Proceedings of Machine Learning Research vol vvv:1–15, 2026

Physics-Informed Neural Operators for Cardiac Electrophysiology

Hannah Lydon HANNAH.LYDON@KCL.AC.UK
Milad Kazemi MILAD.KAZEMI@KCL.AC.UK
Department of Informatics, King's College London, UK
Martin Bishop MARTIN.BISHOP@KCL.AC.UK
Research Department of Digital Twins in Healthcare, King's College London, UK
Nicola Paoletti NICOLA.PAOLETTI@KCL.AC.UK
Department of Informatics, King's College London, UK

Abstract

Accurately simulating systems governed by PDEs, such as voltage fields in cardiac electrophysiology (EP) modelling, remains a significant modelling challenge. Traditional numerical solvers are computationally expensive and sensitive to discretisation, while canonical deep learning methods are data-hungry and struggle with chaotic dynamics and long-term predictions. Physics-Informed Neural Networks (PINNs) mitigate some of these issues by incorporating physical constraints in the learning process, yet they remain limited by mesh resolution and long-term predictive stability. In this work, we propose a *Physics-Informed Neural Operator (PINO)* approach to solve PDE problems in cardiac EP. Unlike PINNs, PINO models learn mappings between function spaces, allowing them to generalise to multiple mesh resolutions and initial conditions. Our results show that PINO models can accurately reproduce cardiac EP dynamics over extended time horizons and across multiple propagation scenarios, including zero-shot evaluations on scenarios unseen during training. Additionally, our PINO models maintain high predictive quality in long roll-outs (where predictions are recursively fed back as inputs), and can scale their predictive resolution by up to $10\times$ the training resolution. These advantages come with a significant reduction in simulation time compared to numerical PDE solvers, highlighting the potential of PINO-based approaches for efficient and scalable cardiac EP simulations. All code used in this work, including experimental results, can be found at https://github.com/janet-9/CardiacEP-PINOS

Keywords: Neural Operators, PDEs, Predictive modelling

1. Introduction

Modelling the behaviour of complex dynamical systems governed by partial differential equations (PDEs) is a common problem in engineering, particularly for simulating physical phenomena such as fluid flow and reaction-diffusion systems. Many biological processes can be modelled by PDE systems, and this framing has led to a growth of computational models that aim to simulate the human body to help guide diagnosis and treatment options (Katsoulakis et al.).

In cardiac healthcare, modelling electrophysiological dynamics poses a key challenge for understanding the origins of potentially dangerous arrhythmias and can aid in planning life-saving treatments, such as catheter ablation or the implantation of cardiac devices. To this purpose, several approaches have been explored to simulate cardiac electrophysiology (EP), e.g. (Jiang et al., 2016; Trayanova et al., 2024; Lydon et al., 2025). These methods are often based on mechanistic modelling, such as finite element simulations (Plank et al., 2021). As such, they tend to be highly sensitive to mesh resolution and model parameter selection, often resulting in trade-offs between maintaining model accuracy and ensuring computationally efficient simulations.

Building on the advances and successes of deep learning, learning-based methods have been recently applied to PDE modelling as well, resulting in accurate approximations of the PDE dynamics at a fraction of the simulation/inference cost. Physics-informed neural networks (PINNs) (Raissi et al., 2019) are a popular technique that combines a usual data loss term with a physics-based loss to penalise deviations from the (known) laws governing the system. In fact, PINNs have already been explored in the field of cardiac EP to model activation sequences in cardiac tissue (Sahli Costabal et al., 2020), as well as more complex cardiac dynamics in both 2D and 3D geometries (Herrero Martin et al., 2022; Chiu et al., 2024; Nazarov et al., 2022; Werneck et al., 2023; Zolotarev et al., 2025). Whilst PINNs have shown significant promise in replicating the accuracy of traditional finite-element EP models, they are limited to simulating on a fixed mesh resolution and often require extensive training budgets to capture the features of the PDE system. Additionally, because PINN models serve as function approximators for a single PDE instance, they require re-training to adapt to unseen model parameters, which limits their generalisation capabilities.

More recent *Neural Operator (NO)* models (Kossaifi et al., 2024; Kovachki et al., 2021). address the above limitations by shifting the training paradigm from fitting a network for one fixed PDE instance (the PINN approach) to learning mappings – or, operators – between function spaces, where the input function normally represents the initial conditions field and the output the field describing the corresponding PDE solution. As such, NO models can learn the dynamics of families of PDEs rather than a single instance (making them independent of the initial condition), can scale efficiently to high dimensions, and are resolution-invariant, i.e., they support inference at different (unseen) discretisation levels. Similar to the development of PINNs, NO models have been extended to incorporate physics losses to enforce agreement with known dynamics, an approach called *Physics-Informed Neural Operators (PINOs)* (Li et al., 2024).

So far, (PI)NO models have primarily been applied to canonical PDE problems such as fluid equations (e.g., Burger's equation and Navier-Stokes model). However, their ability to predict across a family of parametric PDEs (at a fraction of the cost of the original PDE) and generalise across resolutions (akin to numerical methods) makes them an ideal candidate for tackling EP modelling scenarios, which is the main objective of our work.

Contributions: This work presents the first neural operator-based method for the simulation of cardiac electrophysiology scenarios, incorporating known cell models as soft physics constraints with a Fourier neural operator (FNO) (Li et al., 2020) backbone. Through extensive experiments, we demonstrate that our model can:

- Produce model predictions of equivalent accuracy to numerical simulations for a range of cardiac wave propagation scenarios, with a speed up of approximately $\times 800$ at inference time compared to numerical simulations.

- Perform predictions on unseen data with resolutions significantly higher than those available during training, demonstrating the flexibility of our model.

- Achieve high-quality predictions in long roll-out simulations for unseen time horizons. That is, our PINO model also performs well when predictions are made sequentially from past predictions rather than the true inputs, demonstrating its applicability to online prediction scenarios where ground-truth data may be infrequent.

2. Background

2.1. Neural Operator Model

We define our general PDE system in the bounded domain $\mathcal{D}$ as

$$\frac{\partial u}{\partial t} = \mathcal{R}(u) \text{ on } \mathcal{D}\times(0,\infty), \quad u = g \text{ on } \partial\mathcal{D}\times(0,\infty), \quad u = a \text{ on } \overline{\mathcal{D}}\times\{0\}, \tag{1}$$

where we have a partial differential operator $\mathcal{R}$, a known initial condition $a = u(0)$ (where $\overline{\mathcal{D}}$ is the closure of $\mathcal{D}$, all spatial points including the boundary at time $t = 0$), and a known boundary condition g. For operator learning, we consider a family of PDE models described by Equation 1 with Banach spaces $\mathcal{A}$ (the space of initial condition functions) and $\mathcal{U}$ (the space of solution functions). For our initial and unknown conditions, we let $a = u(0) \in \mathcal{A}$ and $u(t) \in \mathcal{U}$ for $t > 0$.

Using this setting, we aim to learn $\mathcal{R}$ using a neural operator model in the method of (Kossaifi et al., 2024; Kovachki et al., 2021). We take the solution operator to be $\mathcal{G}^\dagger : \mathcal{A} \to \mathcal{U}$ induced by the mapping $a \to u$. The solution operator $\mathcal{G}^\dagger$ can be approximated by the neural operator model represented below:

$$\mathcal{G}_\theta = \mathcal{Q}\circ\sigma(\mathcal{W}_L + \mathcal{K}_L + b_L)\circ\ldots\circ\sigma(\mathcal{W}_1 + \mathcal{K}_1 + b_1)\circ\mathcal{P}, \tag{2}$$

which consists of two point wise operators $\mathcal{P} : \mathbb{R}^{d_a} \to \mathbb{R}^{d_L}$ and $\mathcal{Q} : \mathbb{R}^{d_L} \to \mathbb{R}^{d_u}$ such that $\mathcal{P}$ lifts the input function $a \in \mathcal{A}$ from dimension d_a to dimension d_L of the hidden layers, and $\mathcal{Q}$ projects the output of the hidden layers to dimension d_u of the unknown function $u \in \mathcal{U}$. The main section of the network consists of L layers of a parametrised pointwise linear operator $\mathcal{W}_l$ (a weight matrix), a constant bias b_l, and a parametrised kernel operator $\mathcal{K}_l$, which are combined and processed by a fixed activation function σ. All of the parameters of the network are described by θ. The kernel function $\mathcal{K}_l$ can take various forms depending on the task. In our case, we consider $\mathcal{K}_l$ to be the Fourier convolution operator, resulting in a Fourier Neural Operator model (FNO) as described by (Li et al., 2020). This kernel operator is described by Equation 3 below

$$(\mathcal{K}v^l)(x) = \mathcal{F}^{-1}(R\cdot(\mathcal{F}v^l))(x) \tag{3}$$

where $\mathcal{F}, \mathcal{F}^{-1}$ are the fast Fourier transform and its inverse, v^l is the function representation under the lifting operator P in layer l of the model, and R represents a matrix of learnable weights. This kernel function can be viewed as a convolution operating in Fourier space, where R assigns different weights to each frequency component. The motivation for employing this kernel is that by performing parametrisation in the Fourier domain we can reduce the complexity of the kernel calculations whilst still learning global features, which is ideal for PDE modelling[1].

2.2. Physics-informed Learning

A major challenge in applying machine learning to solve PDE problems is that enforcing physical constraints solely from data is difficult and may require large amounts of data, which can be prohibitive for these systems. The field of physics-informed machine learning addresses this problem by incorporating domain knowledge to guide the model construction. Physics-Informed Neural Networks (PINNs) (Raissi et al., 2019) are one of the first and most prominent methods. PINNs incorporate

1. To use the fast Fourier transforms, the functions are assumed to be on a uniform and discrete mesh. If not, then an additional transformation (typically a graph kernel operator) can be applied to map the functions to a uniform mesh.

physics knowledge as a soft constraint on the neural network by creating loss functions that penalise deviations from the PDE residuals as well as any known boundary and initial conditions.

A general form for this loss function, using ℓ_2-norm residuals, can be seen in Equation 4, where u_θ denotes the solution function found by the network, a is a given initial condition, $\mathcal{R}_\Theta$ is a parametrised form of $\mathcal{R}$, and λ_{res}, λ_{bc}, λ_{ic} are weights for each physics loss component[2].

$$\begin{aligned}\mathcal{L}_{\text{phys}}(a,u_\theta) = \mathcal{L}_{res} + \mathcal{L}_{IC} + \mathcal{L}_{BC} = \lambda_{\text{res}} \int_0^T \int_D \left\| \frac{du_\theta}{dt}(t,x) - \mathcal{R}_\Theta(u_\theta(t,x)) \right\|_2 dxdt \\ + \lambda_{\text{bc}} \int_0^T \int_{\partial D} \|u_\theta(t,x) - g(t,x)\|_2 dxdt + \lambda_{\text{ic}} \int_D \|u_\theta(0,x) - a(x)\|_2 dx\end{aligned} \tag{4}$$

As discussed in the Introduction, PINNs can only learn a single PDE instance at a time, requiring retraining with a new instance for each solution function $u \in \mathcal{U}$. Moreover, PINNs rely on discretised meshes to approximate the derivatives required in the PDE residual loss. This requires evaluating the network and its gradients at a large number of collocation points, which scales poorly with both the domain's dimensionality and the complexity of the PDE. Finally, due to their nature as an iterative solver, PINNs can struggle to generalise beyond the spatial and temporal domains seen during training.

These limitations can be addressed by using operator learning (see Section 2.1), which, by working over function spaces, can learn a family of PDE instances and generalise across resolutions in a data-efficient manner. We next formulate data and physics losses in the NO context. Let $\mathcal{G}_\theta$ be the operator model of Equation 2, learned from a dataset $\{a_j, u_j = \mathcal{G}^\dagger(a_j)\}_{j=1}^N$, where $\mathcal{G}^\dagger$ is the true solution operator we seek to approximate. We assume the dataset is induced by a distribution μ of initial conditions of interest. For a given function pair (a,u), the data loss for G_θ is

$$\mathcal{L}_{\text{data}}(a,u) = \int_D \|u(t,x) - \mathcal{G}_\theta(a)(t,x)\|_2 dx \tag{5}$$

where the integrals are calculated as the weighted sum of all available spatial points[3].

Similarly to PINNs, the final loss is obtained by combining $\mathcal{L}_{\text{data}}$ and $\mathcal{L}_{\text{phys}}$ (see Equation 4) via weighting parameters λ_{data} and λ_{phys}, and averaging over the dataset:

$$\begin{aligned}\mathcal{L} = \mathbb{E}_{a\sim\mu}\left[\lambda_{\text{data}}\mathcal{L}_{\text{data}}(a,\mathcal{G}^\dagger(a)) + \lambda_{\text{phys}}\mathcal{L}_{\text{phys}}(a,\mathcal{G}_\theta(a))\right] \approx \\ \frac{1}{N}\left[\sum_{j=1}^N \lambda_{\text{data}}\mathcal{L}_{\text{data}}(a_j,u_j) + \lambda_{\text{phys}}\mathcal{L}_{\text{phys}}(a_j,\mathcal{G}_\theta(a_j))\right].\end{aligned} \tag{6}$$

3. Problem Setting and Model Design

3.1. Problem Setting

We develop a physics-informed neural operator (PINO) approach for PDE-based cardiac electrophysiology. Previous work have considered this class of systems (Herrero Martin et al., 2022; Chiu et al.,

2. Both the weightings of the loss components and the parameters of $\mathcal{R}_\Theta$ can either be fixed or learnt as part of the network parameters, θ.
3. In the case that the functions are structured as grids, these weights are equal.

2024; Zolotarev et al., 2025; Nazarov et al., 2022) but using a PINN approach. By leveraging PINOs, we aim to provide a more flexible model able to 1) reproduce different cardiac propagation scenarios without having to retrain the model for each instance and 2) adapt to varying mesh resolutions, thereby drastically reducing both training and runtime costs.

The specific PDE problem we consider concerns simulating the propagation of voltage fields across 2D meshes using the Aliev-Panfilov cell model (Aliev and Panfilov, 1996). This model captures the shape of cardiac action potentials using a reaction diffusion system represented by the following pair of coupled PDEs:

$$\frac{\partial V}{\partial t} = \nabla \cdot (D \nabla V) - kV(V-a)(V-1) - VW \tag{7}$$

$$\frac{\partial W}{\partial t} = \left(\epsilon + \frac{\mu_1 W}{V+\mu_2}\right)[-W - kV(V-a-1)], \tag{8}$$

where V represents the transmembrane voltage in scaled dimensionless units, W represents a latent recovery variable and t represents a scaled dimensionless time variable. Parameters k and ϵ represent rate constants for excitation and restitution respectively, a[4] represents a threshold parameter for excitation and μ_1 and μ_2 are scalable parameters that control the shape of the action potential curve. In particular, a and μ_1 can be altered to induce chaotic dynamics in the system, as was demonstrated in (Panfilov, 2002). In our model we consider the domain to be isotropic, which allows the diffusion tensor D to be a scalar value and therefore reduces the first term in Equation 7 to the Laplacian $D\nabla^2 V$. Further details on the model implementation, including parameter selection and unit scaling information, can be found in Section S1.

3.2. Model Design

Model architecture Our model uses a Fourier Neural Operator (FNO) backbone (see Equations 2 and 3) to capture the system's global dynamics. The model architecture, as described in Section 2.1, consists of an initial grid embedding layer that encodes the positional information of the field maps, a lifting layer to expand to higher dimensions, four FNO blocks consisting of a Fourier convolution kernel, a combination of skip connections and channel MLPs with soft gating modulation, and a projection layer to map the field back to the original domain.

Data generation The training data consists of 2D spatio-temporal trajectories of transmembrane voltage V and recovery current W fields (the two variables of our PDE). All samples were processed in batches of dimensionality $[B,D,L,H_{grid},W_{grid}]$, with B being the batch size, D the field dimension[5], L the trajectory length and $H_{grid} \times W_{grid}$ the (spatial) training resolution. For data generation, the training time horizon $[0,T]$ – disjoint from the test horizon – was discretised using a regular grid, i.e., $0 = t_1, \ldots, t_k = T$. We considered two dataset formulations:

- *Single Frame Training* ($L = 1$): the model is given a single time frame as input and is trained to predict the time frame at n steps ahead. Specifically, for time point $i = 1, \ldots, k-n$, we associate a data point (a,u) where a is the PDE solution at t_i, i.e., $a = u^\dagger(t_i)$, and u is the solution after n steps, i.e., $u = u^\dagger(t_{i+n})$.

4. Not to be confused, in this context, with the operator input a (a voltage map).
5. In our case, the voltage and recovery fields respectively where $d = 0$ is the voltage field and $d = 1$ is the recovery field.

- *Multi Frame training ($L=m$)*: here, the model receives and predicts trajectories of length m instead of individual frames. In this case, for $i=1,...,k-n-2m$, we define the input a as the trajectory $a=(u^\dagger(t_i),...,u^\dagger(t_{i+m}))$ and its outcome as $u=(u^\dagger(t_{i+m+n}),...,u^\dagger(t_{i+2m+n}))$.

The reason we consider these two different formulations is to assess whether the predictive quality of the model could be improved by training with small sequences that evolve over time as opposed to single time snapshots, particularly when evaluating the model in sequential roll-outs (as described in our experiments of Section 4.1).

Physics loss computation We note that the choice of frames does not affect the data loss term, as this is defined over the spatial domain only (see Equation 5), but it does affect the physics-informed loss $\mathcal{L}_{\text{phys}}$ and specifically, the PDE residual term (Equation 4), as this requires temporal information to approximate the derivatives. In particular, the residual in Equation 5 is approximated using the central difference method, i.e., for time t_i and spatial point x:

$$\frac{du_\theta}{dt}(t_i,x)-\mathcal{R}_\Theta(u_\theta(t_i,x))\approx\left(\frac{u_\theta(t_{i+1})-u_\theta(t_{i-1})}{t_{i+1}-t_{i-1}}\right)-\mathcal{R}_\Theta(u_\theta(t_i,x)) \tag{9}$$

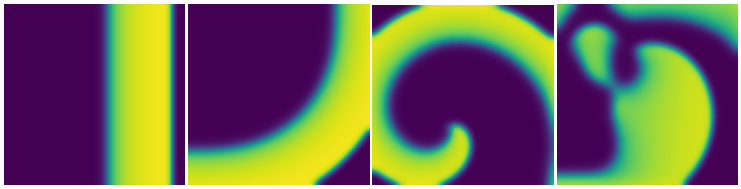
where $u_\theta(t_i)$ is the model prediction at time t_i, and the term $\mathcal{R}_\Theta(u_\theta(t_i,x))$ is obtained by plugging the field $u_\theta(t_i,x)$ into the RHS of the PDE equations 7 and 8.

The way Equation 9 is computed depends on the choice of frame structure. As described in (Li et al., 2024), in the multi-frame approach, the time derivative is calculated within the trajectory, i.e., for sample b in the batch, field dimension d, and spatial point (h,w), the difference $u_\theta(t_{i+1})-u_\theta(t_{i-1})$ is obtained as the prediction indexed by $[b,d,t_{i+1},h,w]$ minus that indexed by $[b,d,t_{i-1},h,w]$. In the single-frame setting, where we have individual points instead of trajectories, the derivative is calculated between consecutive samples of the batch, ensuring first that samples are organised sequentially within the batch: in this case, the above difference is obtained as the prediction indexed by $[t_{i+1},d,1,h,w]$ minus that indexed by $[t_{i-1},d,1,h,w]$.

The residual loss for each of Equations 7 and 8 are combined via a weighted sum, with the weights for each residual being scaled in order to bias capturing either the voltage or recovery field. The boundary conditions in both training methods are modelled as Neumann (no flux) boundary conditions and the initial condition loss is constructed for the multi-frame approach in order to minimise the errors between the first frame of the ground truth and predicted sample trajectories - that is, minimising the difference between $u^\dagger(t_0)$ and as $u_\theta(t_0)$ to ensure consistency with the ground truth at the start of the predicted trajectory.

Training algorithm Training proceeds in two stages: the first stage optimises only the data loss, and is followed by a second stage where both data and physics losses are applied. In particular, we explore three solutions to integrate the data and physics loss components during the second stage:

- *Fixed weights*: λ_{data} and λ_{phys} remain constant.
- *Soft adaptation* (Heydari et al., 2019): λ_{data} and λ_{phys} are adapted based on their relative magnitudes and rates of change.
- *Relative aggregation* (Bischof and Kraus, 2025): λ_{data} and λ_{phys} are scaled relative to their previous values, but are occasionally reset to their initial values to avoid bias to any recent fluctuations.



(a) Planar (b) Centrifugal (c) Spiral (d) Break-up

Figure 1: Propagation scenarios used for training the operator model, all sampled at $t = 500$ ms.

Inference At inference time, we evaluate the prediction quality using only the data loss (called RMSE in the experimental section). We consider two evaluation settings:

- *Point-to-point (P2P)*, where the model receives the ground truth input for prediction.
- *Roll-out*, where the model uses past predictions (rather than ground truth) to make new predictions. This setting is obviously more challenging, but serves as a test bed to assess whether our PINO model can be used as a simulator that operates autonomously without true data.

4. Experiments

We consider four propagation scenarios for our operator-based simulation: planar, centrifugal (propagation from a single corner of the mesh), spiral and spiral-breakup (pictured in Figure 1). The first two scenarios represent typical propagation patterns in healthy heart scenarios, whilst the latter are characteristic of arrhythmic conditions (with the spiral and spiral-breakup being analogous to monomorphic and polymorphic tachycardia/fibrillation, respectively). Training and evaluation data for all of the propagation scenarios were generated using the open-source cardiac modelling software package openCARP (Plank et al., 2021), a high-fidelity PDE simulator for cardiac electrophysiology. Further details of the PDE simulations for data generation can be found in Section S1.

When training the operator model, we split the simulation data into both single and multi-frame input-output pairs using a moving time window and then split the full set of function pairs into a training set representing the first 80% of the trajectory and used the rest of the trajectory as the testing set in order to encourage the model to extrapolate beyond the training time horizon. We also added a buffer of $2n-1$ frames between the testing and training sets in order to avoid data leakage at the border. For the multi-frame training, we used trajectories of length $m = 5$ (corresponding to $\sim 2.6\%$ of the training trajectory).

We trained all of the models using an Adam optimiser with a cosine learning rate scheduler (using an initial learning rate of 5×10^{-4} and a smooth decay over 1000 epochs). On a GPU-enabled server mounting an NVIDIA A40 48GB, the training time was $\approx$1 hour for single-frame training and $\approx$8 hours for multi-frame training.

4.1. Baseline Evaluations

The first experiment we performed was to evaluate each of the PINO models described in Section 3 under baseline conditions, using a fixed mesh resolution for both training and evaluation.

The best performing model of the initial training phase, using only data driven losses, was used as the FNO baseline and as the initial state for the secondary training round that incorporates physics losses[6]. The secondary training phases ran for 1000 epochs (details of variations in training epochs

6. For the fixed weights PINO model, the losses for the secondary training phase were set as $\lambda_{\text{res}}, \lambda_{\text{ic}}, \lambda_{\text{bc}} = 0.01, 0.1, 0.1$.

Table 1: Baseline performances across PINO/FNO models for a fixed mesh resolution. (* indicates cases in which the model prediction collapsed to the unexcited state.)

Scenario	Model	Single Frame		Multi Frame	
		RMSE (P2P)	**RMSE** (Roll-Out)	**RMSE** (P2P)	**RMSE** (Roll-Out)
Planar	FNO	0.3766	0.3819*	0.3752*	0.3794*
	PINO-Fixed	0.0257	**0.4735**	0.0276	0.0442
	PINO-SoftAdapt	0.0120	1.9414	0.0211	**0.0204**
	PINO-RelAdapt	**0.0108**	832.30*	**0.0195**	0.0246
Centrifugal	FNO	0.0466	0.3826*	0.0629	0.3747*
	PINO-Fixed	0.0415	0.3753*	0.0072	0.0146
	PINO-SoftAdapt	**0.0253**	**0.4661**	**0.0057**	**0.0112**
	PINO-RelAdapt	0.0309	0.3852*	0.0129	0.1418
Spiral	FNO	0.0248	0.4554	0.0507	0.1682
	PINO-Fixed	0.0060	0.0304	0.0113	0.0176
	PINO-SoftAdapt	0.0059	**0.0295**	0.0117	0.0179
	PINO-RelAdapt	**0.0056**	0.0326	**0.0113**	**0.0173**
Spiral-Break	FNO	0.0514	0.5939	0.1144	0.3132
	PINO-Fixed	0.0410	0.6004	0.0766	**0.1925**
	PINO-SoftAdapt	0.0432	0.5427	**0.0758**	0.1993
	PINO-RelAdapt	**0.0357**	**0.4797**	0.0832	0.3647

can be found in Section S2.2) and the best-performing model from the secondary training phase for each scenario was evaluated on a fixed-length trajectory to compare scores between scenarios.

As explained in Section 3, we consider three PINO variants based on whether and how the physics loss weights are updated during training; each model is trained in both single-frame and multi-frame modes; and each model is evaluated in a point-to-point (P2P) or roll-out (i.e., sequential) setting.

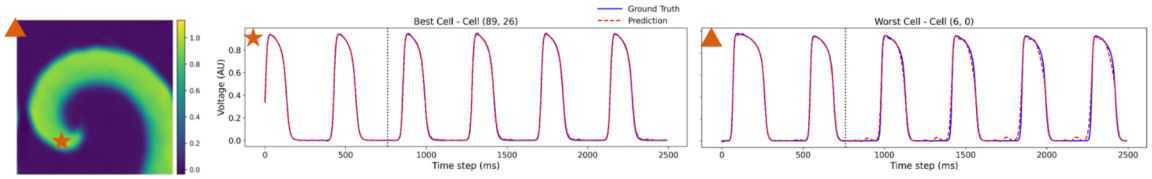
We additionally tested the available models on longer trajectories for the spiral and chaotic datasets, amounting to a full simulation trajectory of $\approx \times 3$ the training horizon. These results can be found in Section S2.1 and Figure 2.

The results of the baseline tests (found in Table 1 and Figures 2 and 3) highlight that all the PINO models show an improvement in performance compared to the FNO model[7]. Almost all models achieved an RMSE < 0.05 in the P2P evaluation (except the spiral-break model, which achieved an RMSE < 0.08 across all PINO models). For the roll-out evaluation, the multi-frame training approach vastly outperformed the single-frame approach (especially in the planar and centrifugal scenarios, where the single-frame roll-out tended to collapse to the unexcited state), and almost all models achieved an RMSE of < 0.03 for multi-frame roll-out predictions.

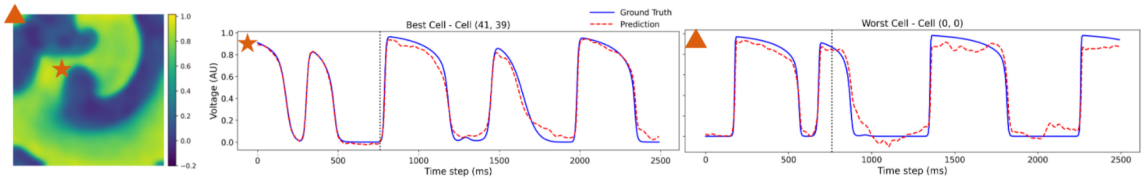
At inference time, the trained model was able to predict a full trajectory via the roll-out method in an average of ≈ 1s. Compared to traditional numerical solvers, such as the openCARP simulator that was used to generate the training data, this represents a speed-up in simulation time (for the same time horizon) by a factor of ≈ 800.

An important note about these experiments is that they were performed using a relatively small number of sample pairs (152 training samples and 30 testing samples for each of the scenarios) compared to previous PINO implementations (Li et al., 2024), which tend to use datasets with > 1000

7. We note that during training, we weighted the residual for Equation 7 at twice that of the residual loss for Equation 8 and evaluated models on the predictions of the voltage field (since the recovery field is latent).



(a)



(b)

Figure 2: Point-to-point long-time horizon predictions for the (a) spiral and (b) spiral break-up propagation scenarios. The dashed line represents the time horizon for the training data. The best- and worst-performing cells are marked on the voltage maps. The sample snapshots were taken at t=1750ms.

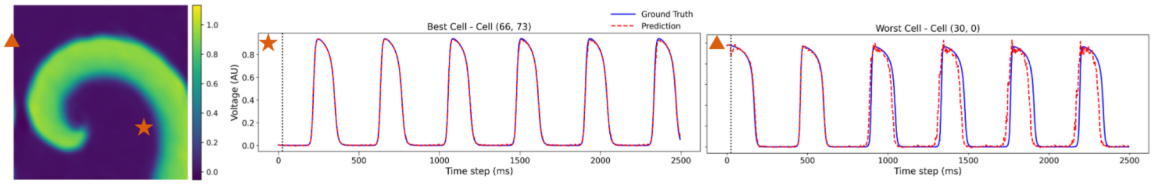


Figure 3: Results using the multi-frame roll-out evaluation method for the spiral scenario. The dashed line represents the initial time window given as an input to the model and the best and worst performing cells are marked on the sample field, taken at $t = 1250$ ms.

samples for training. Despite this limitation, our model is able to produce accurate predictions under both evaluation methods, even with a limited training dataset. This is likely due to the use of moving time windows to generate the training samples, combined with the physics loss constraints that enforce temporal consistency across trajectories.

4.2. Mesh Resolution Testing

In order to evaluate the ability of the operator model to extrapolate between different mesh resolutions at inference time, we trained the best-performing PINO model (from the baseline tests) for each scenario on a mesh that was downsampled in spatial resolution by a factor of 10 from the mesh used to generate the ground truth simulation data. We then evaluated it on target meshes (using P2P evaluation) with resolutions scaled up relative to the training mesh by factors of 1.25, 2.50, 5.00, and 10.00. As shown in Figure 4, even when increasing the mesh resolution by a factor of 10 relative to the training data, the decrease in model accuracy is $< 8\%$ across all propagation scenarios.

4.3. Zero-Shot Transfer

Finally, we evaluated the performance of our operator model in a zero-shot transfer scenario – i.e., on datasets that were completely unseen during training – to test the model's ability to learn the sys-

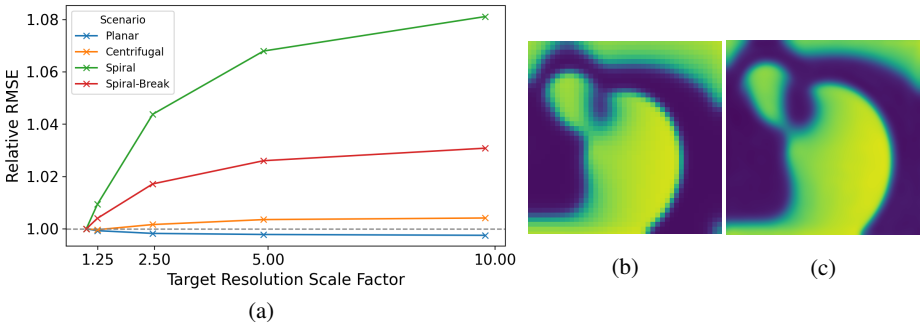


Figure 4: Zero-shot mesh resolution tests. (a) Relative RMSE values for P2P evaluation on increasing target resolution scales. (b) Input for the chaotic scenario at training resolution ($t = 495$ ms). (c) Prediction at the highest evaluation resolution ($t = 500$ ms).

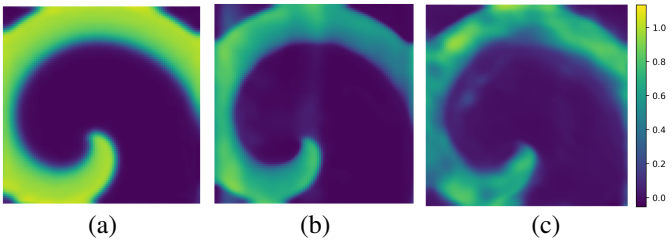


Figure 5: Zero-shot transfer. Samples shown are the predictions for the stable spiral scenario at $t = 500$ms using the (a) Stable spiral model. (b) Planar model. (c) Centrifugal model.

tem's underlying dynamics and use that knowledge to predict the trajectory of an unseen propagation scenario. For example, Figure 5 shows the best-performing models trained exclusively on the planar and centrifugal scenarios, evaluated on the (stable) spiral dataset. Despite being trained on simple scenarios, both models can accurately predict the patterns of a more complex propagation scenario. The only difference in the model predictions is in the scaling of the voltage field amplitudes, while the qualitative dynamics remain faithful.

5. Conclusion

In this work, we demonstrated that physics-informed neural operator (PINO) models can efficiently predict cardiac electrophysiology dynamics with comparable accuracy to numerical solvers, but at a fraction of the inference time. Our PINO models exhibit excellent generalisation across unseen mesh resolutions and propagation scenarios, albeit trained using small datasets and modest training budgets. These results demonstrate the strong potential of PINO models for enabling accurate online predictions in resource-constrained medical devices.

Future work will focus on extending this framework to irregular and 3D meshes in order to reflect realistic cardiac anatomy, potentially through geometry-aware operator layers (Chen et al., 2025; Li et al., 2023). We also aim to develop a general operator trained across multiple PDE instances to handle a wider range of cardiac cell models. Although this would increase training costs, strategies such as query-point residuals or temporal-channel separation (Diab and Al-Kobaisi, 2025) could help maintain computational efficiency.

Acknowledgments

This work was funded by the UKRI EPSRC grant MCPS-VeriSec (EP/W014785/2) and British Heart Foundation (BHF) Project Grant PG/22/10871. We would like to thank Ching-En Chiu and Marta Varela for thoughtful discussions on physics-informed cardiac modelling, Caroline Roney for discussions on data transforms for openCARP simulation data and to Tom Kuipers and the KCL e-research team for access to the computational resources used to run the experiments.

References

Rubin R. Aliev and Alexander V. Panfilov. A simple two-variable model of cardiac excitation. *Chaos, Solitons & Fractals*, 7(3):293–301, 1996. doi: 10.1016/0960-0779(95)00089-5.

Rafael Bischof and Michael A Kraus. Multi-objective loss balancing for physics-informed deep learning. *Computer Methods in Applied Mechanics and Engineering*, 439:117914, 2025.

Hai-Yang Chen, Liang Xue, Li Liu, Gao-Feng Zou, Jiang-Xia Han, Yu-Bin Dong, Meng-Ze Cong, Yue-Tian Liu, and Seyed Mojtaba Hosseini-Nasab. Physics-informed graph neural network for predicting fluid flow in porous media. *Petroleum Science*, 2025. ISSN 1995-8226. doi: 10.1016/j.petsci.2025.06.007. URL https://www.sciencedirect.com/science/article/pii/S199582262500216X.

Ching-En Chiu, Aditi Roy, Sarah Cechnicka, Ashvin Gupta, Arieh Levy Pinto, Christoforos Galazis, Kim Christensen, Danilo Mandic, and Marta Varela. Physics-informed neural networks can accurately model cardiac electrophysiology in 3d geometries and fibrillatory conditions. In *International Workshop on Statistical Atlases and Computational Models of the Heart*, pages 98–109. Springer, 2024.

W. Diab and M. Al-Kobaisi. Temporal neural operator for modeling time-dependent physical phenomena, 2025. URL http://arxiv.org/abs/2504.20249.

Craig S. Henriquez. A brief history of tissue models for cardiac electrophysiology. *IEEE Transactions on Biomedical Engineering*, 61(5):1457–1465, 2014. doi: 10.1109/TBME.2014.2310515.

Clara Herrero Martin, Alon Oved, Rasheda A. Chowdhury, Elisabeth Ullmann, Nicholas S. Peters, Anil A. Bharath, and Marta Varela. EP-PINNs: Cardiac electrophysiology characterisation using physics-informed neural networks. *Frontiers in Cardiovascular Medicine*, 8, 2022. doi: 10.3389/fcvm.2021.768419.

A Ali Heydari, Craig A Thompson, and Asif Mehmood. SoftAdapt: Techniques for adaptive loss weighting of neural networks with multi-part loss functions. *arXiv preprint arXiv:1912.12355*, 2019.

Zhihao Jiang, Houssam Abbas, Kuk Jin Jang, Marco Beccani, Jackson Liang, Sanjay Dixit, and Rahul Mangharam. In-silico pre-clinical trials for implantable cardioverter defibrillators. In *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 169–172, 2016. doi: 10.1109/EMBC.2016.7590667.

Evangelia Katsoulakis, Qi Wang, Huanmei Wu, Leili Shahriyari, Richard Fletcher, Jinwei Liu, Luke Achenie, Hongfang Liu, Pamela Jackson, Ying Xiao, Tanveer

Syeda-Mahmood, Richard Tuli, and Jun Deng. Digital twins for health: a scoping review. 7. ISSN 2398-6352. doi: 10.1038/s41746-024-01073-0. URL https://www.nature.com/articles/s41746-024-01073-0.

Jean Kossaifi, Nikola Kovachki, Zongyi Li, David Pitt, Miguel Liu-Schiaffini, Robert Joseph George, Boris Bonev, Kamyar Azizzadenesheli, Julius Berner, and Anima Anandkumar. A library for learning neural operators. 2024. doi: 10.48550/arXiv.2412.10354.

Nikola B. Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew M. Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces. *CoRR*, abs/2108.08481, 2021.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

Zongyi Li, Nikola Borislavov Kovachki, Chris Choy, Boyi Li, Jean Kossaifi, Shourya Prakash Otta, Mohammad Amin Nabian, Maximilian Stadler, Christian Hundt, Kamyar Azizzadenesheli, and Anima Anandkumar. Geometry-informed neural operator for large-scale 3d PDEs, 2023. URL http://arxiv.org/abs/2309.00583.

Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *ACM/IMS Journal of Data Science*, 1(3):1–27, 2024.

Hannah Lydon, Milad Kazemi Mehrabadi, Martin Bishop, and Nicola Paoletti. SimICD: A closed-loop simulation framework for ICD therapy. In *47th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2025.

Iulia Nazarov, Ihsane Olakorede, Ahmed Qureshi, Shaheim Ogbomo-Harmitt, and Oleg Aslanidi. Physics-informed fully connected and recurrent neural networks for cardiac electrophysiology modelling. In *2022 Computing in Cardiology (CinC)*, volume 498, pages 1–4, 2022. doi: 10.22489/CinC.2022.188.

Alexander V. Panfilov. Spiral breakup in an array of coupled cells: The role of the intercellular conductance. *Physical Review Letters*, 88(11):118101, 2002. doi: 10.1103/PhysRevLett.88.118101.

Gernot Plank, Axel Loewe, Aurel Neic, Christoph Augustin, Yung-Lin Huang, Matthias A.F. Gsell, Elias Karabelas, Mark Nothstein, Anton J. Prassl, Jorge Sánchez, Gunnar Seemann, and Edward J. Vigmond. The openCARP simulation environment for cardiac electrophysiology. *Computer Methods and Programs in Biomedicine*, 208:106223, 2021. doi: 10.1016/j.cmpb.2021.106223.

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

Francisco Sahli Costabal, Yibo Yang, Paris Perdikaris, Daniel E. Hurtado, and Ellen Kuhl. Physics-informed neural networks for cardiac activation mapping. *Frontiers in Physics*, 8, 2020. doi: 10.3389/fphy.2020.00042.

Natalia A. Trayanova, Aurore Lyon, Julie Shade, and Jordi Heijman. Computational modeling of cardiac electrophysiology and arrhythmogenesis: toward clinical translation. *Physiological Reviews*, 104(3):1265–1333, 2024. doi: 10.1152/physrev.00017.2023.

Yan Barbosa Werneck, Rodrigo Weber dos Santos, Bernardo Martins Rocha, and Rafael Sachetto Oliveira. Replacing the FitzHugh-nagumo electrophysiology model by physics-informed neural networks. In Jiří Mikyška, Clélia de Mulatier, Maciej Paszynski, Valeria V. Krzhizhanovskaya, Jack J. Dongarra, and Peter M.A. Sloot, editors, *Computational Science – ICCS 2023*, pages 699–713. Springer Nature Switzerland, 2023. doi: 10.1007/978-3-031-36021-3_67.

A. Zolotarev, S. Ip, K. Vydyula, B. Dhillon, C. Martín, S. Misghina, and C. Roney. opencarp-pinns: Towards faster prediction of cardiac signals propagation, 2025. URL https://stacom.github.io/stacom2025/papers/.

Appendix S1. Training Data Generation

As described in Section 3, the Aliev-Panfilov equations model the transmembrane voltage V, the recovery current W, and the time t in atomic units (AU). The scaling for V and t is described in Equation 10.

$$V[AU] = \frac{V[mV]+80}{100}, \qquad t[AU] = \frac{t[ms]}{12.9}. \tag{10}$$

All of the propagation scenarios described in Section 4 were simulated using the open-source cardiac modelling software package openCARP (Plank et al., 2021) on a 2D $10 \times 10\text{cm}^2$ tetrahedral mesh with a resolution of 250μm (resulting in a grid of resolution 401×401). All of the simulations were performed with isotropic conditions with the monodomain conduction model (Henriquez, 2014), and propagation was modelled using the Aliev-Panfilov cell model (using the parameters described in table 2).

Table 2: Parameters used for the Aliev-Panfilov residual loss

Propagation Scenario	D	a	μ_1	μ_2	k	ϵ
Planar, Centrifugal, Spiral	0.55	0.15	0.2	0.3	8	0.002
Spiral-Break	0.55	0.099	0.1	0.3	8	0.002

To produce the planar and centrifugal waves, the mesh was stimulated from a single wall and a single corner respectively. Both the spiral and spiral-break up waves were simulated using a cross-stimulation approach (pacing from a single wall, and then stimulating from a perpendicular wall at a later time in order to induce re-entrance). The simulations were generated with a time horizon of 1000ms for the planar and centrifugal scenarios in order to capture a single action potential whilst the spiral and spiral-breakup models were generated with a time horizon of 2500ms in order to capture the long term evolution of the system. All of the simulations were saved with a temporal resolution of 5ms.

During the construction of the datasets used in training, the voltage field data was normalised to be in AU. When calculating the residual loss, the time was scaled was also scaled to be in AU. For all of the propagation scenarios, we used the first 1000ms of the simulation to create the testing and training datasets.

Appendix S2. Additional Experiments

S2.1. Long Horizon Predictions

As described in Section 4.1, we evaluated the performance of the spiral and spiral break-up models on long time horizon trajectories, extending the time horizon of the training trajectory by a factor of $\sim \times 3$. The results are in agreement with Table 1 in that the single-frame model tends to perform better for P2P evaluations whilst the multi-frame model outperforms the single-frame model for the roll-out evaluation scenario. As expected, due to the greater challenge of predicting a longer time horizon, the RMSE value is greater across all models compared to the baseline results, particularly for the spiral break-up model (primarily due to the difficult nature of predicting chaotic systems over a long time horizon). Visualisations of the P2P results can be seen in the main body of the text in Figure 2.

Table 3: Long-Horizon Baseline performances for a fixed mesh resolution
(* indicates cases in which the model prediction collapsed to the unexcited state.)

Scenario	**Model**	**Single Frame**		**Multi Frame**	
		RMSE (P2P)	**RMSE** (Roll-Out)	**RMSE** (P2P)	**RMSE** (Roll-Out)
Spiral	FNO	0.0270	0.529	0.0525	0.273
	PINO-Fixed	0.0100	**0.0719**	**0.0200**	0.0486
	PINO-SoftAdapt	0.0101	0.0923	0.0216	0.0535
	PINO-RelAdapt	**0.00946**	0.0742	0.0213	**0.0393**
Spiral-Break	FNO	0.0742	0.632*	0.181	0.397*
	PINO-Fixed	0.0624	0.744*	0.145	**0.414**
	PINO-SoftAdapt	0.0654	0.611*	**0.130**	0.443
	PINO-RelAdapt	**0.0577**	**0.502***	0.149	0.442

S2.2. Training Epoch Variation

Following the baseline experiments in Section 4.1, we additionally experimented with the number of epochs that were used for the second training round assess the effect on the model performance. We both reduced the number of epochs that were used from the PINO baseline to 500 and increased it to 1500. The results can be found in Table 4 and in Fig 6, which demonstrate that in the both the P2P and Roll-Out analysis, reducing the training to 500 epochs almost universally decreased the performance of the model whilst increasing the number of epochs to 1500 didn't consistently improve model performance by any significant amount (with the exception of the spiral-break model). These results suggest that increasing the number of epochs can help the model to capture the more complex PDE dynamics of chaotic systems during training whilst for more stable propagation scenarios, increasing the training volume doesn't necessarily improve performance and can occasionally reduce the performance slightly, potentially as a result of over-fitting. In the interest of balancing model accuracy with computational training expense, we chose to use 1000 epochs as the baseline training budget for the secondary training phase.

Table 4: Effect of the volume of physics based training on model performance using the multi-frame approach for each of the propagation scenarios.

Scenario	**No. PINO training epochs**	**RMSE** (P2P)	**RMSE** (Roll-Out)
Planar	500	0.106	0.380*
	1000	0.0568	0.382
	1500	**0.0238**	**0.165**
Centrifugal	500	0.0516	0.349*
	1000	**0.00572**	**0.0107**
	1500	0.0132	0.0319
Spiral	500	0.0148	0.0269
	1000	0.0126	**0.0186**
	1500	**0.0112**	0.0214
Spiral-Break	500	0.0949	0.253
	1000	0.0791	0.2001
	1500	**0.0765**	**0.176**

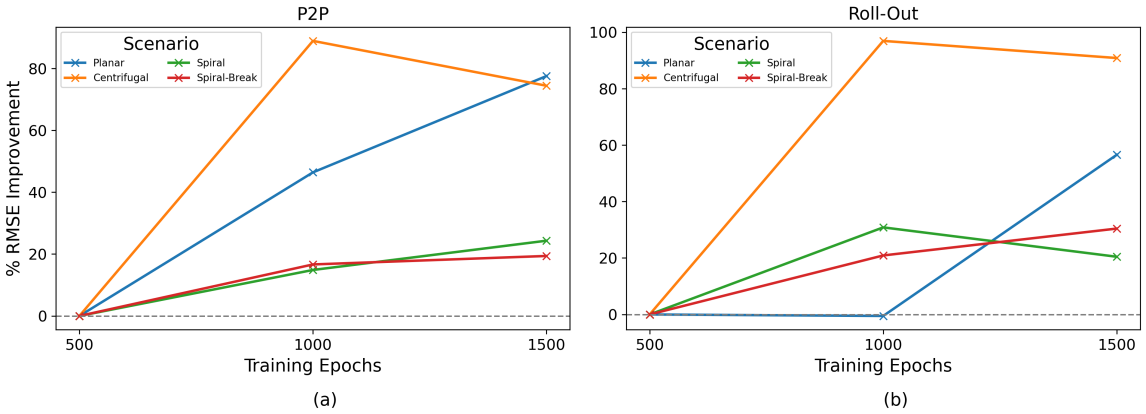


Figure 6: Comparison of Model Performance using Mutli-Frame Training with training budget, showing the % RMSE improvement for (a) for P2P evaluation, (b) Roll-Out evaluation.