 3 million units annually as key markets in Europe, the United States, and China embraced a transition to cleaner mobility solutions [14]. Contemporary EV models now casually achieve ranges of over 300 km per charge and benefit from fast-charging capabilities [14]. Current projections indicate that by 2030, in some regions, EVs could constitute more than 35% of new vehicle sales, driven by both declining battery costs and policy frameworks incentivizing clean energy transitions [12], [14]. Fig. 2.1 breaks down the EV sales share expected by the year 2030 by vehicle type [1].

2.3 Electric Vehicles: Benefits and Market Barriers

2.3.1 Environmental and Economic Advantages

The transition from ICEs to EVs is driven by multiple advantages spanning environmental, energy efficiency, economic, and grid integration domains. Starting with the green benefits, EVs eliminate tailpipe emissions, leading to reductions in local air pollutants (NOx, $PM_{2.5}$) and greenhouse gases (GHG) when compared to ICEs [15]. Well-to-wheel lifecycle assessments confirm that battery EVs produce lower GHG emissions than ICEs, especially in regions with low-carbon electricity generation [15], [16]. Moreover, emissions-responsive charging strategies can further reduce lifetime GHGs by aligning charging with periods of higher renewable output [17]. In urban settings, the removal of tailpipe emissions leads to improved air quality and attendant healthcare cost savings [18]. However, the net environmental gain depends critically on the carbon intensity of the electricity mix and local charging patterns. In areas with high-emission grids, benefits may be less noticeable [19].

Focusing on energy efficiency, EVs achieve higher conversion efficiencies, often exceeding 80% from battery to wheels, whereas typical ICEs convert only 20-30% of fuel energy into traction [20]. Regenerative braking further recovers kinetic energy during deceleration [20]. As a result, EVs can realize two-fold or greater improvements in energy efficiency per kilometre driven relative to ICEs [21].

The higher energy efficiency of EVs, coupled with lower unit electricity costs versus gasoline, leads to significant per-kilometre fuel savings [22]. The simpler mechanical architecture of electric drivetrains also reduces maintenance requirements and unscheduled downtime, driving down total cost of ownership over vehicle lifetimes [23]. When societal side effects, such as reduced healthcare expenditures from cleaner air, are taken into consideration, net economic benefits increase further [24]. Nonetheless, the shift to

EVs can decrease gasoline tax revenues. A recent example is the pausing of the consumer federal carbon tax in Canada. This requires alternative fiscal measures to sustain transportation funding [25].

Far from being passive loads, EV batteries represent distributed energy storage capable of providing supplementary services to the grid. Vehicle-to-grid (V2G) and vehicle-to-building (V2B) schemes allow aggregated EV fleets to engage in peak shaving, frequency regulation, and reactive-power support [26]. By scheduling charging during off-peak hours and discharging when demand is high, EV owners can realize bill savings and revenue streams, while system operators benefit from reduced peak loads and improved stability [27]. One thing that comes to mind when considering this is: How is this going to affect the vehicle's battery? Well, modelling studies demonstrate that large-scale V2G deployment can deliver spinning reserves and support the integration of variable renewables without compromising battery health when coordinated optimally [28].

2.3.2 Persistent Market Barriers

On the other hand, the adoption of EVs is slowed down by several challenges. The main factor mentioned throughout literature that contributes to consumer hesitation in adopting EVs is "range anxiety," which refers to the fear that an EV will run out of battery power before reaching a charging station. Pesch et al., for instance, emphasize that range limitations compared to ICEs remain a primary impediment affecting consumer choices and overall market growth [29].

Adding insult to injury, Olayode et al. note that the availability (or rather unavailability) of charging stations, restricts EV usage in regions with sparse infrastructure, intensifying range anxiety further [30]. The development of adequate charging infrastructure is crucial for boosting EV adoption. Numerous studies highlight the lack of sufficient charging stations as a major concern [31]–[33]. Morrissey et al. argue that

fast-charging networks could significantly boost sales, yet the initial investment and projected low adoption rates make widespread implementation economically challenging in the short-to-medium term [33]. The findings by Benmouna et al. reinforce that establishing a reliable network of charging stations is essential for user confidence in transitioning to electric mobility [34].

The initial purchase price of EVs remains a notable barrier. While battery prices have been decreasing, they continue to be higher than their ICE counterparts, deterring many consumers [35], [36]. Economic incentives from governments can mitigate this barrier by making EVs more financially appealing [37]. Nevertheless, Pesch et al. suggest that despite potential future cost reductions, current expenses and economic viability remain significant roadblocks [29].

Since range anxiety can be considered as the major barrier facing EV adoption, the work in this thesis aims to target it by giving users more peace of mind regarding their EV battery health. The more confidence we have in our methods of estimating the battery's states, the less hesitation users have in moving over to the electric side. The next section discusses the main EV battery states, how batteries age with time, and the problems we face in estimating these states.

2.4 Battery States and Estimation Techniques

2.4.1 Key Battery State Metrics

The SoC and SoH are fundamental metrics in battery management for EVs. Fig. 2.2 visualized the difference between them. SoC is defined as the ratio of the battery's available capacity to its nominal capacity, reflecting the remaining energy and directly informing driving range and charge control strategies. Common SoC estimation techniques include voltage-based open-circuit voltage mapping, Coulomb counting, and dynamic observer models.

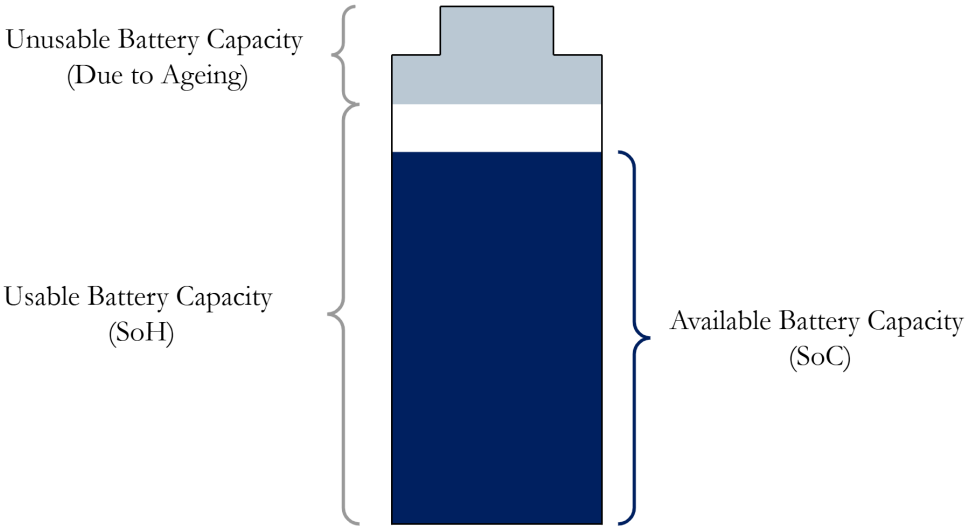

FIGURE 2.2: SoH refers to the available usable battery capacity, while SoC refers to the available capacity with respect to the nominal capacity.

Conversely, SoH quantifies capacity degradation and impedance growth relative to the initial state, serving as an indicator of ageing and performance decline. Accurate SoH assessment supports prognostics, safety management, and replacement scheduling by measuring the loss of usable capacity and power capability over time.

Inaccurate estimation of SoC and SoH pour directly into the range anxiety problem. If they are underestimated, then the vehicles are not pushed to their full potential. It can also result in premature replacement calls for batteries, which leads to unnecessary material waste. On the other hand, overestimation gives users a false sense of optimism, which can end up leaving them stranded without a charging station nearby in the end. Moreover, as batteries age (in other words, as the SoH decreases), EV batteries become less stable and predictable. In some incidents, this instability can lead to thermal runaways and spontaneous fires in the battery, risking the lives of EV users.

Our work directly targets these issues by providing a framework to improve the accuracy of SoH estimation. In the next subsection, we review the different SoH estimation methods, their advantages and their shortcomings, and how our work contributes to

addressing said shortcomings.

2.4.2 SoH Estimation Techniques

Accurate and low-cost SoH monitoring is indispensable for safe fast-charging, warranty decisions and second-life allocation of EV batteries. The SoH estimation techniques can be broadly categorized into three distinct approaches: direct methods, model-based methods, and data-driven techniques. Each approach has unique characteristics, advantages, limitations, and applicability, particularly concerning computational demands and sensor requirements. Table 2.1 compares between the pros and cons of the three approaches.

Direct Methods

Direct methods for SoH estimation typically involve the measurement of key battery parameters such as voltage, current, and temperature. These methods provide immediate insights into the battery's condition but often lack the capacity to generalize across different operational circumstances.

Coulomb Counting: This technique calculates the SoH based on the integral of the current over time, often requiring initial capacity information for accuracy. The equation is mathematically expressed as:

$$Q = \int_0^t I(t)dt \tag{2.1}$$

where Q is the charge, and $I(t)$ is the instantaneous current input. This is straightforward, but it is prone to measurement errors and cumulative drift over time due to self-discharge and inaccurate initial condition references [38].

Table 2.1: High-level comparison of the three principal State-of-Health (SoH) estimation paradigms.

Technique	General advantages	General drawbacks
Direct	• Immediate, sensor-level insight into battery condition • Low algorithmic complexity & minimal on-board compute • Hardware already present in most BMSs (current shunt, voltage taps)	• Limited ability to generalize across usage patterns or chemistries • Sensitive to noise, drift and cumulative measurement error • Some methods (e.g. OCV) require rest periods that vehicles rarely experience in service
Model-based	• Physically interpretable: parameters map to real ageing phenomena • Good accuracy under known operating ranges • Can fuse multiple sensor channels via Kalman or particle filters	• Depend on accurate cell parameters and ageing-dependent re-tuning • Performance degrades outside calibrated temperature/load windows • Requires extensive knowledge of internal battery reactions
Data-driven	• Capture complex, non-linear ageing signatures without explicit physics • Scalability: once trained, inference is fast and sensor-agnostic	• Heavy training-time compute: requires large, diverse datasets • Performance can vary across chemistries and duty cycles without re-training • Security and privacy concerns when using cloud offloading

Voltage-Based Methods: These methods involve measuring the open-circuit voltage (OCV) after the battery is allowed to rest and using voltage thresholds to estimate SoH. The main equation here is derived from the Nernst equation, which relates voltage to the concentration of chemical species within the battery [39]. These methods cannot be used with operating vehicles, since the rest period for the battery is not guaranteed.

Electrochemical Impedance Spectroscopy (EIS): EIS measures the impedance of a battery over a range of frequencies, allowing for the assessment of internal resistance changes, which correlate with ageing and capacity loss. The impedance can be expressed as:

$$Z(j\omega) = R + \frac{1}{j\omega C} \tag{2.2}$$

where R is the resistance and C is the capacitance related to the battery's electrochemical properties [40]. These methods can provide detailed insights but require sophisticated equipment, making them less feasible for on-board applications.

Model-Based Methods

Model-based methods employ mathematical representations of battery behaviour to estimate the SoH, often simulating the electrochemical processes involved.

Equivalent Circuit Models (ECMs): The battery is modelled as a combination of ideal components (resistors, capacitors) to represent its dynamic behaviour. An example of this model can be denoted by the equation:

$$V_{\text{bat}} = V_{\text{oc}} - I \cdot R - \frac{Q}{C} \tag{2.3}$$

where V_{bat} is the battery voltage, V_{oc} is the open-circuit voltage, R is the internal resistance, and C represents capacitance [41]. While ECMs can accurately reflect small-signal behaviours, they require tuning and may not perform well under variable conditions.

Kalman Filters: Kalman filtering techniques are employed to estimate the battery state based on current and voltage measurements. The Kalman filter updates its predictions based on new observations [39].

Particle Filters: This filtering technique uses a set of particles to represent probable states of the system. It is particularly useful for non-linear systems and can incorporate dynamic changes in SoH. A simplified form of the particle filter can be summarized with the equations:

$$Xk = f(Xk-1, Uk) + Wk \tag{2.4}$$

$$Zk = h(Xk) + V_k \tag{2.5}$$

where Xk and Zk represent the state and measurements, respectively [42].

However, these methods require extensive knowledge of internal battery chemistry and reactions [38].

Data-Driven Methods

Data-driven approaches use machine learning algorithms and large datasets to improve estimation accuracy without detailed electrochemical modelling. Initial data-driven methodologies primarily utilized basic statistical techniques and linear regression models.

For instance, [43] demonstrated that simple regression models could provide a baseline for evaluating battery conditions based on historical data. However, these methods often suffered from limitations in accuracy when exposed to real-world data variabilities.

Techniques such as support vector machines (SVM) and neural networks began to receive attention as they capture complex nonlinear relationships. In 2019, Li et al. published a comprehensive review of various data-driven health estimation methods and highlighted the rising use of machine learning [43]. Alongside neural networks, methods using ensemble techniques, like AdaBoost, emerged, further enhancing forecast reliability in estimations [44]. Ref. [45] pointed out that attention mechanisms integrated within neural networks enhance predictive capacities, particularly in understanding temporal patterns of battery health during cycles. Furthermore, researchers such as [46] focused on incorporating Long Short-Term Memory (LSTM) networks to address the dependency of battery performance over time.

Despite these advances, data-driven approaches face challenges regarding high computational requirements, the necessity for extensive training datasets, and performance variability across different battery chemistries. As highlighted by Massaoudi et al., even state-of-the-art deep learning methods require accurate labelling and significant computational resources to perform effectively in various contexts [47]. Moreover, integrating domain knowledge into data-driven models remains a considerable difficulty.

Because of the high accuracy of data-driven approaches, and their independence of electrochemical battery knowledge, they were picked as the focus of our work. Our work aims to target three of these problems:

- High computational requirements: by offloading a significant portion of computation to be run on remote cloud servers, instead of on-board the vehicle.

- The need for extensive labelled datasets: by making use of the vastly available unlabelled datasets through Self-Supervised Learning (SSL)

- Security concerns regarding vehicle data collected: by keeping the data on the vehicle, and only communicating aggregations with the cloud server using Federated Learning (FL).

2.5 Conclusion

This chapter traced the path from early ICEs to today's EVs, showing how environmental pressures, policy mandates, and advances in lithium-ion technology have pushed the industry toward electrification. While EVs deliver clear efficiency and emission benefits, their widespread adoption is slowed by range anxiety, infrastructure charging gaps, and the still high price of battery packs. Because these barriers all hinge on battery reliability, accurate in-field estimates of SoC and, more critically, SoH are essential.

The survey compared direct, model-based, and data-driven SoH estimation methods. Direct and model-based approaches are lightweight and physically interpretable but struggle under real-world noise and diverse duty cycles. Data-driven techniques can capture those complexities, yet they demand extensive labelled datasets, substantial processing power, and strict safeguards for driver privacy. These three pain points define the practical gap that the remainder of this thesis aims to close.

The next chapters therefore turn to three complementary technologies: cloud computing to shoulder heavy training workloads, SSL to unlock the value of unlabelled fleet data, and FL to protect user privacy by keeping raw records on the vehicle. Together, they form the backbone of the combined framework introduced in Chapter 5, which will be tested experimentally in Chapter 6.

Chapter 3

Cloud Computing

3.1 Introduction

This chapter is based on a literature review journal paper that we published in 2024 [2]. Minor modifications were made to the original text to give context and to allow logical flow with the rest of the thesis.

Section 3.2 covers background information on cloud computing, its different service models, benefits, challenges, and recent developments. Section 3.3 reviews the work done in literature and the industry regarding cloud-based battery management systems. Section 3.4 lists the identified gaps in research and recommends areas of research for future work.

3.2 Cloud Computing Overview

Cloud computing, as defined by the National Institute of Standards and Technology (NIST), is a service model that provides easy, on-demand access to a variety of shared computing resources, like networks, servers, storage, and applications [48]. This model is

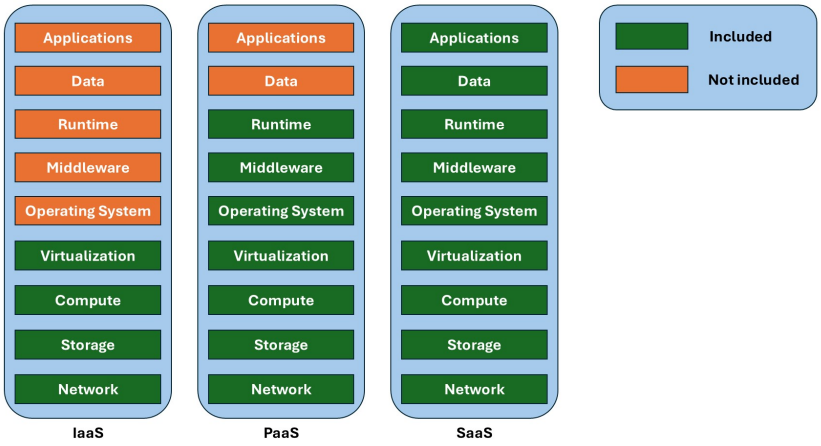

Figure 3.1: Services offered by IaaS, PaaS, and SaaS.

characterized by its ability to offer self-service provisioning of resources as needed, accessibility over a wide range of devices via the network, a multi-tenant environment where resources are shared among multiple users, the flexibility to scale services up or down based on demand, and a pay-per-use system that ensures resource usage is measured and transparent for both the provider and the consumer [48]. These fundamental attributes make cloud computing a dynamic and efficient way to handle computing needs.

Cloud computing offers different service models that cater to various user needs and requirements. The primary cloud service models are Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) [48]–[52]. These models provide distinct levels of abstraction and functionality, allowing users to access computing resources and services based on their specific needs. Fig. 3.1 visualizes a comparison between the different services offered by IaaS, PaaS, and SaaS. It can also be perceived as the separation between what the service model and the user manages.

1. IaaS: IaaS provides virtualized computing resources over the internet. Users can rent virtual machines, storage, and networking resources on a pay-as-you-go basis. This model offers flexibility and scalability, allowing users to manage and control

their infrastructure, including operating systems, applications, and data, while the cloud provider manages the underlying hardware.

2. PaaS: PaaS offers a platform that allows developers to build, deploy, and manage applications without the complexity of infrastructure management. Users can access development tools, databases, middleware, and other resources needed to develop and run applications. PaaS providers manage the underlying infrastructure, allowing developers to focus on application development and deployment.

3. SaaS: SaaS delivers software applications over the internet on a subscription basis. Users can access and use software applications hosted in the cloud without the need for installation or maintenance. SaaS applications cover a wide range of services, including email, collaboration tools, customer relationship management (CRM), and productivity software.

Despite its benefits, cloud computing faces several challenges and drawbacks that impact its adoption and effectiveness. Some of the key challenges and drawbacks facing cloud computing include:

1. Latency: One of the significant challenges in cloud computing is latency, especially for real-time applications. The distance between remote cloud servers and end-users can result in delays in data processing and communication, affecting the performance of time-sensitive applications [53]–[55].

2. Security and Privacy Concerns: Security and privacy issues remain a major drawback of cloud computing. Data breaches, unauthorized access, and lack of control over sensitive data stored in the cloud pose significant risks to organizations and users [53], [54], [56].

3. Reliability and Availability: Cloud services are susceptible to outages and downtime, impacting the availability of applications and services. Reliability concerns arise due to the dependence on third-party cloud providers for critical IT infrastructure [53]–[55].

4. Cost: While cloud computing offers cost-saving benefits, the overall cost of cloud services can escalate, especially for resource-intensive applications. Organizations may face unexpected expenses related to data storage, data transfer, and additional services [53], [54], [57].

5. Data Transfer Bottlenecks: As applications become more data-intensive, data transfer bottlenecks can occur, complicating data placement and transport within cloud environments. This can lead to increased costs and inefficiencies in data management [58].

6. Dependency on Internet Connectivity: Cloud computing heavily relies on Internet connectivity for data access and communication. Any disruptions in internet connectivity can impact the availability and performance of cloud services [54], [55].

7. Lack of Location Awareness: Cloud computing may lack location awareness, leading to challenges in data placement, resource allocation, and ensuring compliance with data sovereignty regulations [58], [59].

8. Limited Mobility Support: Cloud computing may face limitations in supporting mobile and Internet of Things (IoT) devices that require seamless mobility and connectivity. Lack of mobility support can hinder the effectiveness of cloud-based applications [54], [59].

9. Ethical and Legal Concerns: Cloud computing raises ethical and legal challenges

related to data ownership, data privacy, compliance with regulations, and the ethical use of data stored in the cloud [54].

With the wide appeal of cloud computing in different industries and use cases, intensive research has been driving further advancements to address the aforementioned issues and drawbacks. Some relevant advancements are as follows:

1. Edge Computing: Edge computing has emerged as a significant trend in cloud computing, enabling data processing closer to the source of data generation. This approach reduces latency, enhances real-time processing capabilities, and improves overall system efficiency [60].

2. Fog Computing: Fog computing has gained traction as a complementary approach to cloud computing, focusing on dispersing computing resources throughout the network's edge. This model addresses the limitations of traditional cloud computing by bringing computing resources closer to the data source, enabling faster data processing and analysis [60].

3. Hybrid Cloud Solutions: Organizations are increasingly adopting hybrid cloud solutions, which combine public and private cloud services. This approach offers greater flexibility, scalability, and data security, allowing organizations to leverage the benefits of both cloud deployment models [60].

4. Serverless Computing: Serverless computing, also known as Function as a Service (FaaS), has gained popularity for its ability to execute code in response to events without the need to manage servers. This model offers cost-efficiency, scalability, and simplified application development and deployment [61].

5. Green Computing: With a growing focus on environmental sustainability, green computing practices have become a key consideration in cloud computing. Efforts

to reduce energy consumption, optimize resource utilization, and minimize carbon emissions are driving advancements in eco-friendly cloud solutions [62].

6. Blockchain Integration: The integration of blockchain technology with cloud computing has introduced new possibilities for enhancing data security, transparency, and trust in cloud-based systems. Blockchain-based solutions are being explored to address data integrity and security challenges in cloud environments [63].

7. Data Security Enhancements: Advancements in cloud computing have led to improved data security measures, including enhanced encryption techniques, secure data storage solutions, and robust access control mechanisms. These developments aim to address data privacy concerns and ensure the confidentiality of sensitive information [64], [65].

8. IoT Integration: Cloud computing has played a crucial role in supporting the IoT ecosystem by providing scalable infrastructure, real-time data processing capabilities, and seamless connectivity for IoT devices. Cloud platforms are evolving to meet the demands of IoT applications and enable efficient data management and analysis [66], [67].

9. Multicloud Computing: Multicloud Computing strategies involve utilizing multiple cloud services from various providers to fulfill specific business needs, offering benefits like redundancy, performance optimization, cost savings, and enhanced security. This approach ensures operational continuity, allows for performance tuning based on workload requirements, enables cost-effective resource utilization, and provides robust security measures. Additionally, multicloud offers scalability and flexibility, allowing businesses to adjust resources according to demand and adapt to evolving needs without being tied to a single provider's limitations [68]–[71].

The merger of IoT and cloud computing brings forth critical improvements such as the ability to scale systems more effectively, superior processing power for handling vast amounts of data, greater storage capabilities, the provision of analytics in real-time, a reduction in operational costs, and the strengthening of security protocols [72]–[77]. One significant benefit is the scalability that cloud computing provides to IoT systems. Cloud resources can easily scale up or down based on the demand of IoT applications, ensuring optimal performance and resource utilization [75]. Additionally, cloud computing offers high computational power that is essential for processing the vast amounts of data generated by IoT devices, enabling efficient data analysis and insights [73], [76]. Moreover, cloud computing enhances storage capacity for IoT systems, allowing for the seamless storage of large volumes of data generated by IoT devices [74]. Real-time analytics is another advantage, as cloud platforms can process data quickly and provide instant insights, enabling timely decision-making in IoT applications [77]. The cost-effectiveness of cloud computing benefits IoT deployments by reducing infrastructure costs and operational expenses [72].

In our domain of State-of-Health (SoH) estimation, cloud resources offer us an opportunity to tackle the first of the three problems we are addressing: high computation needs. Cloud computing gives us remote resources to offload the training and testing of data-driven methods. In the next section, we discuss a general architecture for cloud-BMS, then review the work that has already been done in both literature and industry in offloading Battery Management System (BMS) functionalities to the cloud.

3.3 Cloud-based Battery Management Systems

3.3.1 Cloud BMS in Literature

By utilizing the vast computational resources and storage capabilities of the cloud, data from multiple BMS can be aggregated and analyzed in real-time. More demanding

estimation techniques can also be offloaded from onboard the car to a cloud server. This allows for advanced algorithms to predict battery life, optimize charging cycles, and prevent failures through predictive maintenance. Moreover, cloud-based platforms can facilitate remote firmware updates, ensuring that BMS are always equipped with the latest software enhancements. The scalability of cloud services also means that as the number of batteries increases, the system can easily expand to meet the growing demand without the need for significant hardware investments. Thus, cloud computing not only overcomes the inherent limitations of standalone BMSs but also adds a layer of intelligence that can lead to more sustainable and efficient battery usage.

Fig. 3.2 showcases a general architecture for cloud-BMS. Each vehicle has on-board sensors for current, voltage, and temperature. On the cloud side, each of the vehicles has a corresponding instance of the BMS with its functionalities operating over its data. The readings from each vehicle join readings collected from multiple other vehicles, over the lifetime of the system. The data from multiple vehicles can be analyzed and managed together to reach what is called fleet management. The previously discussed features of a BMS are now offloaded on the cloud side. This offloading allows access and control of this data through remote machines. This architecture acts as the blueprint for our proposed framework later on.

Regarding proposed frameworks in literature for cloud-based BMS, Ref. [78] presents the CHAIN framework, a cyber hierarchy and interactional network designed to optimize battery performance across its full lifespan. As shown in Fig. 3.3, it introduces a multi-scale, multi-condition control system that uses cloud-based data management and artificial intelligence to enhance the security and stability of battery systems for Electric Vehicles (EVs). Through its multi-scale design, CHAIN enables the integration of data from different scales, from molecular to system level. The incorporation of this multi-scale modelling enhances battery characterization by modelling electrochemical kinetics.

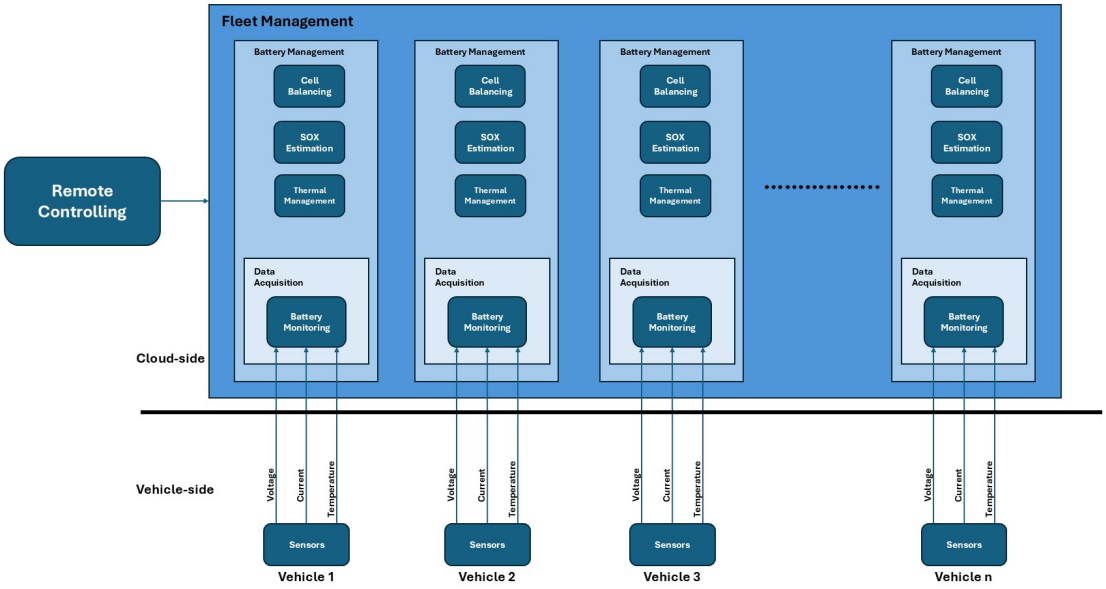

Figure 3.2: General Cloud-BMS architecture.

Multi-condition control refers to managing the battery under various conditions, not just while the vehicle is driving. These conditions include charging, driving, and parking. The CHAIN framework faces five key challenges: Multi-physical Modelling, which requires detailed models to understand electrochemical reactions; Perception and Data Space, which needs high-precision sensors and tools for nondestructive evaluation; Network and Communication, demanding efficient networks with low latency; Hash Rate and Computing, which calls for high computing power for big data and model processing; and Safety and Security, to protect against cyber threats in network-connected vehicles. In our proposed framework, we address some of the problems faced by CHAIN: no reliance on electrochemical reaction understanding, high network overhead, and security.

Capitalizing on the work done in [78], Ref. [79] presents a cloud-based BMS utilizing the CHAIN framework. It introduces a layered "cloud to things" architecture comprising end sensing, edge computing, cloud computing, and a knowledge repository.

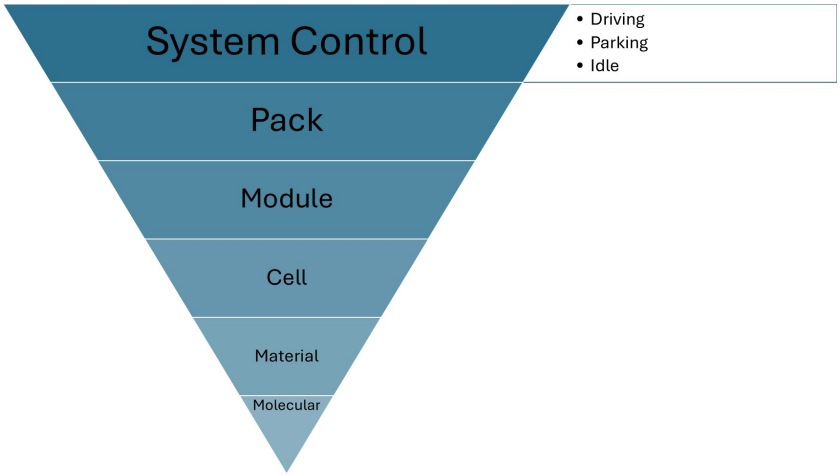

Figure 3.3: CHAIN framework involves data at multi-scales and multiple vehicle conditions.

This structure enables complex detection, prediction, and optimization functions for battery management across multiple scales, from individual cells to entire transportation systems. The CHAIN framework is used to provide multi-scale data visualization and hierarchical functional display, enhancing the BMS's capabilities in state estimation, thermal management, cell balancing, and fault diagnosis. The paper contributes to the CHAIN framework by proposing a multi-scale integrated modelling strategy for batteries and remote upgrading capability of the controller, aiming to improve the precision and adaptability of battery management systems. Remote upgrading is one of the main concepts utilized in our proposed framework to allow all vehicles in the fleet to learn from each other's data.

Staying within the same territory of work under the CHAIN framework, a method for estimating the SoH is presented in Ref. [80]. This method utilizes an end-cloud collaboration approach, which merges a cloud-based deep learning model and an end-based empirical model, thereby achieving high accuracy and real-time performance in SoH estimation. The cloud-side model, which is based on the Transformer architecture, performs feature extraction and estimation, having Root Mean Square Error (RMSE)

and Mean Absolute Error (MAE) of approximately 0.8%. The collaboration between the cloud and the end is facilitated through the application of the Kalman Filter and Unscented Kalman Filter (UKF), which are used to integrate and iteratively update the models. This hybrid approach of a simple model running on the vehicle, supported by a more complex model running on the cloud influenced the design of our proposed framework. Some computationally-intensive processing is pushed to the cloud, while other simpler processing is left on the vehicle. We also propose a hybrid approach that allows shifting between two models according to environmental conditions.

A similar approach under CHAIN was introduced for State-of-Charge (SoC) estimation in Ref. [81]. The proposed method involves end-cloud collaboration, combining a high-accuracy deep learning model on the cloud side based on Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks with a fast estimation model on the end side, and utilizing an Extended Kalman Filter (EKF) to fuse results from both ends. The method achieves an RMSE and MAE of approximately 1.5% and 1%, respectively. However, the paper identifies gaps in the generalizability of SoC estimation under different operating conditions, particularly for different aging states of batteries. Again, the hybrid approach shows up. In our case, our hybrid proposal addresses the issue which this paper faces, inconsistent performance with different operating conditions.

Implementations and proof-of-concepts for different cloud-based BMS have been introduced in literature, with some of them proposing solutions to different problems such as data complexity, network bandwidth consumption, visualization, monitoring, and operational cost efficiency.

Ref. [82] introduces an algorithm for joint estimation of SoC using High Integrity Filter-Particle Filter (HIF-PF), validated under Beijing Bus Dynamic Stress Test (BBDST)

conditions. The system demonstrates monitoring capabilities for battery voltage, temperature, and current, alongside real-time SoC estimation utilizing tools like Texas Instruments' BQ76PL455EVM board, ACS712-ELC-30A Hall current sensor, and software such as 3dsMax and Unity3D. The proposed system features communication in a single direction, physical system to the digital twin system. Communication in the opposite direction can offer updating of the model parameters on the physical side. Bidirectional communication allows for remote updating of model parameters, an essential part of the continuously improving nature of our framework.

Ref. [83] introduces algorithms for SoC and SoH estimation using Adaptive Extended H-infinity Filter (AEHF) and Particle Swarm Optimization (PSO), respectively. The system was validated with prototypes and tested with lithium-ion and lead-acid batteries, demonstrating monitoring capabilities and diagnostics. This approach uses model-based techniques of estimation, which we have already discussed its drawbacks in Subsection 2.4.2.

Ref. [84] proposes a four-layer networked structure that integrates cloud computing and edge computing technologies to enhance BMS performance. The architecture includes an Edge Computing Layer for real-time data processing, a Data Access Layer for secure data transmission, a Data Storage and Analysis Layer for storing and analyzing battery operation data, and a Data-Based Application Layer for lifecycle management applications. This structure aims to improve decision-making and optimization in battery management by facilitating service integration across infrastructure, data, platform, and software domains. An equivalent circuit model is used for the battery's state estimation, which the paper states offers much faster calibration compared to more complicated models. This multilayer structure somewhat resembles our approach of breaking down processing and splitting it on different locations.

Ref. [85] introduces a three-round feature selection (TRFS) approach to reduce

data complexity and measurement noise impact, employing a Random Forest Regressor (RFR) for accurate SoH estimation. The concept of data complexity reduction is used in our framework to reduce network communication overhead, using Federated Learning (FL).

Ref. [86] presents a digital twin architecture for real-time monitoring of EV batteries, focusing on SoC and SoH estimation. The authors propose a model that combines historical data and periodic retraining to reflect battery aging. The proposed architecture deploys the SoC estimation model onboard the vehicle since it needs to be real-time, while it offloads the SoH estimation to the cloud. It utilizes machine learning techniques like Random Forest, Light Gradient Boosting, and a Deep Neural Network. Periodic retraining, again, is a highlight of our proposed framework.

Ref. [87] proposes a cloud-based method for estimating the SoH of lithium-ion batteries using sparse charging data. The authors propose a Health Indicator (HI) feature derived from sparse data and validate its correlation with battery health through experiments with NASA's 18650 lithium-ion batteries. Results show that the method achieves low test errors under 10s sparsity, but accuracy decreases with sparser data. There are limitations in this approach when data sparsity exceeds 30s and suggested future work is to optimize constant voltage (CV) interval selection and validate the method with real-world cloud data from more vehicles. We utilize sparse charging data in our SoH estimation models on data collected from real-world vehicles.

Ref. [88] presents a Decentralized Intelligent BMS designed for smart battery management using cloud computing, which improves the precision of key parameters like SoC and SoH, enhancing the safety and lifespan of battery-based energy storage systems. The contributions include a distributed BMS design for increased security and a Petri Nets-based modelling procedure for clear operational conditions. Future work involves refining measurements, particularly SoH, for accurate predictive diagnostics and

battery remaining useful life (RUL) calculation. This distributed approach to security is one of the influences for our approach of using FL.

Ref. [89] presents the Kubernetes-Orchestrated IoT Online Battery Monitoring Platform (KBMP), which integrates Kubernetes and cloud-edge technology to enhance battery management. The platform ensures low-latency data transmission and analysis, utilizing a K-Means clustering algorithm for accurate thermal runaway (TR) warnings. Experimental results demonstrate that KBMP can provide TR warnings 30 minutes in advance, reduce data transmission latency by up to 20%, and decrease replica scaling latency by 50% compared to non-Kubernetes-integrated platforms. Kubernetes can be used in the implementation of our proposed framework to orchestrate its scaling, upgrading, and management.

By analyzing the work done in the papers mentioned above, the main features of a cloud-based BMS have been identified as follows:

1. Monitoring & Diagnostics: the BMS stores collected data on the cloud, allowing access and further analytics to be performed on it.

2. Online Learning: the BMS models can learn from collected data.

3. Visualization: the BMS platform allows data visualization.

Some BMS offer unique features. A summary of each cloud-based BMS mentioned with the features it provides is featured in Table 3.1. Rows without any asterisk in the "Features" column are cloud-end collaborative algorithms for estimating, rather than a cloud-based BMS platform.

Table 3.1: Summary of work done by cloud BMS papers mentioned.

Paper	States estimated			Estimation algorithms used	Features			
	SoC	SoH	SoP		Diagnostics	Online Learning	Visualization	Other
[80]		✓		Transformer + UKF				Feature extraction
[81]	✓			CNN-LSTM + EKF				
[82]	✓			HIF-PF	✓	✓	✓	Bi-directional communication
[83]	✓	✓		AEHF, PSO	✓			
[84]	✓	✓	✓	PF	✓	✓	✓	
[85]		✓		RFR				Noise reduction with TRFS
[86]	✓	✓		Random Forest, Light Gradient Boosting, Deep Neural Network				
[87]		✓		Linear Regression, Support Vector Machine, Gaussian Process Regression				Health indicator extraction, Sparse data (bandwidth efficiency)
[88]	✓			Petri Nets (for defining operational conditions)	✓			Decentralized synchronized management
[89]				K-Means	✓		✓	TR detection, Scalable

3.3.2 Industry Solutions

This subsection has been adapted from our published paper as-is. The goal of the paper was to review the work that has been done in cloud-BMS in both literature and industry, so we have kept this part here for the sake of completeness.

In addition to the work introduced in the literature, multiple cloud-BMS solutions are offered in the industry market. Table 3.2 lists the features of different cloud BMS solutions available. Bosch's cloud BMS's aging prediction utilizes usage data from connected vehicles to forecast battery conditions up to eight years ahead, combining physical models with AI for high accuracy. Lifetime optimization is achieved through standard, health, and fast charging modes, calculated to extend battery life or expedite charging times. The usage certificate provides a secure, transparent record of battery information, enabling accurate valuation based on certified KPIs, thus facilitating market transactions. Additionally, the 'Usage certificate to go' service assesses the health of batteries in existing vehicles through cloud-uploaded data, ensuring a comprehensive evaluation of battery health [90].

The Elysia Cloud Platform provides battery health and life insights. Its life forecasting feature captures non-linear degradation and predicts various states of health. Elysia also monitors fleet-wide degradation trends, pinpointing batteries that may not meet life targets and offering tailored recommendations for each vehicle. The defect detection capability ensures early identification of potential issues at the cell and pack level. For OEMs, the platform identifies trends in battery production quality. Additionally, its simulation toolkit allows for the assessment of usage impact on battery life and performance, which is crucial for product development and cost management. The platform's embedded BMS algorithms provide health-adaptive BMS parameters by supporting updates over-the-air, optimizing performance while addressing warranty concerns. The platform can also integrate with Simulink models [91].

The TWAICE Battery Analytics Platform features hybrid modelling, combining physical modelling with big data analytics for lifecycle accuracy and resilience. It ensures data security and compliance with industry standards like TISAX and offers real-time data integration services compatible with various systems such as BMS, EMS, and SCADA. Additionally, the platform provides instant battery development tools, SoH monitoring, and high-performance data processing engines for accurate battery insights. This solution is designed to enhance the transparency, efficiency, and reliability of battery systems, ultimately contributing to the sustainability of future energy and mobility markets [92].

Zitara offers a cloud and embedded battery management software called Zitara Live. It works with pack hardware across battery chemistries, including Lithium Iron Phosphate (LFP), and delivers customized, validated SoC, SoH, and safety outputs. Zitara Live continuously monitors the state of every battery in the fleet, updating critical performance parameters as they change over time. It also uses onboard simulations to predict energy, power, and heat generation [93].

Eatron Technologies has developed a BMS known as BMSTAR. This system is built on a platform that is independent of hardware and is based on physical models. It functions efficiently at the edge and is paired with a cloud counterpart for continuous, adaptive enhancements through over-the-air (OTA) updates. The BMS solution from Eatron uses AI algorithms and cloud technology for estimation of the RUL of the battery. Additionally, it includes safety features powered by AI for detecting cell anomalies and early signs of thermal runaway events [94].

Newten has designed a BMS that creates digital replicas of physical systems, integrated into real-time firmware. They utilize cloud-based solutions for accurate estimations of the battery's RUL, thereby reducing battery degradation. Newten also underscores the importance of cloud BMS in enhancing battery lifetime, charging, and

safety [95].

Bacancy's Cloud BMS Solution is designed for real-time visualization and monitoring of large-scale battery systems in EVs. The solution, named IONDASH, integrates IoT and cloud computing technologies. It features remote data logging, real-time parameter tracking, and GPS navigation for precise location tracking. IONDASH provides an interactive user interface for monitoring multiple BMS devices, allowing users to view critical battery parameters such as SoC, SoH, cell temperature, and voltage levels. Additionally, the platform includes in-built fuse circuit protection to safeguard against voltage and current spikes. Bacancy's product range caters to various EV market requirements, offering BMS solutions for 16-cell, 22-cell, and high voltage systems, emphasizing customization as per specific cell requirements [96].

The EVE-Ai Fleet Analytics platform, developed by Electra, offers a BMS solution designed for EV fleets. It provides real-time battery data analysis and insights for any fleet size. The system generates SoH trends and predictive models, identifies potential battery faults and failures, and enhances overall fleet efficiency and performance. Additionally, it offers insights into charging patterns and driver behaviour, enabling cost reduction and improved return on assets. The EVE-Ai 360 Adaptive Controls further optimize fleet management by resolving issues identified by the analytics, facilitated by cloud connectivity and over-the-air updates [97].

3.4 Analysis and Future Work Recommendations

While Section 3.3 listed significant efforts being made to leverage the power of cloud computing to address the drawbacks of BMS, there are still some areas that need further research and investigation:

1. Online learning: Online machine learning refers to a family of machine learning

Table 3.2: Detailed Comparison of Industrial Cloud BMS Solutions

Solution	Health Monitoring	Life Forecasting	Real-Time Data	Safety Features	Unique Features
Bosch's Cloud BMS [90]	✓	✓	✓		'Usage certificate to go' service
Elysia Cloud Platform [91]	✓	✓			Simulation toolkit, integration with Simulink models
TWAICE Battery Analytics Platform [92]	✓		✓		Hybrid modelling, data security and compliance
Zitara Live [93]	✓	✓			Works with various battery chemistries
BMSTAR by Eatron Technologies [94]	✓	✓		✓	Hardware-independent platform
Newten's BMS [95]	✓	✓			Digital replicas of physical systems
IONDASH by Bacancy [96]	✓		✓	✓	GPS navigation
EVE-Ai Fleet Analytics by Electra [97]	✓	✓	✓		EVE-Ai 360 Adaptive Controls

methods where a learner attempts to tackle predictive or decision-making tasks by learning from a sequence of data instances one by one at each time [98]. This approach has received attention for a wide range of applications, such as vehicle-to-grid services, fault diagnosis of EVs, and sentiment analysis in social media [99]–[101]. EVs generate a huge amount of data every day. Harnessing the power of connectivity and cloud computing allows for the collection of this data and using it to continuously train and improve existing models for functionalities like battery state estimation. While a few papers of those mentioned in Section 3.3 address this feature, there is still a need for proof-of-concept using real vehicle data. This investigation could also encompass a comparison between using cloud provider tools and open-source alternatives such as River (online machine learning library for Python) [102] and Beaver (machine learning operations (MLOps) framework for online machine learning) [103] by The Fellowship of Online Machine Learning.

2. Internet connectivity: Offloading certain BMS functionalities to the cloud will add the requirement that the vehicle is continuously connected to the internet. This is unfortunately not the case since many remote areas do not have internet access. A possible solution to this problem that needs to be investigated would be to have simpler and less accurate models onboard the vehicle that can run and provide the required functionalities until the vehicle can connect again to the cloud and get more accurate results using the more complicated models.

3. Large-scale fleet management: Fleet management involves using big data collected from many vehicles for objectives such as monitoring the fleet's performance and making maintenance decisions. In Section 3.3, Ref. [104] used this kind of approach for managing maintenance costs. Fleet management could be taken further in use cases such as planning vehicles' availability for ride-hailing applications based on real-time demand, energy and fuel efficiency, waste management, controlling and

diverting traffic in cities to less crowded routes, and predicting drivers' behaviours for safety.

4. Security: EVs connected to the cloud are susceptible to security threats at different levels. It could range from as simple as snooping on the data transmitted between the vehicle and the cloud, to more dangerous threats such as sending commands to the vehicle that can manipulate its control or sending false data to the cloud that can affect the algorithms that are making decisions. Data transfer can be secured by leveraging blockchain technologies. Their properties such as traceability, decentralization, and encryption allow for safe transmission of data. A notable concept for security on the edge is Data Confidence Fabrics (DCF). DCFs are virtual overlays that enhance data security and trustworthiness. They combine various trust technologies, such as hardware-based integrity checks and blockchain, to measure data confidence. DCFs generate scores, normalize across systems, and enable risk-based decision-making, bridging security, privacy, and trust in interconnected ecosystems [105].

5. Recent cloud computing advancements: Research can be directed to investigating how recent developments in the cloud computing field can be leveraged for EVs. Edge and Fog computing can be used for distributing computation and decreasing latency by moving computation closer to vehicles. Ref. [106] proposes a framework for provisioning resources according to vehicles' locations to reduce latency and cost. Serverless Computing can be used in cases like issuing safety alerts to drivers and sending control commands to vehicles on certain triggers. Multicloud computing can be used for redundancy and a better geographic distribution of computation. Research could be done to study the practical feasibility of using these technologies.

3.5 Conclusion

Cloud computing has matured from a convenient overflow resource to an essential part of modern EV operation. By offloading intensive computation, long-horizon analytics, and rich visualization, the cloud lifts many of the constraints that still hold back on-board BMSs. At the same time, it introduces fresh challenges: most notably intermittent connectivity, privacy preservation, and fleet-scale coordination, that cannot be solved by cloud technology alone.

The next chapter examines exactly those challenges through the lenses of *Self-Supervised Learning (SSL)* and *FL*. Chapter 4 shows how SSL makes use of the vast amounts of unlabelled cloud-aggregated data to improve state estimation without increasing annotation cost, directly addressing the "online learning" gap identified in Section 3.4. It then demonstrates how FL keeps sensitive vehicle data local while still benefiting from cross-fleet knowledge, in turn addressing the security and bandwidth considerations raised in this chapter, and touching upon fleet management.

Chapter 4

Self-Supervised Learning & Federated Learning

4.1 Introduction

Data-driven methods for State-of-Health (SoH) estimation are, as we discussed in Chapter 2 (Section 2.4.2), data-hungry. In practice, unfortunately, the vast majority of data generated by Electric Vehicles (EVs) is unlabelled. Laboratory-grade capacity tests that yield ground-truth SoH labels are expensive, time-consuming, and even infeasible once packs are in the field. As highlighted in the preceding Cloud Computing chapter (Section 3.3), this mismatch between abundant unlabelled data and scarce labels brings us back to two of the issues that we have already discussed:

1. **On-board computational limits:** Battery Management Systems (BMSs) cannot run heavyweight training loops. At the same time, they continuously stream collected sensor data (current, voltage, temperature, etc.). Making use of this data without overwhelming the hardware calls for representation-learning approaches

that can be pre-trained offline (or sparsely on-device) and then executed efficiently during inference.

2. **Bandwidth and connectivity variability:** While cloud resources allow large-scale processing, the intermittent connectivity and privacy considerations push us towards learning strategies that extract as much structure as possible locally before any communication. This argument is also explored further in the Federated Learning (FL) component we introduce in the next chapter.

Self-Supervised Learning (SSL) offers a solution to both challenges. By designing pre-text tasks that only rely on intrinsic structure within the raw signals (e.g., reconstructing masked segments, forecasting future windows, or contrasting augmented views), SSL can turn terabytes of unlabelled driving data into compact, information-rich representations. These representations then act as strong initializations for downstream SoH regressors that require far fewer labelled examples to fine-tune.

The empirical studies later in this thesis build directly on three SSL-related hypotheses:

H1. **Masking ratio matters:** In masked-autoencoding for time-series, the fraction of timesteps hidden during pre-training influences the granularity and quality of learned features. Chapter 6 sweeps multiple ratios to quantify their effect on SoH estimation accuracy.

H2. **Fine-tuning strategy matters:** Whether the SSL encoder is kept *frozen* (feature-extractor paradigm) or *unfrozen* (full/partial fine-tune) can trade off adaptation speed, and risk of overfitting. We evaluate both variants against a common dataset.

H3. **SSL complements FL:** Because SSL reduces label dependence, it lowers the communication load in federated settings where only model updates are shared. This

is explored further in the upcoming Federated Learning chapter.

On the other hand, EV fleets generate vast amounts of operational data that are invaluable for accurate SoH estimation. However, simply streaming this data to a central server is not as straightforward as it sounds. Privacy regulations (e.g., GDPR), commercial sensitivity around driving patterns, and the sheer communication cost of high-resolution time-series data impose strict limits on centralized training. FL offers a practical alternative: each vehicle trains a local model, shares only model parameters, and receives an aggregated global model that improves with every communication round. In this way, FL preserves data privacy while still exploiting the diversity of an entire fleet.

Section 4.2 starts by introducing SSL and the foundations it is based upon. Section 4.3 then showcases the most prominent methods to apply SSL, and motivates our method of choice. A debate on different fine-tuning strategies is presented in Section 4.4 to lay the ground for our experiments addressing it, and finally, Section 4.5 goes over the current works that use SSL for SoH estimation, and pinpoints the gaps that we cover in our work. The chapter then positions FL as the final building block in the thesis's learning framework. An overview of FL and its main objectives and mechanisms is introduced in Section 4.6. The main different methods of client aggregation are showcased in Section 4.7. Finally, Section 4.8 presents how FL has been implemented in SoH estimation so far, and the caveats that our work addresses.

4.2 Self-Supervised Learning Overview

SSL is characterized by its ability to make use of vast amounts of unlabelled data to derive useful representations, given that high-quality labelled datasets are often scarce. This lack of labelled data is not unique to the EV domain, but is experienced in other critical areas like medical imaging and telecommunications [107], [108].

SSL is built on some main foundations. At the core of SSL lies the concept of *implicit supervision*, where the learning process does not rely on human-annotated labels. Instead, SSL algorithms utilize the inherent structures and patterns present within the unlabelled data itself as a source of supervisory signals [109], [110]. This approach relies on the idea that the relational properties of data can provide enough context for a model to learn effectively [111].

Tasks in SSL are categorized into *pretext* and *downstream* tasks. Pretext tasks are designed to generate supervised signals from unlabelled data. In these tasks, the model learns to solve a specific problem, which indirectly helps it learn relevant features of the data for later use in downstream tasks [112], [113]. Since our data does not have ground-truth labels, the pretext task relies on labels that can be derived from the available data. For example, a model can be trained to determine the rotation angle of images fed to it. This activity forces the model to learn about the fundamental components of the image (e.g., edges, shapes), making this information transferable to different tasks such as classification or detection. See, although the images are not labelled, we can perform angle rotations on them and use those angles as ground-truth for the pretext task. This process is called *pre-training* and is how we refer to it in our work in the following chapters.

These learned representations can then be "transferred" to our target task. We use the limited labelled data that is available for our target task and adjust the learned representations from the pretext task. This process is called *fine-tuning*, and the target task is referred to as the downstream task. The process of using the pre-trained model in the pretext task for another downstream task is called *transfer learning*.

4.3 Self-Supervised Learning Methods

4.3.1 Autoassociative Self-Supervised Learning

Autoassociative SSL methods aim to predict parts of the data they are not made aware of using parts they are. This can be in the form of: predicting the future given the past (autoregression), predicting original data given a compressed form of it (autoencoding), predicting masked parts of the data given other visible parts of it (masked modelling), etc. Fig 4.1 shows a comparison between the different methods presented.

Autoregression

Autoregressive SSL predicts each element of a sequence from its predecessors. In practice, models such as GPT employ next-token prediction to learn representations of text that encode rich syntactic and semantic structures of language [114]. Because prediction proceeds sequentially, autoregressive methods are naturally applied to data with inherent order: time series, speech, or text. They excel when one cares about generative sampling or when causality of the sequence must be respected (e.g., speech synthesis or language modelling). Their left-to-right learning also enables efficient online inference but cannot capture bidirectional context without modification.

Autoencoding

Autoencoders compress an entire input into a lower-dimensional latent code and then reconstruct the full input from this code. This end-to-end reconstruction encourages the encoder to capture salient global patterns. Variants include denoising autoencoders, which corrupt inputs and learn to recover the original. Autoencoding SSL is straightforward and model-agnostic, which makes it applicable to a variety of domains from images, to sensor signals, even to genomics. However, pure autoencoding may learn trivial identity mappings unless regularized or paired with sparsity or information-bottleneck constraints. That is where masked modelling steps in.

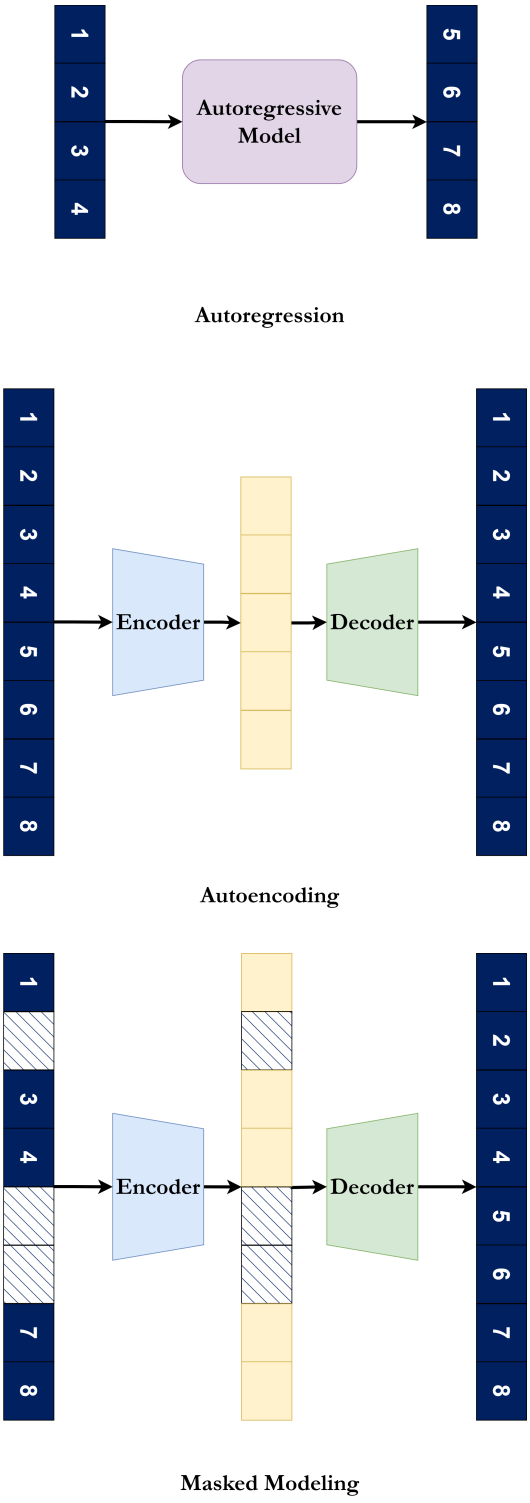

Figure 4.1: A visual comparison between different Autoassociative Self-Supervised Learning methods.

Masked Modelling

Masked modelling randomly hides portions of the input and tasks the model with predicting or reconstructing only the masked parts. In BERT-style masked language modelling, random tokens are masked, and the model predicts them from the surrounding context [115]. In vision, masked image modelling (MIM) hides patches and reconstructs pixel values or latent features [116]. By conditioning on both left and right contexts, masked modelling enables bidirectional representation learning, capturing global dependencies more effectively than autoregression. It is best used when context aggregation (rather than causal generation) is the goal, such as regression. Masked models often leverage transformer architectures and require careful masking strategies (mask ratio, block versus uniform masking) to balance the learning signal and model capacity.

4.3.2 Contrastive Learning

Contrastive learning is a self-supervised paradigm that trains an encoder to differentiate between similar ("positive") and dissimilar ("negative") data pairs without external labels. At its core, it maximizes the agreement of representations for two views of the same instance while dispersing others in the embedding space.

Major frameworks for contrastive learning include SimCLR and MoCo. SimCLR generates two random augmentations per sample (cropping, colour jitter, etc.) and relies on large batch sizes to provide sufficient negatives within each mini-batch [117]. To overcome the need for large batches, MoCo employs a dynamic memory bank ("dictionary") of past embeddings and a momentum-updated key encoder [118].

Due to their discriminative nature, contrastive learning methods are suitable for tasks such as classification when labelled data is scarce. However, they suffer from high computational requirements, an issue BMSs suffer from.

Since SoH estimation is a regression problem and we are looking for lightweight processing, autoencoding was our selection in this work. Due to the aforementioned problem of learning trivial representations, it was coupled with masked modelling. The methodology and masking strategy are further elaborated in Chapter 5 (Section 5.4.1).

4.4 Fine-Tuning Strategies

When it comes to fine-tuning, there is a debate on whether the pre-trained parts of the model should be allowed to train during fine-tuning or not. "Full" fine-tuning (FFT) updates every parameter to maximize the capacity to learn task-specific patterns, but often at the cost of overwriting the general representations acquired during pre-training. This overwriting manifests as *catastrophic forgetting*: the model's performance on the original pre-training distributions or earlier tasks degrades sharply when all parameters are updated on new, possibly limited data [119].

In contrast, the feature-extractor (frozen) paradigm keeps all parameters of the pre-trained model unchanged throughout the fine-tuning process, while partial fine-tuning methods freeze parts of the pre-trained network to preserve general knowledge and reduce computation. A simple variant is linear probing or "head" fine-tuning, in which only the final classification (or regression) layer is trained while the entire feature extractor remains fixed. This strategy is said to be effective when the downstream domain closely matches pre-training, since low-capacity heads suffice to specialize general embeddings without disrupting them.

There is some middle ground between these two approaches. Adapter-based tuning inserts lightweight modules (adapters) into each layer of the frozen backbone. Only these small bottleneck layers are trained, keeping the vast majority of pre-trained weights untouched. Other approaches, such as LoRA and BitFit, inject parameters into the frozen pre-trained model, aiming to optimize the fine-tuning process [120], [121].

In our work, we explore the effect of full fine-tuning and feature-extractor paradigm, and conduct a comparison between both approaches for SoH estimation.

4.5 Self-Supervised Learning in SoH Estimation

Early SSL work for SoH estimation centres on reconstruction objectives. Wang et al. use the same dataset as our work and use autoencoding for the SSL method [122]. However, their approach shares raw data over the network, compromising users' privacy, and it does not offer insights into how different masking ratios or fine-tuning approaches affect performance. Shen et al. pre-train a masked autoencoder on more than a million unlabelled charge curves, then fine-tune with only 20% of the labels [123]. Still, their approach keeps the masking ratio fixed at 75% and uses lab-grade datasets, which do not resemble sensor inaccuracies in real-life data.

Chen et al. add a "degradation" loss that forces their Transformer to learn a strictly downward health curve [124]. The monotonic rule breaks when capacity recovers after rest, something common in real fleets. Our autoencoder has no monotonic assumption, so it captures both fades and small rebounds we see in field data.

On the side of contrastive frameworks, Sadler et al. map real cycles to simulated ones with a large-batch InfoNCE loss [125]. This is computationally-intensive and is not suitable for on-board BMS.

One difference that highlights the suitability of our approach for real-world operation that was not explored across other studies, is allowing the model to build on its own past estimations. Other studies use the ground-truth past SoH for each estimation, rendering the estimation trivial. Our results simulate the model operation throughout the complete lifecycle of the vehicle.

Across studies the same gaps appear:

1. Masking ratio effects were not explored.

2. Fine-tuning techniques were not clear.

3. Real deployment costs were ignored.

4. Data privacy was not taken into consideration.

5. Models were not tested in a real-world-like scenario.

All these gaps are tackled in our proposed framework, presenting experimental results to support our choice of SSL hyperparameters in a real-world scenario, while protecting users' privacy, and taking into consideration the limited resources on EVs.

4.6 Federated Learning Overview

FL has emerged over the last decade as a distributed machine learning paradigm, motivated primarily by the need to exploit large-scale, decentralized data while preserving privacy and reducing communication overhead. Initially proposed by Google in 2016, FL was conceived to enable multiple clients, such as mobile devices or edge nodes, to collaboratively train a shared global model without exchanging raw data [4]. This approach directly addresses privacy regulations (e.g., GDPR), user concerns over data exposure, and the prohibitive costs of centralized data aggregation in settings ranging from smartphones to Internet of Things (IoT) deployments.

At its core, federated learning is a client-server architecture in which each round of training proceeds through three steps [126]:

1. The server broadcasts the current global model to a selected subset of clients.

2. Each client performs local optimization on its private dataset and returns only the updated model parameters or gradients.

3. The server aggregates these updates with an algorithm to form a new global model for the next round.

By iterating this process over multiple rounds, the system converges toward a model comparable to one trained centrally while never collecting sensitive data in a single repository.

FL enables collaborative model training across decentralized devices by exchanging model updates instead of raw data, but real-world deployments suffer when clients hold heterogeneous datasets. Statistical heterogeneity or "non-IID" distributions across clients leads to client-drift, degraded accuracy and slower convergence in vanilla FL [126]. Non-IID refers to the violation of the independent and identically distributed assumption, whereby local datasets differ in data quantity, label proportions or feature distributions due to user-specific behaviour or sampling bias.

FL mitigates non-IID effects through iterative aggregation of local updates (e.g., FedAvg), which effectively smooths distribution discrepancies and uses global data diversity. Some other methods are also employed which are discussed in the next section.

FL has already seen extensive real-world adoption, particularly within IoT ecosystems. In smart city applications, FL enables distributed analytics on sensor and mobility data to optimize traffic flows and public services without infringing on individual privacy [127]. In industrial IoT, FL has been combined with digital twin frameworks to perform adaptive predictive maintenance and anomaly detection across heterogeneous factory assets while safeguarding proprietary operational data [128]. Healthcare represents another key domain: FL allows edge devices such as wearable monitors and in-home health sensors to cooperatively learn predictive models for chronic disease monitoring [129]. Cybersecurity applications in IoT use FL for intrusion detection systems and malware classification, improving detection performance by aggregating threat intelligence

from multiple organizations without exposing local logs [130].

4.7 Federated Learning Methods

There are different methods for aggregation of model parameters from clients, each with its own pros and cons. We go over some of the most prominent ones in this section.

Federated Stochastic Gradient Descent (FedSGD): In FedSGD, each client computes a single gradient on its local data and sends it to the server, which aggregates a weighted sum of client gradients to update the global model [4]. FedSGD is appropriate when communication rounds are inexpensive and clients can compute only one gradient step per round (e.g., highly resource-constrained IoT devices)

$$w^{t+1} = w^t - \eta \sum_{i=1}^{K} \frac{n_i}{n} \nabla F_i(w^t) \tag{4.1}$$

where w^{t+1} represents the updated global model parameters, w^t denotes the current global model parameters, η is the learning rate, K is the number of participating clients, n_i is the number of data samples held by client i, n is the total number of data samples across all clients, and $\nabla F_i(w^t)$ is the gradient of the local objective function F_i at client i, evaluated at w^t.

Federated Averaging (FedAvg): FedAvg accelerates FedSGD by letting each client perform E local SGD epochs before transmitting its model [4]. A weighted average of the resulting models forms the new global iterate. FedAvg is the de facto baseline in heterogeneous data federations because it reduces communication (many local updates per round) while preserving convergence on both IID and non-IID data.

Local update:

$$w_i^{t+1} = w^t - \eta \sum_{e=1}^{E} \nabla F_i(w_i^{t,e}) \tag{4.2}$$

Server aggregation:

$$w^{t+1} = \sum_{i=1}^{K} \frac{n_i}{n} w_i^{t+1} \tag{4.3}$$

where w_i^{t+1} represents the locally updated model parameters on client i, w^t denotes the global model parameters before aggregation, η is the learning rate, E is the number of local epochs performed by each client, $\nabla F_i(w_i^{t,e})$ is the gradient of the local objective function F_i at client i during epoch e, K is the number of participating clients, n_i is the number of data samples held by client i, n is the total number of data samples across all clients, and w^{t+1} is the aggregated global model parameters after receiving updates from all clients.

Federated Proximal (FedProx): FedProx extends FedAvg by adding a proximal term to each client's objective, controlling how far local updates may drift from the current global model [131]. It is well-suited for settings with severe system or statistical heterogeneity, where clients' updates can diverge and slow or destabilize convergence.

Local update:

$$w_i^{t+1} = \arg\min_{w} \left(F_i(w) + \frac{\mu}{2} \|w - w^t\|^2 \right) \tag{4.4}$$

Server aggregation:

$$w^{t+1} = \sum_{i=1}^{K} \frac{n_i}{n} w_i^{t+1} \tag{4.5}$$

where w_i^{t+1} represents the locally updated model parameters on client i, w^t denotes the global model parameters before aggregation, $F_i(w)$ is the local objective function at client i, μ is the proximal term weight that controls how far local updates can drift from

the global model, K is the number of participating clients, n_i is the number of data samples held by client i, n is the total number of data samples across all clients, and w^{t+1} is the aggregated global model parameters after receiving updates from all clients.

SCAFFOLD (Stochastic Controlled Averaging): SCAFFOLD uses control variates to correct for client drift by tracking and compensating for the difference between each local gradient and the global gradient estimate [132]. It is particularly effective under non-IID data distributions, where FedAvg/FedProx suffer from bias due to unequal client gradients.

Local update:

$$w_i^{t+1} = w^t - \eta \nabla F_i(w^t) - c_i^t + c^t \tag{4.6}$$

Client control variate update:

$$c_i^{t+1} = c_i^t - c^t + \frac{1}{\eta}(w^t - w_i^{t+1}) \tag{4.7}$$

Server control variate update:

$$c^{t+1} = \sum_{i=1}^{K} \frac{n_i}{n} c_i^{t+1} \tag{4.8}$$

where w_i^{t+1} represents the locally updated model parameters incorporating control variates, w^t denotes the global model parameters before aggregation, η is the learning rate, c_i^t is the local control variate for client i, c^t is the global control variate, K is the number of participating clients, n_i is the number of data samples held by client i, n is the total number of data samples across all clients, and c^{t+1} is the aggregated global control variate.

FedDANE (Federated Newton-Type Method): FedDANE adapts the Distributed Approximate Newton (DANE) method to federated settings, using local second-order information to accelerate convergence [133]. FedDANE is particularly effective in convex optimization settings, but empirical studies suggest that it may struggle in realistic federated environments due to low device participation and statistical heterogeneity.

Local update:

$$w_i^{t+1} = \arg\min_w \left(F_i(w) - (\nabla F_i(w^t) - \nabla F(w^t))^T w + \frac{\mu}{2} \|w - w^t\|^2 \right) \tag{4.9}$$

Server aggregation:

$$w^{t+1} = \sum_{i=1}^{K} \frac{n_i}{n} w_i^{t+1} \tag{4.10}$$

where w_i^{t+1} represents the locally optimized model parameters on client i, w^t denotes the global model parameters before aggregation, $F_i(w)$ is the local objective function at client i, μ is the proximal term weight ensuring stability, $\nabla F_i(w^t)$ is the gradient of the local objective function, $\nabla F(w^t)$ is the global gradient estimate, K is the number of participating clients, n_i is the number of data samples held by client i, and n is the total number of data samples across all clients.

FedBN (Federated Batch Normalization): FedBN keeps Batch Normalization parameters local to each client while averaging only the remaining (shared) parameters, mitigating feature-distribution shifts across clients [134].

Shared parameter update:

$$w_{\text{shared}}^{t+1} = \sum_{i=1}^{K} \frac{n_i}{n} w_{i,\text{shared}}^{t+1} \tag{4.11}$$

Local BN parameter update:

$$\theta_{\mathrm{BN},i}^{t+1} \leftarrow \theta_{\mathrm{BN},i}^{t+1} \tag{4.12}$$

where w_{shared}^{t+1} represents the updated global shared model parameters, $w_{i,\text{shared}}^{t+1}$ denotes the locally updated shared model parameters on client i, K is the number of participating clients, n_i is the number of data samples held by client i, n is the total number of data samples across all clients, and $\theta_{\mathrm{BN},i}^{t+1}$ represents the batch normalization parameters that remain local to each client.

FedSplit: FedSplit casts federated optimization as a set of local sub-problems linked by a shared global variable, solved via operator-splitting techniques to ensure exact fixed-point convergence [135]. FedSplit is particularly useful when clients can tolerate more complex local computations in exchange for stronger convergence guarantees under heterogeneous data distributions.

Local subproblem:

$$w_i^{t+1} = \arg\min_w \left(F_i(w) + \frac{\rho}{2} \|w - \phi^t\|^2 \right) \tag{4.13}$$

Global variable update:

$$\phi^{t+1} = \phi^t + \gamma \sum_{i=1}^{K} \frac{n_i}{n} (w_i^{t+1} - \phi^t) \tag{4.14}$$

where w_i^{t+1} represents the locally updated model parameters on client i, $F_i(w)$ is the local objective function at client i, ρ is the penalty parameter controlling update stability, ϕ^t is the global shared variable, γ is the relaxation factor controlling aggregation rate, K is the number of participating clients, n_i is the number of data samples held by client i, and n is the total number of data samples across all clients.

In our work, we picked *FedAvg* as our choice of aggregation method, because it strikes the right balance of managing heterogeneous data, minimizing communication, and computation.

4.8 Federated Learning in SoH Estimation

Early attempts to apply FL to SoH estimation simply transplanted FedAvg onto small, homogeneous cell datasets. Chen et al. trained an LSTM on five clients and reported that federated training cut the Mean Absolute Error (MAE) from 2.7% (local) to 1.9%, virtually matching a centrally-trained baseline while keeping raw data on-board the vehicles [136]. However, their study assumed identical cycling profiles, so the method faltered when non-IID field data was injected, and communication cost was not analyzed.

To cope with heterogeneous fleets, several groups added personalization layers or clustering. Wang et al. introduced an adaptive multi-personalized FL framework that forms multiple global models and regularizes each client with an importance-weighted loss [137]. On a 30-vehicle dataset spanning three chemistries they cut MAE by 0.14% versus plain FedAvg and lowered operational risk five-fold, yet the approach still depended on fully-labelled capacity data and omitted unlabelled mileage logs that dominate real BMS storage.

A parallel line of work tackles the communication bottleneck. The FedCBE system compressed model updates for collaborative capacity and State-of-Charge (SoC) estimation, achieving <5 MB per round on eight buses, but its accuracy dropped by $\approx 3\%$ once packet loss exceeded 10% [138]. More recently, the FedACM aggregator introduced client-momentum to stabilize training under severe non-IID drift and halved bandwidth versus FedAvg, yet it still required labelled cycles [139].

Across these studies, we see that every method relies on labels that are costly to

obtain at scale, and most solutions still struggle when usage profiles, temperatures, and chemistries diverge widely. In our work, we overcome the problem of labelled data scarcity by making use of SSL through masked autoencoding, and we study the effect of different fine-tuning methods on the generalizability of our models.

4.9 Conclusion

SSL has emerged as a practical way to tap the vast stores of unlabelled battery data that every electric vehicle already logs. A survey of recent work shows three main routes: *(i)* masked-reconstruction models that learn by filling in missing segments, *(ii)* physics-guided objectives that embed ageing rules, and *(iii)* contrastive frameworks that separate "similar" and "dissimilar" cycle pairs. All three cut the label budget, yet they leave key questions open:

- **Hyper-parameter tuning.** Mask ratios are often fixed with little justification.
- **Fine-tuning strategy.** Many studies do not state whether SSL encoders stay frozen or are fully/partially updated.
- **Real-world deployment.** GPU-heavy training, large batches, or strict monotonic losses clash with on-board compute limits and non-ideal field data.
- **Data privacy.** Sharing of raw sensor data over the network puts users' habits and patterns at risk.

This thesis addresses those gaps by

1. sweeping through mask ratios (0-75 %) to find the most optimal setting,
2. comparing Frozen-SSL, Unfrozen-SSL, and Unfrozen-SL head-to-head,

3. embedding the best encoder in a bandwidth-aware federated loop that stays inside an ECU-sized compute budget and transmits $< 600\,\text{kB}$ per round, and

4. integrating a federated approach to model parameter aggregation.

This chapter has also introduced FL as the keystone that lets an EV fleet learn from its collective experience without surrendering raw, privacy-sensitive battery data. We went with a high-level overview of FL workflows and its real-life applications. We then surveyed the algorithmic landscape from baseline FedAvg to recent personalization and communication-efficient variants, showing how each one trades accuracy, convergence speed, and bandwidth.

Finally, the literature on FL for SoH estimation revealed two persistent gaps: label dependence, and lack of generalization to different working conditions. These shortcomings motivate the design choices in this thesis: combining self-supervised masked autoencoding with a bandwidth-aware and privacy-preserving FedAvg variant.

Together, these insights set the stage for the next chapter, where we translate the conceptual framework into a concrete methodology, detailing data preprocessing and the model architectures that give life to the self-supervised-plus-federated approach introduced here.

Chapter 5

Methodology

5.1 Introduction

The preceding chapters have established both the importance and the difficulty of accurately estimating the State-of-Health (SoH) of Electric Vehicle (EV) batteries. Chapters 3 and 4 reviewed cloud computing, Self-Supervised Learning (SSL), and Federated Learning (FL) as partial answers, each tackling a different problem, yet leaving gaps when considered in isolation.

This chapter converts those conceptual arguments into a concrete workflow. Section 5.2 outlines a pipeline that fuses SSL, FL, and elastic cloud resources so that every vehicle benefits from fleet-wide experience without exposing its raw data. Section 5.3 then introduces the dataset used to test the framework, explains how ground-truth SoH is inferred from charging logs, and justifies an Incremental Capacity (IC) analysis filtering step that mitigates sensor noise. The following sections describe the model architecture in two phases: a masked-autoencoder pre-training stage that learns generic battery representations from unlabelled data, and a fine-tuning stage that couples those representations with historical SoH through a BiLSTM regressor. The chapter closes with a

discussion of how these components interact in a continuous training-deployment cycle.

5.2 Proposed Combined Framework

In this work, the proposed framework combines the three discussed solutions to data-driven approaches in SoH estimation: cloud computing, SSL, and FL to take advantage of their combined advantages.

Each moment, a generous amount of data is generated and collected by sensors operating on each EV. Unfortunately, as already discussed in Chapter 2, the SoH of an EV's battery cannot be measured directly using sensor readings. This problem renders all supervised Artificial Intelligence (AI) techniques helpless before this data. Supervised AI methods require the presence of a label for training purposes. There is knowledge that is embedded within this unlabelled data, which supervised AI techniques have harshly deemed useless. As thoroughly reviewed in Chapter 4, SSL has proven successful in learning from unlabelled data, capturing data distribution, and underlying features. This acts as the first cornerstone for the proposed framework.

Subsequently, sharing this collected data raises privacy concerns among users regarding this approach. A security breach can cause this data to fall into the wrong hands, and the user's trips, habits, and sensitive information will be compromised. FL lends a helping hand to SSL to protect the privacy of the users. Chapter 4 showed the mechanism by which FL works, training local models on remote devices and transmitting the trained model parameters over the network instead of the collected data.

The proposed framework for EV SoH estimation, can be described through the following steps:

1. Each EV collects its own unlabelled data using the various sensors while moving.

2. On board each EV, a local SSL model is pre-trained using the collected unlabelled data. The SSL technique can be any of those discussed in Chapter 4 (Section 4.3).

3. Each EV, after pre-training its local model, shares the model parameters only with the central server over available networks. This is in contrast to the approach of sharing the original collected data without any pre-training happening on the vehicle side.

4. The central server collects the model parameters shared by each EV in the fleet. The server then aggregates all of these model parameters. Aggregation could be applied using any of the techniques discussed and reviewed in Chapter 4 (Section 4.7).

5. Once aggregation is done, the resulting model is then fine-tuned using the available labelled data on the server.

6. Finally, the fine-tuned model parameters are shared with all fleet vehicles. Each vehicle can now use the fine-tuned model for its own SoH estimation, harnessing data distributions collected from the whole fleet.

Fig. 5.1 shows the different steps of the proposed approach.

5.3 Dataset

The dataset used to verify the proposed approach was introduced in [140]. This dataset was created through the collection of charging data from 20 commercial EVs over the span of 29 months in Shanghai, China. Only charging data was published to protect the information and privacy of the EV users. The vehicles are BAIC EU500, which are equipped with CATL NMC battery cell chemistry. The nominal capacity of the batteries is 145Ah. Each pack has 90 cells connected in series and 32 temperature sensors inside.

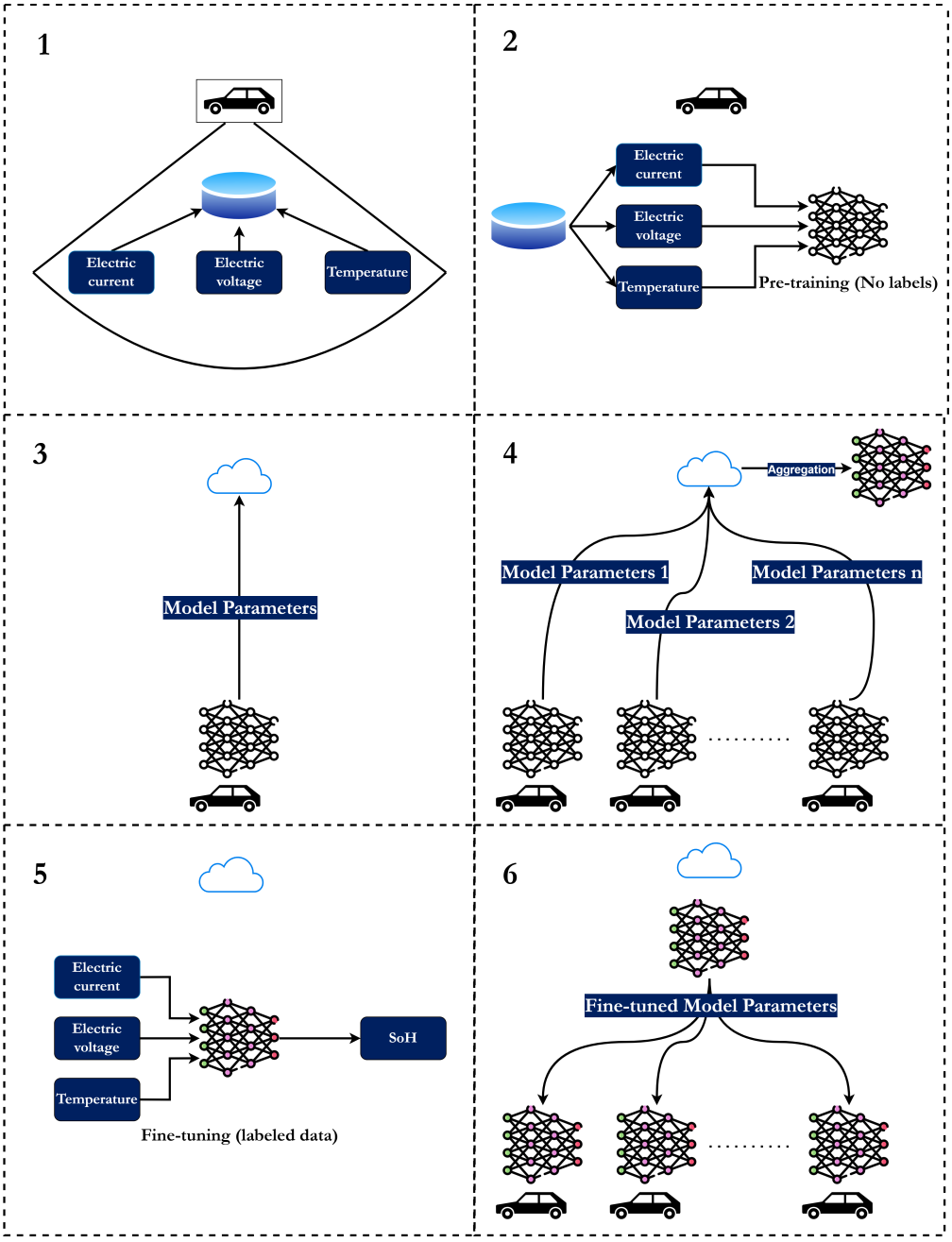

Figure 5.1: Steps of the proposed combined approach from left-to-right, and top-to-bottom

Table 5.1 lists the items provided by the dataset, the measuring unit, and the resolution of each element. The sampling rate of each of these items is 8 seconds. It is worth noting that the resolutions are low due to the fact that the data was collected via control area network communication, which is not as accurate as data collected in a laboratory environment [140]. Table 5.2 lists the parameters of the vehicles in the dataset [122].

Table 5.1: Items included in dataset with their measuring units and resolutions.

Items	**Unit**	**Resolution**
Time	yyyy-mm-dd hh:mm:ss	-
Current	A	-
Pack voltage	V	0.1 V
SOC	-	0.1
Maximum cell voltage	V	0.001
Minimum cell voltage	V	0.001
Maximum cell temperature	°C	1°C
Minimum cell temperature	°C	1°C
Available energy	kWh	-

Table 5.2: Main parameters of tested vehicles.

Vehicle parameter	**Specification**	**Unit**
Range	300	km
Wheelbase	2670	mm
Maximum power	160	kW
Maximum torque	300	N m
Battery pack energy	45.3	kWh
Cell chemistry	C/NMC	-

As can be noticed in Table 5.1, the dataset does not include the SoH (the groundtruth) as one of its items. This is because the data was collected from real vehicles with no periodic laboratory tests to determine the batteries health. To obtain the groundtruth that will be used for testing and fine-tuning our model, a variant of the Ampere hour counting formula is employed, as presented in [140], and demonstrated in Eq. 5.1,

$$C_a = \frac{-\int_{t_1}^{t_2} \Delta t I(t)}{SO\,C_{t_2} - SO\,C_{t_1}} \tag{5.1}$$

where Δt represents the sampling rate (8 seconds in our case), $t1$ and $t2$ are the start and end charging times, respectively, and I is the electric current (negative for charging).

After the SoH was obtained for each vehicle, the sensor inaccuracy problem arose. Fig. 5.2 shows a scatter plot for each of the 20 vehicles after obtaining the SoH against the time period driven. As can be seen, there is a large number of outliers throughout. The SoH of a vehicle should decrease with time, but the figure shows sudden jumps and drops in the battery capacities. This can be attributed to the onboard sensors having low accuracy, State-of-Charge (SoC) estimation errors, and noise in the data.

This problem was addressed in the article that presents the data set [140]. The solution is based on the fact that there is no need in tracking the vehicles' SoH for every charging session. The SoH can be aggregated for a specified period of time. A monthly aggregation of SoH values is applied, with the median and mean chosen as the aggregation methods. Fig. 5.3 shows a comparison between the monthly mean and median SoH values for each of the 20 vehicles. The gaps in the plots resemble months that did not have data (the respective vehicle was not charged during that month). The mean still shows sudden peaks and drops, stemming from its statistical nature of being sensitive to outliers. The median, on the other hand, shows less sensitivity and smoother degradation of SoH. Hence, the monthly median is used as the groundtruth for our experiments.

Moving on to feature engineering, IC analysis is a diagnostic technique that involves examining the derivative of capacity with respect to voltage dQ/dV during the charge or discharge of a battery. By plotting these IC curves, intricate features such as peaks and valleys become apparent that are correlated with various electrochemical processes within the battery. These features can change significantly as the battery ages and degrades, providing information on shifts in electrode kinetics, phase transitions, and other degradation mechanisms [141], [142]. IC analysis is especially valuable because

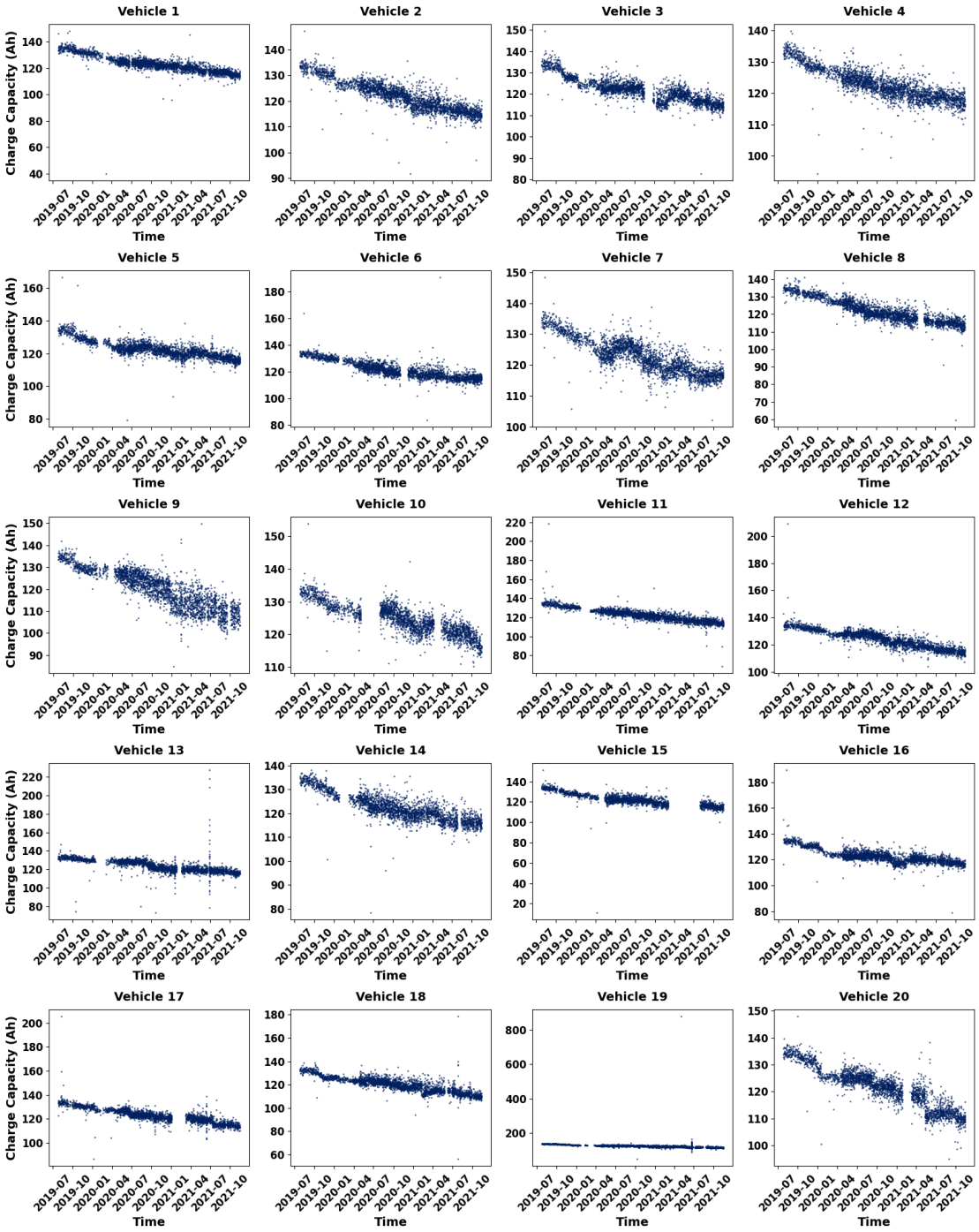

FIGURE 5.2: Scatter plot for the SoH of each of the 20 vehicles in the dataset over time

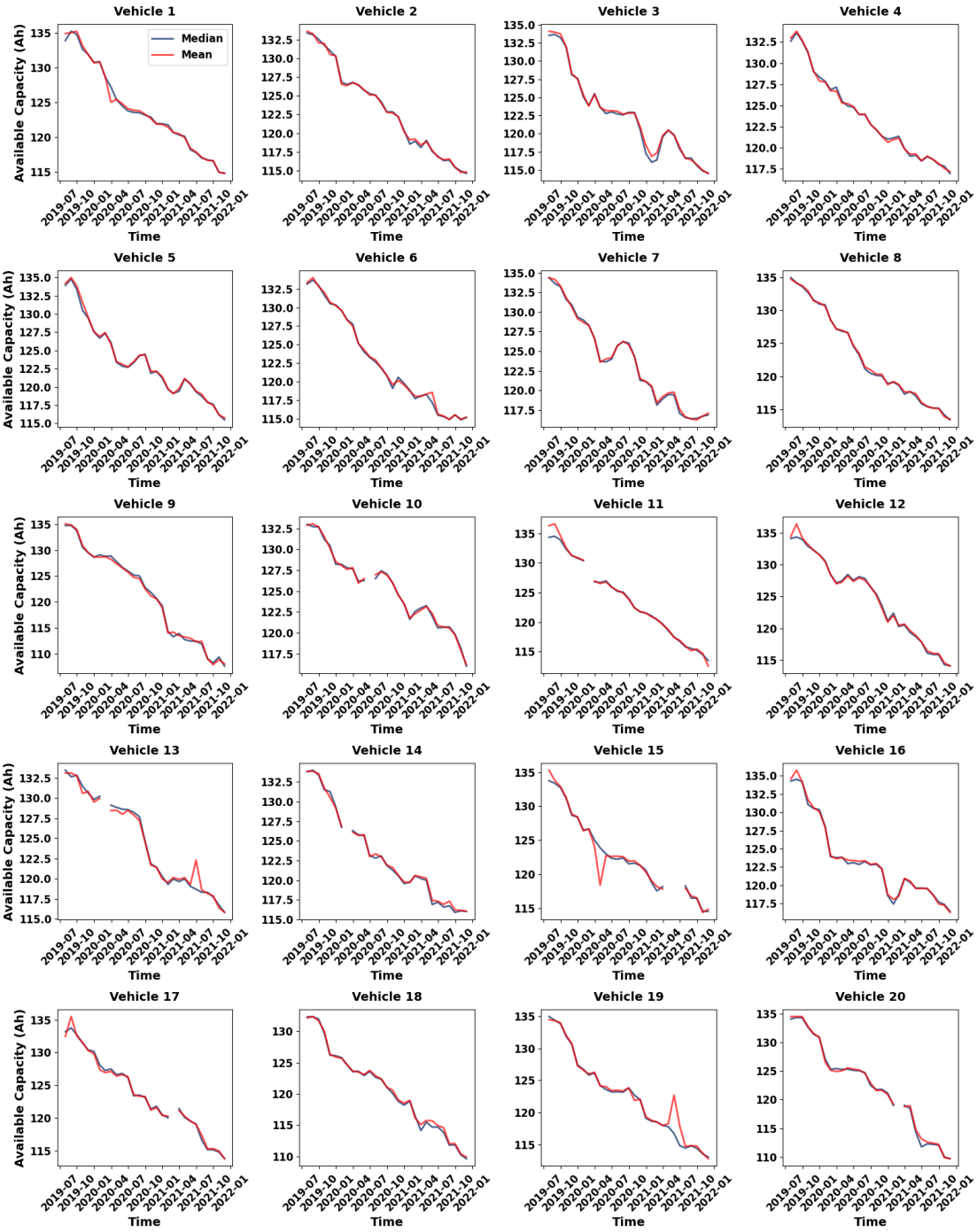

FIGURE 5.3: Line plot for the monthly mean and median SoH of each of the 20 vehicles in the dataset over time

it serves as an indirect measurement of the internal state of the battery. As batteries degrade, the IC curves exhibit characteristic shifts and alterations in peak intensities. These changes are directly associated with a reduction in available capacity and increased internal impedance [143].

Furthermore, the sensitivity of the incremental capacity method to subtle changes in battery behaviour makes it particularly useful for conditions where traditional voltage or current monitoring might not fully capture declining performance. This is essential in anticipation of critical events, such as capacity loss or the risk of abrupt failures. As a result, IC analysis is regularly integrated into both research and practical Battery Management System (BMS) applications to continuously track battery degradation [143].

Research has shown that achieving an accurate SoH estimation is possible through extracting features from short charging sequences in the voltage range where the main IC peak resides throughout battery life [144]. Short charging sequences are used to reduce computational complexity, allowing this kind of model to run on vehicles where computation is limited. These short sequences were found to be sufficient for SoH estimation, as they are in the information-dense range determined through IC analysis [144]. Since this dataset is collected from a battery pack (multiple cells), the IC analysis method was performed on the maximum voltage cells. It was found that the main IC peak resides between 3.6V and 3.8V as shown in Fig. 5.4. To remain consistent with other literature using the same data set and IC analysis method, the length of the charging sequence used for feature extraction is 160 seconds [122]. Fig. 5.5 shows the final monthly median of each of the 20 vehicles calculated using data points in the voltage range 3.6 to 3.8V.

5.4 Model Architecture

The model used as an implementation of the combined approach proposed in Section 5.2 can be broken down into 2 parts: (a) pre-training, and (b) fine-tuning.

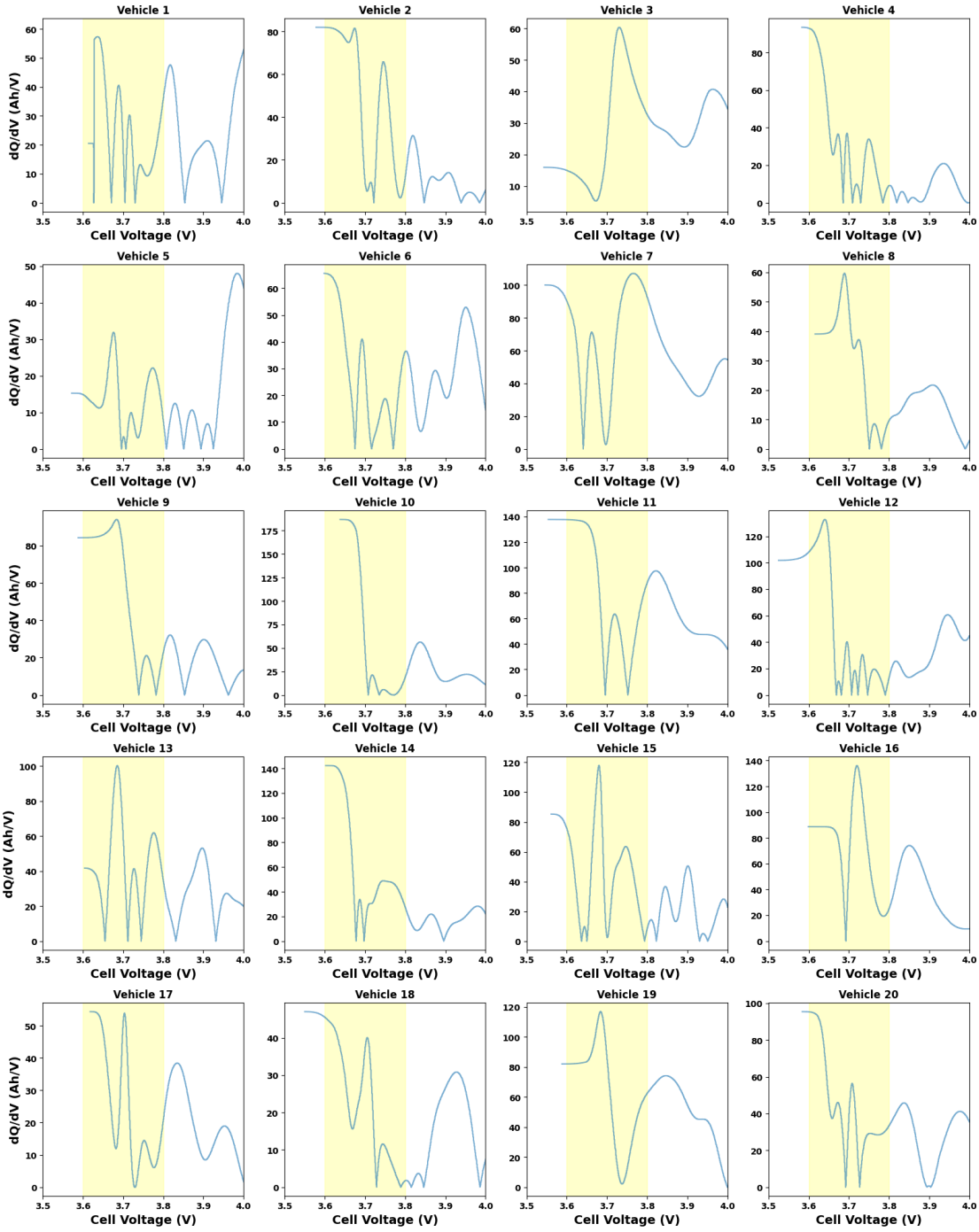

FIGURE 5.4: IC analysis performed on vehicles from the dataset.

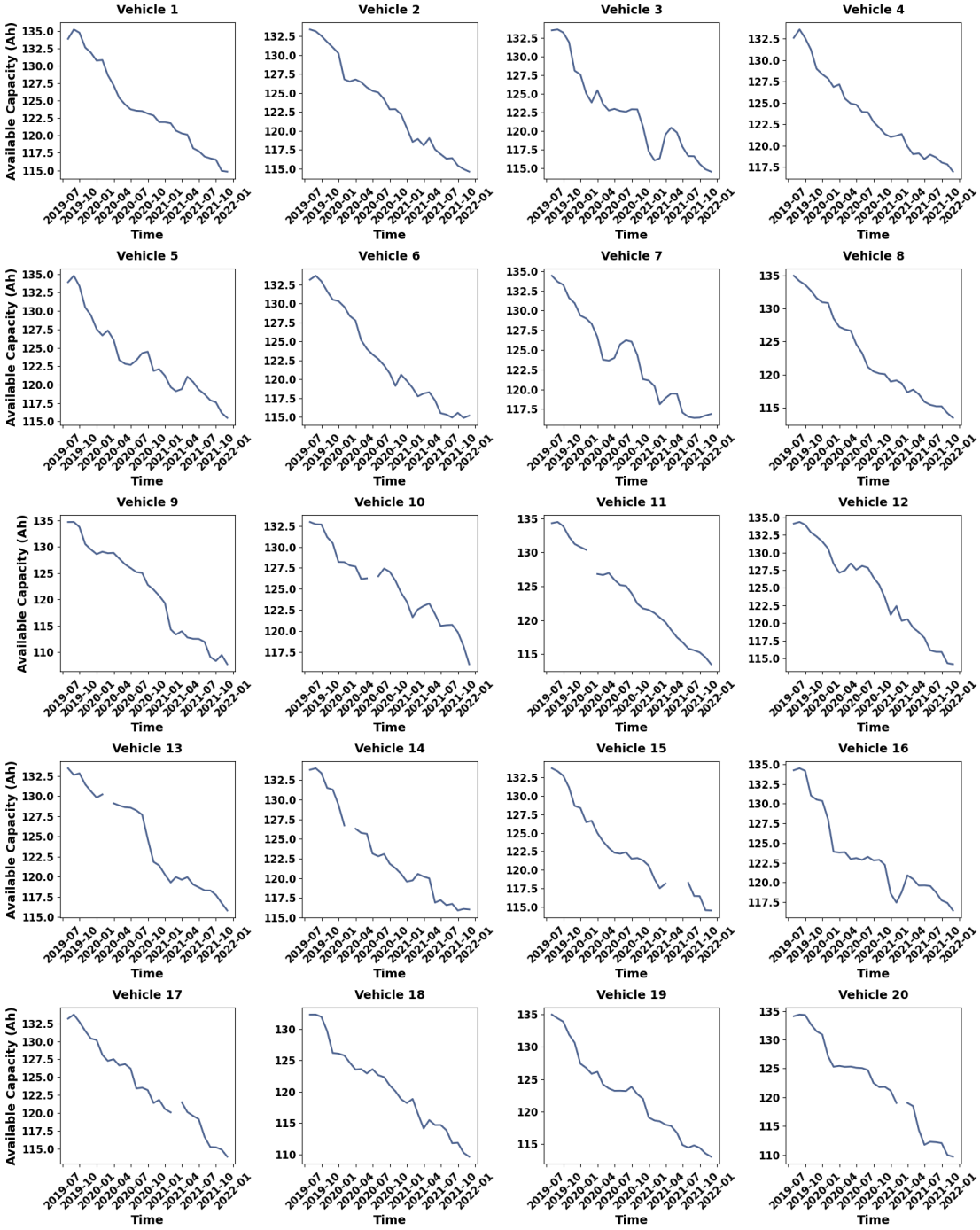

Figure 5.5: Line plot for the monthly median SoH of each of the 20 vehicles in the dataset over time. Aggregation is done with data points within the voltage range determined through IC analysis.

5.4.1 Pre-training

Starting with the pre-training part, there are multiple approaches that can be used as discussed in Chapter 4 (Section 4.3). For this specific application, auto-encoding was used because of its simplicity of implementation and relatively lower computational requirements. The auto-encoder architecture used in this work is based on the self-supervised Time Series Masked Autoencoders (Ti-MAEs) [145].

The recent surge in SSL has enabled the development of masked data modelling approaches that have been successfully applied across a wide range of domains. At its core, masked data modelling involves "masking" portions of the input data and training models to accurately reconstruct the masked regions, which facilitates the extraction of robust and transferable representations without reliance on large-scale annotated datasets [146]. This paradigm parallels the progress observed in masked language modelling within natural language processing (NLP), notably with models like BERT [115], and has been generalized to other modalities by appropriately adapting both the masking strategy and the reconstruction task [146], [147].

In computer vision, masked image modelling has evolved as a prominent pre-training strategy. Multi-view masked autoencoders have demonstrated state-of-the-art performance by reconstructing missing patches while retaining global semantic information from images [148]. Researchers have extended these techniques to specialized applications such as medical image segmentation and COVID-19 detection by exploiting region-guided masking strategies to focus on clinically relevant areas [149], [150]. Moreover, approaches that integrate contrastive learning with masked reconstruction have shown improved performance by enforcing additional semantic constraints during pre-training [151].

When adapting masked modelling to time series, two masking strategies have been explored. Traditional methods for forecasting typically adopt a continuous masking

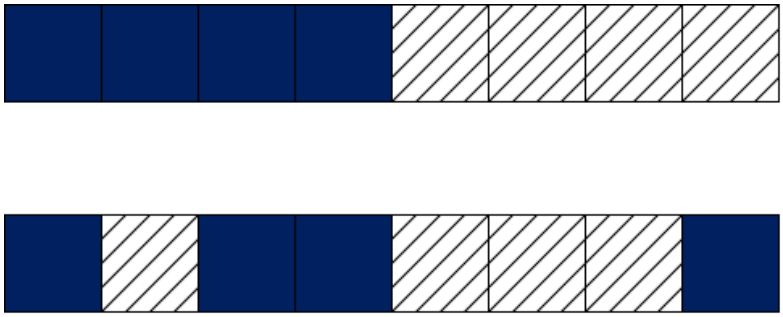

FIGURE 5.6: A visual comparison between different masking approaches for time series data, with dashed blocks being the masked portions. Top figure shows the continuous approach, while the bottom shows the random approach.

approach, where future time steps are masked while past observations are provided to the model. Although this strategy aligns with the sequential nature of forecasting, it restricts the information available during training and can exacerbate distribution shift, especially when the forecasting horizon exceeds the observed window.

Ti-MAE addresses these issues by employing a random masking strategy that masks a fraction of the tokens across the entire sequence. This design choice not only prevents the model from simply interpolating between neighbouring points but also makes use of the full distribution of the input data. The resulting representations capture both local and global temporal dynamics and are more directly aligned with the demands of downstream forecasting and classification tasks. Fig. 5.6 shows a comparison between both approaches.

At a high level, Ti-MAE follows a classical autoencoder paradigm with two primary components (depicted in Fig. 5.7):

- Encoder: Processes the input time series after embedding and random masking. Importantly, only the visible (unmasked) tokens are fed into the encoder Transformer-based blocks, which substantially reduces the computational and memory overhead.

- Decoder: Receives the latent representation generated by the encoder along with corresponding learnable mask tokens. The decoder's role is to reconstruct the original time series at a point-level, ultimately enabling the model to learn relevant representations through reconstruction.

The raw multivariate time series, as denoted in Eq. 5.2, is first transformed into a token representation. This is accomplished by applying a one-dimensional convolutional layer directly over the temporal axis (Eq. 5.3). The convolution serves to extract local temporal features that capture short-term dynamics effectively.

$$\mathbf{X} = [x_1, x_2, \ldots, x_T] \in \mathbb{R}^{T \times m} \tag{5.2}$$

Where X is the time series data matrix, T is the sequence length (number of time steps), and m is the number of features (or channels) in the time series.

$$h_t = W * x_t + b \tag{5.3}$$

Where $x_t \in \mathbb{R}^m$ represents the input at time t, W and b are the convolution weights and bias, and $*$ denotes the convolution operation.

To preserve the ordering and temporal structure, fixed sinusoidal positional embeddings, as commonly adopted in Transformer architectures [152] (Eq. 5.4), are added to the convolved features. Notably, we deliberately refrain from incorporating additional hand-crafted or task-specific embeddings to minimize the inductive bias imposed on the model and to focus on learning representations directly from the raw input data.

$$\mathrm{PE}(t, 2i) = \sin\left(\frac{t}{10000^{2i/d}}\right), \quad \mathrm{PE}(t, 2i+1) = \cos\left(\frac{t}{10000^{2i/d}}\right) \tag{5.4}$$

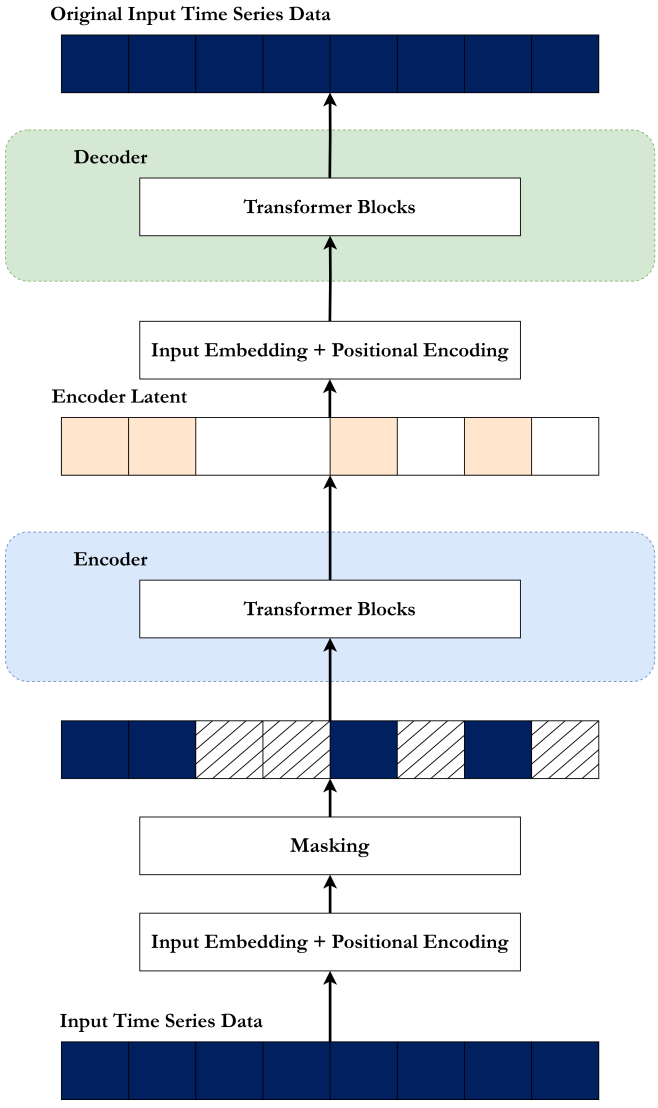

Figure 5.7: A conceptual visualization of the masked autoencoding process employed by Ti-MAE and used for pre-training in the proposed approach.

Where t is the time step, i is the dimension index, and d is the embedding dimension.

Once tokenized, the embedded sequence is subjected to a random masking operation. Specifically, a predefined portion of the tokens is masked out at random using uniform sampling without replacement. Experiments on choosing masking ratios are discussed in Chapter 6 (Section 6.3). This strategy differs from traditional continuous masking approaches that target only future or contiguous segments. Instead, the random masking paradigm ensures that, across training iterations, the model is exposed to varied combinations of visible and masked tokens. This masking strategy yields important advantages:

- Enhanced Generalization: By forcing the model to rely on incomplete information, the network is encouraged to learn high-level semantic and temporal relationships that extend beyond local neighbours.
- Mitigation of Distribution Shift: Since the entire input sequence is considered across iterations, the gap between the training masking regime and the inference stage is reduced.

The encoder in Ti-MAE comprises a stack of Transformer blocks configured in a pre-norm fashion. That is, each block applies layer normalization before the self-attention (Eq. 5.5 and Eq. 5.6) and feed-forward network (Eq. 5.7) sublayers. This architecture ensures stability during training and allows the model to capture long-range dependencies among the visible tokens. Fig. 5.8 shows blocks of a typical Transformer in our encoder scenario [152]. The same blocks apply to the decoder.

Critically, only the unmasked tokens are processed through the encoder. This selective encoding not only reduces the computational burden but also compels the network to derive as much contextual understanding as possible from the limited set of available tokens.

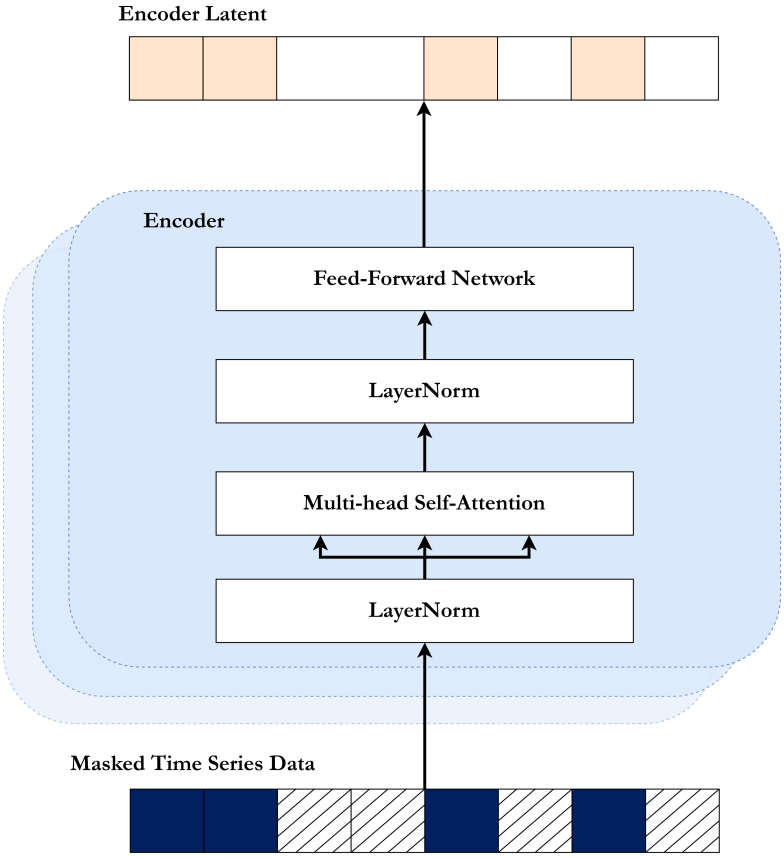

Figure 5.8: Block diagram of operations in a typical Transformer.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \tag{5.5}$$

Where Q, K, and V are the Query, Key, and Value matrices.

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V \tag{5.6}$$

Where $X \in R^{Td}$ is the sequence of token embeddings, and W_Q, W_K, W_V are the learnable weight matrices.

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{5.7}$$

Where x is the input vector, W_1, W_2, b_1, and b_2 are the learnable weights and biases.

The decoder's task is to reconstruct the original time series at the point level. To this end, the input to the decoder is formed by combining:

- The encoded representations corresponding to the visible tokens.
- A set of learnable mask tokens, which stand in for the masked positions.

Before entering the decoder, positional embeddings are added to the combined tokens to maintain the temporal order. The decoder then processes this unified sequence using Transformer blocks, after which a linear projection layer maps the output back to the original time series dimensions. The reconstruction target is defined at the level of individual time steps, and the objective is to minimize the discrepancy between the reconstructed values and the original ones.

5.4.2 Fine-tuning

While the pre-trained Ti-MAE framework learns latent representations of time-series data through a self-supervised masked reconstruction task, the fine-tuning phase utilizes these representations for a supervised regression problem. In contrast to the classification task described in the original paper, our objective is to accurately predict the current month's SoH using an LSTM-based regression model that combines latent representations and historical SoH values. The following lines go over the methodology for the fine-tuning phase, which is also depicted in Fig. 5.9

After pre-training with unlabelled data, the Ti-MAE model provides a latent representation for each input sequence. Let the raw multivariate time series for a given month be denoted by Eq. 5.8.

$$\mathbf{X} \in \mathbb{R}^{B \times T \times m} \tag{5.8}$$

where B is the batch size, T is the number of time steps (historical months), and m is the feature dimension.

Also let the past SoH values for $T-1$ months be as denoted by Eq. 5.9:

$$\mathbf{C} \in \mathbb{R}^{B \times (T-1) \times 1} \tag{5.9}$$

These values are available for all months except the current month, whose SoH is to be predicted.

For each time step $t \in \{1, \ldots, T\}$, the pre-trained Ti-MAE encoder generates a latent representation from the corresponding slice of the input sequence as in Eq. 5.10:

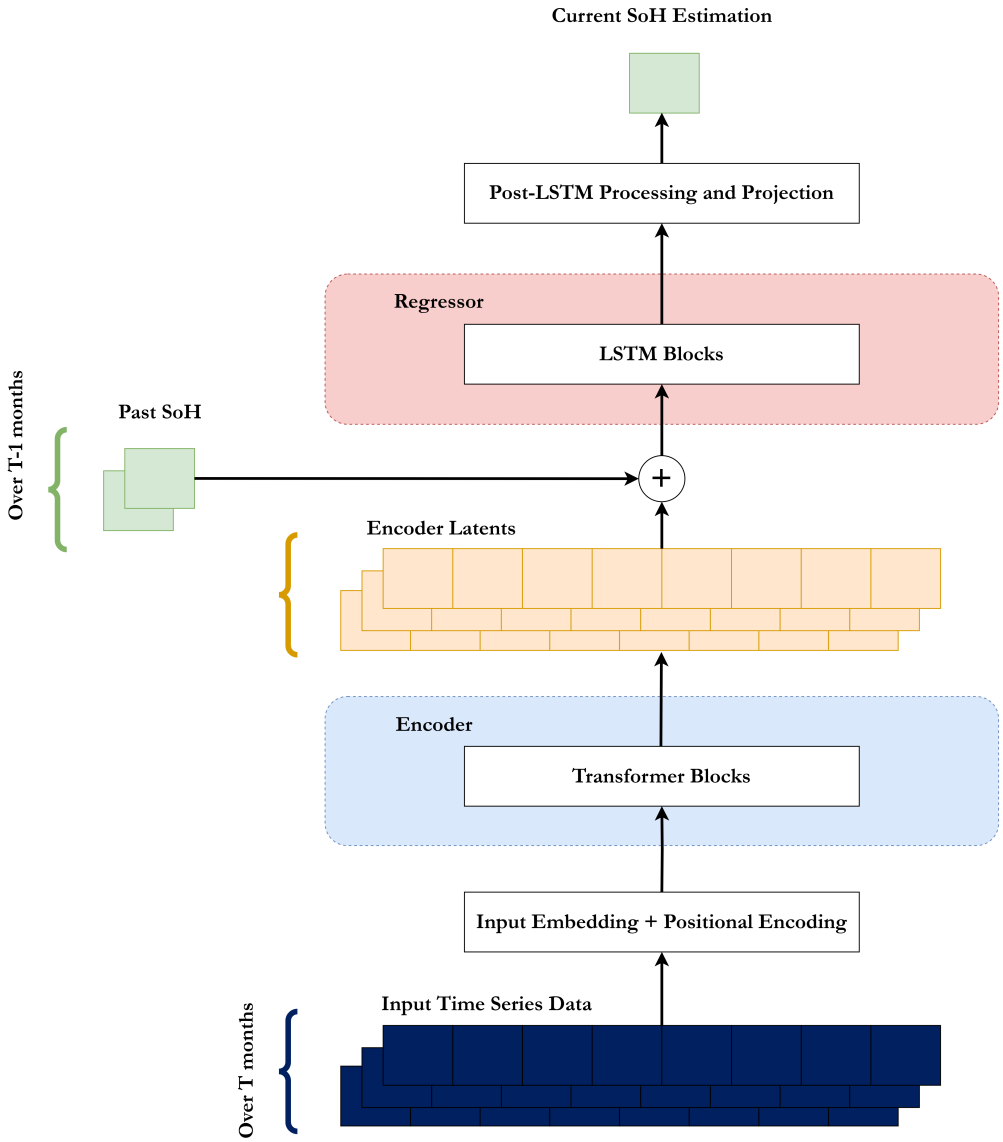

Figure 5.9: A conceptual visualization of the regression process used for fine-tuning in the proposed approach.

$$\mathbf{L}_t = \text{Encoder}(\mathbf{X}_{:,t,:}) \in \mathbb{R}^{B\times E}. \tag{5.10}$$

Here, E is the embedding dimension, and $\mathbf{X}_{:,t,:}$ denotes the data for time step t across the batch. By concatenating the latent representations from all time steps, we form the latent sequence in Eq. 5.11:

$$\mathbf{L} = \left[\mathbf{L}_1; \mathbf{L}_2; \ldots; \mathbf{L}_T\right] \in \mathbb{R}^{B\times T\times E}. \tag{5.11}$$

This sequence represents the full series of latent features extracted from the input. To align the latent features with the LSTM's hidden dimension H, we project $\mathbf{L}$ as follows:

$$\mathbf{L}_{\text{proj}} = \text{LatentProj}(\mathbf{L}) \in \mathbb{R}^{B\times T\times H}. \tag{5.12}$$

Similarly, the past capacity sequence is projected to the same dimension:

$$\mathbf{C}_{\text{proj}} = \text{CapacityProj}(\mathbf{C}) \in \mathbb{R}^{B\times (T-1)\times H}. \tag{5.13}$$

These projection operations (implemented as learned linear layers) transform the latent and capacity inputs into a common representation space.

Since the current month's capacity is unknown, we pad the projected capacity sequence with a zero vector to match the temporal dimension of $\mathbf{L}_{\text{proj}}$:

$$\mathbf{C}_{\text{pad}} = \left[\mathbf{C}_{\text{proj}}; \mathbf{0}\right] \in \mathbb{R}^{B\times T\times H}.$$

We then combine the features by taking the element-wise sum of the projected latent sequence and the padded capacity:

$$\mathbf{Z} = \mathbf{L}_{\text{proj}} + \mathbf{C}_{\text{pad}} \in \mathbb{R}^{B\times T\times H}.$$

In this formulation, the latent representations from past months are augmented with the corresponding SoH values, while the current month's input consists solely of its latent.

The combined feature sequence $\mathbf{Z}$ is processed by a bidirectional Long Short-Term Memory (LSTM) to capture temporal dependencies in both forward and backward directions. The LSTM updates its hidden state according to the following equations for each time step and for each direction, depicted in Fig. 5.10 [153]:

$$\begin{aligned}
i_i &= \sigma\left(W_i \begin{bmatrix} h_{i-1} \\ x_i^* \end{bmatrix} + b_i\right), \\
f_i &= \sigma\left(W_f \begin{bmatrix} h_{i-1} \\ x_i^* \end{bmatrix} + b_f\right), \\
o_i &= \sigma\left(W_o \begin{bmatrix} h_{i-1} \\ x_i^* \end{bmatrix} + b_o\right), \\
\tilde{c}_i &= \tanh\left(W_c \begin{bmatrix} h_{i-1} \\ x_i^* \end{bmatrix} + b_c\right), \\
c_i &= f_i \odot c_{i-1} + i_i \odot \tilde{c}_i, \\
h_i &= o_i \odot \tanh(c_i)\,,
\end{aligned} \tag{5.14}$$

where:

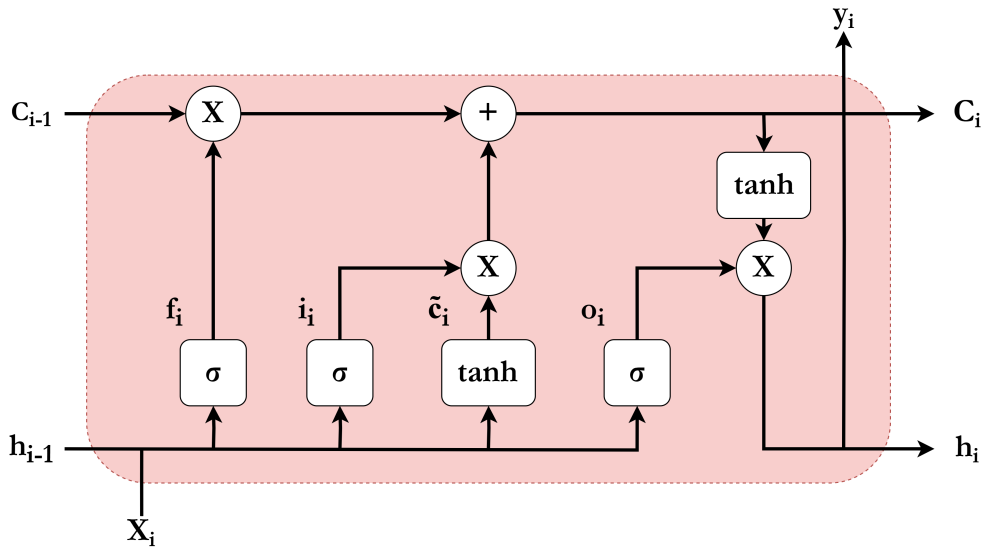

Figure 5.10: A typical LSTM block diagram.

- $x_i^* \in R^{d_x}$ is the input vector at time step i,
- $h_{i-1} \in R^{d_h}$ and $c_{i-1} \in R^{d_h}$ denote the previous hidden and cell states, respectively,
- i_i, f_i, and o_i are the input, forget, and output gates,
- $\tilde{c}_i$ is the candidate cell state,
- W_i, W_f, W_o, W_c and b_i, b_f, b_o, b_c are learnable parameters,
- $\sigma(.)$ represents the sigmoid function,
- tanh(.) is the hyperbolic tangent activation, and
- $\odot$ denotes the element-wise product.

The bidirectional LSTM yields output sequence $\mathbf{H}$ and final hidden states:

$$\mathbf{H},\ (\mathbf{h}_n, \mathbf{c}_n) = \text{BiLSTM}(\mathbf{Z}) \in \mathbb{R}^{B \times T \times 2H}.$$

Here, the output $\mathbf{H}$ has dimension $2H$ due to the concatenation of the forward and backward passes. To emphasize the most informative time steps, a temporal attention mechanism is applied to $\mathbf{H}$:

$$\mathbf{A} = \text{Attention}(\mathbf{H}) \in \mathbb{R}^{B\times T\times 1},$$

which computes an attention score for each time step. These scores are normalized using a softmax function:

$$\boldsymbol{\alpha} = \text{softmax}(\mathbf{A}).$$

A weighted sum of the LSTM outputs produces an aggregated feature vector:

$$\mathbf{V} = \sum_{t=1}^{T} \alpha_t \mathbf{H}_t \in \mathbb{R}^{B\times 2H}.$$

This vector $\mathbf{V}$ succinctly summarizes the temporal dynamics based on the learned attention weights. The aggregated vector $\mathbf{V}$ is fed through several fully connected layers with skip connections for further refinement:

$$\begin{aligned}
\mathbf{X}_1 &= \text{Layer}_1(\text{PostLSTM}(\mathbf{V})),\\
\mathbf{X}_2 &= \text{Layer}_2(\mathbf{X}_1) + \mathbf{X}_1, \quad \text{(Skip Connection)}\\
\mathbf{X}_3 &= \text{Layer}_3(\mathbf{X}_2),\\
\mathbf{X}_4 &= \text{Layer}_4(\mathbf{X}_3).
\end{aligned}$$

Each layer is defined as:

$$\text{Layer}_i(\mathbf{X}) = \text{Dropout}\Big(\text{GELU}\big(\text{LayerNorm}(\text{Linear}(\mathbf{X}))\big)\Big).$$

The skip connections facilitate effective gradient propagation and help preserve salient information. Finally, the refined feature vector $\mathbf{X}_4$ is passed through a final linear layer to produce the SoH prediction for the current month:

$$\mathbf{y} = \text{FinalLayer}(\mathbf{X}_4) \in \mathbb{R}^{B\times 1}.$$

The regression loss, based on the discrepancy between $\mathbf{y}$ and the ground truth SoH, is used to optimize the model during fine-tuning.

Our specific implementation of the proposed approach is visualized in the form of a cycle in Fig. 5.11. The next chapter discusses details of data extraction during the training phase, the different experiments performed and their goals, the results realized, a discussion of the results, and recommended future work.

5.5 Conclusion

This chapter turned the ideas from earlier in the thesis into a step-by-step plan for estimating SoH across a whole fleet of electric vehicles. First, it showed how SSL, FL, and cloud computing can work together so that each car keeps its raw data private but still helps improve a shared model. Next, it introduced the real charging dataset we will use, explained how we calculated true SoH values from those records, and described a simple filtering trick to remove sensor noise. The chapter then walked through the model itself: a masked autoencoder that learns useful patterns from short voltage slices, followed by a bidirectional LSTM that combines those patterns with past SoH readings

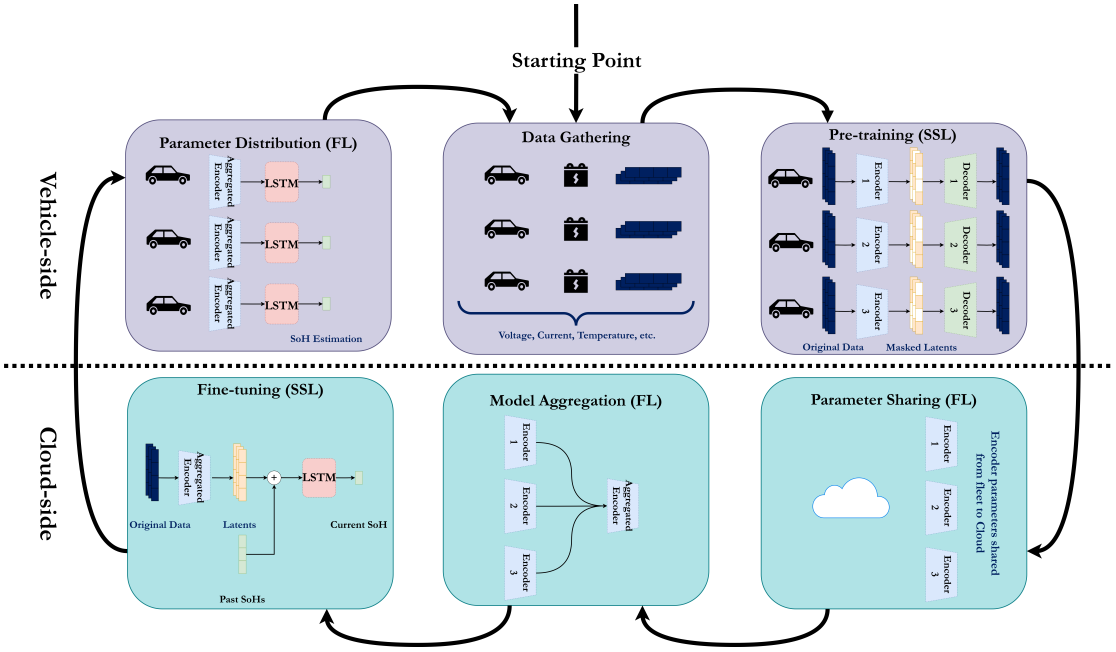

Figure 5.11: The proposed approach cycle with our specific implementation using Ti-MAE and LSTM.

to predict the current month's battery health. Finally, it showed how all these pieces fit into a loop that can keep training and updating the model over time.

With the method now fully laid out, the next chapter will test how well this approach works in practice and see which parts matter most for accurate, privacy-friendly SoH estimation.

Chapter 6

Experiments, Results and Discussions

6.1 Introduction

Now that we have laid the groundwork by thoroughly reviewing the topics implemented in our proposed framework and presenting the models used, along with their intricacies, it is time to put this knowledge into action. In this chapter, we aim to look for answers to the 4 main questions regarding our proposed framework:

RQ1 Does pre-training with masked auto-encoding on unlabelled drive cycles enhance State-of-Health (SoH) prediction compared with purely supervised baselines under limited labelled data?

RQ2 How does the choice of *masking ratio* in Self-Supervised Learning (SSL) influence representation quality and downstream SoH accuracy?

RQ3 How do different fine-tuning strategies for the SSL encoder (fully frozen, full unfreeze) affect model performance and training stability?

RQ4 Does Federated Learning (FL) aggregation for model parameters degrade performance compared with centralized training?

We conducted experiments targeting each of those questions. This chapter starts by detailing procedures and model hyperparameters that are common to all experiments (Section 6.2), then goes over the experiments one by one (Section 6.3). The chapter explains the approach taken to answer the question targeted by the experiments, shows and discusses the results produced from implementing this approach (Section 6.4).

6.2 Common Procedures in Training and Testing

6.2.1 Data Preparation and Preprocessing

Initial preprocessing of the dataset is presented in Section 5.3. Preparation of the data depends on the phase, whether it is pre-training or fine-tuning.

Pre-training Data Preparation

The data is first split into training and validation sets. There is no testing done for the pre-training phase. Testing is done on the overall system after fine-tuning is done by comparing the estimated SoH with the groundtruth. The train-validation split ratio used in our work is 0.85, always preserving the chronological order of the records. This is crucial for time series tasks, as it prevents information from the future from leaking into the past, which could otherwise lead to overly optimistic model performance. Features are then normalized using min-max scaling (Eq. 6.1) to ensure all input variables are on a comparable scale. The data is then segmented into overlapping windows of a fixed sequence length, which serve as the model's input samples. Notably, this process is entirely deterministic, i.e., no randomization is involved, so that each possible window

is included in the dataset in order. This is not the case with the fine-tuning data preparation, as will be discussed next.

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{6.1}$$

Fine-tuning Data Preparation

As presented in Section 5.3, random charging sequences can be used for SoH estimation. So, the approach here is different because we want to take a random sample of data from previous months of count T, and use them to predict the SoH of the current month. Therefore, the data split is done on a month level, rather than a sliding window like in pre-training. We start by finding all months present in the data, and splitting these months into training and validation sets, still preserving the chronological order of the data. Testing is done on whole data sets, as will be discussed later. For example, if our data points lie in the date range of January 2020 to December 2020, one option would be to have the training set be the points from January to August, and the validation set be the data points from September to December. The testing set would be data from another vehicle for the whole period from January to December.

The next step involves the extraction of random charging sequences from each of the months selected for the respective dataset (training, validation, or testing). We implemented two modes for sampling charging sequences:

- Non-sequential sampling: In this mode, we select a random window of months of size T (the historical window length). The features from a random 160s charging sequence from all T months and the SoH from the first $T-1$ months are extracted and used for the fine-tuning process, described in 5.4.2. This produces one data point for fine-tuning. Therefore, this process of randomly selecting a window of months and then randomly selecting a 160s charging sequence from within the

month is repeated a number of times, which we defined as the length of the data set. Continuing on our previous example for the training set and taking $T = 5$ for the sake of this example (this is not the value we used in our experiments), the first random window could be the months March through July. In this case, we are aiming to estimate the SoH in July. Our algorithm extracts a random 160s charging sequence from all these months, in addition to the SoH of the months March - June, to be fed to our model. This produces our first data point. The next data point could be from the months January - May, and so on. We use this mode for training and validation in order to allow data shuffling for our approach.

- Sequential sampling: In this mode, instead of selecting random windows of months, we implement a sliding window on a month level. That is, if the window is of size 5, as in our previous example for the training set, the first window will be the months January - May, the next window will be February - June, the next window will be March - July, and finally April - August. A random charging sequence is still being sampled from each month in each window. We use this mode for testing in order to simulate a real-world scenario where the SoH is predicted month after month. This is also coupled with feeding the model its own previous predictions for an exact real-world scenario, where the Electric Vehicle (EV) does not have access to the real labels or corrections and only builds upon its own previous estimations.

Finally, the min-max scaling is used for normalization, similar to the pre-training data preparation.

6.2.2 Models Hyperparameters and Procedures

Models Hyperparameters

The models hyperparameters were common among all experiments. For the pre-training phase, the input dimension is set to 8, since our data contains 8 features as listed in

Table 5.1. Time was not used as input for the model. Since we are using a charging sequence length of 160 seconds as stated in Section 5.3, and since the data is sampled every 8 seconds, the sequence length is selected to be 20. The number of encoder and decoder heads is set to 3, while the number of layers for each is 1. The intermediate dimension for both is set to 64, with the embedding size set to 30. GELU was selected for the activation function. The masking ratio is experiment-dependent, as will be discussed later. AdamW was selected as the optimizer [154], with a learning rate of 5×10^{-3}. The scheduler selected was OneCycleLR for its fast convergence properties [155]. To penalize larger error values, Mean Square Error (MSE) was used for training and validation (Eq. 6.2). The training set size ratio was set to 0.85, leaving the remaining 0.15 for validation. As previously mentioned, no testing was done in the pre-training phase. Rather, testing was only done on the overall task after fine-tuning. No dropouts were utilized in the pre-training phase. The maximum number of epochs is set to 1000, with an early stopping patience of 50.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{6.2}$$

As for fine-tuning, the encoder hyperparameters remain the same, since it is the same encoder trained in pre-training. As for the Long Short-Term Memory (LSTM), the intermediate dimension is set to 128, the number of layers to 2, and bidirectional processing is enabled. The GELU activation function is used throughout, except for in the temporal attention layer, which uses Tanh (the model architecture is described in Section 5.4.2). The history length was set to 2, which means that only the features and SoH of one previous month are used to predict the SoH of the current month. AdamW was used again as the optimizer with the same learning rate of 5×10^{-3}. OneCycleLR as a scheduler for fine-tuning showed very poor convergence. Instead, ReduceLROnPlateau was used with a factor of 0.5 and a patience of 5, which showed immense improvement.

MSE was also used for training and validation. The training split ratio was again 0.85, with 0.15 for validation. Whole datasets were used for testing. No masking was performed during fine-tuning, and no dropouts were used either. The maximum number of epochs is set to 1000, with a high early stopping patience of 150, due to the models' validation loss plateauing then continuing to decrease after a large number of epochs. Finally, the batch size for both phases is 2048. The hyperparameters are summarized in Table 6.1.

Table 6.1: Summary of hyperparameters for pre-training and fine-tuning phases.

Phase	**Hyperparameter**	**Value**
Pre-training	Input Dimension	8
	Sequence Length	20
	Encoder/Decoder Heads	3
	Number of Layers	1
	Intermediate Dimension	64
	Embedding Size	30
	Activation Function	GELU
	Optimizer	AdamW
	Learning Rate	5×10^{-3}
	Scheduler	OneCycleLR
Fine-tuning	LSTM Intermediate Dimension	128
	LSTM Number of Layers	2
	LSTM Bidirectional	Enabled
	Activation Function (General)	GELU
	Activation Function (Attention)	Tanh
	History Length	2
	Optimizer	AdamW
	Learning Rate	5×10^{-3}
	Scheduler	ReduceLROnPlateau
	Factor (ReduceLROnPlateau)	0.5
	Patience (ReduceLROnPlateau)	5
Batch Size (Both Phases)		2048
Max Epochs (Both Phases)		1000
Early Stopping Patience		50 (pre) / 150 (fine)

Since there is random action in the fine-tuning phase as explained in 6.2.1, a seed of 42 is selected to maintain the same random selections in all training. For testing, the

selected seed was 0. To report on testing loss in Ah, Mean Absolute Error (MAE) is used (Eq. 6.3).

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{6.3}$$

The detailed breakdown of the number of parameters of the pre-training and fine-tuning models are listed in Tables 6.2 and 6.3 respectively.

Table 6.2: Parameter count and memory footprint of the **pre-training** network (Ti-MAE encoder-decoder).

Component	**# Params ($\times 10^3$)**
Encoder (TiMAEEncoder)	7.6
Decoder (TiMAEDecoder)	8.0
Total	**15.6**
Estimated size: **0.062 MB**.	

Table 6.3: Parameter count and memory footprint of the **fine-tuning** network (encoder + LSTM regressor).

Component	**# Params ($\times 10^3$)**
Encoder (TiMAEEncoder)	7.6
Latent projection	4.0
Capacity projection	0.256
LSTM	659
Post-LSTM (Seq.)	33.2
Regressor_layer1	33.5
Regressor_layer2	66.3
Regressor_layer3	33.2
Regressor_layer4	8.4
Final layer	0.065
Temporal attention	33.3
Total	**878**
Estimated size: **3.515 MB.**	

Pre-training phase Training Procedure

This procedure is common for all pre-training phases in our experiments. First, the vehicles whose data will be used for pre-training are determined. Next, for each vehicle,

a separate model is trained on that vehicle's data to simulate the process of pre-training being done on each vehicle. Finally, the parameters of all trained models are collected, and federated aggregation is performed on them to generate a new model that has learned from the data of all vehicles. In our work, the FedAvg technique has been employed for aggregation, which calculates the average of each parameter over all models, as discussed in Chapter 4 (Section 4.7). The encoder from this new model is then passed on to be used for the fine-tuning phase.

Fine-tuning phase Training Procedure

This procedure is common for all fine-tuning phases in our experiments. The model's encoder is initialized using the parameter values from the aggregated encoder coming from the pre-training phase, while the decoder is discarded. As in pre-training, we decide on which vehicles will be used for fine-tuning, which are always separate from the vehicles used in pre-training. This is because we treat the vehicles that are used in pre-training as unlabelled data collected from the field, while vehicles used for fine-tuning are treated as labelled lab data available on the cloud server where fine-tuning is done. Next, we feed the data from selected vehicles into a sequential data loader we implemented, instead of concatenating the datasets from vehicles. The aim of this data loader is to train the model on data from all selected vehicles, while ensuring that no features from two different vehicles are used for a single data point. For instance, if the datasets were concatenated instead, the last month from one vehicle and the first month from the following vehicle could be used to predict the SoH for the second month in the latter vehicle, which does not make sense. This data loader ensures the separation while feeding the model data points from all vehicles.

Testing Procedure

In our testing experiments, we want to simulate a real-world scenario of a vehicle estimating its SoH for its whole lifetime. During a vehicle's lifetime, no lab tests are done on the battery to correct its estimations. Therefore, the vehicle has to rely on and build on its own estimations continuously. To simulate this, we employ the sequential sampling mode for data extraction that we implemented and previously discussed in 6.2.1. Since we are using a history length of 2, the first estimation produced by the model is aware of the starting SoH of the vehicle, and uses it to estimate the SoH for the second month. We then use the model's estimation for the second month as an input for estimating the SoH for the third month, and so on until the end of the dataset.

6.3 Experiments

6.3.1 Experiment 1: Comparative Model Evaluation

The objective of this experiment is to compare our proposed framework which makes use of unlabelled data, with previous approaches that only relied on labelled data. We also expand our experimentation further to include two variants of our framework: one with the pre-trained parts of the model frozen (i.e., not being trained during fine-tuning), and one with the whole model being trained, addressing the discussion brought up in Chapter 4 (Section 4.4).

In this experiment, vehicles #3 to #20 were used as unlabelled for pre-training, while vehicles #1 and #2 were selected for fine-tuning as labelled data. Testing was done on vehicles #3 to #20. We trained and tested 3 different models:

1. A model that follows our proposed approach by being pre-trained on the unlabelled data, then fine-tuned with the labelled data. During fine-tuning, the encoder was

Table 6.4: Aggregate MAE and dispersion across all vehicles.

Metric	Frozen-SSL	Unfrozen-SL	Unfrozen-SSL
Mean MAE (Ah)	**1.57**	1.81	1.65
Median MAE (Ah)	1.64	1.83	**1.54**
Std. dev. (Ah)	**0.39**	0.52	0.53
Vehicle wins*	5	6	**7**

*Vehicle win = lowest MAE for that individual EV.

frozen, while the LSTM was learning. In our figures and plots, this model is referred to as "Frozen-SSL".

2. A model that is only trained on the labelled data. No parts of the model were frozen. In our figures and plots, this model is referred to as "Unfrozen-SL"

3. A model that follows our proposed approach by being pre-trained on the unlabelled data, then fine-tuned with the labelled data. During fine-tuning, the encoder was not frozen, so that the whole model was being fine-tuned. In our figures and plots, this model is referred to as "Unfrozen-SSL".

For Frozen-SSL and Unfrozen-SSL models, the masking ratio used in this experiment was 0.6. Further experimentation with different masking ratios was done in Experiment 2.

Fig. 6.1 shows the SoH estimated by each of the models for each of the test vehicles over the vehicles lifetime. Keep in mind that for each of those estimations, the models use their own previous estimations as input. Fig. 6.2 compares the MAE of the SoH estimation for each model in the respective vehicle. A statistical summary of the MAEs is presented in Table 6.4.

Frozen-SSL exhibits the lowest *mean* error (Table 6.4) and, more importantly, the smallest standard deviation, indicating that weight freezing provides a strong regularization effect. In contrast, the fully supervised Unfrozen-SL attains the highest

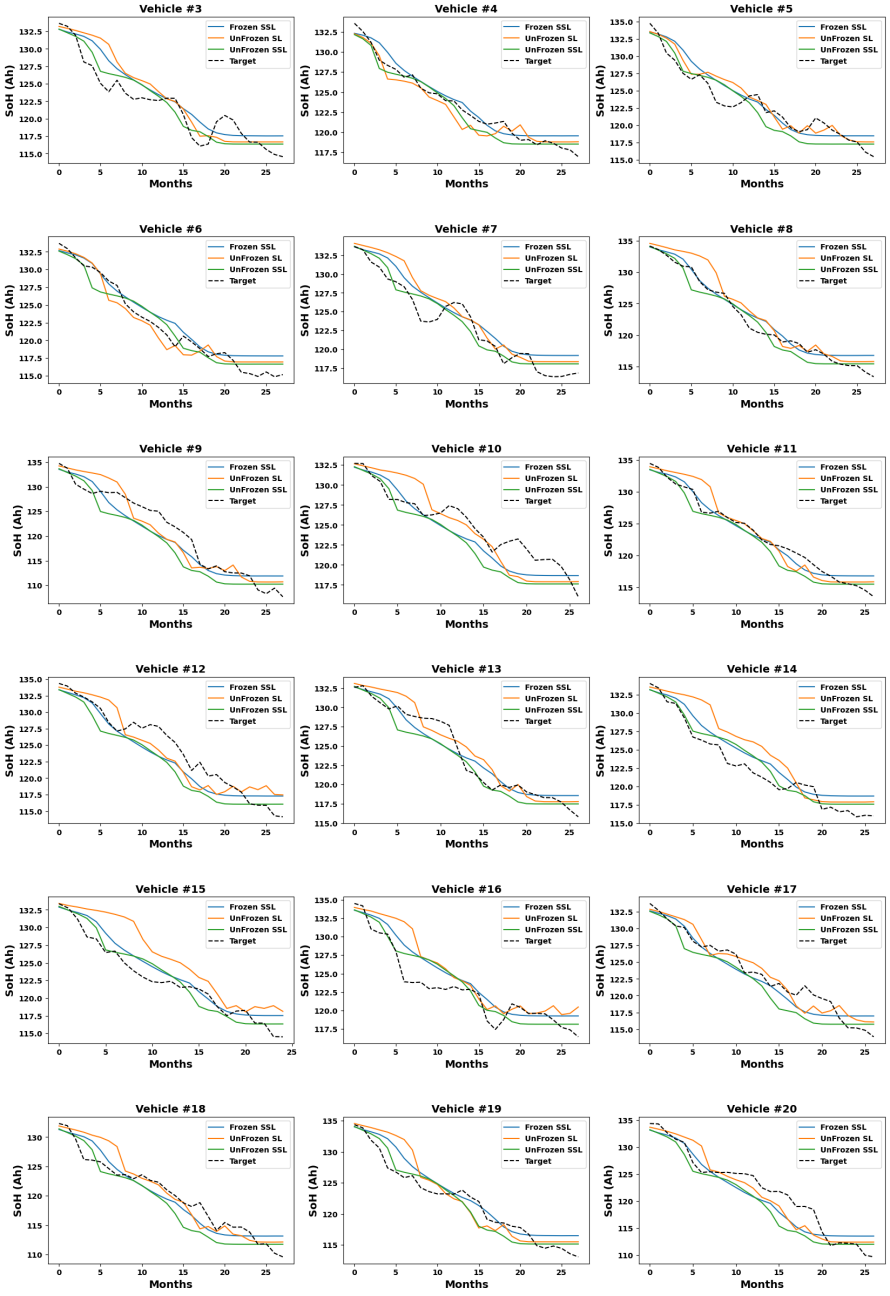

FIGURE 6.1: Month-by-month SoH trajectories estimated by the three models versus the ground-truth target (dashed line) for every test vehicle in the experiment.

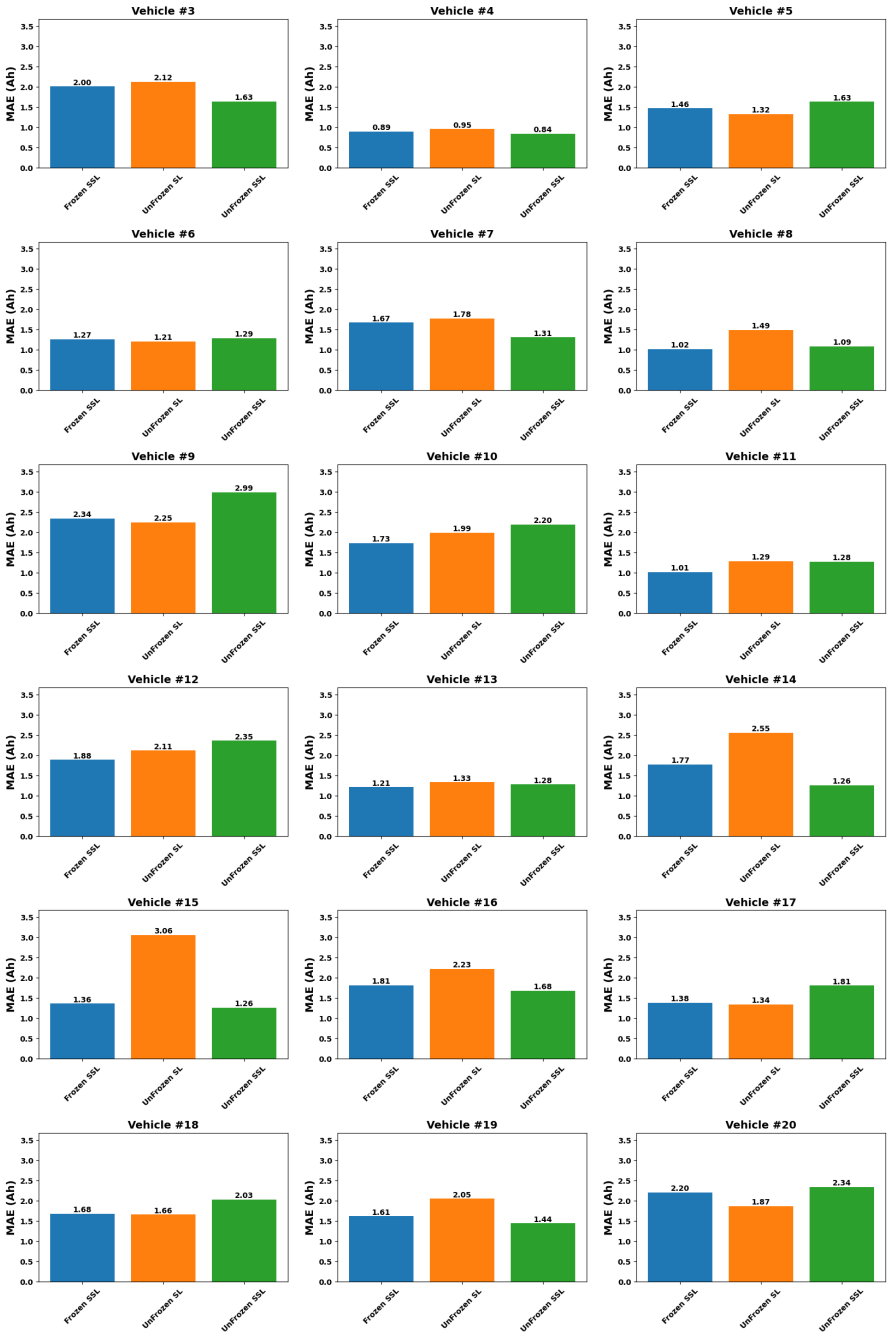

FIGURE 6.2: Per-vehicle MAE of the three models.

dispersion. Its property of changing all its parameters offers flexibility but also increases the risk of overfitting when few labelled cycles are available.

Although Unfrozen-SSL reduces the average error of Unfrozen-SL by $\approx 9\%$, it does so at the cost of higher variance relative to Frozen-SSL. This variance-bias trade-off suggests that the benefit of unfreezing is context-dependent: substantial when a particular battery deviates from the population used during SSL pre-training (specifically with valleys), yet detrimental when the operating regime is already well covered.

Visual inspection of the month-by-month SoH curves (Fig. 6.1) reveals three characteristic phases. **Early life (0-6 months):** All models follow the target closely. Differences emerge mainly as vertical offsets determined by the LSTM. **Mid-life (6-18 months):** Divergence becomes apparent. The frozen encoder yields the smoothest degradation trajectory, seldom overshooting the target capacity. In several vehicles (e.g. #14), Unfrozen-SL displays an overly optimistic SoH because end-to-end training internalizes false calendar drifts. Allowing the SSL encoder to adapt (Unfrozen-SSL) corrects most of this bias, but its not-so-smooth degradation can result in worse performance when inaccurate sensor reading result in unusual SoH patterns (see #9). **Late life ($\geq$22 months):** As capacity fade saturates, all models converge to a plateau. Nonetheless, Frozen-SSL and Unfrozen-SSL end within ≤ 1 Ah of the true value, whereas Unfrozen-SL occasionally undershoots.

If fleet-wide robustness is paramount, Frozen-SSL is preferred. It delivers a 13.3 % average MAE reduction, with up to 35 % reduction, compared with the supervised baseline, while decreasing the risk of large individual errors by 25 %. In settings with anticipated domain shift, for instance, fast-charging taxis or vehicles operating in extreme climates, unfreezing the encoder is advantageous. Unfrozen-SSL lowers the error by up to 59 % in the most atypical vehicles (#14-#16). Conversely, Unfrozen-SL is competitive only when abundant labelled data exists for the target battery and stringent

safety margins are relaxed.

The foregoing analysis motivates a hybrid over-the-air (OTA) update strategy. A fleet can adopt Frozen-SSL as the default firmware, activating the Unfrozen-SSL only after several consecutive cycles exceed a residual-error threshold. Such conditional adaptation harnesses the stability of a frozen encoder while retaining the flexibility to cope with emerging operating envelopes.

Self-supervised pre-training demonstrably enhances SoH estimation. Even the worst SSL variant outperforms the fully supervised baseline on median error. Freezing the encoder provides the safest option in terms of variance and outlier control, whereas selective unfreezing unlocks considerable headroom when batteries deviate from population norms. These findings lead us to shift towards an approach that uses a hybrid of our two SSL models based on the anticipated deployment environment and of devising OTA policies that exploit the complementary strengths of both frozen and unfrozen paradigms.

6.3.2 Experiment 2: Impact of Masking Ratio

This experiment analyses how different masking ratios affect our two SSL variants, Frozen-SSL and Unfrozen-SSL, and how both compare with the fully supervised baseline (Unfrozen-SL). In the original Time Series Masked Autoencoder (Ti-MAE) paper [145], it is claimed that the masking ratio is related to the information density and redundancy of the data. They note that natural language has a high information density, thus requiring a lower masking ratio. For example, BERT uses 15% as its masking ratio [115]. On the other hand, images and time series data contain more redundancy, making a higher masking ratio ($\approx 75\%$) beneficial for learning high-level semantic features. A lower masking ratio leads to the model focusing on local low-level semantic

Table 6.5: Average MAE across 18 test vehicles for each masking ratio. Δ is the percentage change relative to the SL baseline (1.81 A h).

Mask ratio	Frozen-SSL	Δ (%)	Unfrozen-SSL	Δ (%)
0.00	1.58	−13	2.03	12
0.10	1.85	2	1.71	−6
0.20	2.04	13	1.69	−7
0.35	1.79	−1	**1.50**	**-17**
0.50	2.03	12	1.80	0
0.60	**1.57**	−13	1.65	−9
0.75	1.64	−9	1.73	−4

features, potentially relying on interpolation for reconstruction rather than understanding the broader patterns. Conversely, a higher masking ratio degrades performance due to excessive loss of visible tokens, preventing the model from effectively capturing useful representations.

Table 6.5 and Fig. 6.3 show that the Unfrozen-SSL model attains its lowest MAE at a masking ratio of **0.35**, achieving a 17 % reduction relative to the SL baseline. Errors increase again when the ratio is pushed closer to either extreme (0 or 0.75), yielding the typical U-shaped curve observed in masked reconstruction objectives. Frozen-SSL, by contrast, never outperforms the unfrozen variant and only matches or beats SL at the ends of the spectrum (0.00 and 0.60), or at the same 0.35 ratio at which the unfrozen variant performed best.

Except for the zero-masking case, unfreezing consistently lowers the MAE (blue vs. green curves). The benefit peaks at the same ≈0.35-0.40 window where the overall error is smallest, lending empirical support to the intuition that adapting the pre-trained representation to the downstream distribution is most useful once the pre-training task becomes sufficiently challenging.

Figures 6.4 and 6.5 reveal considerable cross-vehicle variability, yet the unfrozen model not only shifts the mean downwards but also tightens the spread. The standard

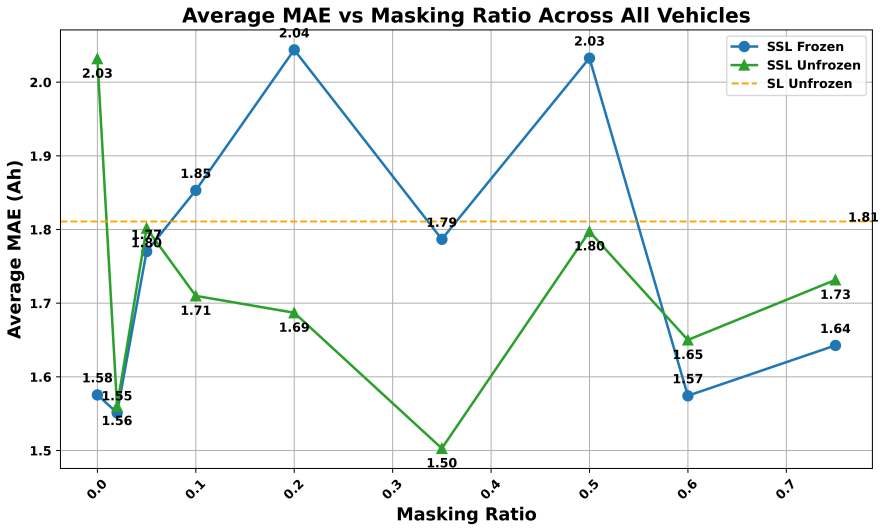

FIGURE 6.3: Fleet-wide average MAE as a function of masking ratio. The dashed orange line denotes the SL baseline (1.81 A h).

deviation drops from roughly 0.42 A h (frozen) to 0.31 A h at the optimum ratio, and the box plots (Fig. 6.6) show fewer and less extreme outliers. It also reaches a maximum improvement of 60% (Vehicle #15). Vehicles #9 and #14 remain challenging when masking exceeds 50 %, suggesting a possible benefit from vehicle-specific fine-tuning in production settings.

From these findings, we can categorize the masking ratios into 3 categories:

- **Too little masking** ($r = 0$)**:** With no tokens removed, the auto-encoder solves a near-identity problem and learns little beyond the supervised objective. UNFROZEN-SSL even lags behind SL here.

- **Too much masking** ($r \geq 0.5$)**:** Excessive masking deprives the model of the context it needs to reconstruct the signal, causing both SSL variants to drift back toward, or above, the SL baseline.

- **Sweet spot** ($r \approx 0.3{-}0.4$)**:** Moderate masking forces the network to model longer-range temporal dependencies while retaining enough context for the reconstruction

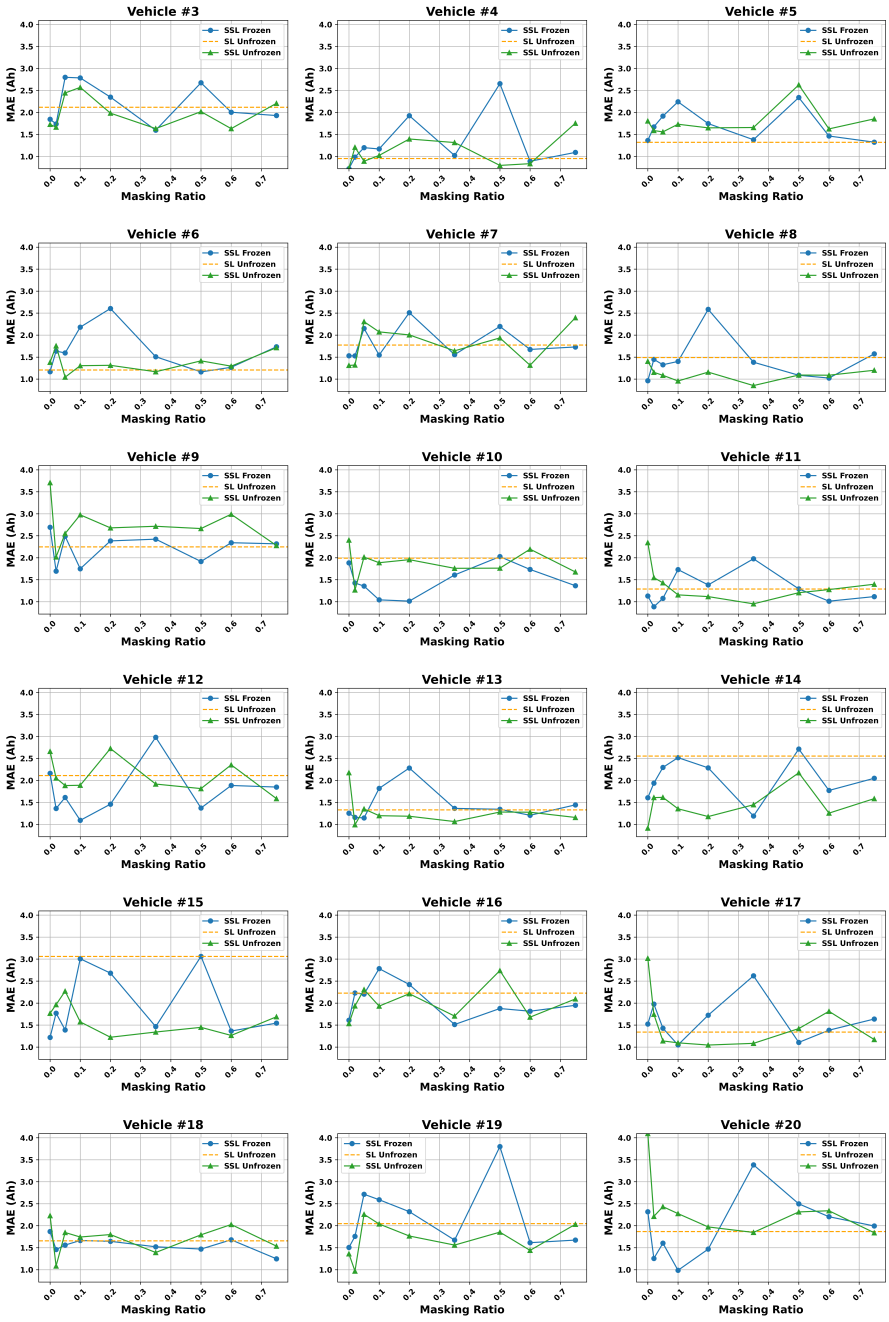

FIGURE 6.4: Per-vehicle MAE curves for FROZEN-SSL, UNFROZEN SSL, and the SL baseline across masking ratios. Each subplot corresponds to one of the 18 vehicles in the testing data set.

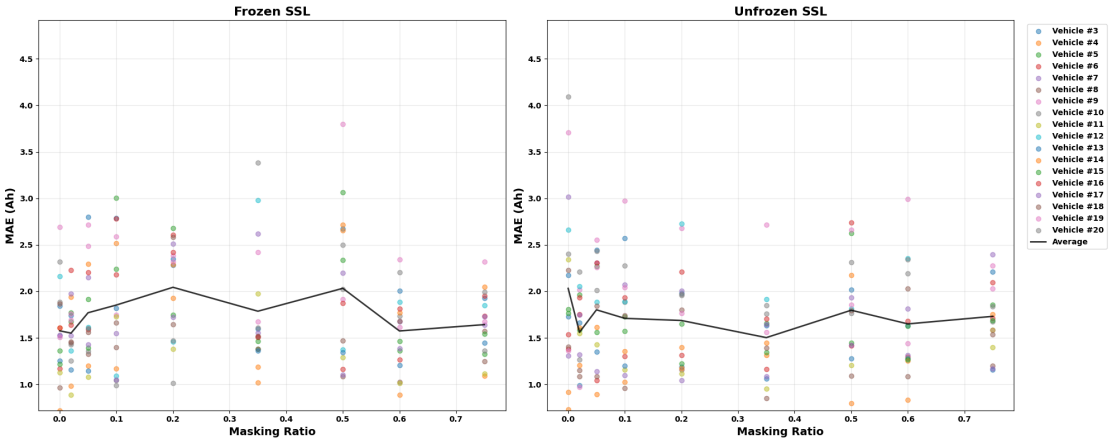

FIGURE 6.5: Per-vehicle scatter plots for Frozen (left) and Unfrozen (right) SSL models. Solid black lines trace the fleet average at each ratio.

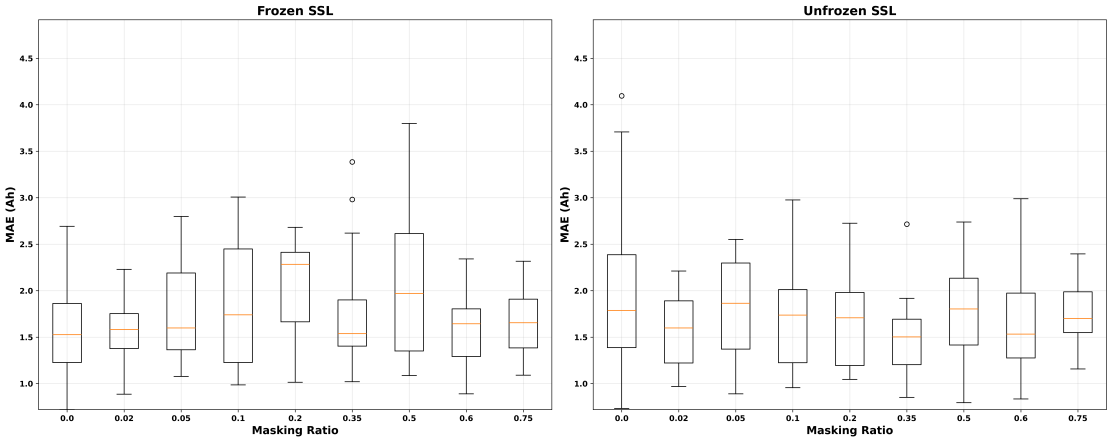

FIGURE 6.6: Distribution of MAE values across vehicles for each masking ratio: FROZEN-SSL (left) and UNFROZEN-SSL (right). Boxes span the interquartile range. Whiskers extend to ± 1.5 IQR. Circles denote outliers.

task, which translates into the lowest MAE and the most stable performance across vehicles.

For SoH estimation on this data set, applying a random masking ratio of **30−40**% during the pre-training phase and *unfreezing* the network during fine-tuning yields the best trade-off between accuracy and robustness. Under these settings, the SSL approach offers up to a 17 % mean improvement over a purely supervised pipeline.

6.3.3 Experiment 3: Effect of Federated Aggregation on SoH Estimation Accuracy

In this experiment, our aim is to compare the performance of our proposed FL approach, with a centralized approach. The **Centralized** model was trained on the union of all vehicle data, while our **Federated** model was trained locally on each vehicle and then aggregated following the standard FedAvg procedure. Both models were evaluated under three masking ratios (0, 0.35, and 0.60) that were applied during self-supervised pre-training. In these results, we use our results from our Frozen-SSL variant. Figure 6.7 reports the MAE per vehicle, while Figure 6.8 shows the fleet-level average.

In a fleet-wide perspective and across masking ratios, the average difference between the two training paradigms never exceeded 0.05 Ah (Table 6.6). At the most demanding ratio of 0.60, the gap shrank to only 0.02 Ah, confirming that the federated pipeline retains the accuracy of the centralized approach. In addition, both models follow the same trend: a small increase in error at a medium mask ratio (0.35) followed by a slight recovery at the higher ratio (0.60). The parallel trajectories indicate that the masking strategy, rather than the learning paradigm, drives the bulk of the performance change.

A closer look at Figure 6.7 reveals that for 14 of the 18 vehicles, the difference between centralized and federated MAE stays within 0.1 Ah for every masking ratio, suggesting that minor data heterogeneity does not translate into systematic error. Vehicles #12

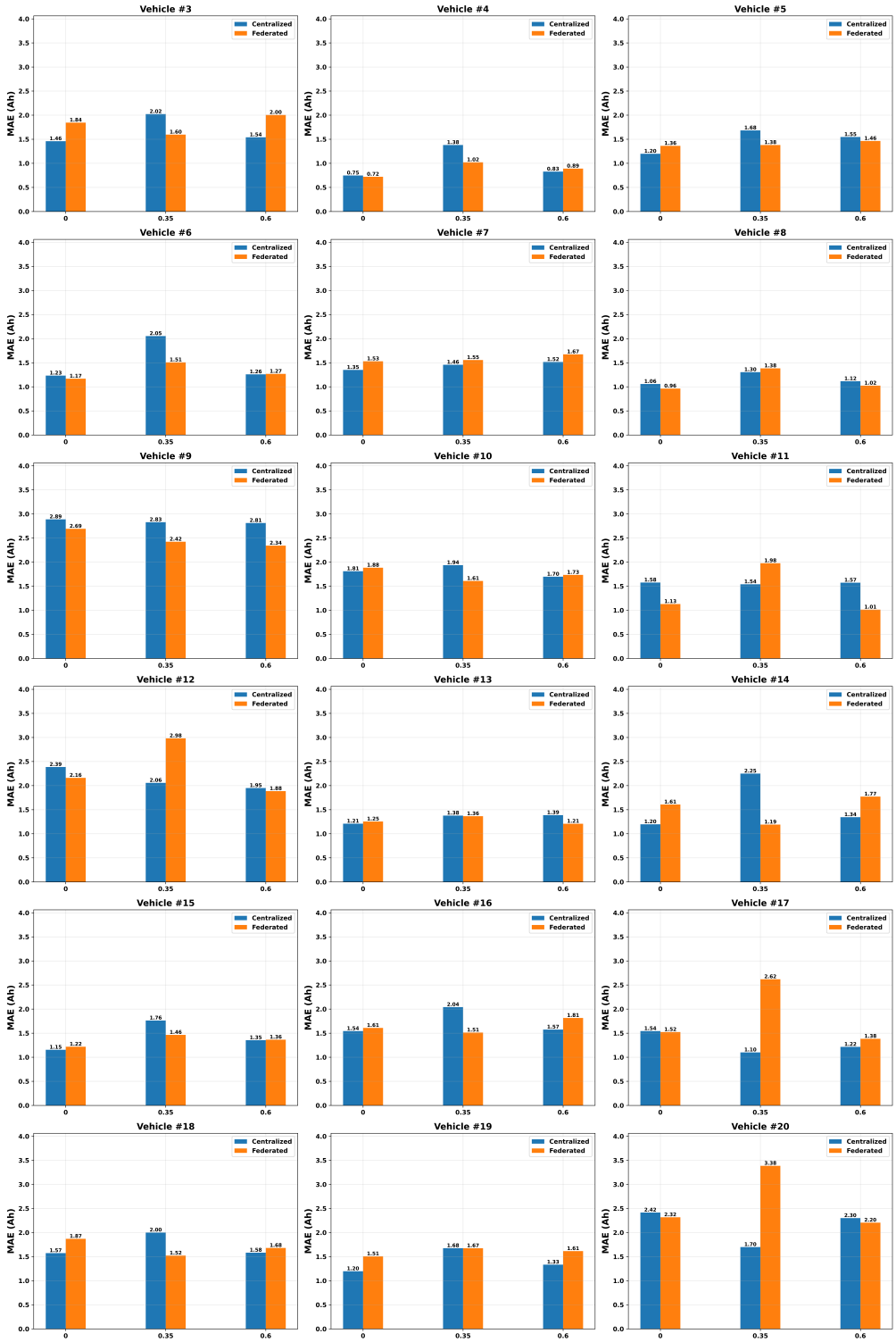

Figure 6.7: Per-vehicle MAE for centralized and federated models under three masking ratios.

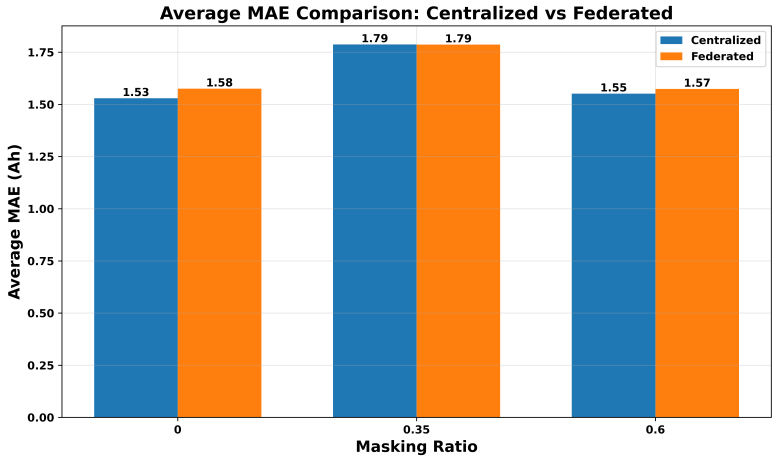

Figure 6.8: Fleet-level average MAE comparison between centralized and federated training.

Table 6.6: Fleet-level average MAE (Ah) for each masking ratio.

	Masking Ratio		
Training Mode	0	0.35	0.60
Centralized	1.53	1.79	1.55
Federated	1.58	1.79	1.57

and #17 show noticeably higher federated errors at the 0.35 ratio, while Vehicles #4 and #6 slightly favour the federated model at some ratios. However, these swings balance out in the fleet average, which remains virtually unchanged (Table 6.6). At a 0.60 mask, all vehicles exhibit near-identical performance for the two training modes. This supports the idea that, once representations become more powerful with heavy masking, the aggregation procedure introduces negligible additional noise.

6.4 Discussion

This section brings together the evidence from the three experiments and answers the research questions posed at the beginning of the chapter.

Both SSL variants lowered the mean MAE relative to the fully-supervised baseline, confirming that pre-training on unlabelled drive-cycle data does help the network extract transferable temporal features. In the first experiment, the Frozen-SSL model achieved the lowest fleet-wide *average* error and, more importantly, the smallest standard deviation, indicating strong regularization and consistent behaviour across vehicles. It showed an improvement in fleet-wide average error of 13.3%, when compared with the SL variant, with a maximum per-vehicle improvement of 35%. Unfrozen-SSL, on the other hand, delivered the single best result on seven of the eighteen test EVs, showing that letting the encoder adapt is useful when a battery departs from the population used in pre-training. While it showed a lower improvement in fleet-wide average error (8.8%), it raised the maximum per-vehicle improvement to a whopping 59%.

Masking ratios below 0.1 or above 0.5 degraded performance for both SSL variants. The sweet spot lay between **30**% and **40**%, where the unfrozen model cut the average error by $\approx$ 17% while also tightening the spread of per-vehicle losses. In addition to this variance shrinkage, we see that for all the masking ratios within this range, our unfrozen variant consistently performed better than the frozen one. With no masking

($r = 0$), the auto-encoder reduces to an identity mapping and learns little beyond the supervised objective, whereas too much masking ($r \geq 0.6$) removes so much context that reconstruction becomes guess-work. These observations are consistent with earlier reports on masked reconstruction for time-series data, where moderate masking forces the network to capture longer-range structure rather than local interpolation patterns.

Replacing a central data pool with FedAvg aggregation altered the fleet-level error by at most 0.05 Ah, around the typical laboratory repeatability limit for capacity measurements. Even at the most demanding masking ratio ($r = 0.6$), the federated model stayed within 0.02 Ah of its centralized counterpart, demonstrating that privacy-preserving training can be adopted without sacrificing accuracy.

As a practical takeaway: We recommend shipping the Unfrozen-SSL encoder, pre-trained with a masking ratio between 30-40%. As an alternative, which can be put to further experimentation in the future, we recommend shipping the Frozen-SSL encoder as the default firmware and enabling an *adaptive unfreeze* when the residual error exceeds a safety threshold for several consecutive cycles. Owing to its **15.6 k** parameters ($\approx$**62 kB**), our encoder can be stored and updated even on low-end Battery Management System (BMS) micro-controllers. The full inference stack used during fine-tuning still weighs in at only **0.88 M** parameters ($\approx$ **3.5 MB**), comfortably below the 4-16 MB flash budgets of modern automotive MCUs. A fixed masking ratio of 35% during pre-training offered the best trade-off between accuracy and robustness.

Wrapping up, self-supervised pre-training with a 35% masking ratio, delivered through an unfrozen encoder emerges as the most practical and accurate solution. The experimental evidence confirms that federated aggregation preserves this accuracy while honouring data privacy. Selective unfreezing appears as an attractive alternative.

Chapter 7

Conclusion and Future Work

7.1 Chapter Introduction

This final chapter brings the thesis full circle. It ties together the research questions posed in the beginning, breaks down the most significant findings of the chapters, and positions those findings within the broader landscape of Electric Vehicle (EV) battery management research. It then outlines the study's limitations, and aggregates actionable avenues for future investigation.

7.2 Restatement of Aims & Research Questions

The overarching aim of this thesis was to design, implement, and evaluate an *efficient*, *accurate*, and *privacy-preserving* framework for State-of-Health (SoH) estimation in EV batteries by combining masked-autoencoding Self-Supervised Learning (SSL), Federated Learning (FL), and cloud deployment.

This aim decomposed into four specific research questions stated originally in Chapter 1:

RQ1 Pre-training Benefit: Does pre-training with masked auto-encoding on unlabelled drive cycles enhance SoH prediction compared with purely supervised baselines under limited labelled data?

RQ2 Masking Sensitivity: How does the choice of *masking ratio* in SSL influence representation quality and downstream SoH accuracy?

RQ3 Fine-tuning Strategy: How do different fine-tuning strategies for the SSL encoder (fully frozen vs. full unfreeze) affect model performance and training stability?

RQ4 FL Impact: Does FL aggregation for model parameters degrade performance compared with centralized training?

7.3 Summary of Major Findings

The answers to **RQ1-RQ4** emerged from the experiments reported in Chapter 6. Rather than reiterating every numeric result, this section integrates the evidence thematically.

7.3.1 Self-Supervision vs. Supervision (RQ1)

Using a fixed masking ratio of 0.6, Table 6.4 showed that adding an SSL pre-training stage reduced the fleet-wide *average* Mean Absolute Error (MAE) from **1.81** Ah (Unfrozen-SL) to **1.57** Ah in the Frozen-SSL variant (a 13.3 % improvement) and to **1.65** Ah in the Unfrozen-SSL variant (8.8 % improvement). Frozen-SSL also exhibited the lowest dispersion (std. dev. 0.39 A h versus 0.52 A h for the fully supervised baseline). These gains validate the first hypothesis that unlabelled drive cycle data contain transferable temporal patterns useful for downstream SoH estimation when labelled data is scarce.

7.3.2 Effect of Masking Ratio (RQ2)

The masking ratio sweep (Table 6.5) produced the expected U-shaped curve. For the Unfrozen-SSL model the minimum MAE, **1.50** Ah, occurred at a masking ratio of **0.35**, a 17 % reduction relative to the 1.81 Ah baseline. Errors rose again toward both extremes, surpassing the baseline when the ratio reached 0.00 or 0.75. For Frozen-SSL, the best result (**1.57** Ah) was obtained at $r = 0.60$, not surpassing the unfrozen variant. Hence a moderate 30-40% masking ratio strikes the best balance between task difficulty and information availability.

7.3.3 Fine-tuning Strategies (RQ3)

Experiment 2 directly compared frozen and unfrozen SSL encoders under seven masking ratios. Throughout the practical range $0.10 \leq r \leq 0.50$ the Unfrozen-SSL variant always beat the frozen one, trimming the fleet-wide MAE by 6-17% (1.50 Ah vs. 1.79 Ah at the optimum $r = 0.35$, see Table 6.5). At the same time, it narrowed the error spread. The standard deviation fell from ≈0.42 Ah to ≈0.31 Ah and the number of outliers in Fig. 6.6 almost halved. Only at the extremes ($r = 0$ and $r \geq 0.60$), where the reconstruction task becomes trivial or impossible, did freezing recover a small advantage. Hence, once a non-trivial masking task is applied, allowing the encoder to adapt consistently improves both accuracy and robustness.

7.3.4 Federated vs. Centralized Training (RQ4)

Experiment 3 (Table 6.6) found that replacing a central data pool with FedAvg aggregation altered the fleet-wide MAE by at most **0.05** Ah (< 3 % of the baseline error). At the most demanding masking ratio $r = 0.60$ the gap shrank to **0.02** Ah (1 %). Per-vehicle curves (Fig. 6.7) showed that 14 of 18 vehicles stayed within 0.1 A h of the centralized reference across all ratios. Hence FL preserves accuracy while eliminating the need to export raw battery logs, fully answering RQ4 in the affirmative.

7.3.5 Integrated Perspective

Collectively, these findings demonstrate that (i) SSL pre-training lowers average error by up to 17 %, (ii) a 30-40 % masking ratio yields the lowest fleet-wide MAE, (iii) unfreezing further improves performance on atypical batteries (up to 59 % per vehicle), and (iv) federated aggregation adds only negligible error (≤ 0.05 A h). The proposed framework therefore addresses the challenges of *data scarcity*, *heterogeneity*, and *privacy* simultaneously, validating all four research hypotheses posed at the beginning of the thesis.

7.4 Theoretical & Practical Contributions

7.4.1 Academic Contributions

- **Quantified SSL Benefit:** By pre-training on unlabelled drive-cycle data, the proposed framework lowered the fleet-wide MAE from **1.81** Ah to **1.50** Ah (a 17 % gain) and achieved a per-vehicle improvement of up to 59 % (Chapter 6, Table 6.4). This work is, to the author's knowledge, the first to combine masked autoencoding with FL for battery SoH estimation.

- **Masking-Ratio Characterization:** A systematic sweep across seven masking ratios revealed a clear optimum at $r = 0.35$, where the unfrozen model reached **1.50** Ah (a 17 % reduction, see Table 6.5). The resulting U-shaped curve refines prior heuristic claims about "high masking" for time-series SSL.

- **Frozen vs. Unfrozen Fine-Tuning Analysis:** The study shows that unfreezing the encoder minimizes dispersion while delivering the best single-vehicle scores. Such a formal comparison has not previously appeared in the battery-health literature.

- **FL Validation:** FL (FedAvg) was shown to increase error by at most **0.05** Ah ($< 3\%$) relative to centralized training, answering long-standing concerns about aggregation noise (Table 6.6).

7.4.2 Practical Contributions

- **Deployment Blueprint:** Alongside the recommended unfrozen variant, results also support a two-tier firmware strategy: ship the Frozen-SSL model by default, but enable on-device *adaptive unfreezing* when residual error exceeds a safety threshold.

- **Memory & Compute Feasibility:** The 62 kB encoder fits comfortably inside the 256-512 kB flash budgets of current Battery Management System (BMS) microcontrollers. The 3.5 MB full stack is within the 4-16 MB range of next-generation automotive MCUs.

- **Hyper-parameter Guidelines:** A random masking ratio of 30-40% during pretraining and a history length of 2 months during fine-tuning yielded the best accuracy/variance trade-off. These concrete numbers can guide industrial calibration.

- **Open-Source Assets:** All code, model weights, and processing scripts will be released to accelerate reproducibility and adoption in commercial BMS design.

7.5 Limitations

Despite its contributions, the study has several caveats:

1. **Chemistry Coverage:** All experiments used an NMC data set drawn from 20 vehicles. Behaviour on LFP, NCA, or solid-state chemistries remains untested.

2. **Fleet Size & Diversity:** Only 18 vehicles were available for evaluation after fine-tuning. Larger, more diverse fleets could expose edge cases not captured here.

3. **Simulation-Only FL:** Federated rounds were emulated offline, so real-world effects such as packet loss, cellular latency, or straggler clients were not modelled.

4. **Single Aggregator:** FedAvg was the sole aggregation rule. Personalized FL, secure aggregation, and differential-privacy noise were left unexplored.

5. **Architectural Breadth:** The work focused on a Time Series Masked Autoencoder (Ti-MAE) encoder-decoder. Alternative SSL paradigms (contrastive, autoregressive) and hybrid masking schedules were not benchmarked.

6. **Physics & Uncertainty:** The models are purely data-driven and deterministic. Electrochemical constraints and predictive uncertainty are absent.

7.6 Recommendations for Future Work

Based on the current results, we cluster findings into four strategic research thrusts that address the limitations identified above and capitalize on the thesis's core contributions.

7.6.1 Thrust 1: Cross-Chemistry and Large-Fleet Validation

- **Multi-Chemistry Benchmarks:** Replicate the SSL + FL pipeline on LFP, NCA and emerging solid-state cells to verify whether the 17 % fleet-wide MAE reduction generalizes beyond NMC.

- **Vehicle Scale-Up:** Extend the evaluation from 18 to $\geq$100 vehicles, enabling confidence intervals narrow enough for certification and uncovering rare ageing trajectories.

7.6.2 Thrust 2: Real-World Federated Deployment

- **Advanced Aggregation:** Evaluate secure aggregation, differential-privacy noise and personalized FL layers to mitigate the single-aggregator limitation noted in the thesis [137], [156].

7.6.3 Thrust 3: Adaptive & Physics-Aware Learning

- **Dynamic Masking:** Replace the fixed $r = 0.35$-0.40 mask with a schedule that increases task difficulty over epochs.
- **Physics-Informed Constraints:** Embed electrochemical balance equations in the loss [157], [158].
- **Uncertainty Quantification:** Add Monte-Carlo dropout or lightweight ensembles so the BMS can flag low-confidence SoH estimates before warranty or safety limits are breached [159].

7.6.4 Thrust 4: Communication & Compute Optimization

- **Bandwidth Reduction:** Investigate gradient sparsification and 8-bit weight quantization to further shrink model parameter uploads, enabling over-the-air (OTA) updates over legacy 3G channels.
- **Edge/Fog Partitioning.** Explore splitting inference between an on-board microcontroller and a roadside edge server to keep the MCU flash budget below 2 MB without sacrificing accuracy.

7.7 Final Remarks

This thesis has demonstrated that coupling SSL with federated aggregation is a pragmatic path toward data-efficient, privacy-preserving battery diagnostics. While challenges remain, the evidence presented herein suggests that the next generation of BMS software can learn collaboratively across fleets and without compromising driver privacy to accelerate the global transition from Internal Combustion Engines (ICEs) to sustainable electric mobility.

Bibliography

[1] IEA. "Global ev outlook 2024." (2024), [Online]. Available: https://www.iea.org/reports/global-ev-outlook-2024.

[2] M. Ismail and R. Ahmed, A comprehensive review of cloud-based lithium-ion battery management systems for electric vehicle applications, *IEEE Access*, vol. 12, 116 259–116 273, 2024, ISSN: 2169-3536. [Online]. Available: http://dx.doi.org/10.1109/access.2024.3446880.

[3] K. He, X. Chen, S. Xie, Y. Li, P. Dollar, and R. Girshick, Masked autoencoders are scalable vision learners, in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2022. [Online]. Available: http://dx.doi.org/10.1109/CVPR52688.2022.01553.

[4] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, *Communication-efficient learning of deep networks from decentralized data*, 2023. [Online]. Available: https://arxiv.org/abs/1602.05629.

[5] T. Asrani, K. Mullachery, S. E, A. B, C. Zephrina, P. Kumar, and V. Kaniappan, Combustion engine vehicles electric vehicles: A comparative analysis, *International Research Journal of Modernization in Engineering Technology and Science*, 2024.

[6] F. Zhao, K. Chen, H. Hao, and Z. Liu, Challenges, potential and opportunities for internal combustion engines in china, *Sustainability*, vol. 12(12), 2020, ISSN: 2071-1050. [Online]. Available: https://www.mdpi.com/2071-1050/12/12/4955.

[7] B. Ayodele and S. Mustapa, Life cycle cost assessment of electric vehicles: A review and bibliometric analysis, *Sustainability*, vol. 12, 2387, 6 2020.

[8] R. Babayev and B. Johansson, Should we walk or take a car for minimum greenhouse gas emissions? *SAE Technical Paper*, 2019.

[9] A. Kurkin, E. Kryukov, O. Masleeva, Y. Petukhov, and D. Gusev, Comparative life cycle assessment of electric and internal combustion engine vehicles, *Energies*, vol. 17(11), 2747, Jun. 2024, ISSN: 1996-1073. [Online]. Available: http://dx.doi.org/10.3390/en17112747.

[10] R. McKitrick, Economic implications of a phased-in ev mandate in canada, *Canadian Journal of Economics/Revue Canadienne D Économique*, vol. 57, 1434–1458, 4 2024.

[11] A. J. Mbusi, Reimagining electric vehicles for a sustainable future, *Preprints*, Jun. 2024. [Online]. Available: https://doi.org/10.20944/preprints202406.1200.v1.

[12] S. Ma, A. H. Sharaai, N. K. Matthew, and N. S. Zainordin, Sustainable development of electric vehicles in china: A bibliometric analysis and systematic review, 2011 onward, *Edelweiss Applied Science and Technology*, vol. 8, 2252–2260, 6 2024.

[13] B. Kailashkumar. and M. Elanchezhiyan., Powering ahead: Current landscape of electric vehicles (ev) in india, *International Journal of Environment and Climate Change*, vol. 14, 639–647, 1 2024.

[14] M. Muratori, M. Alexander, D. J. Arent, M. Bazilian, P. Cazzola, E. M. Dede, J. T. Farrell, C. Gearhart, D. L. Greene, A. Jenn, M. Keyser, T. Lipman, S.

Narumanchi, A. Pesaran, R. Sioshansi, E. Suomalainen, G. Tal, K. Walkowicz, and J. Ward, The rise of electric vehicles—2020 status and future expectations, *Progress in Energy*, vol. 3, 022 002, 2 2021.

[15] J. V. Mierlo and G. Maggetto, Fuel cell or battery: Electric cars are the future, *Fuel Cells*, 2007.

[16] X. He, S. Zhang, Y. Wu, T. J. Wallington, X. Lu, M. A. Tamor, M. B. McElroy, K. M. Zhang, C. Nielsen, and J. Hao, Economic and climate benefits of electric vehicles in china, the united states, and germany, *Environmental Science & Technology*, 2019.

[17] Y. Tang, T. Cockerill, A. Pimm, and X. Yuan, Reducing the life cycle environmental impact of electric vehicles through emissions-responsive charging, *Iscience*, 2021.

[18] A. Perujo, C. Thiel, and F. Nemry, Electric vehicles in an urban context: Environmental benefits and techno-economic barriers, 2011.

[19] S. P. Holland, E. T. Mansur, N. Z. Muller, and A. J. Yates, Are there environmental benefits from driving electric vehicles? the importance of local factors, *American Economic Review*, 2016.

[20] E. O. Hancco Catata, P. J. dos Neto, T. A. S. Barros, and E. R. Filho, Modeling, control and regenerative brakingof bldc machines in electric bycicles, 2020.

[21] V. Muerza, E. Larrodé, and J. M. Moreno-Jiménez, Designing sustainable systems for urban freight distribution through techniques of multicriteria decision analysis, 2016.

[22] J. Q. Xu, Y. Yang, and R. S. Yin, An economic comparison between electric and conventional vehicles based on system dynamics, *Advanced Materials Research*, 2013.

[23] A. Ayob, W. M. Faizal Mahmood, A. Mohamed, M. Z. Che Wanik, M. M. Siam, S. Sulaiman, A. Azit, and M. A. Mohamed Ali, Review on electric vehicle, battery charger, charging station and standards, *Research Journal of Applied Sciences Engineering and Technology*, 2014.

[24] I. Malmgren, Quantifying the societal benefits of electric vehicles, *World Electric Vehicle Journal*, 2016.

[25] L. W. Davis and J. M. Sallee, Should electric vehicle drivers pay a mileage tax? *Environmental and Energy Policy and the Economy*, 2020.

[26] A. Tchagang and Y. Yoo, V2b/v2g on energy cost and battery degradation under different driving scenarios, peak shaving, and frequency regulations, *World Electric Vehicle Journal*, 2020.

[27] M. A. Ortega-Vazquez, Optimal scheduling of electric vehicle charging and vehicle-to-grid services at household level including battery degradation and price uncertainty, *Iet Generation Transmission & Distribution*, 2014.

[28] P. P. Malya, L. Fiorini, M. Rouhani, and M. Aiello, Electric vehicles as distribution grid batteries: A reality check, *Energy Informatics*, 2021.

[29] T. Pesch, H.-J. A. Dirk Müller, and D. Witthaut, Economic viability and infrastructure requirements for the electrification of highway traffic, 2020.

[30] I. O. Olayode, E. Jamei, and F. J. Alex, Towards the sustainability of electric vehicles in africa: An overview, 2024.

[31] A. Wiratmoko, H. Apriyanto, T. D. Tamtomo, Alkadri, Suripto, B. D. Soewargono, Y. I. Dwiananto, and A. Suhendra, Policy analysis of electric vehicle infrastructure supporting sustainable energy use, *Iop Conference Series Earth and Environmental Science*, 2023.

[32] B. Xu, A. W. Davis, and G. Tal, Estimating the total number of workplace and public electric vehicle chargers in california, *Transportation Research Record Journal of the Transportation Research Board*, 2021.

[33] P. Morrissey, P. Weldon, and M. O'Mahony, Informing the strategic rollout of fast electric vehicle charging networks with user charging behavior data analysis, *Transportation Research Record Journal of the Transportation Research Board*, 2016.

[34] A. Benmouna, L. Calmelet, and M. Becherif, Charging stations for large-scale deployment of electric vehicles, *Batteries*, 2024.

[35] M. S. Hadi, D. K. Argeshwara, S. Sendari, M. A. Mizar, E. H. Sanjaya, and M. Irvan, Off-grid electric vehicle charging station with integrated local server ocpp protocol as a management system, *Transport and Telecommunication Journal*, 2024.

[36] M. Mutavđžija, M. Kovačić, and K. Buntak, Assessment of selected factors influencing the purchase of electric vehicles—a case study of the republic of croatia, *Energies*, 2022.

[37] C. A. Soares Machado, H. Takiya, C. L. Kenji Yamamura, J. A. Quintanilha, and F. T. Berssaneti, Placement of infrastructure for urban electromobility: A sustainable approach, *Sustainability*, 2020.

[38] M. Waseem, J. Huang, C.-N. Wong, and C. Lee, Data-driven gwo-brnn-based soh estimation of lithium-ion batteries in evs for their prognostics and health management, *Mathematics*, 2023.

[39] M. R. Ranga, V. R. Aduru, N. V. Krishna, K. D. Rao, S. Dawn, F. Alsaif, S. Alsulamy, and T. S. Ustun, An unscented kalman filter-based robust state of health prediction technique for lithium ion batteries, *Batteries*, 2023.

[40] W. Choi, A study on state of charge and state of health estimation in consideration of lithium-ion battery aging, *Sustainability*, 2020.

[41] J. Wang, C. Zhang, X. Meng, L. Zhang, X. Li, and W. Zhang, A novel feature engineering-based soh estimation method for lithium-ion battery with downgraded laboratory data, *Batteries*, 2024.

[42] S. Jafari and Y.-C. Byun, Xgboost-based remaining useful life estimation model with extended kalman particle filter for lithium-ion batteries, *Sensors*, 2022.

[43] Y. Li, K. Liu, A. Foley, A. Zülke, M. Berecibar, E. Nanini-Maury, J. V. Mierlo, and H. E. Hoster, Data-driven health estimation and lifetime prediction of lithium-ion batteries: A review, *Renewable and Sustainable Energy Reviews*, 2019.

[44] R. Li, P. Liu, K. Li, and X. Zhang, Adaboost.rt-lstm based joint soc and soh estimation method for retired batteries, *Batteries*, 2023.

[45] B. Zou, H. Wang, T. Zhang, M. Xiong, C. M. Xiong, Q. Sun, W. Wang, L. Zhang, C. Zhang, and H. Ruan, A deep learning approach for state-of-health estimation of lithium-ion batteries based on differential thermal voltammetry and attention mechanism, *Frontiers in Energy Research*, 2023.

[46] Z. Chen, Q. Xue, Y. Wu, S. Shen, Y. Zhang, and J. Shen, Capacity prediction and validation of lithium-ion batteries based on long short-term memory recurrent neural network, *IEEE Access*, vol. 8, 172 783–172 798, 2020.

[47] M. Massaoudi, H. Abu-Rub, and A. Ghrayeb, Advancing lithium-ion battery health prognostics with deep learning: A review and case study, *Ieee Open Journal of Industry Applications*, 2024.

[48] P. M. Mell and T. Grance, *The NIST definition of cloud computing*. 2011. [Online]. Available: http://dx.doi.org/10.6028/NIST.SP.800-145.

[49] S. J. Park, Y. J. Lee, and W. H. Park, Configuration method of aws security architecture that is applicable to the cloud lifecycle for sustainable social network, *Security and Communication Networks*, vol. 2022, 2022, ISSN: 19390122. [Online]. Available: https://doi.org/10.1155/2022/3686423.

[50] S. Shin, Y. Kim, and S. Lee, Deadline-guaranteed scheduling algorithm with improved resource utilization for cloud computing, *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*, 814–819, Jul. 2015. [Online]. Available: https://doi.org/10.1109/ccnc.2015.7158082.

[51] M. K. V. Nandimandalam and E. Choi, Comparison of open-source paas architectural components, *Computer Science &; Information Technology (CS &; IT)*, 53–62, Jan. 2016. [Online]. Available: https://doi.org/10.5121/csit.2016.60206.

[52] Neeraj, M. S. Goraya, and D. Singh, A ranking based model for selecting optimum cloud geographical region, *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, 793–797, 10 Aug. 2019, ISSN: 22783075. [Online]. Available: https://doi.org/10.35940/ijitee.j8908.0881019.

[53] N. K. Rajpoot, P. Singh, and B. Pant, Implementation of delay-sensitive smart healthcare framework in cloud and fog environment, [Online]. Available: https://doi.org/10.21203/rs.3.rs-2975684/v1.

[54] M. Talebkhah, A. Sali, M. Marjani, M. Gordan, S. J. Hashim, and F. Z. Rokhani, Iot and big data applications in smart cities: Recent advances, challenges, and critical issues, *IEEE Access*, vol. 9, 55 465–55 484, Jan. 2021, ISSN: 21693536. [Online]. Available: https://doi.org/10.1109/access.2021.3070905.

[55] K. Kreslins, D. V. Novik, and T. Vasiljeva, Challenge of cloud computing for smes: A case of baltic countries, *Journal of Innovation Management in Small*

and Medium Enterprises, vol. 2018, 1–10, Jun. 2018, ISSN: 2166076X. [Online]. Available: https://doi.org/10.5171/2018.238581.

[56] M. S. E. M. ElSayed, Trends towards iot in egypt, *Tehnički Glasnik*, vol. 14, 381–387, 3 Sep. 2020, ISSN: 18466168. [Online]. Available: https://doi.org/10.31803/tg-20200714162443.

[57] S. F. da Costa Bezerra, A. S. Filho, F. C. Delicato, and A. R. da Rocha, Processing complex events in fog-based internet of things systems for smart agriculture, *Sensors*, vol. 21, 21 Nov. 2021, ISSN: 14248220. [Online]. Available: https://doi.org/10.3390/s21217226.

[58] A. Waheed, M. A. Shah, S. M. Mohsin, A. Khan, C. Maple, S. Aslam, and S. Shamshirband, A comprehensive review of computing paradigms, enabling computation offloading and task execution in vehicular networks, *IEEE Access*, vol. 10, 3580–3600, 2022, ISSN: 21693536. [Online]. Available: https://doi.org/10.1109/access.2021.3138219.

[59] G. Dhawan, Conceptualization of cloud computing and its security threats, challenges, technologies and application, *International Journal for Research in Applied Science and Engineering Technology*, vol. V, 629–635, IV Apr. 2017. [Online]. Available: https://doi.org/10.22214/ijraset.2017.4112.

[60] T. Francis, A comparison of cloud execution mechanisms fog, edge, and clone cloud computing, *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 8, 4646–4653, 6 Dec. 2018, ISSN: 20888708. [Online]. Available: https://doi.org/10.11591/ijece.v8i6.pp4646-4653.

[61] B. Aimer, M. Roy, and H. Park, Evaluating serverless architecture for big data enterprise applications, 2021.

[62] A. Bose and S. Nag, Green computing – a survey of the current technologies, *Asia-Pacific Journal of Management and Technology*, vol. 03, 01–15, 02 2022. [Online]. Available: https://doi.org/10.46977/apjmt.2022.v03i02.001.

[63] M. B. Yassein, I. Hmeidi, O. Alomari, W. Mardini, O. AlZoubi, and D. Krstic, Blockchain technology in cloud computing: Challenges and open issues, *Digital Transformation Technology*, vol. 224, 81–98, 2022, ISSN: 23673389. [Online]. Available: https://doi.org/10.1007/978-981-16-2275-5%5C_5.

[64] R. T. A. Haroun and A. AlKabour, Survey on cloud computing techniques for data integrity checking, *International Journal for Innovation Education and Research*, vol. 7, 130–141, 6 Jun. 2019, ISSN: 2411-3123. [Online]. Available: https://doi.org/10.31686/ijier.vol7.iss6.1560.

[65] P. Vishwakarma, Analysis of cloud storage information security and it s various methods, *International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 3, 808–811, 2 2015. [Online]. Available: https://doi.org/10.17762/ijritcc2321-8169.150281.

[66] F. Qaisar, H. Shahab, M. Iqbal, H. M. Sargana, M. Aqeel, and M. A. Qayyum, Recent trends in cloud computing and iot platforms for it management and development: A review, *Pakistan Journal of Engineering and Technology*, vol. 6, 98–105, 1 Mar. 2023, ISSN: 2664-2042. [Online]. Available: https://doi.org/10.51846/vol6iss1pp98-105.

[67] I. Ivkic, S. Wolfauer, T. Oberhofer, and M. G. Tauber, On the cost of cyber security in smart business, *2017 12th International Conference for Internet Technology and Secured Transactions (ICITST)*, 255–260, May 2018. [Online]. Available: https://doi.org/10.23919/icitst.2017.8356395.

[68] G. Cordasco, M. D'Auria, A. Negro, V. Scarano, and C. Spagnuolo, Toward a domain-specific language for scientific workflow-based applications on multicloud

system, *Concurrency and Computation: Practice and Experience*, vol. 33, 18 Sep. 2021, ISSN: 15320634. [Online]. Available: https://doi.org/10.1002/cpe.5802.

[69] H. N. Abed, G. S. Mahmood, and N. H. Hassoon, A secure and efficient data distribution system in a multi-cloud environment, *Malaysian Journal of Science and Advanced Technology*, 109–117, Sep. 2021. [Online]. Available: https://doi.org/10.56532/mjsat.v1i3.21.

[70] S. J. Taylor, A. Anagnostou, T. Kiss, G. Terstyanszky, P. Kacsuk, N. Fantini, D. Lakehal, and J. Costes, Enabling cloud-based computational fluid dynamics with a platform-as-a-service solution, *IEEE Transactions on Industrial Informatics*, vol. 15, 85–94, 1 2018, ISSN: 15513203. [Online]. Available: https://doi.org/10.1109/tii.2018.2849558.

[71] S. V. Nithya and V. R. Uthariaraj, Identity-based provable data possession for multicloud storage with parallel key-insulation, *KSII Transactions on Internet and Information Systems*, vol. 15, 3322–3347, 9 Sep. 2021, ISSN: 22881468. [Online]. Available: https://doi.org/10.3837/tiis.2021.09.013.

[72] N. M. Abdulkareem, S. R. Zeebaree, M. A. Sadeeq, D. M. Ahmed, A. S. Sami, and R. R. Zebari, Iot and cloud computing issues, challenges and opportunities: A review, *Qubahan Academic Journal*, vol. 1, 1–7, 2 Mar. 2021, ISSN: 27098206. [Online]. Available: https://doi.org/10.48161/qaj.v1n2a36.

[73] N. S. Lohitha and P. M. Kumar, An efficient framework for processing big data in internet of things enabled cloud environments, *International Journal of Communication Systems*, vol. 35, 10 Jul. 2022, ISSN: 10991131. [Online]. Available: https://doi.org/10.1002/dac.5146.

[74] J. Sendorek, T. Szydlo, and R. Brzoza-Woch, Software-defined virtual testbed for iot systems, *Wireless Communications and Mobile Computing*, vol. 2018, 2018, ISSN: 15308677. [Online]. Available: https://doi.org/10.1155/2018/1068261.

[75] H. F. Atlam, R. J. Walters, and G. B. Wills, Fog computing and the internet of things: A review, *Big Data and Cognitive Computing*, vol. 2, 1–18, 2 Jun. 2018, ISSN: 25042289. [Online]. Available: https://doi.org/10.3390/bdcc2020010.

[76] M. Elkholy and M. A. Marzok, Light weight serverless computing at fog nodes for internet of things systems, *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 26, 394–403, 1 Apr. 2022, ISSN: 25024760. [Online]. Available: https://doi.org/10.11591/ijeecs.v26.i1.pp394-403.

[77] H. Bangui, S. Rakrak, S. Raghay, and B. Buhnova, Moving to the edge-cloud-of-things: Recent advances and future research directions, *Electronics*, vol. 7, 11 2018, ISSN: 20799292. [Online]. Available: https://doi.org/10.3390/electronics7110309.

[78] S. Yang, R. He, Z. Zhang, Y. Cao, X. Gao, and X. Liu, Chain: Cyber hierarchy and interactional network enabling digital solution for battery full-lifespan management, *Matter*, vol. 3, 27–41, 1 2020.

[79] S. Yang, Z. Zhang, R. Cao, M. Wang, H. Cheng, L. Zhang, Y. Jiang, Y. Li, B. Chen, H. Ling, Y. Lian, B. Wu, and X. Liu, Implementation for a cloud battery management system based on the chain framework, *Energy and AI*, vol. 5, 100 088, Sep. 2021, ISSN: 2666-5468.

[80] W. Wang, K. Yang, L. Zhang, S. Zhou, B. Ren, Y. Lu, R. Tan, T. Zhu, B. Ma, S. Yang, and X. Liu, An end-cloud collaboration approach for online state-of-health estimation of lithium-ion batteries based on multi-feature and transformer, *Journal of Power Sources*, vol. 608, 234 669, 2024, ISSN: 0378-7753. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378775324006219.

[81] P. Jiang, H. Wang, G. Huang, W. Feng, M. Xiong, J. Zhao, W. Hua, Y. Zhang, W. Wang, and T. Zhu, An end-cloud collaboration for state-of-charge estimation

of lithium-ion batteries based on extended Kalman filter and convolutional neural network (CNN)—long short-term memory (LSTM)—attention mechanism (AM), *Journal of Renewable and Sustainable Energy*, vol. 16(2), 024 103, Apr. 2024, ISSN: 1941-7012. [Online]. Available: https://doi.org/10.1063/5.0198089.

[82] H. Tang, Y. Wu, Y. Cai, F. Wang, Z. Lin, and Y. Pei, Design of power lithium battery management system based on digital twin, *Journal of Energy Storage*, vol. 47, 103 679, Mar. 2022, ISSN: 2352-152X.

[83] W. Li, M. Rentemeister, J. Badeda, D. Jöst, D. Schulte, and D. U. Sauer, Digital twin for battery systems: Cloud battery management system with online state-of-charge and state-of-health estimation, *Journal of Energy Storage*, vol. 30, 101 557, Aug. 2020, ISSN: 2352-152X.

[84] Y. Wang, R. Xu, C. Zhou, X. Kang, and Z. Chen, Digital twin and cloud-side-end collaboration for intelligent battery management system, *Journal of Manufacturing Systems*, vol. 62, 124–134, Jan. 2022, ISSN: 0278-6125.

[85] J. Wu, X. Liu, J. Meng, and M. Lin, Cloud-to-edge based state of health estimation method for lithium-ion battery in distributed energy storage system, *Journal of Energy Storage*, vol. 41, 102 974, Sep. 2021, ISSN: 2352-152X.

[86] K. Alamin, Y. Chen, E. MacIi, M. Poncino, and S. Vinco, A machine learning-based digital twin for electric vehicle battery modeling, 2022, ISBN: 9781665483568.

[87] W. Guo, L. Yang, and Z. Deng, A cloud-based state of health estimation method for lithium-ion batteries using sparse charging data, vol. 2456, 2023.

[88] E. García, E. Quiles, and A. Correcher, Distributed intelligent battery management system using a real-world cloud computing system, *Sensors*, vol. 23, 7 2023.

[89] Q. Chen, Y. He, G. Yu, C. Xu, M. Liu, and Z. Li, Kbmp: Kubernetes-orchestrated iot online battery monitoring platform, *IEEE Internet of Things Journal*, vol. 11(14), 25 358–25 370, 2024.

[90] *Battery in the cloud*. [Online]. Available: https://www.bosch-mobility.com/en/solutions/software-and-services/battery-in-the-cloud/battery-in-the-cloud/.

[91] *Cloud platform - powerful battery life & safety insights*. [Online]. Available: https://elysia.co/cloud.

[92] *Battery analytics platform | twaice*. [Online]. Available: https://www.twaice.com/products/battery-analytics-platform?null=.

[93] *Battery management software | zitara live*. [Online]. Available: https://www.zitara.com/zitara-live.

[94] *Isl - eatron technologies*. [Online]. Available: https://eatron.com/intelligent-software-layer/.

[95] *About us - newtwen*. [Online]. Available: https://www.newtwen.com/about-us/.

[96] *Cloud iot battery management system in electric vehicles*. [Online]. Available: https://bacancysystems.com/blog/iot-based-battery-management-system-in-electric-vehicles.

[97] *Eve-ai fleet analytics | electra vehicles*. [Online]. Available: https://www.electravehicles.com/eve-ai-fleet-analytics.

[98] S. C. H. Hoi, J. Lü, and P. Zhao, Online learning: A comprehensive survey, [Online]. Available: https://doi.org/10.48550/arxiv.1802.02871.

[99] R. Shipman, R. Roberts, J. Waldron, C. Rimmer, L. Rodrigues, and M. Gillott, Online machine learning of available capacity for vehicle-to-grid services during the coronavirus pandemic, *Energies*, vol. 14, 21 Nov. 2021, ISSN: 19961073. [Online]. Available: https://doi.org/10.3390/en14217176.

[100] X. Xu, N. Zhang, Y. Yan, L. Qin, and F. Qian, Smooth iteration online support tension machine algorithm and application in fault diagnosis of electric vehicle

extended range, *Advances in Mechanical Engineering*, vol. 10, 12 Dec. 2018, ISSN: 16878140. [Online]. Available: https://doi.org/10.1177/1687814018816563.

[101] J. Alwidian, T. Khasawneh, M. Alsahlee, and A. Safia, An online machine learning approach to sentiment analysis in social media, *Modern Applied Science*, vol. 16, 29, 4 Oct. 2022, ISSN: 1913-1844. [Online]. Available: https://doi.org/10.5539/mas.v16n4p29.

[102] *Online-ml/river: Online machine learning in python*. [Online]. Available: https://github.com/online-ml/river.

[103] *Online-ml/beaver: Mlops for (online) machine learning*. [Online]. Available: https://github.com/online-ml/beaver.

[104] T. Kim, D. Makwana, A. Adhikaree, J. Vagdoda, and Y. Lee, Cloud-based battery condition monitoring and fault diagnosis platform for large-scale lithium-ion battery energy storage systems, *Energies*, vol. 11, 1 2018.

[105] *Alvarium – lf edge: Building an open source framework for the edge*. [Online]. Available: https://lfedge.org/projects/alvarium/.

[106] Z. Nezami, E. Chaniotakis, and E. Pournaras, When 'computing follows vehicles': Decentralized mobility-aware resource allocation in the edge-to-cloud continuum, Apr. 2024. [Online]. Available: https://arxiv.org/abs/2404.13179v1.

[107] R. Wagner, C. F. Lopez, and C. Stiller, Self-supervised pseudo-colorizing of masked cells, *Plos One*, vol. 18, e0290561, 8 2023.

[108] M. Alloulah, A. D. Singh, and M. Arnold, Self-supervised radio-visual representation learning for 6g sensing, 2021.

[109] C. Ren, L. Sun, and D. Peng, A contrastive predictive coding-based classification framework for healthcare sensor data, *Journal of Healthcare Engineering*, vol. 2022, 1–8, 2022.

[110] H. Banville, G. Moffat, I. Albuquerque, D. A. Engemann, A. Hyvärinen, and A. Gramfort, Self-supervised representation learning from electroencephalography signals, *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*, 1–6, 2019.

[111] A. A. Deshmukh, A. Kumar, L. Boyles, D. Charles, E. Manavoglu, and Ü. Doğan, Self-supervised contextual bandits in computer vision, 2020.

[112] S. Gidaris, A. Bursuc, N. Komodakis, P. P. Perez, and M. Cord, Boosting few-shot visual learning with self-supervision, *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 8058–8067, 2019.

[113] V. C. R. Magalhães, M. F. P. Costa, M. Ferreira, T. Pinto, and V. L. F. d. A. Figueiredo, Application of a self-supervised learning technique for monitoring industrial spaces, *Lecture Notes in Computer Science*, 407–420, 2023.

[114] A. Radford and K. Narasimhan, Improving language understanding by generative pre-training, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:49313245.

[115] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, 2018. [Online]. Available: https://arxiv.org/abs/1810.04805.

[116] Z. Xie, Z. Zhang, Y. Cao, Y. Lin, J. Bao, Z. Yao, Q. Dai, and H. Hu, *Simmim: A simple framework for masked image modeling*, 2022. [Online]. Available: https://arxiv.org/abs/2111.09886.

[117] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, *A simple framework for contrastive learning of visual representations*, 2020. [Online]. Available: https://arxiv.org/abs/2002.05709.

[118] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, *Momentum contrast for unsupervised visual representation learning*, 2020. [Online]. Available: https://arxiv.org/abs/1911.05722.

[119] Z. Wang, Z. C. Lipton, and Y. Tsvetkov, On negative interference in multilingual models: Findings and a meta-learning treatment, in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, B. Webber, T. Cohn, Y. He, and Y. Liu, Eds., Online: Association for Computational Linguistics, Nov. 2020, 4438–4450. [Online]. Available: https://aclanthology.org/2020.emnlp-main.359/.

[120] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, *Lora: Low-rank adaptation of large language models*, 2021. [Online]. Available: https://arxiv.org/abs/2106.09685.

[121] E. B. Zaken, S. Ravfogel, and Y. Goldberg, *Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models*, 2022. [Online]. Available: https://arxiv.org/abs/2106.10199.

[122] Q. Wang, M. Ye, S. Celik, Z. Deng, B. Li, D. Uwe Sauer, and W. Li, Unlocking the potential of unlabeled data: Self-supervised machine learning for battery aging diagnosis with real-world field data, *Journal of Energy Chemistry*, vol. 99, 681–691, Dec. 2024, ISSN: 2095-4956. [Online]. Available: http://dx.doi.org/10.1016/j.jechem.2024.08.037.

[123] Y. Shen, Y. Wang, D. Guo, J. Chen, X. Liu, X. Han, Y. Zheng, and M. Ouyang, Enhanced few-shot state-of-health estimation for lithium-ion batteries via masked autoencoder, 2025. [Online]. Available: http://dx.doi.org/10.2139/ssrn.5153802.

[124] J. C. Chen, *Degradation self-supervised learning for lithium-ion battery health diagnostics*, 2025. [Online]. Available: https://arxiv.org/abs/2503.08083.

[125] J. Sadler, R. Mohammed, M. Castle, and K. Uddin, *Accept: Diagnostic forecasting of battery degradation through contrastive learning*, 2025. [Online]. Available: https://arxiv.org/abs/2501.10492.

[126] L. U. Khan, W. Saad, Z. Han, E. Hossain, and C. S. Hong, Federated learning for internet of things: Recent advances, taxonomy, and open challenges, 2020.

[127] M. Hosseinzadeh, A. Hemmati, and A. M. Rahmani, Federated learning-based iot: A systematic literature review, *International Journal of Communication Systems*, vol. 35, 11 2022.

[128] Y. Y. Ghadi, T. Mazhar, S. F. A. Shah, I. Haq, W. Ahmad, K. Ouahada, and H. Hamam, Integration of federated learning with iot for smart cities applications, challenges, and solutions, *PeerJ Computer Science*, vol. 9, e1657, 2023.

[129] S. Ahmed, T. Mahesh, E. Srividhya, V. Kumar, S. Bhatia, A. Albuali, and A. Almusharraf, Towards blockchain based federated learning in categorizing healthcare monitoring devices on artificial intelligence of medical things investigative framework, *BMC Medical Imaging*, vol. 24, 1 2024.

[130] M. A. Ferrag, O. Friha, Λ. M , H. Janicke, and L. Shu, Federated deep learning for cyber security in the internet of things: Concepts, applications, and experimental analysis, *IEEE Access*, vol. 9, 138 509–138 542, 2021.

[131] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, *Federated optimization in heterogeneous networks*, 2020. [Online]. Available: https://arxiv.org/abs/1812.06127.

[132] S. P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh, *Scaffold: Stochastic controlled averaging for federated learning*, 2021. [Online]. Available: https://arxiv.org/abs/1910.06378.

[133] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, *Feddane: A federated newton-type method*, 2020. [Online]. Available: https://arxiv.org/abs/2001.01920.

[134] X. Li, M. Jiang, X. Zhang, M. Kamp, and Q. Dou, *Fedbn: Federated learning on non-iid features via local batch normalization*, 2021. [Online]. Available: https://arxiv.org/abs/2102.07623.

[135] R. Pathak and M. J. Wainwright, *Fedsplit: An algorithmic framework for fast federated optimization*, 2020. [Online]. Available: https://arxiv.org/abs/2005.05238.

[136] X. Chen, Z. Chen, M. Zhang, Z. Wang, M. Liu, M. Fu, and P. Wang, A remaining useful life estimation method based on long short-term memory and federated learning for electric vehicles in smart cities, *PeerJ Computer Science*, vol. 9, e1652, Oct. 2023, ISSN: 2376-5992. [Online]. Available: http://dx.doi.org/10.7717/peerj-cs.1652.

[137] T. Wang, Z. Y. Dong, and H. Xiong, Adaptive multipersonalized federated learning for state of health estimation of multiple batteries, *IEEE Internet of Things Journal*, vol. 11(24), 39 994–40 008, 2024.

[138] R. Lai, J. Wang, Y. Tian, and J. Tian, Fedcbe: A federated-learning-based collaborative battery estimation system with non-iid data, *Applied Energy*, vol. 368, 123 534, Aug. 2024, ISSN: 0306-2619. [Online]. Available: http://dx.doi.org/10.1016/j.apenergy.2024.123534.

[139] M. Yılmaz, E. Çinar, and A. Yazıcı, Federated learning-based state of charge estimation in electric vehicles using federated adaptive client momentum, *IEEE Access*, vol. 13, 72 128–72 141, 2025. [Online]. Available: https://doi.org/10.1109/access.2025.3563188.

[140] Z. Deng, L. Xu, H. Liu, X. Hu, Z. Duan, and Y. Xu, Prognostics of battery capacity based on charging data and data-driven methods for on-road vehicles, *Applied Energy*, vol. 339, 120 954, Jun. 2023, ISSN: 0306-2619. [Online]. Available: http://dx.doi.org/10.1016/j.apenergy.2023.120954.

[141] A. Gismero, M. Dubarry, J. Guo, D. Stroe, and E. Schaltz, The influence of testing conditions on state of health estimations of electric vehicle lithium-ion batteries using an incremental capacity analysis, *Batteries*, vol. 9, 568, 12 2023.

[142] C. Weng, X. Feng, J. Sun, and H. Peng, State-of-health monitoring of lithium-ion battery modules and packs via incremental capacity peak tracking, *Applied Energy*, vol. 180, 360–368, 2016.

[143] D. Stroe and E. Schaltz, Lithium-ion battery state-of-health estimation using the incremental capacity analysis technique, *IEEE Transactions on Industry Applications*, vol. 56, 678–685, 1 2020.

[144] Q. Wang, M. Ye, X. Cai, D. U. Sauer, and W. Li, Transferable data-driven capacity estimation for lithium-ion batteries with deep learning: A case study from laboratory to field applications, *Applied Energy*, vol. 350, 121 747, Nov. 2023, ISSN: 0306-2619. [Online]. Available: http://dx.doi.org/10.1016/j.apenergy.2023.121747.

[145] Z. Li, Z. Rao, L. Pan, P. Wang, and Z. Xu, *Ti-mae: Self-supervised masked time series autoencoders*, 2023. [Online]. Available: https://arxiv.org/abs/2301.08871.

[146] H. Li, X. Shao, X. LI, F. LI, J. Fang, and W. Wei-dong, Effects of social commonweal organizations in the city brand construction of hangzhou, china—a case study of ramunion in zhejiang province, *DEStech Transactions on Social Science, Education and Human Science*, icss 2018.

[147] D. Park and C. W. Ahn, Self-supervised contextual data augmentation for natural language processing, *Symmetry*, vol. 11, 1393, 11 2019.

[148] S. Ji, S. Han, and J. Rhee, Multi-view masked autoencoder for general image representation, *Applied Sciences*, vol. 13, 12 413, 22 2023.

[149] Z. Xing, L. Zhu, L. Yu, Z. Xing, and L. Wan, Hybridmim: A hybrid masked image modeling framework for 3d medical image segmentation, 2023.

[150] G. Li, R. Togo, T. Ogawa, and M. Haseyama, Rgmim: Region-guided masked image modeling for covid-19 detection, 2022.

[151] Y. Chen, R. Yang, and R. Lan, A novel self-learning network integrating contrastive learning, perceptual learning and masked image modelling, *Fifteenth International Conference on Graphics and Image Processing (ICGIP 2023)*, 93, 2024.

[152] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, *Attention is all you need*, 2023. [Online]. Available: https://arxiv.org/abs/1706.03762.

[153] S. Hochreiter and J. Schmidhuber, Long short-term memory, *Neural Computation*, vol. 9(8), 1735–1780, 1997.

[154] I. Loshchilov and F. Hutter, Fixing weight decay regularization in adam, *CoRR*, vol. abs/1711.05101, 2017. [Online]. Available: http://arxiv.org/abs/1711.05101.

[155] L. N. Smith and N. Topin, Super-convergence: Very fast training of residual networks using large learning rates, *CoRR*, vol. abs/1708.07120, 2017. [Online]. Available: http://arxiv.org/abs/1708.07120.

[156] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary, Federated learning with personalization layers, *CoRR*, vol. abs/1912.00818, 2019. [Online]. Available: http://arxiv.org/abs/1912.00818.

[157] F. Wang, Z. Zhai, Z. Zhao, Y. Di, and X. Chen, Physics-informed neural network for lithium-ion battery degradation stable modeling and prognosis, *Nature Communications*, vol. 15(1), May 2024.

[158] Y. Luo, S. Ju, P. Li, and H. Zhang, A method for estimating lithium-ion battery state of health based on physics-informed hybrid neural network, *Electrochimica Acta*, vol. 525, 146 110, 2025.

[159] Y. Ke, M. Long, F. Yang, and W. Peng, A bayesian deep learning pipeline for lithium-ion battery soh estimation with uncertainty quantification, *Quality and Reliability Engineering International*, vol. 40(1), 406–427, 2023.