ARTICLE OPEN

Learning two-phase microstructure evolution using neural operators and autoencoder architectures

Vivek Oommen[1], Khemraj Shukla[2], Somdatta Goswami[2], Rémi Dingreville[3]✉ and George Em Karniadakis[1,2]✉

Phase-field modeling is an effective but computationally expensive method for capturing the mesoscale morphological and microstructure evolution in materials. Hence, fast and generalizable surrogate models are needed to alleviate the cost of computationally taxing processes such as in optimization and design of materials. The intrinsic discontinuous nature of the physical phenomena incurred by the presence of sharp phase boundaries makes the training of the surrogate model cumbersome. We develop a framework that integrates a convolutional autoencoder architecture with a deep neural operator (DeepONet) to learn the dynamic evolution of a two-phase mixture and accelerate time-to-solution in predicting the microstructure evolution. We utilize the convolutional autoencoder to provide a compact representation of the microstructure data in a low-dimensional latent space. After DeepONet is trained in the latent space, it can be used to replace the high-fidelity phase-field numerical solver in interpolation tasks or to accelerate the numerical solver in extrapolation tasks.

npj Computational Materials (2022)8:190 ; https://doi.org/10.1038/s41524-022-00876-7

INTRODUCTION

The phase-field method has emerged as a powerful, heuristic tool for modeling and predicting mesoscale microstructural evolution in a wide variety of material processes[1–5]. This method models interfacial dynamics without the overhead of resorting to advanced interfacial tracking algorithms such as level-set[6] or adaptive meshing[7]. Scalar, auxiliary, continuous field variables (so-called phase-field variables) are used to represent the evolutionary state of the microstructure dynamics such as in crack growth and propagation[8,9], thin-film deposition[10,11], and dislocation dynamics[12] to name a few. The Cahn-Hilliard, nonlinear diffusion equation[1,13,14], is one of the most commonly used governing equations in phase-field models. It describes the process of phase separation, by which a two-phase mixture spontaneously separates and form domains pure in each component. The Cahn-Hilliard equation finds applications in diverse fields ranging from complex fluids to soft matter and serves as the starting point of many phase-filed models for microstructure evolution.

Traditional numerical approaches to solve the fourth-order parabolic Cahn-Hilliard equation include finite differences[15], spectral approximation[16], finite element analysis with mixed methods[17], and isogeometric analysis[18,19]. The coupled stiff equation simultaneously captures a quick phase separation and a very slow coalescence. Evidently, the two sub-processes operate on significantly different spatial and temporal scales, making it challenging to solve efficiently and accurately within realistic time constraints and reasonable computational capabilities[20]. Improvements in computational complexity have been enabled by the growing interest in data-driven models using machine learning (ML) methods. However, striking a balance between computational efficiency and accuracy has often been a challenge while employing these methods. Indeed, for complex and multi-variate phase-field models, the efficient Green's function[21] does not ensure an accurate solution, while Bayesian optimization[22,23] techniques solve such coupled models but to the detriment of a higher computational cost.

Modern ML models have paved the way for the development of fast emulators for solving parametric partial differential equations (PDEs)[23–36]. There are strategies for accelerating the simulation of PDEs. A promising approach for accelerating the predictions of phase field-based microstructure evolution problems consists of using recurrent neural networks (RNNs) to learn the time-dependent, microstructure evolution in latent space[37,38]. Within this framework, statistical functions combined with linear and nonlinear embedding techniques are used to represent the microstructure evolution in latent space. Such RNN-based surrogate models demonstrated success in generating rapid predictions of the time evolution of the microstructural autocorrelation function. The microstructure reconstructed from these statistical functions, using for instance a phase recovery algorithm[39], was then used as an input for a high-fidelity solver that marches ahead in time. The developed approach reported a 5% loss in accuracy against the high-fidelity phase-field solvers. However, this class of models also comes with challenges. First, the training and inference using RNNs as a surrogate model can be relatively slow due to the temporal dependence of the current predicted field on fields predicted at previous time steps, prohibiting the efficiency of the algorithm for large datasets. Second, the RNN-based architecture learns the underlying evolutionary dynamics in terms of statistical functions (non-primitive variables) of the microstructure. Reconstructing a microstructure from these statistical functions is a non-trivial and ill-posed problem[40]. This reconstruction step can incur additional errors especially for interfacial dynamics problems where resolving intricate spatial length scales such as in dendrite growth phase-field problems is key.

In this work, we propose an alternative approach to circumvent the aforementioned challenges. We formulate the microstructure evolution problem as being equivalent to learning a mapping

[1]School of Engineering, Brown University, Providence, RI, USA. [2]Division of Applied Mathematics, Brown University, Providence, RI, USA. [3]Center for Integrated Nanotechnologies, Sandia National Laboratories, Albuquerque, NM 87185, USA. ✉email: rdingre@sandia.gov; george_karniadakis@brown.edu

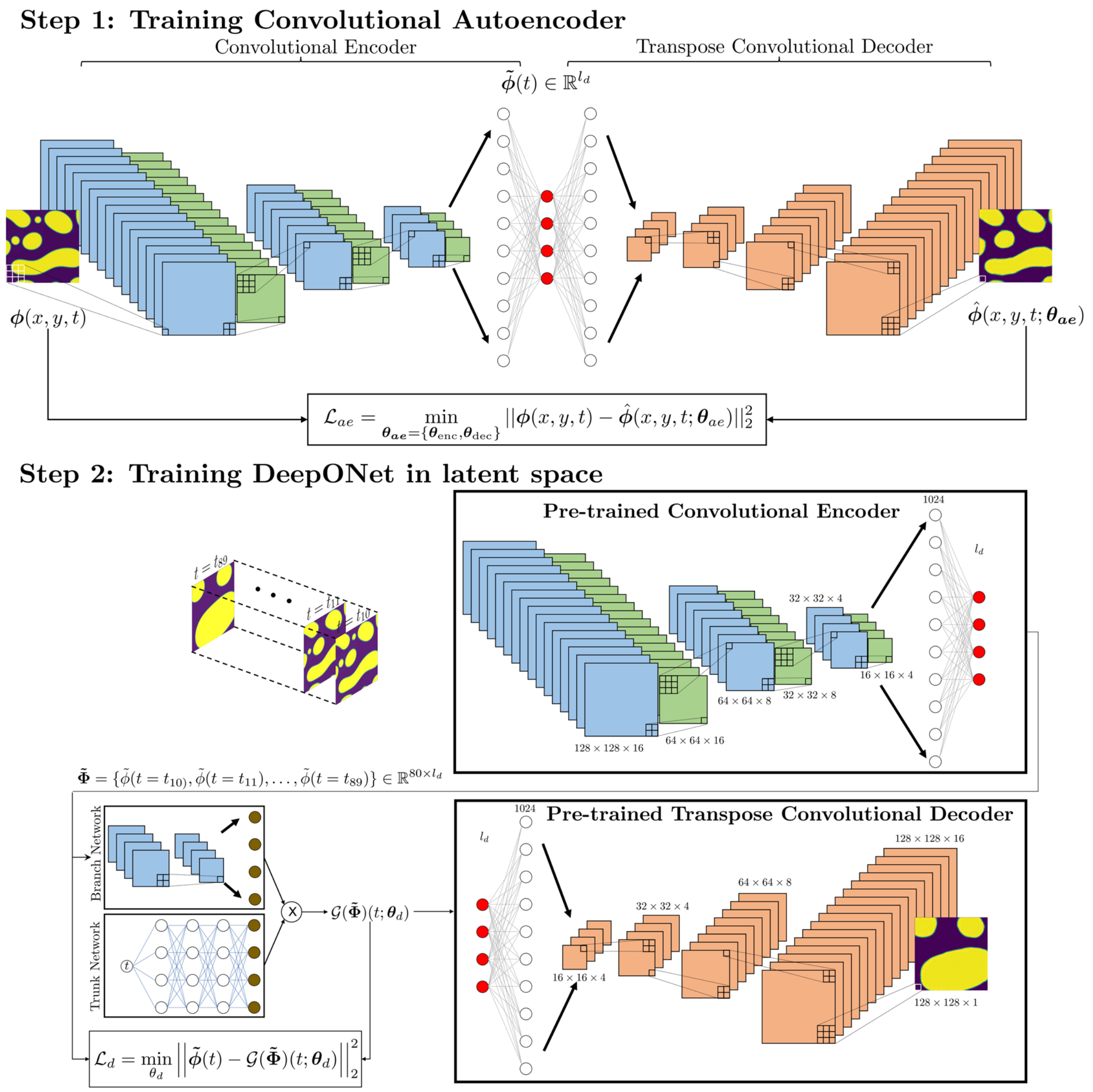

Step 1: Training Convolutional Autoencoder
Convolutional Encoder
Transpose Convolutional Decoder
$\tilde{\phi}(t) \in \mathbb{R}^{l_d}$
$\phi(x,y,t)$
$\hat{\phi}(x,y,t;\theta_{ae})$
$\mathcal{L}_{ae} = \min_{\theta_{ae}=\{\theta_{enc},\theta_{dec}\}} ||\phi(x,y,t) - \hat{\phi}(x,y,t;\theta_{ae})||_2^2$
Step 2: Training DeepONet in latent space
Pre-trained Convolutional Encoder
$t = t_{89}$
$t = t_{11}$
$t = t_{10}$
1024
l_d
32 × 32 × 4
16 × 16 × 4
64 × 64 × 8
32 × 32 × 8
128 × 128 × 16
64 × 64 × 16
$\tilde{\Phi} = \{\tilde{\phi}(t=t_{10}), \tilde{\phi}(t=t_{11}), \ldots, \tilde{\phi}(t=t_{89})\} \in \mathbb{R}^{80 \times l_d}$
Branch Network
Trunk Network
t
$\mathcal{G}(\tilde{\Phi})(t;\theta_d)$
$\mathcal{L}_d = \min_{\theta_d} \left|\left|\tilde{\phi}(t) - \mathcal{G}(\tilde{\Phi})(t;\theta_d)\right|\right|_2^2$
Pre-trained Transpose Convolutional Decoder
1024
l_d
128 × 128 × 16
64 × 64 × 8
32 × 32 × 4
16 × 16 × 4
128 × 128 × 1

Fig. 1 Schematic representation of DeepONet with convolutional autoencoder. Step 1 involves training of the convolutional autoencoder to minimize $\mathscr{L}_{\mathrm{ae}}$. The encoder learns a suitable transformation from the high-dimensional microstructure to a low-dimensional latent space through a series of convolution (blue layers) and MaxPooling (green layers) operations. The decoder remaps the latent representation of the microstructure back to the original, real space by performing transpose convolution (orange layers) operations. A detailed description of the architecture is provided in Table 1. In step 2, we train the DeepONet in the latent space to minimize $\mathscr{L}_{\mathrm{d}}$. The entire history of 80 steps is encoded by the pre-trained convolutional encoder as $\tilde{\Phi}$. DeepONet learns to predict $\tilde{\phi}(t)$ at any desired time t, fed to the trunk network. The latent representation of the microstructure predicted by DeepONet is then re-mapped back to the primitive space by the transpose convolutional decoder.

function $\mathscr{G} : \boldsymbol{u} \rightarrow \boldsymbol{\phi}$ such that,

$$\mathscr{G}(\boldsymbol{u}(x,y,t)) = \boldsymbol{\phi}(x,y,t), \tag{1}$$

where $\boldsymbol{u}$ is the history of the microstructure evolution and $\boldsymbol{\phi}(x,y,t)$ is the state of the microstructure at time t. We develop a framework that integrates a convolutional autoencoder architecture with a Deep Operator Network[41] (DeepONet) to learn this mapping. Figure 1 illustrates the complete end-to-end workflow of the proposed algorithm. We utilize a convolutional autoencoder to provide a compact representation of the microstructure data in a low-dimensional, latent space. This convolutional autoencoder approach is then combined with the DeepONet architecture to learn the dynamics of two-phase microstructures in the autoencoder latent space. The DeepONet architecture has demonstrated its ability to model the governing differential equations (ordinary differential equations (ODEs) and PDEs) of such problems by learning the underlying operator, a mapping from functions to functions, from the available datasets for a broad range of problems[28,42]. We show that such an architecture is more robust than the RNN-based architecture in terms of training, computational efficiency, and sensitivity to noise. The decoder part of the convolutional autoencoder can efficiently reconstruct the time-evolved microstructure from the DeepONet predictions bypassing the challenges associated with reconstruction-induced errors when using statistical functions to represent the microstructure for instance. Overall, the trained autoencoder–DeepONet framework can then be used to replace the high-fidelity phase-field numerical solver in interpolation tasks for parameters inside the distribution of inputs used during training or to accelerate the numerical solver in extrapolation tasks for parameters outside this distribution.

RESULTS

Training and optimization of neural operators and autoencoder architectures

We first investigated the impact of the size of the latent dimension of the autoencoder, l_d, on the model performance. To this end, we trained five autoencoder models with $l_d = 9$, 25, 64, 100, and 196 respectively. Details of the hyper-parameters used in these five convolutional autoencoders are provided in Table 1. For any given time step during the evolution of the microstructure, the encoder reduced a 128×128 microstructure $\phi(x, y, t)$ to a latent vector of size l_d. The decoder mapped the microstructural latent space representation back to a 128×128 microstructure $\hat{\phi}(x, y, t)$ (see Methods for more details). Each autoencoder training took approximately 33 h on one NVIDIA GeForce RTX 3090 GPU. Next, we trained the DeepONet model for 120,000 epochs on the latent space learned by the convolutional encoder for each of the five trained autoencoder models. The last layer of the branch and trunk networks for all the models uses a linear activation function. The output of the DeepONet model was then sent to the trained convolutional decoder, which performed a mapping from the latent space back to the original microstructure space, $\hat{\phi}(x, y, t)$.

We evaluated the effect of the size of the latent dimension of each of the models on the basis of the relative L^2 norm computed across the training and testing dataset for all the time steps, including the forecasting time frames, $t = \{t_{90}, \ldots, t_{99}\}$ not seen by the surrogate model (Note that one time frame is equal to 500,000 time steps, $t = 500{,}000\Delta t$, see Methods for additional details). All the details of this survey analysis, including the DeepONet architecture, the L^2 norm of relative error on train and test datasets, and the computational time taken for training the DeepONet model are reported in Table 2. From this survey, we observe that the model predictions improve when we increase the size of the latent dimension. In general, DeepONet models with tanh and sin activation functions performed better compared to models with a ReLU activation for this particular class of problems. As such, our best model consists of a convolutional autoencoder with $l_d = 196$ and a DeepONet model with architecture 1 and sin activation function (shown in Table 2). Although the training dataset consists of 1,600 different microstructure-evolution trajectories, each represented by over 80 snapshots from $t = \{t_{10}, t_{11}, \ldots, t_{89}\}$, the DeepONet training is faster compared to popular RNN architectures such as the Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU) networks[38,42,43]. Since DeepONet does not have recurrent connections, there are no temporal dependencies during the training or at the inference stage. Instead, the network relies on the convolution operations that encode information about the history through the branch network. In addition, due to the lack of temporal dependencies, the fully connected layers in the trunk network and convolutional layers in the branch network of the DeepONet architecture can be easily parallelized, unlike LSTMs. This makes training and inference of DeepONet significantly faster than the RNN architectures.

We carried out additional simulations to analyze the sensitivity of the proposed approach to the number of samples used for training. We considered training datasets with 25%, 50%, 75% and all the 1600 training samples. We adopted the same methodology proposed in Methods and trained separate autoencoder–DeepONet models on each of these datasets. Details can be found in Supplementary Note 2. The model performance was evaluated on the basis of forecasting errors on test data, shown in Supplementary Figure 2. As expected, we observe better accuracy in the model predictions when increasing the number of training samples. The model trained with 1200 data samples shows similar accuracy to the best model trained with 1,600 data samples, indicating convergence of the training procedure.

Finally, we also evaluated the effect of using different loss functions on training the autoencoder models. Specifically, we trained various autoencoders by minimizing L^1 loss, relative L^1 loss, L^2 loss, relative L^2 loss, and mixed loss (L^2 loss for the initial 5000 epochs and L^1 loss for the remaining epochs). The choice of a loss function determines the landscape in a hyperspace for the optimizer to traverse in pursuit of global minima/best local minima and avoiding the saddle points. For this task, we used a DeepONet model with architecture 1 (Table 2) on each of the learned latent microstructure data and re-transformed the DeepONet predictions using a pre-trained decoder to retrieve the microstructure. We analyzed the model performance by computing the forecasting error, $\mathscr{D}_{\text{test}}(t)$, on unseen test data, as shown in Supplementary Note 3. We observed that models for which the autoencoder is trained on L^2 loss performed better than the one which used L^1 loss. When there are no outliers as solutions, L^2 loss is expected to perform better than L^1. In the presence of outliers, L^2 squares them as compared to linear contribution in the L_1 norm. Similarly, mean values of relative L^1 and L^2 are a better choice for autoencoder loss, $\mathscr{L}_{ae}$, than the mean of L^1 and L^2, respectively. The relative loss values are always of $\mathcal{O}(1)$ and help in achieving convergence faster as the learning rate is of $\mathcal{O}(10^{-3})$. We have observed such an improvement in convergence in other problems, e.g. electro-convection, where we had sharp interfaces and multiscale dynamics[44]. Overall, all the models performed consistently well. As such, our autoencoder architecture of choice is an autoencoder with $l_d = 196$ using a

Table 1. Details of the hyper-parameters used in the convolutional autoencoder.

	Layer	Kernel Size	Width	Activation	Output
1	Conv2D	3×3	16	ReLU	$128 \times 128 \times 16$
2	Max-Pool	2×2			$64 \times 64 \times 16$
3	Conv2D	3×3	8	ReLU	$64 \times 64 \times 8$
4	Max-Pool	2×2			$32 \times 32 \times 8$
5	Conv2D	3×3	4	ReLU	$32 \times 32 \times 4$
6	Max-Pool	2×2			Reshaped to 1024
7	Fully connected		l_d	Linear	l_d
8	Fully connected		1024	ReLU	Reshaped to $16 \times 16 \times 4$
9	Transpose Conv2D	2×2		ReLU	$32 \times 32 \times 4$
10	Transpose Conv2D	2×2		ReLU	$64 \times 64 \times 8$
11	Transpose Conv2D	2×2		ReLU	$128 \times 128 \times 16$
12	Transpose Conv2D	3×3		ReLU	$128 \times 128 \times 1$

l_d is the dimension of latent space.

Table 2. Detailed survey of different latent dimension size, l_d, network architecture, and non-linear activation functions.

l_d	DeepONet Architecture	Activation	$\mathscr{D}_{train}$	$\mathscr{D}_{test}$	DeepONet training time per 1000 epochs (s)
		ReLU	0.03621	0.06803	56
196	Architecture 1	tanh	0.02177	0.06233	56
		sin	0.01408	0.01620	57
		ReLU	0.04076	0.05991	31
100	Architecture 2	tanh	0.03196	0.04699	30
		sin	0.02684	0.03679	31
		ReLU	0.06708	0.07791	31
64	Architecture 3	tanh	0.04773	0.06013	35
		sin	0.04781	0.05739	32
		ReLU	0.16527	0.20097	19
25	Architecture 4	tanh	0.16507	0.20134	19
		sin	0.16551	0.20167	20
		ReLU	0.31536	0.3186	10
9	Architecture 5	tanh	0.31523	0.31903	11
		sin	0.31539	0.31876	11

Architecture	Branch Network	Trunk Network
1	$3 \times$ [conv(32,(3,3))] + [1960]	$2 \times$ [100] + [1960]
2	$2 \times$ [conv(32,(3,3))] + [1100]	$2 \times$ [100] + [1100]
3	$2 \times$ [conv(32,(3,3))] + [512]	$2 \times$ [100] + [512]
4	$1 \times$ [conv(64,(3,3))] + [500]	$2 \times$ [100] + [500]
5	$1 \times$ [conv(128,(3,3))] + [180]	$2 \times$ [100] + [180]

relative L^2 loss function. Taken together, these results demonstrate not only the ability of our framework to accurately provide a compact representation of the microstructure data in a low-dimensional latent space, but they also illustrate the robustness of the training of this framework.

Performance accuracy and forecasting ability

A comparison of the predictions from our accelerated framework with that from high-fidelity phase-field simulations for a representative case of microstructure evolution at three different time steps is shown in Fig. 2. During the initial time steps, the microstructure is rich with multiple features and evolves rapidly with respect to time. Our autoencoder–DeepONet as a surrogate model is able to successfully predict the larger features and the overall morphology of the microstructure. The point-wise error snapshots suggest that the model fails to identify the relatively smaller features in the microstructure and contains significant errors along the sharp boundaries. In other words, the spatial gradient of the phase concentration is not as sharp as that of the true microstructure obtained from high-fidelity, phase-field simulations.

From Fig. 2, we qualitatively get the intuition that the predicted microstructures contain errors at the earlier time steps because of the missing, small-size features and at the later time steps due to smoother boundaries predicted by the model. To confirm and quantify this notion, we computed the L^2 norm of the relative error at each time step, $\mathscr{D}(t)$, defined as:

$$\mathscr{D}(t) = \frac{\sum_x \sum_y \left(\phi(x,y,t) - \hat{\phi}(x,y,t;\boldsymbol{\theta})\right)^2}{\sum_x \sum_y \phi(x,y,t)^2}, t \in [t_{10}, t_{11}, \ldots, t_{99}]. \tag{2}$$

To analyze the accuracy of the prediction at each time step, we calculated $\mathscr{D}(t)$ across the samples in the training and testing datasets, and created a boxplot as shown in Fig. 3. The error is high for the initial time steps, where features span multiple length scales and evolve rapidly with time. However, the predictions improve over time when the evolution process slows down and the microstructure features coarsen. The time steps shown in Fig. 3a were used during the training of the model.

Next, we evaluated the capability of the model to forecast time frames $t = \{t_{90}, t_{91}, \ldots, t_{99}\}$. From Fig. 3b, the error is seen to increase gradually when the model extrapolates at unseen time instances. A closer look at the forecasting predictions offers further insights into the DeepONet predictive performance. We computed the mean of $\mathscr{D}(t)$ across the training and testing datasets for all the models given in Table 2. We also plotted these values in Supplementary Fig. 4 with additional details in Supplementary Note 4. We observe that the mean relative L^2 error reduces when increasing the latent dimension of the autoencoder model. In other words, the model with a larger latent space is able to predict the evolution of the microstructure in forecasting mode. This is intuitive because a larger dimension of the latent space implies that there are more basis functions to express the encoded information about the microstructure and its evolution, and therefore the network has an improved representation capability. However, this trend seems to saturate beyond $l_d = 100$. For the model with $l_d = 100$ or $l_d = 196$, the forecasting error is always less than 6%. The logarithm of the relative L^2 error linearly increases for $l_d = 64{,}100$ and 196 for the forecasting time step, whereas for $l_d = 9$ and $l_d = 25$, the error is high and remains constant.

Robustness of the surrogate DeepONet framework: Sensitivity to noise

We evaluated the accuracy and robustness of the predictions from our surrogate model by systematically increasing the noise levels in the model input. For this analysis, we considered the best model with $l_d = 196$ and DeepONet architecture 1 (see Table 2)

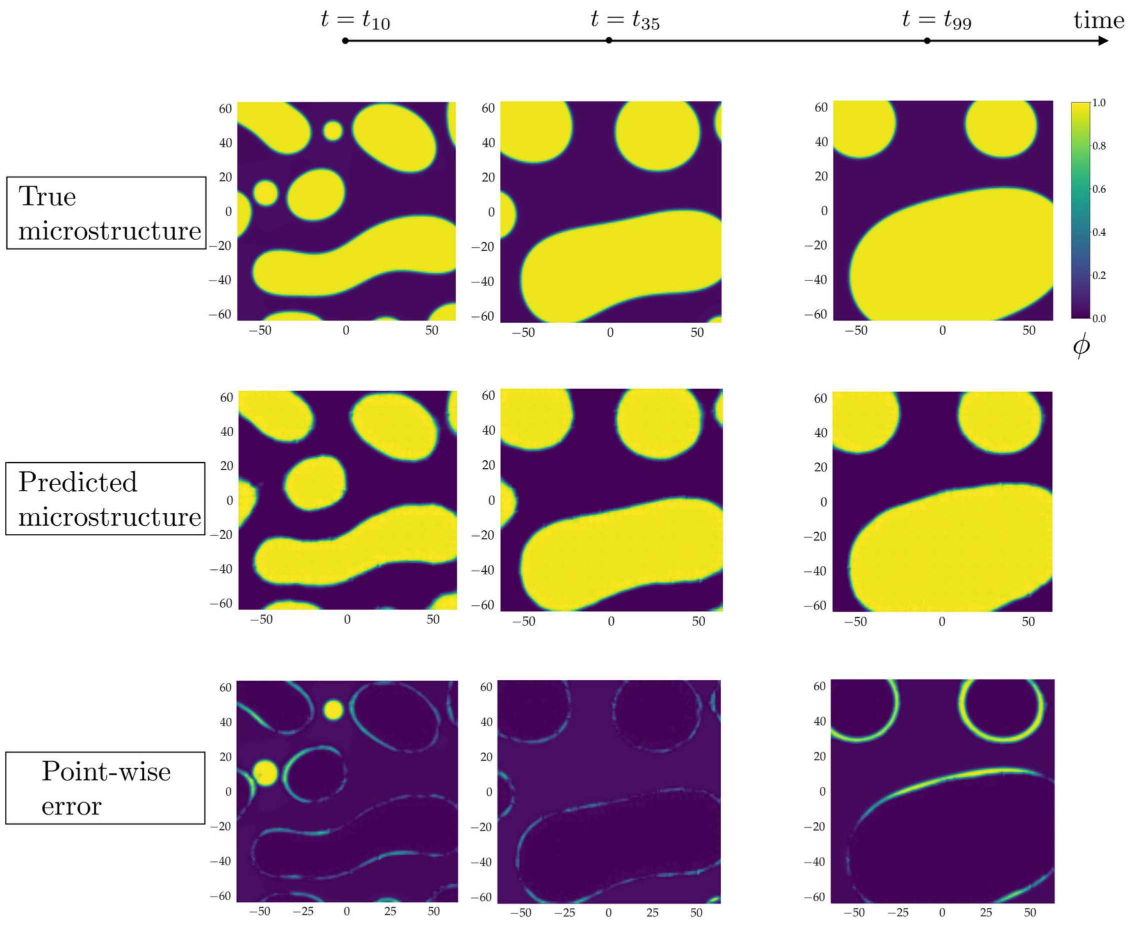


Fig. 2 Predictions of microstructure evolutions. The true (top row), predicted (middle row), and point-wise error (bottom row) for a microstructure realization evolving in time. The snapshots at time frames $t = t_{10}, t_{30}, t_{99}$ are shown here. The network used for this simulation has $l_d = 196$, and uses the following network architecture: Branch network -- $2 \times$ [conv(128, (3, 3))] + [3920]; Trunk network -- $2 \times$ [100] + [3920].

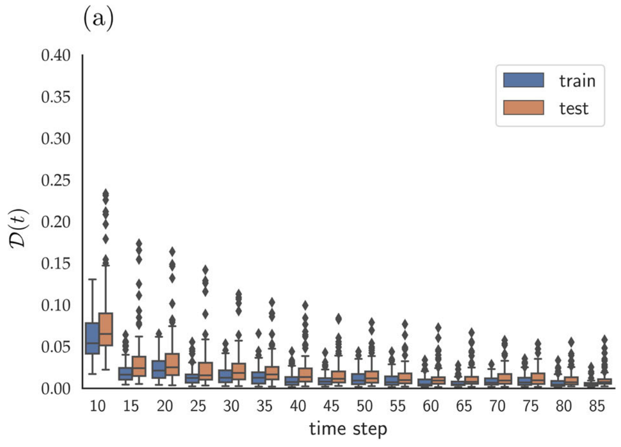
with sin activation functions. We added Gaussian white noise to our microstructure data with zero mean and standard deviations, $\sigma = 0.5\%$, 1%, 2%, 3%, 4%, 5%, 10%. To evaluate the model performance, we used the relative L^2 norm, $\mathscr{D}$, as defined in Eq. (8). The forecasting error, $\mathscr{D}$, is calculated across the samples present in the test dataset. Details can be found in Supplementary Note 5.

From Supplementary Table 1 and Supplementary Figure 5, the relative L^2 norm does not increase noticeably when noise is added to the model input. In fact, the surrogate is almost invariant to noise up to 10% Gaussian white noise, as presented in Supplementary Figure 5. Previous studies[45–47] illustrated the capability of autoencoders to denoise noisy images. The transformation to a low-dimensional latent space forces the autoencoder to retain the dominant features alone while discarding unnecessary noise. The convolutional autoencoder used in our approach does exactly that by denoising the noisy microstructure input. The encoder filters out noise and only retains the dominant energy modes of the microstructure data. The output of the convolutional encoder is almost in its pure form, free from noise and therefore it enables the DeepONet to make stable predictions. The decoder accurately reconstructs the microstructure from the predictions made by DeepONet in the latent space. This performance illustrates the robustness and efficacy of the present framework as compared to other machine-learned frameworks that use statistical functions to encode the microstructure representation[40]. Indeed, it has been illustrated by Herman and coworkers[40] and others[48] that, while statistical functions such as the microstructure auto-correlation functions are sufficient to capture the salient features of the microstructure in latent space, such representation does not uniquely map back to the true microstructure as it is an ill-posed, inverse problem. Here, our results essentially bypass such a challenge by taking advantage of the fact that autoencoders are robust to corruption in the representations they learn. The denoising nature of a trained autoencoder enables the encoder to learn a stable and consistent mapping to the latent space. This makes training of the DeepONet much more stable and results in accurate predictions of the microstructure at any desired time step.

Effect of time resolution

The high-fidelity phase-field forward numerical solver (MEMPHIS) discretizes the time with a time step $\Delta t = 1 \times 10^{-4}$ (see Methods for additional details). The stability of this numerical integration scheme can be achieved by strictly following the Courant-Friedrichs-Lewy (CFL) condition to solve the Cahn-Hilliard equation for 50M time steps. The solver saves snapshots of the solution at every 500,000th time step resulting in 100 microstructure time frames for each realization. We initially utilized 80 equally spaced time frames between the 10th and 90th time frames for training the surrogate model. Therefore, for the surrogate DeepONet model, each time step was $500000 \times \Delta t = 50$.

To investigate the effect of different spacing of physical time on the surrogate DeepONet model, we performed a sensitivity study using data that are spaced differently in time to train the model. Specifically, we trained DeepONet models on datasets with a time spacing of $500\text{k}\Delta t$, $2 \times 500\text{k}\Delta t$, $5 \times 500\text{k}\Delta t$, and $10 \times 500\text{k}\Delta t$. The



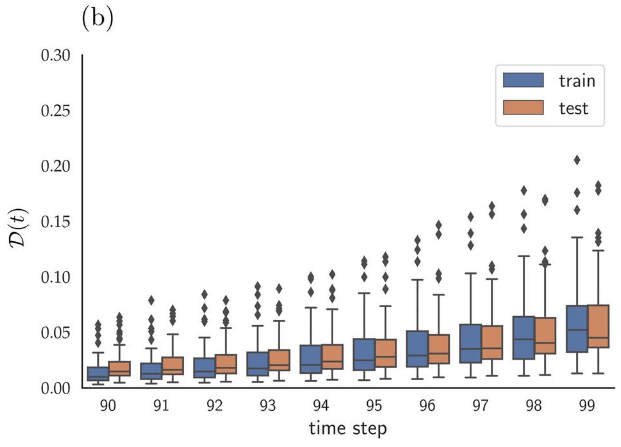


Fig. 3 L^2 norm of the relative error between true and predicted microstructures at each time step. a Box plot with respect to $\mathscr{D}(t)$ computed over the training time steps over the train and test datasets. Error bars are equivalent throughout the figure. **b** Same error metric, but in future time steps never seen during the training phase. Error bars are equivalent throughout the figure.

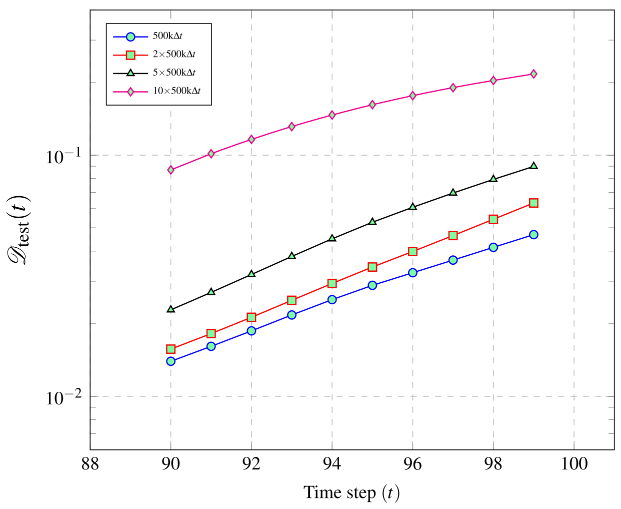


Fig. 4 Variation of forecasting error on test data for models trained on data with different spacing in time. Spacing in time tested are: $500k\Delta t$, $2\times 500k\Delta t$, $5\times 500k\Delta t$, $10\times 500k\Delta t$. $\mathscr{D}_{\text{test}}(t)$ represents the relative L^2 error computed across the samples in test dataset at different time steps.

DeepONet predictions were then remapped to the primitive space using the pre-trained convolutional decoder to recreate the microstructure at the required time step. We plot the mean relative L^2 forecasting error corresponding to the models trained on differently spaced datasets in Fig. 4. As expected, just like with any other time-integration scheme, we observe that the forecasting error increases for larger spacing in physical time. However, the computational efficiency of the DeepONet predictions remained the same for predicting consecutive time frames regardless of the time spacing used.

Training strategy for learning concurrently multi-scale features

We showed in a previous work[37,38] that the microstructure evolution in latent space is non-linear. Indeed, for early time steps the microstructure evolves rapidly and then later on it evolves more slowly once the phase separation dynamics have taken effect.

We noted that the DeepONet model architecture presented above was not able to resolve the small-scale microstructural features as shown in Fig. 2. In the early time steps, which represent fast dynamics, small wavelength features are hard to capture due to spectral bias of neural networks; Fig. 3 quantifies this difficulty.

To circumvent this issue, during training of the DeepONet model, we increased the weight given to earlier time steps of each realization in the dataset. By placing more emphasis on early snapshots during training, we endow DeepONet with an inductive bias to learn the fast dynamic accurately. Practically speaking, we are forcing the DeepONet model, $\mathscr{G}(\tilde{\mathbf{\Phi}})(t;\boldsymbol{\theta}_{\mathrm{d}})$, to predict earlier time steps repeatedly by creating a new training dataset with repeated $\tilde{\boldsymbol{\phi}}(t)$ for each realization. Since the DeepONet model is trained to minimize the mean squared error between the true and predicted microstructures, the model is driven to give greater emphasis to microstructures developing at earlier time steps. The results from this training procedure are depicted in Fig. 5. Indeed, from the comparison of the predicted microstructure without and with an emphasis on earlier time steps in Fig. 5b and c, respectively, we observe that increasing the weight given to the earlier time steps of evolution for each realization results in a DeepONet model capable of recovering smaller, high-frequency components. This ability to accurately resolve multiple length scales is particularly important in dynamic problems such as dendrite or grain growth problems, for instance, where the simulated microstructure dynamics can be extremely sensitive to the development of multiple length scales concurrently.

Integration of DeepONet with a numerical high-fidelity phase-field solver

The results above show that a pre-trained autoencoder–DeepONet model can be used as a robust and efficient surrogate of the numerical solver when inference is requested for initial microstructure and parameters within the distributions of the training datasets (interpolation task). Our proposed framework can also be used for extrapolation tasks and be integrated into the phase-field numerical solver to accelerate the predictions for initial microstructure and parameters that are outside the aforementioned distributions (extrapolation task). To demonstrate this point, we devised a hybrid approach that integrates the autoencoder–DeepONet framework with our high-fidelity phase-field Mesoscale Multiphysics Phase Field Simulator (MEMPHIS solver). This hybrid model unites the efficiency and computational speed of the autoencoder–DeepONet framework with the accuracy of high-fidelity phase-field numerical solvers.

The hybrid framework consists of alternating between predictions from the high-fidelity phase-field simulations and those from the autoencoder–DeepONet model. The high-fidelity phase-field simulation step provides accuracy in the description of the dynamics, while the autoencoder–DeepONet model enables us to

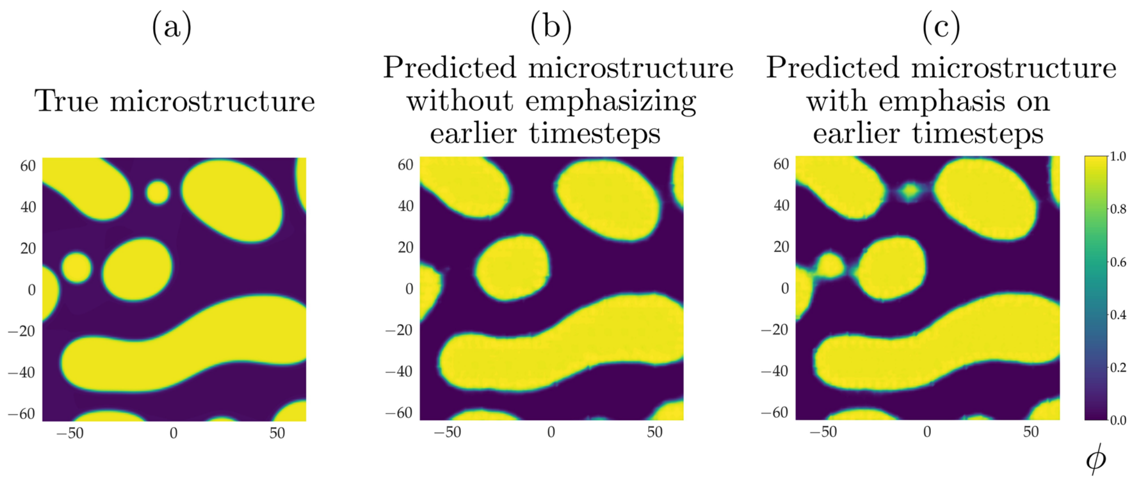


Fig. 5 Capturing small microstructural features at earlier time steps. **a** True microstructure at $t = t_{10}$. **b** Predicted microstructure without emphasizing earlier time steps during training. **c** Microstructure predicted by DeepONet trained on a dataset where the earlier time steps where repeated to increase the importance given to earlier time steps.

'leap in time'. The algorithm is presented in Algorithm 1. Here we choose to split the time evolution predicted between the high-fidelity simulation and those of the autoencoder–DeepONet to be equal to one another. Each solver within this integrated scheme sequentially predicts 10-time frames, which corresponds to $10 \times 500k = 5M$ time steps for the high-fidelity phase-field solver alone. A schematic of the approach and results are shown in Fig. 6 for the training data. Results for the test data are provided in Supplementary Fig. 6. The discontinuity along the centerlines of the predictions made by the autoencoder–DeepONet model arises from splitting each realization into four different realizations, as discussed in the 'Microstructure-evolution dataset' subsection.

Algorithm 1. Integration of DeepONet with high-fidelity phase field simulator (MEMPHIS)

```
Require: $\phi_0$ : Initial condition: $\phi(x, y, 0)$
Require: $N_T$ : Number of total time steps
Require: $n_t$ : Initial number of time steps to be simulated by MEMPHIS
Require: $\text{DON}_{nt}$ : Number of time steps to be leaped by DeepONet
  $n \leftarrow 0$                                              ▷ Initialize
While $n \, != N_T$ do
    $\phi_{n_t} \leftarrow \text{MEMPHIS}(\phi_0, n_t)$             ▷ Solution from MEMPHIS
    $\phi_{n_t+\text{DON}_{nt}} \leftarrow \text{DeepONet}(\phi_{n_t})$   ▷ Prediction from DeepONet
    $\phi_0 \leftarrow \phi_{n_t+\text{DON}_{nt}}$                  ▷ Update the input for MEMPHIS
    $n \leftarrow n_t + \text{DON}_{nt}$          ▷ Leaping $n$: MEMPHIS + DeepONet
end while
```

We see in Fig. 6 that the forecasting from 10 time frames using the high-fidelity phase-field solver MEMPHIS running on 32 CPU-cores (Intel® Xeon®, e5-2670) takes approximately 90 min. The subsequent 10 time frames predicted by the autoencoder–DeepONet model take only 2 s. A comparison of the computational cost between the high-fidelity phase-field solver alone and the hybrid approach is reported in Table 3. Here, we achieve a speed-up of 29% . This performance can be improved by a much greater factor with more extensive offline training with a richer dataset of operating conditions, which will lead to better generalization. For each evolution, our hybrid approach saves 135 min, without loss of accuracy as shown previously in the Results section. The choice of a specific time step splitting, for which the system is evolved using the high-fidelity phase-field framework and then by the autoencoder–DeepONet model, is arbitrary and can be considered as a hyper-parameter. For instance, one could easily consider to use very short time steps within the high-fidelity phase-field solver window to course-correct the physical predictions and much longer time steps when using DeepONet to accelerate the time evolution predictions. This type of time splitting integration scheme would dramatically increase the speedup even further. Additionally, although we showed the microstructure evolution only until $t = t_{105}$, such a hybrid time integration strategy can be adopted to forecast time evolution of the microstructure for time windows that can be much longer, for instance as long as the input time history used to train the DeepONet model while still keeping a good accuracy, leading to substantial savings in CPU hours.

DISCUSSION

In this work, we investigated the effectiveness of a convolutional autoencoder–DeepONet approach for modeling the evolution dynamics of mesoscale microstructures. The proposed framework consists of two parts. First, learning a non-linear mapping to a latent manifold using convolutional autoencoders, and second, learning the dynamics in the latent space (from the first step) using DeepONet. We trained our model on high-fidelity, phase-field data generated by solving the Cahn-Hilliard equation. The results presented above show that the trained DeepOnet architecture can be used robustly to replace the high-fidelity phase-field numerical solver in interpolation tasks or to speed up the numerical solver for extrapolation tasks. We showed that increasing the latent dimension used to describe the micro-structure evolution and putting more emphasis on earlier time steps during the training improve the overall representation capability of the framework. Given its performance, this framework offers several advantages as compared to other machine-learned architectures used for accelerating the prediction of the phase-field-based microstructure evolution.

First, unlike existing methods[37,40] that train machine-learning-based surrogate models using low-dimensional representations of microstructures based on statistical functions (e.g. auto-correlation function), our autoencoder–DeepONet approach learns a suitable low-dimensional latent space using a convolutional autoencoder. We showed that this approach bypasses any post-processing steps (e.g. a phase-recovery algorithm) necessary to reconstruct the microstructure from statistical functions[40]. The advantage of training DeepONet in the autoencoder latent space is two-fold. On one hand, training DeepONet in a low-dimensional space is computationally efficient. On the other hand, the presence of

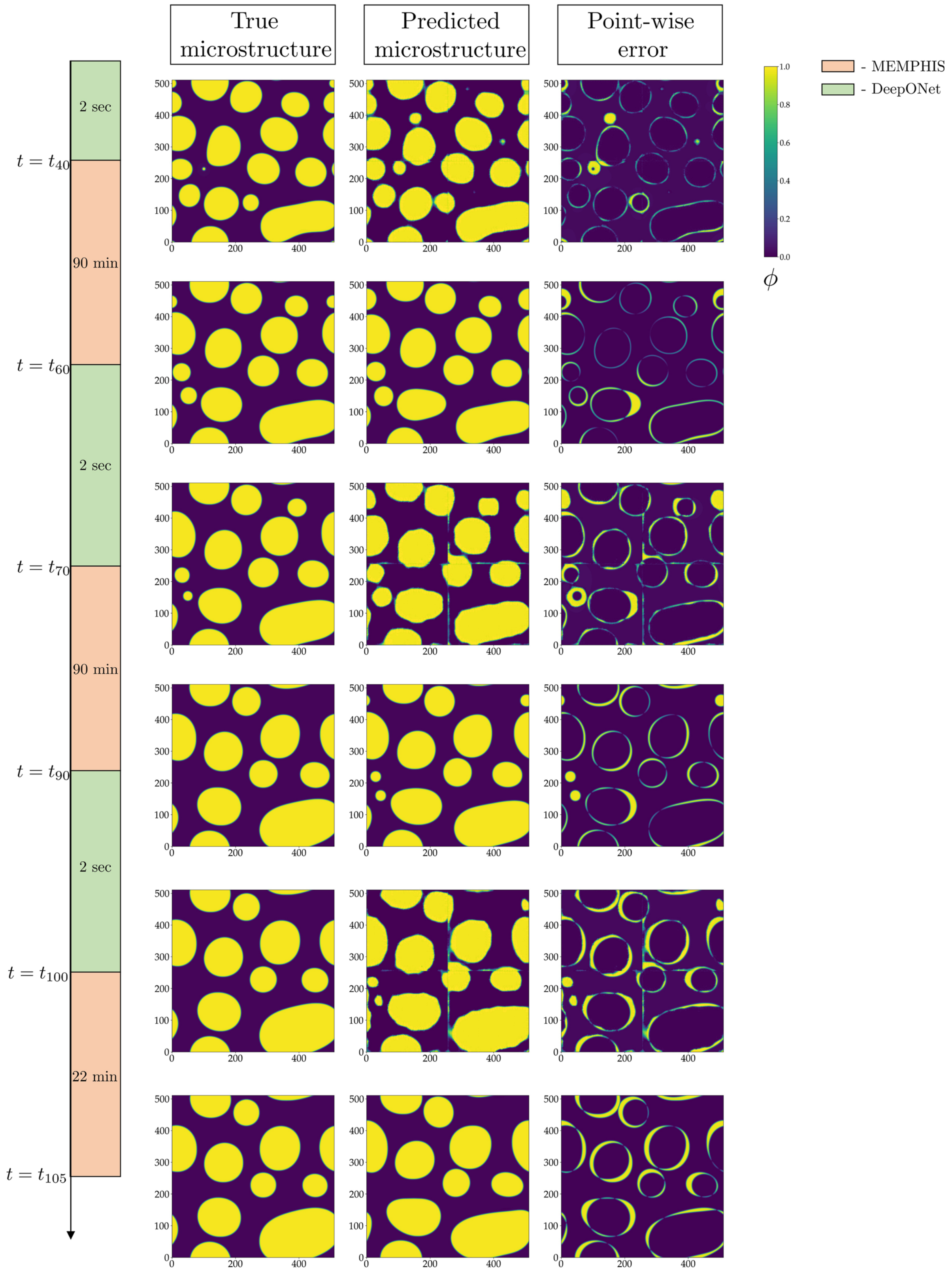


Fig. 6 Schematic of our hybrid approach for integrating DeepONet with the numerical solver MEMPHIS to accelerate phase-field predictions. The computational time corresponding to the autoencoder--DeepONet model and the MEMPHIS solver for one realization in the training dataset is reported in this figure. The error is shown on the third column.

high-gradient regions in the microstructure data (see Fig. 7a) can make the training of the model challenging. However, the encoder transforms microstructure data to a latent space (Fig. 7b), where the gradients are not as high and more gradual . In other words, the encoder learns a non-linear mapping of the microstructure data coming from an untrainable distribution in the primitive space, as shown in Fig. 7c, to a trainable distribution in the low-dimensional latent space, as shown in Fig. 7d. Such a transformation from data with a high gradient to data with gradual, smoother gradient facilitates the training of the DeepONet model. Although we have trained our own autoencoder model in this work, we believe that fine-tuning any autoencoder pre-trained on existing image datasets with similarities to our microstructure images will be suitable for this task. Reusing such readily available pre-trained autoencoders can further save on the computational cost of our workflow.

Table 3. A comparison between computational time for high-fidelity phase-field simulations (MEMPHIS) and proposed hybrid model (Hybrid) for a single microstructure evolution realization from time frame t_1 to time frame t_{105}.

	Computational time	
Time frames	MEMPHIS	Hybrid
t_1 to t_{30}	135 mins	135 mins
t_{30} to t_{40}	45 mins	2 sec
t_{40} to t_{60}	90 mins	90 mins
t_{60} to t_{70}	45 mins	2 sec
t_{70} to t_{90}	90 mins	90 mins
t_{90} to t_{100}	45 mins	2 sec
t_{100} to t_{105}	23 mins	23 mins
Total time	473 mins	338.1 mins

Second, even though the workflow presented in this work is focused on two-dimensional (2D) microstructure data, it can easily be extended to three-dimensional (3D) microstructure data. For the 3D microstructure evolution case, each realization can be represented by a sequence of 3D tensor data structures. Hence, we could use an autoencoder with 3D convolution layers in the encoder and 3D transpose convolution layers in the decoder and learn a suitable non-linear mapping to a low-dimensional latent manifold. Following the same approach, a DeepONet model can be trained to learn the dynamics in the latent space, and be then remapped to primitive space using the already trained decoder. As shown in a recent theoretical paper, DeepOnet can tackle the curse of dimensionality in the input space, so training it in highdimensions is not a prohibitive issue[49].

Third, the autoencoder and DeepONet are trained solely from data, making the proposed approach purely data-driven, independent of the boundary conditions. The boundary conditions are never explicitly assumed as input data to the framework at any stage. They are implicitly fed through the latent representation of the microstructure history data inputted to the branch network of DeepONet. Therefore, any information regarding a change in the boundary condition will be available in the latent microstructure history fed to the branch network, enabling the DeepONet model to predict the dynamics accordingly in the latent manifold. In this manner, the purely data-driven nature makes the proposed autoencoder–DeepONet framework agnostic to any changes in boundary conditions within the considered history. We also need to clarify that periodic boundary conditions were imposed at all four boundaries of the computational domain while generating the data from the numerical solver MEMPHIS. We note that DeepONet can be trained to map boundary conditions to an output field if so desired for a very general set of variable boundary conditions.

There are several extensions to the present framework that can be implemented in order to improve the accuracy, predictability, and acceleration performance. These improvements are related to

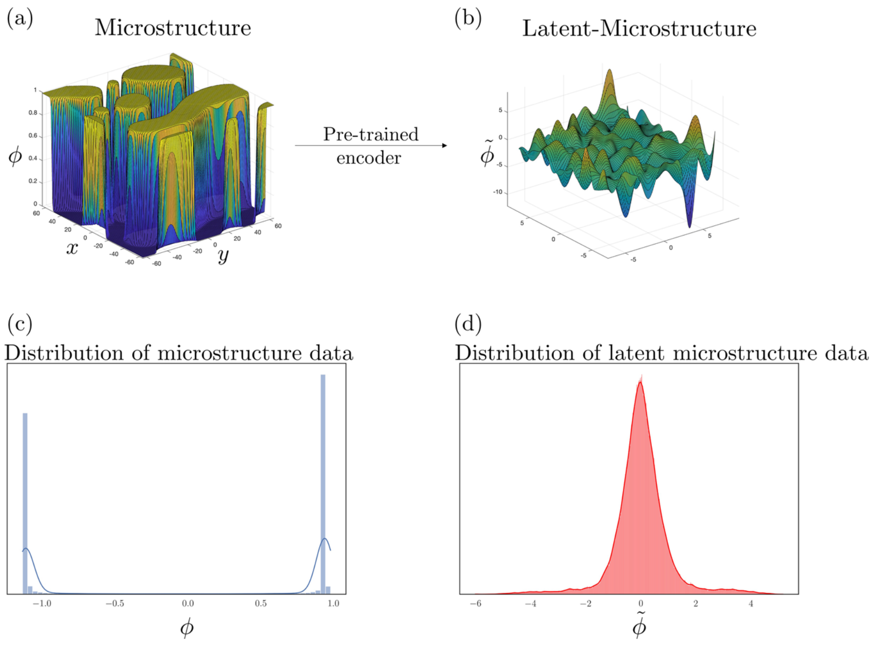


Fig. 7 Representation of microstructure data and statistical insights. **a** represents a 3D visualization of the function to be approximated by the surrogate model. The presence of several high-gradient regions at every time-step, makes it challenging for neural network models to learn the evolution dynamics of microstructures. Panel (**b**) represents a smoother latent-microstructure learned by the encoder during the autoencoder training. **c** The microstructure data, $\phi(x, y, t)$, is predominantly represented by 1s or 0s. **d** The encoder transforms ϕ to a latent space, $\tilde{\phi}$, where deep neural networks can learn easily. The curves in (**c**) and (**d**) represent the smoothed density estimates of the histogram.

the training of the model and extension to a multi-fidelity implementation. The first topic is related to improving the accuracy of the model with physics constraints in order to better capture non-linearities in the model evolution. The second topic is related to fusing different sources of data within our dataset, e.g., 'low' fidelity from simulation and 'high' fidelity from physical experiments of the same nature.

Regarding a physics-informed implementation, Wang et al.[50] for instance put forward a physics-informed DeepONet, where the PDE of the underlying system is added as a soft constraint to the loss functions. In the present study, the training framework is purely data-driven and we are learning the dynamical system in the latent space defined by non-primitive coordinates except for time, which is fed as an input to the trunk net. However, similar to Wang et al., some physical constraints could be injected into the current framework by using, for instance, the fact that mass ϕ is conserved at all times.

Regarding the multi-fidelity implementation, the present approach can be extended to incorporate experimental data coming from similar processes. For instance, recently, several researchers[51–53] explored diverse ways of exploiting the inherent correlations between datasets coming from different sources of data with different levels of fidelity and obtained optimal predictions. In this context, the data obtained from phase-field models using a numerical solver can be considered as a low-fidelity dataset and the limited amounts of experimental microstructure image data from similar processes can be treated as a high-fidelity dataset. The autoencoder–DeepONet framework proposed here can be extended to generate accurate predictions from a limited number of high-fidelity experimental microstructure microscopy image data, by utilizing the high correlation with the surplus low-fidelity phase field data. The assimilation of experimental data in the present DeepONet architecture can be concatenated with numerical data as another realization and the proposed workflow will remain unchanged. Merging and taking advantage of both experimental and modeling efforts is a future direction of our research.

To summarize, we developed and applied a machine-learned framework based on neural operators and autoencoder architectures to efficiently and rapidly predict complex microstructural evolution problems. Such an architecture is not only computationally efficient and accurate, but it is also robust to noisy data. The demonstrated performance makes it an attractive alternative to other existing machined-learned strategies to accelerate the predictions of microstructure evolution. It opens up a computationally viable and efficient path forward for discovering, understanding, and predicting materials processes, where evolutionary mesoscale phenomena are critical, such as in the optimization and design of materials problems.

METHODS

Phase-field model of the spinodal decomposition of a two-phase mixture

We illustrate our accelerated phase-field workflow on the simplest case of the spinodal decomposition of a two-phase mixture. This model is highly relevant to many phase-field models. In the spinodal decomposition of a two-phase mixture uses a single order parameter, $\phi(\mathbf{x}, t)$ to describe the atomic fraction of solute diffusing within a matrix. The free energy of the system is expressed by the Cahn-Hilliard equation based on the Onsager force-flux relationship such that

$$\frac{\partial\phi}{\partial t} = \nabla\cdot\left(M_c(\phi)\nabla[\omega_c(\phi^3-\phi)+\kappa_c\nabla^2\phi]\right), \tag{3}$$

where ω_c is the height of the energy barrier between the two phases, κ_c is the gradient energy coefficient, and M_c denotes the concentration dependent mobility, with $M_c = s(\phi)M_A + (1-s(\phi))M_B$. The function s defines a smooth interpolation to switch from phase 'A' to phase 'B'. This interpolation function is defined as $s(\phi)=\frac{1}{4}(2-\phi)(1+\phi)^2$. In the present model, both the mobility and the interfacial energy are taken to be isotropic and ω_c and κ_c are stet to unity for simplicity. The evolution of one phase is expressed as a symmetric double-well potential, with minima at $\phi\pm 1$.

Microstructure-evolution dataset

The phase-field model described above is implemented using Sandia's in-house multi-physics phase-field modeling code MEMPHIS[10,54]. In order to generate a diverse and large set of simulation results exhibiting a rich variety of microstructure features, we independently sampled the phase fraction ϕ_A, such that each phase has at least a minimum concentration of 0.15 (note that $\phi_B = 1-\phi_A$), and the phase mobilities M_A and M_B of species 'A' and 'B'. Phase mobilities are sampled independently to vary in the range [0.01, 100]. In total we generated 500 triplets (ϕ_A, M_A, M_B) using Latin Hypercube Sampling. In the simple case of the spinodal decomposition, only the tuple (ϕ_A, M_A/M_B) is necessary. As demonstrated in other studies[10,40] that share similar microstructure evolution as the spinodal decomposition, it is, however, necessary to handle (ϕ_A, M_A, M_B) separately since the ratio M_A/M_B by itself will not be sufficient anymore to characterize the dynamics of the microstructure evolution. Herein, we frame the present work in a broader context for generality. All the simulations were performed using a two-dimensional (2D) square domain $\Omega = [0, 1]\times[0, 1]$, discretized with 512×512 grid points, with a dimensionless spatial discretization of unity in either direction, and a temporal discretization of $\Delta t = 1\times10^{-4}$. The simulation domain's composition field is initialized using truncated random Gaussian distribution in the range $[-1, 1]$ with $\mu=\phi_A$, and $\sigma = 0.35$. The microstructure was allowed to evolve and grow for 50,000,000 time steps, saving the state of the microstructural domain every 500,000 time steps, hence a total of 100-time frames were saved from each simulated case.

In order to use the data in the proposed algorithm, we down-sampled each snapshot of our 512×512 domain into four images of 256×256, and later used cubic interpolation[55] to further reduce the resolution to 128×128. Hence, from the 500 microstructure evolution samples, we were able to generate 2000 microstructure evolution samples of 128×128 resolution. From this dataset, we have used 1600 cases for training the DeepONet and 400 cases for testing the network accuracy. Since the compositional field is randomly distributed spatially, the microstructure has no recognizable features at the first frame t_0. The quick development of subdomains is then observed between frames t_0 and t_{10}, followed by a smooth and steady coalescence and growth of the microstructure from time frames t_{10} to t_{100}. We have trained our proposed model based on this observation, starting at time frame, t_{10}, when the microstructure had reached a slow and steady development regime.

Training the autoencoder: learning the latent microstructure representation

In this work, each microstructure evolution is represented by $(N_T, N_x, N_y) = (80\times128\times128)$, with N_T representing the number of snapshots and $N_x\times N_y$ denoting the spatial resolution along $x-$ and $y-$direction, respectively. To handle the entire feature space ($\mathbb{R}^{128\times128}$) 16,384 distinct features are required to represent the microstructure at each time step. Subsequently, to compute the prediction for all 1,600 microstructure evolutions, we will have $1600\times80\times16,384$ (≈2.5 Billion) 32-bit floating data points. Learning microstructure dynamics from such a high-dimensional dataset is challenging.

To circumvent issues pertaining to the data dimensionality and preparing the phase-field microstructure data for DeepONet training, we explored a couple of options. First, we tried using Principal Component Analysis (PCA) with a linear kernel, for reducing the dimensionality of the data[21,56,57]. The low-dimensional representation of the data obtained from PCA is a linear transformation of the high-dimensional data and discards the insignificant modes (eigen/singular) corresponding to the lower eigen/singular values (λ_i). However, the system considered here is non-diffusive, which is confirmed by cumulative explained variance and energy distribution of the system over principal modes. Therefore, using PCA for reducing the dimensionality of the microstructure description could result in the loss of valuable information, if only a convenient low number of principal components are considered. A detailed explanation on PCA of the microstructure dataset is presented in Supplementary Note 1. Learning a non-linear mapping from a high-dimensional to a low-dimensional latent space is one way to compress data without losing as much information as in the PCA. An autoencoder precisely does this by learning a non-linear transformation to a low-dimensional latent space using an encoder. The decoder learns the mapping to retrieve initial high-dimensional data from its latent representation.

In this study, we have used a convolutional autoencoder[58] with convolutional layers in the encoder and transpose convolutional layers in the decoder as shown in Fig. 1. The encoder learns a nonlinear mapping of the high-dimensional microstructure data, $\phi(\boldsymbol{x}, \boldsymbol{y}, t)$, to a low-dimensional latent space represented by $\tilde{\boldsymbol{\phi}}(t)$ and is expressed as

$$\alpha_{\boldsymbol{\theta}_{\text{enc}}} : \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}, t) \rightarrow \tilde{\boldsymbol{\phi}}(t), \tag{4}$$

$$\beta_{\boldsymbol{\theta}_{\text{dec}}} : \tilde{\boldsymbol{\phi}}(t) \rightarrow \hat{\boldsymbol{\phi}}(\boldsymbol{x}, \boldsymbol{y}, t), \tag{5}$$

where α and β represent the mappings performed by the encoder and the decoder, respectively. In Eq. 4, the encoder takes $\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}, t) \in \mathbb{R}^{128\times 128}$ as input, and maps it to $\tilde{\boldsymbol{\phi}}(t) \in \mathbb{R}^{l_d}$, where l_d is the dimension of the latent space. $\boldsymbol{\theta}_{\text{enc}}$ represents the trainable parameters of the convolutional encoder. Equation 5 represents the decoder network, which takes the latent dimensional representation, $\tilde{\boldsymbol{\phi}}(t) \in \mathbb{R}^{l_d}$ as the input and predicts the primitive microstructure, $\hat{\boldsymbol{\phi}}(\boldsymbol{x}, \boldsymbol{y}, t) \in \mathbb{R}^{128\times 128}$, using transpose convolutional operations. The details of the autoencoder architecture are provided in Table 1.

$\boldsymbol{\theta}_{\text{ae}}$ represents the trainable parameters of the autoencoder. These parameters are learned by minimizing the loss function, $\mathscr{L}_{\text{ae}}$, which reads

$$\mathscr{L}_{\text{ae}} = \min_{\boldsymbol{\theta}_{\text{ae}}=\{\boldsymbol{\theta}_{\text{enc}}, \boldsymbol{\theta}_{\text{dec}}\}} \|\boldsymbol{\phi}(x, y, t) - \hat{\boldsymbol{\phi}}(x, y, t; \boldsymbol{\theta}_{\text{ae}})\|_2^2. \tag{6}$$

Alternatively, the autoencoder provides a low-dimensional representation of the microstructure by learning a non-linear transformation to a latent space with l_d features. We also observe that it is easier to learn the microstructure dynamics in the latent space representation, learned by the autoencoder, than the original primitive form of the microstructure in real space. This is due to the presence of several high gradient regions in the original form of the microstructure as shown in Fig. 7a. These high gradient regions in the solution are due to the nature of the governing Cahn-Hilliard equation. The latent microstructure representation in Fig. 7b is smoother and does not have high gradient regions. The latent representation of the data offers higher regularity and therefore, we achieve faster convergence during the training of the surrogate neural network model.

Training the DeepONet: learning the microstructure dynamics in lower dimensions

Neural operators generate nonlinear mappings across infinite-dimensional function spaces on bounded domains, giving a simulation framework for multidimensional complex dynamics prediction in real time. Once properly trained, such models are discretization invariant, which means they share the same network parameters regardless of how the underlying functional data is parameterized. DeepONet, originally proposed by Lu and coworkers[41], allows the mapping between infinite-dimensional functions using deep neural networks. This subsection provides a detailed description of the training of DeepONet to model the evolution of the microstructure in the latent dimension.

The unstacked DeepONet architecture is made up of two concurrent deep neural networks: one encodes the input function at fixed sensor locations (branch network), while the other represents the domain of the output function (trunk network). Time, $t \in \mathbb{R}^1$, is given as input to the trunk network while $\tilde{\boldsymbol{\Phi}} = \{\tilde{\phi}(t_{10}), \tilde{\phi}(t_{11}), \ldots, \tilde{\phi}(t_{89})\} \in \mathbb{R}^{80\times l_d}$ is the input fed to the branch network. $\tilde{\boldsymbol{\Phi}}$ represents the phase field in the latent dimension, l_d, for all the 80-time steps available in the given dataset. The goal of the DeepONet is to learn the solution operator, $\tilde{\phi}(t) \approx \tilde{\phi}(t) = \mathscr{G}(\tilde{\boldsymbol{\Phi}})(t)$ from the 1600 microstructure evolutions provided in the training dataset. The output of the DeepONet is a vector $\in \mathbb{R}^{l_d}$ and is expressed as $\mathscr{G}(\tilde{\boldsymbol{\Phi}})(t; \boldsymbol{\theta}_d)$, where $\boldsymbol{\theta}_d = \{\mathbf{W}_d, \mathbf{b}_d\}$ includes the trainable weights, $\mathbf{W}_d$, and biases, $\mathbf{b}_d$, of the DeepONet model. The framework of the DeepONet allows the branch network to have a flexible architecture. To model the microstructure evolution, we have considered a fully connected neural network for the trunk network. Due to the high-dimensional nature of the branch network input, $\mathbb{R}^{80\times l_d}$, a convolutional neural network is used as the branch network because it utilizes the same kernels across the time axis and enables the branch network to encode the entire history in a memory efficient manner. Hence, the input has to be reshaped to $\mathbb{R}^{80\times \sqrt{l_d}\times\sqrt{l_d}}$ before feeding it to the branch network. The network architecture is presented in Fig. 1. The trainable parameters of the DeepONet, $\boldsymbol{\theta}_d$, are obtained by minimizing a loss function, $\updownarrow_d$, defined as:

$$\mathscr{L}_d = \min_{\theta_d} \left|\left|\tilde{\boldsymbol{\phi}}(t) - \mathscr{G}(\tilde{\boldsymbol{\Phi}})(t; \boldsymbol{\theta}_d)\right|\right|_2^2, \tag{7}$$

where $\tilde{\phi}(t)$ is the ground truth for the low-dimensional phase field representation at time, t obtained from the convolutional encoder. The trained DeepONet is used to predict $\tilde{\phi}(t) \in \mathbb{R}^{l_d}$. The output of the DeepONet is fed into the transposed convolutional decoder to predict, $\hat{\phi}(t) \in \mathbb{R}^{128\times 128}$. The DeepONet is trained using the Adam optimizer[59]. The implementation has been carried out using the `TensorFlow` framework[60]. We use Xavier Initialization[61] to initialize the weights of all the models.

Error metrics

The L^2 norm of relative error, $\mathscr{D}$, is used as the evaluation metric to analyze the performance of each model considered in this study. $\mathscr{D}$ is defined as:

$$\mathscr{D} = \frac{\sum_n \sum_x \sum_y \sum_t \left(\phi^{(n)}(x, y, t) - \hat{\phi}^{(n)}(x, y, t; \boldsymbol{\theta})\right)^2}{\sum_n \sum_x \sum_y \sum_t \phi^{(n)}(x, y, t)^2}, \tag{8}$$

where n corresponds to the nth sample of the given dataset.

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding authors upon reasonable request.

CODE AVAILABILITY

The codes developed (in Python3) to perform the computational experiments in the present study are freely available at this git repository.

Received: 20 April 2022; Accepted: 17 August 2022;
Published online: 02 September 2022

REFERENCES

1. Chen, L.-Q. Phase-field models for microstructure evolution. *Annu. Rev. Mater. Res.* **32**, 113–140 (2002).
2. Moelans, N., Blanpain, B. & Wollants, P. An introduction to phase-field modeling of microstructure evolution. *Calphad* **32**, 268–294 (2008).
3. Yang, Z., Yu, C.-H. & Buehler, M. J. Deep learning model to predict complex stress and strain fields in hierarchical composites. *Sci. Adv.* **7**, eabd7416 (2021).
4. Pokharel, R., Pandey, A. & Scheinker, A. Physics-informed data-driven surrogate modeling for full-field 3D microstructure and micromechanical field evolution of polycrystalline materials. *JOM* **73**, 3371–3382 (2021).
5. Chakraborty, S., Goswami, S. & Rabczuk, T. A surrogate assisted adaptive framework for robust topology optimization. *Computer Methods Appl. Mech. Eng.* **346**, 63–84 (2019).
6. Olsson, E. & Kreiss, G. A conservative level set method for two phase flow. *J. Computational Phys.* **210**, 225–246 (2005).
7. Yue, P., Zhou, C., Feng, J. J., Ollivier-Gooch, C. F. & Hu, H. H. Phase-field simulations of interfacial dynamics in viscoelastic fluids using finite elements with adaptive meshing. *J. Computational Phys.* **219**, 47–67 (2006).
8. Bharali, R., Goswami, S., Anitescu, C. & Rabczuk, T. A robust monolithic solver for phase-field fracture integrated with fracture energy based arc-length method and under-relaxation. *Computer Methods Appl. Mech. Eng.* **394**, 114927 (2022).
9. Goswami, S., Anitescu, C. & Rabczuk, T. Adaptive fourth-order phase field analysis for brittle fracture. *Computer Methods Appl. Mech. Eng.* **361**, 112808 (2020).
10. Stewart, J. A. & Dingreville, R. Microstructure morphology and concentration modulation of nanocomposite thin-films during simulated physical vapor deposition. *Acta Materialia* **188**, 181–191 (2020).
11. Powers, M., Stewart, J. A., Dingreville, R., Derby, B. K. & Misra, A. Compositionally-driven formation mechanism of hierarchical morphologies in co-deposited immiscible alloy thin films. *Nanomaterials* **11**, 2635 (2021).
12. Beyerlein, I. & Hunter, A. Understanding dislocation mechanics at the mesoscale using phase field dislocation dynamics. *Philos. Trans. R. Soc. A: Math., Phys. Eng. Sci.* **374**, 20150166 (2016).
13. Elliott, C. M. & French, D. A. Numerical studies of the Cahn-Hilliard equation for phase separation. *IMA J. Appl. Math.* **38**, 97–128 (1987).
14. Cahn, J. W. & Hilliard, J. E. Free energy of a nonuniform system. I. Interfacial free energy. *J. Chem. Phys.* **28**, 258–267 (1958).

15. Sun, Z. Z. A second-order accurate linearized difference scheme for the two-dimensional Cahn-Hilliard equation. *Math. Comput.* **64**, 1463–1471 (1995).
16. Liu, C. & Shen, J. A phase field model for the mixture of two incompressible fluids and its approximation by a Fourier-spectral method. *Phys. D: Nonlinear Phenom.* **179**, 211–228 (2003).
17. Barrett, J. W., Blowey, J. F. & Garcke, H. Finite element approximation of the Cahn-Hilliard equation with degenerate mobility. *SIAM J. Numer. Anal.* **37**, 286–318 (1999).
18. Gómez, H., Calo, V. M., Bazilevs, Y. & Hughes, T. J. Isogeometric analysis of the Cahn-Hilliard phase-field model. *Computer Methods Appl. Mech. Eng.* **197**, 4333–4352 (2008).
19. Chan, C. L., Anitescu, C. & Rabczuk, T. Strong multipatch C1-coupling for isogeometric analysis on 2D and 3D domains. *Computer Methods Appl. Mech. Eng.* **357**, 112599 (2019).
20. Alikakos, N. D., Bates, P. W. & Chen, X. Convergence of the Cahn-Hilliard equation to the Hele-Shaw model. *Arch. Ration. Mech. Anal.* **128**, 165–205 (1994).
21. Brough, D. B., Kannan, A., Haaland, B., Bucknall, D. G. & Kalidindi, S. R. Extraction of process-structure evolution linkages from X-ray scattering measurements using dimensionality reduction and time series analysis. *Integrating Mater. Manuf. Innov.* **6**, 147–159 (2017).
22. Pfeifer, S., Wodo, O. & Ganapathysubramanian, B. An optimization approach to identify processing pathways for achieving tailored thin film morphologies. *Computational Mater. Sci.* **143**, 486–496 (2018).
23. Teichert, G. H. & Garikipati, K. Machine learning materials physics: Surrogate optimization and multi-fidelity algorithms predict precipitate morphology in an alternative to phase field dynamics. *Computer Methods Appl. Mech. Eng.* **344**, 666–693 (2019).
24. Lin, C., Maxey, M., Li, Z. & Karniadakis, G. E. A seamless multiscale operator neural network for inferring bubble dynamics. *J. of Fluid Mechanics.* **929** (2021).
25. Zhang, X. & Garikipati, K. Machine learning materials physics: Multi-resolution neural networks learn the free energy and nonlinear elastic response of evolving microstructures. *Computer Methods Appl. Mech. Eng.* **372**, 113362 (2020).
26. Goswami, S., Yin, M., Yu, Y. & Karniadakis, G. E. A physics-informed variational deeponet for predicting crack path in quasi-brittle materials. *Computer Methods Appl. Mech. Eng.* **391**, 114587 (2022).
27. Li, Z. et al. Fourier neural operator for parametric partial differential equations. Preprint at: https://arxiv.org/abs/2010.08895 (2020).
28. Goswami, S., Anitescu, C., Chakraborty, S. & Rabczuk, T. Transfer learning enhanced physics informed neural network for phase-field modeling of fracture. *Theor. Appl. Fract. Mech.* **106**, 102447 (2020).
29. Kunselman, C., Attari, V., McClenny, L., Braga-Neto, U. & Arroyave, R. Semi-supervised learning approaches to class assignment in ambiguous microstructures. *Acta Materialia* **188**, 49–62 (2020).
30. Haghighat, E., Raissi, M., Moure, A., Gomez, H. & Juanes, R. A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics. *Computer Methods Appl. Mech. Eng.* **379**, 113741 (2021).
31. Zhang, K. et al. High-throughput phase-field simulations and machine learning of resistive switching in resistive random-access memory. *npj Computational Mater.* **6**, 1–10 (2020).
32. Goswami, S., Anitescu, C. & Rabczuk, T. Adaptive fourth-order phase field analysis using deep energy minimization. *Theor. Appl. Fract. Mech.* **107**, 102527 (2020).
33. Attari, V. & Arroyave, R. Machine learning-assisted high-throughput exploration of interface energy space in multi-phase-field model with calphad potential. *Mater. Theory* **6**, 1–20 (2022).
34. Samaniego, E. et al. An energy approach to the solution of partial differential equations in computational mechanics via machine learning: Concepts, implementation and applications. *Computer Methods Appl. Mech. Eng.* **362**, 112790 (2020).
35. Shukla, K., Jagtap, A. D., Blackshire, J. L., Sparkman, D. & Em Karniadakis, G. A physics-informed neural network for quantifying the microstructural properties of polycrystalline nickel using ultrasound data: A promising approach for solving inverse problems. *IEEE Signal Process. Mag.* **39**, 68–77 (2022).
36. Perera, R., Guzzetti, D. & Agrawal, V. Graph neural networks for simulating crack coalescence and propagation in brittle materials. *Computer Methods Appl. Mech. Eng.* **395**, 115021 (2022).
37. Montes de Oca Zapiain, D., Stewart, J. A. & Dingreville, R. Accelerating phase-field-based microstructure evolution predictions via surrogate models trained by machine learning methods. *npj Computational Mater.* **7**, 1–11 (2021).
38. Hu, C., Martin, S. & Dingreville, R. Accelerating phase-field predictions via recurrent neural networks learning the microstructure evolution in latent space. *Computer Methods Appl. Mech. Eng.* **397**, 115128 (2022).
39. Fullwood, D. T., Niezgoda, S. R. & Kalidindi, S. R. Microstructure reconstructions from 2-point statistics using phase-recovery algorithms. *Acta Materialia* **56**, 942–948 (2008).
40. Herman, E., Stewart, J. A. & Dingreville, R. A data-driven surrogate model to rapidly predict microstructure morphology during physical vapor deposition. *Appl. Math. Model.* **88**, 589–603 (2020).
41. Lu, L., Jin, P., Pang, G., Zhang, Z. & Karniadakis, G. E. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nat. Mach. Intell.* **3**, 218–229 (2021).
42. Lin, C. et al. Operator learning for predicting multiscale bubble growth dynamics. *J. Chem. Phys.* **154**, 104118 (2021).
43. Osorio, J. D. et al. Forecasting solar-thermal systems performance under transient operation using a data-driven machine learning approach based on the deep operator network architecture. *Energy Convers. Manag.* **252**, 115063 (2022).
44. Cai, S., Wang, Z., Lu, L., Zaki, T. A. & Karniadakis, G. E. Deepm&mnet: Inferring the electroconvection multiphysics fields based on operator approximation by neural networks. *J. Computational Phys.* **436**, 110296 (2021).
45. Vincent, P., Larochelle, H., Bengio, Y. & Manzagol, P.-A.Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, 1096–1103 (2008).
46. Vincent, P. et al. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. Journal of Machine Learning Research **11** (2010).
47. Gondara, L.Medical image denoising using convolutional denoising autoencoders. In 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW), 241–246 (IEEE, 2016).
48. Bostanabad, R. et al. Computational microstructure characterization and reconstruction: Review of the state-of-the-art techniques. *Prog. Mater. Sci.* **95**, 1–41 (2018).
49. Lanthaler, S., Mishra, S. & Karniadakis, G. E. Error estimates for DeepONets: A deep learning framework in infinite dimensions. *Trans. Math. Its Appl.* **6**, tnac001 (2022).
50. Wang, S., Wang, H. & Perdikaris, P. Learning the solution operator of parametric partial differential equations with physics-informed deeponets. *Sci. Adv.* **7**, eabi8605 (2021).
51. De, S., Hassanaly, M., Reynolds, M., King, R. N. & Doostan, A.Bi-fidelity modeling of uncertain and partially unknown systems using deeponets. Preprint at https://arxiv.org/abs/2204.00997 (2022).
52. Howard, A. A., Perego, M., Karniadakis, G. E. & Stinis, P.Multifidelity deep operator networks. Preprint at https://arxiv.org/abs/2204.09157 (2022).
53. Lu, L., Pestourie, R., Johnson, S. G. & Romano, G. Multifidelity deep neural operators for efficient learning of partial differential equations with application to fast inverse design of nanoscale heat transport. *Phys. Rev. Res.* **4**, 023210 (2022).
54. Dingreville, R. P. M., Stewart, J. A., Chen, E. Y. & Monti, J. M.Benchmark problems for the Mesoscale Multiphysics Phase Field Simulator (MEMPHIS). Tech. Rep., Sandia National Laboratories, Albuquerque, NM (United States) (2020).
55. Süli, E. & Mayers, D. F. *An Introduction to Numerical Analysis* (Cambridge University Press, 2003).
56. Niezgoda, S. R., Kanjarla, A. K. & Kalidindi, S. R. Novel microstructure quantification framework for databasing, visualization, and analysis of microstructure data. *Integrating Mater. Manuf. Innov.* **2**, 54–80 (2013).
57. Gupta, A., Cecen, A., Goyal, S., Singh, A. K. & Kalidindi, S. R. Structure–property linkages using a data science approach: application to a non-metallic inclusion/steel composite system. *Acta Materialia* **91**, 239–254 (2015).
58. Lee, K. & Carlberg, K. T. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *J. Computational Phys.* **404**, 108973 (2020).
59. Kingma, D. & Ba, J. Adam: a method for stochastic optimization. *International Conference on Learning Representations.* (2014).
60. Abadi, M. et al. TensorFlow: Large-scale machine learning on heterogeneous distributed systems. *Software available from* tensorflow.org. **1** (2015).
61. Glorot, X. & Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 249–256 (2010).

ACKNOWLEDGEMENTS

R.D. and G.E.K. acknowledge funding under the *Beyond*Fingerprinting Sandia Grand Challenge Laboratory Directed Research and Development (GC LDRD) program. V.O. and K.S. acknowledge Dr. Elton Chen from Sandia National Laboratories for their valuable discussions that enabled us to better understand the MEMPHIS code. V.O. and K.S. acknowledge Dr. Aniruddha Bora for his insightful suggestions and valuable discussions at various stages of this project. The phase-field framework is supported by the Center for Integrated Nanotechnologies (CINT), an Office of Science user facility operated for the U.S. Department of Energy. This research was conducted using computational resources and services at the Center for Computation and Visualization, Brown University. Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy National Nuclear Security Administration under contract DE-NA0003525. This

paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

AUTHOR CONTRIBUTIONS

R.D. and G.E.K. designed the study and supervised the project. V.O. and K.S. developed the autoencoder–DeepONet architecture and the hybrid model. V.O., K.S. and S.G. analyzed the results. V.O., K.S., S.G., R.D. and G.E.K. wrote the manuscript.

COMPETING INTERESTS

The authors declare no competing interests.

ADDITIONAL INFORMATION

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41524-022-00876-7.

Correspondence and requests for materials should be addressed to Rémi Dingreville or George Em Karniadakis.