

# Fast and generalisable parameter-embedded neural operators for lithium-ion battery simulation

Amir Ali Panahi <sup>a,b,c,d,\*</sup>, Daniel Luder <sup>a,b,1</sup>, Billy Wu <sup>d,e</sup>, Gregory Offer <sup>c,e,f</sup>, Dirk Uwe Sauer <sup>a,b</sup>, Weihan Li <sup>a,b,\*\*</sup>

<sup>a</sup> Center for Ageing, Reliability and Lifetime Prediction of Electrochemical and Power Electronic Systems (CARL), RWTH Aachen University, Campus-Boulevard 89, Aachen, 52074, Germany

<sup>b</sup> Institute for Power Electronics and Electrical Drives (ISEA), RWTH Aachen University, Campus-Boulevard 89, Aachen, 52074, Germany

<sup>c</sup> Department of Mechanical Engineering, Imperial College London, London, SW7 2AZ, United Kingdom

<sup>d</sup> Dyson School of Design Engineering, Imperial College London, South Kensington Campus, London, SW7 2AZ, United Kingdom

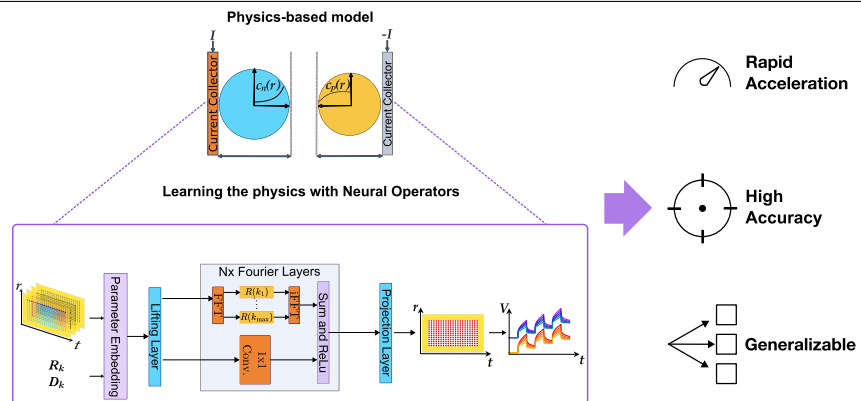
<sup>e</sup> The Faraday Institution, Harwell Science and Innovation Campus, Didcot, OX11 0RA, United Kingdom

<sup>f</sup> Vehicle Futures Hub, Imperial College London, London, SW7 2AZ, United Kingdom

## HIGHLIGHTS

- Applies a discretisation-invariant neural operator to lithium-ion battery equations.
- Handles charge, current, particle size and diffusion via physics-aware embeddings.
- Achieves order-of-magnitude speed gains over conventional methods in fair comparison.
- Examines the forward and inverse trade-offs in Neural Operator for lithium-ion batteries.

## GRAPHICAL ABSTRACT



## ARTICLE INFO

Dataset link: [https://github.com/alighato/spm\\_neuraloperator](https://github.com/alighato/spm_neuraloperator)

### Keywords:

Physics-informed machine learning  
Operator learning  
Deep Operator Network  
Fourier Neural Operator  
Lithium-ion batteries

## ABSTRACT

Digital twins of lithium-ion batteries are increasingly used to enable predictive monitoring, control, and design at system scale. Increasing their capabilities involves improving their physical fidelity while maintaining sub-millisecond computational speed. In this work, we introduce machine learning surrogates that learn physical dynamics. Specifically, we benchmark three operator-learning surrogates for the Single Particle Model (SPM): Deep Operator Networks (DeepONets), Fourier Neural Operators (FNOs) and a newly proposed parameter-embedded Fourier Neural Operator (PE-FNO), which conditions each spectral layer on particle radius and solid-phase diffusivity. We extend the comparison to classical machine-learning baselines by including U-Nets. Models are trained on simulated trajectories spanning four current families (constant, triangular, pulse-train, and Gaussian-random-field) and a full range of State-of-Charge (SOC) (0% to 100%). DeepONet accurately

\* Corresponding author at: Department of Mechanical Engineering, Imperial College London, London, SW7 2AZ, United Kingdom.

\*\* Corresponding author at: Center for Ageing, Reliability and Lifetime Prediction of Electrochemical and Power Electronic Systems (CARL), RWTH Aachen University, Campus-Boulevard 89, Aachen, 52074, Germany.

E-mail addresses: [a.panahi25@imperial.ac.uk](mailto:a.panahi25@imperial.ac.uk) (A.A. Panahi), [weihan.li@isea.rwth-aachen.de](mailto:weihan.li@isea.rwth-aachen.de) (W. Li).

<sup>1</sup> Equal contributions.

replicates constant-current behaviour but struggles with more dynamic loads. The basic FNO maintains mesh invariance and keeps concentration errors below 1 %, with voltage mean-absolute errors under 1.7 mV across all load types. Introducing parameter embedding marginally increases error but enables generalisation to varying radii and diffusivities. PE-FNO executes approximately 200 times faster than a 16-thread SPM solver. Consequently, PE-FNO's capabilities in inverse tasks are explored in a parameter estimation task with Bayesian optimisation, recovering anode and cathode diffusivities with 1.14 % and 8.4 % mean absolute percentage error, respectively, and 0.5918 percentage points higher error in comparison with classical methods. These results pave the way for neural operators to meet the accuracy, speed and parametric flexibility demands of real-time battery management, design-of-experiments and large-scale inference. PE-FNO outperforms conventional neural surrogates, offering a practical path towards high-speed and high-fidelity electrochemical digital twins.

## 1. Introduction

Physics-based models of lithium-ion batteries have become indispensable for electrified transportation, grid energy storage and second-life applications. These models support diverse tasks such as fast-charging optimisation [1], design of experiments [2] and health diagnostics [3,4].

By describing mass-, charge-, and energy- conservation within the porous electrodes and electrolyte, battery models reduce to systems of coupled diffusion–migration and charge-transfer Partial Differential Equations (PDEs). The Doyle-Fuller-Newman (DFN) model remains the gold standard for such physics-based modelling. However, solving its coupled PDEs still takes 3 ms to 115 ms per cycle, even with the fastest open-source solvers under isothermal constant-current conditions, making it challenging for tasks requiring broad condition coverage, degradation analysis, or rapid parameter sweeps. [5,6]

To enable battery digital twins that can operate alongside physical cells and inform real-time decisions, further acceleration beyond the DFN is necessary. Conventional strategies address this by developing surrogate models that retain the governing physics while simplifying geometric or dimensional complexity. The Single Particle Model (SPM), and its electrolyte-enhanced variant SPM with electrolyte dynamics (SPMe), exemplify this strategy by homogenising electrode microstructure and simplifying electrolyte dynamics [7]. Nevertheless, reliance on spatio-temporal grids often leads to computational bottlenecks, particularly when numerous forward simulations are needed—for example, in large-scale parameter estimation [8] or embedded model-predictive control [9]. In parallel, data-driven methods have become omnipresent in battery research, delivering advances in State-of-Charge (SOC) and state-of-health estimation, fast-charging strategies, closed-loop experimentation, and design optimisation [10–19]. Yet, despite their success, purely data-driven models often struggle to accurately extrapolate beyond their training domain and provide limited physical interpretability, making data-driven models less suitable for safety-critical applications [20].

A promising direction is to embed physical knowledge directly into machine learning architectures. This approach, often referred to as physics-informed, physics-inspired or scientific machine learning, seeks to combine neural networks with electrochemical theory to achieve the simulation speed of data-driven surrogates while retaining physical interpretability.

Within scientific machine learning, several strategies now incorporate physical constraints into neural networks. Most prominently, Physics-Informed Neural Networks (PINNs) embed the governing PDE residuals into the loss function by leveraging automatic differentiation in deep learning frameworks. This ensures that the learned solution adheres to the same physical laws as the original system [21,22]. In battery modelling, PINNs have been successfully applied to parameter estimation [23], state estimation [24,25], long-horizon health prognosis [26,27] and thermal modelling [28,29]. Yet, their reliance on specific boundary conditions, such as the applied current in physics-based models, often necessitates custom retraining or costly fine-tuning when drive cycles change. When the governing physics are only partially known, another strategy of scientific machine learning, coined

**Universal Differential Equations**, introduces learnable correction terms to known physical models, allowing neural components to fill gaps in the physics. These hybrid differential equations can be trained end-to-end using experimental data to augment incomplete mechanistic knowledge [30]. This paradigm has been used in [31,32] to increase modelling accuracy of reduced-order models and in [33] to model ageing modes such as solid electrolyte interphase (SEI)-layer growth and pore clogging.

In a similar vein, computationally challenging parts of the battery PDEs, such as root-finding in the algebraic equation of the DFN model, have been replaced by neural networks, allowing for higher speed [34]. Consequently, the idea has been raised to replace simulation altogether, by directly learning the solution operator of the PDE itself. This idea, known as operator learning, uses classical solvers to obtain simulation data of the PDE of interest, based on many different boundaries, initial conditions and parameters and learns the operator described by this mapping in function space [35].

Among various operator-learning architectures developed to date, two stand out as foundational: Deep Operator Network (DeepONet) and Fourier Neural Operator (FNO). DeepONet represents the input field as a vector of sensor values, interprets these as coefficients in a learnable function space, and computes the output through an inner product between this coefficient vector and a set of basis functions generated by a trunk-network at the query point [36].

DeepONets can be extended to incorporate physical constraints by embedding residual terms from the governing equations into the loss function, yielding Physics-Informed DeepONets (PI-DeepONets). This approach preserves the flexibility of operator learning while imposing physical laws. For instance, Zheng et al. [37] demonstrated a composite surrogate for a polynomial-electrolyte SPM using three DeepONets: two model the mapping from current to concentration for the positive and negative electrodes, while a third maps the resulting concentration fields to terminal voltage. Although replacing the algebraic voltage relation with a neural mapping is arguably unnecessary. To further improve physical interpretability, solid-phase diffusivities are introduced as explicit inputs to each branch network and treated as constant parameters. Embedding a residual physics loss in the concentration networks improves accuracy by enforcing consistency with the PDEs. Because the full surrogate is differentiable, it supports gradient-based recovery of diffusivities directly from voltage data. However, the model remained constrained in scope, having been trained solely on linearly ramped single charge–discharge cycles at a fixed SOC, limiting its generalisation to broader operating conditions. The same authors later reformulated their approach as a state-space propagator [38]. Here, a single PI-DeepONet advances the battery states over time, enabling simulations from arbitrary initial SOC and thus relaxing prior constraints on operating conditions. This gain in generality, however, comes at the expense of input diversity; the network is trained and evaluated solely under constant-current profiles and for a fixed parameter set.

To address this limitation, Brendel et al. [39] extended the method to cover a wide range of applied currents by training PI-DeepONets directly on solutions of the SPM equations, bypassing the need for synthetic or experimental data. Their model incorporates scalar diffusivities as inputs via the trunk network, while spatial–temporal coordinates are lifted into a higher-dimensional feature space using multi-scale

spatio-temporal Fourier-feature embeddings. This enables the network to handle a broad, two-decade range of diffusivity values. The resulting surrogate allows for efficient computation of Fisher information matrices, accelerating Design-of-Experiments workflows by facilitating fast gradient evaluations. A key finding across these efforts is that improved generalisation, with respect to both state and parameter spaces, requires significantly longer training times, typically in the range of 6 h to 23 h [37,39].

In contrast to DeepONets, the FNO framework assumes that the underlying operator is realised through integrating with a translation-invariant convolutional kernel, reducing the problem to multiplication in Fourier space. This approach avoids explicit learning of the operator across spatial locations, as the kernel values are directly learned in frequency space. By transforming the input into Fourier space, applying the learned kernel, and then performing an inverse transform, FNOs achieve a spectral convolution across the geometry [40]. Notably, FNOs enable mesh-free extrapolation by learning the operator at coarse resolution and then inferring fine-mesh solutions, which is a property critical for computational efficiency. As a result, FNOs have been classified as “true” neural operators [35], while DeepONets are considered neural regression surrogates due to their resolution dependency in the input domain.

The basic FNO has been further extended by a parameter-embedding logic first introduced by Takamoto et al. [41], which allows the model to handle particle-dependent diffusivity and radius parameters. These scalar inputs are passed through a Multi-Layer Perceptron (MLP), producing channel-wise modulation factors that scale the lifted input. Each Fourier layer is conditioned on these parameters, while the core architecture and learned convolution kernel remain unchanged. This design enables the model to generalise across different particle characteristics without retraining.

To the authors’ best knowledge, no prior work has applied the FNO to learn the solution operator of battery PDEs. The only related study [42] repurposed the architecture as a lithium-ion battery state estimator; an application that is fundamentally distinct due to the ill-posed nature of state estimation and the estimator’s mathematical formulation. The main contributions of this work are as follows:

- (i) The introduction of the first neural-operator surrogate that captures the full spatio-temporal solution of an electrochemical lithium-ion battery model, independent of any fixed mesh or time discretisation.
- (ii) A compact, physically motivated parameter-embedding scheme that maintains surrogate accuracy across a broad range of SOC, dynamic current profiles, particle radii, and solid-phase diffusion coefficients.
- (iii) Comprehensive timing evaluations demonstrating one to two orders of magnitude speed-up over state-of-the-art multithreaded SPM solvers, while maintaining high fidelity.
- (iv) Benchmarking a high-capacity U-Net baseline as a strong classical reference, highlighting the operator surrogates’ resolution invariance and the resulting practical advantage.
- (v) Deployment of the surrogate within a Bayesian optimisation-based parameter estimation task, providing the first analysis of how forward-model errors propagate to inverse-problem accuracy.

## 2. Methods

### 2.1. Electrochemical model

In this work, the SPM described in [7] is used, where the assumptions are made that the electrical conductivity of the electrolyte is sufficiently high, and the timescale for lithium ion migration within the electrolyte is negligible compared to the typical timescale of a charge or discharge. Consequently, the electrolyte concentration is treated

as constant. The battery dynamics are modelled section-wise in the negative and positive domains. As such, the index  $k \in \{n, p\}$  refers to the domain of interest. Lithium transport within each particle is governed by the diffusion equation:

$$\frac{\partial c_k}{\partial t} + \frac{1}{r^2} \frac{\partial}{\partial r} (r^2 N_k) = 0, \quad (1a)$$

$$N_k = -D_k(c_k) \frac{\partial c_k}{\partial r}, \quad 0 \leq r \leq R_k, \quad k \in \{n, p\}, \quad (1b)$$

Here,  $c_k$  is the solid-phase lithium concentration in electrode  $k$ . For simplicity, we drop the additional subscript  $s$  and use  $c_k$  throughout. The solid-phase diffusivity  $D_k(c_k)$  is taken as constant in this work,  $D_k(c_k) \equiv D_k$ . The dimensionless radii and concentrations are defined as  $r'_k = r/R_k$  and  $c'_k = c_k/c_{k,\max}$ , respectively, where  $c_{k,\max}$  is the maximum lithium concentration in the respective electrode. The cell voltage then equates to:

$$V(t) = \underbrace{U_p(c'_p) \Big|_{r'_p=1} - U_n(c'_n) \Big|_{r'_n=1}}_{\text{Open circuit voltage}} - \underbrace{\frac{2RT}{F} \left( \sinh^{-1} \left( \frac{I(t)}{a_p j_{0,p} L_p} \right) + \sinh^{-1} \left( \frac{I(t)}{a_n j_{0,n} L_n} \right) \right)}_{\text{Reaction overpotentials}}, \quad (1c)$$

where  $U_k(\cdot)$  denotes the equilibrium potential of electrode  $k$  as a function of surface stoichiometry,  $R$  is the universal gas constant,  $T$  is the cell temperature and  $F$  is Faraday’s constant. The key geometric/electrode parameters are:  $L_k$ : thickness of electrode  $k$ ,  $a_k$ : specific interfacial area of electrode  $k$ . The exchange-current density is modelled as

$$j_{0,k} = \sqrt{c'_k (1 - c'_k)} \Big|_{r'_k=1}, \quad k \in \{n, p\}. \quad (1d)$$

The model’s boundary and initial conditions are given by:

$$N_k = 0, \quad \text{on } r = 0, \quad (1e)$$

$$N_k = \begin{cases} \frac{IR_k}{3\epsilon_k FL_k A}, & \text{on } r = R_k, \quad k = n, \\ -\frac{IR_k}{3\epsilon_k FL_k A}, & \text{on } r = R_k, \quad k = p, \end{cases} \quad (1f)$$

$$c_k(r, 0) = c_{k0}(r), \quad k \in \{n, p\}, \quad (1g)$$

with  $I > 0$  representing current flowing *into* the cell (charging). Here  $c_{k0}(r)$  is the initial uniform solid concentration in electrode  $k$  and the geometric/electrode parameters are:

- $A$ : cell cross-sectional area,
- $\epsilon_k$ : solid-phase volume fraction in electrode  $k$ .

A schematic of the SPM is visualised in Fig. 1. In this work, we employ a LiFePO<sub>4</sub>–graphite pouch cell parametrised by Prada et al. [43] as the reference chemistry. For the negative electrode, where the OCP curve is not reported, values are adopted from the LG M50 NMC811 cell characterisation of Chen et al. [44]. Parameter sets for both chemistries are given in the Appendix. Although LiFePO<sub>4</sub> is known to lithiate via a two-phase mechanism rather than the solid-solution behaviour assumed by the SPM, we retain the conventional formulation here because the objective of this paper is to benchmark the estimation methodology; this simplification does not influence the methodological insights that follow.

### 2.2. Operator learning

For each electrode  $k \in \{n, p\}$ , let  $X_k = [0, R_k] \times [0, T] \subset \mathbb{R}^2$  denote the space–time domain. Given an applied current profile  $I : [0, T] \rightarrow \mathbb{R}$  and an initial concentration  $c_{0,k} : [0, R_k] \rightarrow \mathbb{R}$ , we define  $a = (I, c_{0,k}) \in$

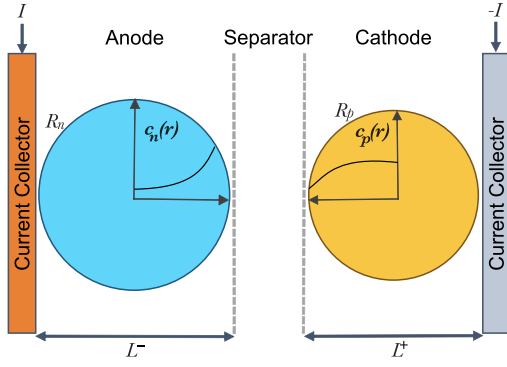


Fig. 1. Single Particle Model during a charging process.

$\mathcal{A} = I \times C_0$ , where  $I$  and  $C_0$  are the corresponding function spaces. For each  $k \in \{n, p\}$ , an operator is learned:

$$G_k : \mathcal{A} \longrightarrow C_k = C(X_k), \quad c_k(\cdot, \cdot) = G_k(I, c_{0,k}),$$

where  $c_k(r, t)$  solves Eqs. (1a)–(1g) on  $X_k$ . Let  $\mathcal{D}_k$  be the data-generating distribution over inputs  $a = (I, c_{0,k})$ . We train an electrode-specific surrogate  $G_{\theta_k} \approx G_k$  by minimising their normalised error on the  $L_2$ -norm over the entire electrode domain and simulation horizon:

$$\min_{\theta_k \in \Theta} \mathbb{E}_{(I, c_{0,k}) \sim \mathcal{D}_k} \left[ \frac{\|G_{\theta_k}(I, c_{0,k}) - G_k(I, c_{0,k})\|_{L_2(X_k)}}{\|G_k(I, c_{0,k})\|_{L_2(X_k)}} \right]. \quad (2)$$

where  $N$  denotes the number of training samples.

In practice, we approximate (2) with an empirical average using i.i.d. samples  $\{(I_j, c_{0,k,j})\}_{j=1}^N \stackrel{\text{i.i.d.}}{\sim} \mathcal{D}_k$ :

$$\min_{\theta_k \in \Theta} \frac{1}{N} \sum_{j=1}^N \frac{\|G_{\theta_k}(I_j, c_{0,k,j}) - G_k(I_j, c_{0,k,j})\|_{L_2(X_k)}}{\|G_k(I_j, c_{0,k,j})\|_{L_2(X_k)}}. \quad (3)$$

To ensure that we operate within ranges where the SPM itself accurately captures the underlying physics, the current profiles are clipped to  $-1.5C \leq I(t) \leq 1.5C$ .

To cover and extend earlier current spaces [37,39], we sample four families:

- **CC**: constant current with random amplitude,
- **TRI**: triangular ramp ( $0 \rightarrow \pm I_{\max} \rightarrow 0$ ),
- **PLS**: rectangular pulse trains with random pulse count, duty cycle and sign,
- **GRF**: smoothly varying profile from a periodic Gaussian random field.

Exact sampling rules are summarised in the Appendix. Training pairs  $\{(I_j, c_{0,k,j}), c_{k,j}\}_{j=1}^N$  are generated with PyBaMM [45]. Because the negative and positive particles interact only through the shared input current, independent learning of  $G_n$  and  $G_p$  is both natural and computationally convenient.

**DeepONet.** In the DeepONet framework, the operator learning task is realised by combining two components: a *branch net* that learns feature-dependent basis coefficients  $\{\beta_i(a)\}_{i=1}^p$  for the current–concentration pairs, and a *trunk net* that encodes geometry dependent basis functions  $\{\tau_i(r, t)\}$  for spatio-temporal sampling points. Their inner product  $\langle \beta(a), \tau(r, t) \rangle$  is itself a function of  $a$  evaluated on  $(r, t)$  and predicts the concentration:

$$c_k(r, t) \approx G_{\theta_k}(a)(r, t) = \sum_{i=1}^p \beta_{k,i}(a) \tau_{k,i}(r, t), \quad k \in \{n, p\}. \quad (4)$$

**FNO.** The Fourier Neural Operator treats the discretised input

$$a(r, t) = [I(t), c_{0,k}(r), r/R_k, t/T] \in \mathbb{R}^4$$

as a four-channel image over the  $(r, t)$  grid. A linear lifting layer  $P$  maps each grid point to a  $d$ -dimensional latent vector  $v^0 = Pa$ . An *FNO block* then performs a spectral convolution combined with a point-wise linear update:

$$v^{\ell+1}(x) = \sigma(W v^{\ell}(x) + \mathcal{F}^{-1}(R_{\phi} \odot \mathcal{F} v^{\ell})(x)), \quad \ell = 0, \dots, L-1, \quad (5)$$

where  $\mathcal{F}$  and  $\mathcal{F}^{-1}$  are the forward and inverse Fourier transforms,  $R_{\phi} \in \mathbb{C}^{k_{\max} \times d \times d}$  is the kernel representation in frequency space storing learnable complex weights for the lowest  $k_{\max}$  modes, and  $\odot$  denotes element-wise multiplication. The pointwise weight matrix  $W \in \mathbb{R}^{C_{\text{out}} \times C_{\text{in}}}$  act as a  $1 \times 1$  kernel, mixing channels identically at every grid point and carrying local information through the network. In combination with the activation function  $\sigma$ , this design captures both low-frequency global correlations (via the spectral convolution) and high-frequency content.

By stacking  $L$  such FNO blocks, we obtain a latent field  $v^L$ . A final projection  $Q$  maps this latent field back to a single channel and gives the prediction for  $c_k(r, t)$ .

**PE-FNO.** To cover the family of SPM problems that arise under varying solid particle diffusivity and radii parameters, we adopt the parameter-embedding strategy of Takamoto et al. [41].

In this framework, three parallel branches are applied to the lifted input of the FNO:

- a  $1 \times 1$  convolution that mixes channels;
- a depth-wise (DW)  $3 \times 3$  convolution that injects local-neighbourhood information;
- a Fourier layer that supplies the first global interaction.

The three outputs are summed and then modulated by the material parameters by multiplication with their latent-space representations through a two-layer MLP.

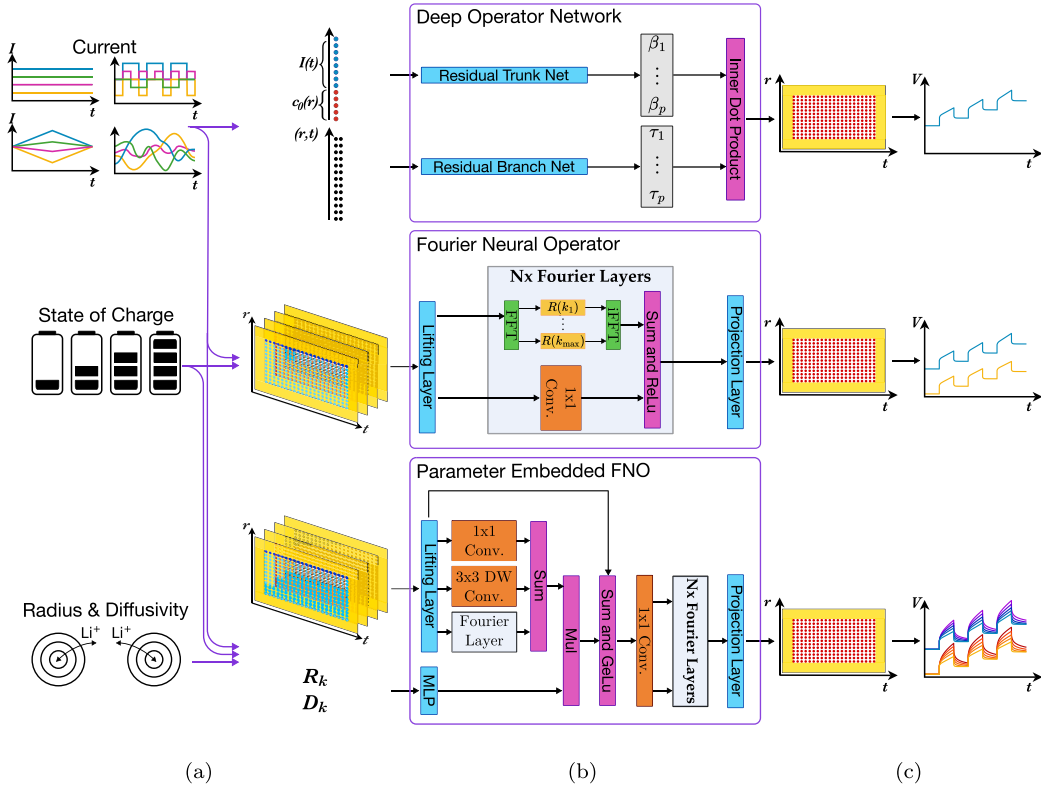
The  $3 \times 3$  depth-wise convolution allows the modulation to influence each node through its immediate neighbours. Mathematically, the layer processes the neighbourhood in a manner analogous to a finite-difference method of a second-order derivative, making it a natural fit for modelling diffusion processes.

Distinct from that, the  $1 \times 1$  convolution and the Fourier layer allow the parameters to affect local and global effects between the channels. This design ensures that geometric parameters, such as particle radius, can influence both the boundary channel at  $r = R_k$  and the interior fields in one stroke, without further architectural adjustments. Together, these mechanisms provide the network with the flexibility to accurately represent the physical effects of varying  $D_k$ ,  $R_k$  without altering the core FNO.

Note that formally, the input to the PE-FNO becomes  $a = (I, c_{0,k}, D_k, R_k) \in I \times C_0 \times D \times R$  where  $D, R$  are the respective parameter spaces. Fig. 2(a) sketches the resulting input spaces and their respective flow into the three architectures, which are themselves illustrated in Fig. 2(b).

**U-Net.** The U-Net architecture is implemented similar to its original implementation in [46]. We replace standard convolutional blocks with depthwise-separable layers — a per-channel  $3 \times 3$  spatial convolution followed by a  $1 \times 1$  pointwise convolution for channel mixing — to reduce parameters and FLOPs while maintaining accuracy and keeping in line with the spectral convolution of the FNO [47]. Each block stacks two such layers with activations. Downsampling is performed by stride-2 depthwise-separable convolutions, forming a learnable reduction. Upsampling uses stride-2 transposed convolutions. The encoder uses widths  $\{1, 2, 4\} \times C$  with skip connections into a symmetric decoder. The U-Net is size matched to have the same amount of parameters as the FNO as this is its most comparable architecture. More detail about the U-Net can be found in the Appendix.





**Fig. 2.** Overview of the architectures. (a) *Input design*. The applied current  $I(t)$  and the initial solid concentration  $c_{0,k}(r)$  are broadcast to a common  $(r, t)$  grid; yellow margins indicate zero-padding cells. For the PE-FNO the scalar diffusivity  $D_k$  and particle radius  $R_k$  is fed to an MLP at this stage. (b) *Operator networks*. **DeepONet**: identical residual branch/trunk nets produce basis coefficients and functions that are combined via a coefficient-basis sum. **FNO**: a stack of  $N$  Fourier layers with a  $1 \times 1$  channel-wise residual path processes the lifted tensor. **PE-FNO**: a parameter-embedding block ( $1 \times 1$  conv, depth-wise  $3 \times 3$  conv, single Fourier layer) is multiplied to a feature lifted  $D_k$  before the standard Fourier stack. (c) *Projection head*. A final linear projection returns the predicted concentration field  $c_k(r, t)$ , which is subsequently converted to terminal voltage  $V(t)$ .

### 2.3. Practical implementation

All model parameters, including the initial SOC, are sampled with a Sobol sequence. For the parameters, we sample (omitting units)  $D_k \in [10^{-18}, 10^{-14}]$ ,  $R_k \in [4 \times 10^{-6}, 1.5 \times 10^{-5}]$  and  $R_p \in [10^{-8}, 1.5 \times 10^{-5}]$ , then apply a logarithmic transform and normalise to  $[-1, 1]$  before passing them to the network. SOC values are rounded to the nearest practically relevant percentage. In this work, we train using 1 h simulations; anecdotal experiments indicated that different simulation lengths yield similar model performance.

Branch and trunk nets of the DeepONet are identical ResNets [48]. The number of basis functions is set to  $p = 16$  for *CC/TRI* inputs and  $p = 64$  for the more irregular *PLS/GRF* inputs. For both FNO variants, every input channel must live on the same  $(r, t)$  grid, so the current  $I(t)$  is broadcast along the  $r$ -axis and the initial concentration  $c_{0,k}(r)$  along the  $t$ -axis. To accommodate non-periodic signals, we zero-pad the input in  $(r, t)$  and include the coordinate channels  $(r/R_k, t/T)$  exactly as in [40]. Numerical values for  $k_{\max}$ , depth, width, and parameter-embedding modulation are given in Table 1. For the PE-FNO, different numbers of frequency modes  $(k_r, k_t)$  are kept for each dimension. A schematic of the input design can be seen in Fig. 2(a).

Branch and trunk nets of the DeepONet are identical ResNets [48]. The number of basis functions is set to  $p = 16$  for *CC/TRI* inputs and  $p = 64$  for the more irregular *PLS/GRF* inputs. This choice reflects signal complexity: *CC/TRI* waveforms are band-limited and smoothly varying in time, whereas *PLS/GRF* introduce higher temporal frequencies; increasing  $p$  improves the branch basis' ability to represent these inputs without overfitting.

For both FNO variants, every input channel must live on the same  $(r, t)$  grid, so the current  $I(t)$  is broadcast along the  $r$ -axis and the initial

concentration  $c_{0,k}(r)$  along the  $t$ -axis. This mirrors the physics:  $I(t)$  is spatially uniform at the particle scale, while  $c_{0,k}(r)$  is time-invariant at  $t=0$  but radially varying across particles.

To accommodate non-periodic signals, we zero-pad the input in  $(r, t)$  and include the coordinate channels  $(r/R_k, t/T)$  exactly as in [40]. Padding reduces wrap-around artefacts in the Fourier blocks for finite-horizon, non-periodic drive, and coordinate channels provide the network with non-dimensional position and time, which not only stabilises training across particles with different  $R_k$  and horizons  $T$ , but also aids in encoding the input distinctly for different resolutions.

Numerical values for  $k_{\max}$ , depth, width, and parameter-embedding modulation are given in Table 1. For the PE-FNO, different numbers of frequency modes  $(k_r, k_t)$  are kept for each dimension. We use fewer radial modes and more temporal modes,  $(k_r, k_t) = (5, 20)$ , because spherical diffusion smooths the concentration profile in  $r$  (low spatial bandwidth), whereas dynamic loads (*PLS/GRF*) inject higher temporal frequencies that must be resolved. The PE-FNO core is made slightly wider/deeper than the plain FNO to absorb the modulation from the parameter-embedding while preserving a receptive field that spans the full time horizon. A schematic of the input design can be seen in Fig. 2(a).

The size of each training set is chosen to match the number of physical degrees of freedom the corresponding learned operator has to cover, using 2,200 data samples as a baseline, which is in accordance with the sample sizes used in the original works [36,40]. During the training, it became apparent that DeepONet can attend to different initial concentrations in general, but not to a varying range in one training process. Therefore, DeepONet is trained only for a single diffusivity pair and a fixed initial SOC, so the baseline 2,200 current profiles are sufficient. In contrast, FNO must generalise over a distribution of

**Table 1**

Key hyper-parameters for the three architectures. Empty entries (–) indicate that the setting does not apply.

Hyper-parameter	DeepONet	FNO	PE-FNO
Core width	500	32	64
Core depth	11	6	8
Basis functions $p$	16 / 64	–	–
$k_{\max}$	–	10	(5,20)
Last input dim.	95 (branch) 20 (trunk)	4	4
PE layer width	–	–	32
PE layer depth	–	–	2
Padding $(r, t)$	–	(+2, +5)	(+2, +5)

initial SOC fields and therefore receives 11,000 samples. Practically, this five-fold increase balances the four current families and attends to the introduction of the initial charge distribution as a functional input to the network. As the PE-FNO additionally learns the effect of varying parameters, the data budget is increased to 33,000 to populate the four-dimensional parameter space. Each dataset is randomly split into a 90%/10% training and test set, respectively.

Training was done using the Adam optimiser [49] with a cosine learning-rate schedule [50]: the rate is linearly warmed from 0 to  $10^{-2}$  during the first epoch, then decays along a cosine curve to  $10^{-4}$  over the remaining epochs, ending with a small step size for fine tuning. All models are implemented in JAX and Flax [51,52]. The voltage and boundary expressions in Eqs. (1c) and (1e) are coded in pure JAX, enabling jit-compilation and end-to-end differentiability. Fig. 2(c) visualises this step.

To avoid overweighting trajectories with larger absolute concentrations, we minimise the *normalised*  $L_2$  error, averaged over the mini-batch:

$$nL_2 = \frac{1}{N} \sum_{j=1}^N \frac{\|\hat{\mathbf{c}}^{(j)} - \mathbf{c}^{(j)}\|_2}{\|\mathbf{c}^{(j)}\|_2}, \quad (6)$$

where  $N$  is the batch size and  $\hat{\mathbf{c}}^{(j)}, \mathbf{c}^{(j)}$  are the flattened predicted and reference concentration fields of sample  $j$  respectively. Concentrations are linearly scaled to  $[0, 1]$  and currents to  $[-1, 1]$  before training begins.

## 2.4. Bayesian identification of electrode diffusivities

Finally, we embed the PE-FNO in a Bayesian parameter-identification loop to demonstrate its utility for inverse problems. We emphasise that the core contribution of this work is the systematic assessment of the surrogate’s accuracy, generalisability, and run-time performance; refining its inverse-problem capabilities is a logical next step once the emulator reliably reproduces the full state space of the SPM. Nevertheless, an “out-of-the-box” Bayesian estimation experiment is included here to provide a preliminary analysis of how well the trained operator supports downstream inference tasks and to assess the accuracy of forward vs. inverse inference.

Let  $\rho = (\log_{10} D_n, \log_{10} D_p) \in \Omega = [-18, -14]^2 \subset \mathbb{R}^2$  represent the unknown log-diffusivities. For a given current profile  $I(t)$  and initial state  $c_{0,k}(r)$ , the trained PE-FNO returns the full spatio-temporal concentration field  $c_k(r, t; \rho)$ . From this, the electrode surface concentrations are used to compute the cell voltage  $V_{\text{pred}}(t; \rho)$ , through Eq. (1c).

The inverse problem is posed as a bounded minimisation of the relative  $L_2$  error:

$$nL_2(\rho) = \frac{\|V_{\text{pred}}(\cdot; \rho) - V_{\text{data}}\|_2}{\|V_{\text{data}}\|_2}, \quad (7)$$

which is evaluated on the discrete voltage trace. A Gaussian-process surrogate  $m(\rho)$  is initialised with  $n_0 = 12$  Sobol points in  $\Omega$  and sequentially refined by choosing the next sample  $\rho^*$  as the maximiser of the expected improvement acquisition function. The optimisation

loop terminates after  $n_{\text{tot}} = 60$  evaluations. The best candidate  $\rho_{\min} = \arg \min nL_2(\theta)$  provides the point estimate of  $(D_n, D_p)$ . The same procedures are repeated for the classical simulation with PyBaMM and compared against each other.

## 3. Results

This section first quantifies the forward accuracy of the three surrogates, then benchmarks their runtime performance, and finally demonstrates how the PE-FNO supports a Bayesian diffusivity-estimation task.

### 3.1. Accuracy of the learned operators

Added on top of the  $nL_2$ , three complementary metrics are reported (explicit formulae are given in the Appendix). Each metric communicates a similar but distinct idea about the prediction accuracy:

- **Relative  $L_2$  error** ( $nL_2$ ). Already used for the training, this error scales by the value of the ground truth, giving a clear metric of learning capability irrespective of magnitude.
- **RMSE**. Root-mean-square error in mV; penalises larger deviations more strongly than MAE and is widely used in battery-model benchmarking.
- **MAE**. Average absolute error in mV; offers an intuitive “per-sample” discrepancy that is less influenced by outliers than RMSE.
- **Relative  $L_\infty$  error** ( $nL_\infty$ ). Ratio of the worst-case pointwise error to the maximum reference value. This is especially important for detecting peak mismatches that may violate safety margins.

Together, these metrics provide a comprehensive view of model performance: unit-aware deviations (RMSE, MAE), normalised averages ( $nL_2$ ), and worst-case behaviour ( $nL_\infty$ ). For concentration errors in Fig. 3, metrics are computed separately for the anode and cathode and then averaged to obtain a single figure.

During data generation, we *did not* enforce the usual SPM safety constraints (e.g., cut-off voltages, admissible stoichiometries, etc.). As a result, a small subset of the simulated trajectories drift marginally outside the real application domain—e.g. surface concentrations exceeding their respective maximum and minimum concentration limits or voltages crossing the 2.5 V/3.65 V limits. We note that allowing this can have effects on saturation and degradation dynamics due to their path dependencies with the fundamental equation variables. This should be taken into account when one extends this work to models that consider more complex or intricate dynamics than the vanilla SPM.

Nevertheless, these borderline cases were deliberately retained in the *training* pool for two reasons:

1. **Edge coverage**. Samples that lie just beyond the admissible manifold provide support points near its boundary and teach the surrogate how the solution behaves as the system approaches extreme states.
2. **Regularisation by extrapolation**. Exposure to slightly out-of-domain inputs encourages the network to learn a smooth continuation of the operator, which improves interpolation in the physical region.

For the accuracy study, these out-of-domain trajectories are removed from the test set, ensuring that all reported metrics correspond strictly to valid operating conditions. We emphasise that the trade-off is that a model trained this way has, on average, a higher error when it is later evaluated *only* on clean data than a model that was exposed to clean data exclusively. I.e., we allow the model to perform worse on average in order to cover all in-domain space at good accuracy. It is for this reason that the  $nL_\infty$  error provides a valuable insight into model capabilities.

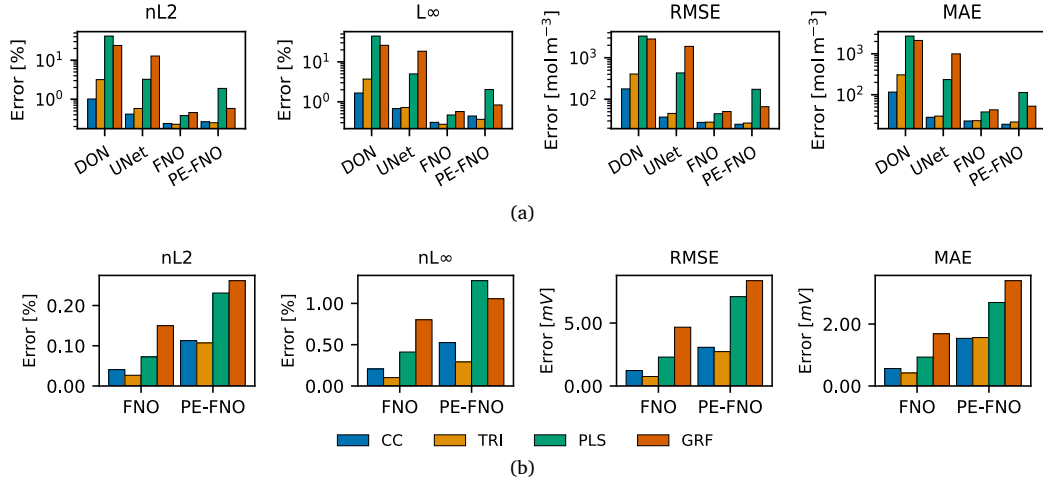


Fig. 3. Error metrics for (a) concentration and (b) voltage.

**Concentration fields (Fig. 3a).** DeepONet shows the largest discrepancies, with  $nL_2 = 1.0\%$  even for the simple CC profile, with this increasing to  $42\%$  ( $RMSE \approx 3.3 \times 10^3 \text{ mol m}^{-3}$ ) for the highly intermittent PLS excitation. As a convolutional baseline, U-Net attains  $nL_2 \approx 0.42\%$  (CC) and  $0.58\%$  (TRI), but rises to  $3.25\%$  (PLS) and  $12.82\%$  (GRF), with worst-case  $nL_\infty$  of  $18.48\%$ . Moving to FNO comes with a reduction in error by nearly two orders of magnitude. Across all four current families,  $nL_2$  remains below  $0.46\%$  and the worst-case  $nL_\infty$  does not exceed  $0.57\%$ , while also enabling varying SOC. Adding the parameter-embedding (PE-FNO) keeps low errors for CC and TRI profiles (e.g.  $0.27\%$  versus  $0.24\%$   $nL_2$ ), while raising PLS and GRF errors to  $0.58\%$  and  $1.9\%$ , respectively. Even so, PE-FNO retains a ten-fold advantage over DeepONet while delivering parameter awareness that the plain FNO lacks.

**Voltage traces (Fig. 3b).** Voltage errors follow the same ranking. DeepONet and U-Net bars are omitted from the plot for clarity, focusing on the two neural operator variants. The basic FNO achieves sub- $0.15\%$   $nL_2$  for CC, TRI and PLS profiles and  $0.15\text{--}0.45\%$   $nL_\infty$ . PE-FNO incurs a factor-of-two penalty across the board ( $nL_2 \approx 0.11\text{--}0.26\%$ ), resulting in a MAE of  $1.5\text{--}3.4 \text{ mV}$ —further illustrating the trade-off for accommodating parameter variation.

Overall, FNO-based surrogates achieve sub-percent relative errors for both concentration and voltage. The basic FNO is most accurate when diffusivity is fixed, whereas the parameter-embedded version offers robustness to parameter shifts at the cost of a modest increase in prediction error. DeepONet and U-Net trail by one to two orders of magnitude across all metrics and current families, underscoring the advantage of neural operator architectures for physics-rich battery problems.

**Resolution invariance.** All models are evaluated a second time *without retraining* on finer, out-of-distribution grids (85 and 150 samples per time horizon). Fig. 4 reports voltage MAE for CC, TRI, PLS, and GRF inputs. U-Net shows large errors when the grid changes (e.g. CC:  $432\text{--}460 \text{ mV}$ ; TRI:  $544 \text{ mV}$  at 85 and  $267 \text{ mV}$  at 150; PLS:  $501\text{--}411 \text{ mV}$ ; GRF:  $392\text{--}594 \text{ mV}$ ). In contrast, FNO remains in the single-tens of millivolts (85:  $\sim 3\text{--}11 \text{ mV}$ ; 150:  $\sim 15\text{--}35 \text{ mV}$ ), and PE-FNO maintains errors in the low-tens of millivolts (85:  $\sim 35\text{--}59 \text{ mV}$ ; 150:  $\sim 37\text{--}67 \text{ mV}$ ), with only mild drift except for GRF at 150. This invariance is practically useful: a single set of weights can be deployed across devices with different battery management system sampling rates, tolerates minor timestamp jitter or packet delays from cloud latency, and operate on downsampled logs without model-specific retraining. As U-Net fails to deliver on this modality, we will omit it from further analysis.

Simulation versus ground truth for one sample of the GRF is visualised in Fig. 5. Here, all three surrogates are queried at the *same* initial SOC and material parameters; the less capable architectures are therefore trained at the states and parameters they cannot attend to.

### 3.2. Speed benchmark

All timings were obtained on a workstation equipped with an AMD Ryzen Threadripper PRO 5965WX CPU (48 physical cores) and an NVIDIA RTX A4000 GPU. The configuration reflects a practical deployment scenario where the SPM is executed on the CPU while the learned operators run on the GPU.

**CPU baseline.** The SPM was solved using PyBaMM v24.9.0 with the solver from Andersson et al. [53]. Runs were repeated for  $n_{\text{thr}} \in \{1, 8, 16\}$  threads, and reported figures correspond to wall-clock time divided by the number of trajectories in the batch, such that every entry represents the average runtime of a *single* simulation.

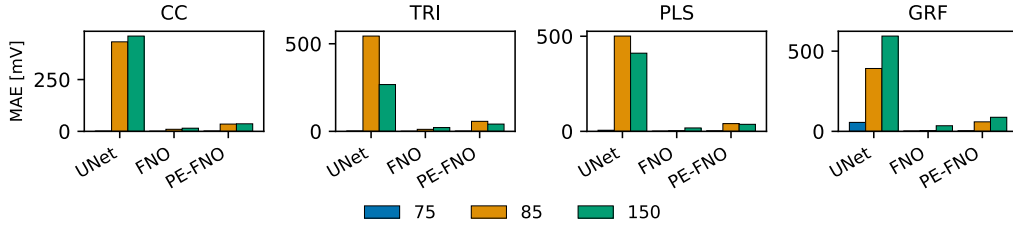
**GPU surrogates.** All neural operators were benchmarked with a fixed inference batch of  $B = 100$  samples, the largest size that fits the PE-FNO into the RTX-A4000's 16 GB VRAM. The raw batch time was divided by  $B$  to obtain a per-trajectory latency directly comparable to the CPU baseline.

Fig. 6 shows that:

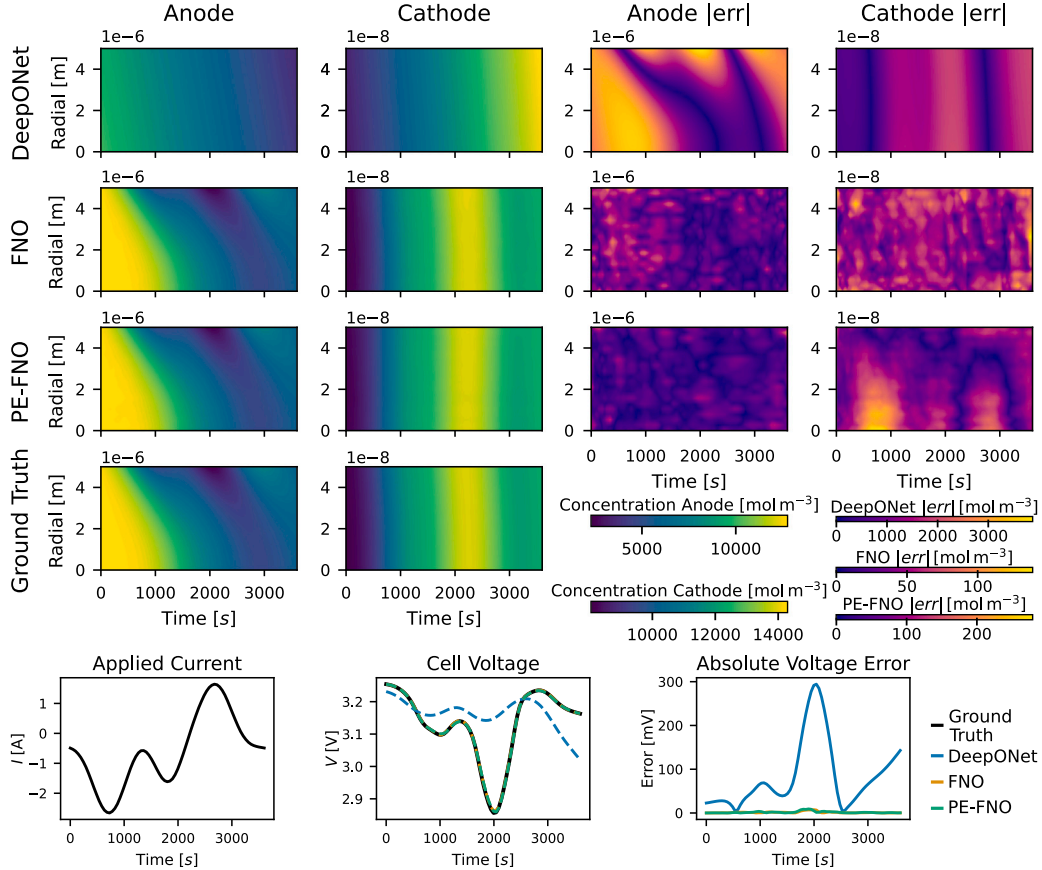
- **DeepONet** achieves the lowest latency,  $\approx 1.6 \mu\text{s}$  per trajectory, delivering a  $> 3,500\times$  speed-up over the CPU baseline.
- **FNO** completes one forward pass in  $10\text{--}15 \mu\text{s}$ ; the FFT and complex multiplications introduce a  $6\text{--}9$  times overhead relative to DeepONet but still yield a  $> 400\times$  speed-up over the numerical baseline.
- **PE-FNO** adds a further  $20\text{--}40\%$  cost for the parameter pre-processing, but remains two orders of magnitude faster than the numerical SPM and is well suited for real-time or embedded applications.

In practice, all surrogates run in sub-milliseconds, enabling latency-critical tasks such as embedded state estimation and inner-loop optimal control.

A fair speed comparison must include the *up-front cost* of producing each surrogate. Table 2 reports the data-generation time and training time needed to reach the test accuracies in Fig. 3. Training a DeepONet requires minimal time ( $\sim 1 \text{ min}$ ), whereas the full PE-FNO takes approximately  $1.5 \text{ h}$  on a single RTX-A4000. Contextualising this with the simulation benchmarks from Fig. 6 leads to the conclusion that PE-FNO becomes advantageous only when the SPM must be evaluated on



**Fig. 4.** Resolution invariance under finer time grids. Voltage MAE (mV) for U-Net, FNO, and PE-FNO on CC, TRI, PLS, and GRF inputs. Models are trained on a 75-sample time grid (shown as a reference) and evaluated *without retraining* on finer, unseen grids of 85 and 150 samples. U-Net errors are large and inconsistent under grid changes, whereas FNO/PE-FNO remain stable at low MAE.



**Fig. 5.** Randomly sampled result for a GRF current input. Rows: DeepONet, FNO, PE-FNO, ground-truth SPM. Columns:  $c_n(r, t)$ ,  $c_p(r, t)$  and their absolute errors; bottom panels give current, voltage, and voltage error. FNO and PE-FNO track the SPM within single digit millivolt range, whereas DeepONet shows spatial artefacts and errors up to 300 mV.

**Table 2**

Data volume and wall-clock cost for training each surrogate on a single RTX-A4000. All times are approximate.

Model	# epochs	Training time	Generation time
DeepONet	15	≈ 1 min	≈ 2 min
FNO	25	≈ 20 min	≈ 11 min
PE-FNO	50	≈ 60 min	≈ 33 min

the order of a million times. A use setting that occurs in typical battery management systems or model predictive control applications.

### 3.3. Inversion capabilities of the PE-FNO

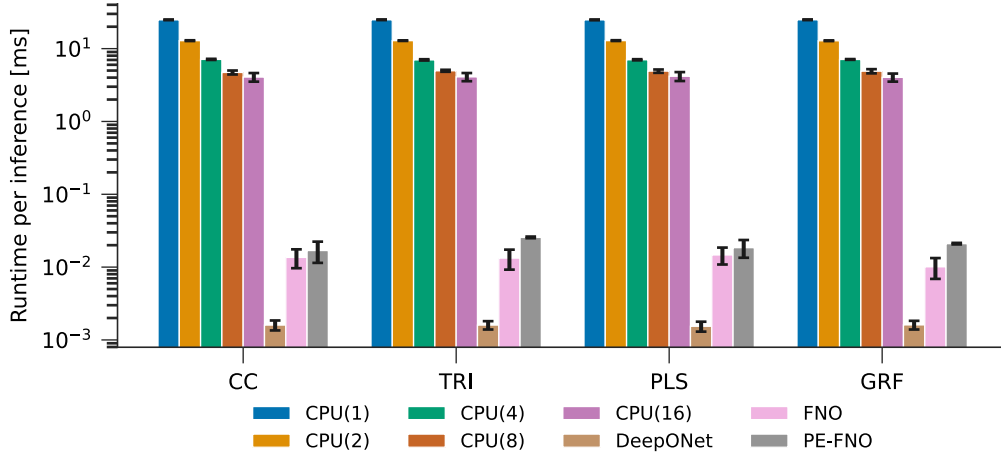
Fig. 7 plots the best-so-far voltage loss (logarithmic) during Bayesian optimisation of  $\log_{10} D_n$  and  $\log_{10} D_p$  for all 1000 samples out of the test set.

PyBaMM converges to a plateau of  $nL_2 \approx 10^{-3}$  after approximately 35 evaluations, whereas PE-FNO settles at  $nL_2 \approx 2 \times 10^{-2}$  within 15 iterations. The gap equals a voltage deviation of 0.049 mV versus 0.56 mV in the RMSE for the depicted median, i.e., an approximately one-decade loss in inverse accuracy for the surrogate.

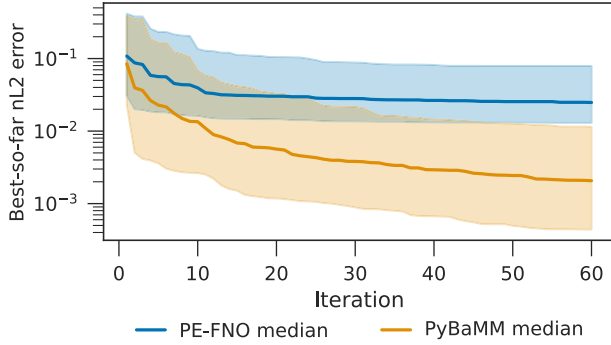
Across the complete sample set, the mean voltage error is  $nL_2 = 0.157\%$  for PyBaMM and 0.491% for the PE-FNO, i.e. a difference of 0.334 percentage points. The surrogate's inverse error is thus about 27% higher than its forward-prediction error ( $nL_2 = 0.262\%$ ), consistent with the fact that the neural operator was only trained on the forward map.

**Recovered parameters.** Table 3 lists the resulting diffusivity errors. For the anode, PE-FNO attains an RMSE = 0.25 dex; eight times less precise than PyBaMM but achieved at millisecond latency. For the cathode, both estimators are limited to  $\approx 1.4$  dex due to the flat LiFePO<sub>4</sub> open-circuit potential, confirming that voltage carries little information about  $D_p$  for this chemistry.





**Fig. 6.** Speed comparison between learned operators and classical solver parallelised to different numbers of cores; all values normalised to a per simulation run basis.



**Fig. 7.** Convergence plot for the voltage fit for estimating  $D_n^*$  and  $D_p^*$  from data. Median in thick line and 25 – 75% ranges over 1000 samples.

**Table 3**  
Diffusivity-estimate error metrics.

Parameter	Model	MAE	RMSE	MAPE [%]
$\log_{10} D_{an}$	PE-FNO	0.1465	0.2524	1.1432
	PyBaMM	0.0191	0.0286	0.1428
$\log_{10} D_{ca}$	PE-FNO	1.1408	1.3971	8.4783
	PyBaMM	1.1163	1.3685	8.2947

#### 4. Discussion

Neural-operator surrogates are most valuable in workflows that require *extensive* simulator sampling — state estimation, real-time control, and design studies — where the cost of generating the training set quickly dwarfs the number of simulation runs needed during deployment. For such applications, it is essential to view a surrogate’s merit holistically: forward accuracy, inverse accuracy, and execution speed must be weighed against training effort, data-generation cost, and, above all, generalisability. By reporting each of these factors side-by-side, the present study paves the way for fair assessments of physics-inspired machine learning frameworks.

Importantly, because the model is mesh-agnostic, the *same architecture* can be deployed on coarse grids for rapid simulation or on fine grids for high-resolution analysis without retraining. This resolution

invariance makes the methodology robust in real applications, where sampling rates may vary across battery-management systems and cloud latency can differ.

We conclude with the remark that the purely data-driven PE-FNO offers a much more favourable balance between training time and accuracy than PINNs and also generalises more robustly across parameters, initial states, and boundary conditions.

Future work should focus on expanding the physical envelope of these surrogates, including extending them to additional parameters and more complex electrochemical models (e.g., DFN). This would capture a wider set of conditions relevant to battery management systems. Likewise, a single operator that copes with constant-current, pulse and stochastic load families would remove the need for family-specific training. Finally, incorporating inverse awareness through joint forward-adjoint training offers a promising route to narrowing the remaining gap in parameter-estimation accuracy while retaining the speed advantage demonstrated here.

#### 5. Conclusion

In this work, we introduced multiple machine learning surrogate models as an alternative to numerical solvers of battery PDEs. These operator-learning architectures were benchmarked for their capabilities of learning the SPM’s solution operator across varying input currents, initial SOC, parameter values, and different resolution. DeepONet was able to learn the solutions for CC and TRI current inputs at a fixed SOC and parameter set, while the FNO extended this capability to PLS and GRF currents and varying SOC, marking the first application of a discretisation-invariant neural operator in lithium-ion battery modelling. Furthermore, a lightweight parameter-embedding module supplies channel-wise scale-shift factors derived from particle radius and solid-phase diffusivities, enabling a single network PE-FNO to cope simultaneously with arbitrary SOC fields, highly dynamic current profiles, and a two-decade range of material parameters. Compared with an average multithreaded PyBaMM SPM solver, the trained PE-FNO is approximately two orders of magnitude faster while still delivering sub-percent errors in both concentration and voltage predictions. Importantly, because the model is mesh-invariant, the *same weights* can be deployed on coarse grids for rapid simulation or on fine grids for high-resolution analysis without retraining.

**Acronyms**

<b>SOC</b>	State-of-Charge
<b>SPM</b>	Single Particle Model
<b>SPMe</b>	SPM with electrolyte dynamics
<b>DFN</b>	Doyle-Fuller-Newman
<b>PDE</b>	Partial Differential Equation
<b>PINN</b>	Physics-Informed Neural Network
<b>DeepONet</b>	Deep Operator Network
<b>PI-DeepONet</b>	Physics-Informed DeepONet
<b>FNO</b>	Fourier Neural Operator
<b>PE-FNO</b>	parameter-embedded Fourier Neural Operator
<b>MLP</b>	Multi-Layer Perceptron

**CRedit authorship contribution statement**

**Amir Ali Panahi:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Conceptualization. **Daniel Luder:** Writing – review & editing, Supervision, Methodology, Conceptualization. **Billy Wu:** Writing – review & editing, Supervision, Funding acquisition. **Gregory Offer:** Writing – review & editing. **Dirk Uwe Sauer:** Writing – review & editing. **Weihan Li:** Writing – review & editing, Supervision, Funding acquisition.

**Code availability**

The code corresponding to this manuscript is hosted at [github.com/alighato/spm\\_neuraloperator](https://github.com/alighato/spm_neuraloperator). We thank Carlos Maximilian Menke for assistance in improving the cleanliness and structure of the code.

**Declaration of competing interest**

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Amir Ali Panahi reports financial support was provided by Shell Research UK Ltd. Billy Wu: Given his role as Editorial Board Member, he had no involvement in the peer review of this article and had no access to information regarding its peer review. Full responsibility for the editorial process for this article was delegated to another journal editor. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

**Acknowledgements**

This work has received funding from the project “SPEED” (03XP0585) funded by the German Federal Ministry of Research, Technology and Space (BMFT) and the project “ADMirABLE” (03ETE053E) funded by the German Federal Ministry for Economic Affairs and Energy (BMWE). The authors thank the support of Shell Research UK Ltd. for the Ph.D. studentship of Amir Ali Panahi and the EPSRC Faraday Institution Multi-Scale Modelling Project (FIRG084).

**Appendix****Parameter values**

Let  $\text{sto}$  be the normalised concentration at the respective particle, then  $U_{OCp}^p$  respective  $U_{OCp}^n$  for [44] are defined by (see Table A.4). :

$$U_{OCp}^p(\text{sto}) = -0.8090 \text{sto} + 4.4875 - 0.0428 \tanh(18.5138(\text{sto} - 0.5542)) - 17.7326 \tanh(15.7890(\text{sto} - 0.3117)) + 17.5842 \tanh(15.9308(\text{sto} - 0.3120)). \quad (\text{A.1})$$

$$U_{OCp}^n(\text{sto}) = +1.9793 \exp(-39.3631 \text{sto}) + 0.2482 \quad (\text{A.2})$$

$$\begin{aligned} & -0.0909 \tanh(29.8538(\text{sto} - 0.1234)) \\ & -0.04478 \tanh(14.9159(\text{sto} - 0.2769)) \\ & -0.0205 \tanh(30.4444(\text{sto} - 0.6103)). \end{aligned}$$

The positive electrode OCP from [43] is given by:

$$U_{OCp}^p(\text{sto}) = +3.4077 - 0.020269 \text{sto} + 0.5 \exp(-150 \text{sto}) - 0.9 \exp(-30(1 - \text{sto})). \quad (\text{A.3})$$

**Current families**

Let  $T_{\max}$  be the experiment horizon and  $n$  the temporal resolution used throughout the data set. Define the uniform grid

$$\mathcal{T} = \{t_i\}_{i=1}^n, \quad t_i = \frac{i-1}{n-1} T_{\max}.$$

(a) *Constant-current family.*

$$I(t) = I_{\text{const}}, \quad t \in [0, T_{\max}],$$

with  $|I_{\text{const}}| \leq 1.5C$ .

(b) *Triangular current family.* Choose  $0 < t_1 < t_2 \leq T_{\max}$  and a peak value  $I_{\Delta} \in [-1.5C, 1.5C]$ ; then

$$I(t) = \begin{cases} I_{\Delta} \frac{t}{t_1}, & 0 \leq t \leq t_1, \\ I_{\Delta} \frac{t_2 - t}{t_2 - t_1}, & t_1 < t \leq t_2, \\ 0, & t > t_2. \end{cases}$$

Setting  $t_1 = 1800$  s and  $t_2 = 3600$  s reproduces the profile used in the numerical experiments.

(c) *Rectangular-pulse family.* A random pulse train  $I_{\text{rect}}(t)$  is generated as follows. Let  $T_{\max}$  be the horizon of the experiment and  $C$  the 1C rate.

1. *Pulse count.* Draw an integer  $N_h \sim \text{Unif}\{1, \dots, 10\}$  (pulses per hour) and set

$$n_p = \max\{1, \lfloor N_h T_{\max}/3600 \rfloor\}.$$

for the total number of pulses

2. *Amplitude.* Select a direction  $s \in \{+1, -1\}$  at random and a magnitude factor  $a \sim \mathcal{U}(0.2, 1.5)$ ; the amplitude equates to:

$$I_p = s a C, \quad |I_p| \in [0.2C, 1.5C].$$

3. *Timing.* The period is  $P = T_{\max}/n_p$ . Draw a duty cycle  $d \sim \mathcal{U}(0.2, 0.7)$  and set the pulse width  $\tau = dP$ . Pulse  $k$  starts at  $t_k = kP$  for  $k = 0, \dots, n_p - 1$ .

The resulting current is

$$I(t) = \begin{cases} I_p, & \exists k: 0 \leq k < n_p, \quad t \in [t_k, t_k + \tau), \\ 0, & \text{otherwise.} \end{cases}$$

(d) *Gaussian-random-field (GRF) family.* For a length scale  $L = 1$  we adopt the periodic squared-exponential covariance

$$k_{\text{per}}(t, t') = \exp[-2 \sin^2(\frac{\pi}{T_{\max}}(t - t')) / L^2], \quad t, t' \in [0, T_{\max}], \quad (\text{A.4})$$

which guarantees  $I(0) = I(T_{\max})$  in distribution and represents loosely a cycling protocol. Let

$$\mathbf{K} = [k_{\text{per}}(t_i, t_j)]_{i,j=1}^n + \varepsilon^2 \mathbf{I}_n, \quad \varepsilon = 10^{-3},$$

and draw

$$\mathbf{y} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}). \quad (\text{A.5})$$

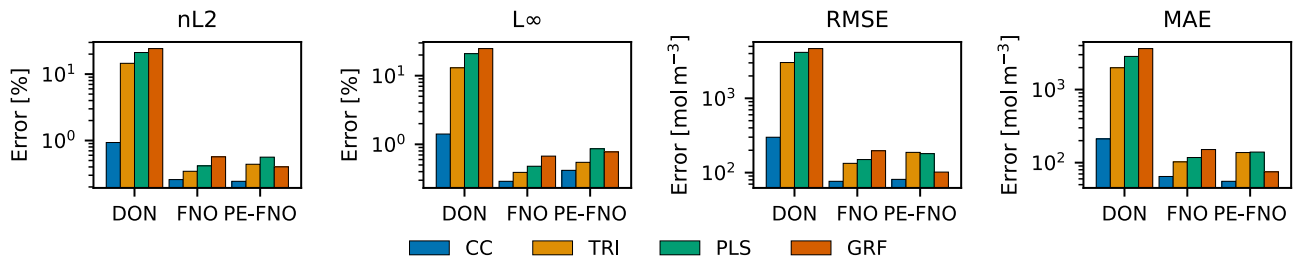
The continuous current is obtained by linear interpolation between the sampled currents. The current is clipped and scaled to be in the preferred operating range:

$$I(t) = \text{clip}(I_{\text{GRF}}(t), -1.5, 1.5) \times C.$$

**Table A.4**

Model parameters for the cell of [44] and the cell of [43].

Symbol	Name	Chen2020 (NMC)	Prada2013 (LFP)	Unit
$C$	Capacity	5	2.3	Ah
$c_{n,max}$	Max. conc. neg. el.	33 133	30 555	mol m <sup>-3</sup>
$c_{p,max}$	Max. conc. pos. el.	63 104	22 806	mol m <sup>-3</sup>
$U_{eq,n}$	OCP neg. electrode	(A.2)	(A.2)	V
$\alpha_n$	Charge-transfer coeff. neg. el.	0.5	0.5	—
$\sigma_n$	Conductivity neg. el.	215	215	S m <sup>-1</sup>
$\epsilon_n$	Porosity neg. el.	0.25	0.36	—
$L_n$	Thickness neg. el.	$8.52 \times 10^{-5}$	$3.4 \times 10^{-5}$	m
$D_n$	Diffusivity neg. particle	$3.3 \times 10^{-14}$	$3 \times 10^{-15}$	m <sup>2</sup> s <sup>-1</sup>
$R_n$	Radius neg. particle	$5.86 \times 10^{-6}$	$5 \times 10^{-6}$	m
$U_{eq,p}$	OCP pos. electrode	(A.1)	(A.3)	V
$\alpha_p$	Charge-transfer coeff. pos. el.	0.5	0.5	—
$\sigma_p$	Conductivity pos. el.	0.18	0.338	S m <sup>-1</sup>
$\epsilon_p$	Porosity pos. el.	0.335	0.426	—
$L_p$	Thickness pos. el.	$7.56 \times 10^{-5}$	$8 \times 10^{-5}$	m
$D_p$	Diffusivity pos. particle	$4 \times 10^{-15}$	$5.9 \times 10^{-18}$	m <sup>2</sup> s <sup>-1</sup>
$R_p$	Radius pos. particle	$5.22 \times 10^{-6}$	$5 \times 10^{-8}$	m

**Fig. A.8.** Accuracy metrics for DeepONet, FNO and PE-FNO on the Chen2020 LG M50 cell across CC/TRI/PLS/GRF input families. Metrics reported: normalised  $L_2$  and  $L_\infty$  (%), RMSE and MAE (mol m<sup>-3</sup>).

#### Error metrics

Given prediction  $\hat{y}$  and ground truth  $y$  on an  $n_r \times n_t$  grid we report four scalar metrics:

$$\begin{aligned}
 \text{MAE} &= \frac{1}{n_r n_t} \sum_{p,q} |\hat{y}_{pq} - y_{pq}|, \\
 \text{RMSE} &= \sqrt{\frac{1}{n_r n_t} \sum_{p,q} (\hat{y}_{pq} - y_{pq})^2}, \\
 \text{nL}_2 &= \frac{\|\hat{y} - y\|_2}{\|y\|_2 + \epsilon}, \quad \|y\|_2 = \left( \sum_{p,q} y_{pq}^2 \right)^{1/2}, \\
 \text{nL}_\infty &= \frac{\|\hat{y} - y\|_\infty}{\|y\|_\infty + \epsilon}, \quad \|y\|_\infty = \max_{p,q} |y_{pq}|,
 \end{aligned} \tag{F2-F5}$$

with a small constant  $\epsilon = 10^{-12}$  to avoid division by zero.

#### UNet architecture

Concretely, for an input of shape (24×85) with  $C_{\text{in}}$  channels:

$$\begin{aligned}
 (24 \times 85) &\xrightarrow{\text{sep\_block}(C)} e_1 \xrightarrow{\text{sep\_conv}, s=2} (12 \times 43) \xrightarrow{\text{sep\_block}(2C)} e_2, \\
 e_2 &\xrightarrow{\text{sep\_conv}, s=2} (6 \times 22) \xrightarrow{\text{sep\_block}(4C)} e_3.
 \end{aligned}$$

The decoder upsamples  $e_3$  with a 2×2 transposed convolution to  $\approx (12 \times 44)$ , crops to (12×43), concatenates with  $e_2$ , applies `sep_block(2C)`; then upsamples to  $\approx (24 \times 86)$ , crops to (24×85), concatenates with  $e_1$ , and applies `sep_block(C)`. A final 1×1 convolution maps to  $C_{\text{out}}$ .

#### Accuracy for the Chen2020 dataset

The Chen2020 dataset represents the LG M50 cell parametrised in [44]. As can be seen in Fig. A.8, the accuracy is comparable to Fig. 3 of the main part of the manuscript.

#### Data availability

Data and a code are available here: [https://github.com/alighato/spm\\_neuraloperator](https://github.com/alighato/spm_neuraloperator).

#### References

- [1] Zou C, Manzie C, Nešić D. Model predictive control for lithium-ion battery optimal charging. *IEEE/ASME Trans Mechatronics* 2018;23(2):947–57. <http://dx.doi.org/10.1109/TMECH.2018.2798930>.
- [2] Román-Ramírez LA, Marco J. Design of experiments applied to lithium-ion batteries: A literature review. *Appl Energy* 2022;320:119305. <http://dx.doi.org/10.1016/j.apenergy.2022.119305>.
- [3] Roman D, Saxena S, Robu V, Pecht M, Flynn D. Machine learning pipeline for battery state-of-health estimation. *Nat Mach Intell* 2021;3(5):447–56. <http://dx.doi.org/10.1038/s42256-021-00312-3>, Publisher: Nature Publishing Group.
- [4] Xiong R, Li L, Tian J. Towards a smarter battery management system: A critical review on battery state of health monitoring methods. *J Power Sources* 2018;405:18–29. <http://dx.doi.org/10.1016/j.jpowsour.2018.10.019>.
- [5] Doyle M, Fuller TF, Newman J. Modeling of galvanostatic charge and discharge of the lithium/polymer/insertion cell. *J Electrochem Soc* 1993;140(6):1526. <http://dx.doi.org/10.1149/1.2221597>, Publisher: IOP Publishing.
- [6] Berliner MD, Cogswell DA, Bazant MZ, Braatz RD. Methods—PETLION: Open-source software for millisecond-scale porous electrode theory-based lithium-ion battery simulations. *J Electrochem Soc* 2021;168(9):090504. <http://dx.doi.org/10.1149/1945-7111/ac201c>.
- [7] Marquis SG, Sulzer V, Timms R, Please CP, Chapman SJ. An asymptotic derivation of a single particle model with electrolyte. 2019. <http://dx.doi.org/10.48550/ARXIV.1905.12553>, Publisher: [object Object] Version Number: 2.
- [8] Hong C, Cho H, Hong D, Oh S-K, Kim Y. An improved thermal single particle model and parameter estimation for high-capacity battery cell. *Electrochim Acta* 2023;439:141638. <http://dx.doi.org/10.1016/j.electacta.2022.141638>.
- [9] Hwang G, Sitapure N, Moon J, Lee H, Hwang S, Sang-Il Kwon J. Model predictive control of lithium-ion batteries: Development of optimal charging profile for reduced intracycle capacity fade using an enhanced single particle model (SPM) with first-principled chemical/mechanical degradation mechanisms. *Chem Eng J* 2022;435:134768. <http://dx.doi.org/10.1016/j.cej.2022.134768>.

- [10] Attia PM, Grover A, Jin N, Severson KA, Markov TM, Liao Y-H, Chen MH, Cheong B, Perkins N, Yang Z, Herring PK, Aykol M, Harris SJ, Braatz RD, Ermon S, Chueh WC. Closed-loop optimization of fast-charging protocols for batteries with machine learning. *Nature* 2020;578(7795):397–402. <http://dx.doi.org/10.1038/s41586-020-1994-5>, Publisher: Nature Publishing Group.
- [11] Li W, Rentemeister M, Badeda J, Jöst D, Schulte D, Sauer DU. Digital twin for battery systems: Cloud battery management system with online state-of-charge and state-of-health estimation. *J Energy Storage* 2020;30:101557, Publisher: Elsevier.
- [12] Yuan Y, Jiang B, Chen Q, Wang X, Wei X, Dai H. A comparative study of battery state-of-charge estimation using electrochemical impedance spectroscopy by different machine learning methods. *Energy* 2025;328:136658. <http://dx.doi.org/10.1016/j.energy.2025.136658>.
- [13] Li W, Sengupta N, Dechent P, Howey D, Annaswamy A, Sauer DU. Online capacity estimation of lithium-ion batteries with deep long short-term memory networks. *J Power Sources* 2021;482:228863, Publisher: Elsevier.
- [14] Gayon Lombardo A. Machine learning and simulation for the optimisation and characterisation of electrodes in batterie. 2021, <http://dx.doi.org/10.25560/91548>, Publisher: [object Object].
- [15] Wang S, Ma C, Gao H, Deng D, Fernandez C, Blaabjerg F. Improved hyperparameter Bayesian optimization-bidirectional long short-term memory optimization for high-precision battery state of charge estimation. *Energy* 2025;328:136598. <http://dx.doi.org/10.1016/j.energy.2025.136598>, URL <https://linkinghub.elsevier.com/retrieve/pii/S0360544225022406>.
- [16] Wang S, Fan Y, Jin S, Takyi-Aninakwa P, Fernandez C. Improved anti-noise adaptive long short-term memory neural network modeling for the robust remaining useful life prediction of lithium-ion batteries. *Reliab Eng Syst Saf* 2023;230:108920. <http://dx.doi.org/10.1016/j.ress.2022.108920>, URL <https://linkinghub.elsevier.com/retrieve/pii/S095183202200535X>.
- [17] Wang S, Gao H, Takyi-Aninakwa P, Guerrero JM, Fernandez C, Huang Q. Improved multiple feature-electrochemical thermal coupling modeling of lithium-ion batteries at low-temperature with real-time coefficient correction. *Prot Control Mod Power Syst* 2024;9(3):157–73. <http://dx.doi.org/10.23919/PCMP.2023.000257>, URL <https://ieeexplore.ieee.org/document/10521737/>.
- [18] Tao S, Ma R, Chen Y, Liang Z, Ji H, Han Z, Wei G, Zhang X, Zhou G. Rapid and sustainable battery health diagnosis for recycling pre-treatment using fast pulse test and random forest machine learning. *J Power Sources* 2024;597:234156. <http://dx.doi.org/10.1016/j.jpowsour.2024.234156>, URL <https://linkinghub.elsevier.com/retrieve/pii/S0378775324001071>.
- [19] Tao S, Guo R, Lee J, Moura S, Casals LC, Jiang S, Shi J, Harris S, Zhang T, Chung CY, Zhou G, Tian J, Zhang X. Immediate remaining capacity estimation of heterogeneous second-life lithium-ion batteries via deep generative transfer learning. *Energy Environ Sci* 2025;18(15):7413–26. <http://dx.doi.org/10.1039/D5EE02217G>, URL <https://xlink.rsc.org/?DOI=D5EE02217G>.
- [20] Borah M, Wang Q, Moura S, Sauer DU, Li W. Synergizing physics and machine learning for advanced battery management. *Commun Eng* 2024;3(1):134. <http://dx.doi.org/10.1038/s44172-024-00273-6>, Publisher: Nature Publishing Group.
- [21] Chen T, Chen H. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Trans Neural Netw* 1995;6(4):911–7. <http://dx.doi.org/10.1109/72.392253>.
- [22] Raissi M, Perdikaris P, Karniadakis G. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J Comput Phys* 2019;378:686–707. <http://dx.doi.org/10.1016/j.jcp.2018.10.045>.
- [23] Wang F, Zhai Z, Zhao Z, Di Y, Chen X. Physics-informed neural network for lithium-ion battery degradation stable modeling and prognosis. *Nat Commun* 2024;15(1):4332. <http://dx.doi.org/10.1038/s41467-024-48779-z>, Publisher: Nature Publishing Group.
- [24] Li W, Zhang J, Ringbeck F, Jöst D, Zhang L, Wei Z, Sauer DU. Physics-informed neural networks for electrode-level state estimation in lithium-ion batteries. *J Power Sources* 2021;506:230034. <http://dx.doi.org/10.1016/j.jpowsour.2021.230034>.
- [25] Wang Y, Han X, Guo D, Lu L, Chen Y, Ouyang M. Physics-informed recurrent neural network with fractional-order gradients for state-of-charge estimation of lithium-ion battery. *IEEE J Radio Freq Identif* 2022;6:968–71. <http://dx.doi.org/10.1109/JRFID.2022.3211841>.
- [26] Hofmann T, Hamar J, Rogge M, Zoerr C, Erhard S, Philipp Schmidt J. Physics-informed neural networks for state of health estimation in lithium-ion batteries. *J Electrochem Soc* 2023;170(9):090524. <http://dx.doi.org/10.1149/1945-7111/acf0ef>, Publisher: IOP Publishing.
- [27] Sun B, Pan J, Wu Z, Xia Q, Wang Z, Ren Y, Yang D, Guo X, Feng Q. Adaptive evolution enhanced physics-informed neural networks for time-variant health prognosis of lithium-ion batteries. *J Power Sources* 2023;556:232432. <http://dx.doi.org/10.1016/j.jpowsour.2022.232432>.
- [28] Kim SW, Kwak E, Kim J-H, Oh K-Y, Lee S. Modeling and prediction of lithium-ion battery thermal runaway via multi-physics-informed neural network. *J Energy Storage* 2023;60:106654. <http://dx.doi.org/10.1016/j.est.2023.106654>.
- [29] Pang H, Wu L, Liu J, Liu X, Liu K. Physics-informed neural network approach for heat generation rate estimation of lithium-ion battery under various driving conditions. *J Energy Chem* 2023;78:1–12. <http://dx.doi.org/10.1016/j.jechem.2022.11.036>.
- [30] Rackauskas C, Ma Y, Martensen J, Warner C, Zubov K, Supekar R, Skinner D, Ramadhan A, Edelman A. Universal differential equations for scientific machine learning. 2021, <http://dx.doi.org/10.48550/arXiv.2001.04385>, arXiv:2001.04385 [cs].
- [31] Kuzhiyil JA, Damoulas T, Widanage WD. Neural equivalent circuit models: Universal differential equations for battery modelling. *Appl Energy* 2024;371:123692. <http://dx.doi.org/10.1016/j.apenergy.2024.123692>.
- [32] Nascimento RG, Corbetta M, Kulkarni CS, Viana FAC. Hybrid physics-informed neural networks for lithium-ion battery modeling and prognosis. *J Power Sources* 2021;513:230526. <http://dx.doi.org/10.1016/j.jpowsour.2021.230526>.
- [33] Kuzhiyil J, Damoulas T, Planella F, Widanage W. Lithium-ion battery degradation modelling using universal differential equations: Development of a cost-effective parameterisation methodology. *Appl Energy* 2025;382. <http://dx.doi.org/10.1016/j.apenergy.2024.125221>.
- [34] Huang Y, Zou C, Li Y, Wik T. MINN: Learning the dynamics of differential-algebraic equations and application to battery modeling. *IEEE Trans Pattern Anal Mach Intell* 2024;46(12):11331–44. <http://dx.doi.org/10.1109/TPAMI.2024.3456475>.
- [35] Kovachki N, Li Z, Liu B, Azizzadenesheli K, Bhattacharya K, Stuart A, Anandkumar A. Neural operator: Learning maps between function spaces with applications to PDEs. *J Mach Learn Res* 2023;24(89):1–97.
- [36] Lu L, Jin P, Pang G, Zhang Z, Karniadakis GE. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nat Mach Intell* 2021;3(3):218–29. <http://dx.doi.org/10.1038/s42256-021-00302-5>, Publisher: Nature Publishing Group.
- [37] Zheng Q, Yin X, Zhang D. Inferring electrochemical performance and parameters of li-ion batteries based on deep operator networks. *J Energy Storage* 2023;65:107176. <http://dx.doi.org/10.1016/j.est.2023.107176>.
- [38] Zheng Q, Yin X, Zhang D. State-space modeling for electrochemical performance of li-ion batteries with physics-informed deep operator networks. *J Energy Storage* 2023;73:109244. <http://dx.doi.org/10.1016/j.est.2023.109244>.
- [39] Brendel P, Mele I, Rosskopf A, Katrašnik T, Lorentz V. Parametrized physics-informed deep operator networks for design of experiments applied to lithium-ion-battery cells. *J Energy Storage* 2025;128:117055. <http://dx.doi.org/10.1016/j.est.2025.117055>.
- [40] Li Z, Kovachki N, Azizzadenesheli K, Liu B, Bhattacharya K, Stuart A, Anandkumar A. Fourier neural operator for parametric partial differential equations. 2021, <http://dx.doi.org/10.48550/arXiv.2010.08895>, arXiv:2010.08895 [cs].
- [41] Takamoto M, Alesiani F, Niepert M. Learning neural PDE solvers with parameterized channel attention. In: *Proceedings of the 40th international conference on machine learning*. PMLR; 2023, p. 33448–67, ISSN: 2640-3498.
- [42] Kwak M, Jin HS, Lkhagvasuren B, Oyunmunkh D. A robust state of charge estimator based on the Fourier neural operator for xEV batteries. *J Electrochem Soc* 2023;170(10):100504. <http://dx.doi.org/10.1149/1945-7111/acfdd3>, Publisher: IOP Publishing.
- [43] Prada E, Di Domenico D, Creff Y, Bernard J, Sauvage-Moynot V, Huet F. A simplified electrochemical and thermal aging model of lifepo<sub>4</sub>-graphite li-ion batteries: Power and capacity fade simulations. *J Electrochem Soc* 2013;160(4):A616–28. <http://dx.doi.org/10.1149/2.053304jes>.
- [44] Chen C-H, Brosa Planella F, O'Regan K, Gastol D, Widanage WD, Kendrick E. Development of experimental techniques for parameterization of multi-scale lithium-ion battery models. *J Electrochem Soc* 2020;167(8):080534. <http://dx.doi.org/10.1149/1945-7111/ab9050>.
- [45] Sulzer V, Marquis SG, Timms R, Robinson M, Chapman SJ. Python battery mathematical modelling (pybamm). *J Open Res Softw* 2021;9(1). <http://dx.doi.org/10.5334/jors.309>.
- [46] Ronneberger O, Fischer P, Brox T. U-net: Convolutional networks for biomedical image segmentation. 2015, <http://dx.doi.org/10.48550/ARXIV.1505.04597>, Version Number: 1. URL <https://arxiv.org/abs/1505.04597>.
- [47] Chollet F. Xception: Deep learning with depthwise separable convolutions. 2017, <http://dx.doi.org/10.48550/arXiv.1610.02357>, arXiv:1610.02357 [cs]. URL <http://arxiv.org/abs/1610.02357>.
- [48] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. 2015, <http://dx.doi.org/10.48550/arXiv.1512.03385>, arXiv:1512.03385 [cs].
- [49] Kingma DP, Ba J. Adam: A method for stochastic optimization. 2017, <http://dx.doi.org/10.48550/arXiv.1412.6980>, arXiv:1412.6980 [cs].
- [50] Loshchilov I, Hutter F. SGDR: Stochastic gradient descent with warm restarts. 2017, <http://dx.doi.org/10.48550/arXiv.1608.03983>, arXiv:1608.03983 [cs].
- [51] Bradbury J, Frostig R, Hawkins P, Johnson MJ, Leary C, Maclaurin D, Neco G, Paszke A, VanderPlas J, Wanderman-Milne S, Zhang Q. JAX: composable transformations of python+numpy programs. 2018, URL <http://github.com/google/jax>.
- [52] Heek J, Levskaya A, Oliver A, Ritter M, Rondepierre B, Steiner A, Zee Mv. Flax: A neural network library and ecosystem for JAX. 2024.
- [53] Anderson JAE, Gillis J, Horn G, Rawlings JB, Diehl M. Casadi: a software framework for nonlinear optimization and optimal control. *Math Program Comput* 2019;11(1):1–36. <http://dx.doi.org/10.1007/s12532-018-0139-4>, Publisher: Springer Science and Business Media LLC.