Master's Thesis in Embedded Systems Engineering

Meta Learning Approach for Battery State Estimation: A Comparative Study of Neural Networks

Markus Plandowski

Examiner:

Adviser:

Albert-Ludwigs-University Freiburg
Faculty of Engineering
Department of Sustainable Systems Engineering
Chair for Solar Energy Systems

29 August 2023

Writing Period
02. 03. 2023 – 29. 08. 2023

Examiner

Second Examiner

Adviser

Declaration

Abstract

This research delves into the use of deep learning techniques for modeling lithium-ion batteries, with a particular focus on sequence-to-sequence architectures. At the heart of this study is a comprehensive synthetic dataset, capturing a wide range of battery dynamics, sourced from validated Equivalent Circuit Models (ECMs). This dataset enables training across diverse scenarios, from static conditions like constant current charging to dynamic situations such as vehicle drive cycles. It also shows the model's ability to generalize across different battery chemistries and capacities.

Various machine learning architectures tailored for battery modeling are assessed and compared in this study. This comparison highlights the advantages and drawbacks of each model, providing insights for further optimization and model selection.

The Transformer architecture, more specifically its decoder-centric variant, is given special emphasis. With an optimized set of parameters, this model excels in predicting the State-of-Charge (SoC). Its rapid computational speed during inference underscores its practical utility. Impressively, the model demonstrates strong transfer learning abilities, adapting to different battery types like LFP/2.6Ah and NMC/64Ah, showcasing the adaptability of deep learning across diverse battery specifications. These achievements are further extended to a general meta-learning model, which is then tested on real-world data.

This research establishes a benchmark for deep learning models in battery modeling, highlighting the innovative application of the Decoder model. While some models have been previously explored, the uniqueness of this research lies in its comprehensive evaluation using a consistent dataset. By merging a dynamic synthetic dataset with a detailed model comparison and introducing the Decoder model, this study sheds light on the potential of deep learning in battery modeling. Although the real-world data accuracy hasn't surpassed traditional ECMs yet, the results emphasize the potential and effectiveness of deep learning meta-models in the battery domain.

Contents

1 Introduction

The lithium-ion battery market, and the broader battery market, is currently a focal point of global attention due to a confluence of environmental, technological, and economic factors. The importance of these is underscored by their potential to revolutionize various aspects of our lives, from transportation to energy storage, and their role in mitigating the impacts of climate change.

Environmental Considerations and Government Policies: The urgency of addressing climate change has led to a global push towards decarbonization. Governments worldwide are implementing policies to reduce carbon emissions and promote the use of renewable energy sources. Lithium-ion batteries play a crucial role in this transition. They provide a means of storing energy from renewable sources such as wind and solar, which can be inconsistent in their output. This energy can be stored when production is high and then used when production is low, thus smoothing out the supply and making renewable energy more reliable. Furthermore, batteries are key to the electrification of transportation, another major source of carbon emissions. Government policies are increasingly favoring electric vehicles (EVs) over internal combustion engine vehicles. For instance, several countries, have announced plans to ban the sale of new petrol and diesel cars within the next couple of decades. This shift towards EVs is driving their demand, which is the dominant technology for EV energy storage due to their high energy density and long cycle life [1].

Impact on Social Life and Urban Transportation: The rise of lithium-ion batteries is also having a significant impact on urban transportation and, by extension, social life. Electric bikes, scooters, and other forms of micro-mobility are becoming increasingly popular in cities around the world. These vehicles, offer a convenient, eco-friendly alternative to traditional forms of transportation, reducing traffic congestion and air pollution. Moreover, their use in public transportation systems, such as buses and trams, is growing. This not only contributes to cleaner air in urban areas but also enhances the quality of life for city dwellers by reducing noise pollution [2].

General Transportation: In the broader transportation sector, lithium-ion batteries are driving a revolution. Electric cars, which are rapidly gaining market share, rely on them for their operation. The performance and range of EVs have improved dramatically in recent years, largely due to advances in technology. This potential extends beyond road vehicles. In the aviation industry, researchers are exploring their use in electric aircraft, which could significantly reduce the sector's carbon footprint. Similarly, in the maritime industry, there is growing interest in electric ships powered by batteries [1].

Economic Implications: The growing demand is also creating economic opportunities. The market is expected to experience significant growth in the coming years, creating jobs and contributing to economic development. Moreover, countries with reserves of lithium, such as Australia, Chile, and Argentina, stand to benefit from the increasing demand for this resource [1].

The significance of the battery market is deeply rooted in its potential to pave the way for a more sustainable, low-carbon future. The development and implementation of these have profound effects on the environment, transportation, urban living, and the economy. Accurate modeling is crucial as it allows for better prediction and understanding of performance, lifespan, and safety. This is particularly important in electric vehicles and renewable energy storage systems, where efficient power management can significantly improve energy utilization and reduce costs. Furthermore, better models can help in avoiding issues such as thermal runaway and improving aging predictions, thereby enhancing safety and reliability. Thus, the importance of modeling extends beyond the battery market itself, playing a vital role in our transition toward a more sustainable and eco-friendly future.

1.1 Motivation

Lithium-ion batteries, encapsulating intricate electrochemical processes, have long posed a significant challenge for accurate modeling. The field, while advancing, has been traditionally dominated by methods such as Equivalent Circuit Models (ECMs), electrochemical and physical models. However, these methods have inherent limitations.

ECMs, which represent battery behavior via electrical circuits, often require comprehensive testing protocols and a significant amount of time for accurate parameter tuning. On the other hand, while electrochemical and physical models offer a deeper understanding of internal processes, they necessitate high expertise and more complex testing protocols. A key drawback of these methods is that once they are established and parameters determined, the data used for this purpose is typically discarded, suggesting a significant underutilization of valuable information. This circumstance indicates a significant research need: improving data efficiency and streamlining the model configuration process for cross-chemistry transfer.

The rise of deep learning and neural networks in various fields has opened new horizons for modeling. With demonstrated transformative potential in areas such as computer vision, bioinformatics, and natural language processing, neural networks are indicative of an ability to learn complex patterns from large datasets [3, 4, 5]. This makes them a promising tool for addressing the intricacies of lithium-ion batteries. However, the practical application of these advanced techniques in the realm of battery modeling remains a key concern that is yet to be fully addressed.

The potential of neural networks in revolutionizing battery modeling can be argued based on three key points:

Comprehensive Data Utilization and Simplification of Testing Protocols: Unlike traditional models, neural networks can learn from a diverse range of data, including arbitrary profiles that span possible battery behaviors. This ability could potentially simplify and reduce the cost of testing protocols, addressing an urgent research requirement unmet by conventional methods.

Democratization of Model Development and Computational Efficiency: The advent of neural networks in the field of battery modeling holds the promise of democratization. Once a neural network architecture is established, it can be trained with minimal human intervention, which is a departure from traditional methods that often require expert interpretation of test data for parameter fitting. Additionally, compared to traditional models, neural networks offer computational advantages during inference, making them a practical choice in scenarios where quick model solutions are required.

Meta-learning Across Different Chemistries with Transformer Models: Transformer models, which have shown remarkable success in various fields, hold promise for meta-learning across different battery chemistries. Unlike methods like ECMs, which often need to be redeveloped for each different chemistry, a Transformer model could potentially be trained on data from multiple chemistries. This could result in a versatile model capable of handling a variety of batteries, addressing the current research gap and streamlining the modeling process.

Thus, the use of neural networks, and Transformer models in particular, in lithium-ion battery modeling offers a promising approach to overcome the limitations of traditional methods. By maximizing data utilization, democratizing the modeling process, and enabling meta-learning across different chemistries, these models have the potential to revolutionize the field of battery modeling. Such advancements could lead to more accurate predictions of battery performance and lifespan, significantly contributing to the development of more efficient and reliable energy storage systems. Nevertheless, the current research landscape emphasizes the need for further exploration, research, and application of these innovative techniques in battery modeling.

1.2 Initial Condition

The initial condition of the thesis is characterized by a well-structured and comprehensive setup, encompassing specific resources and data. This setup provides a robust foundation for the research, ensuring that the necessary tools, information, and computational capabilities are readily available.

Measurement Data and ECM Model: The cornerstone of the thesis is the measurement data and the corresponding Equivalent Circuit Model (ECM) for two distinct types of lithium-ion battery cells: Nickel Manganese Cobalt (NMC/64Ah), and Lithium Iron Phosphate (LFP/2.6Ah). The dataset includes a dynamic profile for the NMC battery, which has not been seen by the ECM. This data is crucial as it provides a test for the model's ability to generalize beyond the conditions it was trained on. In addition to the dynamic profile, the dataset includes other measurements that were collected for the ECMs parameter estimation (pulse characterization, capacity and monotonic degradation measurements).

Functional Mock-up Unit (FMU): The ECMs are provided in a Functional Mock-up Unit (FMU), a format for model exchange and co-simulation of dynamic models. An FMU is essentially a container that encapsulates a model and its solver in a way that allows it to be shared across different simulation environments. The FMUs for this project contain the ECMs for the NMC and LFP batteries, which have been validated and parameterized based on the provided measurement data.

Computational Infrastructure: The work is carried out on a local PC running Windows, equipped with an Nvidia A5000 24GB GPU. This high-performance GPU provides substantial computational power for training neural networks. However, its memory size also sets an upper limit for the model size, complexity, and dataset size that can be handled.

1.3 Goal of this Thesis

The objective of this thesis is to develop a machine learning model for battery modeling. This encompasses several stages, each contributing to the overall integrity and efficacy of the resultant model.

Synthetic Dataset Analysis and Confidence Statement: The initial stage involves an analysis of the synthetic dataset, derived from the FMUs. Evaluating the dataset's diversity is important because it helps reduce uncertainties related to out-of-distribution scenarios, improving the model's ability to generalize. An assessment of the data's quality ensures its accurate representation of battery behavior under a variety of conditions. A confidence statement regarding the model's transferability to real-world scenarios is formulated based on the synthetic dataset. This process entails fine-tuning and testing the model on real-world data and juxtaposing its predictions with actual outcomes. The degree of alignment between the model's predictions and the actual results showcases the confidence in its transferability.

Benchmarking Machine Learning Methods: Subsequently, various machine learning architectures are benchmarked to discern the most effective one for the given battery modeling problem and dataset. This process involves training and testing each architecture on the dataset, and comparing their performance based on appropriate metrics.

Refinement and Advancement of the Premier Architecture: The leading architecture undergoes innovative modifications, with an emphasis on exploring varied encoding strategies and incorporating frequency elements. Additionally, it is subjected to parameter optimization and evaluations, ensuring its capability to manage intricate tasks and demonstrating its adaptability across diverse scenarios and tasks.

Performance Evaluation: The optimized model is evaluated on a real-world dataset to assess its generalization capability. Its performance is benchmarked against an ECM model to determine its relative effectiveness, accuracy, and adaptability.

In conclusion, the aspired target of this thesis is the development of a neural network for battery modeling that is robust, accurate, and challenges traditional models in key respects. This target involves a process of data analysis, model benchmarking and optimization, and performance evaluation.

2 Related Work

This section focuses on the current literature about battery modeling using only data-driven methods. With the fast rate of advancements and the frequent publications of new techniques, architectures, and optimization methods in Artificial Intelligence, staying updated in this area, especially when it involves deep learning, is challenging. Later sections will also cover related topics like time series and sequence-to-sequence modeling.

Deep learning techniques have been extensively utilized for State-of-Charge (SoC, the battery's remaining capacity) estimation in lithium-ion batteries, demonstrating strengths and limitations. These methods are recognized for their robust nonlinear fitting capabilities, particularly under dynamic conditions. They can automatically extract features and learn the relationship between these and the battery behavior, making them effective for SoC estimation tasks. One of the advantages of deep learning methods is their ability to handle time-series data and retain previous information, which is crucial for the desired estimation [6].

The use of 1D **Convolutional Neural Networks (CNN)** [7] and Transfer Learning has been proposed for SoC estimation. The algorithm is trained using two publicly available battery datasets and employs a transfer learning mechanism to enhance its generalization capabilities. The architecture of the 1D CNN allows for the identification of both long and short-term trends in the battery data. Key metrics of this approach are its performance in terms of estimation accuracy, learning speed, and generalization capability. The 1D CNN-based estimator performs at par with or better than existing state-of-the-art estimators. Furthermore, the time required for offline training before deploying the proposed model is lower than that for existing machine learning-based predictor examples. This time is further reduced when transfer learning is employed. The use of transfer learning also enables the model to predict accurately a target battery dataset after being trained with a significantly smaller amount of data than is required when training without transfer learning. This feature enhances the data efficiency of the model [8].

The **Long-Short-Term-Memory (LSTM)** [9] network, a variant of Recurrent Neural Network (RNN), is adept at handling sequential data, making it an ideal choice for time-series data. The GridLSTM architecture extends the LSTM methodology to both time and depth dimensions, thereby enhancing the LSTM's functionality along the depth dimension and improving the model's performance. The model excels in estimating the internal states of lithium-ion batteries by utilizing synthetic data from electrochemical simulations. It can predict internal concentrations and potentials of both electrodes using voltage, current, and temperature measurements. The model also showcases high interpolation and extrapolation capabilities, performing well under new ambient temperatures and significant measurement noise. However, the model also has certain limitations. While LSTM networks are proficient at handling long-term dependencies, they can still encounter the vanishing gradient problem during training. Similarly, the GridLSTM architecture, despite mitigating the vanishing gradient problem, is not entirely immune to it. Additionally, the model's performance can degrade at extreme temperatures due to the significant change in electrochemical dynamics. The model's generalization ability is demonstrated by its high performance at new ambient temperatures, not included in the training data. As more data becomes available, the performance of the proposed machine learning-based state estimation method will further improve [10].

Hybrid models for SoC estimation in lithium-ion batteries have been developed, combining various neural network architectures and other methods to leverage their strengths. One such model is the Convolutional Neural Network-Bidirectional Weighted Gated Recurrent Unit (CNN-BWGRU) model, which integrates the feature extraction capabilities of CNNs with the time-series data handling of the BWGRU. The CNN component extracts spatial features from voltage, current, and temperature measurements, which are subsequently processed by the BWGRU component to establish a nonlinear relationship with the SoC. The BWGRU network, considering the influence of future information on output results, is advantageous for applications with solid correlations such as battery SoC. Despite its high accuracy, robustness, and good generalization ability, the model faces challenges such as gradient disappearance and explosion during training [11].

Another hybrid model combines an RNN with an attention mechanism and a Kalman filter. The LSTM network is utilized to capture temporal dependencies in the data, while the attention mechanism is integrated to enable the model to focus on more relevant time points in the data. This attention mechanism aids the model in distinguishing between more "useful" and "less important" data, leading to faster convergence and a smaller training loss. The Kalman filter is then applied to enhance the output results of the attention network, providing an optimal estimation of the system state using data from output observations of the system. Despite the complexity of this hybrid model, it has demonstrated superior performance compared to a conventional LSTM network in terms of final loss value, convergence speed, and smoothness during training, thus underscoring its robustness and efficiency in SoC estimation [12].

Transformer-based [13] deep learning models have been employed for SoC estimation, demonstrating their suitability for multivariate time-series data. The model's training involves two stages: unsupervised pre-training and downstream fine-tuning. The unsupervised pre-training stage uses unlabeled data to train the model, while the downstream fine-tuning phase uses labeled data for supervised learning. The proposed model demonstrates superior performance in SoC estimation under various conditions, including extreme cold temperatures. Furthermore, the model's learned weights can be transferred to another cell type, showcasing the potential for transfer learning. However, the authors note that the model's performance could potentially be improved with more computational resources, allowing for larger contextual information from the past. Despite these challenges, the proposed model presents a promising approach to SoC estimation, addressing challenges related to data availability, transfer learning, training speed, and model accuracy [14].

A hybrid Transformer network leverages the strengths of deep learning and robust control theory. The architecture of this network includes two parallel encoders and one decoder. Each encoder, processing different input features, consists of a linear layer and three homogeneous encoder layers. The outputs from these encoders are then concatenated and sent to the decoder. To enhance the robustness of the SoC estimation, an L1 robust observer is integrated into the architecture. This observer is designed to correct potential learning fluctuations or errors, providing a more reliable and stable estimation. The proposed method was validated using open datasets of a lithium-ion battery under various temperatures. The authors highlight the transferability and generalizability of their model, demonstrating its potential to predict accurately under different conditions. This makes the proposed method a promising approach for SoC estimation, demonstrating the potential of hybrid Transformer networks in this application [15].

However, all these methods also have their limitations. The complexity of deep learning algorithms and the computational power required can be challenging to apply in practice, especially when it comes to the design and optimization of the neural network architecture. Furthermore, most of the existing works have conducted experiments at fixed charging and discharging current, which may not be compatible with the actual operation [6].

In conclusion, deep learning techniques, particularly hybrid models and transformer-based approaches have shown significant promise in SoC estimation for lithium-ion batteries. As more data under various scenarios become available, the performance of these machine learning-based state estimation methods will likely continue to improve.

3 Methodology

The intricate nature of battery systems necessitates the adoption of advanced modeling approaches. This is addressed through a thorough examination of the fundamental behavior of general lithium-ion battery cells, accompanied by a brief overview of traditional state-of-the-art modeling methods. The generation of machine learning-based datasets, which mirror real-world scenarios and span the full operating range of a battery, are also discussed. This highlights the critical role of comprehensive datasets in achieving robust state estimation. Machine learning models, specifically tailored for battery modeling, are analyzed based on theoretical principles, with a particular emphasis on the Transformer model. This model is singled out due to its ability to handle complex patterns and dependencies in data, as well as its high potential for transfer and meta-learning across different chemistry datasets. Moreover, a high-level description of the software architecture that establishes these models is provided in this chapter.

3.1 Fundamentals of Lithium-Ion Batteries

The Electrochemical Model (often represented by the Doyle-Fuller-Newman model) and the Reduced Order Models (often represented by the Equivalent Circuit Model) are fundamental to lithium-ion battery modeling. The Electrochemical Model, while computationally demanding and requiring extensive parameter estimation and testing, provides a high level of accuracy, making it valuable for a detailed understanding of battery behavior. The Reduced Order Model, which is partly derived from the Electrochemical Model, is less complex and requires fewer computational resources and time for parameter estimation, making it suitable for quick, system-level simulations. Both models are crucial in the battery modeling world, with the Electrochemical Model offering depth and accuracy, and the Reduced Order Model providing efficiency and speed.

The Doyle-Fuller-Newman (DFN) model is a widely used electrochemical model for predicting the behavior of lithium-ion batteries and is shown in Figure 1. It balances computational efficiency with accuracy by using a set of partial differential equations to describe complex processes like lithium-ion diffusion (movement of ions due to concentration gradients) and electrochemical reactions within the battery. This balance makes it a practical tool for battery researchers and engineers, enabling the design of more efficient and reliable batteries. A comprehensive model typically requires hours to spatially discretize the system and solve it numerically as a set of differential equations. Most current lithium-ion battery models are derived from theories that describe charge/discharge and species (chemical participants in battery reactions) transport in the solid and electrolyte phases across a simplified spatial cell structure. This model considers dynamics along the x-axis and neglects the dynamics along the remaining two axes to reduce complexity.

3.1.1 Doyle-Fuller-Newman Model

A typical lithium-ion cell consists of four main components: the negative composite electrode, the positive composite electrode, the separator, and the electrolyte. The negative electrode contains lithium stored in graphite lattice sites, while the positive electrode can have various chemistries, usually a metal oxide. During discharge, lithium ions diffuse to the surface of active material particles in the negative electrode, undergo electrochemical reactions, and transfer into a liquid electrolyte solution. The ions then travel through the electrolyte solution to the positive electrode where they react and diffuse towards the inner regions of metal oxide active material particles. The process is reversed for charging.

Lithium is considered to exist in two disjoint states called phases: the solid phase in the electrode material and the liquid phase in the dissolved state in the electrolyte. The solid and electrolyte phases are treated as superimposed continua without regard to microstructure [17].

The lithium-ion concentration in a single spherical active material particle can be described by the equation

$$\frac{\partial c_{s,k}(x,r,t)}{\partial t} = \frac{D_{s,k}}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial c_{s,k}(x,r,t)}{\partial r}\right) \tag{1}$$

where $c_{s,k}(x,r,t)$ is the concentration of lithium ions in the solid phase at position x, radius r, and time t, and $D_{s,k}$ is the diffusion coefficient in the solid phase.

The lithium ions in the electrolyte phase change due to the changes in the gradient diffusive flow of lithium ions, which can be described by the equation

$$\epsilon_k\frac{\partial c_{\mathrm{e},k}(x,t)}{\partial t} = \frac{\partial}{\partial x}\left(D_{\mathrm{eff},k}\frac{\partial c_{\mathrm{e},k}(x,t)}{\partial x}\right) + a_k\left(1-t_+\right)J_k(x,t) \tag{2}$$

where $c_{\mathrm{e},k}(x,t)$ is the concentration of lithium ions in the electrolyte phase at position x and time t, ϵ_k is the electrolyte volume fraction, $D_{\mathrm{eff},k}$ is the effective diffusion coefficient in the electrolyte phase, a_k is the electrode surface area per unit volume, t_+ is the transference number, and $J_k(x,t)$ is the molar flux.

Charge conservation in the solid phase of each electrode can be described by the equation

$$\sigma_{\mathrm{eff},k}\frac{\partial^2\Phi_{\mathrm{s},k}(x,t)}{\partial x^2} = a_k F J_k(x,t) \tag{3}$$

where $\sigma_{\mathrm{eff},k}$ is the effective conductivity in the solid phase, $\Phi_{\mathrm{s},k}(x,t)$ is the solid phase potential at position x and time t, F is the Faraday constant, and $J_k(x,t)$ is the molar flux.

Combining Kirchhoff's law with Ohm's law in the electrolyte phase yields the equation

$$-\sigma_{\mathrm{eff},k}\frac{\partial\Phi_{\mathrm{s},k}(x,t)}{\partial x} - \kappa_{\mathrm{eff},k}\frac{\partial\Phi_{\mathrm{e},k}(x,t)}{\partial x} + \frac{2\kappa_{\mathrm{eff},k}(x,t)RT}{F}\left(1-t_+\right)\frac{\partial\ln c_{\mathrm{e},k}}{\partial x} = I \tag{4}$$

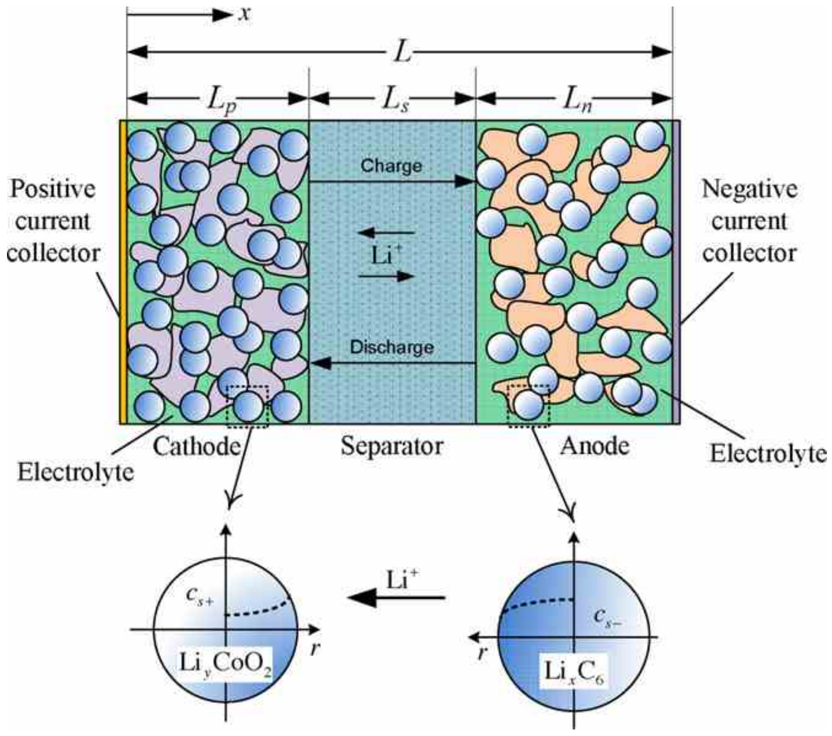

Figure 1: Doyle-Fuller-Newman model: **Anode**: The negative electrode in the battery where lithium ions are stored during charging and released during discharging. **Cathode**: The positive electrode in the battery that receives lithium ions during charging and releases them during discharging. **Separator**: A porous insulating layer that prevents direct contact between the anode and cathode, while allowing lithium ions to pass through. **Electrolyte**: The medium that facilitates the movement of lithium ions between the anode and cathode during charging and discharging. **Pos. Current Collector**: A conductive material at the cathode side that collects the electrical current generated by the movement of electrons during discharging. **Neg. Current Collector**: A conductive material at the anode side that collects the electrical current during charging. **Mass Transfer (Li-Ions)**: Refers to the movement of lithium ions from the anode to the cathode during discharging, and vice versa during charging. **Charge Transfer (Electrons)**: Refers to the movement of electrons from the anode to the cathode through an external circuit during discharging, and vice versa during charging. This movement of electrons is what we use as electrical power. **Particle (Cathode)**: In the DFN model, the lithium-ion particle at the cathode represents the concentration of lithium ions at the particle surface. This concentration typically increases during discharge due to the diffusion (movement of ions due to concentration gradients) of lithium ions from the core to the surface, indicating the influx of lithium ions and vice versa for **Particle (Anode)** [16].

where $\kappa_{\mathrm{eff},k}$ is the effective conductivity in the electrolyte phase, $\Phi_{\mathrm{e},k}(x,t)$ is the electrolyte phase potential at position x and time t, R is the gas constant, T is the temperature, and I is the current.

The molar flux depends on the concentration of lithium ions in the electrode, the concentration of lithium ions in the electrolyte, and the intercalation over-potential through the Butler–Volmer equation. This over-potential can be described by

$$\mu_{s,k}(x,t) = \Phi_{s,k}(x,t) - \Phi_{e,k}(x,t) - U_k\left(\theta_k(x,t)\right) \tag{5}$$

where $\mu_{s,k}(x,t)$ is the over-potential, $\Phi_{s,k}(x,t)$ and $\Phi_{e,k}(x,t)$ are the solid and electrolyte phase potentials at position x and time t respectively, $U_k(\theta_k(x,t))$ is the open-circuit potential as a function of the stoichiometry $\theta_k(x,t)$.

The Butler–Volmer equation describes the relationship between the current density, concentrations and over-potential is given by the equation

$$\begin{aligned} J_k(x,t) = {} & K_k\left(c_{s,k,\mathrm{max}} - c_{s,k,\ \mathrm{surf}}\right)^{0.5}\left(c_{s,k,\ \mathrm{surf}}\right)^{0.5} c_{\mathrm{e},k}^{0.5}(x,t) \\ & \times\left[\exp\left(\frac{0.5F}{RT}\mu_{\mathrm{s},k}(x,t)\right) - \exp\left(-\frac{0.5F}{RT}\mu_{s,k}(x,t)\right)\right] \end{aligned} \tag{6}$$

where $J_k(x,t)$ is the molar flux, K_k is the reaction rate constant, $c_{s,k,\mathrm{max}}$ is the maximum concentration in the solid phase, $c_{s,k,\ \mathrm{surf}}$ is the surface concentration in the solid phase, $c_{\mathrm{e},k}(x,t)$ is the concentration in the electrolyte phase at position x and time t, F is the Faraday constant, R is the gas constant, T is the temperature, and $\mu_{\mathrm{s},k}(x,t)$ and $\mu_{s,k}(x,t)$ are the over-potentials.

In summary, the battery model is a mixed system with a certain number of nonlinear partial differential-algebraic equations with the same number of unknowns. However, it's important to note that only the most critical equations are stated here to provide a general understanding of the complexity involved. The complete model, based on the differential equations and finite difference method, contains additional equations and initial conditions, further adding to the complexity and the computational requirements for solving the model [16].

For instance, the terminal voltage of the battery can be calculated from the difference between the solid phase potential in the positive electrode ($\Phi_{s,p}$) and the solid phase potential in the negative electrode ($\Phi_{s,n}$), minus the overpotential due to the reaction kinetics and ionic transport in the electrolyte (η). This can be represented by the following equation:

$$V(t) = \Phi_{s,p}(t) - \Phi_{s,n}(t) - \eta(t) \tag{7}$$

Similarly, the State-of-Charge (SoC) can be calculated by integrating the lithium ion concentration in the solid phase over the volume of the electrode. This is because the amount of lithium in the electrode material corresponds to the amount of charge stored in the battery. The equation for the SoC can be expressed as follows:

$$SoC(t) = \frac{1}{V_{\text{total}}} \int_{V_{\text{electrode}}} c_{s,k}(x, r, t)\, dV \tag{8}$$

With V_{total} is the total volume of the battery, $V_{\text{electrode}}$ is the volume of the electrode, and $c_{s,k}(x, r, t)$ is the concentration of lithium ions in the solid phase at position x within the electrode, at a radial distance r from the center of the active material particle, at time t.

This level of detail already goes beyond the scope of this thesis. For a more comprehensive understanding and further information on the full model, please refer to the original sources Dao et al., Doyle et al., Fuller et al., Newman et al. [16, 18, 19, 20].

3.1.2 Equivalent Circuit Model

Equivalent Circuit Models (ECMs) offer a balance between theoretical quantitative models, which require complex measurements and parameters, and purely mathematical models, which rely heavily on empirical data. The strength of ECMs lies in their ability to simulate the electrical behavior of batteries without needing extensive physical information. They can be tailored to the specific cell chemistry and characteristics, avoiding the risk of over- or under-modeling. The choice of the equivalent circuit is critical, and it is guided by understanding the basic structure of a battery and the estimation of the parameters of the equivalent circuit. The use of techniques such as the least-squares method aids in parameter extraction. The versatility and adaptability of ECMs make them an invaluable asset in the modeling of lithium-ion batteries [17].

ECMs are used to represent the electrochemical and electrical components of a battery cell. The series connection of these equivalent circuit elements forms the entire model for a battery cell. The architecture of an ECM typically includes resistors, capacitors, inductors, constant-phase-elements (CPE, a non-ideal capacitor), and Warburg elements (diffusion impedance element), each representing different aspects of the battery's behavior. These elements, as shown in Figure 2, represent ohmic resistances, and capacitive behavior due to charge accumulation and diffusion processes, respectively.

Parameter estimation is crucial in ECM modeling. The parameters are adjusted to minimize the difference between the measured impedance spectrum from Electrochemical Impedance Spectroscopy (EIS) and the ECM's impedance. EIS measures the battery's impedance over a range of frequencies, providing a spectrum that reflects its electrochemical characteristics and processes. This spectrum informs the parameter estimation for the ECM, as represented by the following equation:

$$\underset{R,C,L,CPE,W}{\arg\min} \sum_{n=f_{\min}}^{f_{\max}} \left| Z_{\text{EC}}(R, C, L, CPE, W, f_n) - Z_{\text{measure},n} \right|^2 \tag{9}$$

In the given equation, Z_{EC} denotes the impedance of the equivalent circuit. The parameters R, C, L, CPE, and W define the circuit's characteristics. f_n is the frequency, and $Z_{\text{measure},n}$ is the impedance obtained from EIS measurements. The objective is to minimize the discrepancy between the equivalent circuit's impedance and the measured

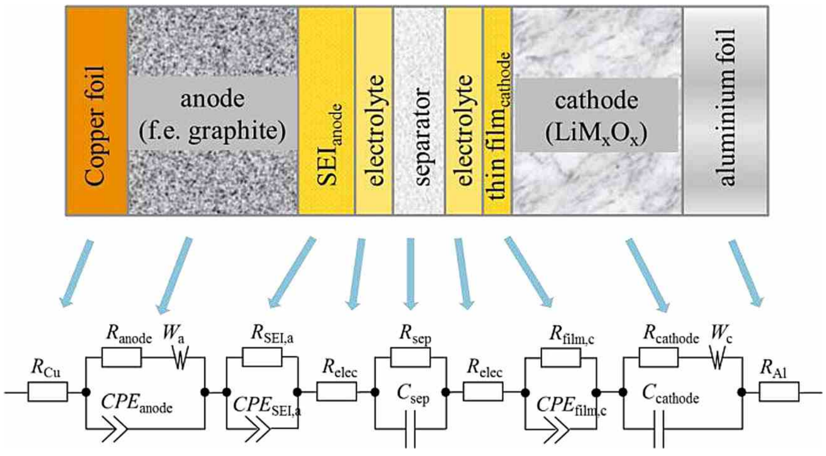

Figure 2: Resulting equivalent circuit elements from individual cell components. $\mathbf{R}_{Cu}$: Resistance of the copper current collector at the anode, accounting for ohmic losses. $\mathbf{R}_a$, $\mathbf{W}_a$, $\mathbf{CPE}_a$: R_a is the anode resistance, W_a is the Warburg impedance (diffusion - movement of ions due to concentration gradients), and CPE_a is a Constant Phase Element representing non-ideal capacitive behavior, often due to material properties. $\mathbf{R}_{SEI,a}$, $\mathbf{CPE}_{SEI,a}$: Represent the SEI layer at the anode, a passivation layer that stabilizes the cell but adds to resistance and limits ion transport, impacting battery performance. $\mathbf{R}_{electrolyte}$: Represents the resistance of the electrolyte, the medium for lithium-ion migration (movement of ions due to potential difference) between the anode and cathode. $\mathbf{R}_{seperator}$, $\mathbf{C}_{seperator}$: Represent the separator, a barrier allowing ion flow but preventing direct anode-cathode contact. $R_{seperator}$ is the separator resistance, and $C_{seperator}$ is its capacitance. $\mathbf{R}_{filmcathode}$, $\mathbf{CPE}_{filmcathode}$: Represent the film layer at the cathode, a passivation layer similar to the SEI at the anode. This layer adds to resistance and impacts ion transport, affecting battery performance. $\mathbf{R}_{cathode}$, $\mathbf{W}_{cathode}$, $\mathbf{C}_{cathode}$: $R_{cathode}$ is the cathode resistance, $W_{cathode}$ is the Warburg impedance (diffusion), and $C_{cathode}$ is the capacitance, typically exhibiting more ideal behavior than the anode. $\mathbf{R}_{Al}$: Resistance of the aluminum current collector at the cathode, accounting for ohmic losses [17].

impedance across frequencies ranging from $f_{\min}$ to $f_{\max}$. EIS offers experimental data essential for adjusting the ECM parameters, ensuring the model mirrors the battery's actual behavior. This approach is pivotal for parameter estimation, making it essential for developing a dependable and precise battery model [17]. Often, simplified versions of this intricate yet accurate ECM are employed. These versions exclude CPEs and Warburg elements, typically comprising one to three RC branches. They may not require EIS measurements for parameter estimation, as the necessary parameters can be derived from the time domain of pulse characterization.

Furthermore, like electrochemical models, ECMs can also be used to estimate the State-of-Health (SoH) of a battery, which is a measure of its current condition compared to its ideal state. The SoH estimation is a complex process that requires extensive measurement data, often collected over several months. This data, which includes the battery's response to various operating conditions and states of charge, is used to adjust the parameters of the ECM over time, allowing the model to accurately reflect the aging processes and degradation mechanisms occurring within the battery. This long-term tracking and modeling of the battery's behavior provide valuable insights into its SoH, enabling more effective battery management and prolonging its useful life. It is only for completeness stated here and will not be part of the study.

3.2 Synthetic Dataset Generation for Machine Learning

Synthetic data generation is a pivotal technique in machine learning, particularly relevant to battery technology applications. This approach creates artificial data that closely mirror real-world battery operation data, proving invaluable when such data is limited, expensive to gather, or when privacy concerns restrict its use. A diverse synthetic dataset that encapsulates the entire operational spectrum of a battery system is crucial. It ensures that the machine learning model is exposed to a wide array of battery performance scenarios, enabling it to learn and generalize effectively. This is highly important in battery technology where the operational field is vast, encompassing various charge-discharge cycles, ambient conditions, and usage patterns. Furthermore, the quality and diversity of the dataset significantly influence the speed and stability of model convergence, a point where the model's learning plateaus after being trained on the dataset. A well-constructed synthetic dataset can ensure faster and more stable convergence, leading to a more accurate and reliable model for predicting battery performance and lifespan [21, 22].

3.2.1 Dataset Design for Comprehensive Battery Operation

The generation of synthetic battery data is a process that is based on the utilization of an ECM with multiple Resistance-Capacitance (RC) branches. This model is validated based on corresponding measurements for parameter estimation. The ECM model is simple in its requirements, needing only a current profile and conditions such as ambient temperature and initial State-of-Charge (SoC). The output from the ECM model is terminal voltage, surface temperature, and SoC.

The current profile is generated using a set of subprofiles. These include constant, linear, step, sinusoidal, and drive-cycle slices. Each of these subprofiles represents a different real-world scenario that a battery might encounter. For instance, a constant current profile could represent a battery providing a steady power output, while a linear profile could represent a battery whose load is gradually increasing or decreasing. A step profile could mimic a sudden change in load, such as turning on a large appliance, while a sinusoidal subprofile could represent a cyclic load pattern. Drive-cycle slices, on the other hand, are derived from real-world driving data and are representative of the varied and unpredictable power demands of an electric vehicle. These drive cycles are sourced from PyBaMM, an open-source battery modeling software [23]. The drive-cycle profile is split into buckets, each of which is randomly chosen and concatenated to form the complete subprofile.

The dataset contains individual instances of these subprofiles, a concatenated and superimposed version of them. The concatenated version is created by joining different subprofiles end-to-end until the total time length is reached. The superimposed version is created by overlaying different subprofiles on top of each other until the total time length is reached. The subtimeframes for the concatenated and superimposed profiles are generated at random. Each parameter for these profiles is sampled from a uniform distribution. This includes time, amplitude, and offsets for all profiles. For the sinusoidal subprofile, frequency and phase shift are also sampled from a uniform distribution. These profiles are not generated without constraints. They are designed to not exceed the SoC range and not exceed the maximum allowable current for charging and discharging, ensuring the synthetic data remains representative of feasible real-world conditions.

The process begins with the ECM model, into which the constrained current profile is fed. The model processes this input and generates outputs containing terminal voltage, surface temperature and State-of-Charge.

The generic equations for the random subprofile can be represented as follows:

For a constant current profile:

$$I(t) = I_0 \tag{10}$$

For a linear current profile:

$$I(t) = I_0 + kt \tag{11}$$

For a step current profile:

$$I(t) = \begin{cases} I_1 & \text{if } t < t_1 \\ I_2 & \text{if } t \geq t_1 \end{cases} \tag{12}$$

For a sinusoidal current subprofile:

$$I(t) = I_0 + A\sin(\omega t + \phi) \tag{13}$$

For a drive-cycle subprofile:

$$I(t) = \text{concat}(I_{\text{bucket}_1}, I_{\text{bucket}_2}, ..., I_{\text{bucket}_n}) \tag{14}$$

For a concatenated profile with n segments:

$$I(t) = \text{concat}(I_1(t), I_2(t), ..., I_n(t)) \tag{15}$$

For a superimposed profile with n segments:

$$I(t) = \sum_{i=1}^{n} I_i(t) \tag{16}$$

In these equations, $I(t)$ represents the current at time t, I_0 is the initial current, k is the rate of change of current, t_1 is the time at which the step change occurs, I_1 and I_2 are the current values before and after the step change, A is the amplitude of the sinusoidal current, ω is the angular frequency, and ϕ is the phase. I_{bucket_i} represents each bucket in the drive-cycle subprofile.

The constraints on the profiles can be represented as:

$$\begin{gathered} 0 \leq SoC \leq 1 \\ -I_{max} \leq I(t) \leq I_{max} \end{gathered} \tag{17}$$

where I_{max} is the maximum allowable charging or discharging current and the SoC is determined by the integration of the current over time and standardized afterward. The synthetic data generated is representative of a wide range of real-world scenarios. The random generation of subtimeframes for the concatenated and superimposed profiles, along with the random sampling of parameters from a uniform distribution, ensures a diverse and comprehensive dataset, further enhancing its utility in real-world applications. These profiles are generated multiple times with different initial conditions for ambient temperature and corresponding different current boundaries, further increasing the diversity and representativeness of the synthetic data.

3.2.2 Dataset Analysis for Diversity and Learning Convergence

The intricate dynamics of battery systems necessitate comprehensive data analysis to understand their behavior and performance. A detailed examination of this data is solely illustrated on the LFP cell for one-third of the dataset.

Over a defined period, a concatenated current profile is shown in Figure 3. This profile, created from the integration of several subprofiles, follows battery-specific criteria such as current and capacity limits. It displays multiple charge and discharge cycles in the series. Parameters like voltage, temperature, and SoC are derived from an ECM based on this profile.

From Figure 3, two main observations can be made. Firstly, the current, which is crucial for the ECM, displays significant variations, affecting the voltage. These variations result from the battery's electrochemical reactions. When current flows through the battery, the cell voltage experiences an immediate change due to internal resistance, intensified by rapid ion movements.

In contrast, temperature and SoC exhibit more stable behaviors. The temperature

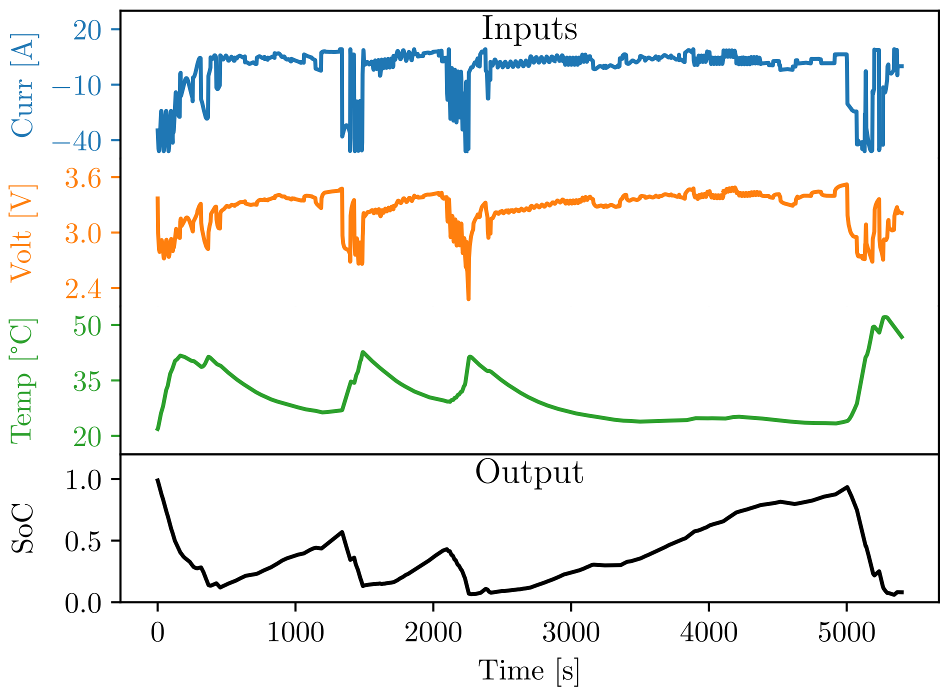

Figure 3: Concatenated synthetic battery data profile over a specified duration. The top three subplots present the inputs: current, voltage, and temperature. The bottom subplot illustrates the SoC evolution. Derived from an ECM using diverse subprofiles, the high dynamics in current and voltage are attributed to immediate electrochemical reactions. Conversely, the temperature and SoC dynamics, shaped by cumulative effects, are more tempered. These profiles illuminate the battery's behaviors across different real-world scenarios.

reflects both reversible and irreversible reactions and its gradual change represents the combined heat effects, differing from the current and voltage behaviors. The SoC's steady changes are due to its role in summarizing cumulative charge and discharge activities, smoothing out quick fluctuations.

Outside the shown profile, the larger dataset contains various subprofiles. These are essential for enhancing the synthetic battery data. The profiles increase the dataset's adaptability, ensure model stability for different inputs, support in-depth analysis, and provide adaptability for specific research needs.

Figure 4 displays a 3D visualization of synthetically generated current profiles, which have been reduced in dimensionality using Principal Component Analysis (PCA). Initially, these profiles existed in high-dimensional sequences. PCA transforms these sequences into three principal components, capturing the most significant features and variances of the data.

PCA aims to maximize data variance with the fewest components. Thus, the positioning and sizes of clusters in this reduced space reflect their relationships in the original space.

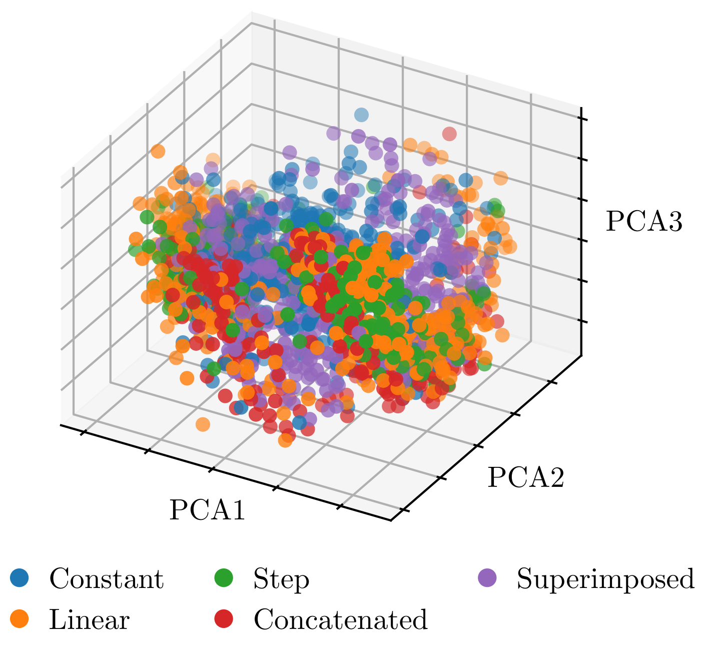

Figure 4: Three-dimensional PCA representation of synthetically generated current profiles. Distinct clusters correspond to various profile types, with overlaps indicating shared electrochemical characteristics. The plot encapsulates the intricate balance of foundational patterns, complexities from sinusoidal and drive-cycle subprofiles, and operational constraints, highlighting the diverse, realistic, and safe spectrum of synthetic battery data.

The patterns and differences seen among clusters represent the relationships in the full-length current profiles.

Clusters in the scatter plot symbolize different current profiles. Overlaps between clusters indicate similarities in profile traits. For instance, segments in the "concatenated" profile might resemble those in "linear" or "step" profiles. These overlaps highlight the nuances in synthetic current profile creation and suggest that various generation methods can produce similar electrochemical patterns.

The dynamics in the "concatenated" and "superimposed" clusters arise from the inclusion of sinusoidal and drive-cycle subprofiles. Sinusoidal subprofiles introduce periodic changes, while drive-cycle subprofiles, derived from real driving scenarios, add unpredictability. Their combination results in a complex pattern, evident in the broad distribution of these clusters.

Examining cluster positions offers insights. Distinct clusters indicate unique generation methods, while close clusters suggest shared characteristics. A cluster's size might indicate the diversity within that profile type.

The scatter plot's boundaries reflect the limits set during synthetic current profile creation. Amplitude boundaries correspond to charging and discharging limits for battery safety.

Continuing to the according to input representation, Figure 5 presents a 3D visualization of the synthetic dataset derived from the ECM, illustrating the relationships between three critical battery parameters: current, voltage, and temperature.

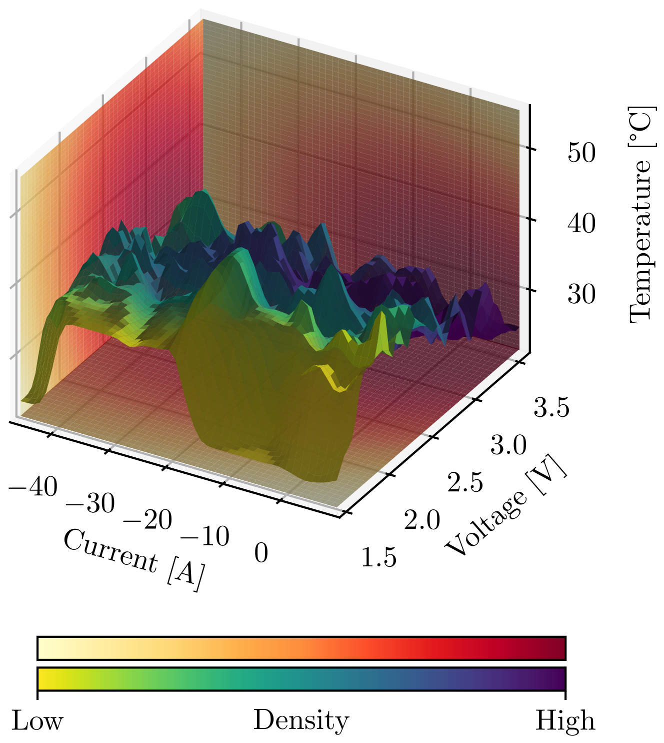

Figure 5: Three-dimensional visualization of synthetic battery data from the ECM, showcasing the relationships between current, voltage, and temperature. Each axis defines their operational ranges. Color gradients indicate data point densities, emphasizing regions of higher concentration and shedding light on the battery scenario distribution.

Each visualization axis has specific constraints. The current axis is limited by the battery's maximum charging and discharging currents. The voltage axis defines the battery's safe operational range, while the temperature axis covers the battery's viable temperature range.

The primary 3D surface plot, smoothed using Gaussian filtering, depicts temperature as a function of current and voltage. The filtering technique reduces potential high-frequency noise, making the visualization clearer. Color gradients, determined using Kernel Density Estimation (KDE), highlight data point densities. KDE offers insights into dataset patterns and concentrations.

Examining further, positive current regions correspond to charging, while negative ones relate to discharging. The surface's variations suggest transients in the original profiles. In an unsmoothed version, these would manifest as distinct spikes, indicating sudden battery state changes reflecting response to unpredictable demands. Slow or minimal changes represent steady-state operations.

The visualization emphasizes both dynamic and static battery processes. The charging and discharging phases, combined with transient data, attest to the dataset's comprehensive nature. It provides a wide perspective on potential battery scenarios and confirms the reliability of its synthetic current profile generation. The visualization represents a dataset that captures a broad spectrum of real-world battery behaviors, balancing diversity, realism, and safety.

Examination of model complexity about dataset size is crucial for optimal performance. Figure 6 delves into this relationship, analyzing the interplay between the optimizer's update steps, dataset size, and the losses from an optimized decoder-only model. As training progresses, a marked reduction in loss is evident, signifying the model's capacity to refine its predictions over iterations. Interestingly, despite the application of regularization techniques during training, the training loss remains consistently higher than the validation loss.

When analyzing the test loss with the growing dataset size, a clear inverse relationship emerges. The loss diminishes, suggesting an optimal dataset size that ensures efficient training while preserving the model's predictive accuracy.

A deeper dive into the decoder-only model structure reveals that, given adequate training time and model size, the validation loss converges to near-perfect accuracy levels. This precision starkly contrasts with the test loss, which accounts for concatenated outputs across various time windows. The model's training technique, predicting subsequent time steps based on prior accurate data rather than its predictions, further supports this observation. It's worth noting that during both training and evaluation phases, the model consistently relies on actual data inputs, avoiding its predictions, unlike in the test scenario with multiple windows.

In conclusion, thorough evaluations validate the dataset's robustness, its fitting size, and its diverse nature, making it well-suited for intricate battery modeling challenges. The model discussed, which will be elaborated on in subsequent sections, represents the final optimized version. This section's emphasis is on the dataset's appropriateness and the reasoning behind its size to the selected model.

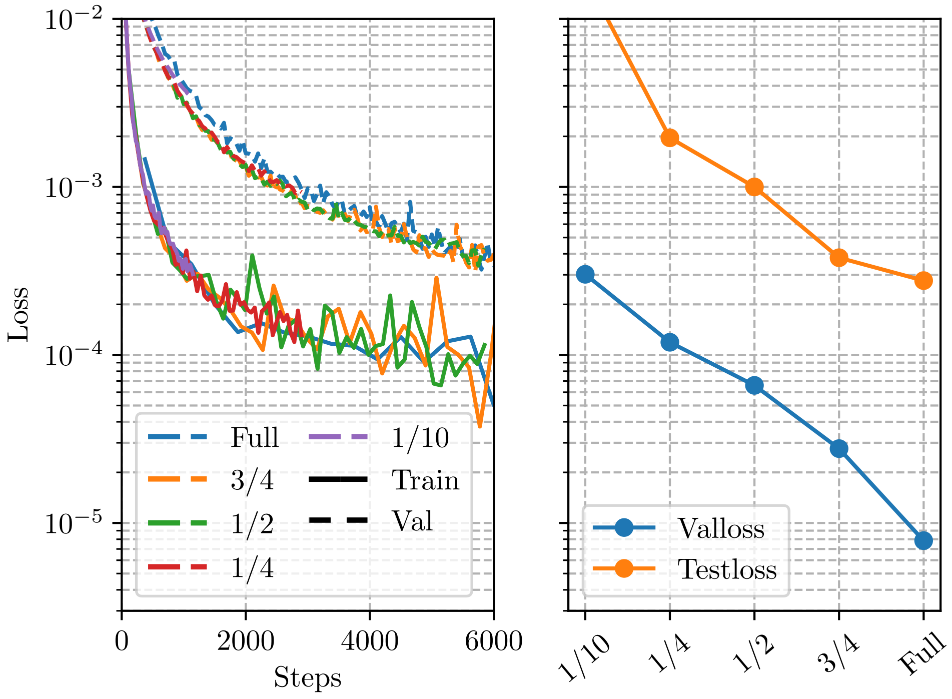

Figure 6: Visualization highlights the connection between optimizer update steps, dataset size, and model losses. Regularization techniques result in a higher training loss compared to validation. An increasing dataset size correlates with a decreasing test loss, pointing towards an optimal data threshold. Within the decoder-only setup, the validation loss achieves high accuracy, contrasting with the test loss derived from concatenated outputs.

3.3 Principles of Selected Machine Learning Models

In battery technology, computational modeling is essential for analyzing complex mechanisms and accurate predictions. The Extreme Gradient Boosting (XGBoost) algorithm is valued for its ability to model non-linear battery relationships. State-Space Neural Networks capture battery dynamics over time, while Convolutional Neural Networks (CNNs) analyze spatial data structures in battery systems. Recurrent Neural Networks (RNNs) are effective for sequence data, making them suitable for time-dependent battery attributes. Transformer-based Neural Networks, with attention mechanisms, provide a deeper understanding of battery behaviors. These machine learning models aim to enhance battery modeling and analysis techniques.

3.3.1 Extreme Gradient Boosting

The XGBoost regression model, a scalable tree-boosting system, is a compelling choice for tasks such as battery modeling. This model operates by creating an ensemble of decision trees, where each subsequent tree is built to correct the errors made by the previous ones. This is achieved through a process known as gradient boosting, which involves the application of the gradient descent algorithm to minimize the loss when adding new models.

The model is trained in an additive manner, with each tree being added to minimize the objective function, which is a combination of a differentiable loss function and a regularization term. The regularization term helps to control the model's complexity, reducing overfitting and improving generalization. The objective function is optimized using a second-order approximation, which can be computed efficiently and makes the model robust to outliers.

The XGBoost regression model is popular due to its scalability and efficiency. It runs significantly faster than many existing solutions and can handle large datasets, making it suitable for real-world applications. It also supports parallel and distributed computing, which further enhances its speed and scalability.

The main parameters of the XGBoost regression model include the maximum depth of the trees, the learning rate (also known as shrinkage), and the subsampling ratio. The maximum depth controls the complexity of the trees, the learning rate determines the step size at each iteration of the gradient descent algorithm, and the subsampling ratio specifies the fraction of the training data to be used for growing each tree, which can help to prevent overfitting.

In the context of battery modeling, the XGBoost regression model can capture complex nonlinear relationships between the input features and the target variable. However, it's important to note that the XGBoost regression model requires a flattened input structure. This means that all input features must be transformed into a one-dimensional array or vector. The advantage of this approach is that it allows the model to handle a wide range of data structures, from simple numerical data to more complex time-series data.

The flattening process maintains the correspondence between each input and its associated output during training. This is a crucial aspect of the XGBoost algorithm's (Algorithm 1) design, as it ensures that the model learns the correct associations between the input features and the target variable, even when the input data is transformed.

Algorithm 1 High-Level XGBoost Working Principle

```
1: inputs ← flatten(inputs)
2: model ← initialize model
3: for $t \in$ range(iterations) do
4:     Compute gradient and hessian of loss based on flattened inputs
5:     tree ← fit new decision tree to the gradient statistics
6:     Update the model by adding the new tree
7: end for
8: predictions ← model.predict(inputs)
9: loss ← loss(predictions, outputs)
```

By maintaining the correspondence between the flattened inputs and outputs, the XGBoost regression model can effectively learn the complex relationships in the data, making it a powerful tool for battery modeling [24].

3.3.2 State-Space Neural Network

The S4 model, also known as Structured State Spaces, is a method for battery modeling and other applications that necessitate the management of long-range dependencies (LRDs). This model is built upon several key components, each contributing to its overall functionality and efficiency illustrated in Figure 7.

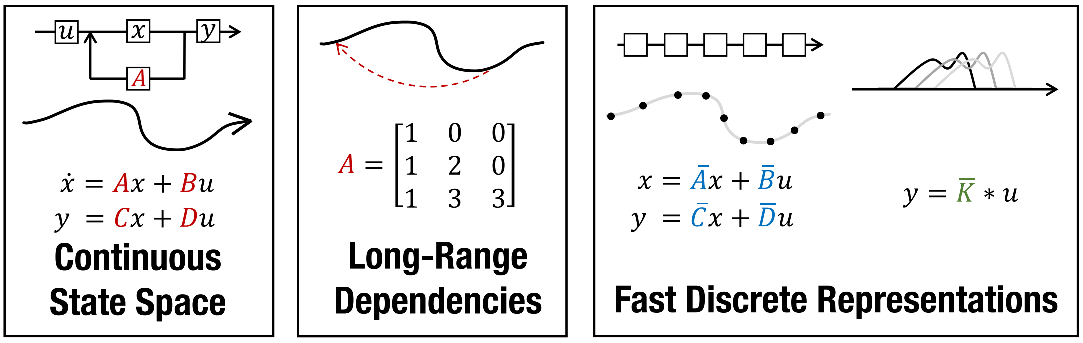
u
x
y
A
$\dot{x} = Ax + Bu$
$y = Cx + Du$
Continuous State Space
$A = \begin{bmatrix}1 & 0 & 0\\1 & 2 & 0\\1 & 3 & 3\end{bmatrix}$
Long-Range Dependencies
$x = \bar{A}x + \bar{B}u$
$y = \bar{C}x + \bar{D}u$
$y = \bar{K} * u$
Fast Discrete Representations

Figure 7: Left: State Space Models (SSMs), characterized by matrices **A**, **B**, **C**, and **D**, facilitate the transformation of an input signal $u(t)$ into an output $y(t)$ via a hidden state $x(t)$. **Center**: Recent theoretical advancements in continuous-time memorization has led to the development of unique **A** matrices, enabling SSMs to mathematically and empirically encapsulate Long-Range Dependencies (LRDs). **Right**: SSMs can be calculated either through a recurrent method or a convolutional approach. However, actualizing these theoretical perspectives necessitate the use of different parameter representations (indicated in red, blue, green), which are computationally intensive. S4 presents an innovative parameterization that can efficiently alternate between these representations, thereby making it capable of managing a broad spectrum of tasks, optimizing both training and inference processes and excelling in handling extended sequences [25].

The foundation of the S4 model is centered around the state-space representation, emphasizing the continuous-time latent state model. This can be represented as:

$$\begin{aligned} x'(t) &= \mathbf{A}x(t) + \mathbf{B}u(t) \\ y(t) &= \mathbf{C}x(t) + \mathbf{D}u(t) \end{aligned} \tag{18}$$

In this formulation, $x(t)$ is the latent state of the system at time t, encapsulating a hidden or internal representation of the system's instantaneous status. $x'(t)$ quantifies the temporal evolution of this latent state, reflecting its rate of change. Concurrently, $y(t)$ is the observable output, representing the system's externally perceivable prediction based on its current state and the immediate input. The matrix **A** is the state transition matrix, containing the dynamics of the system. This intrinsic behavior is modulated by the external input $u(t)$ through the input matrix **B**. The matrices **C** and **D** jointly describe the interplay between the latent state and the system's output, with **C** steering the output based on the internal state and **D** underscoring any direct influence of the input on the output.

To manage long-range dependencies, the S4 model incorporates the HiPPO (Optimal function approximation with time-varying measures) method [26]. This method is encapsulated by the equation:

$$\mathbf{A}_{n,k} = -\begin{cases} (2n+1)^{1/2}(2k+1)^{1/2} & \text{if } n > k \\ n+1 & \text{if } n = k \\ 0 & \text{if } n < k \end{cases} \tag{19}$$

Where $\mathbf{A}_{n,k}$ is the element of the HiPPO matrix at the n-th row and k-th column.

The S4 model also includes a Recurrent Representation, which is expressed as:

$$\begin{aligned} x_k &= \bar{\mathbf{A}}x_{k-1} + \bar{\mathbf{B}}u_k & \bar{\mathbf{A}} &= (\mathbf{I} - \frac{\Delta}{2}\cdot\mathbf{A})^{-1}(\mathbf{I} + \frac{\Delta}{2}\cdot\mathbf{A}) & & \\ y_k &= \bar{\mathbf{C}}x_k & \bar{\mathbf{B}} &= (\mathbf{I} - \frac{\Delta}{2}\cdot\mathbf{A})^{-1}\Delta\mathbf{B} & \bar{\mathbf{C}} &= \mathbf{C} \end{aligned} \tag{20}$$

Where x_k and x_{k-1} represent the latent states at times k and $k-1$ respectively, and u_k is the input at time k.

The convolutional representation in the S4 model establishes a connection between linear time-invariant (LTI) State Space Models (SSMs) and continuous convolutions (Algorithm 2). The S4 model introduces a novel parameterization that efficiently swaps between these representations, allowing it to handle a wide range of tasks, be efficient at both training and inference and excel at long sequences.

Technically, S4 reparameterizes the structured state matrices by decomposing them as the sum of a low-rank and normal term. Additionally, instead of expanding the standard SSM in coefficient space, it computes its truncated generating function in frequency space. Combining these two ideas, the low-rank term can be corrected while the normal term can be diagonalized stably. This results in a significant improvement in both computation and memory usage, making the S4 model more efficient and practical for a wide range of applications [25].

In summary, the S4 model is a powerful and efficient tool for handling long sequences and complex data structures. It leverages the strengths of State Space Models and the HiPPO method for handling long-range dependencies, and introduces novel computational strategies to improve efficiency. Its Convolutional Representation and Structured State Spaces method allows it to handle a wide range of tasks and excel at long sequences. The S4 model's potential in computation and memory usage makes it efficient and practical for a wide range of applications.

3.3.3 Convolutional Neural Network

An approach to sequence modeling, leveraging the principles of wavelet analysis to create a multiresolution memory model shown in Figure 8. This model is designed to capture both

Algorithm 2 High-Level S4Model Working Principle

```
 1: procedure S4Model($x$)
 2:     $x \leftarrow$ Encoder($x$)                    ▷ Shape: [B, L, Inputdim] to [B, L, dmodel]
 3:     $x \leftarrow$ Transpose($x$)                  ▷ Shape: [B, L, dmodel] to [B, dmodel, L]
 4:     for each Layer in S4Layers do
 5:         $z \leftarrow x$
 6:         if PreNorm then
 7:             $z \leftarrow$ Normalize($z$)
 8:         end if
 9:         $z \leftarrow$ S4DLayer($z$)
10:         $z \leftarrow$ Dropout($z$)
11:         $x \leftarrow z + x$                       ▷ Residual Connection
12:         if not PreNorm then
13:             $x \leftarrow$ Normalize($x$)
14:         end if
15:     end for
16:     $x \leftarrow$ Transpose($x$)
17:     $x \leftarrow$ Mean($x, dim = 1$)              ▷ Pooling over Sequence Length
18:     $x \leftarrow$ Decoder($x$)                    ▷ Shape: [B, dmodel] to [B, Outputdim]
19:     return $x$
20: end procedure

21: procedure S4DLayer($u$)
22:     $L \leftarrow$ GetLength($u$)
23:     $k \leftarrow$ GenerateKernel($L$)             ▷ SSM Kernel
24:     $k_f \leftarrow$ FFT($k, n = 2L$)
25:     $u_f \leftarrow$ FFT($u, n = 2L$)
26:     $y \leftarrow$ IFFT($u_f * k_f, n = 2L$)[:, : $L$]   ▷ Convolution
27:     $y \leftarrow y + u * D$                       ▷ Skip Connection
28:     $y \leftarrow$ Activation($y$)
29:     $y \leftarrow$ Dropout($y$)
30:     $y \leftarrow$ OutputLinear($y$)
31:     return $y$
32: end procedure
```

long-term and short-term dependencies in sequence data, which is particularly beneficial for tasks such as battery modeling.

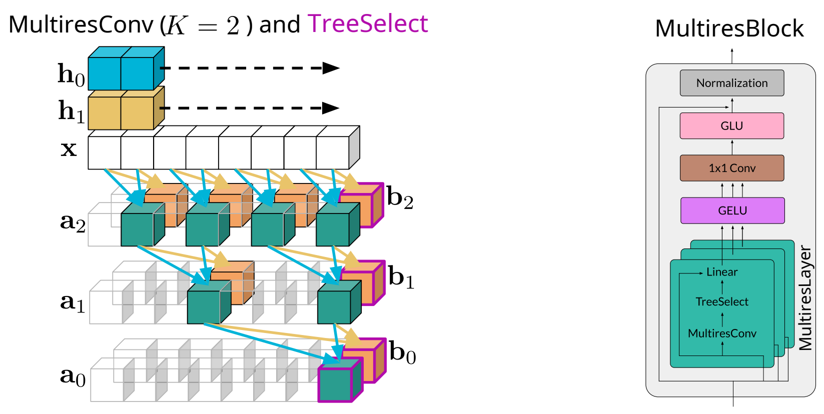

Figure 8: Left: The architecture of MultiresConv comprises a series of dilated convolutions that utilize the same filters h_0, h_1 across all levels. This illustration showcases TreeSelect with the "resolution fading" strategy, which maintains the right-most coefficient at each level of a_0, $b_{0:J-1}$, as highlighted by the magenta outlines. **Right**: This is a schematic representation of the MultiresBlock structure. Each channel of the input sequence is processed independently in the MultiresLayer. The 1x1 convolution facilitates the mixing of information across channels. Constructing a deep sequence model by stacking multiple MultiresBlocks [27].

The foundation of this approach is the Discrete Wavelet Transform (DWT), which is a mathematical tool used to decompose a signal into different frequency components. The DWT is defined by the following recursion:

$$\begin{aligned} \mathbf{a}_j(n) &\triangleq a_{j,n} = \sum_{k=0}^{K-1} \mathbf{a}_{j+1}(2n+k)\mathbf{h}_0(k) \\ \mathbf{b}_j(n) &\triangleq b_{j,n} = \sum_{k=0}^{K-1} \mathbf{a}_{j+1}(2n+k)\mathbf{h}_1(k) \end{aligned} \tag{21}$$

where $\mathbf{a}_j(n)$ and $\mathbf{b}_j(n)$ are the approximation and detail coefficients at scale j, $\mathbf{h}_0(k)$ and $\mathbf{h}_1(k)$ are the low-pass and high-pass filters, and K is the length of these filters.

The multiresolution convolution (MultiresConv) operation is then defined as:

$$\mathbf{a}_0, \mathbf{b}_{0:J-1} = MultiresConv(\mathbf{x}, \mathbf{h}_0, \mathbf{h}_1, J), \tag{22}$$

where $\mathbf{x}$ is the input signal, $\mathbf{h}_0$ and $\mathbf{h}_1$ are the low-pass and high-pass filters, and J is the number of scales. This operation can be viewed as multiple layers of dilated causal convolutions on the sequence up to a timestep.

The TreeSelect operation is then defined as:

$$\mathbf{a}_0^t, \mathbf{b}_{0:J-1}^t = MultiresConv(\mathbf{x}(0:t), \mathbf{h}_0, \mathbf{h}_1, J), \tag{23}$$

$$\mathbf{z}_t = TreeSelect((\mathbf{a}_0^t, \mathbf{b}_{0:J-1}^t), M), \tag{24}$$

where $\mathbf{a}_0^t, \mathbf{b}_{0:J-1}^t$ are selected and relevant subset of the representation coefficients at time t, and M is the memory size. The TreeSelect operation selects the most relevant scales from the multiresolution memory.

The output of the TreeSelect operation is then fed into a linear layer to produce the final output (Algorithm 3).

Algorithm 3 High-Level MulitresConv Working Principle

```
 1: procedure MulitresConvModel($x, h_0, h_1, J, M$)
 2:     for $j \leftarrow 0$ to $J$ do
 3:         $a, b \leftarrow$ MultiresConv($x, h_0, h_1$)
 4:         $x \leftarrow a$
 5:     end for
 6:     $z_t \leftarrow$ Treeselect($a, b, M$)
 7:     $output \leftarrow$ LinearLayer($z_t$)
 8:     return $output$
 9: end procedure

10: procedure MultiresConv($x, h_0, h_1$)
11:     $a \leftarrow \sum_{k=0}^{K-1} x(2n+k)h_0(k)$
12:     $b \leftarrow \sum_{k=0}^{K-1} x(2n+k)h_1(k)$
13:     return $a, b$
14: end procedure

15: procedure Treeselect($a, b, M$)
16:     $a_0^t, b_{0:J-1}^t \leftarrow$ Multiresconv($x(0:t), h_0, h_1, J$)
17:     $z_t \leftarrow$ Select($(a_0^t, b_{0:J-1}^t), M$)
18:     return $z_t$
19: end procedure
```

The proposed model is compared with the S4 algorithm, demonstrating superior performance in terms of capturing both long-term and short-term dependencies. The model is particularly well-suited to battery modeling, where it is important to capture the complex dynamics of battery behavior over time [27].

3.3.4 Recurrent Neural Networks

The Long Short-Term Memory (LSTM) network, a type of recurrent neural network, processes a sequence of input and target pairs. For each pair, the LSTM takes the new input x_i and produces an estimate for the target y_i based on all previous inputs. The state of the network, comprising a hidden vector h and a memory vector m, is determined by the past inputs. The network then calculates a series of gate values:

$$\begin{aligned} g_u &= \sigma(W_u H) \\ g_f &= \sigma(W_f H) \\ g_o &= \sigma(W_o H) \\ g_c &= \tanh(W_c H) \\ m' &= g_f \odot m + g_u \odot g_c \\ h' &= \tanh(g_o \odot m') \end{aligned} \tag{25}$$

Here, g_u, g_f, g_o, and g_c are the update, forget, output, and cell gates respectively. W_u, W_f, W_o, and W_c are the recurrent weight matrices of the network, and H is the concatenation of the new input x_i and the previous hidden vector h.

The memory vector is then updated as a combination of the forget gate applied to the previous memory and the update gate applied to the cell gate m'.

The new hidden vector is computed as the output gate applied to the updated memory vector h' [9, 28].

The Unbounded Recurrent Unit (UR-LSTM) is an improvement over the standard LSTM model. It introduces a new gate, the UR gate, which is computed in addition to the standard LSTM gates. The UR gate allows the model to control the flow of information in a more nuanced way, leading to better performance on tasks with complex dependencies and hierarchical structure. The UR gate is defined as follows:

$$\begin{aligned} r_t &= \sigma(P(x_t, h_{t-1})) \\ g_t &= r_t \cdot (1 - (1 - f_t)^2) + (1 - r_t) \cdot f_t^2 \\ c_t &= g_t \cdot c_{t-1} + (1 - g_t) \cdot u_t \end{aligned} \tag{26}$$

where f_t is the original forget gate, r_t is the refine gate, with P a parameterized linear function, u_t is the input gate, and c_t is the cell state. The gate r_t linearly interpolates between the lower band f_t^2 and the symmetric upper band $1 - (1 - f_t)^2$. In other words, the original gate f_t is the coarse-grained determinant of the effective gate g_t, while the gate r_t "refines" it [29].

The Stacked LSTM extends the LSTM model by stacking multiple LSTM layers on top of each other, with the output hidden vector h' from one LSTM layer serving as the input to the layer above. Essentially, the Stacked LSTM uses the same equations as the standard LSTM, but the input to each layer (except the first) is the hidden vector from the layer below depicted in Figure 9. The operation of a Stacked LSTM can be represented as Algorithm 4.

The 2D Grid LSTM further extends the concept of the Stacked LSTM to multiple dimensions, arranging LSTM cells in a grid with two dimensions: time and depth (Figure 9). At each time step t and layer l, the input to the LSTM is a concatenation of the hidden state from the previous time step at the same layer and the hidden state from the same time step at the previous layer. The memory vector at each time step and layer is updated based on the input and the previous memory vector, using the same gate computations

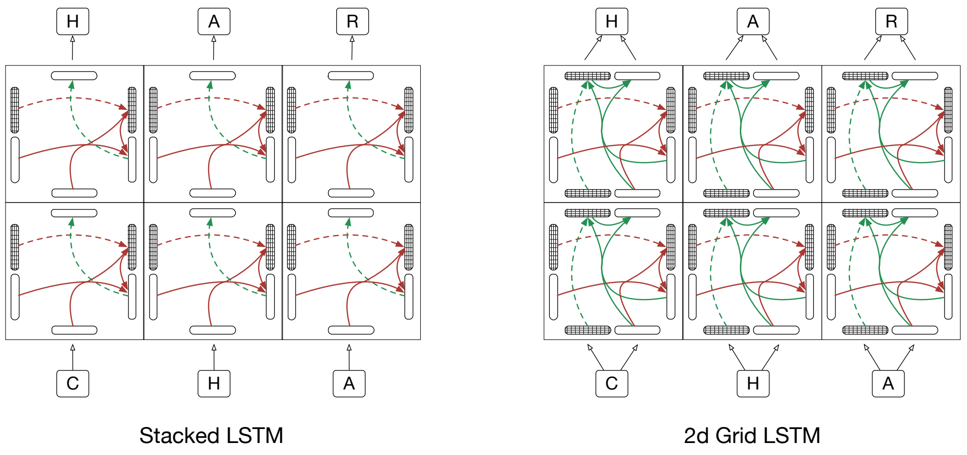

Figure 9: The application of Stacked LSTM and 2D Grid LSTM is demonstrated in character prediction. In the Grid LSTM, it's noteworthy that the signal traverses through LSTM cells (represented by shaded rectangles) in both time and depth dimensions. The dashed lines symbolize identity transformations. Unlike the standard LSTM block, which lacks a memory vector in the vertical dimension, the 2D Grid LSTM block incorporates a memory vector along the vertical axis [28].

Algorithm 4 High-Level Stacked-LSTM Working Principle

```
1: procedure StackedLSTM(src)
2:     Initialize hidden states with zeros
3:     Initialize a list of LSTM layers
4:     for each layer $i$ in the stack do
5:         $out, (h_0, c_0) = \text{LSTM}(src, (h_0, c_0))$          ▷ Update states
6:         $src = out$                                  ▷ Pass output to next layer
7:     end for
8:     $output = \text{Linear}(out)$                                ▷ Final output
9:     return $output$
10: end procedure
```

as in the standard LSTM [28]. The operation of a 2D-GridLSTM can also be represented as Algorithm 5.

Algorithm 5 High-Level 2D-GridLSTM Working Principle

```
 1: procedure GridLSTM(input, hx)
 2:     Initialize a 2D grid of LSTM cells
 3:     Initialize a list of hidden states $hxs$
 4:     for each layer $i$ in the grid do
 5:         if $hx[i]$ is None then
 6:             Initialize $hx[i]$ with zeros                          ▷ Initial states
 7:         end if
 8:         $h_{\text{time}}, h_{\text{depth}}, c_{\text{time}}, c_{\text{depth}} = hx[i]$
 9:         $h_{\text{time}}, h_{\text{depth}}, c_{\text{time}}, c_{\text{depth}} \leftarrow \text{GridCell}(input, h_{\text{time}}, h_{\text{depth}}, c_{\text{time}}, c_{\text{depth}})$
10:         Store states
11:         Store layer outputs to $hxs$
12:     end for
13:     $output = \text{Linear}(hxs)$                                ▷ Final layer cell states
14:     return $output$
15: end procedure

16: procedure GridCell($input, h_{\text{time}}, h_{\text{depth}}, c_{\text{time}}, c_{\text{depth}}$)
17:     $h_{\text{time}}, c_{\text{time}} = \text{Time-LSTM}(\text{cat}(input, h_{\text{depth}}), h_{\text{time}}, c_{\text{time}})$
18:     $h_{\text{depth}}, c_{\text{depth}} = \text{Depth-LSTM}(\text{cat}(input, h_{\text{time}}), h_{\text{depth}}, c_{\text{depth}})$
19:     return $h_{\text{time}}, h_{\text{depth}}, c_{\text{time}}, c_{\text{depth}}$
20: end procedure
```

In the context of battery modeling, the input at each time step could be a vector of features representing the current state of the battery, such as the current charge level, the temperature, the load, etc. The LSTM could be trained to predict the future state of the battery based on these features. A Stacked LSTM or 2D Grid LSTM could potentially capture more complex dependencies between these features and provide more accurate predictions. The UR-LSTM, with its refined gating mechanism, could further enhance the model's ability to capture long-term dependencies and complex patterns in the data, providing a more nuanced and accurate model for battery state prediction.

3.3.5 Transformer-Based Neural Networks

The Transformer architecture, introduced in the groundbreaking paper "Attention is All You Need", has revolutionized the field of sequence transduction [13]. The architecture's ability to capture both short-term and long-term dependencies in data makes it a powerful tool for tasks that require an understanding of temporal dynamics, such as battery modeling.

The Transformer architecture is composed of two main components: the encoder and the decoder visualized in Figure 10. Each of these components is constructed using a stack of identical layers, which allows the model to learn more complex patterns in the data. The stacking of layers is a key feature of the Transformer, enabling it to model intricate dependencies.

At the heart of the Transformer is the attention mechanism, which operates on the concept of queries, keys, and values. The query is the current time step in a time series being processed, the key is a set of time steps (the input sequence) to be compared with the query, and the value is the information to be retrieved based on the comparison. The attention mechanism calculates the similarity between the query and each key, and then uses these similarity scores to weigh the corresponding values. This results in a context vector that represents the relevant information from the entire sequence for the current value.

The attention mechanism in the Transformer is called "Scaled Dot-Product Attention". The inputs to this function are queries and keys of dimension d_k, and values of dimension d_v. The dot products of the query with all keys are computed, and divided by $\sqrt{d_k}$, and a softmax (converts scores to probabilities summing to one) function is applied to obtain the weights on the values. The attention function can be described as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{27}$$

The Transformer extends this basic attention mechanism with the concept of "Multihead Attention". Instead of applying a single attention mechanism, the Transformer applies multiple attention mechanisms (or "heads") in parallel, each with its own learned linear projections. This allows the model to capture various aspects of the input sequence at different positions. The queries, keys, and values are linearly projected h times with different learned linear projections, and then the attention function is applied in parallel to produce the output. The Multihead Attention can be described as:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O \tag{28}$$

where each head is computed as:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{29}$$

The Transformer also employs cross-attention in the "encoder-decoder attention" layers. The queries come from the previous decoder layer, and the memory keys and values come from the output of the encoder. This allows every position in the decoder to attend to all positions in the input sequence.

To account for the order of the time steps in a sequence, the Transformer uses positional encodings, which are added to the input embeddings. This allows the model to make use of the order of the words when processing the sequence. The positional encodings use sine and cosine functions of different frequencies, allowing the model to easily learn to attend by relative positions. The positional encoding for position p and dimension i can be described as:

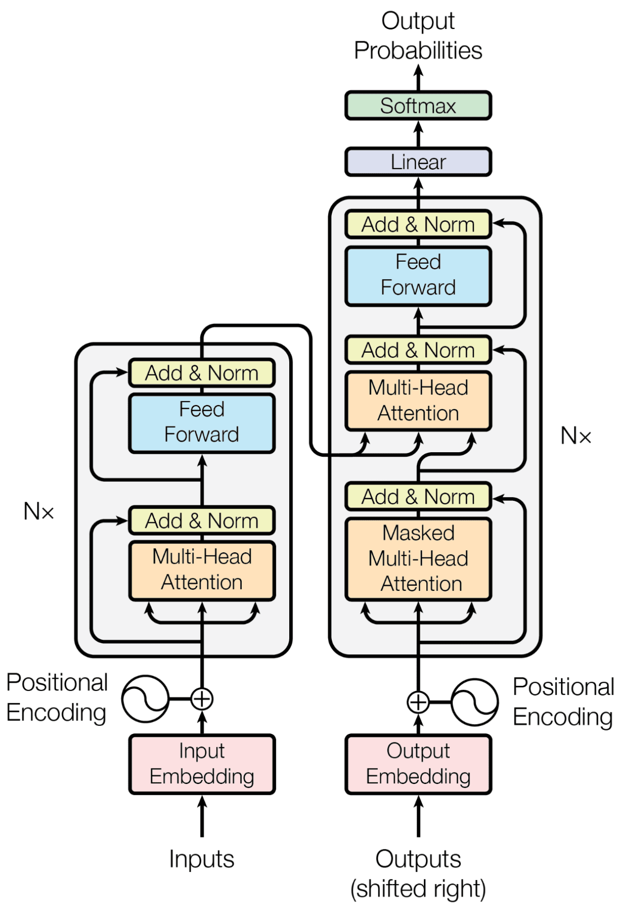

Figure 10: The **Encoder** (Left Part) processes the input sequence, which is an input tensor of shape [batch, seqlen, dim]. This input is transformed via **Input Embeddings** to shape [batch, seqlen, dmodel] and then receives **Positional Encoding** to add absolute position information. The **Nx Encoder Layers** each include **Multi-Head Attention** and **Feed Forward** networks, with **Add & Norm** for residual connection and layer normalization. The **Outputs** of the encoder serve as the latent space in an encoder-decoder setup, or are fed to a linear layer in an encoder-only setup. The **Decoder** (Right Part) generates the output sequence, with **Nx Decoder Layers** each including **Masked Multi-Head Attention**, **Multi-Head Attention**, and **Feed Forward** networks. **Add Norm** provides residual connection and layer normalization. The **Outputs** of the decoder are the final sequence output, which are transformed by a **Linear Layer** and then converted to probabilities for prediction by a **Softmax Layer** [13].

$$\begin{aligned} PE_{(p,2i)} &= \sin\left(\frac{p}{10000^{2i/d_{\text{model}}}}\right) \\ PE_{(p,2i+1)} &= \cos\left(\frac{p}{10000^{2i/d_{\text{model}}}}\right) \end{aligned} \tag{30}$$

The encoder-only architecture allows for bidirectional communication without masking, making it suitable for tasks where the entire context of the input sequence is needed for each value. The decoder, on the other hand, is autoregressive and uses causal masking to ensure that the prediction for a certain position can depend only on the known outputs at positions less than the current one. This enforces a certain directionality or causality in the model.

The latent state in the Transformer architecture is the output of the encoder, which is used as input to the decoder cross-attention. This state captures the information from the input sequence and is used to generate the output sequence. This makes the encoder-decoder architecture particularly useful for tasks where the input and output sequences are related but not directly aligned [13].

In conclusion, the Transformer's ability to model global dependencies and its flexibility in handling different sequence lengths make it a good candidate for transfer-learning and meta-learning. These properties also make it a promising approach for modeling new battery chemistries, where only limited data may be available. The Transformer's ability to capture complex patterns in data can be particularly useful for tasks like predicting the remaining SoC of a battery based on historical performance data. The autoregressive and causal nature of the model, in particular, make it a promising approach for tasks that require an understanding of temporal dynamics.

3.4 Overview of Software Architecture

The software architecture is designed to automate and streamline the process of battery model training and evaluation. It commences with the generation of diverse current profiles, simulating a wide range of battery operating conditions. The resulting data is then used to train deep learning models. The architecture incorporates automatic saving and loading of model state dictionaries and hyperparameters, ensuring reproducibility and facilitating iterative model development. Additionally, it supports automatic hyper-parameter optimization runs, further improving model performance. The entire process is designed to be efficient and user-friendly, minimizing manual intervention and maximizing the effectiveness of the training process.

3.4.1 Profile Generation and Data Extraction

The provided algorithms serve as a comprehensive simulation framework for generating diverse and extensive datasets, which are crucial for training robust and generalizable deep learning models. The framework leverages a validated ECM of a battery encapsulated in an FMU. The FMU is a standard for model exchange and co-simulation, which allows for seamless integration of the battery model into the simulation environment.

The process begins with the generation of various current profiles, including constant, linear, pulse, sinusoidal, drive, superimposed, and concatenated profiles. These profiles represent the current flowing into or out of the battery over time. Each profile is generated with random parameters for duration and amplitude, introducing variability into the simulations. This variability is key in ensuring that the resulting dataset covers a wide range of operating conditions, which is crucial for training a deep learning model that can generalize well to out-of-distribution data (Algorithm 6).

Algorithm 6 Current Profile Generation

```
1: $profileTypes \leftarrow$ [const, lin, pulse, sin, drive, superimpose, concat]
2: $profiles \leftarrow$ empty list
3: for $type$ in $profileTypes$ do
4:     $profile \leftarrow$ GenerateProfile($type$)
5:     Append $profile$ to $profiles$
6: end for
7: FinalChecks($profiles$)

8: procedure GenerateProfile(type)
9:     $duration \leftarrow$ random value
10:    $amplitude \leftarrow$ random value
11:    $profile \leftarrow$ generate profile of $type$ with $duration$ and $amplitude$
12:    if capacity of $profile$ exceeds maximum or minimum then
13:        Adjust pulse sign and retry
14:    end if
15:    Pad $profile$ with zeros
16:    return $profile$
17: end procedure

18: procedure FinalChecks(profiles)
19:    for $profile$ in $profiles$ do
20:        Check boundary conditions
21:    end for
22: end procedure
```

The generated current profiles are then used as input for the battery simulation. For each current profile, the simulation is run using the FMU, which contains the ECM battery model. The simulation takes into account the initial State-of-Charge (SOC), State-of-Health (SOH), thermal resistances, and ambient temperature. These factors are crucial in influencing battery behavior and are therefore important features to include in the dataset for the deep learning model (Algorithm 7).

The simulation results are consolidated into a data array, which, along with the corresponding headers, is saved. The dataset, encompassing a broad range of temperatures, currents, voltages, and State-of-Charge values, serves as a rich resource for training deep learning models to predict battery behavior.

Algorithm 7 FMU Battery Simulation

```
1: $fmu, currFile$ ← load FMU and current profile file
2: $ivar$ ← initialize initial values for SOC, SOH, thermal resistances, and ambient temperature
3: $head$ ← define headers for data set
4: for each $currProfile$ in $currFile$ in parallel do
5:     SimulateAndStore($currProfile$)
6: end for
7: $newData$ ← concatenate $head$ and $data$
8: save $newData$ to file

9: procedure SimulateAndStore($currProfile$)
10:     $input$ ← create input array from $t$ and $currProfile$
11:     $result$ ← run simulation with $fmu$, using $input$ and $ivar$
12:     $resultData$ ← extract results from $result$
13:     return $resultData$
14: end procedure
```

In essence, this code offers a robust framework for extracting battery behavior, generating diverse current profiles, and creating exhaustive datasets for deep learning model training. It stands as a potent tool for exploring and comprehending battery behavior, as well as for developing predictive models of battery performance.

3.4.2 General Training and Inference

The process of battery modeling using deep learning typically commences with the configuration of hyperparameters. These parameters, which govern various facets of the model and the training process, are generally sourced from an external file. The system is designed to accommodate adjustments or fine-tuning of these parameters, providing a mechanism for customization based on specific experimental conditions. After the hyperparameter setup, the focus shifts to data preparation. The raw battery data is subjected to a series of preprocessing steps, potentially including standardizing and partitioning into distinct subsets for training, validation, and testing. This preparation of data is a prerequisite for effective training and subsequent evaluation of the model. The selection of the appropriate deep learning model is critical in the process. With the model selected, the training process is initiated. This involves several steps, including the computation of initial losses, the configuration of the optimizer and the scheduler, and the setup of various monitoring mechanisms. These mechanisms, often implemented as callbacks, offer real-time insights into the training process and can include functionalities such as checkpoint saving, early stopping, and learning rate monitoring. During the training process, the model learns from the training data, and its performance is concurrently evaluated using the validation data. The best score achieved during the training process is recorded. In the final stages of the process, the trained model is subjected to a rigorous test. If checkpoints were saved during training, these can be loaded to restore the model to its optimal state. The model is then evaluated using the test data set. Upon successful completion of the experiment, a success indicator is set to true, and the final loss, indicative of the model's performance, is returned and shown in Algorithm 8.

Algorithm 8 Pseudocode for Combined Procedures

```
1: procedure RunExperiment
2:     $hyperparams$ ← Load hyperparameters from yaml file
3:     $trainData, valData, testData$ ← Setup data
4:     $baseModel$ ← setup specific model with $hyperparams$
5:     $trainer, task$ ← InitExperiment($baseModel, hyperparams$)
6:     Fit $trainer$ with $task$, $trainData$, and $validationData$
7:     $loss$ ← Get best score from $trainer$'s callbacks
8:     Load checkpoint and weights for best configuration
9:     Testset evaluation
10:    return $loss$
11: end procedure

12: procedure InitExperiment($baseModel$, $kwargs$)
13:    $task$ ← Initialize with $baseModel$, loss, optimizer, scheduler, etc.
14:    Setup logging
15:    Configure optimizer and scheduler
16:    Define training, validation, and testing procedures
17:    Save attention weights (optional)
18:    Set model parameters to the optimizer
19:    Load model from checkpoint (optional)
20:    Save hyperparameters to yaml file
21:    Setup checkpoint callback, early stopping and learning rate monitoring
22:    $trainer$ ← Initialize with callbacks, max epochs, logger, and devices
23:    return $trainer, task$
24: end procedure
```

3.4.3 Model Parameter Optimization

The process commences with the definition of a configuration space, which is combined with a general hyperparameter file. This space encapsulates possible combinations of hyperparameters that the model will be tuned on.

Subsequently, a Scenario is established, incorporating the configuration space and other parameters. The Scenario provides a comprehensive overview of the information required to execute the optimization process. An initial design is procured from the MFFacade, a facade for various types of multi-fidelity model optimization algorithms. This design serves as the starting point for the optimization process. The intensifier, set as Hyperband in this instance, is a component that determines the configurations to evaluate next based on their performance [30]. Hyperband is an intensification strategy that strikes a balance between exploration and exploitation by allocating resources to configurations by their performance. In combination with the MFFacade it is the closest implementation to Bayesian Optimization Hyperband (BOHB) [31]. The SMAC optimizer is then instantiated, incorporating the scenario, the training function, the initial design, and the intensifier. The optimizer navigates the configuration space to identify the configuration that minimizes the loss returned by the training function. The incumbent configuration, representing the optimal configuration identified thus far, is obtained by invoking the optimization method of the SMAC optimizer. The training function is executed with the seed, budget, and configuration space as parameters. The final stage of the process involves the validation of the default configuration and the incumbent. This could entail a comparison of the performance of the incumbent with the default configuration. The training procedure encapsulates the function that trains and evaluates a model given a configuration. It initiates by loading the hyperparameters from the configuration space and a yaml file. The training and validation data loaders are then established, managing the batching and shuffling of the data. Following the setup of the model using the loaded hyperparameters, the training process is initiated. Upon completion of the training, the model is subjected to testing, likely on a separate test set. The loss, indicative of the model's performance, is returned. This loss is utilized by the SMAC optimizer to assess the performance of the configuration as shown in Algorithm 9. The objective of the optimization process is to identify the configuration that minimizes this loss [32].

Algorithm 9 HPO run with SMAC3

```
1: procedure ARCHITECTURE CONFIG
2:     Define Scenario and Configuration Space
3:     Get initial design from MFFacade
4:     Define intensifier as Hyperband
5:     Define smac as MFFacade with scenario, training, initial design, intensifier
6:     $incumbent \leftarrow$ smac.optimize(TRAINING($seed, budget, configspace$))
7:     Validate default configuration and incumbent
8: end procedure

9: procedure TRAINING($seed, budget, configspace$)
10:    Load hyperparameters from Configuration Space and yaml file
11:    Setup data
12:    Setup model
13:    Setup and start training
14:    Setup and start testing
15:    Return loss
16: end procedure
```

4 Results

Sequence-to-sequence modeling, essential for battery performance prediction, is evaluated across different architectures. The efficiency and appropriateness of these architectures for the task become evident as the study progresses. Notably, the transformer model, due to its decoder component, exhibits superior predictive performance, emphasizing the importance of architectural design in such tasks.

Hyperparameter tuning is also discussed, emphasizing its role in achieving a balance between model complexity and regularization. This balance is crucial to avoid overfitting or underfitting, ensuring effective pattern recognition.

Further analysis of the transformer model reveals the significance of attention mechanisms. These mechanisms emphasize the importance of selecting appropriate model parameters. The model's extensive temporal focus enhances its predictive ability, underscoring the role of attention mechanisms in battery modeling.

The study also explores modifications and techniques beyond basic architecture, aiming to increase the model's capability. While significant performance improvements are not always achieved, the research offers valuable insights into the relationship between data augmentation and output quality.

Additionally, the performance of various architectures is assessed using real-world data. This assessment and subsequent fine-tuning offer a practical evaluation of the architectures' resilience and efficiency, leading to the determination of the best configuration for the final battery model.

4.1 Baseline Model Analysis

An in-depth evaluation of various neural model architectures was undertaken to determine their proficiency in modeling complex battery data. The main goal was to identify the architecture best suited for handling the complexities of synthetic battery profiles.

A subset of the dataset, rather than the full set, was used to produce surrogate loss and models. This method was chosen based on the premise that the outcomes would apply to the entire dataset, thus saving computational time and resources.

These methods lead to a streamlined neural architecture search. This search focuses on surrogate models, believed to be indicative of the complete dataset, ensuring a methodical and evidence-based process to pinpoint the most effective model structures, as shown in Figure 11.

The y-axis, displayed on a logarithmic scale, outlines loss values, offering a numerical assessment of model performance. The x-axis lists various neural model architectures, each representing a distinct analytical approach. Two separate mean squared error (MSE)

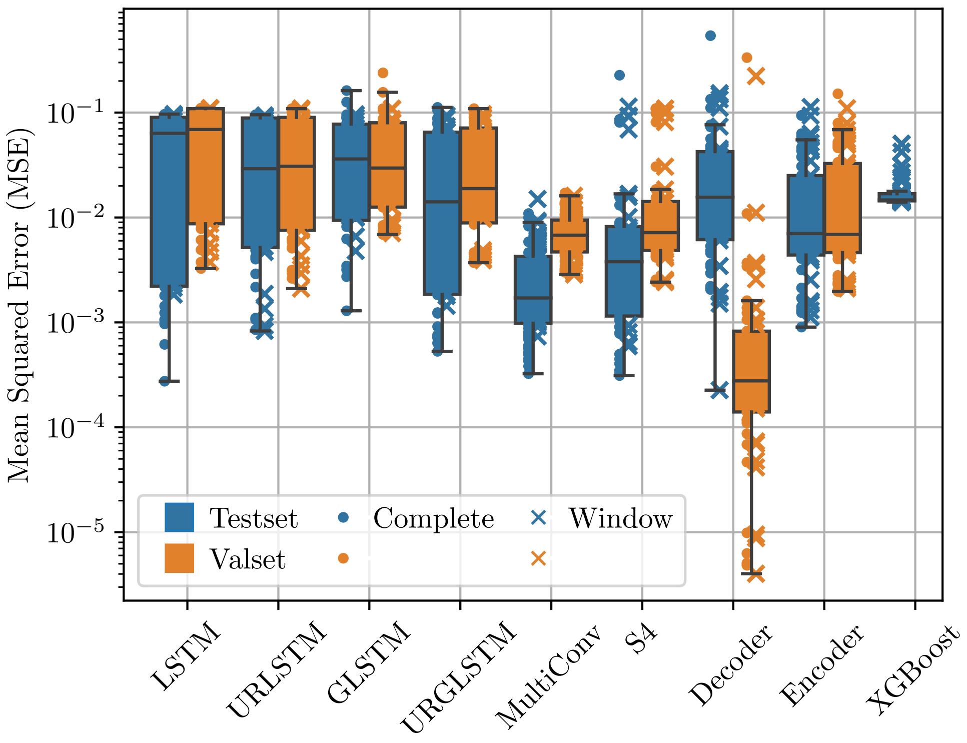

Figure 11: Analyzing neural architectures with synthetic battery profiles, each model's performance, inclusive of varied budgets, is denoted by its loss values. Crosses represent models optimized for validation loss (Window), where the model is evaluated on a single "window" length, explicitly defined as the model's input sequence length. Dots symbolize models optimized for test loss (Complete), where predictions merge multiple such windows across the entire sequence. The marker distribution indicates stability and parameter sensitivity. The XGBoost model's lack of validation loss results is due to a distinct training method.

categories are applied, corresponding to the test and validation datasets. The test dataset, which uses multiple windows (where a window is equivalent to the model's input sequence length) and is a subset of the full sequence, employs a sliding window approach. This method sequentially produces predictions based on the model's fixed window length, and then combines these results. Conversely, the validation dataset provides a more focused assessment, evaluating the model over a single window. These evaluation methods highlight the model's challenges, from the long-term predictions required by the test set to the shorter sequences of the validation set. Direct relationships between these two evaluation methods are not always evident, as shown by optimization cycles.

The visual representation includes specific markers, dots, and crosses, which indicate the optimization approach used. Dots symbolize models fine-tuned using SMAC3 in a high-fidelity environment, where an epoch-based budget is set, and optimization is based on test loss [32]. On the other hand, crosses represent models with an optimization focus on validation loss. These optimization goals lead to subtle differences in model performance across datasets.

The analysis of architecture-specific results reveals distinct patterns:

Decoder: A wide range of loss values for the Decoder model indicates its sensitivity to hyperparameter changes. With careful tuning, it demonstrates strong predictive performance. The Decoder effectively navigates different prediction horizons, especially in shorter sequences, as evidenced by the reduced validation loss.

S4 and MultiConv: Both architectures show broad loss distributions, emphasizing their pronounced sensitivity to parameter changes. This highlights the importance of hyperparameter tuning, given the dataset's complexity.

LSTM and UR-LSTM: These models benefit from the incorporation of multiple LSTM layers, enabling the extraction of hierarchical temporal features. The UR-LSTM, with its advanced gating mechanism, offers an improved method for information flow. However, their performance distribution suggests that while they can capture complex temporal patterns, optimal performance requires accurate parameter adjustments.

GLSTM and URGLSTM: The underwhelming performance of these models can be linked to a potential architectural misalignment. The GLSTM is designed for spatial data processing, but the dataset is mainly temporal, which might cause inefficiencies. The URGLSTM, being a grid-based version of the UR-LSTM, may have performance variations due to the grid mechanism's redundancy in a primarily temporal setting.

XGBoost: Although XGBoost is versatile across various use cases, its alignment with the sequence-to-sequence structure of this dataset seems limited, as shown by its higher loss values. The lack of validation loss results from a distinct training approach that doesn't include evaluations after each training iteration.

Recognizing outliers emphasizes the importance of hyperparameter settings, highlighting the delicate balance between optimal and suboptimal model outcomes.

Figure 12 offers a visual comparison of these architectures as they predict battery State-of-Charge (SoC) using a synthetic battery profile.

In the graph, the black line represents the ground truth, while individual traces illustrate each model's time-based predictions. The Decoder model stands out for its alignment with the ground truth throughout, reflecting its ability to interpret the intricate patterns of the synthetic battery dataset.

Other models, such as LSTMs, S4, and MultiConv, show oscillatory predictions. These oscillations, more evident in the magnified inset, suggest difficulties in the optimization process. Gated LSTM models, in particular, tend to produce outliers in dynamic regions.

When comparing these visual oscillations with prior MSE loss values, it's evident that the MSE loss is less sensitive to outliers. As a result, models like the Decoder, which produce consistent predictions, might have MSE values similar to those of oscillating models that accurately predict certain linear segments. However, the practical implications of their predictions can differ significantly.

The absence of XGBoost in the graphical comparison underscores its challenges with sequence-to-sequence tasks, suggesting that while XGBoost excels with structured data, it struggles with sequential datasets.

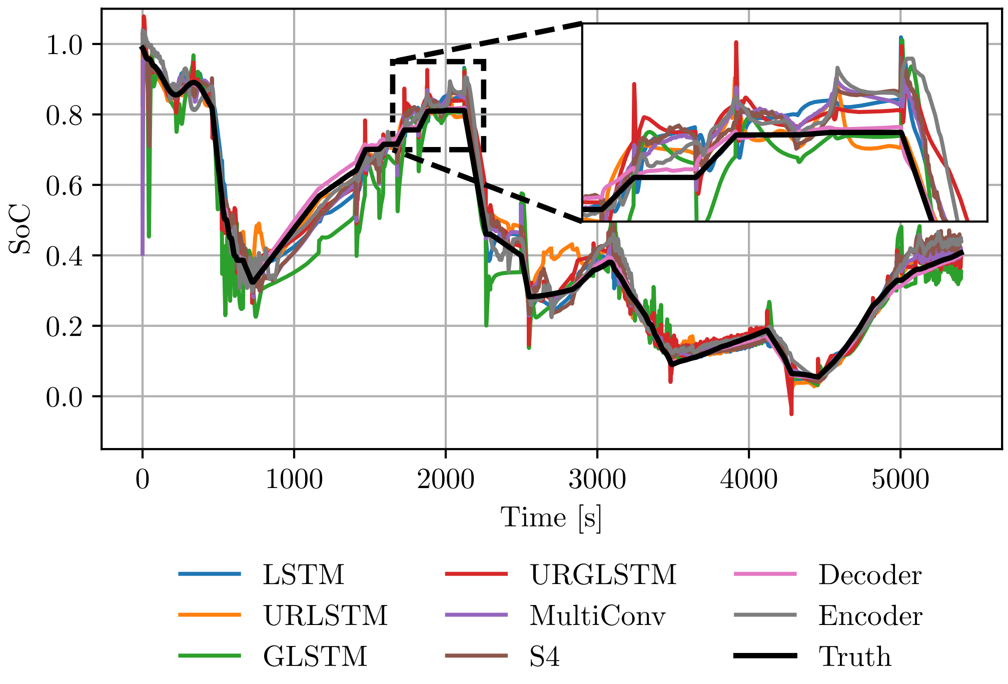

Figure 12: Graphical comparison of neural architectures predicting battery State-of-Charge (SoC) over time. The black trace indicates the ground truth, with colored traces representing predictions from different models. The Decoder model shows significant alignment with the ground truth. Conversely, models like LSTM and MultiConv display oscillatory predictions. The inset magnifies these oscillations, emphasizing the difficulties some models encounter in understanding the dataset's subtleties.

Figure 13 provides a detailed comparison of the top-performing models. It's important to clarify that the test loss in this evaluation is based on the entire sequence length using concatenated windows, which differs from previous evaluations that used multiple window lengths over a section of the complete sequence.

LSTM: The LSTM architecture is designed to capture patterns over long sequences. Its layered design allows for the extraction of temporal dynamics. The memory cells in LSTMs enable them to store and access information over extended periods, making them suitable for sequential datasets. However, the complexity of this architecture can lead to extended training and inference times due to its sequential nature, which limits parallel processing.

MultiConv: The MultiConv design uses multiple convolutional layers, each designed to detect different resolutions in the input data. Due to the numerous convolutional kernels and weights, these architectures require more computational resources, explaining the increased memory and time requirements. However, their ability to detect spatial data hierarchies often results in better performance metrics, evidenced by their test loss value of 8.84×10^{-4}.

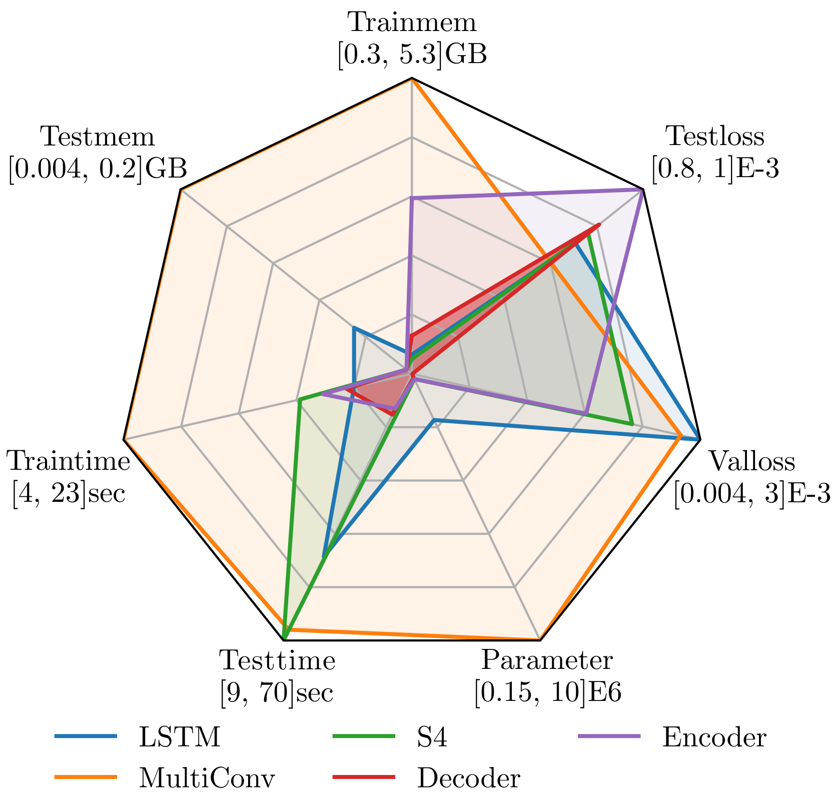

Figure 13: Radar plot showing the performance metrics of leading neural architectures [min, max]. The plot includes metrics like memory usage, training time, and parameter count. The test loss (MSE), calculated over the full sequence with concatenated windows, provides insights into each model's sequence prediction capability.

S4: The S4 model is structured to handle long sequences by organizing state spaces systematically. This design allows it to detect long-term data dependencies without excessive computational demands, leading to its moderate memory and time metrics.

Decoder: Transformer models, especially the decoder-focused variant, are known for their parallel processing capabilities, allowing them to handle different sequence positions simultaneously. This parallelism explains the time efficiency of such a sophisticated architecture. The self-attention mechanism in the decoder enables it to focus on various parts of the input sequence, effectively detecting long data dependencies. Its performance is further supported by a test loss of 1.19×10^{-3}, a reduced validation loss of 4.02×10^{-6}, and a parameter count of 150×10^{3}.

Encoder: Similar to the decoder, the encoder-only Transformer variant also benefits from parallel processing. However, without the auto-regressive context of the decoder, its ability to interpret sequential data may be limited, as seen in its performance metrics.

In summary, while the MultiConv model exhibits the lowest test loss, its oscillatory predictions might pose challenges in practical applications. The Decoder architecture, given its consistent results and with proper parameter adjustments, stands out as the most suitable for battery state estimation.

4.2 Refinement of Transformer Architecture

After evaluating multiple model architectures, the Transformer structure was identified as a promising candidate, especially when compared to baseline models. While the encoder-only variant was previously analyzed and found lacking, a complete assessment of the full Transformer on the same dataset, encompassing both encoder and decoder, was yet to be conducted. Parameters associated with the decoder played a key role in defining the configuration for the entire Transformer. These parameters encompassed aspects like model dimensions and the number of attention heads. An in-depth analysis was conducted on different training methods for the encoder, experimenting with various masking techniques. This included distorting the input to ensure the model predicted only the output values and using forward prediction based on causal masking [33]. Alongside the decoder, the potential of cross-attention causal masking was investigated. Adjustments to parameters, such as the number of layers, were also examined. Results showed that the performance of the full Transformer was comparable to the decoder-only variant. However, the decoder-only model, with its simpler training process and reduced parameter space, emerged as a more streamlined choice for further development.

Hyperparameter Tuning: Hyperparameter tuning is fundamental in machine learning, essential for determining configurations that align with specific dataset attributes, as illustrated in Figure 14. In this research, hyperparameter selection was limited to the most impactful ones, with choices for model dimensions and attention heads informed by previous studies.

Throughout the experiments, a consistent batch size of 32 was maintained. The optimizer, scheduler, and seed were kept constant, with the seed reapplied for each run to ensure consistent results. The dropout rate, a technique to mitigate overfitting, involves randomly nullifying a fraction of units during training, promoting model robustness [34]. The weight decay parameter, vital for the AdamW optimization algorithms, acts as a regularization tool [35, 36]. Proper tuning of this parameter is crucial to balance overfitting and underfitting. The Tmax hyperparameter, linked with cosine annealing schedulers, adjusts the learning rate, potentially helping to avoid local minima [37]. Optimal results were observed with minimal dropout and weight decay values, suggesting that excessive regularization might not be beneficial for smaller models and datasets. A moderate engagement of cosine annealing, as indicated by Tmax, was also advantageous.

Model depth, represented by the number of layers, is a central topic in deep learning. Although deeper models can capture intricate patterns, results indicated that a model with a moderate number of layers was best suited for the given data. The sequence length, crucial for predictions, defines the model's temporal analysis scope. A median sequence length was found optimal, suggesting that altering this parameter might not necessarily improve model precision. Given a restricted evaluation budget, the optimization aimed to achieve the lowest loss within set boundaries.

The learning rate, essential for gradient-based optimization, determines the step size during loss minimization. Precise calibration of this parameter is vital to prevent issues like oscillations or slow convergence.

For a thorough exploration of the hyperparameter space, SMAC3, known for its Bayesian optimization capabilities, was utilized within a structured multi-fidelity framework [32]. This procedure was limited to a third of the dataset, with a maximum epoch count of 15. The results in Figure 14 emphasizes the significant impact of each hyperparameter on overall model performance.

Final Hyperparameter Selection: Hyperparameters are chosen through a rigorous combination of empirical evaluations and theoretical insights. For this research, the final selection was largely influenced by the partial dependencies assessment, with certain modifications made to enhance model efficacy. Specifically, the sequence length and the number of layers were increased, anticipating future dataset expansion and potential meta-learning applications. In such contexts, maximizing the parameter count is essential to fully leverage the data. Regularization and training parameters were directly sourced from the previous analysis, striking a balance between model intricacy and generalizability. The batch size was also modified to maximize GPU efficiency. Table 1 details the final hyperparameter selection.

In summary, the selected hyperparameter set reflects careful consideration and design, aiming to extract optimal performance from the model given the dataset's specific challenges and traits.

Decoder Attention Mechanism: Attention mechanisms in the Transformer architecture plays a pivotal role, in enhancing the model's ability to identify and prioritize significant portions of input data, especially in sequence-to-sequence prediction tasks. This mechanism defines the temporal scope from which the model extracts information for inference, as outlined by the parameters in Table 1.

Figure 15 illustrates the decoder's attention distribution for this task. Contrary to initial assumptions, the model's attention isn't solely focused on the sequence's end. Instead, attention spans the entire sequence, indicating that the model considers a broad temporal range of data points when making predictions.

The model exhibits increased attention to areas with significant input changes and varies across layers. This suggests that the Transformer's decoder is proficient at recognizing and prioritizing both recent and historical data, especially during significant input variations. This aligns with insights from the hyperparameter tuning phase, reinforcing the model's predictive reliability in battery modeling tasks.

Exploring Advanced Architectures: Beyond conventional architecture, various novel techniques were explored. One approach integrated convolutional layers, aiming to replace standard positional encoding, with the goal of better capturing local input dependencies. However, the expected improvements were not realized. Another method applied the Short-Time Fourier Transform (STFT) to input signals. After transformation, these signals passed through a linear layer to match the necessary model dimensions and were then integrated into the model similar to positional encoding, termed "dynamic encoding". This approach also did not yield significant enhancements. These experiments

Hyperparameter	**Value**
Dataset Range	Standardized [0,1]
Number Sequences	3000 (3 different ambient temperatures)
Full Sequence Length	5400
Data Dimensions	4 (Current, Voltage, Temperature, SoC)
Optimizer	AdamW
Learning Rate (Lr)	0.002
Weight Decay	0.001
Scheduler	CosineAnnealing
Tmax	15
Batch Size	128
Loss Function	Mean Squared Error
Model Dimensions	64
Sequence Window Length (seqlen)	350
Number of Heads (nheads)	16
Layers	8
Dropout	0.05
Total Parameters	350×10^3

Table 1: Final hyperparameter configuration derived from a combination of empirical evaluations and theoretical considerations. This configuration is informed by partial dependencies evaluation with modifications to cater to the potential dataset expansion and meta-learning scenarios. The table showcases the balance achieved between model complexity and generalization, with parameters optimized for computational efficiency and GPU utilization.

highlight that simply adding or diversifying data and input features doesn't guarantee improved model performance.

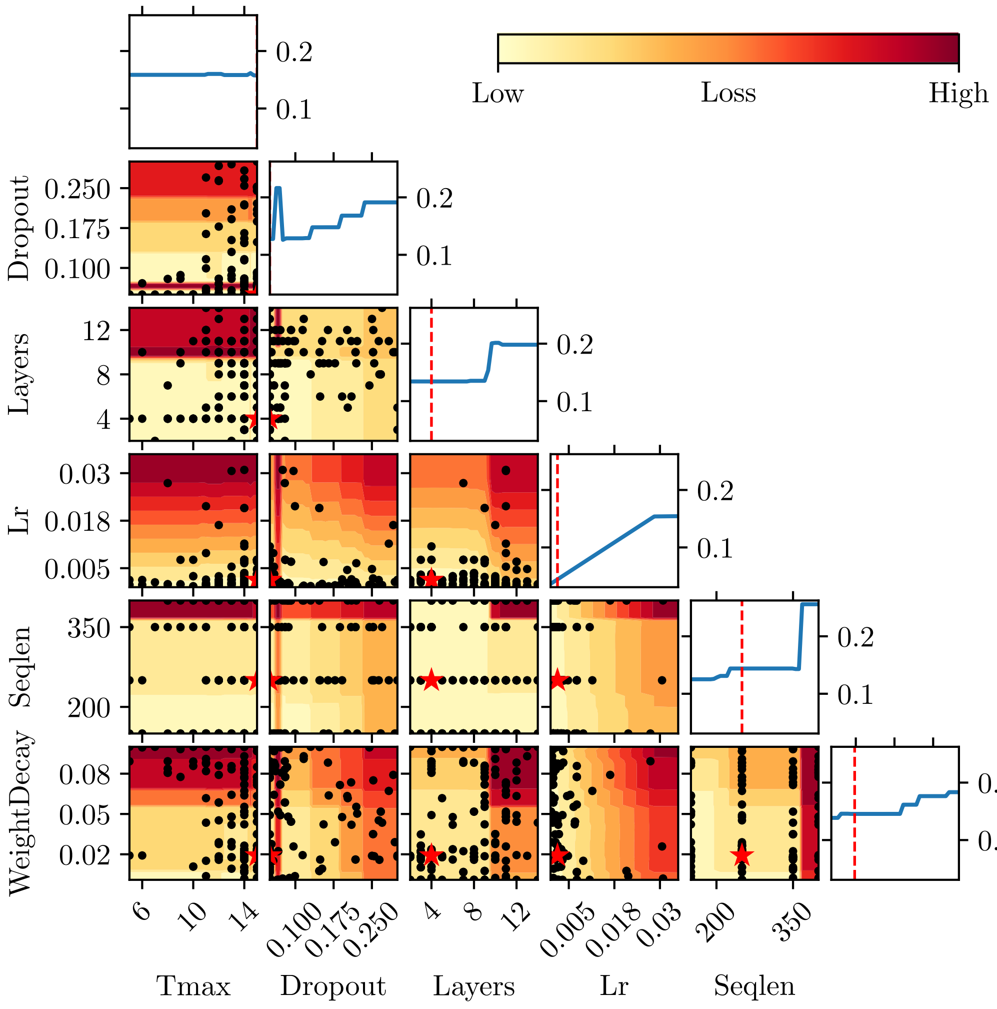

Figure 14: Detailed visualization of the hyperparameter tuning landscapes using SMAC3 in battery modeling. This representation emphasizes the interdependencies between parameters, highlighting the dynamics that contribute to optimal model performance.

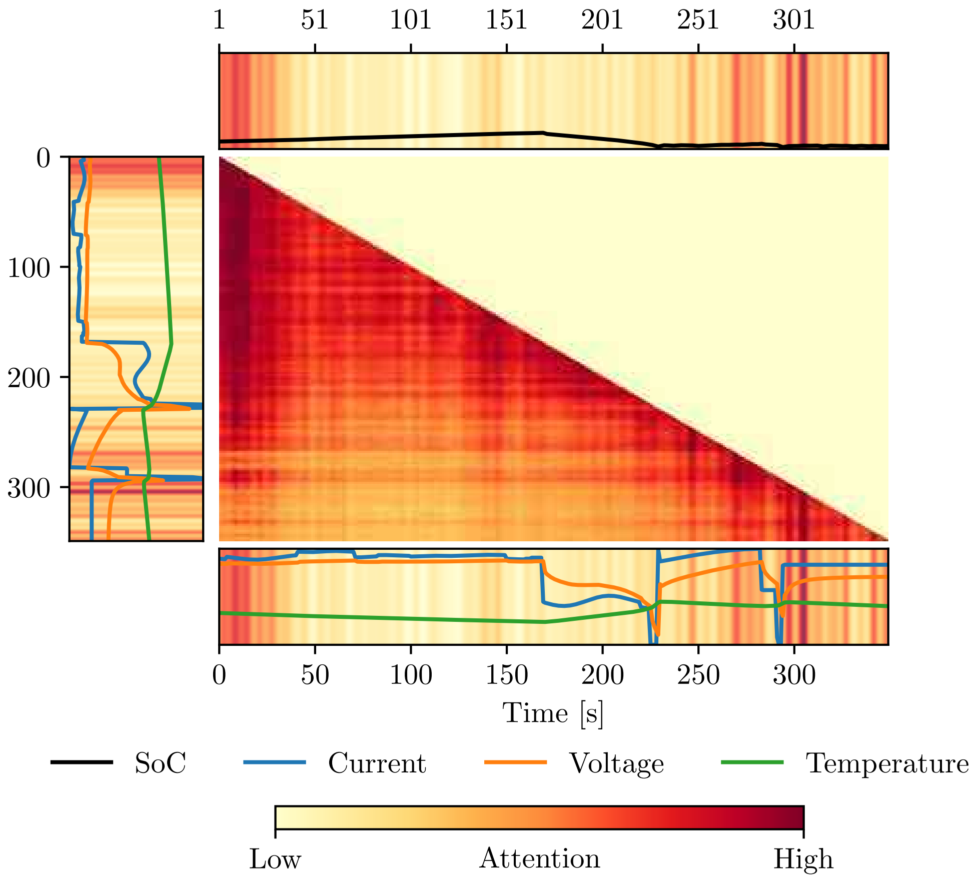

Figure 15: Decoder Attention Distribution across sequence length for the fourth layer. This visualization demonstrates the dispersion of the decoder's attention (center) throughout the input sequence (bottom, left) when predicting SoC (top). Rather than a unilateral focus on terminal data points, the attention is uniformly distributed, with pronounced peaks at regions corresponding to significant input variations. The pattern accentuates the Transformer decoder's ability to assimilate both recent and historical data, especially during pivotal input shifts, to enhance predictive fidelity in battery modeling tasks.

4.3 Final Assessment

The Decoder model's efficacy in battery modeling was assessed using a randomized current profile tailored to real-world applications. Specifically designed for the Nickel Manganese Cobalt (NMC) cell, this profile emulates practical scenarios, underscoring the model's applicability. In contrast, the Lithium Iron Phosphate (LFP) models were refined and evaluated using only laboratory data. Four distinct models were developed for each chemistry.

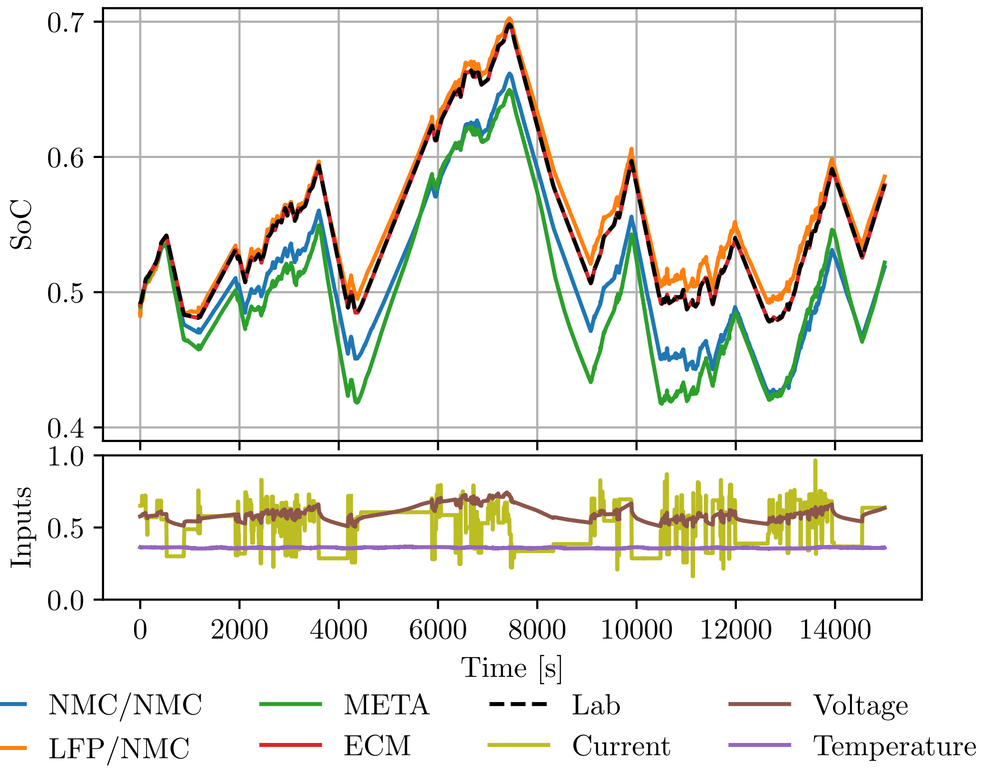

Figure 16: Comparison of model performances for NMC chemistry. The models include: one trained solely on the synthetic NMC dataset (NMC/NMC), one initially trained on the synthetic LFP dataset and fine-tuned on NMC data (LFP/NMC), one trained using a meta-learning approach incorporating both synthetic datasets (META), and the Equivalent Circuit Model (ECM). The performance of each model is visualized against the ground truth, which employs coulomb counting to estimate the SoC.

For NMC, the first model was trained on the synthetic NMC dataset and refined with NMC lab data, which also provided parameters for the Equivalent Circuit Model (ECM). The second model began its training on the synthetic LFP dataset, showcasing cross-chemistry adaptability, and was later refined using a mix of NMC lab and dynamic dataset. The third model employed a meta-learning strategy, initially trained on both synthetic datasets (NMC and LFP) and then refined with NMC lab and a portion of the dynamic dataset. The fourth model was based on the ECM framework, closely mirroring lab measurements. The NMC models were tested on unseen segments of the dynamic dataset, as shown in Figure 16.

The LFP models followed a similar structure, as depicted in Figure 17. The first model was trained on the LFP synthetic dataset and refined with LFP lab data, which also facilitated ECM parameter extraction. The second model started on the NMC synthetic dataset and was later refined with LFP lab data, highlighting model transferability. The third model utilized meta-learning, with initial training on both synthetic datasets and subsequent refinement on LFP lab data. The fourth model was the LFP's ECM.

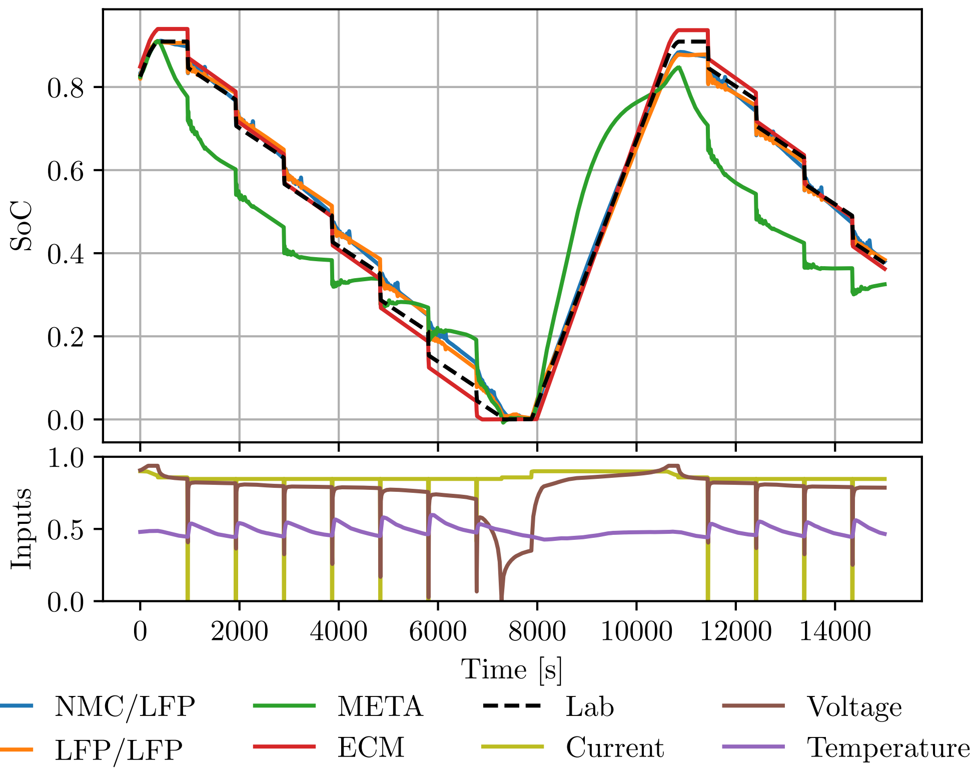

Figure 17: Comparative analysis of model performances for LFP chemistry. The models include: one trained solely on the synthetic LFP dataset (LFP/LFP), one initially trained on the synthetic NMC dataset and fine-tuned on LFP data (NMC/LFP), one trained using a meta-learning approach incorporating both synthetic datasets (META), and the Equivalent Circuit Model (ECM). The performance of each model is visualized against the ground truth, which utilizes coulomb counting to estimate the SoC.

All models, regardless of training variations, demonstrated a robust grasp of the data structure. Models initially trained on the LFP synthetic dataset and later refined for specific chemistries showed superior performance.

Dataset	**Model**	**MSE**	**MAE**
LFP	LFP/LFP	<u>7.3e-4</u>	<u>2.0e-2</u>
	NMC/LFP	5.8e-4	1.9e-2
	META	1.9e-2	1.2e-1
	ECM	**3.5e-4**	**1.5e-2**
NMC	NMC/NMC	1.6e-3	3.8e-2
	LFP/NMC	<u>8.3e-5</u>	<u>8.0e-3</u>
	META	2.7e-3	4.9e-2
	ECM	**1.0e-7**	**2.9e-4**
NMC 10°C	LFP/NMC	3.0e-4	1.5e-2
NMC 20°C		1.6e-1	3.6e-1
NMC 40°C		1.6e-1	3.6e-1

Table 2: Quantitative metrics for various models. The table displays the MSE and MAE for each model trained on NMC and LFP datasets, benchmarked against the ground truth derived from coulomb counting. Models include those trained on synthetic datasets of the same chemistry (NMC/NMC, LFP/LFP), those trained on synthetic data of a different chemistry and refined on the target chemistry (NMC/LFP, LFP/NMC), meta-learning models (META), and the Equivalent Circuit Model (ECM). The ECM provides values aligned with lab measurements, offering a quantitative perspective on each model's accuracy in estimating the State-of-Charge (SoC)

It's vital to recognize that the ECM model's SoC estimation aligns with lab measurements, both employing coulomb counting—a method integrating current over time—to estimate SoC. This straightforward method is effective, providing a reliable comparison baseline for the various models. Even with its inherent constraints, like potential drift in current measurements due to temperature fluctuations, the SoC discrepancy in the tested scenarios remains minimal, affirming its validity as a benchmark.

The Decoder model's performance was quantitatively validated using metrics such as Mean Squared Error (MSE) and Mean Absolute Error (MAE), as detailed in Table 2. For the LFP dataset, the model trained and refined on LFP data exhibited the lowest errors. Conversely, for the NMC dataset, the model initially trained on LFP and subsequently refined on NMC outperformed the rest.

The top-performing model for NMC was further assessed across various ambient temperatures, as depicted in Figure 18. Even though this data was part of the training set within a specific window length, the model's predictions closely matched the ground truth across multiple windows. This consistency underscores the model's capability to manage temperature fluctuations, emphasizing its practical relevance.

It's important to note that ECMs are designed for specific cell types, with their capacities inherently integrated. This design makes them less flexible for transfer learning. In this study, two chemistries, NCM and LFP, were examined, each with distinct capacities: 64Ah and 2.6Ah. Transfer learning attempts using these ECMs would produce misaligned outcomes, as the State-of-Charge depends on this set capacity, always ranging between 0

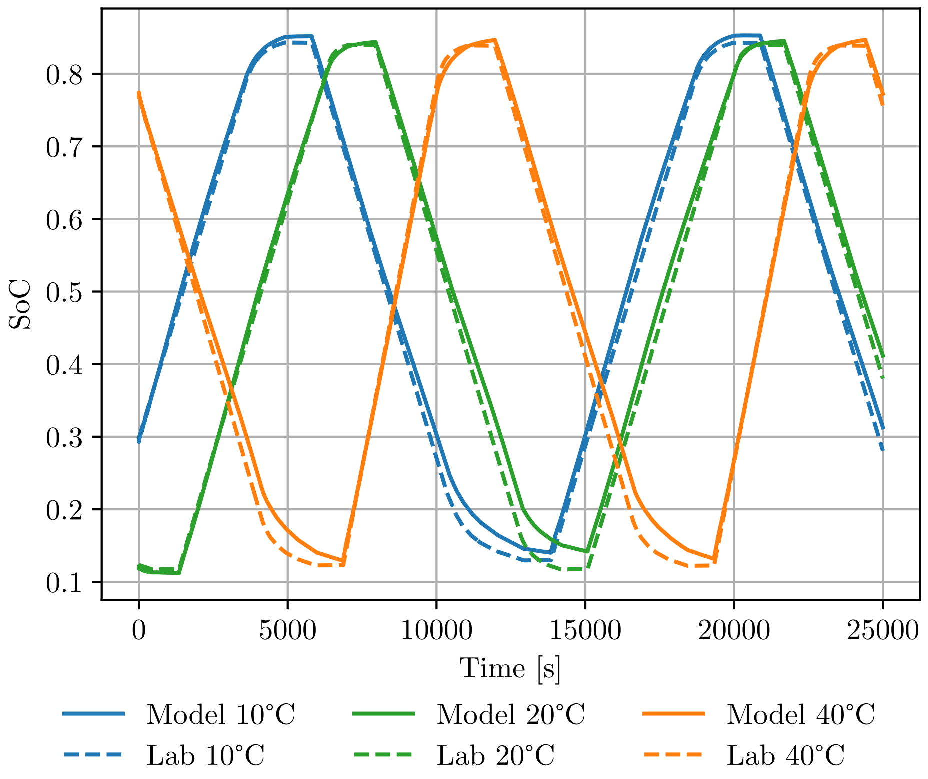

Figure 18: Performance evaluation of the leading model for NMC across different temperatures. This visualization highlights the model's accuracy, initially trained on the synthetic LFP dataset and refined on NMC data, across diverse temperatures (10°C, 20°C, 40°C). The model's predictions across multiple window lengths align closely with the ground truth, indicating its resilience to temperature variations in practical settings.

and 1. This observation underscores the need for adaptable modeling techniques that can cater to different chemistries and capacities.

5 Discussion

This study builds upon previous research by establishing a benchmark on a single dataset and introducing a model novel to the battery domain, while also exploring the concept of cross-battery learning. The research centers on battery modeling using deep learning, with a particular emphasis on the application of a decoder-only model. This model has shown significant potential when applied to battery modeling, especially in processing complex time-series data, crucial for a detailed and accurate battery representation.

The model's performance is largely due to the dataset employed, a combination of synthetic and real-world data. This dataset encompasses diverse battery behaviors and usage scenarios, facilitating tests ranging from different chemistries to varied usage patterns. It serves as a robust foundation for deep learning in battery modeling.

The research offers a detailed analysis of various deep learning architectures, assessing their suitability for battery modeling. This analysis provides insights into the current state and potential of deep learning in battery modeling, highlighting the strengths, weaknesses, and challenges of each model. However, certain limitations accompany the study's findings.

Data: The synthetic dataset, crafted within time limitations, aimed to robustly evaluate deep learning's role in battery modeling. It effectively balanced detailed data with model simplicity. While the dataset is comprehensive, there's potential for expansion, especially in diverse dynamic scenarios. Current data focus mainly on automotive drive cycles, omitting areas like maritime and urban profiles beyond automotive. A deeper dive into data entropy could reveal intricate dependencies. A more detailed dataset analysis might lead to more versatile models. The field data, though extensive, is predominantly based on specific ECM testing data. Expanding the data to cover diverse real-world scenarios would offer a holistic battery behavior view. Future research could benefit from diverse dynamic profiles and exploring varying step sizes. Considering time explicitly, rather than implicitly, might also provide valuable insights. For a comprehensive battery life prediction meta-model, it's essential to include diverse battery chemistries.

Baseline Models: The study employed a diverse range of architectures to assess their suitability for time series modeling in the context of battery data. Among these, certain models exhibited commendable performance, effectively capturing the intricacies of the battery behavior. However, as indicated in Figure 12, some models displayed tendencies towards oscillation. The root causes of these oscillations could span from inherent architectural instability to the necessity for more tailored adjustments. Hyperparameter optimization, particularly in areas like the learning rate and the optimizer, also emerged as a pivotal factor influencing model behavior.

Furthermore, fusion approaches, which combine the strengths of multiple architectures or methodologies, present a promising avenue for exploration. Such approaches can potentially harness the strengths of individual models while mitigating their weaknesses, leading to more robust and accurate battery models.

While the current architectures provided valuable insights, the study acknowledges the vast landscape of deep learning architectures and fusion techniques yet to be explored. Incorporating more diverse and advanced architectures, along with and beyond state-of-the-art fusion strategies, in future studies, could further enhance the understanding and performance in battery modeling.

Decoder: Decoder-only Transformer models effectively addressed the oscillation challenges seen in other models, thanks to their attention mechanism. However, the $O(n^2)$ memory and time complexity of the Transformer architecture is a concern. While Flash Attention offers a solution to the quadratic memory complexity, but the time complexity remains a challenge [38]. The Transformer, even with fewer parameters than some advanced models, effectively comprehends data structures, suggesting optimization potential. Explorations might include new activation functions, optimization techniques, and embeddings. Significant architectural modifications, however, would demand increased computational resources.

Evaluation: Figure 11 shows the validation loss is lower than the test loss, indicating potential overfitting. This is evident in Figure 16 and 17, where initial model performance is strong but wanes over time. The teacher-forcing training method, which makes models rely on actual previous inputs over their predictions, might be a factor, as noted in Figure 6. Using models' outputs as inputs in an autoregressive fashion might address this. Techniques like "scheduled sampling" could enhance model robustness over longer sequences and reduce ground truth reliance [39]. Exploring different loss functions might also influence autoregressive predictions. Transfer learning capabilities, as shown in Table 2, indicates that models trained on one chemistry might be adaptable to others.

In summary, this research underscores the transformative potential of deep learning in battery modeling. The decoder-only models, in particular, have showcased remarkable proficiency, especially in processing complex time-series data. The comprehensive dataset, encompassing both synthetic and real-world data, has been instrumental in achieving these results. Furthermore, the study's detailed analysis of various architectures provides valuable insights into the current state of deep learning in this domain. While the results are promising, there's still room for improvement, particularly in dataset diversification and model optimization. As the field continues to evolve, this research serves as a foundational step, highlighting both the achievements so far and the avenues for future exploration.

6 Conclusion

This study embarked on a journey to harness the capabilities of deep learning for lithium-ion battery modeling, emphasizing decoder-only architectures. Given the dynamic nature of the battery market, where new cell types are frequently introduced, there's often a scarcity of comprehensive data for these novel cells. However, the use of synthetic data, especially derived from FMUs, offers a promising solution. This dataset, encompassing a broad spectrum of battery behaviors, not only validated its quality but also showcased its applicability in real-world scenarios. Such an approach underscores the potential of deep learning models to efficiently process diverse data, optimizing its utilization.

In the quest to identify the most suitable architecture, the research benchmarked various machine learning models. The decoder-only model stood out, reinforcing the adaptability and efficiency of deep learning and championing the democratization of model development. Although the ambitious goal of meta-learning wasn't fully actualized, the transformer model displayed a commendable ability to transfer learning across different chemistries. When tested against an unseen dataset, the Decoder exhibited robust generalization capabilities. However, it should be mentioned that the model did not surpass the performance of traditional ECM models.

In summation, the research extended the field of battery modeling using machine learning. The model demonstrated accuracy, adaptability, and computational efficiency, especially during inference. The study's venture into meta-learning, though not fully actualized, highlighted its potential in model adaptability across battery chemistries and capacities. The transformer model, with its meta-learning approach, exhibited strong performance in transfer learning. While there were certain limitations, the study underscored the combined potential of deep learning and meta-learning in battery modeling, pointing toward future research directions.

Bibliography

[1] International Energy Agency, "Global Electric Vehicle Outlook 2022," 2022.

[2] Kersten Heineke, Benedikt Kloss, Darius Scurtu, and Florian Weig, "Sizing the micro mobility market | McKinsey." https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/micromobilitys-15000-mile-checkup#/.

[3] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah, "Transformers in Vision: A Survey," *ACM Computing Surveys*, vol. 54, pp. 1–41, Jan. 2022.

[4] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, A. Bridgland, C. Meyer, S. A. A. Kohl, A. J. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen, D. Reiman, E. Clancy, M. Zielinski, M. Steinegger, M. Pacholska, T. Berghammer, S. Bodenstein, D. Silver, O. Vinyals, A. W. Senior, K. Kavukcuoglu, P. Kohli, and D. Hassabis, "Highly accurate protein structure prediction with AlphaFold," *Nature*, vol. 596, pp. 583–589, Aug. 2021.

[5] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom, "Llama 2: Open Foundation and Fine-Tuned Chat Models," July 2023.

[6] Z. Cui, L. Wang, Q. Li, and K. Wang, "A comprehensive review on the state of charge estimation for lithium-ion battery based on neural network," *International Journal of Energy Research*, vol. 46, pp. 5423–5440, Apr. 2022.

[7] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.

[8] A. Bhattacharjee, A. Verma, S. Mishra, and T. K. Saha, "Estimating State of Charge for xEV batteries using 1D Convolutional Neural Networks and Transfer Learning," Jan. 2021.

[9] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Computation*, vol. 9, pp. 1735–1780, Nov. 1997.

[10] W. Li, J. Zhang, F. Ringbeck, D. Jöst, L. Zhang, Z. Wei, and D. U. Sauer, “Physics-informed neural networks for electrode-level state estimation in lithium-ion batteries,” *Journal of Power Sources*, vol. 506, p. 230034, Sept. 2021.

[11] Z. Cui, L. Kang, L. Li, L. Wang, and K. Wang, “A hybrid neural network model with improved input for state of charge estimation of lithium-ion battery at low temperatures,” *Renewable Energy*, vol. 198, pp. 1328–1340, Oct. 2022.

[12] X. Zhang, Y. Huang, Z. Zhang, H. Lin, Y. Zeng, and M. Gao, “A Hybrid Method for State-of-Charge Estimation for Lithium-Ion Batteries Using a Long Short-Term Memory Network Combined with Attention and a Kalman Filter,” *Energies*, vol. 15, p. 6745, Sept. 2022.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention Is All You Need,” Dec. 2017.

[14] M. A. Hannan, D. N. T. How, M. S. H. Lipu, M. Mansor, P. J. Ker, Z. Y. Dong, K. S. M. Sahari, S. K. Tiong, K. M. Muttaqi, T. M. I. Mahlia, and F. Blaabjerg, “Deep learning approach towards accurate state of charge estimation for lithium-ion batteries using self-supervised transformer model,” *Scientific Reports*, vol. 11, p. 19541, Dec. 2021.

[15] H. Shen, X. Zhou, Z. Wang, and J. Wang, “State of Charge Estimation for Lithium-ion Batteries in Electric Vehicles by Transformer Neural Network and L $_1$ Robust Observer,” in *2022 American Control Conference (ACC)*, (Atlanta, GA, USA), pp. 370–375, IEEE, June 2022.

[16] T.-S. Dao, C. P. Vyasarayani, and J. McPhee, “Simplification and order reduction of lithium-ion battery model based on porous-electrode theory,” *Journal of Power Sources*, vol. 198, pp. 329–337, Jan. 2012.

[17] U. Westerhoff, K. Kurbach, F. Lienesch, and M. Kurrat, “Analysis of Lithium-Ion Battery Models Based on Electrochemical Impedance Spectroscopy,” *Energy Technology*, vol. 4, pp. 1620–1630, Dec. 2016.

[18] M. Doyle, T. F. Fuller, and J. Newman, “Modeling of Galvanostatic Charge and Discharge of the Lithium/Polymer/Insertion Cell,” *Journal of The Electrochemical Society*, vol. 140, pp. 1526–1533, June 1993.

[19] T. F. Fuller, M. Doyle, and J. Newman, “Simulation and Optimization of the Dual Lithium Ion Insertion Cell,” *Journal of The Electrochemical Society*, vol. 141, pp. 1–10, Jan. 1994.

[20] J. Newman and W. Tiedemann, “Porous-electrode theory with battery applications,” *AIChE Journal*, vol. 21, pp. 25–41, Jan. 1975.

[21] H. Prasanna Das, R. Tran, J. Singh, X. Yue, G. Tison, A. Sangiovanni-Vincentelli, and C. J. Spanos, “Conditional Synthetic Data Generation for Robust Machine Learning Applications with Limited Pandemic Data,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, pp. 11792–11800, June 2022.

[22] F. K. Dankar and M. Ibrahim, “Fake It Till You Make It: Guidelines for Effective Synthetic Data Generation,” *Applied Sciences*, vol. 11, p. 2158, Feb. 2021.

[23] V. Sulzer, S. G. Marquis, R. Timms, M. Robinson, and S. J. Chapman, “Python Battery Mathematical Modelling (PyBaMM),” preprint, ECSarXiv, Feb. 2020.

[24] T. Chen and C. Guestrin, “XGBoost: A Scalable Tree Boosting System,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, Aug. 2016.

[25] A. Gu, K. Goel, and C. Ré, “Efficiently Modeling Long Sequences with Structured State Spaces,” Aug. 2022.

[26] A. Gu, T. Dao, S. Ermon, A. Rudra, and C. Ré, “HiPPO: Recurrent Memory with Optimal Polynomial Projections,”

[27] J. Shi, K. A. Wang, and E. B. Fox, “Sequence Modeling with Multiresolution Convolutional Memory,” May 2023.

[28] N. Kalchbrenner, I. Danihelka, and A. Graves, “Grid Long Short-Term Memory,” Jan. 2016.

[29] A. Gu, C. Gulcehre, T. L. Paine, M. Hoffman, and R. Pascanu, “Improving the Gating Mechanism of Recurrent Neural Networks,” June 2020.

[30] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, “Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization,” June 2018.

[31] S. Falkner, A. Klein, and F. Hutter, “BOHB: Robust and Efficient Hyperparameter Optimization at Scale,” July 2018.

[32] M. Lindauer, K. Eggensperger, M. Feurer, A. Biedenkapp, D. Deng, C. Benjamins, T. Ruhopf, R. Sass, and F. Hutter, “SMAC3: A Versatile Bayesian Optimization Package for Hyperparameter Optimization,” Feb. 2022.

[33] Y. Tay, M. Dehghani, V. Q. Tran, X. Garcia, J. Wei, X. Wang, H. W. Chung, S. Shakeri, D. Bahri, T. Schuster, H. S. Zheng, D. Zhou, N. Houlsby, and D. Metzler, “UL2: Unifying Language Learning Paradigms,” Feb. 2023.

[34] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,”

[35] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” Jan. 2017.

[36] I. Loshchilov and F. Hutter, “Decoupled Weight Decay Regularization,” Jan. 2019.

[37] I. Loshchilov and F. Hutter, “SGDR: Stochastic Gradient Descent with Warm Restarts,” May 2017.

[38] T. Dao, D. Y. Fu, S. Ermon, A. Rudra, and C. Ré, “FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness,” June 2022.

[39] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, “Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks,” Sept. 2015.