Surrogate Models for Fast Prediction and Parameter Inference of Li-ion Batteries

Siddhant Singh (sidme@umich.edu), Haonan Zhao (haonanzh@umich.edu), Atila Haimiti (hatila@umich.edu)
University of Michigan
Ann Arbor, MI, USA

Abstract

Rapid assessment and accurate forecasting of the battery state-of-health are essential for Lithium-ion battery usage and lifespan optimization. However, physics-based solvers are computationally intensive, and traditional surrogate models fail to handle time-series data and dynamically changing conditions. In this paper, the authors investigate a deep surrogate learning models, including the traditional LSTM and the neural operator network, DeepONet. To solve time-series sequential problem, the authors propose Mamba-DeepONet, a state-space model modification of the vanilla DeepONet neural operator, for surrogate modeling of Single Particle Model simulations. Our results show that vanilla DeepONet fails to capture the battery slow- and fast-varying dynamics, but both LSTM and Mamba-DeepONet can capture general battery dynamics. Furthermore, the Mamba-DeepONet effectively simulates both slow- and fast-varying battery dynamics of the Li-ion battery, capable of handling long sequences. We then conduct parameter inference using Tree-structured Parzen Estimator (TPE) algorithm and find that both LSTM and Mamba-DeepONet models show similar performance for parameter inference. Owing to the high parallelizability and the potential to train on much longer time-series sequences, we make a case for why Mamba-DeepONet should be used for surrogate battery models in the future. Website: https://siddhantmich.github.io/battery-ai-surrogates/index.html

Keywords

Physics-informed Deep Neural Operators, DeepONet, Li-battery, electronic devices

1 Introduction

As global efforts to reduce carbon emissions intensify, governments and companies are increasingly turning to electric-powered equipment and appliances as part of the solution. Li-ion batteries are pivotal in this transition, as they are extensively used in commercial electronics, electric vehicles, and power grids. Accurate and fast prediction of the battery performance and state-of-health using battery models are crucial for optimizing battery usage and maximizing their lifespan, especially for safety and efficiency [16] purposes. These battery models are usually systems of ordinary differential equations (ODEs) or partial differential equations (PDEs) that capture the dynamics in the battery at across different lengths and time scales.

Accurately modeling Li-ion batteries is essential for performance prediction under varying conditions. The physics-based single particle model (SPMe) provides detailed insights into the solid-phase particle diffusion, liquid-phase ion transport, and electrochemical kinetics at interfaces across spatial-temporal dimensions that introduce significant computational cost [15]. Meanwhile, traditional surrogate neural networks struggle to manage time-series data and often rely on pre-defined metrics. Therefore, a key challenge in this domain is to accurately model dynamic time-series data while maintaining computational efficiencies and low data requirements. Secondly, simplified assumptions of constant parameters, such as real-world ion diffusivity and temperature, may not hold in the SPMe model[5]. These assumptions can lead to inaccuracies in predicting battery behavior under real-world conditions.

We propose to use neural operators and recurrent neural networks (RNNs) for the above challenges. Neural operators have the potential to capture physics with a rich library of physical variables and also excel at learning complex mappings between function spaces, making them ideal for handling the dynamic and spatiotemporal complexities of battery systems. RNNs resemble the actual modeling process of the battery state, where the battery's state is influenced by its past states and present external excitation functions. Recently, the state-space model, particularly Mamba, combines the structure of the RNNs, the representation ability of transformers and the computing efficiency of convolutional neural networks (CNNs)[9]. However, their performance in the complex physics-based models has been scarcely tested. This task has two parts. First, we seek to leverage neural operators, RNNs and Mamba-based neural operators as surrogates for the SPMe model. Secondly, we also explore parameter inference or the inverse problem using Bayesian estimation models once we have discovered effective surrogate neural networks for battery and battery-like physical systems. Our work is available here: https://siddhantmich.github.io/battery-ai-surrogates/index.html.

2 Problem Definition

2.1 The SPMe Model

The details of the model are taken from Marquis's asymptotic derivation of the SPMe model [15].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Conference'17, July 2017, Washington, DC, USA
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2.1.1 Governing Equations

$$\frac{\partial c^*_{s,k}}{\partial t^*} = \frac{1}{(r^*)^2}\frac{\partial}{\partial r^*}\left((r^*)^2 D^*_{s,k}(c^*_{s,k})\frac{\partial c^*_{s,k}}{\partial r^*}\right), \quad k \in \{n,p\}, \quad \text{(SPM a)}$$

$$\epsilon_k \frac{\partial c^*_{e,k}}{\partial t^*} = -\frac{\partial N^*_{e,k}}{\partial x^*} + \begin{cases} \frac{I^*}{F^* L^*_n}, & k = n, \\ 0, & k = s, \\ -\frac{I^*}{F^* L^*_p}, & k = p, \end{cases} \quad \text{(SPM b)}$$

$$N^*_{e,k} = -\epsilon^b_k D^*_e(c^*_{e,\text{typ}})\frac{\partial c^*_{e,k}}{\partial x^*} + \begin{cases} \frac{x^* + t^* I^*/F^*}{L^*_n}, & k = n, \\ \frac{t^* I^*}{F^*}, & k = s, \\ \frac{(L^* - x^*)t^* + I^*}{F^* L_p}, & k = p, \end{cases} \quad \text{(SPM c)}$$

$$V^* = U^*_{\text{eq}} + \eta^*_r + \eta^*_c + \Delta\Phi^*_{\text{Elec}} + \Delta\Phi^*_{\text{Solid}}. \quad \text{(SPM j)}$$

The variable descriptions are as follows -

- $c^*_{s,k}$: Dimensionless concentration of lithium in the solid phase for electrode $k \in \{n,p\}$, where n and p represent the negative and positive electrodes, respectively.
- t^*: Dimensionless time.
- r^*: Dimensionless radial position within the electrode particles.
- $D^*_{s,k}$: Dimensionless diffusion coefficient of lithium in the solid phase of electrode k.
- ϵ_k: Volume fraction of electrolyte in region $k \in \{n,s,p\}$, where s represents the separator region.
- $c^*_{e,k}$: Dimensionless concentration of lithium ions in the electrolyte for region k.
- $N^*_{e,k}$: Dimensionless molar flux of lithium ions in the electrolyte for region k.
- I^*: Dimensionless applied current.
- F^*: Faraday's constant, representing the charge per mole of electrons.
- L^*_n and L^*_p: Dimensionless thicknesses of the negative and positive electrodes, respectively.

The boundary conditions, initial conditions, and expanded form of the terminal voltage are provided in the Appendix.

2.2 Data Formatting and Overall Picture

The model setup involves an SPMe (Single Particle Model with electrolyte) battery model, which predicts the cell voltage based on two main inputs:

- **Input Current (d)**: This represents the time-varying current applied to the battery, with a dimensionality of 8. This input captures the varying electrical current profiles that the battery experiences over time.
- **Parameters or Constants (θ)**: These are model parameters or constants that define the characteristics of the battery system, such as material properties, reaction rates, and transport properties. The dimensionality of this input is 9, representing the key parameters necessary for the model to perform accurate predictions. The parameters are expanded to the same length as the input current to represent constant parameter over the whole battery dynamics.

The SPMe model processes these inputs to generate the **Voltage Output (y)**. This output represents the predicted voltage of the battery over time and has a dimensionality on the order of 10,000. The parameter-controlled, full battery dynamics includes the slow-varying test, usually occurring in the hybrid pulse power characterization (HPPC), and the fast-varying chirp data, usually occurring in the electrochemical impedance spectroscopy (EIS). The sampling rate between these two phases sometimes differentiate over 100 times. To the traditional deep learning model, such frequency difference poses a challenge in predicting the long time-series data without large data requirements or model size. This is summarized in Figure 1. We seek to replace the computationally heavy SPMe model with a neural operator surrogate.

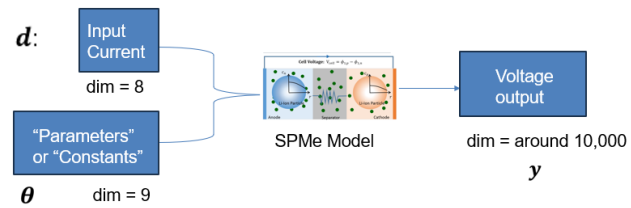


Figure 1: High level formulation of our problem. The data inputs to the SPMe model are input design variables d and physical parameters θ and the output is the voltage y. In the forward problem, we seek to replace the computationally intensive SPMe model with a neural operator surrogate.

3 Related Work

Machine learning algorithms have been widely applied to approximate physics-based simulations, a concept that has garnered significant attention in recent research. For instance, Alvaro et al.[2] leveraged graph neural networks (GNNs) to model complex fluid dynamics under varying initial conditions. However, purely data-driven approaches often demand large-scale datasets for training, which poses limitations in engineering domains where data availability is scarce. To address this challenge, researchers have increasingly adopted physics-informed neural networks (PINNs) for surrogate modeling. Such trends also arise in the field of Li-ion battery simulation. For example, Hassanaly et al.[10] and Singh et al.[18] employed PINNs to model the internal states of batteries based on the single particle model (SPM), incorporating governing physical laws directly into the learning process. In subsequent work, Hassanaly et al.[11] extended this approach to include the more complex pseudo-2D (P2D) model, significantly reducing computational costs associated with Bayesian calibration for parameter inference. Furthermore, Zheng et al.[21] used deep operator networks to predict the electrochemical performance with great accuracy. While these works represent important advancements, they are still constrained by simplifying assumptions. Notably, they often overlook key factors such as mechanical degradation and material uncertainties within the battery. To the best of the authors' knowledge, limited research has addressed these additional complexities, highlighting a critical gap in the existing literature.

One of the earliest applications of PINNs to Li-ion batteries was by Hassanaly et al. [10], who developed a PINN-based surrogate model for the Single Particle Model (SPM). Their work demonstrates how a physics-constrained learning framework could reduce the computational cost associated with solving partial differential equations, while still maintaining reasonable accuracy. The use of multi-fidelity training allowed their model to generalize better in data-limited environments. However, while PINNs offer a good compromise between data requirements and accuracy, they still face challenges in modeling complex, real-world dynamics such as dynamic current profiles. Zheng et al.[21] have shown the use of DeepONets for use as surrogates for battery modeling, while FNO and GINO can be potential useful extensions to batteries owing to the common battery frequency responses.

Another crucial topic in both scientific machine learning and battery research is parameter inference. Parameter inference involves estimating key physical and chemical parameters—such as diffusivity, reaction rates, and state-of-health indicators—from experimental or simulated data. These parameters are vital for understanding battery behavior, optimizing performance, and extending battery life. Traditionally, this process can be computationally expensive and time-consuming, especially when relying on high-fidelity physics-based models. Machine-learning-based surrogate models combined with probabilistic optimization frameworks can greatly accelerate the parameter inference process. Hassanaly et al. [11] demonstrated the use of Bayesian optimization techniques for parameter inference following the successful training of a PINN. In addition to standard Bayesian optimization algorithms, more advanced frameworks, such as the tree-structured Parzen estimator (TPE), offer lower computational complexity compared to traditional methods like Gaussian Process optimization [3]. These approaches expand the toolkit for efficient and scalable parameter inference in complex systems.

4 Methods

4.1 Long short-term memory (LSTM) for Surrogate Modeling

Long Short-Term Memory (LSTM) is a type of RNN architecture designed to capture long-term dependencies in sequential data. Unlike traditional RNNs, LSTMs address the vanishing gradient problem by using a memory cell and gating mechanisms to regulate the flow of information. Numerous studies have demonstrated the efficacy of RNNs, particularly Long Short- Term Memory (LSTM) networks, as potent sequence learning tools[7, 12, 20]. This is attributed to their ability to incorporate past output information when making current predictions. In the context of predicting battery voltage dynamic responses, it's crucial to recognize that the present behavior of a battery voltage is influenced not only by the present input current signal but also by its previous responses. Hence, LSTM emerges as a compelling choice for metamodeling in this domain.

An LSTM unit consists of the following components:

- **Forget Gate:** Determines which information to discard from the cell state. It is computed as:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f),$$

where f_t is the forget gate activation, W_f and b_f are the weights and biases, h_{t-1} is the previous hidden state, and x_t is the current input.

- **Input Gate:** Decides what new information to store in the cell state. It involves:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i),$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C),$$

where i_t is the input gate activation, and $\tilde{C}_t$ is the candidate cell state.

- **Cell State Update:** The cell state is updated as:

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t,$$

where $\odot$ denotes element-wise multiplication.

- **Output Gate:** Determines the output of the LSTM unit:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o),$$
$$h_t = o_t \odot \tanh(C_t),$$

where o_t is the output gate activation, and h_t is the hidden state.

σ represents the sigmoid activation function, and tanh represents the hyperbolic tangent function. These mechanisms enable LSTMs to selectively retain relevant information over long sequences, ensuring the behavior of the battery voltage is influenced by both the present current input signal and the past voltage responses. For the authors' case study, the input to the LSTM model is the system parameters along the current signal, whereas the output of the LSTM is set to be the voltage signal.

4.2 Mamba-DeepONet for Surrogate Modeling

Battery data, such as the state-of-health data, are often time series represented by long sequences with long-range correspondences. General state-space models use hidden state variables to solve dynamical systems. Therefore, it is natural to use state-space deep learning models to recurrently update the hidden state variables and model the sequential data. An emerging alternative to the conventional neural operators, particularly Mamba[9], offers stronger long-range representation ability than fully connected networks, better computational efficiency and scalability than Transformers and RNNs[6, 13]. However, the real-world applications of state-space neural operators are yet to be tested.

4.2.0.1 Mamba

Mamba (S6) is the Structured State-Space model (S4) with Selective Scan (S6), combining the recurrent nature of RNNs, the formulation of state-space models and the parallel computing elements like Convolutional Neural Networks (CNNs)[9]. The ability to handle sequential data is by using the first-order differential equations controlled by the hidden states, see Fig.2 (left):

$$h'(\tau) = Ah(\tau) + Bu(\tau), \qquad y(\tau) = Ch(\tau) + Du(\tau) \tag{1}$$

where $h'(\tau) \in \mathbb{R}^N$ is the hidden state, and $u(\tau), y(\tau) \in \mathbb{R}^N$ are input and output of the PDE model, respectively. The learnable state-space parameters are $A \in \mathbb{C}^{N\times N}$ and $B, C \in \mathbb{C}^N$. The output is finally connected to a linear map D. Mamba adopts a discretization using zero-order holds of the Tayler's expansion, i.e.:

$$\bar{A} = exp(\Delta A), \quad \bar{B} = (\Delta A)^{-1}(exp(\Delta A) - \mathbf{I})\Delta B \tag{2}$$

where $\bar{A}, \bar{B}, C, \Delta$ are selective parameters. Importantly, the above discretization features properties such as resolution invariance that ensuring properly normalized states. It also has connections to gating mechanisms of RNNs using the selective timestep Δ that can weight the focus on the current input $u(\tau)$ in the D channel[9]. This provides the capability of handling data with broadband frequency components and irregular grids.

4.2.0.2 DeepONet

DeepONet is based on the universal approximation theorem for operators, defined as[8, 14]:

THEOREM. *Suppose that $K_1 \subset X$, $K_2 \subset \mathbb{R}^{D_{in}}$ are two compact sets in a Banach space X and $\mathbb{R}^{D_{in}}$, respectively, V is a compact set in $C(K_1)$. Assume that $\mathcal{G} : V \mapsto C(K_2)$ is a nonlinear continuous operator. Then for any $\epsilon > 0$, there exist positive integer m, p, branch nets br_k and trunk nets tr_k, and $\eta_1, \eta_1, \ldots, \eta_m \in K_1$ such that:*

$$\sup_{v \in V} \sup_{\xi \in K_2} |\mathcal{G}(v)(\xi) - \sum_{k=1}^{p} \underbrace{br_k\left(v\left(\eta_1\right), v\left(\eta_2\right), \ldots, v\left(\eta_m\right)\right)}_{branch} \underbrace{\mathrm{tr}_k(\xi)}_{trunk}| < \epsilon. \tag{3}$$

where $u_i(\xi)$ and N are the ground truth and the number of functions in the branch net, respectively. The conventional DeepONet consists of two deep neural networks, a branch net and a trunk net. The branch net encodes the input function, $v(\eta)$, at fixed sensing points to provide coefficients, and the trunk net encodes the spatiotemporal coordinates ξ to provide basis functions. Given the parameters, θ, the output of the DeepONet and the loss function are defined as:

$$\begin{aligned} \mathcal{G}_\theta\left(v_i\right)(\xi) &= \sum_{j=1}^{p} br_j\left(v_i\left(\eta_1\right), v_i\left(\eta_2\right), \ldots, v_i\left(\eta_m\right)\right) \cdot tr_j(\xi) \\ \mathcal{L}(\theta) &= \frac{1}{N} \sum_{i=1}^{N} w_i \left|u_i(\xi) - \mathcal{G}_\theta\left(v_i\right)(\xi)\right|^2 \end{aligned} \tag{4}$$

Once trained, DeepOnet can be evaluated at any discretization for variables defined in the trunk net. We refer to the DeepONet with FNN branch nets as Vanilla DeepONet, see Fig. 2 (middle).

4.2.0.3 Mamba-DeepONet

We propose Mamba-DeepONet to combine the benefits of both worlds. We add Mamba blocks before the FNN in the branch net to encode the sequential functional space of the input, including the current and parameter sequences, before its dot-product interaction with the encoded temporal domain in the trunk net. In general, the Mamba block in the branch net is expected to represent frequency- and resolution-invariant long sequence, where the trunk net is expected to represent the structure of the output domain as in [14]. Therefore, Mamba-DeepONet has the capability of handling longer sequences than Transformers, both fast- and slow-varying system dynamics, as well as the representation of multidimensional spaces and parallel computation that conventional RNN lacks.

4.3 Tree-structured Parzen Estimator Algorithm (TPE) for Parameter Inference

The Tree-structured Parzen Estimator (TPE) is a Bayesian optimization method for hyperparameter search. It models the objective function $f(x)$, where x represents the hyperparameters, as a probabilistic model. TPE is more effective as compared to the standard Gaussian process for cases where hyper-parameter spaces are high-dimensional as well as with tree-like structures. The major difference between the well-known Gaussian process (GP) and TPE is that TPE models $p(x|y)$, the distribution of hyperparameters given the performance metric y, rather than directly modeling $p(y|x)$. TPE further split the $p(x|y)$ into to categories $l(x) = p(x|y \leq y^*)$, the density of "good" observation, and $g(x) = p(x|y > y^*)$, the density of bad observation. y^* is the user-defined quantile of observed performances. The densities can be estimated through the kernel density estimation using the "good" and the "bad" observations. The initiation of TPE follows that the optimal parameter is likely to have a higher probability in the "good" observations and a lower probability in the "bad" observations. Based on this initiation, the expected improvement is defined as:

$$EI(x) = \frac{g(x)}{l(x)},$$

Thus, the next optimal hyperparameter candidate is determined using $x^* = \arg\max EI(x)$. In the authors' case study, system parameters are treated as hyperparameters once the forward simulation is trained. The objective is to identify the optimal system parameters that best match the ground truth voltage response for a given current signal. This assumes that both the voltage response and the current signal are readily available, as is typically the case in inverse problem settings.

5 Experiments/Results

The experiments conducted in this study aim to address two pivotal questions: firstly, can surrogate models effectively predict the behavior of lithium-ion batteries with high accuracy? And secondly, how well do these surrogate models perform in parameter inference tasks, particularly when optimized using advanced Bayesian techniques?

5.0.0.1 Data Generation Process/Data Sets

The dataset generation was done using the Python-based battery modeling framework PyBaMM [19]. PyBaMM enables the flexible and efficient modeling of lithium-ion battery systems by solving the relevant electrochemical equations that govern battery dynamics. The input parameters for the dataset include a range of current profiles and various battery-specific parameters, such as ion diffusivity, porosity of battery components, and bi-layer capacitance, all of which influence the physical state and performance of the battery. The current profiles used for this study were drawn from a proposed experimental current design involving the use of a fast, chirped EIS signal and slow-varying HPPC signal[4, 17]. First, we focus on the use of HPPC characterization combined with standard cycling. Second, we set out on the neural operator performances on the additional use of a chirp signal, that is, having a frequency sweep from a low frequency to a high frequency, that differs in scale by as

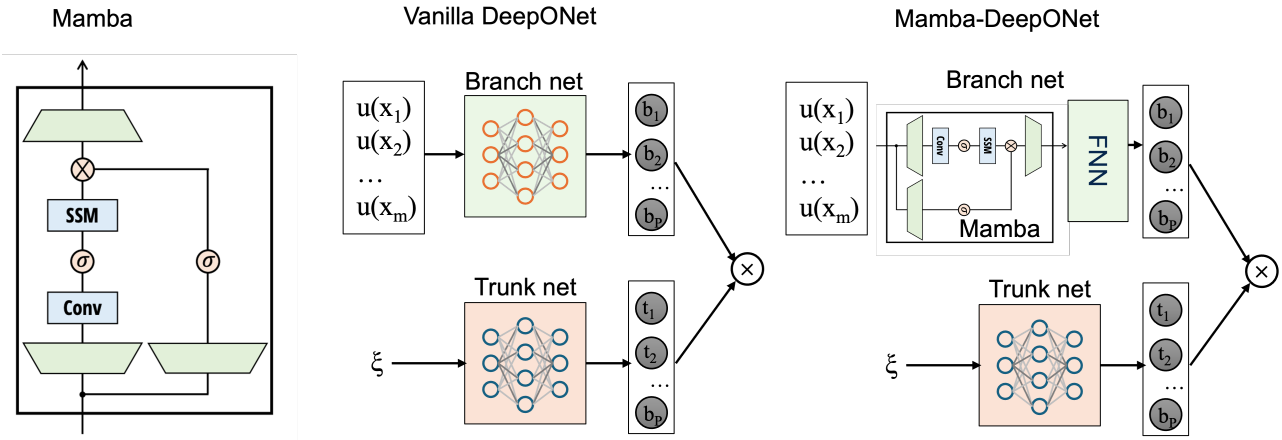
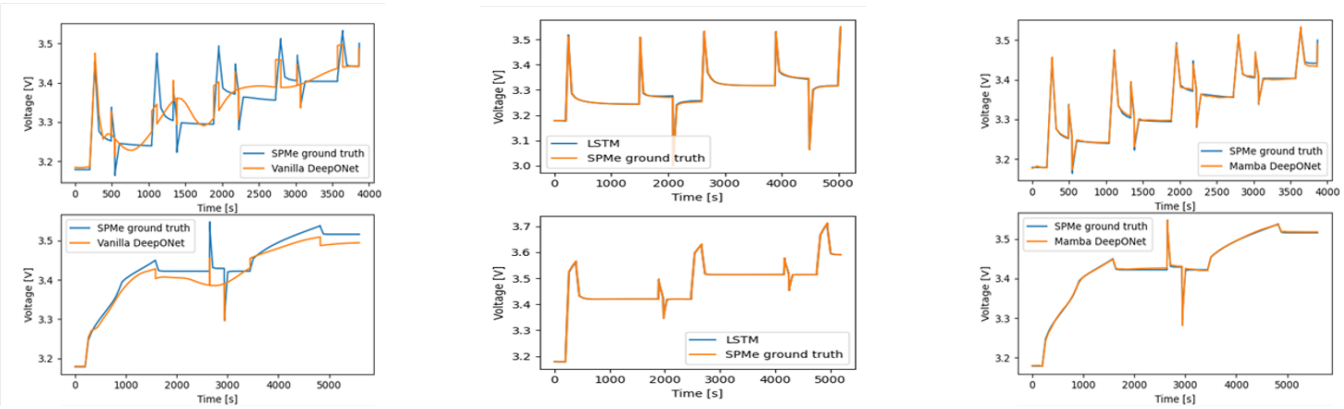


Figure 2: Structures of Mamba (block), Vanilla DeepONet and Mamba-DeepONet



(a) Vanilla-DeepONet **(b) LSTM** **(c) Mamba-DeepONet**

Figure 3: Voltage predictions compared to slow-varying HPPC ground truth across multiple test cases, demonstrating the poor fitting behavior of DeepONet to the model dynamics. LSTM and Mamba-DeepONet show better fitting.

much as 10^4 Hz. In this study, we down-sampled the high-frequency part of the data by a factor 10, while keeping the low-frequency data unchanged. The full dynamics of the battery sequence length is reduced to 4096 rather than >50000. The output of the dataset will consist of the voltage profiles corresponding to the input current profiles, which will encapsulate the time-dependent response of the battery. PyBaMM uses variable time stepping, which ensures that the model can efficiently handle the complex dynamics of the system, adjusting the time step according to the intricacies of the input profile. In some cases, time steps may be as small as 10^{-8} seconds when capturing rapid transient behavior in the system. We generated 4000 model runs for HPPC-only (slow-varying) and 1028 model runs for the full dynamics (HPPC +EIS), with each run representing the battery's performance over a period of hours.

5.0.0.2 Baseline: Surrogate Modeling with DeepONet

The DeepONet model is a foundational neural operator architecture that maps input sequences to output sequences, leveraging separate networks for encoding the input features and the temporal dependencies. The branch network is implemented as a fully-connected neural network (FNN) designed to process the system parameters and transform them into a latent representation. The trunk network handles the temporal aspects of the problem, mapping the time-dependent input signals to the output voltage predictions. In this experiment, the branch and trunk nets are identical, consisting of 4 fully connected layers with a hidden width of 50 and Sigmoid local activations at each layer to introduce non-linearity. The final layers of branch and trunk nets undergo a dot-product interaction, and projects the transformed input into a latent space with a hidden width of 50. The final output is summed over the hidden state dimension, representing the predicted voltage signal at each time step. In experiments, the DeepONet model struggled

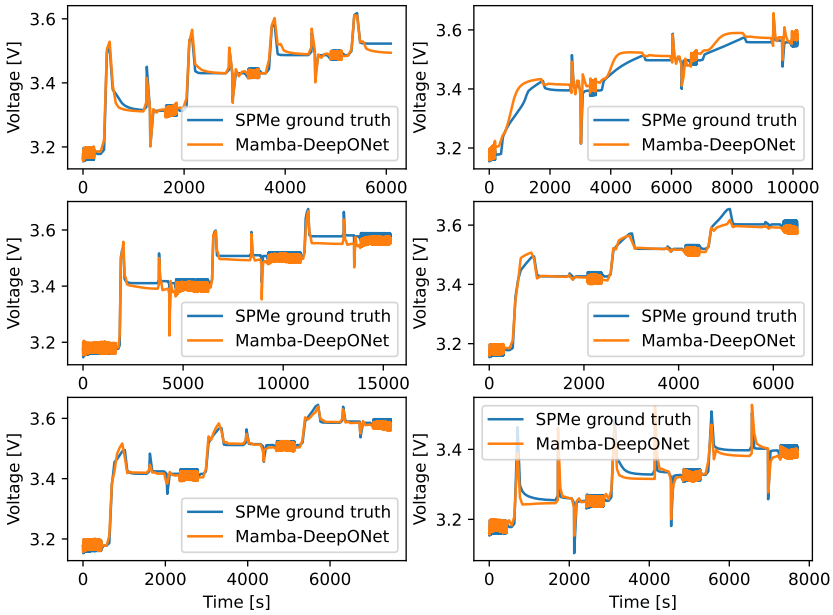


Figure 4: Mamba-DeepOnet is capable of predicting dynamics with both high-frequency region and low-frequency region. Total sequence length = 4096. Sampling rate ratio between the high and low frequency region is 10:1.

to capture the intricate dynamics of lithium-ion batteries. With an MSE of 1.41×10^{-3}, its performance significantly lags behind both LSTM and Mamba-DeepONet models. While the model is capable of capturing coarse-grained trends in the voltage response, it fails to represent finer details, such as pulse responses and the slow-varying transitions, as seen in the examples in Figure 3a and Appendix Figures 6 and 7.

5.0.0.3 Surrogate Modeling With LSTM

The LSTM model used in the case study comprises three key components. First, a multilayer perceptron (MLP) with 64 hidden units and 64 output features transforms the input data into a higher-dimensional space, enabling the model to better capture intricate relationships. Second, an LSTM module with a hidden size of 64 is employed to handle time-series prediction effectively. Finally, another MLP with 32 hidden units and an output size of 1 maps the LSTM predictions to the desired output dimension for each time step.

The LSTM model demonstrated exceptional predictive accuracy for HPPC, achieving a testing loss of 3.71×10^{-6}. A visual comparison of the predicted voltage responses with the SPMe ground truth across multiple test cases (Figure 3b) illustrates the model's ability to accurately capture both rapid transitions and slower dynamics in the voltage profiles. Thus because of the LSTM's capacity to handle non-linear interactions and multi-scale temporal features, we are able to successfully replicate the dynamic behavior of lithium-ion batteries.

5.0.0.4 Surrogate Modeling with Mamba-DeepONet

The Mamba-DeepONet maintains the baseline Vanilla DeepONet structure, except for adding one Mamba block before the branch FNN net. The Mamba block starts with a linear projection to a 64-dimensional hidden state, then a local convolution layer and SiLu activation before feeding into the state-space model. After the state-space model, the results undergo a dot-product interaction with the input duplicate that is projected using a linear layer with 64 hidden dimensions and local SiLu acitvation.

The Mamba-DeepONet outperforms the vanilla DeepONet in the HPPC data, achieving a significantly lower MSE of 1.43×10^{-5}. Visual comparisons show that Mamba-DeepONet captures both the rapid transitions and finer details of the voltage response that the base DeepONet struggled with (see Figure 3c). The inclusion of the Mamba state-space architecture in the branch network allows the model to better capture system behavior, making it well-suited for systems with long temporal sequences and complex multi-scale dynamics.

Moreover, Mamba-DeepONet can also handle the fast-varying chirp response better than Vanilla DeepONet, see Appendix Fig. 7. Mamba-DeepONet exhibits a test MSE of 7.57×10^{-7}, whereas Vanilla DeepONet has a higher MSE of 7.33×10^{-6}. Also, although the LSTM performs better than Mamba in the relatively short (sequence length =105), slow-varying responses in the HPPC regimes, rapidly changing data are often long sequences due to a higher sampling rate, where RNN-based models suffer from increasing model complexity with long sequence inputs. Mamba-DeepONet is

capable of handling long sequences (4096 on a personal computer) as shown in Appendix Fig. 7 and has better scalability in general. Combining the HPPC and EIS together, the Mamba-DeepONet has achieved more accurate predictions in 4, as well as lower train and test loss over the vanilla DeepONet, see 8. Therefore, Mamba-DeepONet has potential for dynamic systems with long-sequence and broadband components.

5.0.0.5 Parameter Inference Results

We ran a tree-structured Parzen Estimator study over the HPPC data using Optuna, a hyperparameter optimization library in Python. We used a Gaussian kernel with an adaptive bandwidth with the default loss threshold (0.01). The optimization process involved 2000 trials, with each trial minimizing the error between the surrogate model's predicted voltage responses and the SPMe ground truth.

The results reveal that TPE is effective for parameter inference, with normalized errors ranging from 0.0115 for parameters with high sensitivity to 35.69 for parameters with more complex dependencies (see Table 1). From a physical battery modeling standpoint, these results also make sense: parameters 0, 1, 2, and 3 all correspond to positive-electrode diffusivity, and there is a lot of literature showing that this is often not identifiable for a range of battery operating conditions (that is, several different values of the parameters satisfy the output voltage recorded)[1]. On the other hand, parameters 4 to 8 correspond to parameters that should have a more unique correspondence with the output voltage (negative ion diffusivity, negative electrode porosity, separator porosity, positive electrode porosity, and double-layer capactiance respectively), and it is encouraging to see the low error for parameter inference for either forward model used.

We also visualize the parameter fits across 400 testing samples as histograms showing the effectiveness of TPE for parameter inference for batteries in Figure 5 .

Parameter Index	LSTM Error (Normalized)	Mamba-DeepONet Error (Normalized)
0	1.285372	1.91261
1	13.12026	18.138
2	30.7648	35.69853
3	2.425946	3.031413
4	0.011579	0.012378
5	0.245476	0.245792
6	0.202349	0.251153
7	0.250003	0.232978
8	0.567224	0.607218

Table 1: Comparison of Normalized Errors For Parameter Identification Using TPE between LSTM and Mamba-DeepONet

6 Conclusion and Discussion

The results demonstrate that the proposed approaches, Mamba-DeepONet and LSTM, outperform the standard DeepONet in terms of prediction accuracy of slow-varying, hybrid pulsed power characterization of bartteries. Notably, in the case study, Mamba-DeepONet outperforms the vanilla DeepONet in the full HPPC and EIS tests of batteries. LSTM has exhibited higher prediction accuracy than Mamba-DeepONet in the HPPC data. However, Mamba-DeepONet has significant advantages over LSTM for longer time series and broadband times-series modeling. This is due to its superior parallelization capabilities and its ability to capture long-range interactions in the time series, attributed to the incorporation of the Mamba framework. Using accurate surrogate models, the authors employed the tree-structured Parzen estimator algorithm for parameter inference. The results indicate that the authors achieved reasonable prediction accuracy for the parameters. However, due to the nature of the formulation of the battery models, different parameter combinations can sometimes produce the same voltage dynamic response. This phenomenon may hinder achieving higher accuracy in parameter inference.

7 Future Works

In the future, the authors aim to extend the prediction sequence length and more model physics to better align with real-world application scenarios. The SPMe model can provide underlying parameters and the mesh grids. The authors plan to incorporate these information into their forward and inverse problems to enhance the model's applicability. Furthermore, to facilitate publication in conferences and international journals, it is critical to investigate and articulate the underlying mechanisms that make Mamba-DeepONet outperform other models, such as LSTM or standalone DeepONet. To this end, the authors plan to conduct a comprehensive ablation study across multiple case studies, potentially including battery modeling, civil engineering structural dynamics, electronic device modeling. Finally, the authors aspire to incorporate physical loss functions into the Mamba-DeepONet framework to further improve its predictive accuracy and robustness, ensuring it remains highly relevant to practical engineering challenges.

References

[1] Antti Aitio, Scott G Marquis, Pedro Ascencio, and David Howey. 2020. Bayesian parameter estimation applied to the Li-ion battery single particle model with electrolyte dynamics. *IFAC-PapersOnLine* 53, 2 (2020), 12497–12504.

[2] Sanchez-Gonzalez Alvaro, Godwin Jonathan, Pfaff Tobias, and et.al. 2020. Learning to Simulate Complex Physics with Graph Networks. *International conference on machine learning* (2020).

[3] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems* 24 (2011).

[4] Brian Bullecks, Resmi Suresh, and Raghunathan Rengaswamy. 2017. Rapid impedance measurement using chirp signals for electrochemical system analysis. *Computers & chemical engineering* 106 (2017), 421–436.

[5] Kudakwashe Chayambuka, Grietus Mulder, Dmitri L Danilov, and Peter HL Notten. 2021. Determination of state-of-charge dependent diffusion coefficients and kinetic rate constants of phase changing electrode materials using physics-based models. *Journal of Power Sources Advances* 9 (2021), 100056.

[6] Chun-Wun Cheng, Jiahao Huang, Yi Zhang, Guang Yang, Carola-Bibiane Schönlieb, and Angelica I Aviles-Rivero. 2024. Mamba Neural Operator: Who Wins? Transformers vs. State-Space Models for PDEs. arXiv:2410.02113 [cs.LG] https://arxiv.org/abs/2410.02113

[7] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. 2000. Learning to forget: Continual prediction with LSTM. *Neural computation* 12, 10 (2000), 2451–2471.

[8] Somdatta Goswami, Aniruddha Bora, Yue Yu, and George Em Karniadakis. 2023. Physics-informed deep neural operator networks. In *Machine Learning in Modeling and Simulation: Methods and Applications*. Springer, 219–254.

[9] Albert Gu and Tri Dao. 2024. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. arXiv:2312.00752 [cs.LG] https://arxiv.org/abs/2312.00752

[10] Malik Hassanaly, Ryan Weddle, Peter ad King, and et.al. 2024. PINN surrogate of Li-ion battery models for parameter inference,PartI: Implementation and multi-fidelity hierarchies for the single-particle model. *Journal of Energy Storage* (2024).

[11] Malik Hassanaly, Ryan Weddle, Peter ad King, and et.al. 2024. PINN surrogate of Li-ion battery models for parameter inference,PartII: Regularization and application of the pseudo-2Dmodel. *Journal of Energy Storage* (2024).

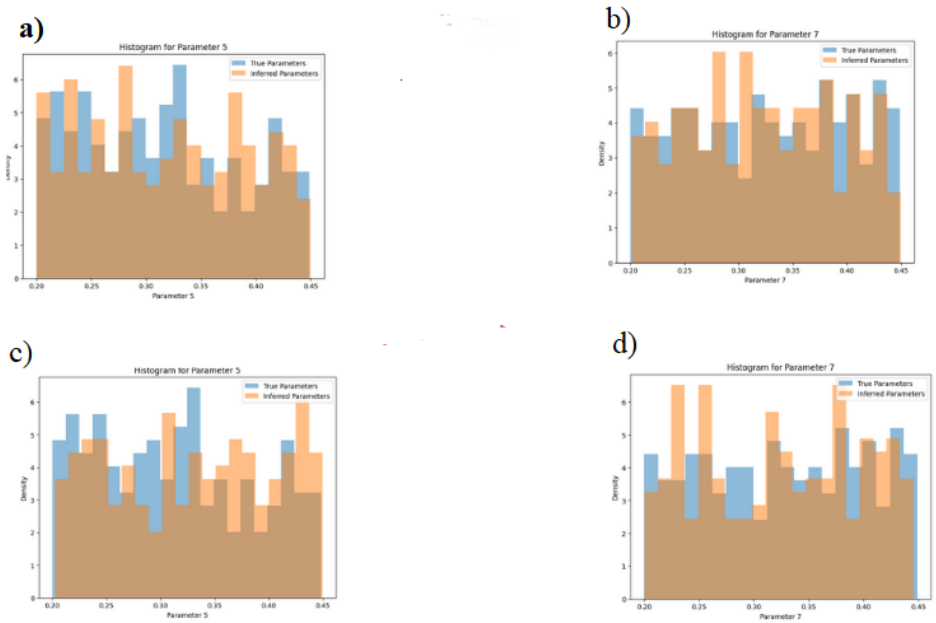


Figure 5: Some selected histograms showing the distribution of inferred parameters versus true parameters. Parameters 5 and 7 physically represent negative electrode porosity and positive electrode porosity respectively. a) and b) correspond to the LSTM model and c) and d) correspond to Mamba-DeepONet.

[12] S Hochreiter. 1997. Long Short-term Memory. *Neural Computation MIT-Press* (1997).

[13] Zheyuan Hu, Nazanin Ahmadi Daryakenari, Qianli Shen, Kenji Kawaguchi, and George Em Karniadakis. 2024. State-space models are accurate and efficient neural operators for dynamical systems. arXiv:2409.03231 [cs.LG] https://arxiv.org/abs/2409.03231

[14] Lu Lu, Pengzhan Jin, and George Em Karniadakis. 2019. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193* (2019).

[15] Scott G Marquis, Valentin Sulzer, Robert Timms, Colin P Please, and S Jon Chapman. 2019. An asymptotic derivation of a single particle model with electrolyte. *Journal of The Electrochemical Society* 166, 15 (2019), A3693.

[16] Miquel Martí-Florences, Andreu Cecilia, and Ramon Costa-Castelló. 2023. Modelling and Estimation in Lithium-Ion Batteries: A Literature Review. *Energies* 16, 19 (2023), 6846.

[17] BH SHH and GAT EE. 1999. PNGV battery test manual.

[18] S. Singh, Y.E. Ebongue, S. Rezaei, and K.P. Birke. 2023. Hybrid Modeling of Lithium-Ion Battery: Physics-Informed Neural Network for Battery State Estimation. *Batteries* (2023).

[19] Valentin Sulzer, Scott G Marquis, Robert Timms, Martin Robinson, and S Jon Chapman. 2021. Python battery mathematical modelling (PyBaMM). *Journal of Open Research Software* 9, 1 (2021).

[20] Zheng Zhao, Weihai Chen, Xingming Wu, Peter CY Chen, and Jingmeng Liu. 2017. LSTM network: a deep learning approach for short-term traffic forecast. *IET intelligent transport systems* 11, 2 (2017), 68–75.

[21] Qiang Zheng, Xiaoguang Yin, and Dongxiao Zhang. 2023. Inferring electrochemical performance and parameters of Li-ion batteries based on deep operator networks. *Journal of Energy Storage* (2023).

8 Appendix

8.1 SPMe Model Continued

8.1.1 Boundary Conditions

$$\left.\frac{\partial c^*_{s,k}}{\partial r^*}\right|_{r^*=0} = 0, \tag{5}$$

$$-D^*_{s,k}(c^*_{s,k})\left.\frac{\partial c^*_{s,k}}{\partial r^*}\right|_{r^*=R^*_k} = \begin{cases} \frac{I^*}{F^* a^*_n L^*_n}, & k=n, \\ -\frac{I^*}{F^* a^*_p L^*_p}, & k=p, \end{cases} \quad k \in \{n,p\}, \tag{SPM d}$$

$$N^*_{e,n}\big|_{x^*=0} = 0, \quad N^*_{e,p}\big|_{x^*=L^*} = 0, \tag{SPM e}$$

$$c^*_{e,n}\big|_{x^*=L^*_n} = c^*_{e,s}\big|_{x^*=L^*_n}, \quad N^*_{e,n}\big|_{x^*=L^*_n} = N^*_{e,s}\big|_{x^*=L^*_n}, \tag{SPM f}$$

$$c^*_{e,s}\big|_{x^*=L^*-L^*_p} = c^*_{e,p}\big|_{x^*=L^*-L^*_p}, \quad N^*_{e,s}\big|_{x^*=L^*-L^*_p} = N^*_{e,p}\big|_{x^*=L^*-L^*_p}. \tag{SPM g}$$

The variable descriptions are as follows -

- R^*_k: Radius of active material particles in electrode k.
- $D^*_{s,k}$: Diffusion coefficient of lithium in the solid phase of electrode k.
- I^*: Applied current.
- F^*: Faraday's constant.
- a^*_k: Surface area density of the electrode k.

- L_n^* and L_p^*: Thicknesses of the negative and positive electrodes, respectively.
- $N_{e,k}^*$: Molar flux of lithium ions in the electrolyte for region k.

8.1.2 Initial Conditions

$$c_{s,k}^*(r^*, 0) = c_{k,0}^*, \quad k \in \{n, p\}, \tag{SPM h}$$

$$c_{e,k}^*(x^*, 0) = c_{e,\text{typ}}^*, \quad k \in \{n, s, p\}. \tag{SPM i}$$

Where we have:

- $c_{k,0}^*$: Initial concentration of lithium in the solid phase for electrode k at $t = 0$.
- x^*: Position along the thickness of the battery.
- $c_{e,\text{typ}}^*$: Typical initial concentration of lithium ions in the electrolyte.

8.1.3 Terminal Voltage

$$U_{\text{eq}}^* = U_p^*\left(c_{s,p}^*\big|_{r^*=R_p^*}\right) - U_n^*\left(c_{s,n}^*\big|_{r^*=R_n^*}\right), \tag{SPM k}$$

$$\eta_r^* = -\frac{2R^*T^*}{F^*}\sinh^{-1}\left(\frac{I^*}{a_p^* j_{0,p}^* L_p^*}\right) - \frac{2R^*T^*}{F^*}\sinh^{-1}\left(\frac{I^*}{a_n^* j_{0,n}^* L_n^*}\right), \tag{SPM l}$$

$$\eta_c^* = \frac{2R^*T^*}{F^* c_{e,\text{typ}}^*}(1 - t^+)(c_{e,p}^* - c_{e,n}^*), \tag{SPM m}$$

$$\bar{j}_{0,n}^* = \frac{1}{L_n^*}\int_0^{L_n^*} m_n^*(c_{s,n}^*)^{1/2}(c_{s,n,\max}^* - c_{s,n}^*)^{1/2}(c_{e,n}^*)^{1/2}\,dx^*, \tag{SPM n}$$

$$\bar{j}_{0,p}^* = \frac{1}{L_p^*}\int_{L^*-L_p^*}^{L^*} m_p^*(c_{s,p}^*)^{1/2}(c_{s,p,\max}^* - c_{s,p}^*)^{1/2}(c_{e,p}^*)^{1/2}\,dx^*, \tag{SPM o}$$

$$\Delta\Phi_{\text{Elec}}^* = -\frac{I^*}{k_e^*(c_{e,\text{typ}}^*)}\left(\frac{L_n^*}{3\epsilon_n^b} + \frac{L_s^*}{\epsilon_s^b} + \frac{L_p^*}{3\epsilon_p^b}\right), \tag{SPM p}$$

$$\Delta\Phi_{\text{Solid}}^* = -\frac{I^*}{3}\left(\frac{L_p^*}{\sigma_p^*} + \frac{L_n^*}{\sigma_n^*}\right). \tag{SPM q}$$

The following notations are used in the preceding section:

- V^*: Terminal voltage.
- U_{eq}^*: Equilibrium potential difference between the positive and negative electrodes.
- η_r^*: Reaction overpotential, related to the kinetics of lithium-ion exchange.
- η_c^*: Concentration overpotential, representing the concentration difference across the electrolyte.
- $\Delta\Phi_{\text{Elec}}^*$: Ohmic losses in the electrolyte.
- $\Delta\Phi_{\text{Solid}}^*$: Ohmic losses in the solid phase.
- U_p^*, U_n^*: Open-circuit potentials of the positive and negative electrodes, respectively.
- $c_{s,p}^*$, $c_{s,n}^*$: Concentrations of lithium in the solid phase of the positive and negative electrodes, respectively.
- R^*: Universal gas constant.
- T^*: Absolute temperature.
- F^*: Faraday's constant.
- I^*: Applied current.
- a_k^*: Specific surface area of electrode k.
- $j_{0,k}^*$: Exchange current density for electrode k.
- $c_{e,\text{typ}}^*$: Typical concentration of lithium ions in the electrolyte.
- t^+: Transference number, representing the fraction of current carried by lithium ions.
- $\bar{j}_{0,n}^*$, $\bar{j}_{0,p}^*$: Electrode-averaged exchange current densities for the negative and positive electrodes, respectively.
- m_k^*: Reaction rate constant for electrode k.
- $c_{s,k,\max}^*$: Maximum concentration of lithium in the solid phase of electrode k.
- k_e^*: Conductivity of the electrolyte.
- ϵ_k^b: Bruggeman coefficient for region k, representing the porous structure of the electrode.
- L_n^*, L_s^*, L_p^*: Thicknesses of the negative electrode, separator, and positive electrode, respectively.
- σ_k^*: Conductivity of the solid phase in electrode k.

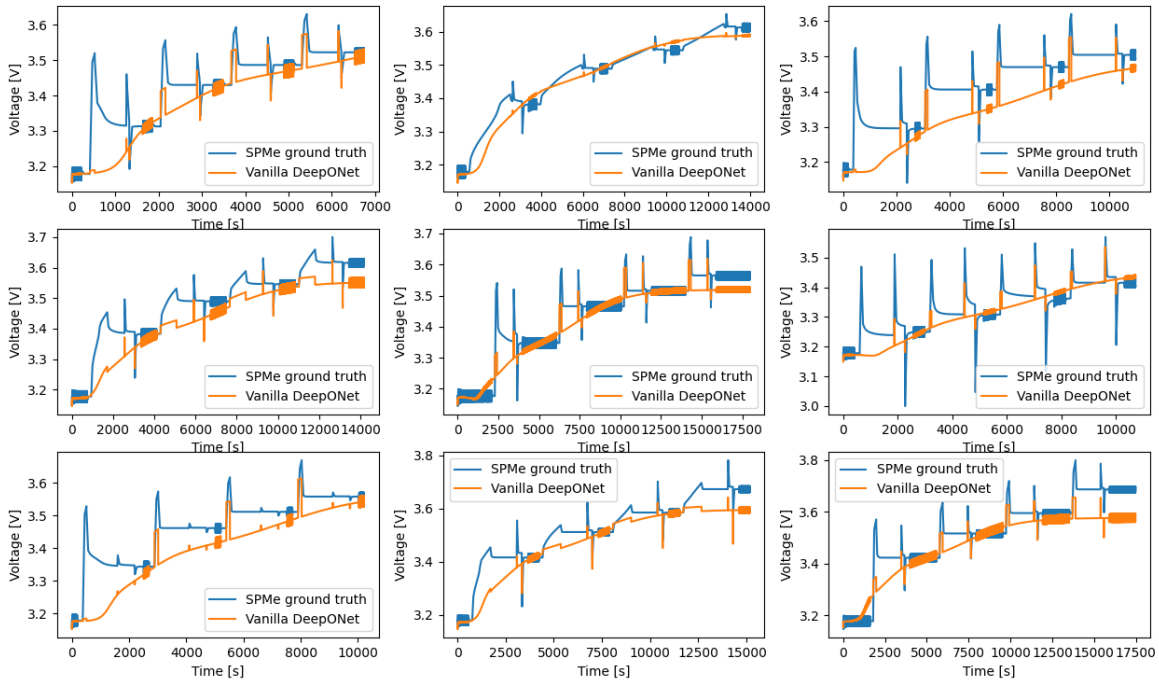


Figure 6: Results of vanilla DeepONet compared to PyBAMM ground truth

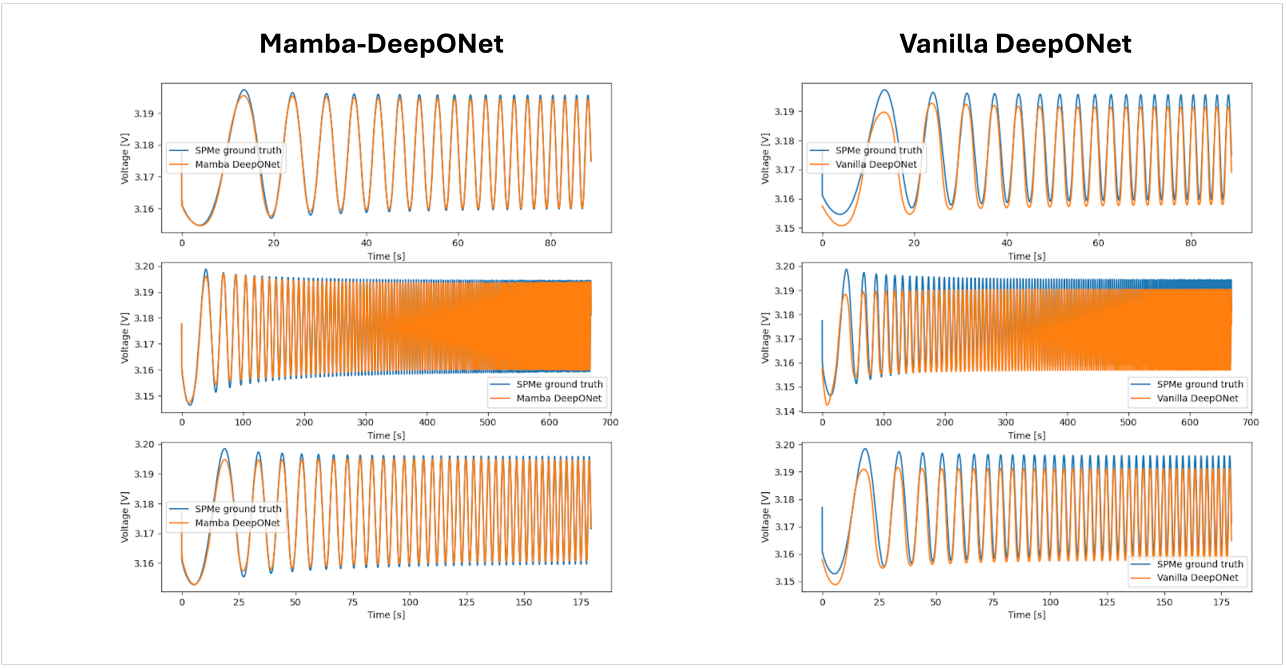
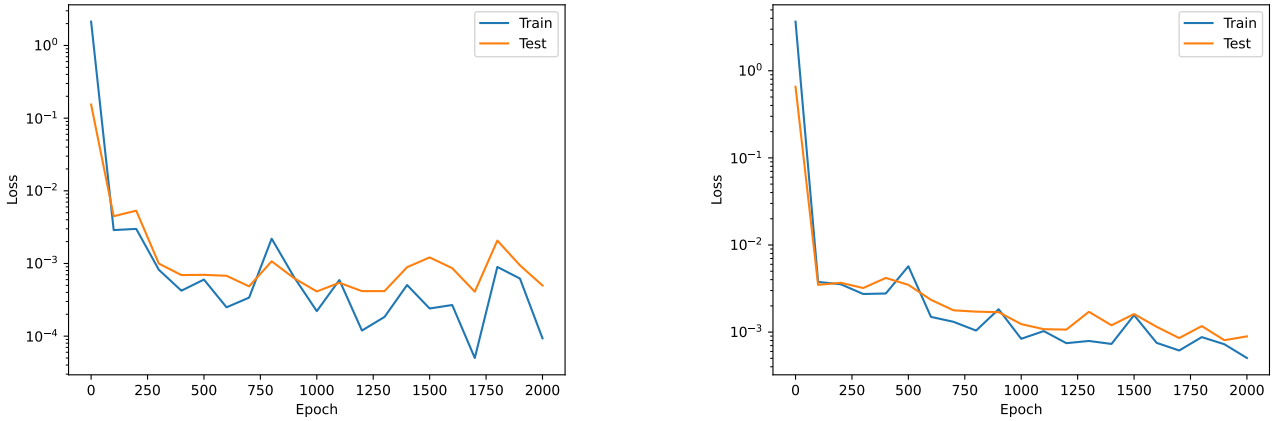


Figure 7: High-frequency chirp response prediction. Mamba-DeepONet vs. Vanilla DeepONet. Sequence length is 4096.



(a) Mamba-DeepONet **(b) Vanilla-DeepONet**

Figure 8: Training and testing losses for full Li-ion battery dynamics.