Journal of Scientific Computing (2024) 101:71
https://doi.org/10.1007/s10915-024-02709-9

Generative Downscaling of PDE Solvers with Physics-Guided Diffusion Models

Yulong Lu[1] · Wuzhe Xu[2]

Received: 18 April 2024 / Revised: 31 August 2024 / Accepted: 16 October 2024 /
Published online: 5 November 2024

Abstract
Solving partial differential equations (PDEs) on fine spatio-temporal scales for high-fidelity solutions is critical for numerous scientific breakthroughs. Yet, this process can be prohibitively expensive, owing to the inherent complexities of the problems, including nonlinearity and multiscale phenomena. To speed up large-scale computations, a process known as downscaling is employed, which generates high-fidelity approximate solutions from their low-fidelity counterparts. In this paper, we propose a novel Physics-Guided Diffusion Model (PGDM) for downscaling. Our model, initially trained on a dataset comprising low-and-high-fidelity paired solutions across coarse and fine scales, generates new high-fidelity approximations from any new low-fidelity inputs. These outputs are subsequently refined through fine-tuning, aimed at minimizing the physical discrepancies as defined by the discretized PDEs at the finer scale. We evaluate and benchmark our model's performance against other downscaling baselines in three categories of nonlinear PDEs. Our numerical experiments demonstrate that our model not only outperforms the baselines but also achieves a computational acceleration exceeding tenfold, while maintaining the same level of accuracy as the conventional fine-scale solvers.

Keywords Diffusion model · Data driven model · Optimization · Numerical PDEs

Mathematics Subject Classification 65C20 · 65N21 · 65M22 · 65K10

1 Introduction

Numerical simulation of PDEs play an indispensable role in science and engineering. Traditional numerical methods, such as finite difference method and finite element method, often become computationally intensive with an increase in mesh grids. This increase is typically

✉ Wuzhe Xu
wuzhexu@umass.edu

Yulong Lu
yulonglu@umn.edu

1 School of Mathematics, University of Minnesota Twin Cities, Minneapolis, MN 55414, USA

2 Department of Mathematics and Statistics, University of Massachusetts Amherst, Amherst, MA 01003, USA

necessary to accurately resolve PDEs, given their complexities, such as nonlinearity, scale-separation stiffness, and high dimensionality. In recent times, the adoption of deep learning techniques to develop more efficient numerical methods has seen a significant rise in popularity. Numerous studies have explored the direct approximation of solutions using neural networks. The work [32] proposed the physics-informed neural networks (PINNs) that minimizes the L^2-loss associated with the governing physics, and it has proven to be highly efficient in addressing various complex PDE problems, such as the fluid dynamic [5], inverse problem [49] and multiscale problem [18, 29]. Furthermore, it is worth mentioning several variations and enhancements of PINNs, such as [20, 21, 42]. Many other approaches have also been developed, such as Deep Ritz Method [26, 28, 45], based on the variational (or Ritz) formulation of PDEs, the deep BSDE method [12, 13] for certain class of parabolic PDEs based on their probabilistic and control formulation, and the weak adversarial networks [50] based on the Galerkin (or weak) formulation. Additionally, deep learning has been leveraged to expedite classical iterative solvers [2, 3, 6, 16, 30, 38], showcasing its versatility and potential in enhancing computational efficiency across various PDE-solving methodologies.

In many scientific domains, there's a notable interest in discerning mappings or operators between infinite-dimensional function spaces. Recent advancements have seen neural networks being harnessed to approximate these solution operators. Among the leading neural operator models are the Deep operator networks (DeepONet) [27, 41] and the Fourier neural operator (FNO) [9, 24, 25]. Nevertheless, these standard neural operators often necessitate a substantial training dataset composed of numerous parameter-solution pairs, posing challenges in scenarios where solution labels are costly to obtain. To circumvent this issue, physics-informed neural operators [10, 43] have been introduced, merging PDE constraints with operator learning by embedding known differential equations directly into the training loss function. Notably, neural operators have found applications in downscaling climate data [17, 48] and enhancing super-resolution in imaging [44], as well as general PDE problems [22], showcasing their versatility. The method we propose in this paper will also be compared with the neural operator baselines.

In this paper, we focus on accelerating the computation of PDEs from a downscaling viewpoint. In climate modeling and simulation, downscaling [46] refers to a class of methods that generate high-fidelity climate data out of their low-fidelity counterpart. Similar processes may carry with different names. For instance in the community of imaging and computer vision, this process is named super-resolution. Such a downscaling/super-resolution is appealing because low-fidelity solutions can be generated via solving PDEs on coarse-grids which is computationally much cheaper compared to the high-fidelity solutions. Classical downscaling techniques in climate science and meteorology have ranged from pointwise regression [34, 39] to super-resolution [40] and maximum likelihood estimation [4]. Recent initiatives have seen Fourier neural operators being applied for downscaling [48], effectively bridging fast low-resolution simulations to high-resolution climate outputs. Moreover, the use of deep generative models for climate data downscaling, inspired by their success in computer vision for super-resolution, has gained traction. In [23, 31], Generative Adversarial Networks (GANs) were adopted for downscale precipitation forecasting. In [11], the authors proposed a ClimAlign approach to downscaling with normalizing flows. Recently, diffusion models have demonstrated their ability to produce high quality samples, beating many competing generative models such as GANs in numerous machine learning problems [7]. Specifically, diffusion-models [14, 36, 37] are capable of generating high-fidelity (super-resolution) images [15]. Inspired by their great success in machine learning tasks, we propose to deploy diffusion models for downscaling PDE solvers. Unlike the purely data-driven nature of downscaling in the aforementioned applications, our PDE-focused generative downscaling necessitates adherence to the physical laws governing the PDE model, integrating a unique challenge to this innovative approach.

1.1 Related Work

The utilization of deep generative models for solving PDEs is not new. Initial studies, such as those by [8, 19], harnessed adversarial generative models to tackle PDEs by integrating a physics-informed loss with a uniquely tailored adversarial loss. More recent efforts, like the study by [47], showcased the diffusion model's efficacy as an alternative solution operator, mapping initial conditions to solutions at subsequent times, and displaying competitive prowess alongside other neural operator approaches. To overcome the hurdle of scarce data in operator learning, [1] utilized the diffusion model for the creation of synthetic data samples, thereby enriching the training dataset. The research work [35] is most relevant to ours, where the authors developed a physics-informed diffusion model designed to accurately reconstruct high-fidelity samples from low-fidelity ones. While both our approach and the method proposed in [35] intend to reconstruct high-fidelity data from low-fidelity sources, our approach differs from [35] in several aspects. First, they added physics-informed loss in the denosing score-matching loss in the training of their diffusion models, which can be very expensive due to the enforcement of PDE information in each gradient step during the training. In contrast, our physics-guided diffusion model decouples the step of purely data-driven conditional diffusion model from the physics-enhancement step. The pre-trained model when combined with low-fidelity input produces a high-fidelity output that can be used as a good warm-start for minimizing the physics-informed loss in the second step. This two-step procedure improves substantially the efficiency of training and accuracy of generated solutions.

1.2 Our Contributions

We introduce the physics-guided diffusion model as a universal framework for downscaling PDE solutions from low-resolution to high-resolution.

- We first reformulate the downscaling problem as a conditional sampling task, where the objective is to sample from the posterior distribution of unknown high-fidelity solutions, given any arbitrary low-fidelity input. This reformulation allows for a more targeted and accurate generation of high-resolution outputs from their lower-resolution counterparts.
- The first step involves conditional sampling via a diffusion model to produce preliminary high-fidelity samples. Subsequently, these samples are refined through a physics-informed loss minimization step, ensuring they adhere to the physical laws governing the PDEs. This dual-step approach effectively merges data-driven sample generation with physics-based accuracy enhancement.
- The proposed method consistently outperforms several existing downscaling baselines in a range of nonlinear static and time-dependent PDEs. Remarkably, it not only matches the accuracy of traditional high-fidelity solvers at the fine scale but also achieves this with a significant reduction in computational expenditure, cutting costs by more than tenfold.

2 Problem Set-Up

2.1 Problem Description

Our primary focus is on developing efficient approximations for the solution u to a generic PDE, subject to appropriate boundary conditions, as outlined below:

$$\mathcal{L}u = a, \tag{1}$$

where $\mathcal{L}$ is the (possibly nonlinear) differential operator and a is the source term. Traditional grid-based PDE solvers typically approach this by solving a discretized version of the problem:

$$\mathcal{L}_h \boldsymbol{u}_h = \boldsymbol{a}_h + \boldsymbol{\epsilon}_h.$$

Here h represents the spatial-temporal grid-size, $\boldsymbol{\epsilon}_h$ denotes certain (unknown) noise that potentially encapsulating errors in the pointwise evaluations of the functions, and $\mathcal{L}_h$, $\boldsymbol{u}_h$ and $\boldsymbol{a}_h$ represent the discrete approximations to the operator $\mathcal{L}$, the solution u and the source a respectively. Discretization on fine grids (characterized by a small h) usually results in high-fidelity (high-resolution) solutions but at the expense of significantly increased computational costs. Therefore, finding an optimal cost-accuracy balance is crucial. Downscaling, in this context, refers to a series of techniques that first solve PDEs on coarse grids and subsequently convert the low-fidelity solutions obtained on these coarse grids to their high-fidelity equivalents on fine grids, offering a strategic approach to manage the trade-offs between computational expense and solution accuracy.

2.2 Downscaling as Conditional Sampling

To describe our diffusion-based downscaling approach, we would like to first present a conditional-sampling formulation to the downscaling problem. To ease the notation, we will suppress the dependence of quantities on the grid size h and denote by $\boldsymbol{u}^c$ and $\boldsymbol{u}^f$ the low-fidelity solution and high-fidelity solution respectively. Similarly one can define for $g = c, f$ the operators $\mathcal{L}^g$, the source terms $\boldsymbol{a}^g$ and the noise $\boldsymbol{\epsilon}^g$. Moreover, we have

$$\mathcal{L}^g \boldsymbol{u}^g = \boldsymbol{a}^g + \boldsymbol{\epsilon}^g.$$

Assume that $a^c = \mathcal{R}a^f$ with some fine-to-coarse restriction operator $\mathcal{R}$, one has

$$\mathcal{R}\mathcal{L}^f \boldsymbol{u}^f = \mathcal{L}^c \boldsymbol{u}^c + \mathcal{R}\boldsymbol{\epsilon}^f - \boldsymbol{\epsilon}^c.$$

Assuming the invertibility of $\mathcal{L}^c$, we can rewrite above as

$$\boldsymbol{u}^c = (\mathcal{L}^c)^{-1}\mathcal{R}\mathcal{L}^f \boldsymbol{u}^f + \boldsymbol{\epsilon}, \text{ where } \boldsymbol{\epsilon} := -(\mathcal{L}^c)^{-1}(\mathcal{R}\boldsymbol{\epsilon}^f - \boldsymbol{\epsilon}^c).$$

In another words, the downscaling problem is an inverse problem of recovering $\boldsymbol{u}^f$ from the noisy downscaling observation $\boldsymbol{u}^c$ via

$$\boldsymbol{u}^c = \mathcal{G}\boldsymbol{u}^f + \boldsymbol{\epsilon}, \text{ where } \mathcal{G} := (\mathcal{L}^c)^{-1}\mathcal{R}\mathcal{L}^f.$$

We adopt the Bayesian approach for solving the inverse problem. Given a prior $p(\boldsymbol{u}^f)$ on the set of fine solutions, one can define by the Bayes' rule the posterior distribution

$$p(\boldsymbol{u}^f|\boldsymbol{u}^c) \propto p(\boldsymbol{u}^c|\boldsymbol{u}^f) \times p(\boldsymbol{u}^f), \tag{2}$$

where $p(\boldsymbol{u}^c|\boldsymbol{u}^f)$ is the likelihood function. With above, we have recast the downscaling as the problem of conditional sampling from the Bayesian posterior $p(\boldsymbol{u}^f|\boldsymbol{u}^c)$ given an arbitrary low-fidelity input $\boldsymbol{u}^c$.

2.3 Challenges and Our Approach

Despite the appealing conditional sampling framework offered by downscaling, direct sampling from the Bayesian posterior (2) presents infeasibility and numerous challenges. First, the prior $p(\boldsymbol{u}^f)$ is unknown and needs to be learned from the data.

Secondly, evaluating the likelihood function poses significant difficulties due to two primary reasons: (1) the forward map $\mathcal{G}$ is either inaccessible, owing to an unknown fine-to-coarse restriction or, even if known, the computation may be prohibitively expensive, and (2) the noise distribution is typically unknown, leading to an intractable likelihood. Last but not the least, the conditional samples $\boldsymbol{u}^f$ given $\boldsymbol{u}^c$ even can be generated may not fulfil the discrete PDE problem on the fine grid, especially given a limited amount of data. To address these issues, we introduce a physics-guided diffusion model designed to learn and draw physics-conformal high-fidelity samples from any low-fidelity inputs. Our approach is methodically divided into two pivotal steps:

(1) **Pre-training step**: We pre-train a conditional diffusion model using a finite collection of low-and-high fidelity solution pairs $\{(\boldsymbol{u}_i^c, \boldsymbol{u}_i^f)\}_{i=1}^n$ laying the groundwork for subsequent refinements.
(2) **Refining step**: Upon receiving any new low-fidelity input, we refine the output via the pre-trained model to ensure an enhanced fit with the fine-grid PDE, thereby further improving the solution's fidelity.

The pre-training phase of our approach is primarily data-driven and accounts for the majority of computational expenditure. In contrast, the refining step is more computationally economical and aims to enhance the high-fidelity output by minimizing the physics misfit loss. This enhancement could be achieved, for instance, by executing few, such as two, Gauss-Newton iterations, starting with the initial output from the pre-trained model, thereby streamlining the process towards achieving superior solution accuracy with reduced computational demand.

3 Methodology

3.1 Unconditioned Diffusion Model

To introduce our conditional diffusion models for downscaling, we start with a general overview of unconditioned diffusion models, with our focus on the Denoising Diffusion Probabilistic Models and one of its accelerated version called Denoising Diffusion Implicit Models.

3.1.1 Denoising Diffusion Probabilistic Models(DDPM)

Let $q(\boldsymbol{x})$ represent the target data distribution. DDPM constructs a Markovian noising process that incrementally contaminates the data $\boldsymbol{x}_0$ with Gaussian noise over

T steps, ultimately transforming it into pure Gaussian noise. This noising process is denoted by $q(\boldsymbol{x}_{0:T})$, where the $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_T$ are progressively noised versions of the data, all maintaining the same dimensionality as $\boldsymbol{x}_0$, and $q(\boldsymbol{x}_T)$ is approximately an isotropic Gaussian distribution. This forward process of $\boldsymbol{x}_{0:T}$ can be described by the Markov process with the transition kernel defined by

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) := \mathcal{N}(\boldsymbol{x}_t; \sqrt{\alpha_t}\boldsymbol{x}_{t-1}, (1-\alpha_t)\mathbf{I}) \tag{3}$$

where $\{\alpha_t\}_{t=0}^T \subset (0, 1)$ is a sequence of user designed parameters. Since the noises we add in each step are Gaussian, we have

$$\boldsymbol{x}_t|\boldsymbol{x}_0 \overset{d}{=} \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \boldsymbol{\epsilon}\sqrt{1-\bar{\alpha}_t}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}), \tag{4}$$

where $\bar{\alpha}_t = \Pi_{t=1}\alpha_t$. Generation of new data samples can be done via the backward process in DDPM. More precisely, with the assumption that the reverse process $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ can be modeled as Gaussians with trainable mean and fixed variance, a reversed Markov process is parameterized in the form of

$$p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) := \mathcal{N}\Big(\boldsymbol{x}_{t-1}; \frac{1}{\sqrt{\alpha_t}}\Big(\boldsymbol{x}_t + \frac{\sqrt{(1-\alpha_t)}}{\sqrt{(1-\bar{\alpha}_t)}}\boldsymbol{s}_\theta(\boldsymbol{x}_t, t)\Big), \sigma_t^2\mathbf{I}\Big), \tag{5}$$

where $\sigma_t^2 = \frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}$ and is trained with the weighted evidence lower bound (ELBO)

$$\hat{\theta} = \arg\min_\theta \sum_{t=1}^T (1-\bar{\alpha}_t)\mathbf{E}_{q(\boldsymbol{x}_0)}\mathbf{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left\|\frac{1}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{s}_\theta(\boldsymbol{x}_t, t) - \nabla_x \log(p(\boldsymbol{x}_t|\boldsymbol{x}_0))\right\|^2.$$

It can be shown further by integration by parts that minimizing the ELBO is equivalent to the denoising problem

$$\hat{\theta} = \arg\min_\theta \sum_{t=1}^T \mathbf{E}_{\boldsymbol{x}_0\sim q(\boldsymbol{x}_0)}\mathbf{E}_{\boldsymbol{\epsilon}_t\sim\mathcal{N}(0,\mathbf{I})}\left\|\frac{(1-\alpha_t)^2}{2\sigma_t^2\alpha_t(1-\bar{\alpha}_t)}\boldsymbol{s}_\theta(\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \boldsymbol{\epsilon}_t\sqrt{1-\bar{\alpha}_t}, t) + \boldsymbol{\epsilon}_t\right\|^2.$$

The optimized neural network $\boldsymbol{s}_{\hat{\theta}}$ enables us to generate new samples $\boldsymbol{x}_0$ through the backward process iterates: starting with $\boldsymbol{x}_T \sim \mathcal{N}(0, \mathbf{I})$,

$$\boldsymbol{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\Big(\boldsymbol{x}_t + \frac{\sqrt{(1-\alpha_t)}}{\sqrt{(1-\bar{\alpha}_t)}}\boldsymbol{s}_\theta(\boldsymbol{x}_t, t)\Big) + \sigma_t\xi_t, \quad t = T, T-1, \cdots, 1,$$

where $\{\xi_t\}_{t=1}^T \overset{i.i.d.}{\sim} \mathcal{N}(0, \mathbf{I})$.

3.1.2 Accelerating Sampling with Denoising Diffusion Implicit Models (DDIM)

One major drawback of DDPM is that generating a new sample from the data distribution requires simulating the whole Markov backward process for many (typically hundred or thousand) steps (or equivalently the forward network passes), which can be computationally intensive and time-consuming. Recently, Song et. al. [36] proposed the denoising diffusion implicit models (DDIM), accelerating the generative process by using a non-Markovian deterministic diffusion pathway, culminating in implicit models capable of producing samples at an faster pace without compromising on quality. More specifically, given a selective increasing sequence of length L, denoted by $\{\tau_i\}_{i=1}^L \subset [1, 2, \cdots, T]$, DDIM generates a sample $\boldsymbol{x}_{\tau_{i-1}}$ from $\boldsymbol{x}_{\tau_i}$ by making the following update:

$$\boldsymbol{x}_{\tau_{i-1}} = \frac{\sqrt{\bar{\alpha}_{\tau_{i-1}}}}{\sqrt{\bar{\alpha}_{\tau_i}}}\boldsymbol{x}_{\tau_i} + \Big(\frac{\sqrt{1-\bar{\alpha}_{\tau_i}}\sqrt{\bar{\alpha}_{\tau_{i-1}}}}{\sqrt{\bar{\alpha}_{\tau_i}}} - \sqrt{1-\bar{\alpha}_{\tau_{i-1}}}\Big)\boldsymbol{s}_{\hat{\theta}}\left(\mathbf{x}_{\tau_i}, \tau_i\right), \quad i = 1, 2, \cdots, L,$$

where $s_{\hat{\theta}}$ is the optimal score network trained in the same manner as in DDPM.

3.2 Conditioned Diffusion Model

Now let us move on to the problem of conditional sampling using conditional diffusion model. Recall that our goal is to sample from the posterior distribution

$$p(\boldsymbol{u}^f|\boldsymbol{u}^c) \propto p(\boldsymbol{u}^c|\boldsymbol{u}^f) \times p(\boldsymbol{u}^f)$$

for any given low-fidelity solution $\boldsymbol{u}^c$. One straightforward idea for doing so would be to train a conditional score network $s_\theta(\boldsymbol{u}_t^f, \boldsymbol{u}^c, t)$ that minimizes the ELBO in the conditional setting:

$$\hat{\theta} = \arg\min_\theta \sum_{t=1}^{T}(1-\bar{\alpha}_t)\mathbf{E}_{\boldsymbol{u}_0^f \sim p(\boldsymbol{u}^f)}\mathbf{E}_{p(\boldsymbol{u}_t^f|\boldsymbol{u}_0^f)}\left\| \frac{1}{\sqrt{1-\bar{\alpha}_t}} s_\theta(\boldsymbol{u}_t^f, \boldsymbol{u}^c, t) - \nabla_{\boldsymbol{u}_t^f} \log(p(\boldsymbol{u}_t^f|\boldsymbol{u}_0^f, \boldsymbol{u}^c))\right\|^2.$$

By the Bayes' formula, the true conditional score function $\nabla_{\boldsymbol{u}_t^f} \log(p(\boldsymbol{u}_t^f|\boldsymbol{u}_0^f, \boldsymbol{u}^c))$ can be written as

$$\nabla_{\boldsymbol{u}_t^f} \log(p(\boldsymbol{u}_t^f|\boldsymbol{u}_0^f, \boldsymbol{u}^c)) = \nabla_{\boldsymbol{u}_t^f} \log p(\boldsymbol{u}_t^f) + \nabla_{\boldsymbol{u}_t^f} \log(p(\boldsymbol{u}^c|\boldsymbol{u}_0^f, \boldsymbol{u}_t^f)),$$

where the first term represents the score function corresponding to the prior $p(\boldsymbol{u}^f)$ and the second term encodes the conditional likelihood. While the prior $p(\boldsymbol{u}^f)$ can be learned from high-fidelity training samples, the conditional likelihood is often computationally intractable and existing conditional diffusion models resort to various approximations to the conditional likelihood, such as the pseudo-inverse in the inverse problem setting or the posterior mean in the general nonlinear inverse problem setting. Unfortunately, it is impossible to construct those approximations in our setting due to the lack of the complete knowledge on the forward operator $\boldsymbol{\mathcal{G}}$ as we illustrated in Sect. 2.3. To bypass these issues, we seek a purely data-driven approach to learn the conditional score without incorporating the forward model. To be concrete, given a training set of low-and-high fidelity solution pairs $\{(\boldsymbol{u}_k^c, \boldsymbol{u}_k^f)\}_{k=1}^N$, we seek to optimize the score network $s_\theta(\boldsymbol{u}_t^f, \boldsymbol{u}^c, t)$ with respect to the parameter θ such that

$$\hat{\theta} = \arg\min_\theta \sum_{t=1}^{T} \frac{1}{N} \sum_{k=1}^{N} \left[\frac{(1-\alpha_t)^2}{2\sigma_t^2 \alpha_t (1-\bar{\alpha}_t)} \left\| s_\theta(\sqrt{\bar{\alpha}_t}\boldsymbol{u}_k^f + \boldsymbol{\epsilon}_{t,k}\sqrt{1-\bar{\alpha}_t}, \boldsymbol{u}_k^c, t) + \boldsymbol{\epsilon}_{t,k} \right\|^2 \right], \quad (6)$$

where $\{\boldsymbol{\epsilon}_{t,k}\} \overset{i.i.d.}{\sim} \mathcal{N}(0, \mathbf{I})$, $t = 1, \cdots, T$; $k = 1, \cdots, N$. As discussed in [14], it is beneficial to sample quality and simpler to implement to omit the time dependent coefficient $\frac{(1-\alpha_t)^2}{2\sigma_t^2\alpha_t(1-\bar{\alpha}_t)}$, and the training process for the the conditional diffusion model is summarized in Algorithm 1 below.

Similar to the unconditioned setting, with the optimal score network $s_\theta(\boldsymbol{u}_t^f, \boldsymbol{u}^c, t)$, we can generate a new high-fidelity sample $\boldsymbol{u}^f = \boldsymbol{u}_0^f$ conditioned on a new low-fidelity solution $\boldsymbol{u}^c$ by evolving the backward process with a terminal sample $\boldsymbol{u}_T^f \sim \mathcal{N}(0, \mathbf{I})$. In the framework of DDPM, such a backward process is given by

$$\boldsymbol{u}_{t-1}^f = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{u}_t^f + \frac{\sqrt{(1-\alpha_t)}}{\sqrt{(1-\bar{\alpha}_t)}} s_{\hat{\theta}}(\boldsymbol{u}_t^f, \boldsymbol{u}^c, t)\right) + \sigma_t \xi_t, \quad t = T, T-1, \cdots, 1.$$

where $\{\xi_t\}_{t=1}^T \overset{i.i.d.}{\sim} \mathcal{N}(0, \mathbf{I})$.

In the case of DDIM, the backward process updates according to

$$\boldsymbol{u}^f_{\tau_{i-1}} = \frac{\sqrt{\bar{\alpha}_{\tau_{i-1}}}}{\sqrt{\bar{\alpha}_{\tau_i}}}\boldsymbol{u}^f_{\tau_i} + \left(\frac{\sqrt{1-\bar{\alpha}_{\tau_i}}\sqrt{\bar{\alpha}_{\tau_{i-1}}}}{\sqrt{\bar{\alpha}_{\tau_i}}} - \sqrt{1-\bar{\alpha}_{\tau_{i-1}}}\right)\boldsymbol{s}_{\hat{\theta}}\left(\boldsymbol{u}^f_{\tau_i}, \boldsymbol{u}^c, \tau_i\right), \quad i = 1, 2, \cdots, L.$$

In practice, we observe that incorporating the information of the source term a in the training of conditional score network can improve the sample quality in the sense of better fitting the PDE on the fine scale. Detailed numerical examples of this comparison are provided in Sect. 4. Therefore throughout the paper we look for a score network depend on $\boldsymbol{a}^f$ that solves (6) with the score network $\boldsymbol{s}_\theta(\boldsymbol{u}^f, \boldsymbol{u}^c, t)$ replaced by $\boldsymbol{s}_\theta(\boldsymbol{u}^f, \boldsymbol{u}^c, \boldsymbol{a}, t)$.

Algorithm 1 Training of conditional diffusion models

```
Require: Training dataset $\mathcal{S} := \{\boldsymbol{u}^c_k, \boldsymbol{u}^f_k, \boldsymbol{a}_k\}_{k=1}^N$, hyperparameter $\{\alpha_t\}_{t=0}^T \subset (0, 1)$, batch size $B$.
1: repeat
2:     Sample $\{\boldsymbol{u}^c_j, \boldsymbol{u}^f_j, \boldsymbol{a}_j\}_{j=1}^B \sim S$, let $\boldsymbol{u}^f_{0,j} = \boldsymbol{u}^f_j,\ j = 1, \cdots, B$
3:     $t \sim \text{Uniform}(\{1, \cdots, T\})$
4:     $\boldsymbol{\epsilon}_{t,j} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}),\ j = 1, \cdots B$
5:     Compute $\boldsymbol{u}^f_{t,j} = \sqrt{\bar{\alpha}_t}\boldsymbol{u}^f_{0,j} + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_j$
6:     Take gradient descent step on
7:             $\nabla_\theta\Big[\sum_{t=1}^T \frac{1}{B}\sum_{j=1}^B \|\boldsymbol{s}_\theta(\boldsymbol{u}^f_{t,j}, \boldsymbol{u}^c_i, t) + \boldsymbol{\epsilon}_{t,j}\|^2\Big]$
8: until converged.
```

3.2.1 Physics-Guided Diffusion Model

The generated high-fidelity sampled solution through the conditional diffusion model contains rich information from the training high-fidelity training data while informed by the low-fidelity input. Yet, the generated sample may not fulfil the PDE on the fine scale and hence need to be further enhanced to better conform with the physics. To improve the solution quality, we refine the solution by solving the (nonlinear) least square problem

$$\boldsymbol{u} \in \arg\min_{\boldsymbol{u}} \|\mathcal{L}^f\boldsymbol{u} - \boldsymbol{a}\|_2, \tag{7}$$

where the boundary term in the loss for simplicity and could be included in practice. In our experiments, we generate a refined solution by solve problem (7) with few Gauss-Newton iterations and initial guess chosen as the generated output from the conditioned diffusion model.

By combining the conditional sample generation step with condition diffusion models with the refining step with Gauss-Newton, we present below the overall physics-guided diffusion model for downscaling. Assume that we have access to a pre-trained conditional diffusion model or specifically the conditional score function $\boldsymbol{s}_\theta$ (see Algorithm 1). We present the algorithm 2 for the refined sample generation process in the framework of DDIM. Through our experiments, we have determined that a refining step of $t_s = 1$ sufficiently enhances the solution quality to be on par with that of the fine solver. For clarity, the $\boldsymbol{u}^f_{t,j}$, in Algorithm 1, denotes the jth noised data at diffusion time step t, while the $\boldsymbol{u}^f_{\tau_{i-1}}$ in Algorithm 2 denotes the reconstruction at diffusion time step τ_{i-1}. Furthermore, we employ the Gaussian-Newton algorithm to refine solutions produced by the diffusion model. Detailed implementation can be found in Algorithm 4 in the Appendix.

Algorithm 2 Physics-guided diffusion model (PGDM) for downscaling

```
Require: A given low-fidelity $\boldsymbol{u}^c$ and the source $\boldsymbol{a}^f$ evaluated on the fine scale, hyperparameters $\{\alpha_t\}_{t=0}^T \subset (0, 1)$ and a set of indices $\{\tau_i\}_{i=1}^L \subset [1, 2, \cdots, T]$ with length $L$.
1: $\boldsymbol{u}_T^f \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: for $i = L-1, \ldots, 0$ do
3:     $\boldsymbol{u}_{\tau_{i-1}}^f = \frac{\sqrt{\bar{\alpha}_{\tau_{i-1}}}}{\sqrt{\bar{\alpha}_{\tau_i}}}\boldsymbol{u}_{\tau_i}^f + (\frac{\sqrt{1-\bar{\alpha}_{\tau_i}}\sqrt{\bar{\alpha}_{\tau_{i-1}}}}{\sqrt{\bar{\alpha}_{\tau_i}}} - \sqrt{1-\bar{\alpha}_{\tau_{i-1}}})\boldsymbol{s}_\theta\left(\boldsymbol{u}_{\tau_i}^f, \boldsymbol{u}^c, \boldsymbol{a}^f, \tau_i\right)$
4: end for
5: for j=1, $\cdots$, $t_s$ do
6:     Refine $\boldsymbol{u}_0^f$ by Gaussian Newton Algorithm 4
7: end for
8: return $\boldsymbol{u}_0^f$
```

4 Numerical Experiments

In this section, we demonstrate the accuracy and efficiency of the proposed generative downscaling method in three types of nonlinear PDEs: the nonlinear Poisson equation in both 2D and 3D, the 2D Allen-Cahn equation, and the 2D Navier–Stokes equation. Specifically, we evaluate the performance of PGDM against several baseline downscaling techniques, including Fourier Neural Operator (FNO), Cubic Spline Interpolation (CSI), the vanilla conditional diffusion models DDPM and DDIM without the refinement via Gauss-Newton iterations, a conditional diffusion model with the conditional score without dependence on the source term (labeled as DDPM-II), as well as the Physics-Informed Diffusion Model (PIDM) [35]. Our numerical results indicate that PGDM surpasses all baseline methods in performance, achieving comparable accuracy to high-fidelity solvers while significantly reducing computational costs by more than tenfold. Below, we present in detail the data generation process, the neural network architectures employed, and the numerical results for each test case. The detailed descriptions of the neural network architecture and hyperparameters used for the diffusion models and FNO can be found in Appendix A.

4.1 Data Generation

Below we outline the process of generating training and validation data. Let us start with describing the process of generating the low-fidelity solution, high-fidelity solution, and reference solution. Our investigation covers both stationary PDEs and time evolutional PDEs. For stationary PDEs, we restrict our attention on the homogeneous Dirichlet boundary condition:

$$\begin{cases} \mathcal{L}u = a \text{ on } \Omega \\ u = 0 \text{ on } \partial\Omega. \end{cases} \tag{8}$$

We employ the finite difference method to discretize the nonlinear differential operator $\mathcal{L}$, adhering to the specified boundary condition $g(x)$. This discretization, denoted as $\mathcal{L}_d$, transforms the problem into a nonlinear optimization problem:

$$\boldsymbol{u} \in \arg\min_{\boldsymbol{u}} \|\mathcal{L}_d\boldsymbol{u} - \boldsymbol{a}\|_2, \tag{9}$$

While many nonlinear optimization solvers could potentially be used, in this paper, we opt for the Levenberg–Marquardt (LM) algorithm due to its adaptivity. Specifically, the LM algorithm interpolates between the gradient descent method and the Gauss-Newton method.

Throughout the iteration process, it adjusts its behavior, resembling the gradient descent method when the iterates are distant from a local minimum and resembling the Gauss-Newton method when they approach a local minimum. See more details on LM algorithm in Algorithm 3 of Appendix B. To generate low-fidelity solution, high-fidelity solution, and reference solution, we execute the the LM algorithm for until the L^2-misfit (9) decreases below a pre-defined error precision η =1e-5.

We also consider evolutional PDEs modeled by

$$\partial_t u = \mathcal{L}u,$$

with either Dirichlet or periodic boundary condition. To ensure the stability while maintaining a reasonable time step size, we adopt an implicit-Euler scheme for evolutional PDEs. More concretely, let K_t be the total number of iteration steps and for $n = 0, \cdots, K_t - 1$, the approximation solutions $\boldsymbol{u}^{n+1}$ are obtained by solving the following optimization problem:

$$\boldsymbol{u}^{n+1} \in \arg\min_{\boldsymbol{u}} \|(\mathcal{I} - \Delta t\mathcal{L}_d)\boldsymbol{u} - \boldsymbol{u}^n\|_2, \tag{10}$$

where Δt be the time step size. Similar to the static case, we employ the LM algorithm as our numerical solver for (10) with the stopping criterion set to be that the L^2-misfit is below η =1e-5. For the two evolutional PDEs considered in the paper, namely the Allen-Cahn and Navier–Stokes equations, we consider spatial super-resolution only. Specifically, we employ spatial mesh grids denoted as K_c and K_f, along with a time step size of $\Delta t = 0.05$ for both the low-fidelity and high-fidelity solutions. The reference solutions for the Allen-Cahn equation are computed on a spatial mesh size of $2K_f$ and a time step size of $\Delta t = 0.025$. For the reference solutions of Navier-Stokes equation, we adopt a Crank-Nicolson scheme commonly used in literature, such as the one utilized in [24]. Specifically, we set the time step size to be Δt =5e-5 and utilize a spatial mesh grid size of $2K_f$. It is important to note that there is a trade-off between using an implicit scheme with a larger time step and using a semi-implicit scheme with a smaller time step. The former allows for a larger time step, leading to faster evolution, but it introduces an error of $O(\Delta t)$.

In the data preparation step, we generate the source terms and the initial conditions from the Gaussian random field $\mathcal{N}(0, (-\Delta + b^2\mathcal{I})^{-c})$, where b and c are two hyperparameters adjusting the length-scale and smoothness of the field. For comparison purposes, all comparisons in this section are conducted at the resolution of the high-fidelity solution, indicated by a spatial mesh size K_f. Let us introduce three classical solvers for solving the nonlinear systems (9) or (10). The coarse solver generates low-fidelity solutions by solving these systems on a coarser mesh grid K_c using LM algorithm and subsequently enhances resolution through cubic spline interpolation. The fine solver produce high-fidelity solutions by directly solves the nonlinear systems employing a finer spatial mesh grid K_f. Additionally, the reference solver utilize even finer spatial mesh grid $2K_f$ coupled with a significantly finer time step size, followed by downsampling to match the resolution of fine solver. In each of the following numerical examples, we employ the aforementioned methodology to generate N training sample tuples $\{\boldsymbol{u}_i^c, \boldsymbol{u}_i^f, \boldsymbol{a}_i\}_{i=1}^N$ and M testing sample tuples $\{\boldsymbol{u}_j^c, \boldsymbol{u}_j^f, \boldsymbol{u}_j^r, \boldsymbol{a}_j\}_{j=1}^M$. Here $\boldsymbol{a}$ is generated by sampling from the Gaussian random field followed by restriction on the grids, $\boldsymbol{u}^c$ is obtained by the CSI solver, $\boldsymbol{u}^f$ is obtained by the fine solver and $\boldsymbol{u}^r$ is obtained by the reference solver. Given our focus on scenarios with limited data, we set N in this paper to be as small as $N = 30$. For a better illustration, we summarize the previously mentioned notations and hyperparameters along with their definitions in Table 5 in Appendix. Additionally, the detailed descriptions of the neural network architecture and hyperparameters used for the diffusion models and FNO can be found in Appendix A.

Table 1 Comparison of relative L^2-error for 2D nonlinear Poisson equation at 8x super-resolution on $M = 30$ testing examples

	$N = 30$ $c = 1.6$	$N = 100$ $c = 1.6$	$N = 30$ $c = 1.2$	$N = 100$ $c = 1.2$	Time
CSI	2.97e−1	2.97e−1	5.82e−1	5.82e−1	3.05e−1
Fine	3.69e−3	3.69e−3	1.36e−2	1.36e−2	8.73e0
FNO	1.84e−1	1.73e−1	2.36e−1	2.22e−1	1.66e−1
DDPM	8.74e−2	6.48e−2	9.03e−2	8.52e−2	3.22e0
DDIM	1.18e−1	6.83e−2	1.66e−1	1.38e−1	6.15e−1
DDPM-II	2.89e−1	2.53e−1	5.51e−1	5.32e−1	3.13e0
PIDM	2.55e−1	2.32e−1	5.02e−1	4.86e−1	3.56e0
Coarse+GN	6.17e−2	6.17e−2	1.91e−1	1.91e−1	6.27e−1
FNO+GN	3.93e−2	2.28e−2	1.07e−1	4.67e−2	6.14e−1
PGDM	1.31e−2	5.20e−3	2.64e−2	2.01e−2	1.29e0

The last column shows the average computational time over $M = 30$ realizations of different solutions

4.2 Nonlinear Poisson Equation

The first example is the nonlinear Poisson equation with zero Dirichlet boundary condition,

$$\begin{aligned} -0.0005\Delta u(x) + u(x)^3 &= a(x), \quad x \in (0,1)^d, \\ u(x) &= 0, \qquad x \in \partial(0,1)^d. \end{aligned} \tag{11}$$

Here, d indicates the physics dimensionality and $a(x)$ denotes the source term, which is sampled from a Gaussian random fields described by $\mathcal{N}(0, (-\Delta + 49\mathcal{I})^{-c})$, where the inverse Laplacian is equipped with zero boundary condition. Our investigation spans various values of c and the size of training set N.

For 2D cases, i.e. $d = 2$, we select $K_c = 16$, $K_f = 128$ as the mesh grid sizes for the coarse solver and the fine solver, respectively. In addition to evaluating DDPM against the baseline data-driven method FNO, we also compare its performance with two additional diffusion models: PIDM and DDPM-II. The DDPM-II solver is the diffusion model with the score function conditioned only on the coarse solution, whereas PIDM is a Physics-Informed Diffusion Model proposed by [35]. We compare the performances of different solvers under these conditions for the 2D scenario in Table 1. Solutions computed from various solvers in 2D with $c = 1.6$ and $N = 100$ are depicted in Fig. 1, while the corresponding solutions for $c = 1.2$ and $N = 100$ are illustrated in Fig. 2. Additionally, we present the error distributions of the numerical solutions corresponding to all solvers in Fig. 3, indicating that DDPM performs better than the other three data-driven methods (FNO, DDPM-II, PIDM). In 3D scenarios, we select $K_c = 16$, $K_f = 64$ for the mesh grid size of the coarse and the fine solver respectively. The performances of different solvers under analogous conditions in 3D settings are comprehensively detailed in Table 2. Visual comparisons for 3D solver outputs corresponding to $c = 1.2$ and $N = 100$ are also presented are presented in Fig. 4, and the results for $c = 1.4$ and $N = 100$ are shown in Fig. 5. As demonstrated in Table 1 and Table 2, PGDM maintains the same level of accuracy as the fine solver while reducing computational time by a significant factor of ten.

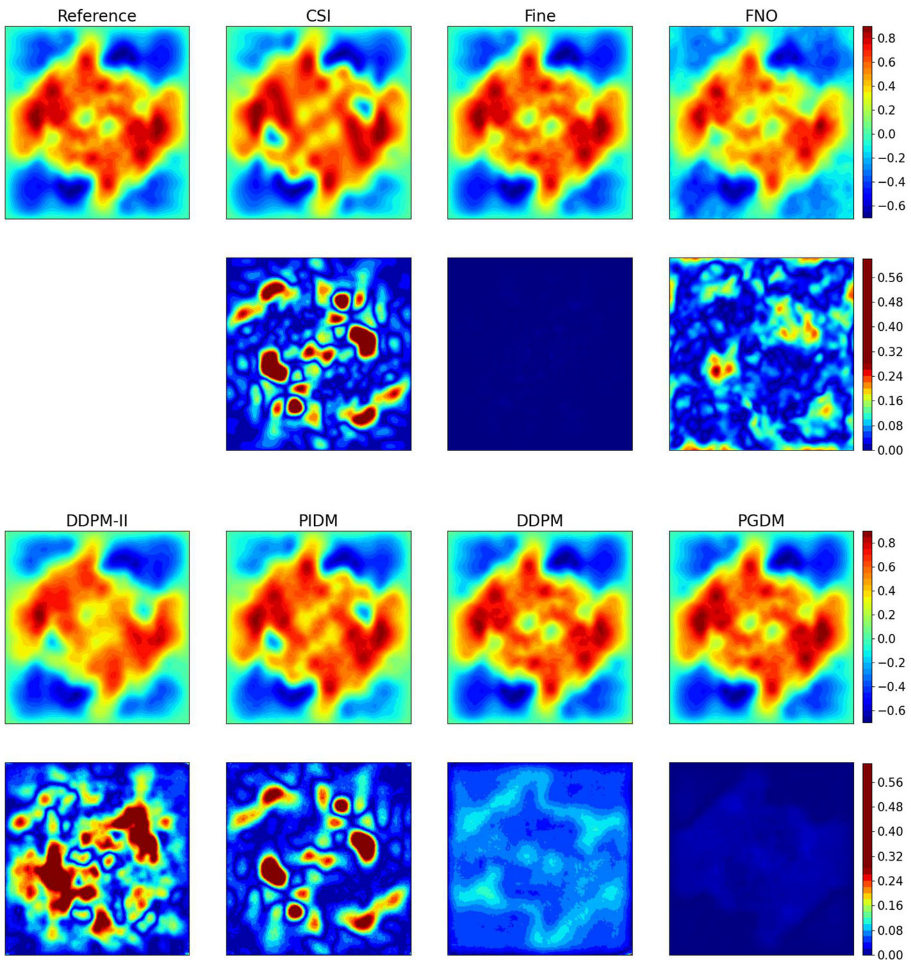

Fig. 1 Comparison of predictions and absolute errors for the 2D nonlinear Poisson equation on $M = 30$, generated by different solvers with $c = 1.6$ and $N = 30$ training samples. The first and third rows display the solutions generated by the different solvers, and the second and fourth rows present the corresponding absolute errors to the reference solutions

4.3 2D Allen-Cahn Equaiton

Consider 2D Allen-Cahn equaiton with periodic boundary condition:

$$\begin{aligned} \partial_t u(x) &= \kappa \Delta u(x) + \gamma u(x)\Big(\frac{1}{4} - u(x)^2\Big), \quad x \in (0,1)^2, t \in (0, 0.5], \\ u(0,x) &= u_0(x), \quad x \in (0,1)^2. \end{aligned} \tag{12}$$

Here the diffusion coefficient set to κ =1e-3 and the reaction coefficient set to $\gamma = 5$. The initial conditions $u_0(x)$ draw from Gaussian random field $\mathcal{N}(0, (-\Delta + 49\mathcal{I})^{-c})$ where the inverse Laplacian Δ is applied with periodic boundary conditions. We explore different values of reaction coefficient γ and various c. The time step size is set to $\Delta t = 0.05$, with the

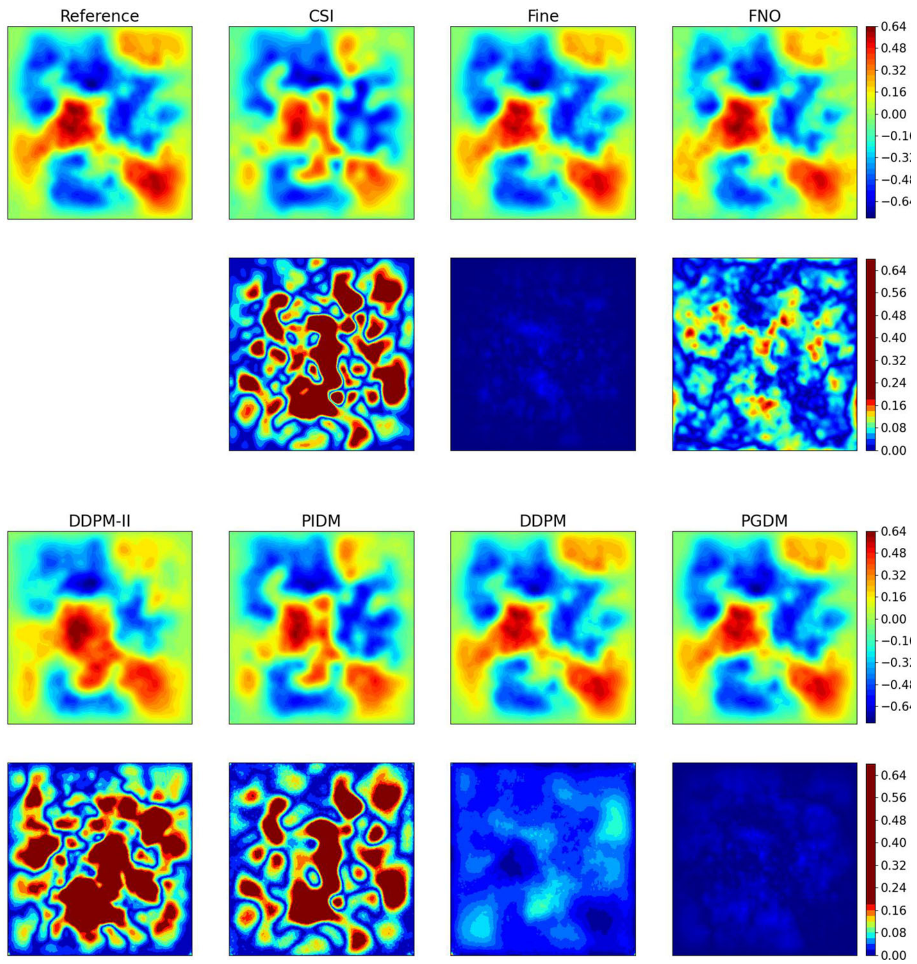

Fig. 2 Comparison of predictions and absolute errors for the 2D nonlinear Poisson equation on $M = 30$, generated by different solvers with $c = 1.2$ and $N = 30$ training samples. The first and third rows display the solutions generated by the different solvers, and the second and fourth rows present the corresponding absolute errors to the reference solutions

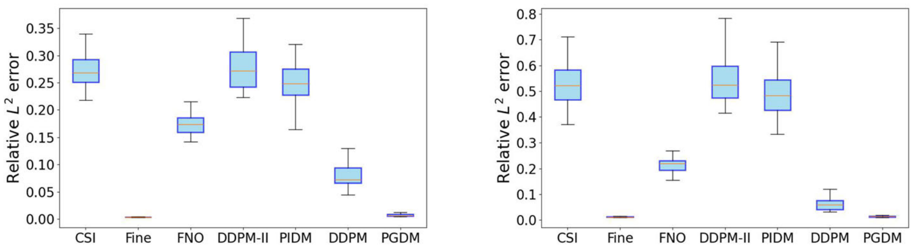

Fig. 3 2D nonlinear Poisson: Boxplot of relative L^2 errors for $M = 30$ test samples compared to the reference solutions, generated by different solvers trained on $N = 30$ training samples. The left and right figures show the numerical results corresponding to $c = 1.6$ and $c = 1.2$ respectively. The yellow lines indicate the median, and the blue boxes represent the interquartile range for each solver. The whiskers extend to the smallest and largest values within 1.5 times the IQR from the first and third quartiles

Table 2 Comparison of relative L^2 error for 3D nonlinear Poisson equation at 4x super-resolution on $M = 30$ testing set

	$N = 30$ $c = 1.6$	$N = 100$ $c = 1.6$	$N = 30$ $c = 1.4$	$N = 100$ $c = 1.4$	Time
CSI	4.69e−1	4.69e−1	6.72e−1	6.72e−1	2.18e−1
Fine	2.66e−2	2.66e−2	4.74e−2	4.74e−2	1.60e2
FNO	2.29e−1	1.71e−1	2.73e−1	2.03e−1	6.63e−1
DDPM	1.13e−1	1.10e−1	1.12e−1	1.10e−1	3.23e1
DDIM	1.46e−1	1.33e−1	1.43e−1	1.36e−1	6.68e0
Coarse+GN	9.61e−2	9.61e−2	1.61e−1	1.61e−1	9.91e0
FNO+GN	3.97e−2	2.99e−2	6.26e−1	5.19e−2	1.03e1
PGDM	2.79e−2	2.77e−2	4.77e−2	4.74e−2	1.63e1

The last column documents the average computational time over $M = 30$ realizations of different solutions

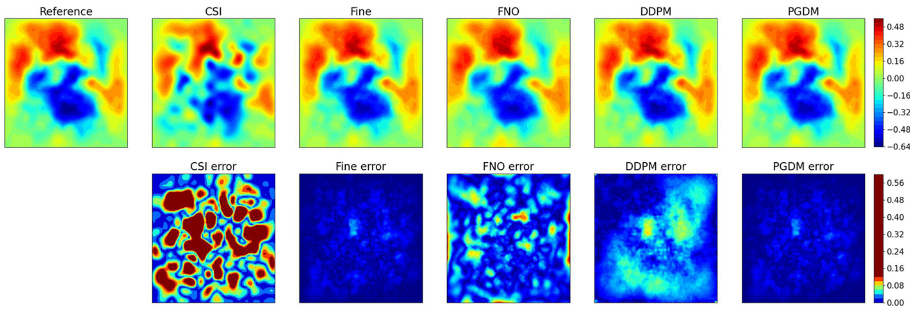

Fig. 4 3D nonlinear Poisson: Predictions and corresponding absolute errors generated by different solvers with $c = 1.6$ and $N = 100$ training samples

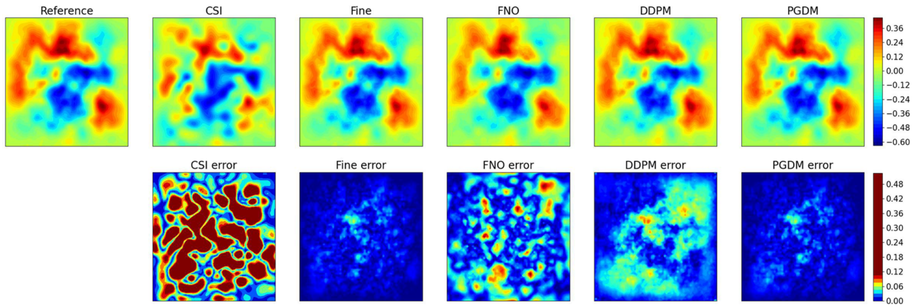

Fig. 5 3D nonlinear Poisson: Predictions and corresponding absolute errors generated by different solvers with $c = 1.4$ and $N = 100$ training samples

total number of steps K_t set to 10. For computational mesh grids, sizes are set at $K_c = 16$ for the coarse solver and $K_f = 128$ for the fine solver, respectively. The performance of various solvers across these settings is detailed in Table 3. Predictions at $t = 0.5$ of different solvers with $c = 1.6$, $\gamma = 5$ and $N = 30$ are presented in Fig. 6, and the snapshots of predictions of PGDM are shown in Fig. 7.

Table 3 Comparison of relative L^2 error for 2D Allen-Cahn equation at 8x super-resolution on $N = 30$ training set and $M = 20$ testing set

	$\gamma = 1$ $c = 1.6$	$\gamma = 1$ $c = 1.2$	Time	$\gamma = 5$ $c = 1.6$	$\gamma = 5$ $c = 1.2$	Time
CSI	4.16e−1	7.84e−1	5.59e−2	4.57e−1	8.37e−1	2.64e−1
Fine	2.67e−2	6.84e−2	1.71e1	4.73e−2	7.73e−2	5.84e1
FNO	4.57e−1	7.27e−1	8.21e−2	4.61e−1	7.84e−1	8.21e−2
DDPM	1.66e−1	1.71e−1	1.42e1	2.05e−1	1.89e−2	1.42e1
DDIM	1.82e−1	3.2e−1	2.84e0	2.11e−1	3.54e−1	2.84e0
Coarse+GN	8.73e−2	1.91e−1	3.18e−1	1.06e−1	1.94e−1	4.87e−1
FNO+GN	1.03e−1	1.87e−1	3.42e−1	1.29e−1	2.05e−1	3.42e−1
PGDM	5.96e−2	9.09e−2	3.11e0	6.67e−2	1.05e−1	4.23e0

The fourth and the last column show the averaged computational time over $M = 20$ realizations of different solutions

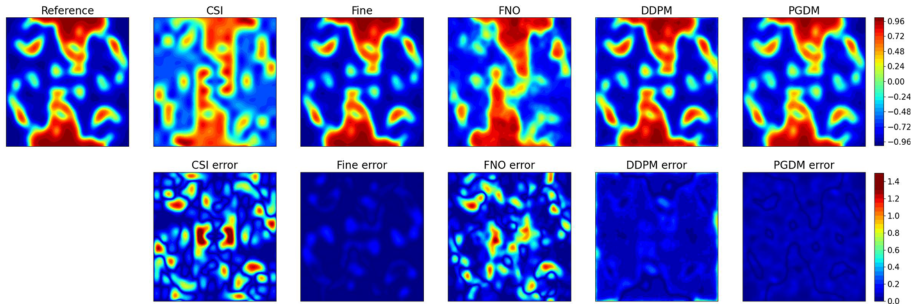

Fig. 6 2D Allen-Cahn: Predictions at $t = 0.5$ and corresponding absolute errors generated by different solvers with $\gamma = 5$, $c = 1.6$ and $N = 30$

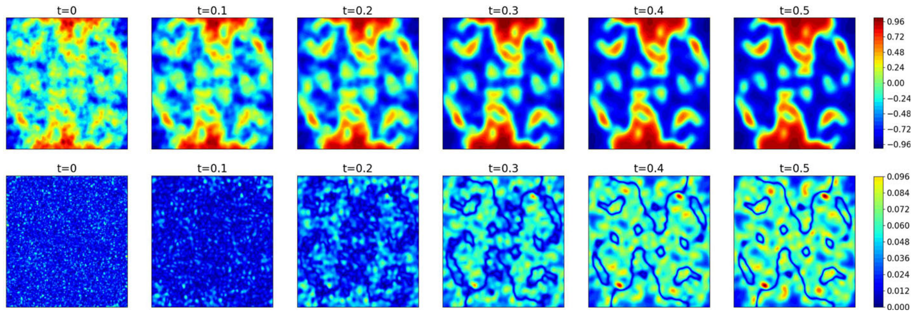

Fig. 7 2D Allen-Cahn: Snapshots of evolution of PGDM and the corresponding absolute errors compared to the reference solution with $\gamma = 5$, $c = 1.6$ and $N = 30$

Table 4 Comparison of relative L^2 error for 2D Navier-Stokes equation at 4x super-resolution on $M = 20$ testing set

	ν = 2e-4	ν = 1e-4	ν = 2e-5	Time
CSI	1.54e−1	1.63e−1	2.35e−1	1.38e0
Fine	4.23e−2	6.51e−2	1.19e−1	2.26e2
FNO	3.43e−1	3.65e−1	4.88e−1	1.38e−2
DDPM	8.09e−2	1.06e−1	1.56e−1	8.99e−1
DDIM	9.59e−2	1.43e−1	2.01e−1	2.32e−1
Coarse+GN	5.42e−2	7.98e−2	1.33e−1	6.28e0
FNO+GN	1.22e−1	1.41e−1	1.84e−1	6.13e0
PGDM	4.23e−2	6.51e−2	1.19e−1	6.34e0

The last column documents the average computational time over $M = 20$ realizations of different solutions

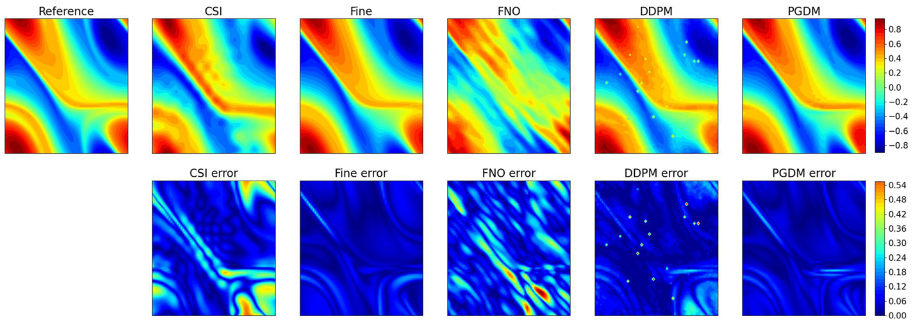

Fig. 8 2D Navier–Stokes: Predictions at $t = 2$ and the corresponding absolute errors generated by different solvers with ν =2e-4 and $N = 30$

4.4 2D Navier–Stokes Equation

Consider the 2D Navier–Stokes equation in the vorticity form with periodic boundary condition:

$$\begin{aligned} \partial_t w(t,x) + \mu u(t,x) \cdot \nabla w(t,x) &= \nu \Delta w(t,x) + f(x), \quad x \in (0,1)^2, t \in (0,2], \\ w(0,x) &= w_0(x), \qquad\qquad x \in (0,1)^2. \end{aligned} \tag{13}$$

The transportation coefficient is set to $\mu = 4$, and the forcing term is selected as $f(x) = 0.2(\sin(2\pi(x+y)) + \cos(2\pi(x+y)))$. To generate initial condition, we draw functions from the same Gaussian random field $\mathcal{N}(0, (-\Delta + 25\mathcal{I})^{-5})$ where the inverse Laplacian Δ is applied with periodic boundary conditions. These functions are subsequently evolved using the reference solver for two seconds. The time step size is set to $\Delta t = 0.05$, with the total number of steps set to $K_t = 40$. The mesh grid sizes are set to $K_c = 16$ for the coarse solver and $K_f = 64$ for the fine solver, respectively. We fix the training set size N at 30 and examine three different viscosity coefficients ν = 2e-4, 1e-4, 2e-5. The performance of various solvers across these settings is detailed in Table 4. Predictions at $t = 2$ of different solvers with ν =2e-4 are presented in Fig. 8, and the snapshots of predictions of PGDM are shown in Fig. 9; Predictions at $t = 2$ of different solvers ν =1e-4 are presented in Fig. 10, and the snapshots of predictions of PGDM are shown in Fig. 11.

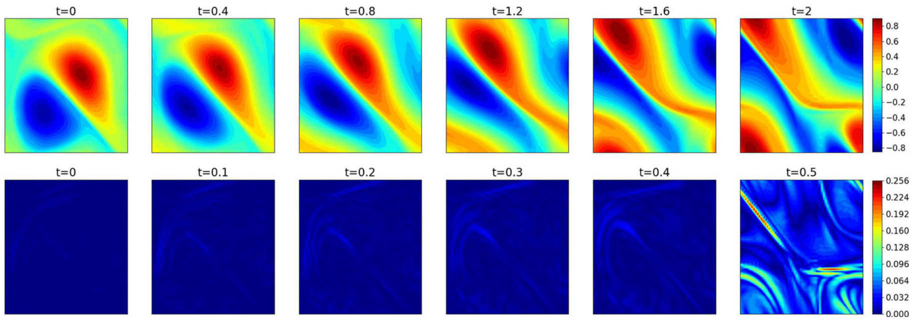

Fig. 9 2D Navier–Stokes: Snapshots of evolution of PGDM and the corresponding absolute errors to the reference solution with ν =2e-4 and $N = 30$

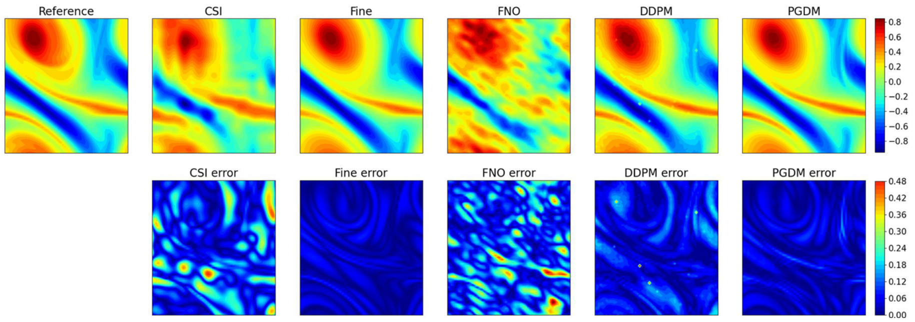

Fig. 10 2D Navier–Stokes: Predictions at $t = 2$ and the corresponding absolute errors generated by different solvers with ν =1e-4 and $N = 30$

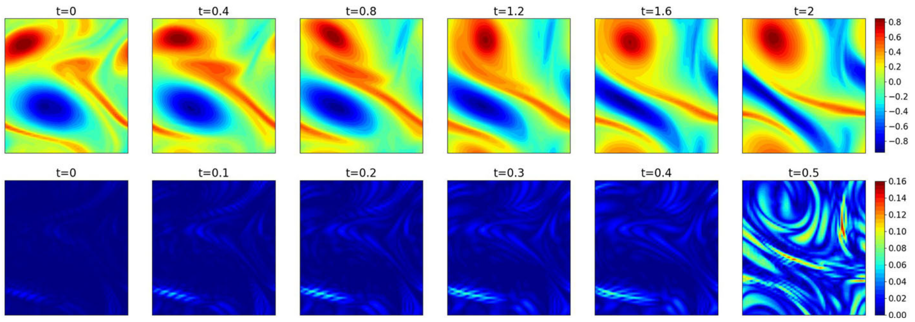

Fig. 11 2D Navier–Stokes: Snapshots of evolution of PGDM and the corresponding absolute errors to the reference solution with ν =1e-4 and $N = 30$

5 Conclusion

We propose a data-driven surrogate method called PGDM for accelerating the computation of (nonlinear) PDEs. PGDM first generates a high-fidelity solution conditional on a low-resolution input, followed by a mild refinement of the former with a PDE solver on the fine-grid. Our numerical results show that PGDM can produce high-fidelity solutions that are comparable to those generated by fine-scale solvers, while requiring very limited training data (as few as 30 instances). More importantly, we demonstrate that our PGDM also significantly

Table 5 Table of notations

Notation	Meaning
K_c	Uniform grid size of coarse solver
$\boldsymbol{u}^c$	CSI solution
K_f	Uniform grid size of fine solver
$\boldsymbol{u}^f$	Fine solution
K_t	Number of evolution steps
Δt	Time step size in evolution problem
T	Total time steps in DDPM
τ	Sequence of skipped time steps in DDIM
$\{\beta_i\}_{i=0}^{T}$	Scale of Noise in DDPM
$\{\alpha_i\}_{i=0}^{T}$	Hyperparameter in DDPM
N	Number of training samples
M	Number of testing samples
b, c	Hyperparameters of Gaussian random fields

reduce the computational time, especially in 3D examples, where we observe a tenfold decrease compared to fine-scale solvers.

Appendix

A Neural Networks Architecture and Hyperparameters

Our diffusion models are based on the DDPM architecture [14], which uses U-Net [33] as the backbone (Table 5). During our experiments, we omit the use of self-attention, resulting in significant reductions in training time while maintaining similar sample quality. The base channel count, the list of Down/Up channel multipliers and the list of middle channel refer to the hyperparameters of the U-Net, which is detailed in Table 6. To accelerate sampling process using DDIM, we take skipped time steps τ be $[1, 5, 10, 15, 20, 25, \cdots, T-5, T]$. The linear noise schedule is configured from $\beta_0 = 0.0001$ to $\beta_T = 0.02$.

During training, we utilize the Adam optimizer with a dynamic learning rate that linearly decays every 5000 steps with a decay rate of 0.05. The total number of training epochs is set to 10000.

The architecture of FNO follows that described in [24]. The number of lifting channels, number of FFT truncation modes, and number of Fourier layers for different examples are specified in Table 7. During training, we utilize the Adam optimizer with a dynamic learning rate that linearly decays every 5000 steps with a decay rate of 0.05. Training continues until the loss drops below 1e-6 or reaches the maximum iteration number of 50000.

Our model training were performed on an NVIDIA RTX 3070 graphics card, while predictions and refinements with Gaussian-Newton were executed on an AMD Ryzen 7 3700X processor.

Table 6 Table of DDPM hyperparameters

	2D Nonlinear poisson	3D Nonlinear poisson	1Dt + 2Dx Allen-Cahn	1Dt + 2Dx Navier–Stokes
Target resolution	128×128	$64 \times 64 \times 64$	$10 \times 128 \times 128$	$40 \times 64 \times 64$
Super-resolution scale	8	4	8	4
Timesteps T	400	400	400	200
Base Channels	128	128	256	128
Down/Up channel multipliers	1,2,4,8	1,2,4,8	1,2,4,8	1,2,4,8
Middle channel list	[512, 512]	[1024, 1024]	[1024, 1024]	[1024, 1024]
Batch size	4	2	2	2
Training time $N = 30$	≈ 1.2 h	≈ 5.2 h	≈ 2.8 h	≈ 3.4 h
Training time $N = 100$	≈ 4.2 h	≈ 22.4 h	≈ 7.6 h	≈ 11.8 h

Table 7 Table of FNO hyperparameters

	layers	modes	lifting channel
Nonlinear Poisson 2D	2	16	32
Nonlinear Poisson 3D	4	12	64
Allen-Cahn $1Dt + 2Dx$	4	16	64
Navier–stokes $1Dt + 2Dx$	4	16	64

B Levenberg–Marquardt Algorithm

In this part, we present the Levenberg–Marquardt (LM) algorithm for solving the nonlinear optimization problem (9) and (10) in Algorithm 3. In all of our numerical experiments, we fix $\lambda = 0.5$ and η =1e-5.

Algorithm 3 Levenberg–Marquardt algorithm

```
Require: Initial guess $\boldsymbol{u}_0$, the source term $\boldsymbol{a}$, the discretization of operator $\mathcal{L}_d$, initial damping parameter $\lambda$
    and stopping criterion $\eta$.
1: Let $\boldsymbol{u} = \boldsymbol{u}_0$
2: repeat
3:     Compute residual $\boldsymbol{r} = \mathcal{L}_d \boldsymbol{u} - \boldsymbol{a}$
4:     Compute Jacobian matrix $J = \frac{\partial \mathcal{L}_d \boldsymbol{u}}{\partial \boldsymbol{u}}$
5:     Solve linear system $[J^T J + \lambda \text{diag}(J^T J)]\boldsymbol{\delta} = J^T \boldsymbol{r}$ for $\boldsymbol{\delta}$
6:     Update $\boldsymbol{u}_{new} = \boldsymbol{u} + \delta$
7:     if $\|\mathcal{L}_d \boldsymbol{u}_{new} - \boldsymbol{a}\| > \|\mathcal{L}_d \boldsymbol{u} - \boldsymbol{a}\|$ then
8:         $\lambda = \lambda * 2$
9:     else
10:        $\lambda = \lambda / 2$
11:    end if
12:    $\boldsymbol{u} = \boldsymbol{u}_{new}$
13: until $\|\boldsymbol{r}\| < \eta$
14: return $\boldsymbol{u}$
```

C Physics-Informed Diffusion Model

For the physics-informed diffusion model [35], our numerical test suggests that conditioning the diffusion model on both gradient information and the coarse solution yields better performance compared to the vanilla PIDM, which is conditioned solely on gradient information. In the application of the PIDM to 2D nonlinear Poisson equation, the conditioning information is defined as the gradient of the L^2 misfit, i.e.

$$\boldsymbol{g}_t = \frac{\partial r_t}{\partial \boldsymbol{x}_t},$$

where

$$r_t := \| -0.0005\Delta \boldsymbol{x}_t + \boldsymbol{x}_t^3 - \boldsymbol{a}\|_2^2.$$

We employ the same architecture to construct and train the model using Algorithm 1, with a modified loss function:

$$\nabla_\theta \Big[\sum_{t=1}^{T} \frac{1}{B} \sum_{j=1}^{B} \| \boldsymbol{s}_\theta(\boldsymbol{u}_{t,j}^f, \boldsymbol{u}_i^c, \boldsymbol{g}_{t,j}, t) + \boldsymbol{\epsilon}_{t,j} \|^2 \Big].$$

The gradient guidance strength is set to $w = 1$. Various time-step locations t_s in the backward diffusion process were tested ($t_s = [20, 100, 200, 400]$), and it was determined that $t_s = 20$ provides optimal performance, leading to its adoption in the model. As shown Fig. 3, DDPM outperforms PIDM, and we provide some heuristic explanations for this below. In fact, the inputs of the two score networks of the two methods are different. At a specific time t, the score of PIDM takes $\boldsymbol{x}_t, t, \boldsymbol{u}^c, \boldsymbol{g}$ as the inputs, where $\boldsymbol{g}$ is the output of a fixed problem-dependent function of $\boldsymbol{x}_t$ and source term $\boldsymbol{a}$. In contrast, the score of DDPM takes $\boldsymbol{x}_t, t, \boldsymbol{u}^c, \boldsymbol{a}$ as the inputs. Intuitively, including gradient information as an additional input provides more comprehensive information than simply the source term. However, this significantly increases training complexity, especially when the residual function is complicated and the total time step N_t is large, making training much more difficult and potentially leading to poor performance when the training data is limited.

D Gaussian-Newton Algorithm

To refine the solution obtained from the coarse solver, diffusion model and the FNO, we introduce the one-step Gaussian-Newton refinement process, outlined in Algorithm 4.

Algorithm 4 One step Gaussian Newton update

```
Require: The $\boldsymbol{u}$ to be refined, the source term $\boldsymbol{a}$, and the discretization of operator $\mathcal{L}_d$.
1: Compute Jacobian matrix $J = \frac{\partial \mathcal{L}_d \boldsymbol{u}}{\partial \boldsymbol{u}}$
2: Compute residual $\boldsymbol{r} = \mathcal{L}_d \boldsymbol{u} - \boldsymbol{a}$
3: Solve linear system $J^T J \boldsymbol{\delta} = J^T \boldsymbol{r}$ for $\boldsymbol{\delta}$
4: $\boldsymbol{u}_{new} = \boldsymbol{u} + \boldsymbol{\delta}$
5: return $\boldsymbol{u}_{new}$
```

Acknowledgements YL thanks the support from the National Science Foundation through the award DMS-2343135 and the support from the Data Science Initiative at University of Minnesota through a MnDRIVE DSI Seed Grant.

Data Availability All data reported in the manuscript were generated through a Python implementation of the methods outlined in the paper. The source code is available at https://github.com/woodssss/Generative-downsscaling-PDE-solvers.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Apte, R., Nidhan, S., Ranade, R., Pathak, J.: Diffusion model based data generation for partial differential equations. arXiv preprint arXiv:2306.11075 (2023)
2. Arisaka, S., Li, Q.: Principled acceleration of iterative numerical methods using machine learning. Proceedings of the 40th International Conference on Machine Learning (2023)
3. Azulay, Y., Treister, E.: Multigrid-augmented deep learning preconditioners for the Helmholtz equation. SIAM J. Sci. Comput. **45**(3), S127-51 (2022)
4. Baño-Medina, J., Manzanas, R., Gutiérrez, J.M.: Configuration and intercomparison of deep learning neural models for statistical downscaling. Geosci. Model Dev. **13**(4), 2109–2124 (2020)
5. Cai, S., Mao, Z., Wang, Z., Yin, M., Karniadakis, G.E.: Physics-informed neural networks (pinns) for fluid mechanics: a review. Acta Mech. Sin. **37**(12), 1727–1738 (2021)
6. Chen, Y., Dong, B., Xu, J.: Meta-mgnet: meta multigrid networks for solving parameterized partial differential equations. J. Comput. Phys. **455**, 110996 (2022)
7. Dhariwal, P., Nichol, A.: Diffusion models beat gans on image synthesis. Adv. Neural Inf. Process. Syst. **34**, 8780–8794 (2021)
8. Farimani, A.B., Gomes, J., Pande, V.S.: Deep learning the physics of transport phenomena. arXiv preprint arXiv:1709.02432 (2017)
9. Goswami, S., Bora, A., Yu, Y., Karniadakis, G.E.: Physics-informed neural operators. arXiv preprint arXiv:2207.05748 (2022)
10. Goswami, S., Bora, A., Yu, Y., Karniadakis, G.E.: Physics-informed deep neural operator networks. In: Machine Learning in Modeling and Simulation: Methods and Applications, pp. 219–254. Springer (2023)
11. Groenke, B., Madaus, L., Monteleoni, C.: Climalign: Unsupervised statistical downscaling of climate variables via normalizing flows. In: Proceedings of the 10th International Conference on Climate Informatics, pp. 60–66 (2020)
12. Han, J., Jentzen, A.E.W.: Solving high-dimensional partial differential equations using deep learning. Proc. Natl. Acad. Sci. **115**(34), 8505–8510 (2018)
13. Han, J., Jentzen, A., et al.: Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. Commun. Math. Stat. **5**(4), 349–380 (2017)
14. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. Adv. Neural Inf. Process. Syst. **33**, 6840–6851 (2020)
15. Ho, J., Saharia, C., Chan, W., Fleet, D.J., Norouzi, M., Salimans, T.: Cascaded diffusion models for high fidelity image generation. J. Mach. Learn. Res. **23**(1), 2249–2281 (2022)
16. Hsieh, J.T., Zhao, S., Eismann, S., Mirabella, L., Ermon, S.: Learning neural pde solvers with convergence guarantees. In: International Conference on Learning Representations (2018)
17. Jiang, P., Yang, Z., Wang, J., Huang, C., Xue, P., Chakraborty, T., Chen, X., Qian, Y.: Efficient super-resolution of near-surface climate modeling using the fourier neural operator. J. Adv. Modeling Earth Syst. **15**(7), e2023MS003800 (2023)
18. Jin, S., Ma, Z., Wu, K.: Asymptotic-preserving neural networks for multiscale time-dependent linear transport equations. J. Sci. Comput. **94**(3), 57 (2023)
19. Joshi, A., Shah, V., Ghosal, S., Pokuri, B., Sarkar, S., Ganapathysubramanian, B., Hegde, C.: Generative models for solving nonlinear partial differential equations. In: Proc. of NeurIPS Workshop on ML for Physics (2019)
20. Kharazmi, E., Zhang, Z., Karniadakis, G.E.: Variational physics-informed neural networks for solving partial differential equations. arXiv preprint arXiv:1912.00873 (2019)

21. Kharazmi, E., Zhang, Z., Karniadakis, G.E.: hp-vpinns: variational physics-informed neural networks with domain decomposition. Comput. Methods Appl. Mech. Eng. **374**, 113547 (2021)
22. Kovachki, N., Li, Z., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., Anandkumar, A.: Neural operator: Learning maps between function spaces. arXiv preprint arXiv:2108.08481 (2021)
23. Leinonen, J., Nerini, D., Berne, A.: Stochastic super-resolution for downscaling time-evolving atmospheric fields with a generative adversarial network. IEEE Trans. Geosci. Remote Sens. **59**(9), 7211–7223 (2020)
24. Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., Anandkumar, A.: Fourier neural operator for parametric partial differential equations. arXiv preprint arXiv:2010.08895 (2020)
25. Li, Z., Zheng, H., Kovachki, N., Jin, D., Chen, H., Liu, B., Azizzadenesheli, K., Anandkumar, A.: Physics-informed neural operator for learning partial differential equations. arXiv preprint arXiv:2111.03794 (2021)
26. Lu, J., Lu, Y.: A priori generalization error analysis of two-layer neural networks for solving high dimensional schrödinger eigenvalue problems. Commun. Am. Math. Soc. **2**(1), 1–21 (2022)
27. Lu, L., Jin, P., Karniadakis, G.E.: Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. arXiv preprint arXiv:1910.03193 (2019)
28. Lu, Y., Lu, J., Wang, M.: A priori generalization analysis of the deep ritz method for solving high dimensional elliptic partial differential equations. In: Conference on learning theory, pp. 3196–3241. PMLR (2021)
29. Lu, Y., Wang, L., Xu, W.: Solving multiscale steady radiative transfer equation using neural networks with uniform stability. Res. Math. Sci. **9**(3), 45 (2022)
30. Nikolopoulos, S., Kalogeris, I., Stavroulakis, G., Papadopoulos, V.: Ai-enhanced iterative solvers for accelerating the solution of large-scale parametrized systems. Int. J. Numer. Methods Eng. **125**(2), e7372 (2024)
31. Price, I., Rasp, S.: Increasing the accuracy and resolution of precipitation forecasts using deep generative models. In: International conference on artificial intelligence and statistics, pp. 10555–10571. PMLR (2022)
32. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J. Comput. Phys. **378**, 686–707 (2019)
33. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18, pp. 234–241. Springer (2015)
34. Sachindra, D., Ahmed, K., Rashid, M.M., Shahid, S., Perera, B.: Statistical downscaling of precipitation using machine learning techniques. Atmos. Res. **212**, 240–258 (2018)
35. Shu, D., Li, Z., Farimani, A.B.: A physics-informed diffusion model for high-fidelity flow field reconstruction. J. Comput. Phys. **478**, 111972 (2023)
36. Song, J., Meng, C., Ermon, S.: Denoising diffusion implicit models. arXiv preprint arXiv:2010.02502 (2020)
37. Song, Y., Sohl-Dickstein, J., Kingma, D.P., Kumar, A., Ermon, S., Poole, B.: Score-based generative modeling through stochastic differential equations. arXiv preprint arXiv:2011.13456 (2020)
38. Um, K., Brand, R., Fei, Y.R., Holl, P., Thuerey, N.: Solver-in-the-loop: learning from differentiable physics to interact with iterative pde-solvers. Adv. Neural Inf. Process. Syst. **33**, 6111–6122 (2020)
39. Vandal, T., Kodra, E., Ganguly, A.R.: Intercomparison of machine learning methods for statistical downscaling: the case of daily and extreme precipitation. Theor. Appl. Climatol. **137**, 557–570 (2019)
40. Vandal, T., Kodra, E., Ganguly, S., Michaelis, A., Nemani, R., Ganguly, A.R.: Deepsd: Generating high resolution climate change projections through single image super-resolution. In: Proceedings of the 23rd acm sigkdd international conference on knowledge discovery and data mining, pp. 1663–1672 (2017)
41. Wang, S., Perdikaris, P.: Long-time integration of parametric evolution equations with physics-informed deeponets. J. Comput. Phys. **475**, 111855 (2023)
42. Wang, S., Sankaran, S., Wang, H., Perdikaris, P.: An expert's guide to training physics-informed neural networks. arXiv preprint arXiv:2308.08468 (2023)
43. Wang, S., Wang, H., Perdikaris, P.: Learning the solution operator of parametric partial differential equations with physics-informed DeepONets. Sci. Adv. **7**(40), eabi8605 (2021)
44. Wei, M., Zhang, X.: Super-resolution neural operator. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 18247–18256 (2023)
45. Weinan, E., Yu, B.: The deep ritz method: a deep learning-based numerical algorithm for solving variational problems. Commun. Math. Stat. **6**(1), 1–12 (2018)

46. Wilby, R.L., Wigley, T., Conway, D., Jones, P., Hewitson, B., Main, J., Wilks, D.: Statistical downscaling of general circulation model output: a comparison of methods. Water Resour. Res. **34**(11), 2995–3008 (1998)
47. Yang, G., Sommer, S.: A denoising diffusion model for fluid field prediction. arXiv e-prints pp. arXiv–2301 (2023)
48. Yang, Q., Hernandez-Garcia, A., Harder, P., Ramesh, V., Sattegeri, P., Szwarcman, D., Watson, C.D., Rolnick, D.: Fourier neural operators for arbitrary resolution climate data downscaling. arXiv preprint arXiv:2305.14452 (2023)
49. Yu, J., Lu, L., Meng, X., Karniadakis, G.E.: Gradient-enhanced physics-informed neural networks for forward and inverse pde problems. Comput. Methods Appl. Mech. Eng. **393**, 114823 (2022)
50. Zang, Y., Bao, G., Ye, X., Zhou, H.: Weak adversarial networks for high-dimensional partial differential equations. J. Comput. Phys. **411**, 109409 (2020)