A MULTIFIDELITY APPROACH TO CONTINUAL LEARNING FOR PHYSICAL SYSTEMS

Amanda Howard, Yucheng Fu, and Panos Stinis
Advanced Computing, Mathematics and Data Division
Pacific Northwest National Laboratory
Richland, WA 99354
amanda.howard@pnnl.gov

= Do not understand
= Review concept
= Comment / Important
= Error / Inconsistency

April 11, 2023

ABSTRACT

We introduce a novel continual learning method based on multifidelity deep neural networks. This method learns the correlation between the output of previously trained models and the desired output of the model on the current training dataset, limiting catastrophic forgetting. On its own the multifidelity continual learning method shows robust results that limit forgetting across several datasets. Additionally, we show that the multifidelity method can be combined with existing continual learning methods, including replay and memory aware synapses, to further limit catastrophic forgetting. The proposed continual learning method is especially suited for physical problems where the data satisfy the same physical laws on each domain, or for physics-informed neural networks, because in these cases we expect there to be a strong correlation between the output of the previous model and the model on the current training domain.

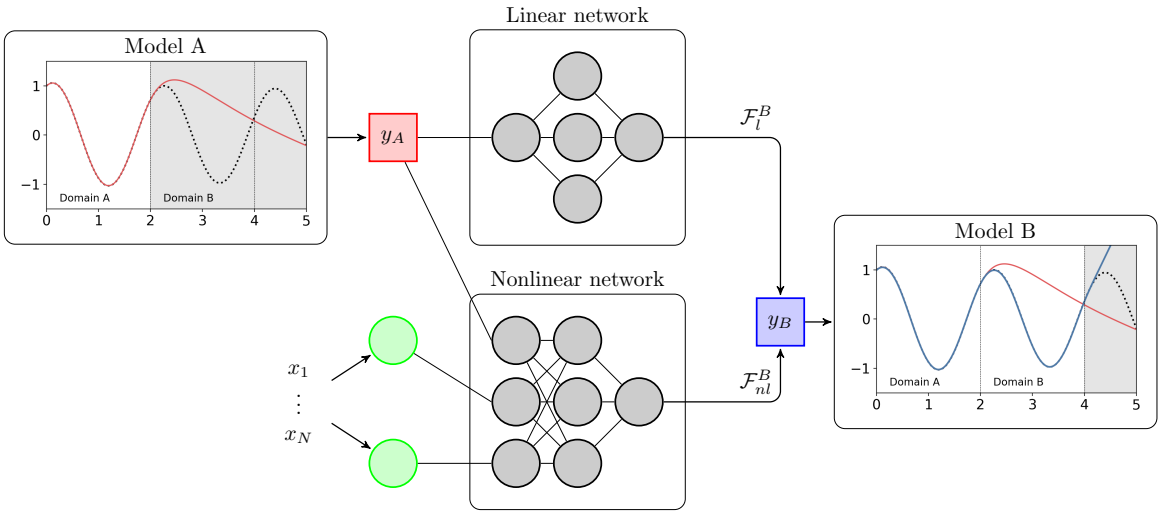


Figure 1: Graphical abstract

1 Introduction

In many real world applications of machine learning, data is received sequentially or in discrete datasets. When used as training data, new information received about the system requires completely retraining a given neural network. Much

recent work has focused on how to instead incorporate the newly received training data into the machine learning model without requiring retraining with the full dataset and without forgetting the previously learned model. This process is referred to as continual learning [1]. One key goal in continual learning is to limit catastrophic forgetting, or abruptly and completely forgetting the previously trained data.

Many methods have been proposed to limit forgetting in continual learning. In replay (rehearsal), a subset of the training set from previously trained regions are used in training subsequent models, so the method can limit forgetting by reevaluating on the previous regions [2]. However, replay requires access to the previously used training data sets. This both requires large storage capabilities for large datasets, and also physical access to the previous dataset. However, data privacy can limit access to prior datasets, so replay may not be a feasible option. An alternative to replay are regularization methods, where a regularizer is used to assign weights to each parameter in the neural network, representing the parameter's importance. Then, a penalty is applied to prevent the parameters with the largest weights from changing. Multiple methods have been proposed for how to calculate the importance weights. Among the top choices are Synaptic Intelligence [3], elastic weight consolidation (EWC) [4], and memory aware synapses (MAS) [5]. Subsequent work has shown that MAS performs among the best in multiple use cases, and is more robust to the choice of hyperparameters, so here we use MAS [6, 7]. Finally, a third category of continual learning methods includes those that employ task-specific modules [8], ensembles [9], adapters [10], reservoir computing based architectures [11], slow-fast weights [12, 13] and more.

In recent years, a huge research focus has been on scientific machine learning methods for physical systems [14], for example fluid mechanics and rheology [15, 16], metamaterial development [17, 18, 19], and high speed flows [20]. In particular, physics-informed neural networks, or PINNs [21], allow for accurately representing differential operators through automatic differentiation, allowing for finding the solution to PDEs without explicit mesh generation. Work on continual learning for PINNs is limited. While as a first attempt PINNs can be trained on the entire domain because the issues of data acquisition and privacy do not apply, many systems have been identified for which it is not possible to train a PINN for the entire desired time domain. For example, even the simple examples used in this work, a pendulum and the Allen-Cahn equation, cannot be trained by a PINN for long times. Recent work has looked at improving the training of PINNs for such systems, including applications of the neural tangent kernel [22], but more work remains to be done. The closest work we are aware of for continual learning with PINNS is the backward-compatible PINNs in [23]. They train N PINNs on a sequence of N time domains, and in each new domain enforce that the output from the current PINN satisfies the PINN loss function in the current domain and the output of the previous model on all previous domains. We note that this work is distinct from the replay approach taken with PINNs in this work, both in the single fidelity and multifidelity cases, because we enforce that the Nth neural network satisfies the residual in all prior domains, not the output from the previous model.

We will introduce the multifidelity continual learning method in Sec. 2. We will then show the performance of the method on physics-informed problems in Sec. 3 and on data-informed problems in Sec. 4.

2 Multifidelity continual learning method

We assume that we have a domain Ω, which we divide into N subdomains $\Omega = \cup_{i=0}^{N} \Omega_i$. We will learn sequential models on each subdomain Ω_i, with the goal that the ith model can provide accurate predictions on the domain $\cup_{j=0}^{i} \Omega_j$. That is, the ith model does not forget the information learned on earlier domains used in training. We will focus on applications to physical systems, where we either have data available or knowledge of the physical laws the system obeys. We will begin this section with a brief overview of physics-informed neural networks (PINNs), then discuss the multifidelity continual learning method (MFCL), and conclude with a description of methods we use to limit catastrophic forgetting.

2.1 Physics-informed neural networks

In this section we give a brief introduction to single-fidelity and multifidelity physics-informed neural networks (PINNs), which were introduced in [21] and have been covered in depth for many relevant applications [24, 14]. PINNs are generally used, in these applications, for initial-boundary valued problems.

$$\mathbf{s}_t + \mathcal{N}_{\mathbf{x}}[\mathbf{s}] = \mathbf{0}, \ \mathbf{x} \in \Omega, t \in [0, T] \tag{1}$$

$$\mathbf{s}(\mathbf{x}, t) = \mathbf{g}(\mathbf{x}, t) \ \mathbf{x} \in \partial\Omega, t \in [0, T] \tag{2}$$

$$\mathbf{s}(\mathbf{x}, 0) = \mathbf{u}(\mathbf{x}) \ \mathbf{x} \in \Omega \tag{3}$$

where $\Omega \in \mathbb{R}^N$ is an open, bounded domain with boundary $\partial\Omega$, $\mathbf{g}$ and $\mathbf{u}$ are given functions, and $\mathbf{x}$ and t are the spatial and temporal coordinates, respectively. $\mathcal{N}_{\mathbf{x}}$ is a general differential operator with respect to $\mathbf{x}$. We wish to find an

approximation to $\mathbf{s}(\mathbf{x}, t)$ by a (series) of deep neural networks with parameters θ, denoted by $\mathbf{s}^\theta(\mathbf{x}, t)$. The neural network is trained by minimizing the loss function

$$\mathcal{L}(\theta) = \lambda_{bc}\mathcal{L}_{bc}(\theta) + \lambda_{ic}\mathcal{L}_{ic}(\theta) + \lambda_r\mathcal{L}_r(\theta) + \lambda_{data}\mathcal{L}_{data}(\theta) \tag{4}$$

where the subscripts *bc*, *ic*, *r*, and *data* denote the terms corresponding to the boundary conditions, initial conditions, and residual, and any provided data, respectively. We take N_{bc}, N_{ic}, and N_r to be the batch sizes of the boundary, initial, and residual data point, and denote the training data by $\left\{(\mathbf{x}_{bc}^i, t_{bc}^i), \mathbf{g}(\mathbf{x}_{bc}^i, t_{bc}^i)\right\}_{i=0}^{N_{bc}}$, $\left\{(\mathbf{x}_{ic}^i), \mathbf{u}(\mathbf{x}_{bc}^i)\right\}_{i=0}^{N_{ic}}$, and $\left\{(\mathbf{x}_r^i, t_r^i)\right\}_{i=0}^{N_r}$. The boundary and initial collocation points are randomly sampled uniformly in their respective domains. The selection of the N_r residual points will be discussed in Sec. 2.3. If data representing the solution $\mathbf{s}$ is available, we can also consider an additional dataset $\left\{(\mathbf{x}_{data}^i, t_{data}^i), \mathbf{s}(\mathbf{x}_{data}^i, t_{data}^i)\right\}_{i=0}^{N_{data}}$. This term is included to capture the data-based training we will cover in Sec. 4.

The individual loss terms are given by the mean square errors,

$$\mathcal{L}_{bc}(\theta) = \frac{1}{N_{bc}}\sum_{i=0}^{N_{bc}} \left|\mathbf{s}_\theta(\mathbf{x}_{bc}^i, t_{bc}^i) - \mathbf{g}(\mathbf{x}_{bc}^i, t_{bc}^i)\right|^2 \tag{5}$$

$$\mathcal{L}_{ic}(\theta) = \frac{1}{N_{ic}}\sum_{i=0}^{N_{ic}} \left|\mathbf{s}_\theta(\mathbf{x}_{ic}^i, 0) - \mathbf{u}(\mathbf{x}_{ic}^i)\right|^2 \tag{6}$$

$$\mathcal{L}_r(\theta) = \frac{1}{N_r}\sum_{i=0}^{N_r} \left|\mathbf{r}_\theta(\mathbf{x}_r^i, t_r^i)\right|^2 \tag{7}$$

$$\mathcal{L}_{data}(\theta) = \frac{1}{N_{data}}\sum_{i=0}^{N_{data}} \left|\mathbf{s}_\theta(\mathbf{x}_{data}^i, t_{data}^i) - \mathbf{s}(\mathbf{x}_{data}^i, t_{data}^i)\right|^2 \tag{8}$$

where

$$\mathbf{r}_\theta(\mathbf{x}, t) = \frac{\partial}{\partial t}\mathbf{s}_\theta(\mathbf{x}, t) + \mathcal{N}_\mathbf{x}[\mathbf{s}_\theta(\mathbf{x}_{data}^i, t_{data}^i)]. \tag{9}$$

The weighting parameters λ_{bc}, λ_{ic}, λ_r, and λ_{data} are chosen before training by the user.

Multifidelity PINNs, as used in this work, are inspired by [25]. We assume we have a low fidelity model in the form of a deep neural network that approximates a given dataset or differential operator with low accuracy. We want to train two additional neural networks to learn the linear and nonlinear correlations between the low fidelity approximation and a high fidelity approximation or high fidelity data. We denote these neural networks as $\mathcal{NN}_l$ for the linear correlation and $\mathcal{NN}_{nl}$ for the nonlinear correlation. The output is then $\mathbf{s}_\gamma(\mathbf{x}, t) = \mathcal{NN}_{nl}(\mathbf{x}, t; \gamma) + \mathcal{NN}_l(\mathbf{x}, t; \gamma)$, where γ is all trainable parameters of the linear and nonlinear networks. The loss function includes an additional term,

$$\mathcal{L}_{MF}(\gamma) = \lambda_{bc}\mathcal{L}_{bc}(\gamma) + \lambda_{ic}\mathcal{L}_{ic}(\gamma) + \lambda_r\mathcal{L}_r(\gamma) + \lambda_{data}\mathcal{L}_{data}(\gamma) + \lambda\sum(\gamma_{nl,ij})^2, \tag{10}$$

where $\{\gamma_{nl,ij}\}$ is the set of all weights and biases of the nonlinear network $\mathcal{NN}_{nl}$. No activation function is used in $\mathcal{NN}_l$ to result in learning a linear correlation between the previous prediction and the high fidelity model.

2.2 Multifidelity continual learning

In the MFCL method, we exploit correlations between the previously trained models on prior domains and the expected model on the current domain. *Explicitly, we use the prior model $\mathcal{F}_{i-1}$ as a low fidelity model for domain Ω_i. Then, we learn the correlation between $\mathcal{F}_{i-1}$ on domain Ω_i and the data or physics given on the domain.* By learning a general combination of linear and nonlinear terms, we can capture complex correlations. Because the method learns only the correlation between the previous model and the new model, we can in general use smaller networks in each subdomain. The procedure requires two initial steps:

1. Train a (single-fidelity) DNN or PINN on Ω_1, denoted by $\mathcal{NN}^*(\mathbf{x}, t; \theta^*)$. This network will approximate the solution in a single domain.
2. Train a multifidelity DNN or PINN in Ω_1, which takes as input the single fidelity model $\mathcal{NN}^*(\mathbf{x}, t; \theta^*)$ as a low fidelity approximation. This initial multifidelity network is denoted by $\mathcal{NN}_1(\mathbf{x}, t; \theta_1)$

Then, for each additional domain Ω_i, we train a multifidelity DNN or PINN in Ω_i, denoted by $\mathcal{NN}_i(\mathbf{x}, t; \theta_i)$, which takes as input the previous multifidelity model $\mathcal{NN}_{i-1}(\mathbf{x}, t; \theta_{i-1})$ as a low fidelity approximation. The goal is for

Why do we need two NNs to be trained on Ω_1? It seems that we could skip step 2.

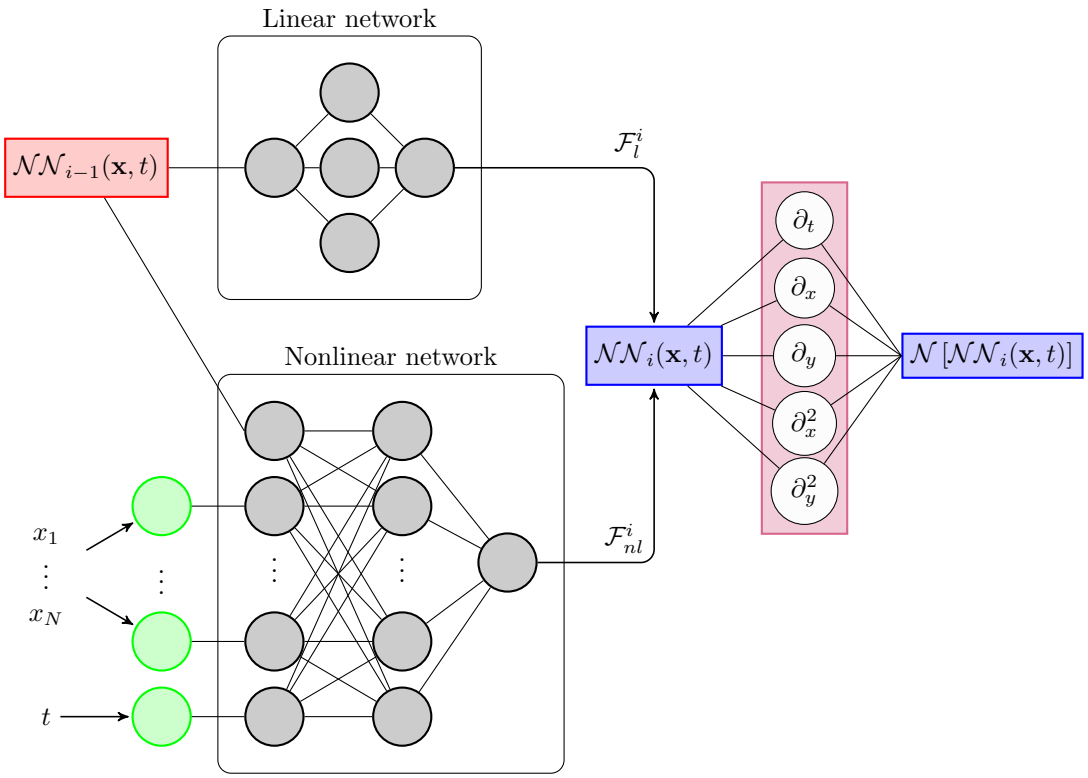


Figure 2: Diagram of the MFCL method on domain Ω_i. The output from the previously trained neural network, $\mathcal{NN}_{i-1}(\mathbf{x},t;\theta_{i-1})$, is used as input to the linear and nonlinear subnets for a point $(\mathbf{x},t) \in \Omega_i$. The output neural network is the sum of the linear and nonlinear subnetworks.

$\mathcal{NN}_i(\mathbf{x},t;\theta_i)$ to provide an accurate solution on $\cup_{j=1}^i \Omega_i$, even when data from Ω_j, $j<i$, is not used in training the multifidelity network $\mathcal{NN}_i$. A diagram of the method is given in Fig. 2.

As we will show, the MF-CL method provides more accurate results with less forgetting than single fidelity training on its own, however, the method can be improved by a few methods that have been previously developed both for reducing forgetting in continual learning and for selecting collocation points for training PINNs. These methods are discussed below.

We use transfer learning in all cases, so the weights of each new network are initialized with the trained values from the previous network, $\gamma_i^0 = \gamma_{i-1}$. We note that some previous work has found less forgetting by initializing each subsequent network randomly [26], and leave the exploration of this option for future work.

2.3 Residual-driven point selection

The points at which the residual is evaluated can greatly influence the training of PINNs [27]. We follow a modified version of the residual-based adaptive refinement with distribution (RAR-D) method proposed by [28, 27]. For training on region Ω_i, we consider the model Ω_{i-1}. We evaluate the residual output from $\mathcal{NN}_{i-1}$ at N points densely distributed in domain Ω_i. Then, we evaluate:

$$y_j = |\mathbf{r}_{i-1}(\mathbf{x}_j,t_j);\theta_{i-1})| \tag{11}$$

for $(\mathbf{x}_j,t_j) \in \Omega_i$. We create a probability density function,

$$p(x^j) = y_j^k/mean(y_j) + c \tag{12}$$

with $k=2$, $c=0$.

We denote this method by residual-driven point selection (RDPS). To train $\mathcal{NN}_i$ on domain Ω_i, at each iteration we select $N_{RDPS} = \lfloor N_r/2 \rfloor$ points according to the probability density function p and $N_{ran} = N_r - N_{RDPS}$ points randomly in domain Ω_i. We note that while the points are randomly selected at each iteration, in contrast with the

RAR-D method proposed in [27], the probability density function is precomputed before training from the previously trained model, and is not updated during training of the new model.

The points at which the initial and boundary condition losses are computed are selected randomly with the correct batch size at each iteration, and RDPS is not used. When RDPS is not used, we take $N_{RDPS} = 0$, so each iteration uses $N_{ran} = N_r$ randomly chosen collocation points (this method is denoted by Random-R in [27].)

2.4 Memory aware synapses

Memory aware synapses (MAS) attempts to limit forgetting in continual learning by assigning an importance weight to each neuron in the neural network. Then, a penalty term is added to the loss function to prevent large deviations in the values of important weights when the next networks are trained. The importance weights are found by measuring how sensitive the output of neural net $\mathcal{NN}_n$ is to changes in the network parameters [5]. For each weight and bias γ_{ij} in the neural network we calculate the importance weight parameter

$$\Omega_{ij}^n = \frac{1}{N}\sum_{k=1}^{N}\left\|\frac{\partial\left(\ell_2^2\mathcal{NN}_n(x_k;\gamma)\right)}{\partial\gamma_{ij}}\right\| \tag{13}$$

where ℓ_2^2 denotes the squared ℓ_2 norm of the output of the neural network $\mathcal{NN}_n$ applied at x_k. The loss function in eq. 10 is then modified to read:

$$\begin{aligned}\mathcal{L}_{MF,MAS}(\gamma^n) =& \lambda_{bc}\mathcal{L}_{bc}(\gamma^n) + \lambda_{ic}\mathcal{L}_{ic}(\gamma^n) + \lambda_r\mathcal{L}_r(\gamma^n) + \lambda_{data}\mathcal{L}_{data}(\gamma^n) + \lambda\sum_{i,j}(\gamma_{nl,ij}^n)^2 \\ &+ \lambda_{MAS}\sum_{i,j}\Omega_{ij}^{n-1}\left(\gamma_{ij}^n - \gamma_{ij}^{n-1}\right)^2\end{aligned} \tag{14}$$

Why are they only penalizing nonlinear weights?

As noted in [29], it may benefit training to detect outliers in the MAS weights and apply scaling to minimize their impact. In this work, we consider scaling the MAS terms by the 0.99th quantile, to prevent very large outliars from impacting the scaling. We denote this by scaled MAS.

When applying MAS to multifidelity neural networks, we calculate the MAS terms separately:

$$\Omega_{ij}^{n,nl} = \frac{1}{N}\sum_{k=1}^{N}\left\|\frac{\partial\left(\ell_2^2\mathcal{NN}_n^{nl}(x_k;\gamma)\right)}{\partial\gamma_{ij}^{nl}}\right\|, \quad \Omega_{ij}^{n,l} = \frac{1}{N}\sum_{k=1}^{N}\left\|\frac{\partial\left(\ell_2^2\mathcal{NN}_n^{l}(x_k;\gamma)\right)}{\partial\gamma_{ij}^{l}}\right\| \tag{15}$$

where nl denotes the nonlinear network and l denotes the linear network. In this way, roughly, the importance in the weights in calculating the linear and nonlinear terms is found separately, instead of determining the importance in the overall output of the sum of the networks. The parameter λ_{MAS} is kept the same for the linear and nonlinear parts, and when scaling is applied, the 99th quantile is calculated across the combined set of linear and nonlinear MAS weights.

2.5 Replay

In replay, a selection of points in the previously trained domains, $\cup_{i=1}^{n-1}\Omega_i$ are selected at each iteration and the residual loss, $\mathcal{L}_r(\gamma^n)$ is evaluated at the points. In this way, the multididelity training still satisfies the PDE across the earlier trained domains. For PINNs, the replay approach only requires knowledge of the geometry of $\cup_{i=1}^{n-1}\Omega_i$, and not the value of the output of the model on this domain.

2.6 Transfer learning

In all cases in this work, the values of the trainable parameters in each subsequent network $\mathcal{NN}_i$, $i \geq 2$, is initialized from the final values of the trainable parameters in the previous network, $\mathcal{NN}_{i-1}$. This approach allows for faster training because the network is not initialized randomly.

	Single fidelity	Multifidelity
Applied alone	17.23	2.15
RDPS	12.32	1.35
Replay-RDPS	0.028	0.055
MAS	1.39	2.33
MAS scaled	7.76	1.69
MAS-RDPS	2.32	1.68
MAS scaled-RDPS	2.59	0.95

Table 1: RMSE of the final output $\mathcal{NN}_5$ on the full domain for the pendulum problem. For the MAS cases, the network is trained for six values of λ_{MAS}, and the case with the lowest RMSE is shown in the table above.

3 Physics-informed training

3.1 Pendulum dynamics

In this section, we consider the gravity pendulum with damping from [22]. The system is governed by an ODE for $t \in [0, T]$

$$\frac{ds_1}{dt} = s_2, \tag{16}$$

$$\frac{ds_2}{dt} = -\frac{b}{m}s_2 - \frac{g}{L}\sin(s_1). \tag{17}$$

The initial conditions are give by $s_1(0) = s_2(0) = 1$. We take $m = L = 1$, $b = 0.05$, and $g = 9.81$, and we take $T = 10$.

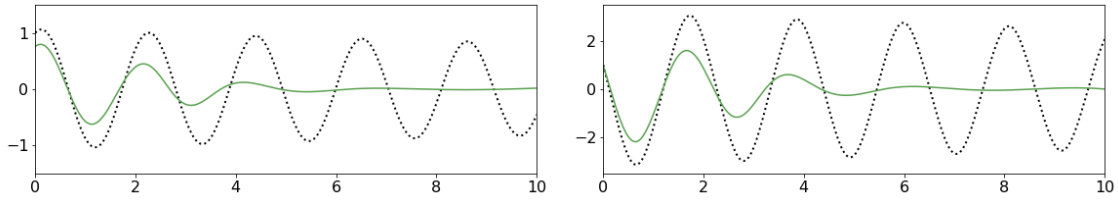


Figure 3: Results from training a single PINN to satisfy Eqs. 16 and 17 (solid lines) compared with the exact solution (dotted line) for s_1 (left) and s_2 (right). The results decay to zero quickly and the learned solution does not agree well with the exact solution.

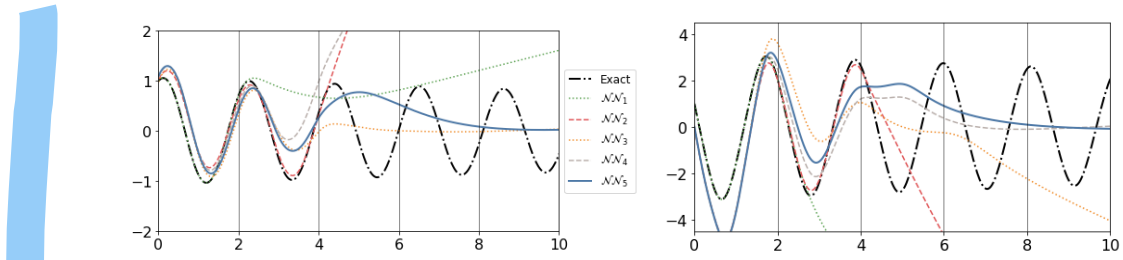
We first consider a single PINN trained in $t \in [0, 10]$ in Fig. 3. We train one PINN with three hidden layers with 200 neurons per layer for 10^5 iterations. The batch size is 100 and the learning rate is given by optimizers.exponentialdecay(1e-3, decaysteps=2000, decayrate=0.95). The solution quickly goes to zero, showing that a single PINN cannot capture the longtime dynamics of even this simple system. Similar results were shown in [22]. We will note that there are recent advances that have been developed for improving the training of PINNs for long-time integration problems [22, 30]. In this section, we will explore how continual learning can also allow for accurate solutions over long times by dividing the time domains into subdomains.

We divide the domain into five subdomains, $\Omega_i = [2(i-1), 2i]$ and train on each domain using both traditional single fidelity continual learning and MF-CL, and SF-CL and MF-CL approaches augmented by RDPS, Replay, and MAS. For each case, we calculate the root mean square error (RMSE) of the final output $\mathcal{NN}_5$ on the full domain, $\Omega = [0, 10]$ by

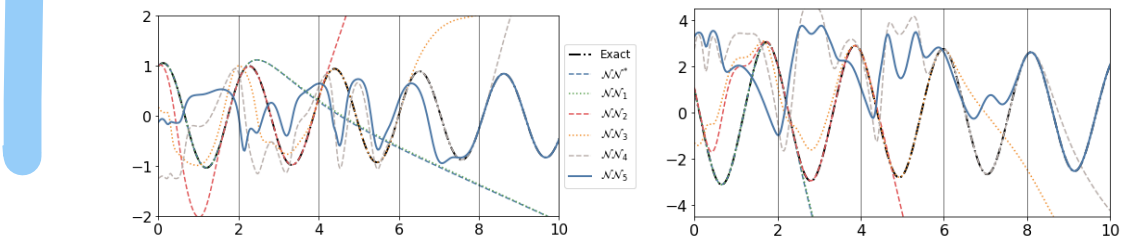
$$error = \sqrt{\frac{1}{N}\sum_{j\in\Omega}^{N} \mathcal{NN}_5(x_j)^2}. \tag{18}$$

If forgetting is limited, the final solution should have a small RMSE on the full domain.

It is clear from Table 1 that replay performs the best in both cases, and significantly better than any other approach. It is no surprise that the no CL and RDPS alone Single Fidelity cases have large RMSEs, as they do not have any incorporation of techniques to limit forgetting. These cases are shown in Fig. 4.

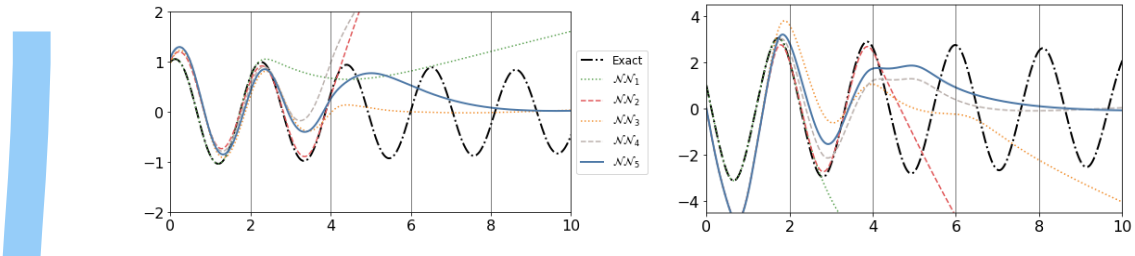


(a) Single fidelity

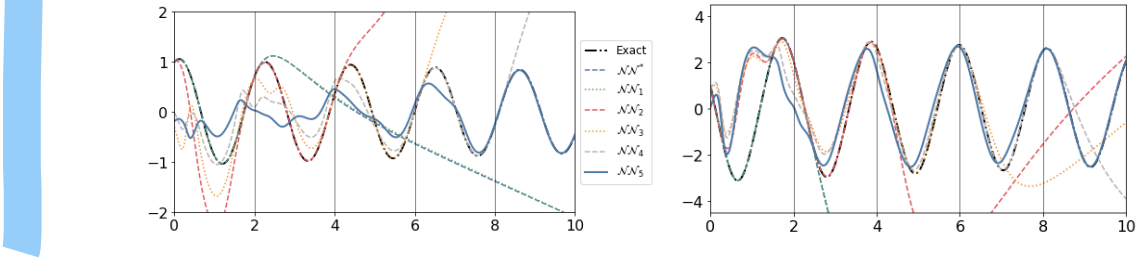


(b) Multifidelity

Figure 4: Results from training the single fidelity (a) and multifidelity (b) alone to satisfy Eqs. 16 and 17 compared with the exact solution (dash-dotted line) for s_1 (left) and s_2 (right). Of particular importance is the final network, $\mathcal{NN}_5$ (blue solid line), which is trained on $\Omega_5 = [8, 10]$. While the multifidelity results in (b) have significant errors, the are substantially better than the single fidelity results in (a). In the single fidelity training, each network $\mathcal{NN}_i$ is only accurate on the subdomain Ω_i, and extrapolation outside Ω_i presents significant difficulties.



(a) Single fidelity

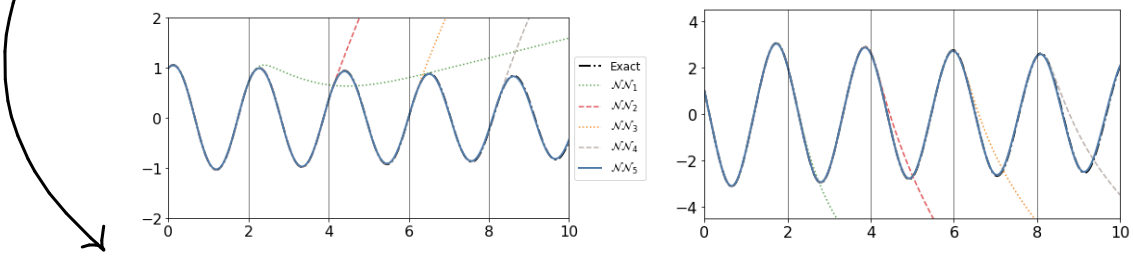


(b) Multifidelity

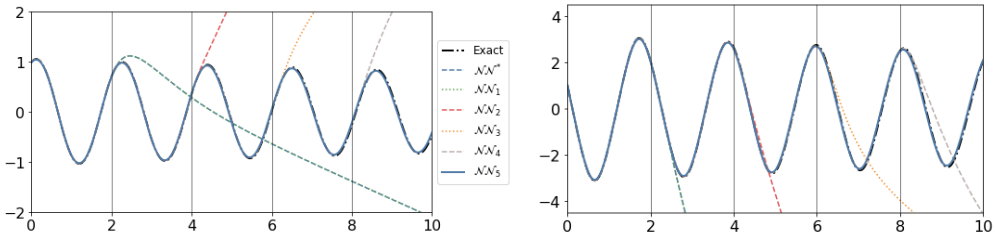
Figure 5: Results from training the single fidelity (a) and multifidelity (b) with MAS to satisfy Eqs. 16 and 17 compared with the exact solution (dash-dotted line) for s_1 (left) and s_2 (right). Of particular importance is the final network, $\mathcal{NN}_5$ (blue solid line), which is trained on $\Omega_5 = [8, 10]$. These simulations plotted here have the smallest RMSEs of $\mathcal{NN}_5$ on Ω of any of the sets of hyperparameters tested. In the single fidelity case, MAS appears to cause restrictions in training that are too strict, and later networks $\mathcal{NN}_i$ are no longer accurate on their respective domains Ω_i. For the multifidelity training, the solutions are accurate across a wider portion of the full domain, and the RMSE is decreased compared with multifidelity training alone.

Fig. 5 gives the best MAS results for each of the sets of hyperparameters considered, MAS alone with $\lambda = 100$ for single fidelity and scaled MAS with RDPS with $\lambda = 0.01$ for multifidelity. As is unsurprising given the smaller RMSE, the multifidelity outperforms the single fidelity training with MAS.

How much data was retrained on for the previous domains?



(a) Single fidelity



(b) Multifidelity

Figure 6: Results from training the single fidelity (a) and multifidelity (b) with Replay-RDPS to satisfy Eqs. 16 and 17 compared with the exact solution (dash-dotted line) for s_1 (left) and s_2 (right). Both cases show very limited forgetting.

N	Single fidelity	Multifidelity
25	2.90	0.059
50	2.63	0.031
100	0.028	0.055
150	2.95	0.061
200	2.88	0.054

Table 2: RMSE of the final output $\mathcal{NN}_5$ on the full domain for the pendulum problem. The SF case has five hidden layers with N neurons each. In the MF case, each nonlinear network has five hidden layers with N neurons. The multifidelity linear network has one hidden layer with 20 neurons.

As shown in Fig. 6, the SF-replay-RDPS case does appear to outperform the MF-replay-RDPS case. However, it is interesting to look at the RMSE as we change the network size in Table 2. While the MF-replay-RDPS case is robust to changes in the network size, the single fidelity case only achieves a small RMSE with a very specific architecture.

3.2 Allen-Cahn equation

The Allen-Cahn equation is given by

$$u_t - c_1^2 u_{xx} + 5u^3 - 5u = 0, \quad t \in (0, 1], x \in [-1, 1] \tag{19}$$

$$u(x, 0) = x^2 \cos(\pi x), \quad x \in [-1, 1] \tag{20}$$

$$u(x, t) = u(-x, t), \quad t \in [0, 1], x = -1, x = 1 \tag{21}$$

$$u_x(x, t) = u_x(-x, t), \quad t \in [0, 1], x = -1, x = 1 \tag{22}$$

We take $c_1^2 = 0.0001$. The Allen-Cahn equation is notoriously difficult for PINNs to solve by direct application [31], see Fig. 7. Modifications of PINNs have successfully been able to solve the Allen-Cahn equation, including by using a discrete Runge-Kutta neural network [21], adaptive sampling of the collocation points [31], and backward compatible PINNs [23]. In this section we show that we can accurately learn the solution to the Allen-Cahn equation by applying the multifidelity continual learning framework.

We divide the domain into four subdomains, $\Omega_i = [2(i-1), 2i]$, and report the RMSE of $\mathcal{NN}_4$ on the full domain Ω. When the multifidelity and single fidelity are trained alone, in Fig. 8, the multifidelity method overall has slightly lower errors than the single fidelity training. In particular, the errors in Ω_1 have significantly lower magnitude with multifidelity training. MAS and replay both improve the results, in Figs. 9 and 10, respectively. The multifidelity

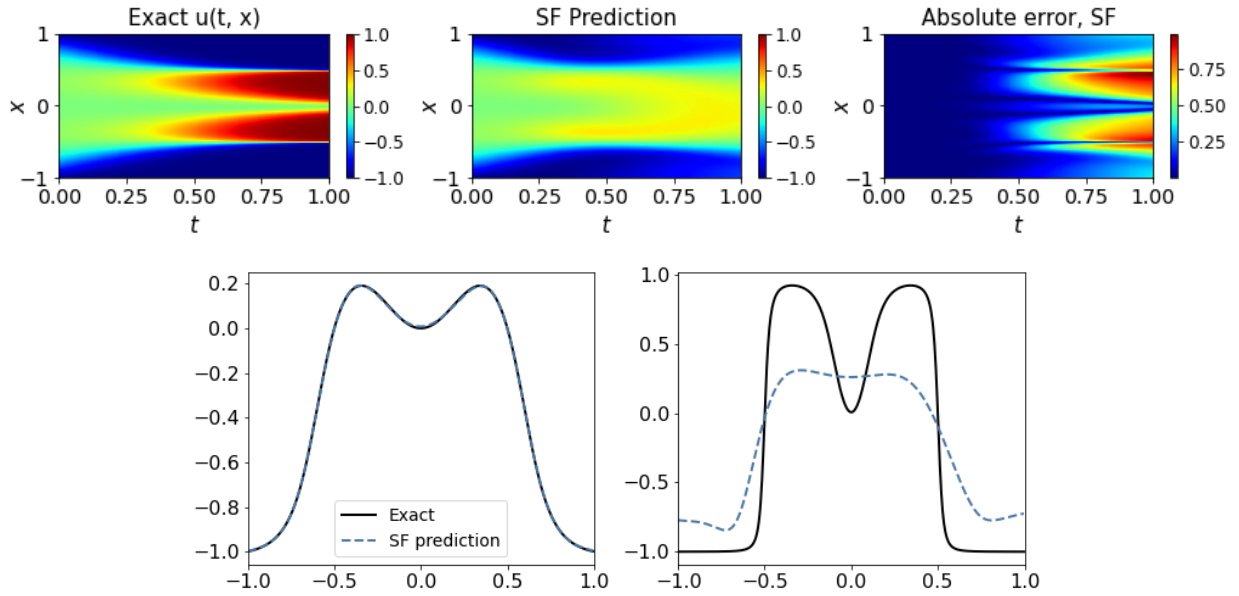


Figure 7: Results from training a single PINN training for the Allen-Cahn equation. The bottom figures are taken at $t = 0.25$ (left) and $t = 0.75$ (right). While the PINN trains well until about 0.3, the solution degrades with increasing t.

	Single fidelity	Multifidelity
No CL	0.15	0.11
RDPS	0.16	0.21
Replay	0.15	0.08
MAS	0.15	0.21
MAS scaled	0.15	0.08
MAS-RDPS	0.16	0.16
MAS scaled-RDPS	0.16	0.06

Table 3: RMSE of the final output $\mathcal{NN}_4$ on the full domain for the Allen-Cahn equation. For the MAS cases, the network is trained for seven values of λ_{MAS}, and the case with the lowest RMSE is shown in the table above.

method has approximately equal RMSE using both methods, suggesting both are good candidates to limit forgetting for this problem. A summary of the results is given in Table 3.

4 Data-informed training

4.1 Batteries

This is a case where if an additional dataset is added, it not is clear a priori which subdomain it lies in. Therefore, it is essential that the final model can predict the current accurately for the entire domain without forgetting.

For testing, a vanadium redox-flow battery (VRFB) system was selected to generate datasets. The left image in Fig. 11 shows a typical configuration of a VRFB, which consists of electrodes, current collectors and a membrane separator. The negative and positive side have a storage tank each to store the redox couple of V^{2+}/V^{3+} and V^{4+}/V^{5+}, respectively. We applied the MFCL method for the problem of identifying the applied charge current from a given charge voltage curve. To generate the VRFB charge curve dataset, a highly computationally efficient 2-D analytical model was utilized [32, 33]. This model fully resolves the coupled physics of active species transport, electrochemical reaction kinetics, and fluid dynamics within the battery cell, thereby providing a faithful representation of the VRFB system. Further details on the model and its parameters can be found in [32]. Typical charge curves are visualized in the right plot of Fig. 11 for five selected current levels. For a given charge current, the battery voltage (E) is calculated at different state-of-charge (SOC) values to form the charge curve which is used as input data. The applied charge current I which gives rise to the charge curve is the output quantity we want to predict.

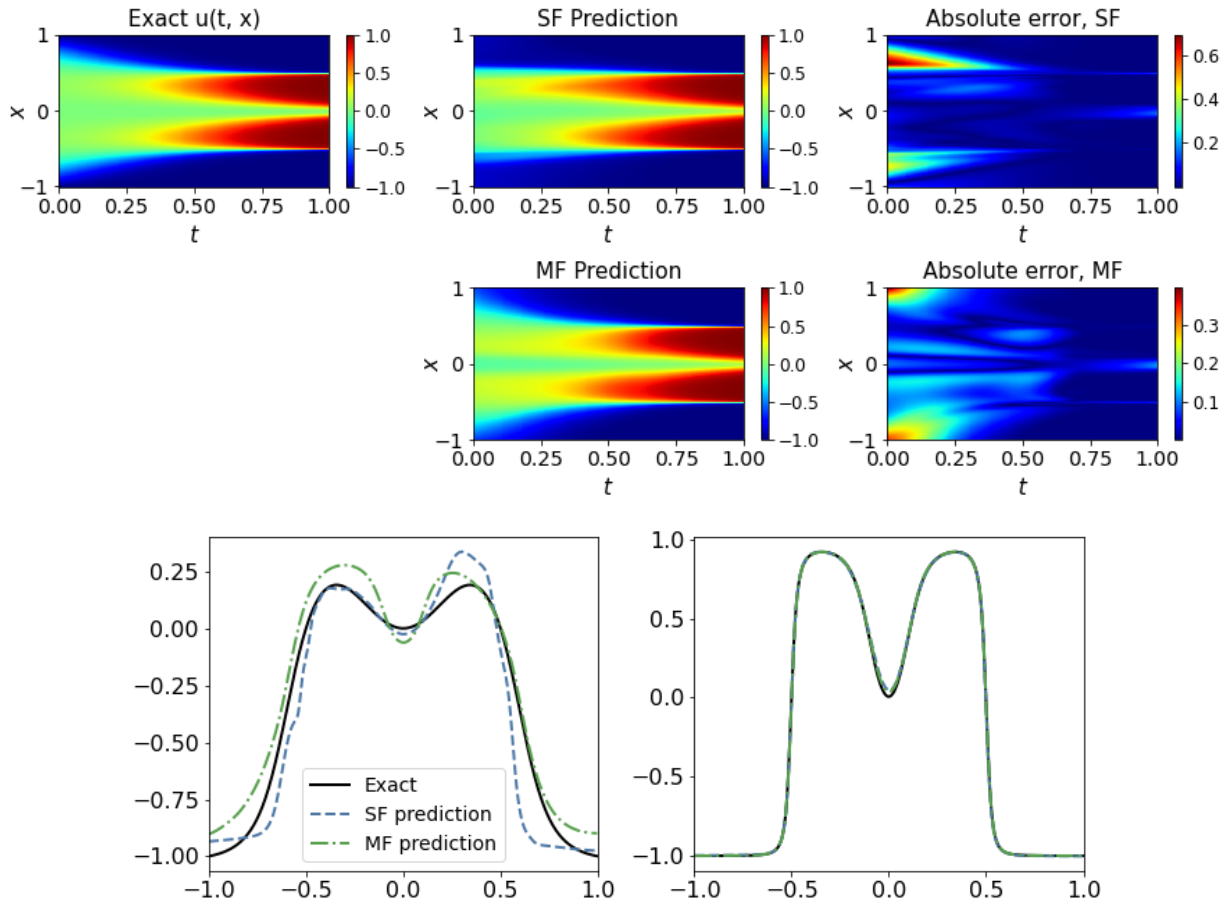


Figure 8: $\mathcal{NN}_4$ results from training a single fidelity and multifidelity PINN training alone for the Allen-Cahn equation. The bottom figures are taken at $t = 0.25$ (left) and $t = 0.75$ (right). The multifidelity results have errors about half as large as those of the single fidelity results.

We divide the data set in five sets by charge current and train with and without MAS in the single fidelity case. The subdomains are $\Omega_1 = [0.1, 2)$, $\Omega_2 = [2, 4)$, $\Omega_3 = [4, 6)$, $\Omega_4 = [6, 8)$, and $\Omega_5 = [8, 9]$. The errors are calculated by the RMSE of the output of $\mathcal{NN}_5$ on a test test selected from $\Omega = \cup_{i=1}^{5} \Omega_i$. We test two network architectures, a wide network which has two hidden layers with 80 neurons each, and a deeper and narrower network which has three hidden layers with 40 neurons each. We first train with the single fidelity and multifdelity approaches alone, see Fig. 12. The multifidelity continual learning results are slightly better than those from the single fidelity continual learning.

We then consider the impact of adding MAS. We consider the narrow and wide networks with and without MAS scaling, for a total of four cases. The multifidelity MAS results show significant improvement, see Fig. 13. In Fig. 14, we compare the performance across the value of the MAS hyperparameter λ_{MAS}. we see that the single fidelity approach performance is robust, since it is insensitive to the value of λ_{MAS}. However, it is not very accurate. On the other hand, the multifidelity approach can be substantially more accurate than the single fidelity approach for most values of λ_{MAS}. Overall, the multifidelity results significantly outperform the single fidelity results.

5 Discussion and future work

We have introduced a novel continual learning method based on multifidelity deep neural networks. The premise of the method is the existence of correlations between the output of previously trained models and the desired output of the model on the current training dataset. The discovery and use of these correlations can limit catastrophic forgetting. On its own, the multifidelity continual learning method has shown robustness and limited forgetting across several datasets for physics-informed and data-driven training examples. Additionally, it can be combined with existing continual

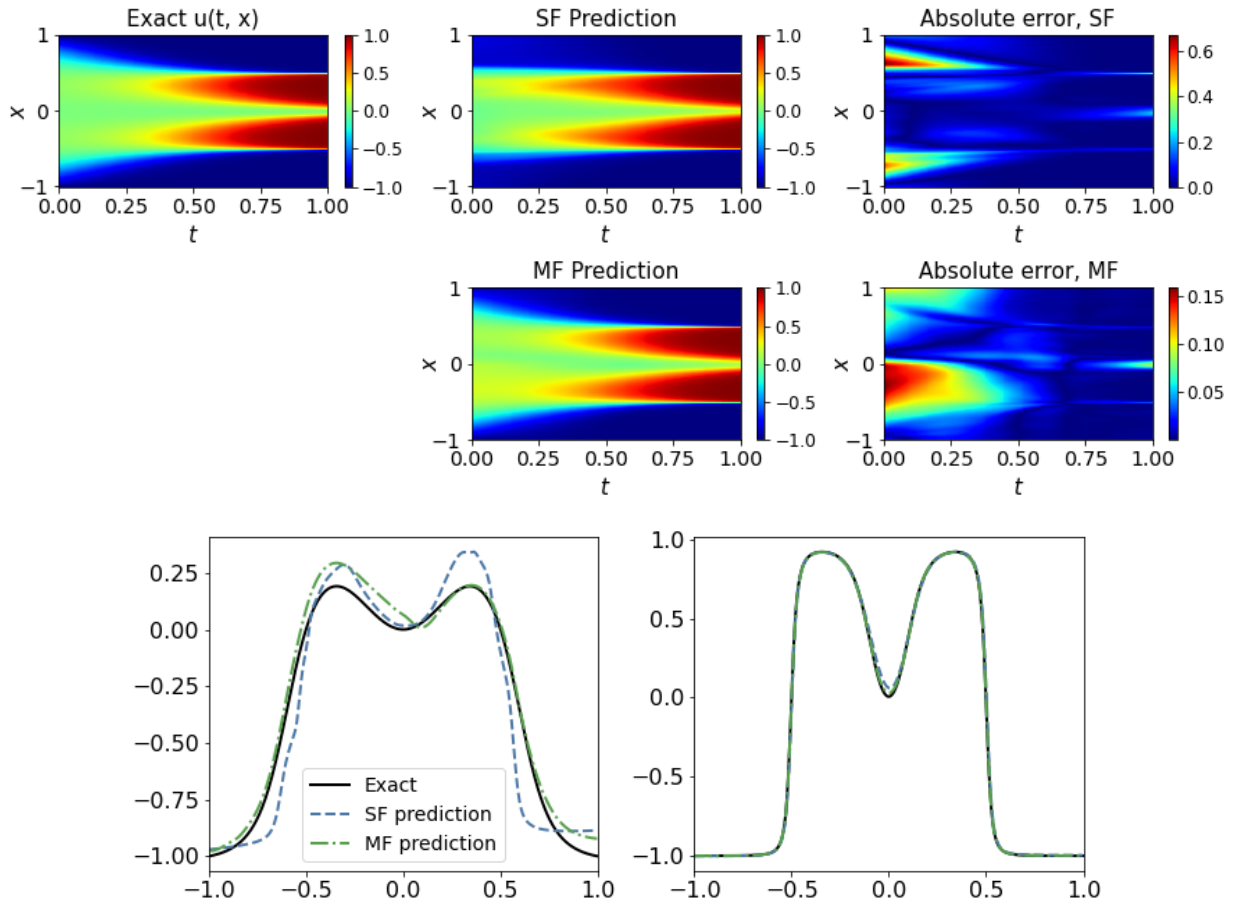


Figure 9: $\mathcal{NN}_4$ results from training a single fidelity and multifidelity PINN training with MAS for the Allen-Cahn equation. The bottom figures are taken at $t = 0.25$ (left) and $t = 0.75$ (right). These results represent the best MAS results from all sets of hyperparameters considered. The multifidelity results have errors about a quarter as large as those of the single fidelity results.

learning methods, including replay and memory aware synapses (MAS), to further limit catastrophic forgetting. In all examples, we show that properly scaling the MAS weights improves the output with multifidelity training.

The proposed continual learning method is especially suited for physical problems where the data satisfy the same physical laws on each domain, or for a physics-informed neural network, because in these cases we expect there to be a strong correlation between the output of the previous model and the model on the current training domain. As a result of exploiting the correlation between data in the various domains instead of training from scratch for each domain, the method can afford to continue learning in new domains using smaller networks. Specifically, its training accuracy is more robust to the size of the network employed in the new domain. This can lead to computational savings during both training and inference. The approach is particularly suited for situations where privacy concerns can limit access to prior datasets. It can also offer new possibilities in the area of federated learning by allowing the design of new algorithms for processing sensor data in a distributed fashion. These topics are under investigation and results will be reported in a future publication.

6 Acknowledgements

This research was supported by the Energy Storage Materials Initiative (ESMI), under the Laboratory Directed Research and Development (LDRD) Program at Pacific Northwest National Laboratory (PNNL). The computational work was performed using PNNL Institutional Computing at Pacific Northwest National Laboratory. PNNL is a multi-program national laboratory operated for the U.S. Department of Energy (DOE) by Battelle Memorial Institute under Contract No. DE-AC05-76RL01830.

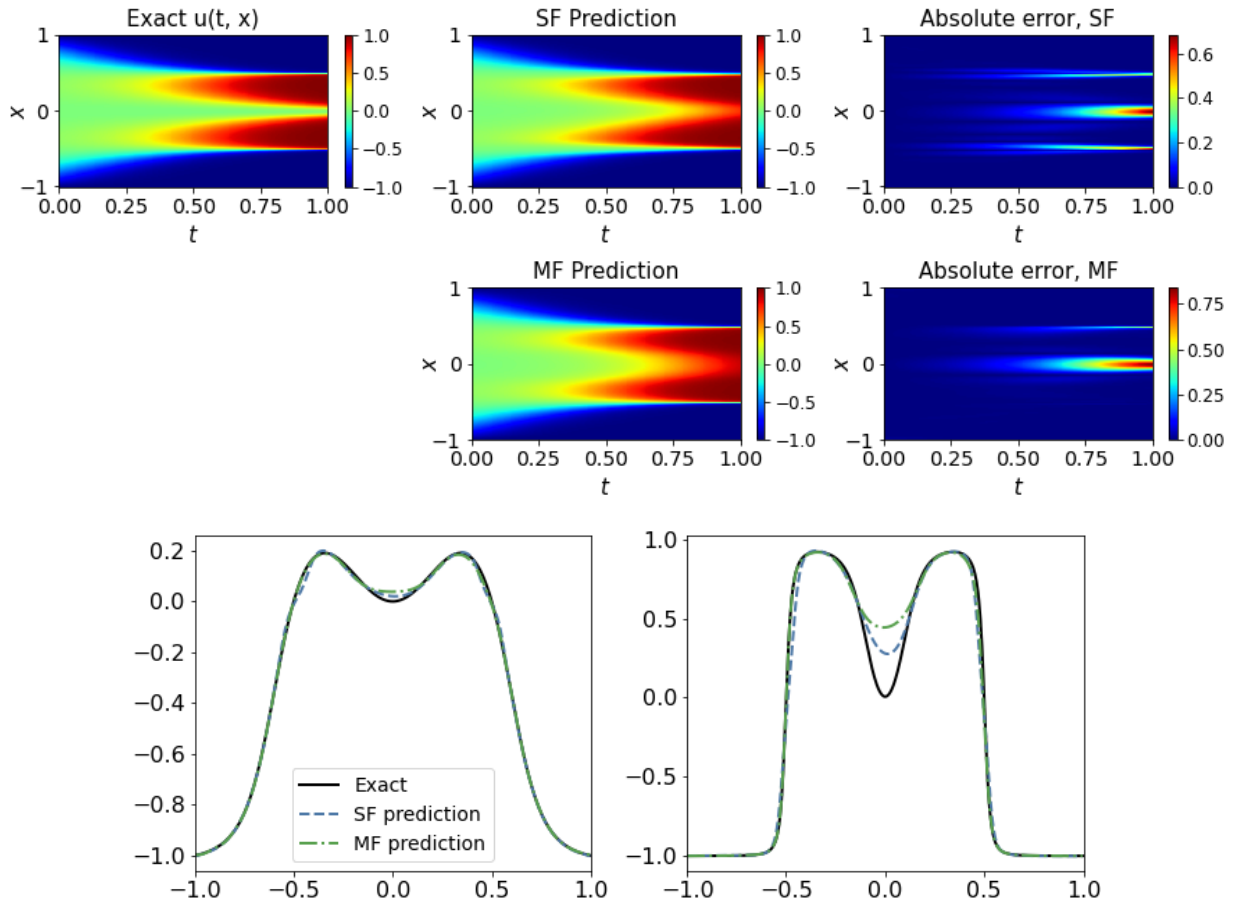


Figure 10: $\mathcal{NN}_4$ results from training a single fidelity and multifidelity PINN training with replay for the Allen-Cahn equation. The bottom figures are taken at $t = 0.25$ (left) and $t = 0.75$ (right).

References

[1] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural networks*, 113:54–71, 2019.

[2] Eli Verwimp, Matthias De Lange, and Tinne Tuytelaars. Rehearsal revealed: The limits and merits of revisiting samples in continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9385–9394, 2021.

[3] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International conference on machine learning*, pages 3987–3995. PMLR, 2017.

[4] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

[5] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 139–154, 2018.

[6] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385, 2021.

[7] Yen-Chang Hsu, Yen-Cheng Liu, Anita Ramasamy, and Zsolt Kira. Re-evaluating continual learning scenarios: A categorization and case for strong baselines. *arXiv preprint arXiv:1810.12488*, 2018.

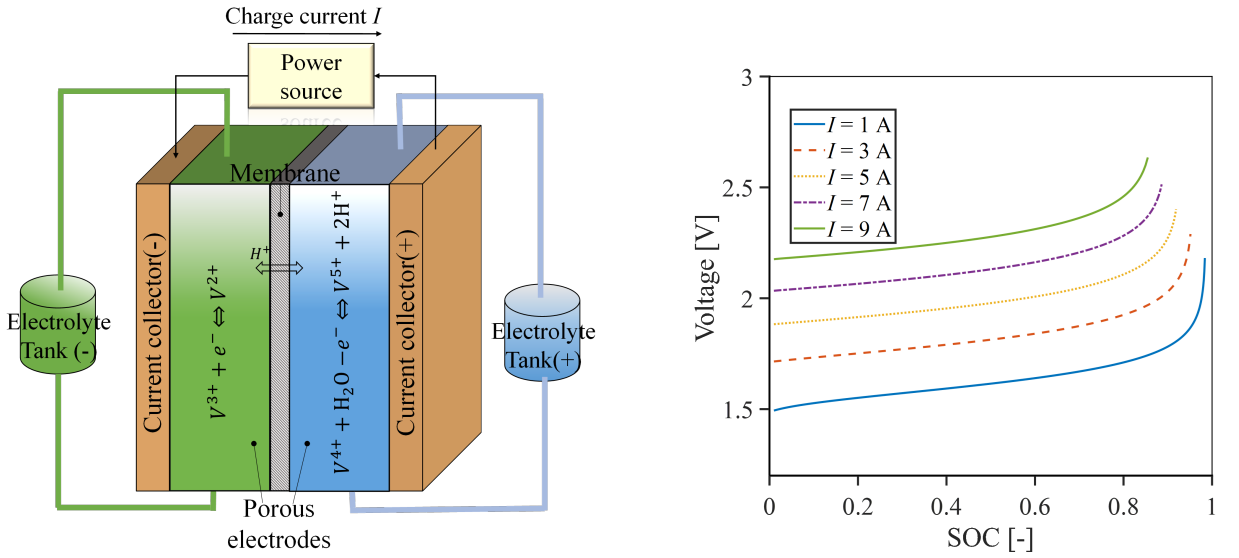


Figure 11: The VRFB system used for battery data generation (left). Sample charge curve distribution at different charge current (right).

[8] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[9] Yeming Wen, Dustin Tran, and Jimmy Ba. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. *arXiv preprint arXiv:2002.06715*, 2020.

[10] Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. Adapterhub: A framework for adapting transformers. *arXiv preprint arXiv:2007.07779*, 2020.

[11] Leonard Bereska and Efstratios Gavves. Continual learning of dynamical systems with competitive federated reservoir computing. In *Conference on Lifelong Learning Agents*, pages 335–350. PMLR, 2022.

[12] Tsendsuren Munkhdalai and Hong Yu. Meta networks. In *International conference on machine learning*, pages 2554–2563. PMLR, 2017.

[13] Max Vladymyrov, Andrey Zhmoginov, and Mark Sandler. Continual few-shot learning using hypertransformers. *arXiv preprint arXiv:2301.04584*, 2023.

[14] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

[15] Maziar Raissi, Alireza Yazdani, and George Em Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367(6481):1026–1030, 2020.

[16] Shengze Cai, Zhiping Mao, Zhicheng Wang, Minglang Yin, and George Em Karniadakis. Physics-informed neural networks (pinns) for fluid mechanics: A review. *Acta Mechanica Sinica*, 37(12):1727–1738, 2021.

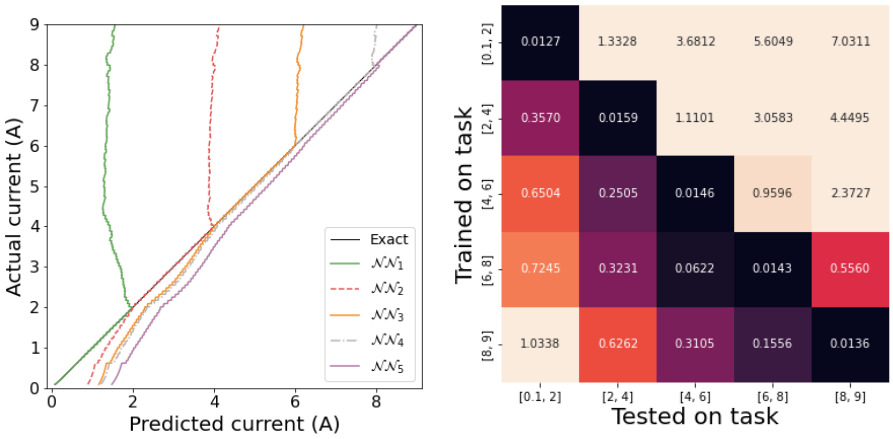
[17] Dehao Liu and Yan Wang. Multi-fidelity physics-constrained neural network and its application in materials modeling. *Journal of Mechanical Design*, 141(12), 2019.

[18] Yuyao Chen, Lu Lu, George Em Karniadakis, and Luca Dal Negro. Physics-informed neural networks for inverse problems in nano-optics and metamaterials. *Optics express*, 28(8):11618–11633, 2020.

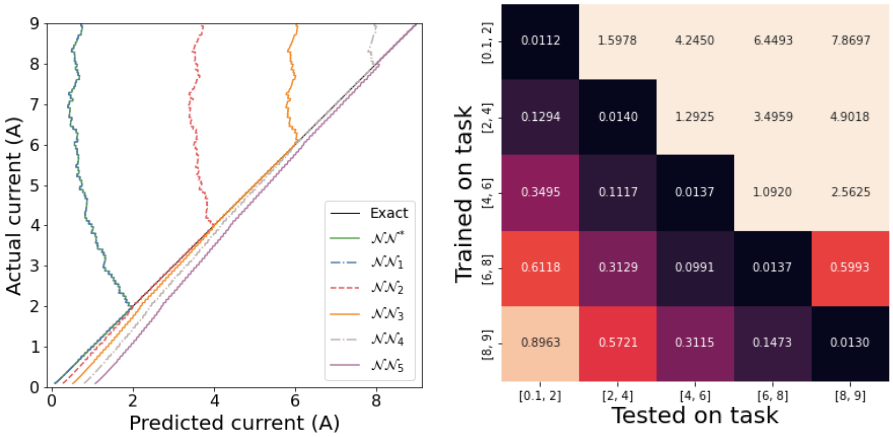
[19] Zhiwei Fang and Justin Zhan. Deep physical informed neural networks for metamaterial design. *IEEE Access*, 8:24506–24513, 2019.

[20] Zhiping Mao, Ameya D Jagtap, and George Em Karniadakis. Physics-informed neural networks for high-speed flows. *Computer Methods in Applied Mechanics and Engineering*, 360:112789, 2020.

[21] 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.



(a) Single fidelity



(b) Multifidelity

Figure 12: Results from the single fidelity (a) and multifidelity (b) training alone for the battery test case. The left column has the network outputs of each task on all the tasks, and the right column shows the RMSE of each task tested on each other task.

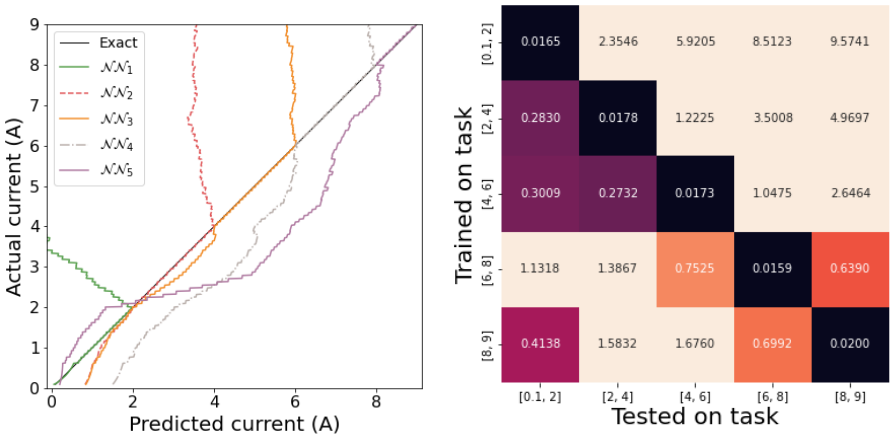
[22] Sifan Wang and Paris Perdikaris. Long-time integration of parametric evolution equations with physics-informed deeponets. *Journal of Computational Physics*, 475:111855, 2023.

[23] Revanth Mattey and Susanta Ghosh. A novel sequential method to train physics informed neural networks for allen cahn and cahn hilliard equations. *Computer Methods in Applied Mechanics and Engineering*, 390:114474, 2022.

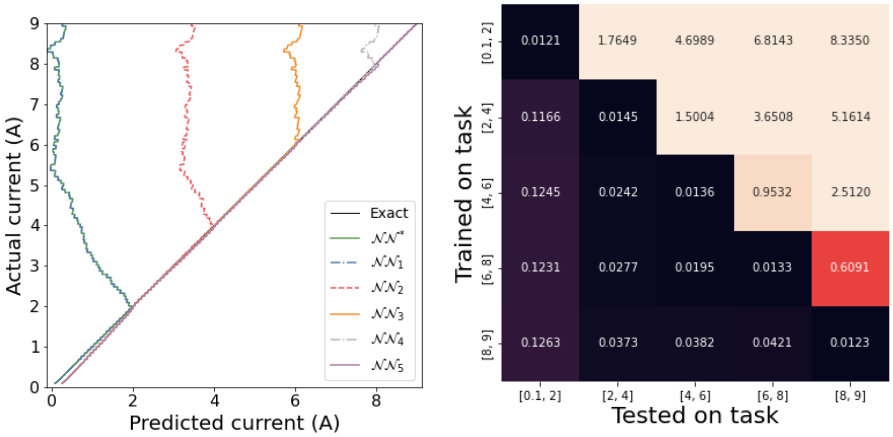
[24] Majid Rasht-Behesht, Christian Huber, Khemraj Shukla, and George Em Karniadakis. Physics-informed neural networks (pinns) for wave propagation and full waveform inversions. *Journal of Geophysical Research: Solid Earth*, 127(5):e2021JB023120, 2022.

[25] Xuhui Meng and George Em Karniadakis. A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse pde problems. *Journal of Computational Physics*, 401:109020, 2020.

[26] Frederik Benzing. Unifying regularisation methods for continual learning. *arXiv preprint arXiv:2006.06357*, 2020.



(a) Single fidelity



(b) Multifidelity

Figure 13: Results from the single fidelity (a) and multifidelity (b) training with MAS for the battery test case. The single fidelity case struggles to train accurately, while multifidelity has very limited forgetting. The left column has the network outputs of each task on all the tasks, and the right column shows the RMSE of each task tested on each other task. The results shown represent the best output from the MAS hyperparameters tested. For the single fidelity case, the results are from the narrow network with $\lambda = 10^{-2}$, and for the multifidelity case, the results are from the narrow network with MAS scaling with $\lambda = 10^{-1}$.

[27] Chenxi Wu, Min Zhu, Qinyang Tan, Yadhu Kartha, and Lu Lu. A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 403:115671, 2023.

[28] Mohammad Amin Nabian, Rini Jasmine Gladstone, and Hadi Meidani. Efficient training of physics-informed neural networks via importance sampling. *Computer-Aided Civil and Infrastructure Engineering*, 36(8):962–977, 2021.

[29] Sinan Özgün, Anne-Marie Rickmann, Abhijit Guha Roy, and Christian Wachinger. Importance driven continual learning for segmentation across domains. In *Machine Learning in Medical Imaging: 11th International Workshop, MLMI 2020, Held in Conjunction with MICCAI 2020, Lima, Peru, October 4, 2020, Proceedings 11*, pages 423–433. Springer, 2020.

[30] Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why pinns fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022.

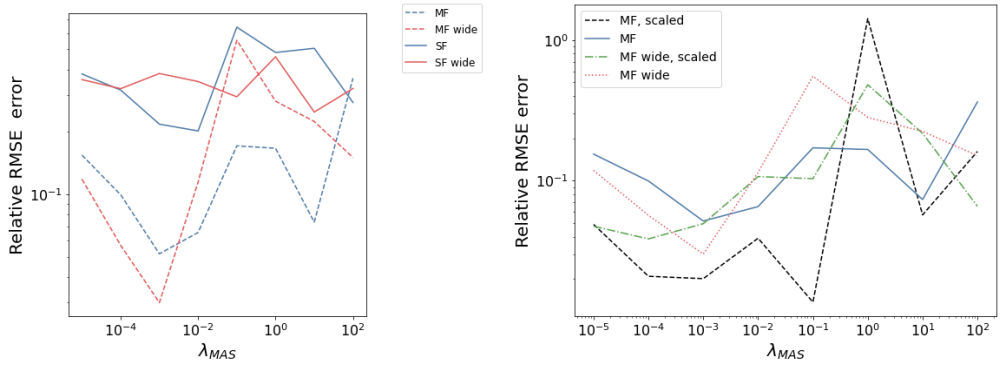


Figure 14: Comparison of the relative RMSE with MAS. On the left we compare the single fidelity and multifidelity frameworks without scaling the MAS matrices. On the right, we compare the four multifidelity MAS architectures.

[31] Colby L Wight and Jia Zhao. Solving allen-cahn and cahn-hilliard equations using the adaptive physics informed neural networks. *arXiv preprint arXiv:2007.04542*, 2020.

[32] Yunxiang Chen, Jie Bao, Zhijie Xu, Peiyuan Gao, Litao Yan, Soowhan Kim, and Wei Wang. A two-dimensional analytical unit cell model for redox flow battery evaluation and optimization. *Journal of Power Sources*, 506:230192, 2021.

[33] Yunxiang Chen, Zhijie Xu, Chao Wang, Jie Bao, Brian Koeppel, Litao Yan, Peiyuan Gao, and Wei Wang. Analytical modeling for redox flow battery design. *Journal of Power Sources*, 482:228817, 2021.