PINN-Phase: A Physics-Informed Neural Network Hybrid Framework for Energy-based Transfer Learning in Diffuse Interface Problems

Seifallah Elfetni[*,1] and Reza Darvishi Kamachali[**,1,2]

[1]Federal Institute for Materials Research and Testing (BAM), Unter den Eichen 87, 12205 Berlin, Germany

[2]Institute of Materials Physics, University of Münster, 48149 Münster, Germany

[*]seifallah.elfetni@bam.de; [**]reza.kamachali@bam.de

Abstract

Recently we have proposed PINNs-MPF framework for studying interface dynamics based on multi-phase-field method [1]. In this work, we presents PINN-Phase, to enhances the core elements of the PINNs-MPF framework by incorporating a hybrid framework based on sequential learning, residual connection and an energy-based approach. This framework can be applied independently to various engineering problems. A primary objective of the PINN-Phase framework is to leverage transfer learning to significantly reduce training time and computational resources when addressing new dynamical systems. This is achieved through pre-trained models that minimize the required training data while ensuring adherence to governing physical laws. The architecture allows for incremental training, where the FNN is validated before integration with the ConvRNN, ensuring robust performance across diverse physical scenarios. Key contributions include the incorporation of energy minimization principles into the loss function, facilitating effective modeling of diffuse interface problems. PINN-Phase advances the PINNs-MPF framework by combining the strengths of FNNs and ConvRNNs within a flexible and efficient framework, offering a physically flexible approach to upscaling PINNs-MPF framework in studying interface dynamics and other complex scenarios.

Keywords: PINNs, Phase-field method, Microstructure evolution, ConvLSTM, Machine learning, Neural networks Energy-based schemes.

Contents

1 Introduction

PINNs have emerged as a transformative approach in computational modeling, integrating machine learning with physical laws to solve complex partial differential equations (PDEs) [2–7]. Unlike traditional numerical methods, PINNs offer a mesh-free framework that eliminates the need for discretization or local approximations, thereby enabling flexible and efficient modeling of complex systems [8–16]. Recent advances in this domain, such as the development of frameworks like PhyCRNet [17], have demonstrated the ability of Convolutional-Recurrent Neural Networks (ConvRNN) to effectively address spatiotemporal dynamics while adhering to physical constraints. This capability is particularly advantageous in problems that involve highly nonlinear and multiscale processes, such as those encountered in fluid dynamics, solid mechanics, and phase-field modeling [4, 18–23]. Furthermore, by integrating physical principles directly into the training process, PINNs have the potential to significantly reduce the computational burden associated with traditional methods, such as finite element and finite difference approaches [24, 25]. These advances open up new avenues for applying machine learning techniques to physical sciences, fostering more accurate and computationally efficient solutions across various scientific fields [26]. Regarding diffuse interface problems, only a limited number of frameworks currently address this area. Recent applications of PINNs for handling diffuse interfaces have been presented, primarily focused on the evolution of multiple interacting phases or components [18, 19, 21–23].

Positioning a challenge in microstructure modelling and simulation: Today's computational techniques have profoundly changed the way we conduct materials research, not only expediting our pace in finding solutions, but also serving as invaluable guides for optimizing experiments. On the mesoscale, phase-field methods for modelling and simulation have grown to a versatile tool with tremendous capacity to capture intricate microstructure evolution, with high accuracy and efficiency, in multi-phase, multi-component materials [27–29].

A chief feature of the phase-field approach is that it can be flexibly expanded to different degrees of microstructure complexity, specific to the problem at hand. A representative demonstration of this flexibility is explored in dealing with polycrystalline microstructures: The research in this case started by formulating the core curvature-driven interface motion into the multi-phase-field (MPF) model [30–33], followed by various additional physics being progressively added to the picture, from vacancy drag and annihilation to solute segregation/trapping/drag, secondary-phase precipitation, GB-dislocation interactions and so on [34–37]. This quality (flexibility) gives a huge advantage to the phase-field methods, (i) to adopt for the investigation of emerging multi-physics problems and (ii) to facilitate systematic separation of various physical phenomena over broad range of length and time scales. Yet, there are critical drawbacks attached here: As any additional feature to the model changes the underlying free energy functional and the corresponding equations of motion, huge reprogramming and re-optimization efforts are involved in each stage of development, often down to the deep levels of redefining key parameters and functions, even modifying the memory/storage structures in the software. As a result of this drawback, phase-field simulation packages often contain a large body of codes and develop at a much slower pace (thus being much less popular) than their atomistic compartments, such as ab-initio and molecular dynamics simulation toolkits.

PINNs' perspective to synthesize with phase-field methods: Recently, we have proposed that a synergy between Physics-Informed Neural Networks (PINNs) and MPF methods can level the phase-field frameworks up, while maintaining their flexibility in coping with emerging multi-physics problems: The idea here is to progressively teach the MPF method to PINNs and let it to intra-/extrapolate the

solutions for upcoming physical problems that require reasonably small but still significant departure from the existing MPF model. As a first step towards this aim, we have established the PINNs-MPF framework [1] that efficiently manages the presence of multiple phases within the same spatio-temporal domain through coordinated interactions between different neural networks, referred to as Worker-PINNs. The PINNs-MPF framework adopts multi-optimizers (LFBGS-Adam), uses no data for training and is built on a rich strategy in sub-dividing the spatio-temporal space and properties attributed to a MPF model- PINNs-MPF has been applied to fundamental aspects of simulating interfacial dynamics, namely, the curvature- and force-driven interface motion and the evolution of a triple junction were successfully simulated [1].

As a natural next step for PINNs-MPF is to work on upscaling its capabilities, necessary for addressing increasingly intricate problems in materials science and beyond [17, 38]. Existing architectures, such as PhyCRNet, have demonstrated notable success in capturing spatiotemporal dynamics in partial differential equations (PDEs), employing an encoder-decoder convolutional-recurrent structure to extract low-dimensional features while simulating time evolution through residual connections [17]. However, PhyCRNet and similar models encounter significant computational challenges, particularly when applied to diffuse interface problems. For example, accurate simulations of certain equations, such as Burgers' Equation, often require over 1000 time steps, leading to high computational costs and inefficiencies. These challenges underscore the need for improved generalization capabilities and optimization strategies that minimize energy while maintaining fidelity to underlying physical principles [4, 39]. Similarly, while traditional Feedforward/Artificial Neural Networks (FNNs/ANNs) excel at modeling spatial relationships, they often struggle to capture the temporal dependencies essential for accurately resolving dynamic interface problems [39–42]. Existing models frequently face issues such as overfitting and limited generalization when applied to new configurations or unseen conditions. The complexity of phase transitions and grain boundary dynamics in material systems necessitates frameworks that can evolve to handle diverse physical scenarios. To address these limitations, more sophisticated loss functions that incorporate energy minimization criteria and maintain physical consistency are needed, especially for applications that involve complex diffuse interfaces [4, 42].

The PINNs-MPF framework [1] is primarily based on FNNs, offering an efficient configuration for handling multi-phase interactions that is a challenging area in diffuse interface modeling. In the current work, we aim to build on incorporating sequential learning, as in recurrent networks, to further support temporal modeling in diffuse interface problems and enhance the transfer of learning [43, 44]. By introducing these adjustments, with attention to energy minimization and time-dependent interactions, we aim to refine our approach for capturing the complexities of phase transitions in material science [26, 45, 46]. Hybrid architectures that combine different neural network models have shown considerable success across a variety of applications. These models effectively leverage the temporal modeling capabilities of ConvRNNs, which combine convolutional layers for spatial feature extraction with recurrent layers to capture temporal dependencies. Variants such as ConvLSTM (Convolutional Long Short-Term Memory) and ConvGRU (Convolutional Gated Recurrent Unit) serve as alternative architectures within this category, providing additional flexibility in handling temporal sequences by addressing different memory and gating mechanisms. These networks are complemented by the spatial feature extraction strengths of Graph Neural Networks, Transformers, and FNNs. This synergy enhances predictive performance in complex tasks that require both spatial resolution and temporal continuity. Recent research highlights the efficacy of such hybrid approaches, showcasing their potential to advance capabilities in dynamic system modeling and real-time predictions [47–52].

Here we propose a hybrid framework to integrates FNN and ConvRNN within a unified architecture. This new configuration aims to combine the strengths of both neural network types, allowing for effective spatial feature extraction alongside robust temporal dynamics modeling. By enabling transfer learning and reducing the training data requirements, such a model could offer a comprehensive solution for simulating complex multi-phase-field equations. In this, we pursue to add another core model to the PINNs-MPF hat not only enables a unified temporal solution but also supports larger spatial domains, with the ultimate goal of simplifying parallelization handling. For simplicity and focusing on upscaling aspects, the current development targets single-phase applications and therefore designated under *PINN-Phase*. Figure 1 depicts how the PINN-Phase is targeted to advance PINNs-MPF.

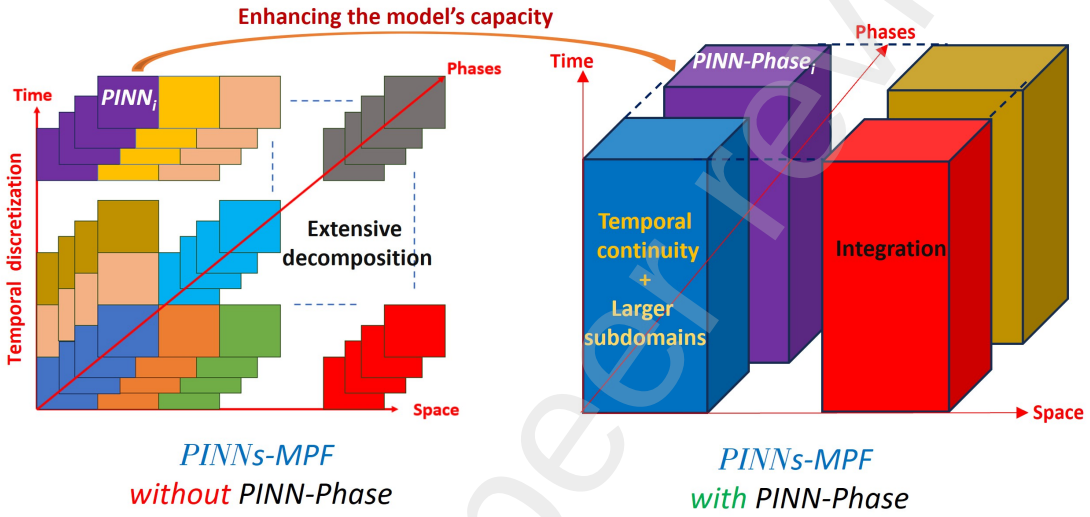


Figure 1: The purposess of the development of the PINN-Phase model: to replace the core model (a standard FNN) in the PINNs-MPF framework [1] , enabling the handling of larger spatial subdomains and providing a unified temporal solution across all phases. This framework also serves as an independent, transfer learning-based approach for diffuse interface problems.

2 The PINN-Phase Framework

2.1 First Trials and the Necessity of an Upscaled Framework

Initially we attempted to incorporate the governing equations of the MPF model directly into the PhyCRNet framework, see details in the appendices. These early trials failed to demonstrate any observable motion of the initial microstructure, suggesting that the inherent complexity and challenging nature of the MPF equations played a critical role. As highlighted in our recent work [1], the MPF equations present unique difficulties when compared to simpler problems, such as Burgers' equation, which PhyCRNet handles more effectively. The PINNs-MPF framework, which is grounded in the FNN architecture, has shown significant capability in managing nonlinearities in MPF set of equations. Yet proper upscaling is required to achieve a unified temporal solution that enhances learning transfer across time-dependent simulations. Handling larger grid sizes is also crucial to extending the framework's applicability to more complex systems. To address these challenges, we here propose and examine a comprehensive architecture designed to better handle the temporal and spatial intricacies of multi-phase-field simulations. This new architecture, outlined in the following sections, is aimed

at overcoming the limitations encountered in earlier models and justifying the need for a more robust solution.

2.2 Architecture and Basic Assumptions of the PINN-Phase Framework

The architecture of the PINN-Phase framework builds upon the PhyCRNet model, specifically tailored to address the complexities of interface problems encountered in MPF simulations. By integrating ConvRNNs, FNNs, and a Variational Autoencoder (VAE), the framework offers a comprehensive solution for modeling spatiotemporal dynamics. A notable enhancement over PhyCRNet is the incorporation of residual connections between the ConvRNN and FNN components. This design choice enables the model to effectively capture spatial and temporal dependencies, particularly in scenarios where traditional neural networks struggle with the diffuse nature of grain boundaries. Additionally, the integration of a multi-head spatial attention (MHSA) module (or simply a Spatial Attention Mechanism) aims to improve the model's focus on critical interfacial regions during phase transitions. The VAE further contributes by capturing uncertainty and variability, which enhances the model's generalization across diverse physical conditions.

The overall architecture of the PINN-Phase framework is illustrated in Figure 2. This flowchart highlights the flow of information through the encoder, ConvLSTM cells, and the residual connection between the FNN and ConvRNN components, which ensures a smooth transition of learning across time steps. The decoder reconstructs the final solution, providing a detailed representation of the evolving phases.

Basic Ansatzes

The architecture of the PINN-Phase framework is built on several core ansatzes:

- **Ansatz 1:** The solution at any given time t is represented as a combination of a ConvRNN, which captures the temporal dynamics, and a FNN, which refines spatial features.
- **Ansatz 2:** To ensure the effective training of the ConvRNN model, it is necessary to first independently validate the solution provided by the FNN. This step ensures the reliability of the smaller FNN model before integrating it into the larger ConvRNN framework for joint learning.
- **Ansatz 3:** The PINN-Phase framework integrates multiple components, including ConvRNN, layers, FNN, and a VAE. Thus, the architecture is designed to be flexible, allowing for the transfer of learned components (either the entire model or specific parts) between phases, facilitating modular and adaptable learning transfer.
- **Ansatz 4:** Similar to classical diffuse interface solvers, the PINN-Phase model relies on minimizing the system's internal energy, which is a crucial principle incorporated into the loss function.
- **Ansatz 5:** While the model is designed to learn from data, including labeled data and ground-truth solutions, validating an unsupervised version of the model is essential to ensure it adheres fully to its physically-guided foundations.

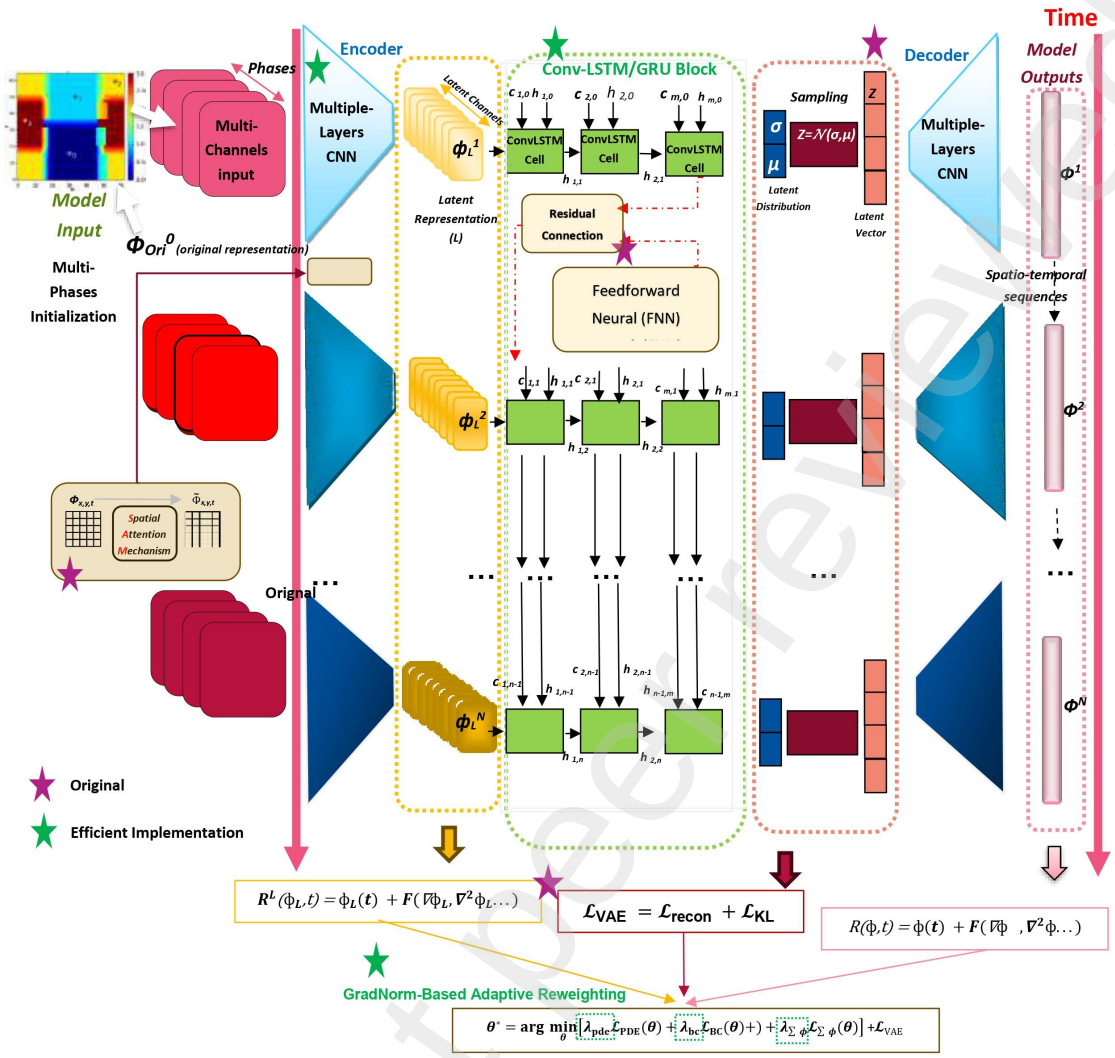


Figure 2: PINN-Phase: An architecture integrating ConvRNN, FNN, MHSA, and VAE components for solving phase-field problems. Readers are encouraged to refer to the architecture of the PhyCRNet framework [17] for comparison, illustrating how the PINN-Phase model is upscaled to address diffuse interface problems and can incorporate data for transfer learning.

Algorithm 1 Resolution using a ConvRNN-FNN hybrid framework with Residual Connections.

Input: Initial state, Input data x, Number of time steps N, Time step size Δt, Encoder, Decoder, ConvLSTM, FNN, VAE flag

Output: Final outputs, Latent states, VAE loss (if applicable)

Initialize internal state and output containers;
Create spatial grid based on input dimensions;
for *each time step* t *from* 0 *to* $N-1$ **do**
    Compute FNN output using spatial and temporal input;
    Apply spatial attention to capture interfacial dynamics (See Appendix B for more details on the spatial attention mechanism.);
    Update encoded input through the Encoder;
    if *VAE is enabled* **then**
        Compute latent variables and reconstruct using the VAE;
    else
        Reconstruct using the standard decoder;
    Update internal state through ConvLSTM cells;
    Combine FNN and ConvLSTM outputs using residual connections (Eq. 1);
    Apply activation and normalization to the combined output;
    Store outputs and update latent representations;
Return final outputs, latent states, and VAE loss (if applicable);

The residual connection, mentioned in Algorithm 1, can be expressed as follows to include a learnable weighting parameter γ:

$$x_t = (1 - \gamma) \cdot x_{FNN,t} + \gamma \cdot \Delta t \cdot x_{RNN,t} \tag{1}$$

Where: x_t or x is the output of the residual connection, $\gamma \in [0, 1]$ is a learnable parameter that balances the contributions of the FNN and ConvRNN components, $(1 - \gamma) \cdot x_{FNN,t}$ controls the contribution of the FNN, $\gamma \cdot \Delta t \cdot x_{RNN,t}$ controls the contribution of the ConvRNN, scaled by the time step Δt. This formulation allows the model to learn the optimal balance between the spatial dependencies captured by the FNN and the temporal dynamics captured by the RNN. In Appendix A, an additional graphical illustrations are provided to better showcase how the interaction between the larger model and the smaller model is ensured, as well as how the marching scheme is implemented.

3 Methods

3.1 Multi-Phase-Field Method

Consistent with the PINNs-MPF framework, we work with the multi-phase-field (MPF) model to describe grain boundary dynamics in polycrystalline materials. The evolution of each grain is governed by phase-field variables, ϕ_α, where α represents the phase (grain) index. The dynamics of these phase-fields follow the governing equation [28]:

$$\dot{\phi}_\alpha = \sum_{\beta=1..N} \frac{\mu_{\alpha\beta}}{N} \left[\sigma_{\alpha\beta}(I_\alpha - I_\beta) + \frac{\pi^2}{4\eta} \Delta g_{\alpha\beta} \right] \tag{2}$$

where $\mu_{\alpha\beta}$ denotes the pairwise grain boundary mobility, $\sigma_{\alpha\beta}$ represents the grain boundary energy, and $\Delta g_{\alpha\beta}$ accounts for non-interfacial contributions to the free energy. The term I_α in this equation is expressed as:

$$I_\alpha = \nabla^2 \phi_\alpha + \frac{\pi^2}{\eta^2} \phi_\alpha \tag{3}$$

To ensure consistency, the sum constraint $\sum_{\alpha=1..N} \phi_\alpha = 1$ is imposed on the phase-field variables, ensuring that the system respects the conservation of order parameters.

As earlier discussed, in order to focus our effort on the hybrid upscaling aspects of the PINNs, in this study we deal here with two-phase systems. For such a system, the evolution equation reduces to:

$$\dot{\phi} = \mu \left[\sigma \left(\nabla^2 \phi + \frac{\pi^2}{2\eta^2} (2\phi - 1) \right) + \frac{dh}{d\phi} \Delta g \right] \tag{4}$$

where the function $\frac{dh}{d\phi} = \frac{\pi \sqrt{\phi(1-\phi)}}{\eta}$ is associated with the double obstacle potential, and $\mu = \frac{8\eta L}{\pi^2}$ is the kinetic coefficient of the interface. Note that the size of the system and the geometry of the interfaces for which we must solve this equation are arbitrary and therefore still complex. We consider isotropic interfacial mobility (μ) and energy (σ), as well as a constant driving force Δg.

The total time derivative of the free energy can be expressed as an integral over the domain Ω, with two distinct residual components that should compensate for each other during the motion. The first term corresponds to the energy derivative, while the second term relates to the evolution of the order

parameter ϕ. This leads to the following expression for the additional residual term:

$$\frac{dF}{dt} = \int_{\Omega} \left\{ \frac{4\sigma}{\eta} \left[\frac{d\phi}{dt}(1-2\phi) + \frac{\eta^2}{\pi^2} 2(\nabla\phi) \cdot \nabla \left(\frac{d\phi}{dt} \right) \right] + \frac{dh}{d\phi} \frac{d\phi}{dt} \Delta g \right\} \tag{5}$$

In this form, the first residual is associated with the energy derivative term $\frac{d\phi}{dt}(1-2\phi) + \frac{\eta^2}{\pi^2} 2(\nabla\phi) \cdot \nabla \left(\frac{d\phi}{dt} \right)$, while the second residual is related to the evolution of the order parameter through $\frac{dh}{d\phi} \frac{d\phi}{dt} \Delta g$. The negative sign in the energy values is purely numerical, reflecting the shrinking nature of the grain in the grain shrinkage scenario. Integrating the energy residual into the model as a loss term allows the system to better account for deviations from the expected energy dynamics, thereby improving the model's accuracy in adhering to physical conservation principles [53, 54].

3.2 Loss Formalism

The total loss L_{Total} in the PINN-Phase framework comprises multiple components that capture different aspects of the model's performance. These losses are derived from four key parts: the partial differential equation (PDE) residuals, the Variational Autoencoder (VAE) reconstruction, the Feedforward Neural Network (FNN) contributions, and the energy dynamics. The total loss is expressed as:

$$L_{\text{Total}} = \alpha_{\text{PDE}}(L_{\text{PDE}} + L_{\text{Latent}}) + \alpha_{\text{IC}} L_{\text{IC}} + \alpha_{\text{Energy}} L_{\text{Energy}} + \alpha_{\text{VAE}} L_{\text{VAE}} + \alpha_{\text{FNN}} L_{\text{FNN}} + \alpha_{\text{Data}} L_{\text{Data}}. \tag{6}$$

The PDE-based loss L_{PDE} ensures that the model adheres to the physical laws governed by the system's partial differential equations. This loss has two components: the standard PDE residual and a latent-space representation of the residual. The standard PDE residual is defined as:

$$R(\phi, t) = \phi(t) + F(\nabla\phi, \nabla^2\phi, \dots), \tag{7}$$

where $\phi(t)$ is the phase field variable, and F is a functional representing the physical constraints such as spatial gradients and higher-order terms. In addition to the standard residual, the latent-space residual is also considered to evaluate how well the model captures the essential dynamics in the latent space. This latent PDE residual is expressed as:

$$R^L(\phi_L, t) = \phi_L(t) + F(\nabla\phi_L, \nabla^2\phi_L, \dots), \tag{8}$$

where $\phi_L(t)$ represents the latent variables in the multi-channel latent space derived from the original PDE.

The VAE loss consists of the reconstruction loss L_{recon} and the Kullback-Leibler (KL) divergence loss L_{KL}, ensuring that the latent variables in the VAE follow a desired distribution while reconstructing the original data [55–57]. The VAE loss is defined as:

$$L_{\text{VAE}} = L_{\text{recon}} + L_{\text{KL}}. \tag{9}$$

The FNN loss L_{FNN} ensures that the Feedforward Neural Network handles initial conditions and motion within the phase-field system. The FNN loss is composed of two parts: $L_{\text{FNN}} = L_{\text{IC}} + L_{\text{motion}}$, where L_{IC} corresponds to the loss on the initial condition handled by the FNN, and L_{motion} accounts for the motion prediction in the phase-field system.

The energy loss L_{Energy} penalizes deviations from the expected physical energy dynamics, ensuring that the model adheres to energy conservation principles in the real dimensions of the system.

Finally, all components are combined into the overall optimization objective, where GradNorm-based adaptive reweighting is applied to balance the different losses dynamically during training:

$$\theta^* = \arg\min_\theta \left[\mathcal{L}(\theta)\right] = \arg\min_\theta \Big(\alpha_{\text{PDE}}\mathcal{L}_{\text{PDE}}(\theta) + \alpha_{\text{IC}}\mathcal{L}_{\text{IC}}(\theta) + \alpha_{\text{Energy}}\mathcal{L}_{\text{Energy}}(\theta) + \alpha_{\text{Latent}}\mathcal{L}_{\text{Latent}}(\theta) + \alpha_{\text{VAE}}\mathcal{L}_{\text{VAE}}(\theta) + \alpha_{\text{FNN}}\mathcal{L}_{\text{FNN}}(\theta) + \alpha_{\text{Data}}\mathcal{L}_{\text{Data}}(\theta) \Big). \tag{10}$$

This formulation ensures that the model effectively integrates all components of the system dynamics and adapts to the different learning objectives through the dynamic reweighting of loss terms.

This formulation captures the multi-faceted nature of the model's learning process, allowing it to leverage various losses to enhance its predictive capabilities and generalization performance.

3.3 Modified GradNorm approach

When training large models with multiple loss components, it has been observed in various studies that the model often struggles to optimize all losses simultaneously. This issue increases the risk of overfitting to certain losses while others remain under-optimized. To address this, several methods, including Gradient Normalization for Adaptive Loss Balancing (GradNorm), have been proposed to balance the contribution of multiple losses dynamically [58, 59]. In this work, we introduce a modified GradNorm approach (c.f. the original implementation of the GradNorm in Appendix C), which is especially advantageous within the PINN-Phase framework. GradNorm is appealing due to its minimal additional hyperparameters, but it does increase computational cost by requiring two backpropagation steps and retention of the computation graph, a limitation widely acknowledged to demand more resources. Our trials with the original GradNorm [58] revealed significantly slower training. This slowdown arises from the dual iteration process needed for computing the total and GradNorm losses: one iteration to update model parameters using the total loss at each epoch, and an additional iteration for computing the GradNorm loss. Consequently, this induces a high number of total iterations (e.g., 300,000 in [60]), resulting in very slow training. Our trials with the standard GradNorm approach highlight these computational challenges. To overcome this, we propose a modification to GradNorm, where we explicitly compute the weighting terms at each iteration and inject the updated values directly into the main loss computation. This approach should promote a balanced weighting across all loss components, enhancing model stability and performance without requiring additional backpropagation steps. Solution convergence, reasonable training times, and an adequate number of epochs to span the entire temporal domain serve as criteria to validate this assumption.

Algorithm 2 The proposed Modified GradNorm for Multi-Loss Models.

Input: Loss components $L_1, L_2, \ldots, L_M$, initial weights $w_1, w_2, \ldots, w_M$, initial loss values $L_{1,0}, L_{2,0}, \ldots, L_{M,0}$, learning rate β, constant α

Output: Updated weights $w_1, w_2, \ldots, w_M$, updated model parameters θ

Initialize weights $w_m = 1.0$ for all $m \in \{1, 2, \ldots, M\}$; Initialize sum of weights $T = \sum_{m=1}^{M} w_m$;

for *each epoch* **do**

  Compute combined loss $L_{\text{combined}} = \sum_{m=1}^{M} w_m \cdot L_m$;

  for *each loss component* L_m **do**

    Compute gradient $g_m = \|\nabla_\theta (w_m \cdot L_m)\|_2$; Compute loss ratio $r_m = \frac{L_m}{L_{m,0}}$;

  Compute average gradient norm $g_{\text{avg}} = \frac{1}{M} \sum_{m=1}^{M} g_m$;

  for *each loss component* L_m **do**

    Compute target gradient norm $t_m = g_{\text{avg}} \cdot \left(\frac{r_m}{\frac{1}{M} \sum_{k=1}^{M} r_k} \right)^{\alpha}$;

  for *each weight* w_m **do**

    Update weight $w_m = w_m \cdot (1 - \beta) + \beta \cdot \left(\frac{t_m}{g_m} \right)$;

  Normalize weights $w_m = \frac{w_m}{\sum_{k=1}^{M} w_k} \cdot T$;

  Perform backpropagation on L_{combined} to update model parameters θ;

4 Results and Discussion

The simulations were performed on an HPC cluster with 25 nodes, a total of 2200 cores, and 10240 GB of RAM, providing a high-performance environment suitable for large-scale scientific simulations.

For the 64×64 configuration, the spatial grid is defined with $N_x = 64$ and $N_y = 64$, corresponding to the number of grid points along the x- and y-directions, respectively. The simulation involves $N_t = 50$ time steps. The spacing between grid points in both the x- and y-directions is computed as Δx and Δy, ensuring a uniform grid distribution across the domain. The total number of phase is num_phases $= 1$, where the substrate is basically the absence of the considered phase. The time step size, Δt, is derived from the simulation bounds and the number of time steps. The interfacial energy is set to $\sigma = 1$, and the mobility parameter, μ, is 5×10^{-5}, representing the movement of the phase interface. The driving force is zero, with $\Delta g = 0$, and the interface width is calculated as $\eta = 7 \times \Delta x$.

The latent representation of the system is handled through a sequence of different architectures, which includes convolutional layers for both encoding and decoding. The hidden channels for these layers are set to [8, 32, 64, 64], which determine the number of output channels at each layer. The kernel size is fixed at [3, 3, 3, 3], with strides [2, 2, 2, 1] for the auto-encoders and ConvLSTM layer, providing spatial reduction and temporal processing capabilities. Each layer uses a padding of [1, 1, 1, 1] to preserve spatial dimensions. The architecture includes three auto-encoder layers followed by one ConvLSTM layer. Intermediate dimensions are set to 512, with a latent dimension of 128 to effectively capture the reduced representation of the phase-field system. The model parameters are gathered in 1.

When transitioning to a 128×128 domain, the benchmark parameters (physical and geometrical) are adjusted as follows: The number of spatial grid points is increased to $N_x = 128$ and $N_y = 128$, while the number of time steps remains at $N_t = 50$. The grid spacing in both the x- and y-directions (Δx and Δy) becomes finer, leading to improved spatial resolution. The time step size (Δt) is recalculated based on the updated upper bound for the temporal dimension. The interface energy remains the same while the mobility is adjusted to $\mu = 7.5 \times 10^{-6}$, and the interface width (η) increases to $15 \times \Delta x$, allowing for a wider interface for phase transitions.

Table 1: Model Parameters and Architecture of the PINN-Phase Framework

Layer	Details	Activation / Dropout
Encoder		
Encoder Block 1	Conv2d(1, 8, 3x3, stride=2, padding=1, mode=circular)	ReLU, Dropout(0.5)
Encoder Block 2	Conv2d(8, 32, 3x3, stride=2, padding=1, mode=circular)	ReLU, Dropout(0.5)
Encoder Block 3	Conv2d(32, 64, 3x3, stride=2, padding=1, mode=circular)	ReLU, Dropout(0.5)
MHSA (Multi-Head Spatial Attention)		
MHSA	Linear(4096, 4096) for Q, K, V	-
ConvLSTM Cells		
ConvLSTMCell 1	Conv2d(64, 64, 3x3) for Wxr, Whr, Wxz, Whz, Wxh, Whh	Dropout(0.5)
ConvLSTMCell 2	Conv2d(64, 64, 3x3) for Wxr, Whr, Wxz, Whz, Wxh, Whh	Dropout(0.5)
ConvLSTMCell 3	Conv2d(64, 64, 3x3) for Wxr, Whr, Wxz, Whz, Wxh, Whh	Dropout(0.5)
ConvLSTMCell 4	Conv2d(64, 64, 3x3) for Wxr, Whr, Wxz, Whz, Wxh, Whh	Dropout(0.5)
ConvLSTMCell 5	Conv2d(64, 64, 3x3) for Wxr, Whr, Wxz, Whz, Wxh, Whh	Dropout(0.5)
Fully Connected (FC)		
FC 1	Linear(4096, 512)	-
FC_mu	Linear(512, 128)	-
FC_logvar	Linear(512, 128)	-
Decoder		
Decoder Block 1	ConvTranspose2d(64, 32, 3x3, stride=2, padding=1, output_padding=1)	ReLU, Dropout(0.5)
Decoder Block 2	ConvTranspose2d(32, 8, 3x3, stride=2, padding=1, output_padding=1)	ReLU, Dropout(0.5)
Decoder Block 3	ConvTranspose2d(8, 1, 3x3, stride=2, padding=1, output_padding=1)	ReLU, Dropout(0.5)
FNN		
Linear Layer 1	Linear(3, 128)	ReLU
Linear Layers 2-6	5x Linear(128, 128)	Dropout(0.05)
Residual Blocks 1-6	6x Residual Block with Linear(128, 128)	ReLU
Output Layer	Linear(128, 1)	-

In the ConvLSTM architecture, the number of hidden channels is modified to [8, 32, 128, 128], and the kernel sizes for the auto-encoders and ConvLSTM layers are changed to [4, 4, 4, 3] to accommodate the larger grid. The stride and padding values remain consistent at [2, 2, 2, 1] and [1, 1, 1, 1], respectively, while the number of layers is maintained at three auto-encoders and one ConvLSTM layer. Finally, the intermediate and latent dimensions are kept at 512 and 128, respectively, ensuring a balance between computational load and latent space size.

The configurations of the four benchmarks are presented in Table 2. Starting from an initial single-grain shrinkage configuration with a grid size of 64×64, we progressively increase the complexity to a final benchmark containing 35 particles, initialized on a grid size of 128×128. The initial benchmark operates in a fully unsupervised mode to evaluate the model's capacity to predict solutions without relying on data, allowing only auto-transfer of learning. To further explore the scope of this work, transfer learning is applied from one benchmark to the next (from Benchmark 2 to 3, then 3 to 4), with the model given minimal data reliance. Specifically, for these benchmarks, 10% of the total sequences are supervised, while the model operates in an unsupervised mode for the remaining 90% of the training, thereby testing its ability to generalize effectively.

Table 2: Configurations of the four benchmarks: hyperparameters, mean radius (MR), standard deviation (STD), transfer learning approach (TL), and supervision mode (supervised or unsupervised).

Benchmark	Grid size	Initial grains number	MR, STD	Transfer of Learning (TL)	Supervision
Grain shrink 1	64 x 64	01	0.42, 0	Auto TL	unsupervised
Grain shrink 2	128 x 128	01	0.42,0	Auto TL	10 %
Grain growth (two particles)	128 x128	02	0,325, 0.1	From Bencharmk 2	10 %
Grain growth (multi-grain)	128 x128	35	0.4 ,0.15	From Benchmark 3	10 %

In Table 3, we highlight some key model training choices as well as additional hyperparameters (in addition to those shown in Table 1) introduced by the use of the residual connection and the modified GradNorm proposed here. While the global architecture relies on a VAE, it is reasonable to start

Table 3: Key training strategy choices and hyperparameters for the model across each benchmark.

Benchmark	VAE/AE	Residual connection (γ)	GradNorm		RNN (LSTM/ GRU)	Threshold	Training intervals	Training time ($\approx$)
			α	β				
Grain shrink 1	VAE	0.85	0.5	0.015	LSTM/GRU	1.7×10^{-2}	50	3 h
Grain shrink 2	AE	0.9	0.5	0.01	LSTM	1.85×10^{-2}	100	6.5 h
Grain growth (two- grain)	AE	0.9	0.5	0.01	LSTM	5×10^{-3}	50	8 h
Grain growth (multi-grain)	AE	0.9	0.5	0.01	LSTM	2.1×10^{-2}	50	12 h

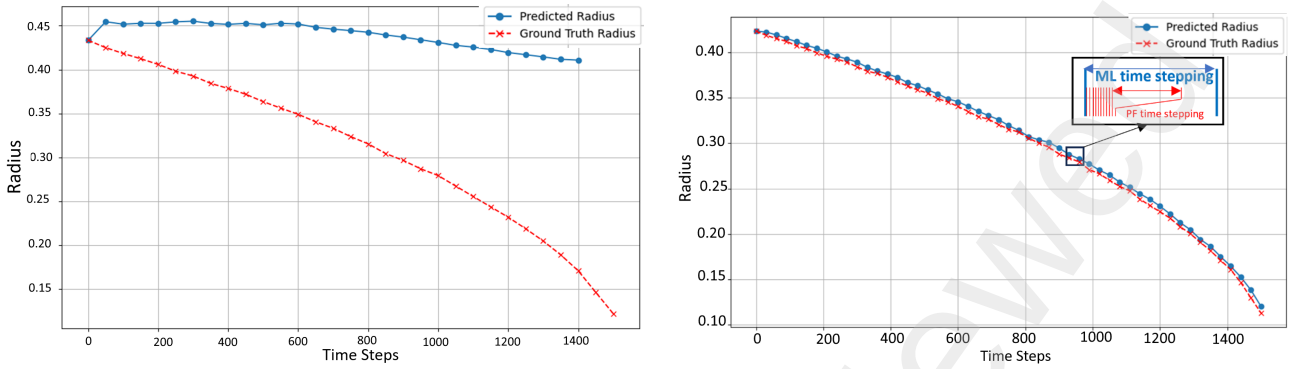
with relatively simple configurations, where the reconstruction loss of the VAE is utilized, and the KL divergence loss is not employed to reduce the number of trainable parameters in the model. Thus, the full VAE configuration is only tested within the first benchmark, while a simple Autoencoder (AE) is then adopted for the others. Similarly, the first benchmark was tested with both Convolutional Gated Recurrent Unit (ConvGRU) and ConvLSTM networks, while a ConvLSTM is used for the remaining benchmarks to accommodate a larger recurrent neural network as complexity increases. For the additional hyperparameters, specifically those related to the residual connection and the modified GradNorm, the values are kept practically unchanged. A value of 0.9 for γ was chosen after initial trials, where it was found that the model performs well when relying more on the FNN or ANN. The α and γ values are conventional starting values from the literature. The threshold allowing for the progressive increase of the time domain was identified through trial and error. This was done by determining an optimal value of the global loss that allows for a correct solution within the first time interval; this value could then be retained as a constant for simplification or adjusted at each transition, considering that the global loss must fall below the threshold to establish a new threshold.

4.1 Grain shrinkage scenario and justification of the framework

Figure 3 presents a comparison of the model's performance in a grain shrinkage scenario. It is important to note that we utilize grid-based inputs rather than collocation points in this analysis. The results indicate that without the application of the modified GradNorm approach, the model fails to reproduce accurate solutions. In particular, the behavior of the FNN becomes problematic, leading to overfitting on the initial condition (IC) loss of the FNN, expressed as $L_{\text{FNN}} = L_{\text{IC}} + L_{\text{motion}}$, where L_{IC} denotes the initial condition loss, penalizing deviations from expected initial conditions, and L_{motion} captures the accuracy of the model's representation of temporal dynamics. This overfitting causes the observed irregular behavior.

Upon applying the modified GradNorm method, we observe a significant improvement, with a strong fit between the PINN-Phase predictions and the theoretical solution obtained from a pure phase-field solver, achieving a mean squared error (MSE) loss below 10^{-3}. It is also noteworthy that only 50 time intervals were used in this case, compared to 1500 for the ground truth solution, highlighting the high nonlinear capacity of the model. The model effectively minimizes free energy; however, its performance is significantly influenced by the correction of other losses and appropriate weighting.

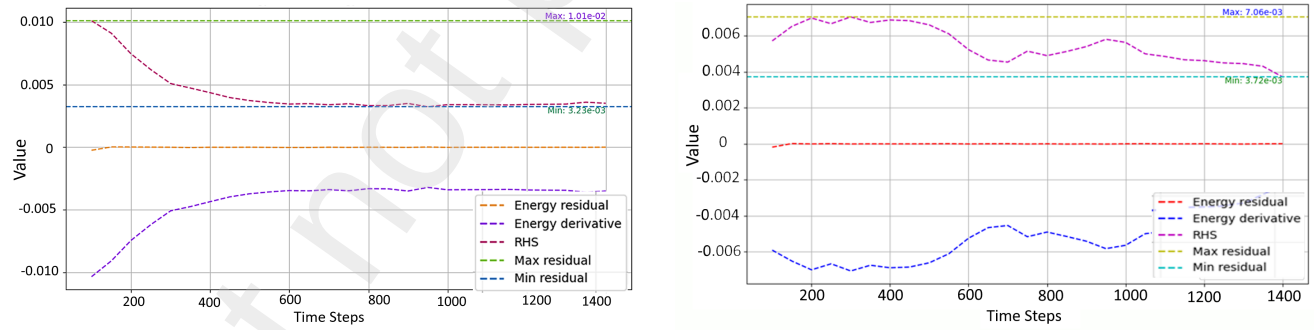
As illustrated in Fig. 4a, the energy loss has a mean absolute error (MAE) of 10^{-2}, which can be considered relatively high. This elevated residual value, observed during the initial stage of the simulation, corresponds to the undesirable motion noted from the very first simulation time step. It is worth noting that while the energy minimization loss serves as a critical component of the model, it should be regarded as an additional criterion for controlling the solution rather than a mechanism for the model to autonomously adjust its parameters. This distinction is crucial for understanding how the model balances the different loss components and the overall impact on the simulation's accuracy.



(a) without loss weighting (b) with loss weighting

Figure 3: Justification of the GradNorm method: Comparison between a standard approach for FNN training without dynamic weighting of the losses (and consequently the global model) versus the application of a modified GradNorm approach.

The results of the normal grain shrinkage simulation, without the presence of any external driving forces, are shown in Fig. 5. The model successfully simulates the shrinkage of an isolated grain, a configuration that is known to be computationally challenging. This challenge arises from the need for the model to learn the nonlinear curvature-driven dynamics, particularly in the absence of external forces. The grain shrinkage equation contains a key term, $\eta\nabla^2\phi$ and $\frac{\pi^2}{\eta}(\phi-\frac{1}{2})$, which adds complexity to the simulation. Accurately representing this term is essential, as it governs the curvature-driven grain boundary motion. Correctly simulating this grain shrinkage scenario demonstrates the model's ability to handle one of the major requirements for studying grain growth phenomena. This establishes the basis for addressing more complex configurations reported in the literature.

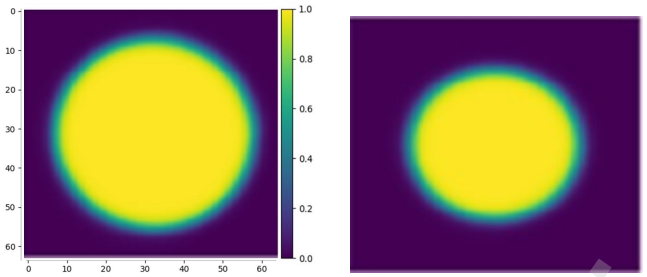


(a) Without loss weighting (b) With loss weighting

Figure 4: Justification of the integration of the energy loss in the global loss expression: Comparison between a standard approach for energy evolution without dynamic weighting of the losses (and consequently the global model) versus the application of a modified GradNorm approach.

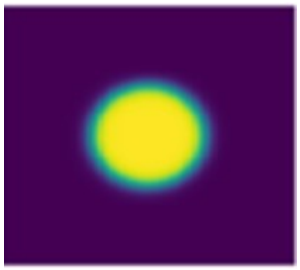
The current results surpass the previous output provided by the PINNs-MPF framework, which, despite relying on a multi-network approach, struggles to achieve this level of both quantitative and qualitative accuracy. As PINNs-MPF is categorized among "small models," our findings suggest that employing larger models, like the one used here, is highly beneficial for tackling such complex problems. Although the calibration and tuning of larger models may seem challenging at first, the performance improvements observed in this case justify the effort. It is worth noting that PINNs-MPF is fundamentally designed to handle multi-phase systems, a specificity that for PINN-Phase has not been yet demonstrated and remains for a future study. Therefore, it would be recommended

that the two frameworks work in a complementary manner when addressing large-scale simulations involving multiple grains.



(a) ϕ at Time (1 %)

(b) ϕ at Time (22 %)



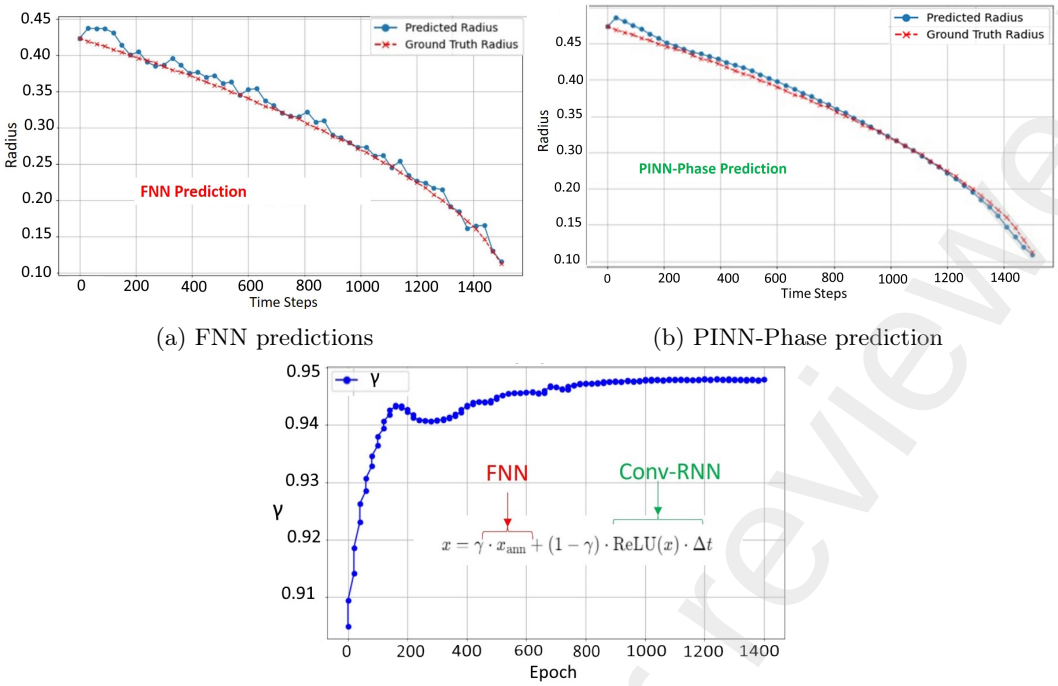
(c) ϕ at Time (66 %)

(d) ϕ at Time (100 %)

Figure 5: PINN-Phase results of the normal grain shrinkage without external driving force.

Figure 6 demonstrates how the model dynamically updates the residual connection. Although the model tends to rely more on the FNN, it is important to note that the parameter γ is a hyperparameter that requires empirical tuning. We initialize $\gamma = 0.9$ after preliminary trials. This choice was based on observations that the global behavior of the model began smoothly. Subsequent trials confirmed this initial choice as γ progressively increased. We observe that FNN predictions are discrete and may fluctuate, whereas the PINN-Phase predictions are continuous and effectively capture the entire temporal domain. This distinction underscores that the numerical values produced by the FNN do not imply a minor role for the ConvRNN. In fact, the ConvRNN plays a crucial role in learning the solution across the global temporal domain, which the FNN alone cannot ensure as it excels only in discrete time intervals (marching PINN). In summary, the training strategy is validated by the fact that the FNN assists the model, while the ConvRNN ensures a smooth and continuous solution.

The evolution of all losses is showcased in Fig. 7. Fig. 7(a) shows that the losses start with a substantial imbalance, underscoring the need for dynamic adjustment to avoid adverse effects on model training. This observation highlights the importance of the modified GradNorm approach, especially as the different weights converge to a steady value of 1, indicating balanced consideration of all loss components by the model. However, when imbalances reappear, such as near epochs 700 and 1000, model performance may be affected. Notably, this approach serves as a compromise between training speed—particularly in comparison to the original GradNorm implementation [58]—and the appropriate weighting of different losses. Given that the model can partially rely on data, this method proves to be practical. Here, we note that the data loss are used for tracking purposes and not integrated in the main loss term for training. In Figs. 7(b) and 7(c), the substantial discrepancies among the different losses are clearly illustrated, ranging from 10^{-1} to 10^{-9} in MSE. Such a situation poses challenges for the model, as it cannot effectively manage adaptive weighting under these conditions.

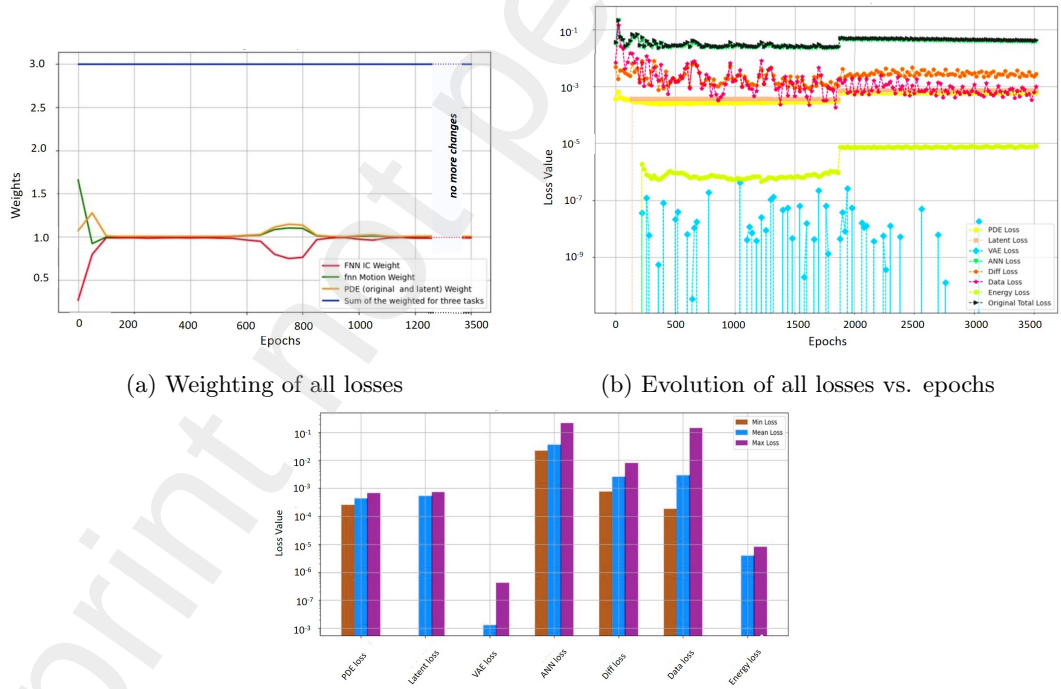


(a) FNN predictions

(b) PINN-Phase prediction

(c) Residual evolution

Figure 6: Justification of the residual connection and the effectiveness of combining a hybrid architecture: FNN and ConvRNN to constitute the global model.



(a) Weighting of all losses

(b) Evolution of all losses vs. epochs

(c) Histogram representation of all losses.

Figure 7: Weighing of the relevant losses (FNN (IC, Motion) + ConvLSTM(Motion (original + latent)).

4.2 Transfer of Learning

The results of the transfer learning campaign are shown in Fig. 8. It is important to note that the PINN-Phase framework has already satisfied the second assumption by successfully addressing a

complex grain shrinkage scenario without reliance on data. At this stage, the model is now equipped to accept data after verifying its robustness. Otherwise, a PINN solution like the current one could be seen as inefficient compared to classical solvers if limited to this initial stage. Instead, the current solution is proposed as a transfer learning-based model that can also learn from data and be used as a pre-trained model while progressively upscaling towards generalization.

Initially, we assess the auto-transfer of learning using the model's checkpointing option. Next, we utilized the pre-trained model to simulate a grain shrinkage scenario with a decreased interface width, presenting a more challenging condition. The model quickly learns the solution and ensures partial training; it was trained for 600 epochs (approximately 20 minutes), whereas without pre-training, modeling the grain shrinkage would have taken about 3 hours. An early stopping criterion is employed to prevent the model from overfitting and making undesirable predictions.

In the third configuration, the transfer of learning is again applied within the same solution context to accelerate simulation scenarios. For this, the model is trained on only 10% of the total time interval using available data, after which it switches to an unsupervised mode to predict the remainder of the solution. This configuration is proposed to be tested in general scenarios where it is possible to compute the solution using classical solvers, but where the total time interval is computationally expensive. In such cases, machine learning can substitute the classical solver to predict the remainder of the solution through training.

4.3 Generalization

As a next step, we performed extensive studies of systems with more complex geometries, in which we have a distribution of our phase within the domain. For a two-grain system, the results of the transfer learning campaign are shown in Fig. 9. For this simulation, we utilized a grid size of $N_x = 128$ and $N_y = 128$, with $N_t = 50$ time steps. The lower bounds were set as $\text{lb} = [0, 0, 0]$ and the upper bounds as $\text{ub} = [1, 1, 1000]$. The physical parameters included one phase ($\text{num\_phases} = 1$), the interface energy is set to $\sigma = 1$, mobility defined as $\mu = 7.5 \times 10^{-6}$, a driving force of $\delta_g = 0$, and interface width determined as $\eta = 15 \times dx$.

The evolution of the mean radius is computed by the FNN, as shown in Fig. 10(a), and the larger model's predictions are shown in Fig. 10(b). The model exhibits good agreement between the ground truth solution and the predictions, highlighting that the PINN-Phase solution acts as a smoothing of the FNN results. This indicates effective collaboration between the ConvRNN in handling spatial resolution, while the FNN captures the nonlinearities well. The slight differences between the smoothed solution and the ground truth can be attributed to variations in the computing schemes, particularly considering that our model accurately captures the positions of the peaks.

The result of the benchmark with multi-grain initialization is shown in Fig. 11. It is notable that this benchmark inherits learning from the previous case (two-grain system) to utilize a transfer learning approach. A minimal amount of supervision is provided by using the ground truth solution for 10% of the total training time. This setup allows the model to rely on a foundational level of supervision and data. The model quickly adapts to the new physical behavior and achieves the expected output.

The results for the mean grain radius evolution are shown in Fig. 12, providing quantitative validation of the mean radius trend over time. This behavior is typical in multi-grain scenarios without conservation, where particles in proximity coalesce, while isolated grains tend to shrink and eventually disappear.

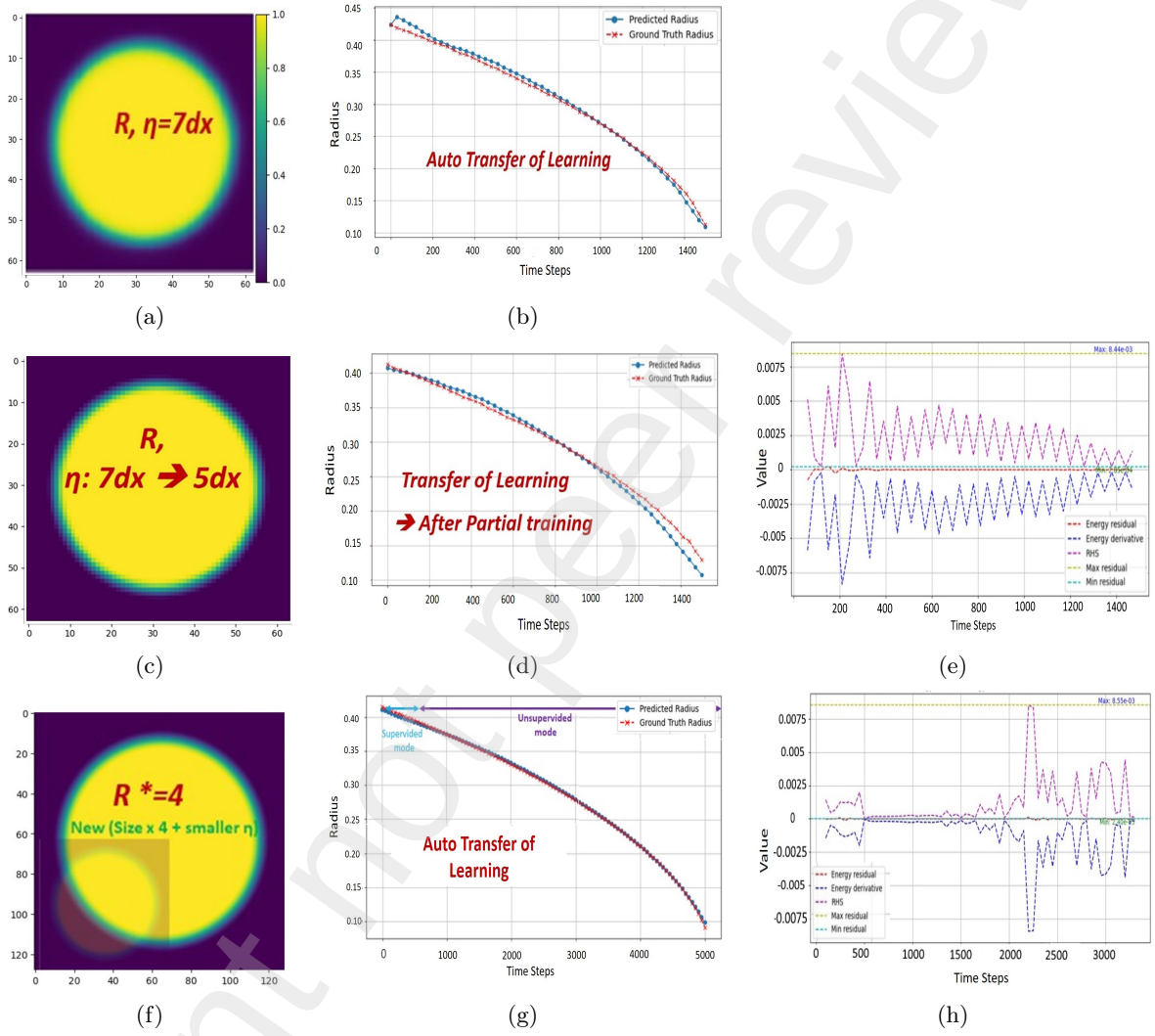


Figure 8: Results of the transfer learning campaign. The upper row illustrates the auto-transfer of learning using the model's checkpointing option. The middle row presents the transfer of learning from the initial configuration to a new configuration with a smaller interface width (η). The bottom row shows a configuration where the grid size is four times larger than the initial configuration, highlighting the effects of partial training.

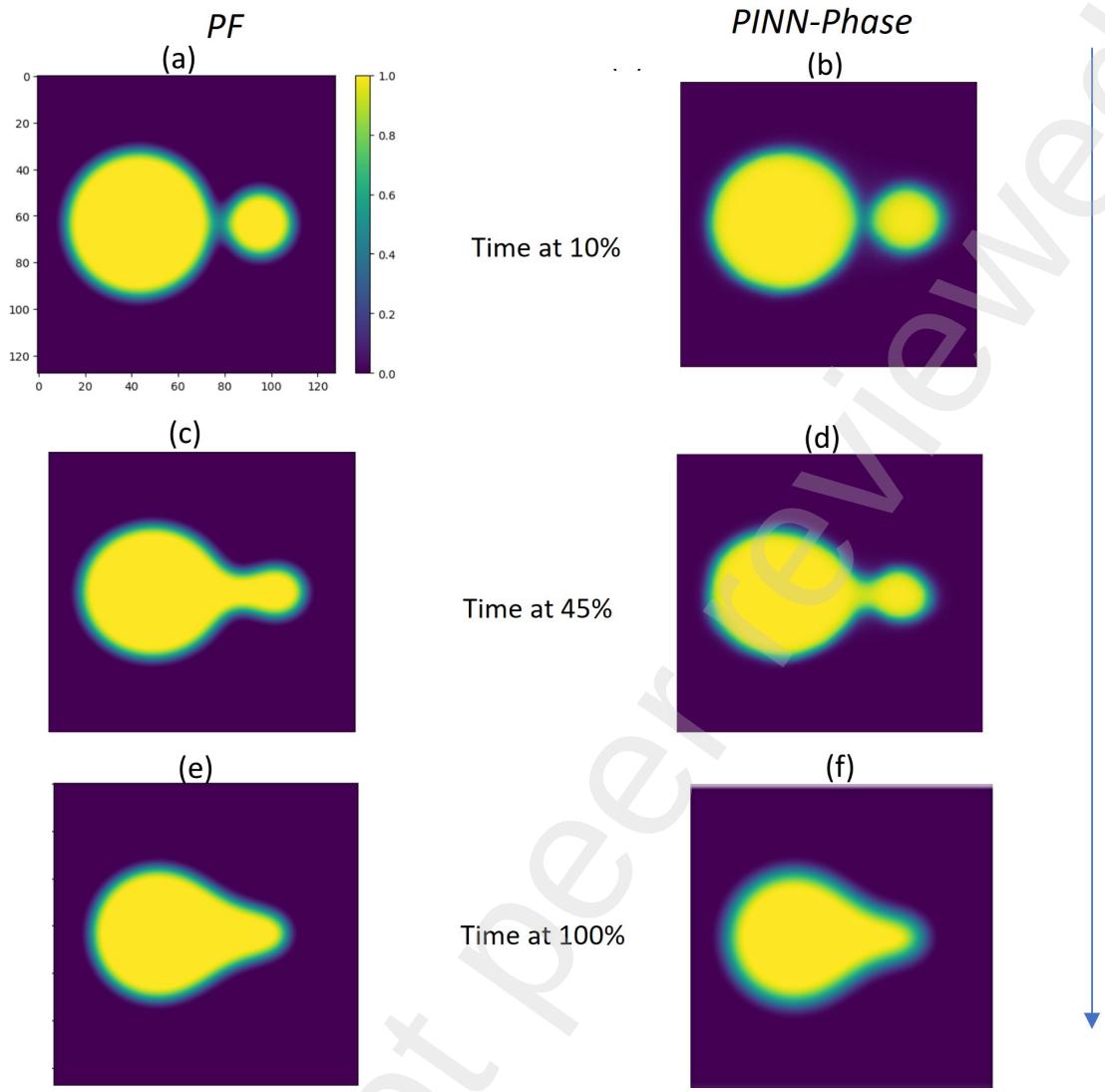
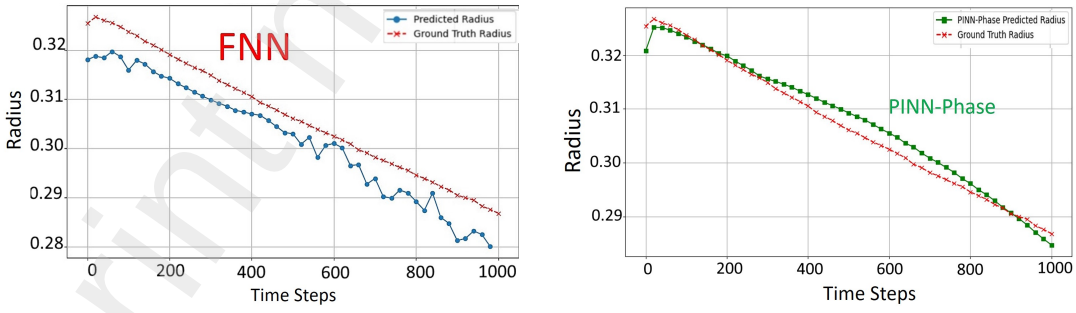


Figure 9: PINN-Phase results of the two-grain system against the ground truth PF solution.



(a) (b)

Figure 10: Comparison of the mean grain radius evolution during the two-grain system.

4.4 Potential upscaling to handle multi-phases

PINN-Phase, through its advanced and flexible configuration, offers an upscaled network that efficiently handles larger grids and a higher number of grains compared to the core element of the PINNs-MPF framework, which is essentially a basic Feedforward Neural Network. Specifically, PINN-Phase

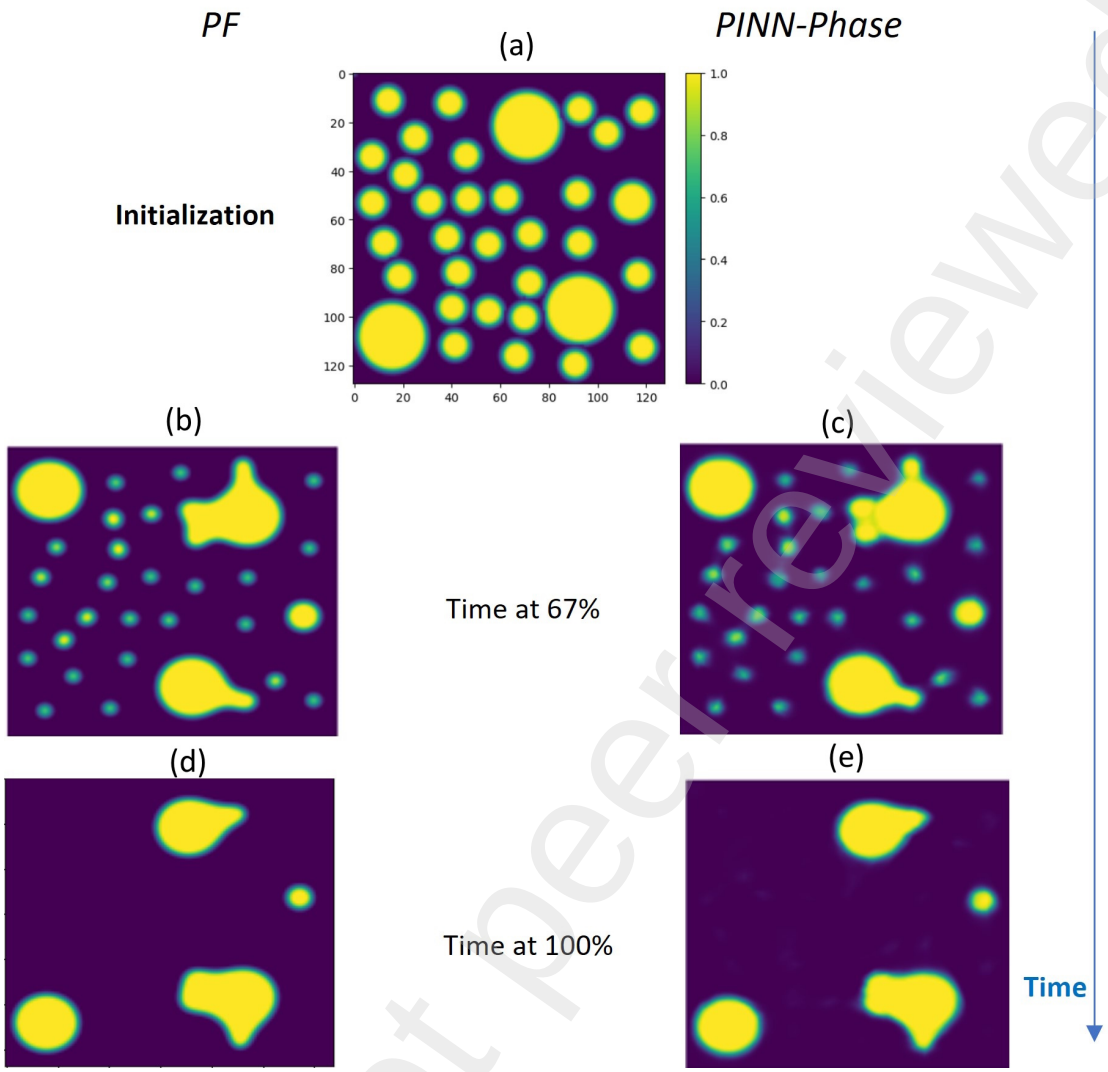
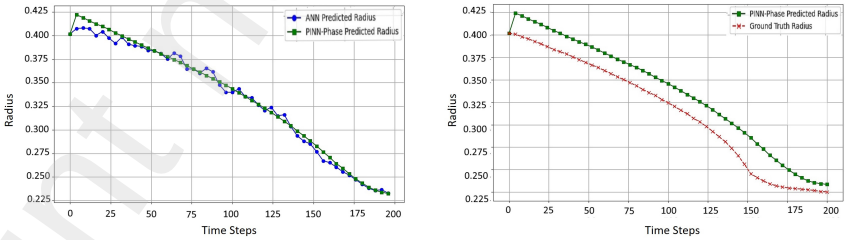


Figure 11: PINNs-Ühase results of the multi-grain system against the ground truth PF solution.



(a) (b)

Figure 12: Comparison of the mean grain radius evolution during the multi-grain growth.

allows the transition from handling a 32×32 grid to a 128×128 grid. Additionally, PINN-Phase is capable of handling multi-phase problems by adjusting the number of channels during convolutional operations and initializing the system with multiple phases.

However, for a more rigorous strategy, we chose to focus PINN-Phase on effectively handling a single phase with a larger spatial resolution, which serves as an enhancement over the core material of PINNs-MPF. Therefore, a potential further upscaling phase could involve merging the two frameworks. This

merging, while technically feasible, is beyond the scope of the current work and is strategically considered as a potential next step.

5 Conclusion

The PINN-Phase framework represents a promising advancement in tackling the complexities of multi-phase-field problems, especially those involving diffuse interfaces. By integrating ConvRNN, FNN, and VAE components, PINN-Phase effectively combines spatial and temporal modeling capabilities with robust handling of non-linearities and variability. This architecture enables a modular and flexible approach, allowing for transfer learning to reduce data requirements and improve computational efficiency—one of the core goals of physics-based machine learning frameworks, particularly in minimizing data dependence.

Modern machine learning frameworks increasingly benefit from combining different architectures, particularly through the use of Large Models, to tackle complex problems. Our framework aligns with this trend by not only acting as a large model but also offering the flexibility to experiment with additional architectures and configurations. For example, transitioning to 3D simulations to study grain growth in polycrystalline materials could benefit from incorporating other architectures, such as Graph Neural Networks, within the residual connections. This would allow GNNs to replace existing models (such as FNN or ConvLSTM) or work in conjunction with them, thanks to the modularity offered by the residual connections.

Moreover, the inclusion of attention mechanisms, widely used in Natural Language Processing, has been adapted to grid spaces within modern machine learning frameworks. This addition could further enhance the model's ability to focus on critical regions within the grid, improving its efficiency and performance when applied to spatially complex problems, such as multi-phase-field dynamics.

Appendices

A Details about the incremental training approach

In this section, we introduce key ideas illustrated in Fig. A.1:

- Implement a gradual/incremental training approach to accommodate the discrete nature of the FNN and the continuous nature of the ConvRNN.
- Train the FNN on small time intervals, resulting in a marching PINN approach similar to [61].
- Use the prediction as the new initial condition once the loss reaches a sufficiently low value. This threshold serves as a hyperparameter, tuned initially for the first time interval and kept constant for subsequent intervals to simplify the process.

In Fig. A.2, we further elucidate the crucial role of the residual connection. Here, the ConvRNN captures both temporal and spatial dependencies, while the FNN excels at modeling highly nonlinear spatial relationships.

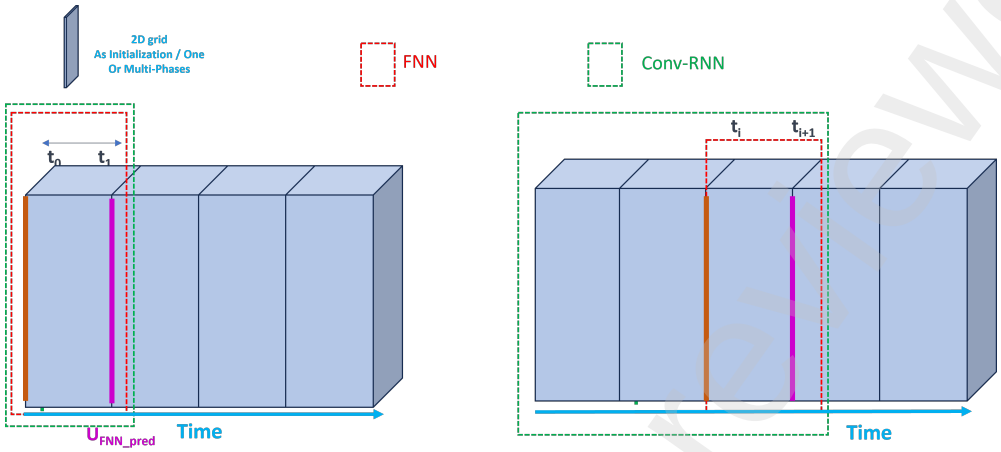


Figure A.1: Gradual/Incremental training approach (discrete FNN while the ConvRNN is continuous).

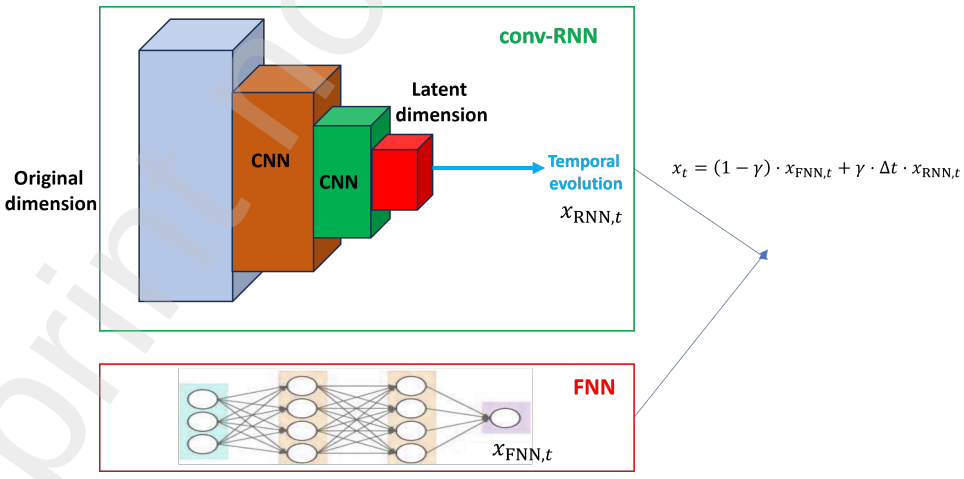


Figure A.2: Residual connection based scheme: the solution at time t can be represented as a combination of a ConvRNN for capturing temporal dynamics and an FNN for refining spatial features.

B Spatial Attention Mechanism / Multi-Head Spatial Attention

The spatial attention mechanism is illustrated in Fig. B.1. We briefly describe the key components of the mechanism, alongside essential formulas. Given a latent grid $\mathbf{x}_t$, which consists of M elements (representing the latent representation: sequence, channels, width, and height), the mechanism operates similarly to the attention mechanism in Transformers [62–64]. First, we compute the Multi-Head Attention (MHSA) mechanism using the query, key, and value (QKV) approach. Specifically, we transform the input data $\mathbf{x}_t$ (a 2D grid snapshot at time t) using a fully connected layer with ReLU activation to obtain self-information, which is represented as $\tilde{x}_t = \text{ReLU}(W_x \cdot \mathbf{x}_t)$. Then, we compute the attention scores by applying the transformation $\mathbf{s}_{i,t} = \tanh(W_s \cdot \tilde{x}_t + W_Q KV \cdot QKV)$, where $\mathbf{s}_{i,t}$ represents the raw attention scores. These attention scores $\alpha_{i,t}$ are computed using a softmax function:

$$\alpha_{i,t} = \frac{\exp(\mathbf{s}_{i,t})}{\sum_{j=1}^{M} \exp(\mathbf{s}_{j,t})}$$

Finally, an attended grid is obtained by applying the attention weights $\alpha_{i,t}$ to the input grid, similar to how the attention mechanism in Transformers generates weighted representations of the input sequence. A comparison of the attention weights computed by the MHSA mechanism at the beginning

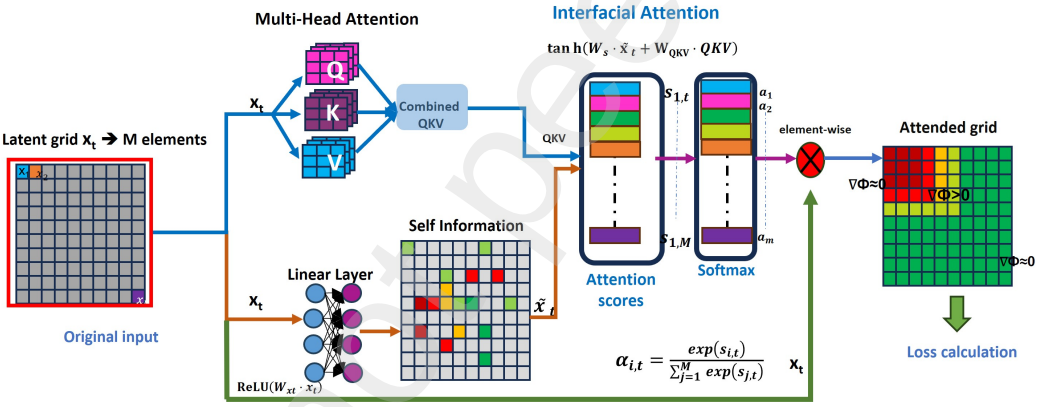
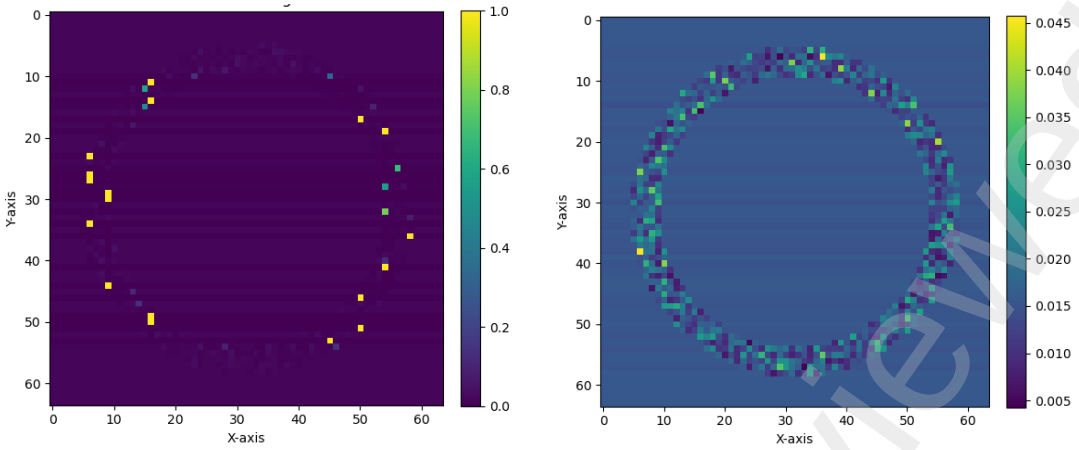


Figure B.1: Graphical illustration of the mechanism of spatial attention (MHSA).

of the training and once the threshold is reached (first benchmark: grain shrinkage) is illustrated in Fig. B.2. From this comparison, we observe that the attention weights are concentrated in the interfacial zones of the grid, which represent the boundaries between the grain and non-grain regions. This concentration of attention in the interfacial zones is a strong indication that the attention mechanism is functioning effectively and/or the model is well capting the interfacial areas. Here, we carefully report this observation.

C Detailed implementation about the GradNorm approach

The GradNorm algorithm is a robust approach for multi-task learning, effectively managing the challenges of training models on different tasks with diverse loss functions. By ensuring that all tasks contribute appropriately to the overall training process, GradNorm can enhance model performance and stability. However, it does introduce additional computational cost, as it requires two back-propagation steps—one for computing the gradient norms of each task and another for the standard



(a) Attention weights at the beginning of training. (b) Attention weights after training convergence.

Figure B.2: Comparison of the attention weights computed by the MHSA mechanism at the beginning of the training and once the threshold is reached (first benchmark: grain shrinkage).

parameter updates. This extra computation adds overhead but is a valuable trade-off for more balanced and efficient multi-task learning.

Algorithm C.1 Standard GradNorm for multi-loss models [58]

Input: Loss components $L_1, L_2, \ldots, L_M$, initial weights $w_1, w_2, \ldots, w_M$, initial loss values $L_{1,0}, L_{2,0}, \ldots, L_{M,0}$, learning rate β, constant α
Output: Updated weights $w_1, w_2, \ldots, w_M$, updated model parameters θ
Initialize weights $w_m = 1.0$ for all $m \in \{1, 2, \ldots, M\}$; Initialize sum of weights $T = \sum_{m=1}^{M} w_m$;
for *each epoch* **do**
  Step 1: First backpropagation to Compute Gradients
  for *each loss component* L_m **do**
    Perform backpropagation to compute the gradient $g_m = \|\nabla_\theta(w_m \cdot L_m)\|_2$ Compute the loss ratio $r_m = \frac{L_m}{L_{m,0}}$
  Compute the average gradient norm $g_{\text{avg}} = \frac{1}{M} \sum_{m=1}^{M} g_m$
  for *each loss component* L_m **do**
    Compute the target gradient norm $t_m = g_{\text{avg}} \cdot \left(\frac{r_m}{\frac{1}{M} \sum_{k=1}^{M} r_k} \right)^{\alpha}$
  Compute the GradNorm loss $L_{\text{gradnorm}} = \sum_{m=1}^{M} |g_m - t_m|$
  for *each weight* w_m **do**
    Update the weight $w_m = w_m \cdot (1 - \beta) + \beta \cdot \left(\frac{t_m}{g_m} \right)$
  Normalize the weights $w_m = \frac{w_m}{\sum_{k=1}^{M} w_k} \cdot T$
  Step 2: Second Backpropagation to Update Model Parameters
  Perform backpropagation on $L_{\text{combined}} = \sum_{m=1}^{M} w_m \cdot L_m$ to update the model parameters θ
.

Acknowledgments

RDK acknowledges financial support from the German Research Foundation (DFG) within projects *DA 1655/2-1* (Heisenberg program) and *DA 1655/5-1*.

Data availability

The PINNs-MPF repository is available on GitHub at the following link:
`https://github.com/SFETNI/PINN_Phase.git`

Competing financial interests

The authors declare no competing financial interests.

References

[1] Seifallah Elfetni and Reza Darvishi Kamachali. "PINNs-MPF: A Physics-Informed Neural Network Framework for Multi-Phase-Field Simulation of Interface Dynamics". In: *Engineering Analysis with Boundary Elements* 176 (2025), p. 106200.

[2] M. Raissi, P. Perdikaris, and G.E. Karniadakis. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". In: *Journal of Computational Physics* 378 (Feb. 2019), pp. 686–707. ISSN: 00219991. DOI: 10.1016/j.jcp.2018.10.045. URL: https://linkinghub.elsevier.com/retrieve/pii/S0021999118307125.

[3] G. P. Purja Pun et al. "Physically informed artificial neural networks for atomistic modeling of materials". In: *Nature communications* 10.1 (2019), pp. 1–10.

[4] Salvatore Cuomo et al. "Scientific Machine Learning Through Physics–Informed Neural Networks: Where we are and What's Next". In: *Journal of Scientific Computing* 92.3 (2022), p. 88. ISSN: 0885-7474. DOI: 10.1007/s10915-022-01939-z.

[5] Yuekun Yang and Youssef Mesri. "Learning by neural networks under physical constraints for simulation in fluid mechanics". In: *Computers Fluids* 248 (2022), p. 105632. ISSN: 0045-7930. DOI: https://doi.org/10.1016/j.compfluid.2022.105632.

[6] S. Hanrahan, M. Kozul, and R.D. Sandberg. "Studying turbulent flows with physics-informed neural networks and sparse data". In: *International Journal of Heat and Fluid Flow* 104 (2023), p. 109232. ISSN: 0142-727X. DOI: https://doi.org/10.1016/j.ijheatfluidflow.2023.109232. URL: https://www.sciencedirect.com/science/article/pii/S0142727X23001315.

[7] Xingyu Chen, Jianhuan Cen, and Qingsong Zou. "Adaptive trajectories sampling for solving PDEs with deep learning methods". In: *Applied Mathematics and Computation* 481 (2024), p. 128928. ISSN: 0096-3003. DOI: https://doi.org/10.1016/j.amc.2024.128928.

[8] P. Ramuhalli, L. Udpa, and S.S. Udpa. "Finite-element neural networks for solving differential equations". In: *IEEE Transactions on Neural Networks* 16.6 (2005), pp. 1381–1392. DOI: 10.1109/TNN.2005.857945.

[9] Yu Diao et al. "Solving multi-material problems in solid mechanics using physics-informed neural networks based on domain decomposition technology". In: *Computer Methods in Applied Mechanics and Engineering* 413 (2023), p. 116120. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2023.116120. URL: https://www.sciencedirect.com/science/article/pii/S004578252300244X.

[10] Alexander Henkes, Henning Wessels, and Rolf Mahnken. "Physics informed neural networks for continuum micromechanics". In: *Computer Methods in Applied Mechanics and Engineering* 393 (2022), p. 114790. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2022.114790. URL: https://www.sciencedirect.com/science/article/pii/S0045782522001268.

[11] Xin-Yu Guo and Sheng-En Fang. "Structural parameter identification using physics-informed neural networks". In: *Measurement* 220 (2023), p. 113334. ISSN: 0263-2241. DOI: https://doi.org/10.1016/j.measurement.2023.113334.

[12] Jinshuai Bai et al. *An introduction to programming Physics-Informed Neural Network-based computational solid mechanics*. 2023. arXiv: 2210.09060 [cs.CE].

[13] Jinshuai Bai et al. "An Introduction to Programming Physics-Informed Neural Network-Based Computational Solid Mechanics". In: *International Journal of Computational Methods* 20.10 (2023), p. 2350013. DOI: 10.1142/S0219876223500135.

[14] Yao Li and Caleb Meredith. "Artificial neural network solver for time-dependent Fokker–Planck equations". In: *Applied Mathematics and Computation* 457 (2023), p. 128185. ISSN: 0096-3003. DOI: https://doi.orFg/10.1016/j.amc.2023.128185.

[15] Somdatta Goswami et al. "Transfer learning enhanced physics informed neural network for phase-field modeling of fracture". In: *Theoretical and Applied Fracture Mechanics* 106 (2020), p. 102447. ISSN: 0167-8442. DOI: https://doi.org/10.1016/j.tafmec.2019.102447. URL: https://www.sciencedirect.com/science/article/pii/S016784421930357X.

[16] M.S. Khorrami et al. "An artificial neural network for surrogate modeling of stress fields in viscoplastic polycrystalline materials". In: *npj Comput Mater* 9 (2023), p. 37. DOI: 10.1038/s41524-023-00991-z.

[17] Pu Ren et al. "PhyCRNet: Physics-informed convolutional-recurrent network for solving spatiotemporal PDEs". In: *Computer Methods in Applied Mechanics and Engineering* 389 (2022), p. 114399. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2021.114399. URL: https://www.sciencedirect.com/science/article/pii/S0045782521006514.

[18] Carlos J.G. Rojas, Jos L. Boldrini, and Marco L. Bittencourt. "Parameter identification for a damage phase field model using a physics-informed neural network". In: *Theoretical and Applied Mechanics Letters* 13.3 (2023), p. 100450. ISSN: 2095-0349. DOI: https://doi.org/10.1016/j.taml.2023.100450.

[19] Wen Zhang and Jian Li. "The robust physics-informed neural networks for a typical fourth-order phase field model". In: *Computers Mathematics with Applications* 140 (2023), pp. 64–77. ISSN: 0898-1221. DOI: https://doi.org/10.1016/j.camwa.2023.03.016.

[20] Ehsan Haghighat, Danial Amini, and Ruben Juanes. "Physics-informed neural network simulation of multiphase poroelasticity using stress-split sequential training". In: *Computer Methods in Applied Mechanics and Engineering* 397 (2022), p. 115141. ISSN: 0045-7825.

[21] Zhao Zhang et al. "A physics-informed convolutional neural network for the simulation and prediction of two-phase Darcy flows in heterogeneous porous media". In: *Journal of Computational Physics* 477 (2023), p. 111919. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2023.111919.

[22] Danial Amini, Ehsan Haghighat, and Ruben Juanes. "Inverse modeling of nonisothermal multiphase poromechanics using physics-informed neural networks". In: *Journal of Computational Physics* 490 (2023), p. 112323. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2023.112323.

[23] Colby L. Wight and Jia Zhao. *Solving Allen-Cahn and Cahn-Hilliard Equations using the Adaptive Physics Informed Neural Networks*. 2020. arXiv: 2007.04542 [math.NA].

[24] Jiang-Xia Han et al. "Physics-informed neural network-based petroleum reservoir simulation with sparse data using domain decomposition". In: *Petroleum Science* 20.6 (2023), pp. 3450–3460. ISSN: 1995-8226. DOI: https://doi.org/10.1016/j.petsci.2023.10.019. URL: https://www.sciencedirect.com/science/article/pii/S1995822623002947.

[25] Dongkun Zhang et al. "Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems". In: *Journal of Computational Physics* 397 (2019), p. 108850. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2019.07.048.

[26] Kamal Choudhary et al. "Recent advances and applications of deep learning methods in materials science". In: *npj Computational Materials* 8.1 (). ISSN: 2057-3960. DOI: 10.1038/s41524-022-00734-6.

[27] Ingo Steinbach. "Phase-field models in materials science". In: *Modelling and Simulation in Materials Science and Engineering* 17.7 (2009). ISSN: 09650393. DOI: 10.1088/0965-0393/17/7/073001.

[28] Ingo Steinbach and Markus Apel. "Multi phase field model for solid state transformation with elastic strain". In: *Physica D: Nonlinear Phenomena* 217 (2006), pp. 153–160.

[29] Long-Qing Chen. "Phase-field models for microstructure evolution". In: *Annual Review of Materials Research* 32 (2002), pp. 113–140.

[30] Reza Darvishi Kamachali. "Grain boundary motion in polycrystalline materials". PhD thesis. Univ.-Bibliothek, 2013.

[31] Reza Darvishi Kamachali and Ingo Steinbach. "3-D phase-field simulation of grain growth: topological analysis versus mean-field approximations". In: *Acta Materialia* 60.6-7 (2012), pp. 2719–2728.

[32] Reza Darvishi Kamachali et al. "Geometrical grounds of mean field solutions for normal grain growth". In: *Acta Materialia* 90 (2015), pp. 252–258.

[33] Eisuke Miyoshi and Tomohiro Takaki. "Multi-phase-field study of the effects of anisotropic grain-boundary properties on polycrystalline grain growth". In: *Journal of Crystal Growth* 474 (2017), pp. 160–165.

[34] Reza Darvishi Kamachali et al. "Multiscale simulations on the grain growth process in nanostructured materials: paper presented at the 2nd Sino-German symposium on computational thermodynamics and kinetics and their applications to solidification". In: *International journal of materials research* 101.11 (2010), pp. 1332–1338.

[35] Christian Schwarze, Reza Darvishi Kamachali, and Ingo Steinbach. "Phase-field study of zener drag and pinning of cylindrical particles in polycrystalline materials". In: *Acta Materialia* 106 (2016), pp. 59–65.

[36] Reza Darvishi Kamachali, Se-Jong Kim, and Ingo Steinbach. "Texture evolution in deformed AZ31 magnesium sheets: Experiments and phase-field study". In: *Computational Materials Science* 104 (2015), pp. 193–199.

[37] Y Rezaei et al. "Multi-phase-field modeling of grain growth in polycrystalline titanium under magnetic field and elastic strain". In: *Applied Physics A* 128.10 (2022), p. 874.

[38] Revanth Mattey and Susanta Ghosh. "A novel sequential method to train physics informed neural networks for Allen Cahn and Cahn Hilliard equations". In: *Computer Methods in Applied Mechanics and Engineering* 390 (2022), p. 114474. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2021.114474.

[39] Ameya D. Jagtap and George Em Karniadakis. "Extended Physics-Informed Neural Networks (XPINNs): A Generalized Space-Time Domain Decomposition Based Deep Learning Framework for Nonlinear Partial Differential Equations". In: *Communications in Computational Physics* 28.5 (2020), pp. 2002–2041. DOI: https://doi.org/10.4208/cicp.OA-2020-0164.

[40] Ehsan Kharazmi, Zhongqiang Zhang, and George E.M. Karniadakis. "hp-VPINNs: Variational physics-informed neural networks with domain decomposition". In: *Computer Methods in Applied Mechanics and Engineering* 374 (2021), p. 113547. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2020.113547.

[41] Michelle Tindall Prakhar Sharma Llion Evans and Perumal Nithiarasu. "Hyperparameter selection for physics-informed neural networks (PINNs) – Application to discontinuous heat conduction problems". In: *Numerical Heat Transfer, Part B: Fundamentals* 0.0 (2023), pp. 1–15. DOI: 10.1080/10407790.2023.2264489.

[42] Xuhui Meng et al. "PPINN: Parareal physics-informed neural network for time-dependent PDEs". In: *Computer Methods in Applied Mechanics and Engineering* 370 (2020), p. 113250. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2020.113250. URL: https://www.sciencedirect.com/science/article/pii/S0045782520304357.

[43] Darioush Jalili et al. "Transfer learning through physics-informed neural networks for bubble growth in superheated liquid domains". In: *International Journal of Heat and Mass Transfer* 232 (2024), p. 125940. ISSN: 0017-9310. DOI: https://doi.org/10.1016/j.ijheatmasstransfer.2024.125940.

[44] Konstantinos Prantikos, Stylianos Chatzidakis, Lefteri H. Tsoukalas, et al. "Physics-informed neural network with transfer learning (TL-PINN) based on domain similarity measure for prediction of nuclear reactor transients". In: *Scientific Reports* 13 (2023), p. 16840. DOI: 10.1038/s41598-023-43325-1.

[45] Michael Penwarden et al. "A unified scalable framework for causal sweeping strategies for Physics-Informed Neural Networks (PINNs) and their temporal decompositions". In: *Journal of Computational Physics* 493 (2023), p. 112464. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2023.112464.

[46] Jia Guo, Haifeng Wang, and Chenping Hou. "An adaptive energy-based sequential method for training PINNs to solve gradient flow equations". In: *Applied Mathematics and Computation* 479 (2024), p. 128890. ISSN: 0096-3003. DOI: https://doi.org/10.1016/j.amc.2024.128890. URL: https://www.sciencedirect.com/science/article/pii/S0096300324003515.

[47] S Divya Meena et al. "Hybrid Neural Network Architecture for Multi-Label Object Recognition using Feature Fusion". In: *Procedia Computer Science* 215 (2022). 4th International Conference on Innovative Data Communication Technology and Application, pp. 78–90. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2022.12.009. URL: https://www.sciencedirect.com/science/article/pii/S1877050922020804.

[48] Van-Nui Nguyen et al. "Using a hybrid neural network architecture for DNA sequence representation: A study on N4-methylcytosine sites". In: *Computers in Biology and Medicine* 178 (2024), p. 108664. ISSN: 0010-4825. DOI: https://doi.org/10.1016/j.compbiomed.2024.108664. URL: https://www.sciencedirect.com/science/article/pii/S0010482524007492.

[49] K. Cao, T. Zhang, and J. Huang. "Advanced hybrid LSTM-transformer architecture for real-time multi-task prediction in engineering systems". In: *Scientific Reports* 14 (2024), p. 4890. DOI: 10.1038/s41598-024-55483-x.

[50] Koldo Portal-Porras et al. "Hybrid LSTM+CNN architecture for unsteady flow prediction". In: *Materials Today Communications* 35 (2023), p. 106281. ISSN: 2352-4928. DOI: https://doi.org/10.1016/j.mtcomm.2023.106281. URL: https://www.sciencedirect.com/science/article/pii/S2352492823009728.

[51] Po-Chih Kuo et al. "GNN-LSTM-based fusion model for structural dynamic responses prediction". In: *Engineering Structures* 306 (2024), p. 117733. ISSN: 0141-0296. DOI: https://doi.org/10.1016/j.engstruct.2024.117733. URL: https://www.sciencedirect.com/science/article/pii/S0141029624002955.

[52] Luo Hao et al. "Hybrid Graph Neural Networks with LSTM Attention Mechanism for Recommendation Systems in MOOCs". In: *2024 20th IEEE International Colloquium on Signal Processing Its Applications (CSPA)*. 2024, pp. 63–68. DOI: 10.1109/CSPA60979.2024.10525319.

[53] Francisco Guillén-González and Giordano Tierra. "Second order schemes and time-step adaptivity for Allen–Cahn and Cahn–Hilliard models". In: *Computers Mathematics with Applications* 68.8 (2014), pp. 821–846. ISSN: 0898-1221. DOI: https://doi.org/10.1016/j.camwa.2014.07.014.

[54] Jingying Wang et al. "A high order accurate numerical algorithm for the space-fractional Swift-Hohenberg equation". In: *Computers Mathematics with Applications* 139 (2023), pp. 216–223. ISSN: 0898-1221. DOI: https://doi.org/10.1016/j.camwa.2022.09.014.

[55] Xuming Ran et al. "Detecting out-of-distribution samples via variational auto-encoder with reliable uncertainty estimation". In: *Neural Networks* 145 (2022), pp. 199–208. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2021.10.020. URL: https://www.sciencedirect.com/science/article/pii/S0893608021004111.

[56] Zijing Luo, Yihui Xiong, and Renguang Zuo. "Recognition of geochemical anomalies using a deep variational autoencoder network". In: *Applied Geochemistry* 122 (2020), p. 104710. ISSN: 0883-2927. DOI: https://doi.org/10.1016/j.apgeochem.2020.104710. URL: https://www.sciencedirect.com/science/article/pii/S088329272030202X.

[57] Xirui Ma et al. "Structural damage identification based on unsupervised feature-extraction via Variational Auto-encoder". In: *Measurement* 160 (2020), p. 107811. ISSN: 0263-2241. DOI: https://doi.org/10.1016/j.measurement.2020.107811. URL: https://www.sciencedirect.com/science/article/pii/S0263224120303493.

[58] Zhao Chen et al. *GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks*. 2018. arXiv: 1711.02257 [cs.CV]. URL: https://arxiv.org/abs/1711.02257.

[59] Stefano Bini et al. "A multi-task network for speaker and command recognition in industrial environments". In: *Pattern Recognition Letters* 176 (2023), pp. 62–68. ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2023.10.022.

[60] Nanxi Chen et al. "Pf-Pinns: Physics-Informed Neural Networks for Solving Coupled Allen-Cahn and Cahn-Hilliard Phase Field Equations". In: (2023). Available at SSRN: https://ssrn.com/abstract=4761824 or http://dx.doi.org/10.2139/ssrn.4761824.

[61] Zhaolin Chen, Siu-Kai Lai, and Zhichun Yang. "AT-PINN: Advanced time-marching physics-informed neural network for structural vibration analysis". In: *Thin-Walled Structures* 196 (2024), p. 111423. ISSN: 0263-8231. DOI: https://doi.org/10.1016/j.tws.2023.111423.

[62] Ashish Vaswani et al. "Attention Is All You Need". In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[63] Haixin Lv et al. "Attention mechanism in intelligent fault diagnosis of machinery: A review of technique and application". In: *Measurement* 199 (2022), p. 111594. ISSN: 0263-2241. DOI: https://doi.org/10.1016/j.measurement.2022.111594.

[64] Zhuo Long et al. "Motor fault diagnosis using attention mechanism and improved adaboost driven by multi-sensor information". In: *Measurement* 170 (2021), p. 108718. ISSN: 0263-2241. DOI: https://doi.org/10.1016/j.measurement.2020.108718.