MeshfreeFlowNet: A Physics-Constrained Deep Continuous Space-Time Super-Resolution Framework

Chiyu "Max" Jiang*[1], Soheil Esmaeilzadeh*[2], Kamyar Azizzadenesheli[3], Karthik Kashinath[4], Mustafa Mustafa[4], Hamdi A. Tchelepi[2], Philip Marcus[1], Prabhat[4], and Anima Anandkumar[3,5]

[1]University of California, Berkeley, CA 94720, USA
[2]Stanford University, Stanford, CA 94305, USA
[3]California Institute of Technology, Pasadena, CA, 91125, USA
[4]Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA
[5]NVIDIA, Santa Clara, CA 95051, USA

Abstract

We propose MeshfreeFlowNet, a novel deep learning-based super-resolution framework to generate continuous (grid-free) spatio-temporal solutions from the low-resolution inputs. While being computationally efficient, MeshfreeFlowNet accurately recovers the fine-scale quantities of interest. MeshfreeFlowNet allows for: (i) the output to be sampled at all spatio-temporal resolutions, (ii) a set of Partial Differential Equation (PDE) constraints to be imposed, and (iii) training on fixed-size inputs on arbitrarily sized spatio-temporal domains owing to its fully convolutional encoder.

We empirically study the performance of MeshfreeFlowNet on the task of super-resolution of turbulent flows in the Rayleigh–Bénard convection problem. Across a diverse set of evaluation metrics, we show that MeshfreeFlowNet significantly outperforms existing baselines. Furthermore, we provide a large scale implementation of MeshfreeFlowNet and show that it efficiently scales across large clusters, achieving 96.80% scaling efficiency on up to 128 GPUs and a training time of less than 4 minutes. We provide an open-source implementation of our method that supports arbitrary combinations of PDE constraints [1].

1 Introduction

In recent years, along with the significant growth of computational resources, there has been an increasing attempt towards accurately resolving the wide range of length and time scales present within different physical systems. These length and time scales often differ by several orders of magnitude. Such time and length scale disparities are commonly observed in different physical phenomena such as turbulent flows, convective diffusive systems, subsurface systems, multiphase flows, chemical processes, and climate systems [1–10]. Yet many key physical quantities of interest are highly dependent on correctly resolving such fine scales, such as the energy spectrum in turbulent flows.

From a numerical perspective, resolving the wide range of spatio-temporal scales within such physical systems is challenging since extremely small spatial and temporal numerical stencils would be required. In order to alleviate the computational burden of fully resolving such a wide range of spatial and temporal scales, multiscale computational approaches have been developed. For instance, in the subsurface flow problem, the main idea of the multiscale approach is to build a set of operators that map between the unknowns associated with the computational cells in a fine-grid and the unknowns on a coarser grid. The operators are computed numerically by solving localized flow problems. The multiscale basis functions have subgrid-scale resolutions, ensuring that the fine-scale heterogeneity is correctly accounted for in a systematic manner [11–13]. Similarly, in turbulent flows [14, 15], climate forecast [16], multiphase systems [17, 18], reactive transport [19, 20], complex fluids [21] and biological systems [22], multiscale approaches attempt to relate the coarse-scale solutions to the fine-scale local solutions taking into account the fine-scale physics and dynamics. In these multiscale approaches, although the small scale physics and dynamics are being captured and accounted for at larger scales, the fine-grid solutions are extremely costly and hence impractical to solve for. Reconstructing the fine-scale subgrid solutions by having coarse-scale solutions in space and/or time still remains a challenge.

In this manuscript, we refer to such a process of reconstructing the fine-scale subgrid solutions from the coarse-scale solutions as *super-resolution*, where the high-resolution solutions are reconstructed using the low-resolution physical solutions in space and/or time. One key insight is that there are inherent statistical correlations between such pairs of low-resolution and high-resolution solutions that can be leveraged to reconstruct high-resolution solutions from the low-resolution inputs. Furthermore, this super-resolution process can be effectively modeled using deep learning models that learn such statistical correlations in a self-supervised manner from low-resolution and high-resolution pairs that exist in a training dataset. A successful super-resolution model should be able to efficiently represent the high-resolution outputs, effectively scale to large spatio-temporal domains as in a realistic problem setting, and allow for incorporating physical constraints in the form of PDEs to regular-

*Denotes equal contribution.
[1]source code available: https://github.com/maxjiang93/space_time_pde

ize the outputs to physically valid solutions. Furthermore, in order for a learning-based methodology to be applied to real problems such as Computational Fluid Dynamics (CFD) simulations and climate modeling, the methodology needs to address the High Performance Computing (HPC) challenges of scalability to large scale computing systems, for both the training and inference stages.

To this end, we propose MeshfreeFlowNet, a novel physics-constrained, deep learning based super-resolution framework to generate continuous (grid-free) spatio-temporal solutions from the low-resolution inputs. MeshfreeFlowNet first maps the low-resolution inputs to a localized latent context grid using a convolutional encoder, which can then be continuously queried at arbitrary resolutions. In summary, our main contributions are as follows:

- We propose MeshfreeFlowNet, a novel and efficient physics-constrained deep learning model for super-resolution tasks.
- We implement a set of physics-based metrics that allow for an objective assessment of the reconstruction quality of the super-resolved high-resolution turbulent flows.
- We empirically assess the effectiveness of the MeshfreeFlowNet framework on the task of super-resolving turbulent flows in the Rayleigh–Bénard convection problem, showing a consistent and significant improvement in recovering key physical quantities of interest in super-resolved solutions over competing baselines.
- We demonstrate the scalability of our framework to larger and more challenging problems by providing a large scale implementation of MeshfreeFlowNet that scales to up to 128 GPUs while retaining a $\sim 97\%$ scaling efficiency.

2 Related Works

Recently, deep learning models have been applied for studying fluid flow problems in different applications. In particular, a certain class of research works has studied flow problems on a grid representation where dynamic flow properties (*e.g.*, velocity, pressure, etc.) could be considered as *structured* data similar to images. For instance, Guo et al. [23] used convolutional architectures for approximating non-uniform steady laminar flow around 2D and 3D objects. Similarly, Bhatnagar et al. [24] used convolutional neural networks for prediction of the flow fields around airfoil objects and for studying aerodynamic forces in a turbulent flow regime. Zhu and Zabaras [25] used Bayesian deep convolutional neural networks to build surrogate flow models for the purpose of uncertainty quantification for flow applications in heterogeneous porous media.

Alternatively, a different area of research has attempted to solve PDEs governing different physical phenomena by using simple fully connected deep neural networks. In order to enforce physical constraints, the PDEs under consideration are added as a term to the loss function at the training stage. For instance, E and Yu [26] used fully connected deep neural networks with skip connections for solving different classes of PDEs such as Poisson equations and eigenvalue problems. Bar and Sochen [27] used fully connected deep neural networks for solving forward and inverse problems in the context of electrical impedance tomography governed by elliptical PDEs. Long et al. [28] proposed a framework to uncover unknown physics in the form of PDEs by learning constrained convolutional kernels. Raissi et al. [29] applied fully connected neural networks for solving PDEs in a forward or inverse setup. Smith et al. [30] deployed residual neural networks to solve the Eikonal PDE equation and perform ray tracing for an application in earthquake hypo-center inversion and seismic tomography. Using their proposed framework they solved PDEs in different contexts such as fluids, quantum mechanics, seismology, reaction-diffusion systems, and the propagation of nonlinear shallow-water waves.

More recently, alternative representations for spatial functions have been explored. Li et al. [31] proposed using message passing on graph networks in order to map between input data of PDEs and their solutions. They showed that the learned networks can generalize between different PDE approximation methods (*e.g.*, Finite Difference and Finite Element methods) and between approximations that correspond to different levels of discretization. Similar to our MeshfreeFlowNet framework in using a latent context grid that can be continuously decoded into an output field, Jiang et al. [32] used such representations for the computer vision task of reconstructing 3D scenes, where latent context grids are decoded into continuous implicit functions that represent the surfaces of different geometries.

In the area of fluid dynamics, for turbulent flow predictions, Wang et al. [33] developed a physics-informed deep learning framework. They introduced the use of trainable spectral filters coupled with Reynolds-averaged Navier-Stokes (RANS) and Large Eddy Simulation (LES) models followed by a convolutional architecture in order to predict turbulent flows. Jiang et al. [34] presented a differentiable physics layer within the neural networks using spectral methods in order to enforce hard linear constraints. By performing a linear projection in the spectral space, they were able to enforce a divergence-free condition for the velocity. Accordingly, they could conserve mass within the physical system both locally and globally, allowing the super-resolution of turbulent flows to be statistically accurate.

In computer vision applications, the *super-resolution* (SR) process has been proposed in the context of reconstructing high-resolution (HR) videos/images from the low-resolution (LR) ones. In the literature, different classical super-resolution methods have been proposed such as prediction-based methods [35, 36], edge-based methods [37, 38], statistical methods [39, 40], patch-based methods [41, 42] and sparse representation methods [43, 44]. Most recently, deep learning based super-resolution models have also been actively explored and differ mainly in their types of specific intended application, architectures, loss functions, and learning principles [45–50].

3 Preliminaries

In this section, we provide a detailed explanation of the problem set up, notations, data sets, along with the learning paradigm and evaluation metrics.

3.1 Problem Setup

Consider a space and time dependent partial differential equation (PDE) with a unique solution, written in the general form as

$$\Gamma \boldsymbol{y} = s \quad \forall \boldsymbol{x} \in \Omega\,, \tag{1a}$$
$$\Lambda \boldsymbol{y} = b \quad \forall \boldsymbol{x} \in \partial\Omega\,, \tag{1b}$$

defined on a spatio-temporal domain $\Omega := \Omega_{Spatial} \times [0, T] \in \mathbb{R}^{d+1}$ where $\Omega_{Spatial}$ represents a d dimensional spatial domain, and $[0, T]$ represents the one dimensional temporal domain. Here the Γ is the operator defining the PDE within the domain (Ω), s is the source function, b is the boundary condition function, Λ is the operator defining the PDE on the boundary ($\partial\Omega$), and the solution to the PDE in Eqn. 1 is $\boldsymbol{y} : \Omega \to \mathbb{R}^m$.

For a partial differential equation (Eqn. 1), a problem instance is realized with a given source and boundary condition functions s and b. For a pair (s, b), we consider $\mathcal{H}$ to be an operator that generates the high-resolution solution $\boldsymbol{y}_{\mathcal{H}}$, *i.e.*, $\boldsymbol{y}_{\mathcal{H}} = \mathcal{H}(s, b)$, and $\mathcal{L}$ to be an operator that produces the low-resolution solution $\boldsymbol{y}_{\mathcal{L}}$, *i.e.*, $\boldsymbol{y}_{\mathcal{L}} = \mathcal{L}(s, b, \boldsymbol{y}_{\mathcal{H}})$. Consider a compact normed space of operators $\Pi_{\mathcal{F}}$, a set of operators $\mathcal{F} \in \Pi_{\mathcal{F}}$, mapping low-resolution solutions to high-resolution ones. For a given compact normed space of functions $\Pi_{u,b}$, let $\varepsilon\left(\Pi_{\mathcal{F}}, \Pi_{s,b}\right)$ denote the approximation gap with respect to $\Pi_{\mathcal{F}}$, *i.e.*,

$$\varepsilon\left(\Pi_{\mathcal{F}}, \Pi_{u,b}\right) := \max_{(b,u)\in\Pi_{u,b}} \min_{\mathcal{F}\in\Pi_{\mathcal{F}}} \left\| \mathcal{H}(s,b) - \mathcal{F}\left(\mathcal{L}(s, b, \boldsymbol{y}_{\mathcal{H}})\right) \right\|, \tag{2}$$

with continuous approximation error $\|\mathcal{H}(s,b) - \mathcal{F}\left(\mathcal{L}(s,b,\boldsymbol{y})\right)\|$ in F, s, b. Here the norm is with respect to a desired $L^p(\mu)$ space where $p \geq 1$ and μ is a preferred measure[2]. In this work, given a problem instance specified with (s, b), we are interested in learning $\mathcal{F}$ using parametric models. In order to map a low-resolution solution to its corresponding super-resolution with low approximation error, we require $\varepsilon\left(\Pi_{\mathcal{F}}, \Pi_{s,b}\right)$ to be small and comparable with a desired error level. Therefore, given a problem set $\Pi_{u,b}$, a proper and rich class of operators, $\Pi_{\mathcal{F}}$, allows for a desirable approximation error.

For a given PDE in Eqn 1, defined on a hyper-rectangular domain in $\mathbb{R}^{d+1}$, we consider a spatio-temporal grid of resolutions $(n_1, n_2, \ldots, n_{d+1})$ with n_{d+1} denoting the number of discretized points in time domain, and a data set $\mathcal{D} := \{(\boldsymbol{x}^i, \boldsymbol{y}^i)\}_{i=1}^{n_1 \cdot n_2 \cdot \ldots \cdot n_{d+1}}$, of points and their solution values.

[2]Note that in the max-min game of the Eqn. 2, the action of the environment player, (s, b), is revealed to the approximator player to choose $\mathcal{F}$. Therefore, since the game is in the favor of the approximating player, the value of the game, $\varepsilon\left(\Pi_{\mathcal{F}}, \Pi_{s,b}\right)$ can be much smaller than its min-max counterpart. We require the min-max value to be desirably small when we aim to have a single model $\mathcal{F}$ to perform well on a set of designated problems.

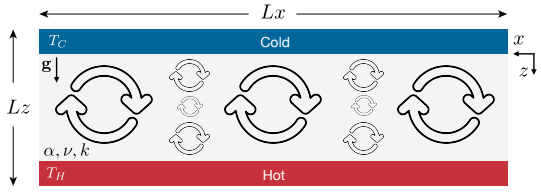


Figure 1: Configuration of the Rayleigh–Bénard instability problem - Cold and hot plates have temperatures of T_C and T_H and separated with a distance of L. Flow parameters α, ν, κ respectively are the thermal expansion coefficient, kinematic viscosity, and thermal diffusivity. g is the gravity acceleration in the $z-$direction, and L_x, L_z respectively are the length of the plates and their separation distance. x and z respectively refer to the coordinates of the Cartesian frame of reference.

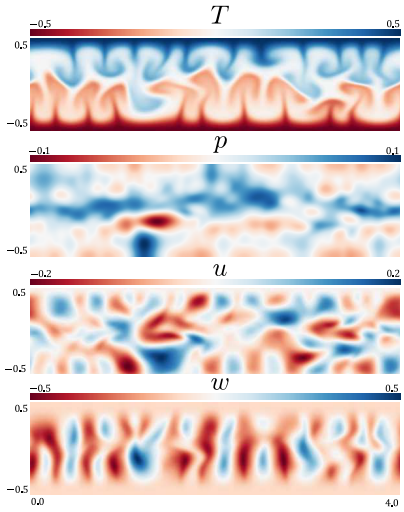


Figure 2: An illustration of a typical solution to the Rayleigh–Bénard problem (Eqns. (3a)-(3c)). The contours respectively show the solution for temperature (T), pressure (p), and the two velocity components (u and w). For this simulation case $Pr = 1$, $Ra = 10^6$, and $L_x = 4L_z = 4$ $[m]$. The spatial resolution is $n_x = 4 \times n_z = 512$, and upon an adaptive time stepping scheme the presented solution above is obtained at $t = 25$ $[s]$.

3.2 Dataset Overview

In this work, we generate the dataset as the solution to a classical fluid dynamics system with a chaotic nature. We consider the well-known Rayleigh–Bénard instability problem in 2D where a static bulk fluid (kinematic viscosity ν, thermal diffusivity α) is initially occupying the space between two horizontal plates (see Fig. 1). The lower and upper plates are considered to be respectively hot and cold with temperatures of T_H and T_C. Gradually the temperature of the bulk fluid adjacent to the hot (cold) plate increases (decreases) and due to the buoyancy effects and density gradient the bulk fluid ascends (descends), leading to the formation of vortices and growth of flow instability in a chaotic and turbulent regime. The governing partial differ-

ential equations for the Rayleigh–Bénard instability problem are

$$\nabla \cdot \mathbf{u} = 0\,, \tag{3a}$$

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} + \nabla p - T\hat{z} - R^*\nabla^2\mathbf{u} = 0\,, \tag{3b}$$

$$\frac{\partial T}{\partial t} + \mathbf{u}\cdot\nabla T - P^*\nabla^2 T = 0\,, \tag{3c}$$

where $P^* = (Ra\,Pr)^{-1/2}$ and $R^* = (Ra/Pr)^{-1/2}$ with Ra and Pr being the Rayleigh and Prandtl numbers respectively defined as $Ra = g\alpha\Delta T L^3\nu^{-1}\kappa^{-1}$ and $Pr = \nu\kappa^{-1}$, with g, α, ν, κ, ΔT, and L respectively being the gravity acceleration, thermal expansion coefficient, kinematic viscosity, thermal diffusivity, temperature difference between hot and cold plates, and the separation length between the plates. From a physical perspective Ra quantifies the balance between the gravitational forces and viscous damping, and Pr quantifies the ratio between momentum diffusivity and thermal diffusivity (balance between heat convection and conduction).

We use the Dedalus framework [51] in order to numerically solve the system of Equations (3a)-(3c) using the spectral method approach. We solve Equations (3a)-(3c) for a duration of t_f in time with a time step size of Δt. In a coordinate system of x-axis, and z-axis, we consider a plate length L_x and a separation distance L_z, and discretize the domain with n_x and n_z points respectively in the x and z directions.

For the simulation cases, we consider Rayleigh and Prandtl numbers respectively in the range of $Ra \in [10^4, 10^8]$ and $Pr \in [0.1, 10]$. Upon solving the system of Eqns. (3a)-(3c), we create a high-resolution Rayleigh–Bénard simulation dataset $\mathcal{D}_H$, unless otherwise mentioned, with a spatial resolution of $n_x = 4 \times n_z = 512$, and a temporal resolution of $n_t = 400$ (upon adaptive time stepping). We consider a normalized domain size of unit length in z-direction with a domain aspect ratio of 4, *i.e.*, $L_x = 4 \times L_z = 4$ $[m]$, and solve the Rayleigh–Bénard problem for a duration of 50 $[s]$. Then we create a low-resolution dataset $\mathcal{D}_L$, by downsampling the high-resolution data in both space and time. We use downsampling factors of $d_t = 4$ and $d_s = 8$ for creating the low-resolution data in the temporal and spatial dimensions respectively.

3.3 Evaluation Metrics

In this work, we use multiple physical metrics, each accounting for different aspects of the flow field, in order to report the evaluation of the MeshfreeFlowNet model for super-resolving low-resolution data. As the specific metrics of evaluation, we report the Normalized Mean Absolute Error (NMAE) and R2 score for the physical metrics between the ground truth and the predicted high-resolution data. Such physical metrics are listed in the following.

- *Total Kinetic Energy (E_{tot})*: the kinetic energy per unit mass associated with the flow is defined as the total kinetic energy and can be expressed as $E_{tot} = \frac{1}{2}\langle u_i u_i\rangle$.
- *Root Mean Square Velocity (u_{rms})*: the square root of the scaled total kinetic energy is defined as the root mean square (RMS) velocity as $u_{rms} = \sqrt{(2/3)E_{tot}}$.
- *Dissipation (ε)*: is the rate at which turbulence kinetic energy is converted into thermal internal energy by molecular viscosity and defined as $\varepsilon = 2\nu\langle S_{ij}S_{ij}\rangle$ with S_{ij} and ν respectively being the rate of strain tensor and the kinematic viscosity.
- *Taylor Microscale (λ)*: is the intermediate length scale at which viscous forces significantly affect the dynamics of turbulent eddies and defined as $\lambda = \sqrt{15\,\nu\,u_{rms}^2\,\varepsilon^{-1}}$. Length scales larger than the Taylor microscale (*i.e.*, inertial range) are not strongly affected by viscosity. Below the Taylor microscale (*i.e.*, dissipation range) the turbulent motions are subject to strong viscous forces and kinetic energy is dissipated into heat.
- *Taylor-scale Reynolds (Re_λ)*: is defined as the ratio of RMS inertial forces to viscous forces and is expressed as $Re_\lambda = u_{rms}\,\lambda\,\nu^{-1}$.
- *Kolmogorov Time (τ_η) and Length (η) Scales*: Kolmogorov microscales are the smallest scales in turbulent flows where viscosity dominates and the turbulent kinetic energy dissipates into heat. Kolmogorov time and length scale can be respectively expressed as $\tau_\eta = \sqrt{\nu/\varepsilon}$ and $\eta = \nu^{3/4}\,\varepsilon^{-1/4}$.
- *Turbulent Integral Scale (L)*: is a measure of the average spatial extent or coherence of the fluctuations and can be expressed as $L = \frac{\pi}{2u_{rms}^2}\int \frac{E(k)}{k}dk$.
- *Large Eddy Turnover Time (T_L)*: is defined as the typical time scale for an eddy of length scale L to undergo significant distortion and is also the typical time scale for the transfer of energy from scale L to smaller scales, since this distortion is the mechanism for energy transfer and expressed as $T_L = L/u_{rms}$.

3.4 Systems, Platforms and Configuration

In order to further illustrate the feasibility of utilizing our proposed MeshfreeFlowNet framework for large physical systems requiring processing orders of magnitude more computation, here we study the scalability of MeshfreeFlowNet. We scale our model on a GPU stack on the Cori supercomputer at the National Energy Research Scientific Computing Center (NERSC). We use data distributed parallelism with synchronous gradient descent, and test performance up to 16 nodes with a total of 128 GPUs.

In a data-parallel distribution strategy, the model is replicated over all GPUs. At every step, each GPU receives its own random batch of the dataset to calculate the local gradients. Gradients are averaged across all devices with an `all_reduce` operation. To achieve better scaling efficiency, the communication of one layer's gradients is overlapped with the `backprop` computation of the previous layer. PyTorch `torch.distributed` [52] package provides communication primitives for multiprocess parallelism with different collective communication backends. We use

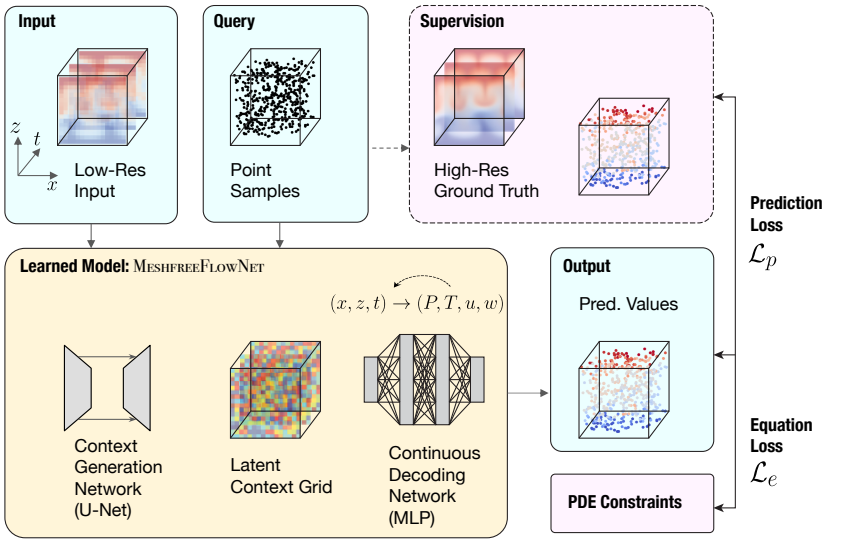


Figure 3: Schematic for the training pipeline for MeshfreeFlowNet model for continuous space-time super-resolution. An input low-resolution grid is fed to the Context Generation Network that creates a Latent Context Grid. A random set of points in the corresponding space-time domain is sampled to query the Latent Context Grid, and the physical output values at these query locations can be continuously decoded using a Continuous Decoding Network, implemented as a Multilayer Perception. Due to the differentiable nature of the MLP, any partial derivatives of the output physical quantities with respect to the input space-time coordinates can be effectively computed via backpropagation, that can be combined with the PDE constraints to produce an Equation Loss. On the other hand, the predicted value can be contrasted with the ground truth value at these locations produced by interpolating the high-resolution ground truth to produce a Prediction Loss. Gradients from the combined losses can be backpropagated to the network for training.

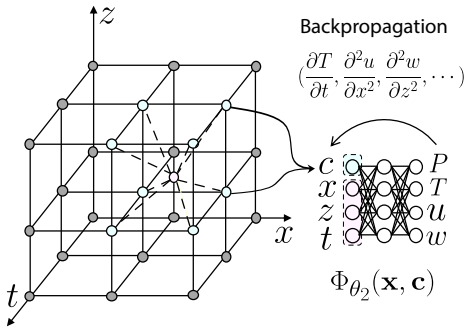


Figure 4: Schematic for the continuous decoding module for MeshfreeFlowNet. The continuous decoding network is a Multilayer Perception that inputs the spatio-temporal coordinates of a query point, along with a latent context vector, and is decoded into the required physical channels of interest. Since each query point falls into a cell bounded by 8 neighboring vertices, the query is performed 8 times, each using a different latent context vector and a different relative spatio-temporal coordinate with respect to each vertex. The values are then interpolated using trilinear interpolation to get the final value at the query point.

NVIDIA's Collective Communications Library (NCCL) [53] backend which gave us the best performance when running on GPUs, both within and across multiple nodes. The PyTorch `torch.nn.parallel.DistributedDataParallel` [54] wrapper, which we use for the results in this paper, builds on top of the `torhch.distributed` package to provide efficient data-parallel distributed training. With this setup, we achieve more than 96% throughput efficiency on 16 Cori GPU nodes, 128 GPUs in total. Cori has 8 V100 (Volta) GPUs per node, the GPUs are interconnected with NVLinks in hybrid cube-mesh topology [55]. The nodes are equipped with Mellanox MT27800 (ConnectX-5) EDR InfiniBand network cards.

4 MeshfreeFlowNet

In this work, we propose the MeshfreeFlowNet framework as a novel computational algorithm for constructing the super-resolution solutions to partial differential equations using their low-resolution counterpart solutions.

A schematic for the training pipeline for the MeshfreeFlowNet model is presented in Fig. 3. The model inputs a low-resolution spatio-temporal grid that can be acquired from a PDE solver, along with query locations that can be any continuous value within the range of the input domain. MeshfreeFlowNet produces a predicted output vector at each query location. MeshfreeFlowNet consists of two subnetworks, namely the *Context Generation Network*, and the *Continuous Decoding Network*. The Context Generation Network produces a *Latent Context Grid*, which is a learned localized representation of the function. The Latent Context Grid contains latent context vectors $\boldsymbol{c} \in \mathbb{R}^{n_c}$ at each grid vertex, which along with spatio-temporal coordinates $\boldsymbol{x} := \{x, z, t\}$, can be concatenated and fed into the Continuous Decoding Network, modeled as a Multilayer Preceptron (MLP), to generate the physical outputs $\boldsymbol{y} \in \mathbb{R}^m$. For training the model, two loss terms, namely the prediction loss and the equation loss are used. We present a more detailed architectural breakdown for our method and our baseline method in Fig. 5. We discuss the details of the Context Generation Network (Sec. 4.1), Continuous Decoding Network (Sec. 4.2), and loss functions (Sec. 4.3) in the following subsections.

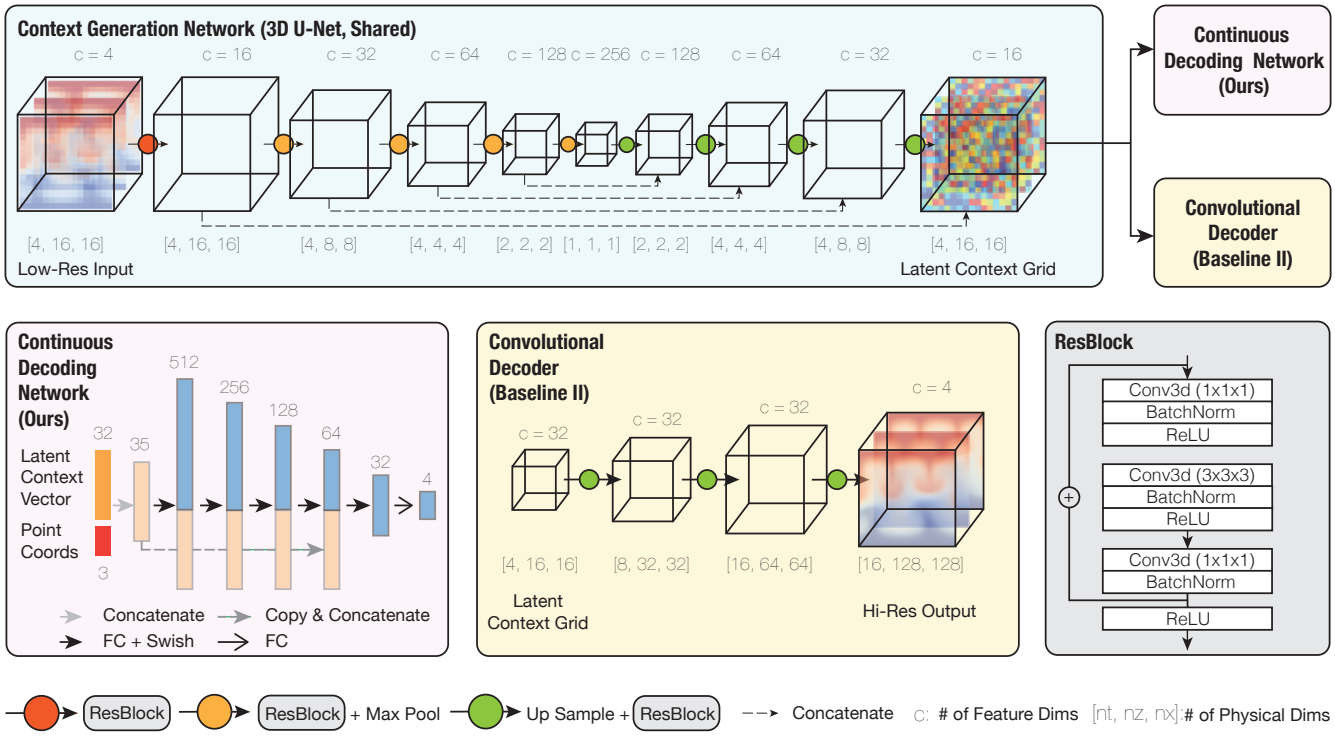


Figure 5: A schematic for the architecture of the MeshfreeFlowNet framework. MeshfreeFlowNet consists of two end-to-end trainable modules: the Context Generation Network, and the Continuous Decoding Network. The Baseline (II) method that we exhaustively compare with share the same 3D U-Net structure in the encoding arm, but instead uses a convolutional decoder for producing the final output. In comparison, the MeshfreeFlowNet allows for continuous output that can be sampled at arbitrary space-time locations.

	100×NMAE (R2)									
γ	E_{tot}	u_{rms}	ε	λ	Re_λ	τ_η	η	L	T_L	avg. R2
0	0.667 (0.9991)	0.768 (0.9987)	0.666 (0.9991)	0.545 (0.9985)	0.444 (0.9989)	0.753 (0.9981)	0.752 (0.9984)	0.837 (0.9968)	0.455 (0.995)	0.9986
0.0125	0.650 (0.9990)	0.616 (0.9992)	0.639 (0.9991)	0.457 (0.9989)	0.435 (0.9986)	0.589 (0.9988)	0.588 (0.9990)	**0.670** (0.9982)	0.432 (0.9994)	0.9989
0.025	0.671 (0.9993)	0.699 (0.9992)	0.671 (0.9993)	0.454 (0.9990)	**0.332** (0.9994)	0.705 (0.9985)	0.698 (0.9988)	0.781 (0.9973)	0.430 (0.9996)	0.9989
$\gamma^* = 0.05$	**0.621** (0.9993)	**0.603** (0.9992)	**0.617** (0.9993)	**0.431** (0.9992)	0.429 (0.9989)	**0.461** (0.9994)	**0.483** (0.9994)	0.857 (0.9972)	**0.418** (0.9996)	**0.9991**
0.1	3.209 (0.9894)	1.790 (0.9954)	3.015 (0.9907)	1.024 (0.9907)	1.013 (0.9917)	1.628 (0.9902)	1.636 (0.9925)	1.970 (0.9900)	2.443 (0.9922)	0.9914
0.2	1.396 (0.9979)	5.059 (0.9679)	1.373 (0.9978)	3.964 (0.8835)	3.553 (0.8781)	4.329 (0.9270)	4.415 (0.9436)	3.589 (0.9739)	1.636 (0.9962)	0.9518
0.4	3.560 (0.9854)	13.270 (0.7909)	3.533 (0.9854)	10.119 (-0.177)	10.056 (-0.366)	9.9149 (0.466)	10.392 (0.5923)	5.787 (0.9343)	1.751 (0.9917)	0.5780
0.8	11.433 (0.8507)	15.727 (0.6805)	11.171 (0.8701)	13.502 (-1.610)	14.664 (-2.556)	11.778 (0.0830)	12.385 (0.3005)	2.092 (0.9819)	8.033 (0.9231)	0.0582
1.0	13.617 (0.7954)	17.970 (0.6031)	13.441 (0.8206)	15.080 (-2.536)	16.508 (-3.746)	12.868 (-0.121)	13.631 (0.1487)	6.165 (0.9179)	8.546 (0.9154)	-0.2447

Table 1: Normalized Mean Absolute Error (NMAE) and R2-score of the flow-based evaluation metrics evaluated for the predicted vs. the ground truth high-resolution validation data. γ refers to the coefficient of the Equation loss ($\mathcal{L}_e$) in the total loss function (Eqn. 10).

4.1 Context Generation Network

$$\mathcal{G} = \Psi_{\theta_1}(\mathcal{D}_L)\,, \tag{4}$$

The *Context Generation Network* is a convolutional encoder that produces a Latent Context Grid from the low-resolution physical input $\mathcal{D}_L$. Denote this network as

where θ_1 is the set of learnable parameters associated with this network, $\mathcal{G}$ is the generated context grid, $\mathcal{G} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_{d+1} \times n_c}$ where n_c is the number of latent chan-

nels or the dimensions of each latent context vector. It is worth noting that the function Ψ_{θ_1} can be applied on a small fixed-size sub-window of Ω, since the model is fully-convolutional. By applying it to the input in a fully-convolutional manner, the size of the domain at test time (both spatial and temporal sizes) can be orders of magnitude greater than the dimensions of $\mathcal{D}_L$.

In this work, we implement the Context Generation Network as a 3D variant of the *U-Net* architecture which was originally proposed by Ronneberger et al. [56]. U-Net has successfully been applied for different computer vision tasks that involve dense localized predictions such as image style transfer [57], image segmentation [58, 59], image enhancement [60], image coloring [61, 62], and image generation [63, 64]

Different from the original U-Net architecture, we replace the 2D convolutions with 3D counterparts and utilize residue blocks instead of individual convolution layers for better training properties of the network. The U-Net comprises of a contractive part followed by an expansive part. The contractive part is composed of multiple stages of convolutional residue blocks, each followed by a max-pooling layer (of stride 2). Each residue block consists of 3 convolution layers (1x1, 3x3, 1x1) interleaved with batch normalization layers and ReLU activation. The expansive part mirrors the contractive part, replacing max-pooling with nearest neighbor upsampling. In between the layers with similar grid sizes within the contractive and expansive parts, a skip connection concatenates the features from the contractive layer with the features in the expansive layer as the input to the subsequent layers in the contractive part in order to preserve the localized contextual information.

4.2 Continuous Decoding Network

One unique property of our super-resolution methodology is that the output is continuous instead of discrete. This removes the limitations in output resolution, and additionally, it allows for an effective computation of the gradients of predicted output physical quantities, enabling an easy way of enforcing PDE-based physical constraints.

The continuous decoding network can be implemented using a simple Multilayer Perceptron, see Fig. 4. For each query, denote the spatio-temporal query location to be $\boldsymbol{x}_i$ and the latent context grid to be $\mathcal{G} := \{(\boldsymbol{x}_j, \boldsymbol{c}_j); j \leq ||\mathcal{G}||\}$, where $(\boldsymbol{x}_j, \boldsymbol{c}_j)$ are the spatio-temporal coordinates and the latent context vector for the j-th vertex of the grid. Denote $\mathcal{N}_i$ as the set of neighboring vertices that bound x_i, where for a $(d+1)$ dimensional spatio-temporal grid $||\mathcal{N}_i|| = 2^{d+1}$. Denote the continuous decoding network, implemented as a Multilayer Perception as

$$\Phi_{\theta_2}(\boldsymbol{x}, \boldsymbol{c}) \,, \tag{5}$$

where θ_2 is the set of trainable parameters of the Multilayer Perception network. The query value at $\boldsymbol{x}$ with respect to the shared network $\Phi_{\theta_2}(\boldsymbol{x})$ and the latent context grid $\mathcal{G}$ can be calculated as

$$\mathcal{C}(\boldsymbol{x}_i, \mathcal{G}, \Phi_{\theta_2}) = \sum_{j \in \mathcal{N}_i} w_j \Phi_{\theta_2}(\frac{\boldsymbol{x}_i - \boldsymbol{x}_j}{\Delta \boldsymbol{x}}, \boldsymbol{c}_j) \,, \tag{6}$$

where $\sum_{j \in \mathcal{N}_i} w_j = 1$, w_j is the trilinear interpolation weight with respect to the bounding vertex j, $\Delta \boldsymbol{x} := \{\Delta x, \Delta z, \Delta t\}$ is the stencil size corresponding to the discretization grid vertices.

Since the Continuous Decoding Network is implemented as an MLP, arbitrary spatio-temporal derivatives of the output quantities: $\Gamma \boldsymbol{y}$ can be effectively computed via back-propagation through the MLP. Denote the approximation of the derivative operator to be Γ_Φ. We combine the partial derivatives to compute the equation loss as the norm of the residue of the governing equations. In the continuous decoding network we consider two infinitely differentiable activation functions namely Softplus and Swish, where we have found the results obtained by Swish to outperform the ones obtained by Softplus.

4.3 Loss Function

We use a weighted combination of two losses to train our MeshfreeFlowNet network: the norm of the difference between the predicted physical outputs and the ground truth physical outputs, which we refer to as *Prediction Loss*, and the norm of the residues of the governing PDEs, which we refer to as *Equation Loss*.

Denote the set of sample locations within a mini-batch $\mathcal{B}$ of training samples to be $\{(\boldsymbol{x}_j^i, \boldsymbol{y}_j^i, \mathcal{D}_L^i); i \in \mathcal{B}, j \in \mathcal{B}^i\}$ where $\mathcal{B}^i$ is the mini-batch of point samples for the i-th low-resolution input, $\boldsymbol{y}$ is the vector that represents the ground truth physical output quantities. In the case of the Rayleigh-Bénard Convection example in this study, we have $\boldsymbol{y} := \{P, T, u, v\}$ where P, T, u, v are the pressure, temperature, x-velocity and y-velocity terms respectively. The super-resolution for the learned model queried at $\boldsymbol{x}$ conditioning on low-resolution input $\mathcal{D}_L$ is

$$\hat{\boldsymbol{y}}(\boldsymbol{x}, \mathcal{D}_L; \theta_1, \theta_2) = \mathcal{C}(x, \Psi_{\theta_1}(\mathcal{D}_L), \Phi_{\theta_2}) \,, \tag{7}$$

where $\hat{\boldsymbol{y}}$ is the predicted output vector. The *prediction loss* $\mathcal{L}_p$ for a mini-batch can be formulated as

$$\mathcal{L}_p = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \frac{1}{||B^i||} \sum_{j \in \mathcal{B}^i} ||\boldsymbol{y}_j^i - \hat{\boldsymbol{y}}_j^i||_l \,, \tag{8}$$

where $||\cdot||_l$ is the Frobenius l-norm of the difference. We use the L1 Norm for computing the prediction loss. The *Equation loss* $\mathcal{L}_e$ for a mini-batch can be formulated as

$$\mathcal{L}_e = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \frac{1}{||B^i||} \sum_{j \in \mathcal{B}^i} ||\Gamma_\Phi \hat{\boldsymbol{y}}_j^i - s||_l \,, \tag{9}$$

which is the norm of the PDE equation residue, using the PDE definition in Eqn. 1. Finally, a single loss term for training the network can be represented as a weighted sum of the two loss terms as

$$\mathcal{L} = \mathcal{L}_p + \gamma \mathcal{L}_e \,, \tag{10}$$

where γ is a hyperparameter for weighting the equation loss.

Model	100×NMAE (R2) E_{tot}	u_{rms}	ε	λ	Re_λ	τ_η	η	L	T_L	avg. R2
Baseline (I)	69.6360 (-19.7894)	3470.132 (-57.7717)	76.338 (-14)	78.164 (-11096)	75.729 (-4934)	77.410 (-12599)	122.55 (-3147)	137.376 (-0.5190)	109.398 (-3.0092)	-3541
Baseline (II)	6.489 (0.9557)	8.769 (0.8967)	6.144 (0.9593)	3.903 (0.9490)	2.489 (0.9711)	5.584 (0.9382)	6.019 (0.94019)	2.902 (0.9597)	5.076 (0.9644)	0.9482
MeshfreeFlowNet, $\gamma = 0$	0.667 (0.9991)	0.768 (0.9987)	0.666 (0.9991)	0.545 (0.9985)	0.444 (0.9989)	0.753 (0.9981)	0.752 (0.9984)	**0.837** (0.9968)	0.455 (0.995)	0.9986
MeshfreeFlowNet, $\gamma = \gamma^*$	**0.621** (0.9993)	**0.603** (0.9992)	**0.617** (0.9993)	**0.431** (0.9992)	**0.429** (0.9989)	**0.461** (0.9994)	**0.483** (0.9994)	0.857 (0.9972)	**0.418** (0.9996)	**0.9991**

Table 2: Comparison between the performance of MeshfreeFlowNet framework for super-resolving the low-resolution data vs. two baseline models.

# Dataset(s) ($\gamma = \gamma^*$)	100×NMAE (R2) E_{tot}	u_{rms}	ε	λ	Re_λ	τ_η	η	L	T_L	avg. R2
1	0.621 (0.9993)	0.603 (0.9992)	0.617 (0.9993)	0.431 (0.9992)	0.429 (0.9989)	**0.461** (0.9994)	**0.483** (0.9994)	0.857 (0.9972)	0.418 (0.9996)	0.9991
10	**0.609** (0.9995)	**0.599** (0.9993)	**0.603** (0.9991)	**0.428** (0.9991)	**0.411** (0.9987)	0.470 (0.9990)	0.497 (0.9991)	**0.673** (0.9998)	**0.345** (0.9998)	**0.9993**

Table 3: For a MeshfreeFlowNet model that has been respectively trained on 1 dataset and 10 datasets each having a different initial condition, the super-resolution performance evaluation is reported on a dataset with an unseen initial condition.

Ra	100×NMAE (R2) E_{tot}	u_{rms}	ε	λ	Re_λ	τ_η	η	L	T_L	avg. R2
1×10^4	1.180 (0.9973)	0.401 (0.99963)	0.709 (0.9992)	44.6435 (0.2038)	1.357 (0.9977)	2.983 (0.4588)	2.374 (0.8726)	70.34 (0.0104)	0.137 (0.9996)	0.7266
1×10^5	0.670 (0.9987)	0.537 (0.9995)	0.611 (0.9990)	0.450 (0.9992)	0.472 (0.9989)	0.589 (0.9992)	0.574 (0.9993)	0.713 (0.9990)	0.331 (0.9996)	0.9992
5×10^6	1.043 (0.9981)	1.0601 (0.9927)	1.017 (0.9963)	1.054 (0.9963)	0.954 (0.9961)	1.441 (0.9921)	1.692 (0.9924)	0.792 (0.9934)	0.441 (0.993)	0.9997
1×10^7	1.395 (0.9958)	3.044 (0.9743)	1.405 (0.9957)	2.767 (0.9691)	1.821 (0.9862)	2.900 (0.9714)	2.924 (0.9725)	2.9817 (0.9749)	1.077 (0.9969)	0.9819
1×10^8	3.184 (0.9870)	4.454 (0.9664)	3.221 (0.9869)	6.362 (0.8826)	7.471 (0.8110)	6.051 (0.9219)	5.704 (0.9375)	3.658 (0.9692)	2.027 (0.9930)	0.9395

Table 4: For a MeshfreeFlowNet model that has been trained on 10 datasets each having a different boundary condition (Rayleigh number) as $Ra \in [2, 90] \times 10^5$ with $Pr = 1$, the super-resolution performance evaluation is reported for: a Rayleigh number within the range of boundary conditions of the training sets (*i.e.*, $Ra = 5 \times 10^6$), Rayleigh numbers slightly below and above the range of boundary conditions of the training sets (*i.e.*, $Ra = 1 \times 10^5$ and $Ra = 1 \times 10^7$ respectively), and Rayleigh numbers far below and above the range of boundary conditions of the training sets (*i.e.*, $Ra = 1 \times 10^4$ and $Ra = 1 \times 10^8$ respectively).

5 Experiments

In all the experiments, we use an Adam optimizer with learning rate of 10^{-2}, and l_1 regularization for the loss function, 3000 random samples per epoch, and train for 100 epochs.

5.1 Prediction Loss vs. Equation Loss

In this part, we investigate the influence of the importance given to the Equation loss and the prediction loss (see, Section 4.3) on the performance of MeshfreeFlowNet. As presented in Eqn. 10, the total loss ($\mathcal{L}$) comprises of the Prediction loss ($\mathcal{L}_p$) and the Equation loss ($\mathcal{L}_e$) where the Equation loss is weighted with a scaling coefficient γ. Accordingly, we study the influence of the hyperparameter γ in the loss function given in Eqn. 10 on the accuracy of MeshfreeFlowNet. For this purpose, we consider $\gamma \in \{0, 0.0125, 0.025, 0.05, 0.1, 0.2, 0.4, 0.8, 1.0\}$. Considering both Softplus and Swish activation functions in the continuous decoding network, we have found that the results obtained by the latter one outperform the ones obtained by the former one, for which, the values of the evaluation metrics for MeshfreeFlowNet trained with each of the γ values are presented in Table 1. $\gamma = 0$ indicates a loss function which only depends on the Prediction loss, and the physical aspects (PDE imposed constraints) of the predicted high-resolution data are not accounted for. As presented in Table 1, the best performance on the validation set is achieved for a loss function with the Equation loss weighting coefficient of $\gamma = 0.05$. Allover this work we refer to this optimum weighting coefficient as γ^* and perform the training tasks using the loss function in Eqn. 10 where the Equation loss is weighted with $\gamma^* = 0.05$. As presented in Table 1, a model that is trained with $\gamma = 0$, which only focuses on the data (*i.e.*, uses Prediction loss only) and does not account for the physics and the PDE constraints (*i.e.*, ignores Equation loss) underperforms compared to the trained model with $\gamma = \gamma^*$. On the other hand models trained with a significant focus on the physical constraints only (*i.e.*, large γ) underper-

form in super-resolving the low-resolution data. In general, a balance between the focus of the MeshfreeFlowNet on the model and the physical constraints leads to an optimal super-resolution performance. In achieving that, in Eqn. 10, the Prediction loss ($\mathcal{L}_p$) captures the global structure of the data and the Equation loss ($\mathcal{L}_e$) further guides the model in accurately reconstructing the local structure of the data.

5.2 MeshfreeFlowNet vs. Baseline Models

In this section, we present a comparison between the performance of our proposed MeshfreeFlowNet framework for super-resolving low-resolution data against two baselines, namely: a classic trilinear interpolation algorithm (Baseline (I)), and a deep learning based 3D U-Net model (Baseline (II)). Specifically for the 3D U-Net model for Baseline (II), we use the same U-Net backbone as in our MeshfreeFlowNet framework, with the difference being that while MeshfreeFlowNet uses the 3D U-Net to generate a latent context grid, the baseline U-Net continues with 3D up-sampling and transpose-convolution operations up to the target high-resolution space (see, Fig. 5). Table 2 presents the comparison of the MeshfreeFlowNet with the baseline models. As shown in Table 2, the Baseline (I) is a purely interpolation-based approach and fails to reconstruct the high-resolution data and resolve the fine-scale details, leading to large errors in flow-based evaluation metrics that characterize the flow dynamics. The extremely large normalized mean absolute error (NMAE) of the calculated *rms* velocity for the fine-scale solution found by Baseline (I) indicates that a merely interpolative scheme cannot accurately reconstruct the fine-scale *local* dynamics (*e.g.*, the flow velocities). Moreover, the NMAE of the total kinetic energy (E_{tot}), as a *global* characterizing parameter of the turbulent dynamics, is also very large for the high-resolution solution by Baseline (I). On the other hand, the deep learning based Baseline (II) which utilizes the 3D U-Net model directly maps the low-resolution data to the high-resolution space, achieves better performance compared to Baseline (I). However, as presented in Table 2, our MeshfreeFlowNet model performs significantly better than Baselines (I) and (II). The specification of γ in Table 2 refers to the weighting coefficient of the Equation loss component in the loss function in Eqn. 10. In Table 2, $\gamma = 0$ indicates that only the prediction loss has been considered for training the MeshfreeFlowNet model, whereas $\gamma = \gamma^*$ refers to the optimum value for the weighting coefficient of the Equation loss from the ablation study (see, Table 1, Sec. 5.1).

5.3 Generalizability of MeshfreeFlowNet

We further evaluate the robustness of MeshfreeFlowNet for resolution enhancement of low-resolution datasets that have physical initial and boundary conditions different from the datasets the MeshfreeFlowNet model has been trained on. We refer to such initial and boundary conditions as *unseen* initial/boundary conditions. In order to investigate the generalizability of MeshfreeFlowNet on unseen initial and boundary conditions, we study the effect of each condition separately in the following setups.

5.3.1 Unseen Physical Initial Conditions

We investigate the robustness of a trained MeshfreeFlowNet for enhancing the resolution of a low-resolution unseen data with physical initial conditions different than the training datasets that the MeshfreeFlowNet has been trained on. Table 3 shows the performance of the MeshfreeFlowNet on a dataset with unseen initial conditions. The first row of Table 3 shows the values of the evaluation metrics for unseen test data when MeshfreeFlowNet is trained only on one dataset whereas the second row shows the values of the evaluation metrics when MeshfreeFlowNet is trained on 10 datasets each with a different initial condition. As can be observed from the results, the performance of MeshfreeFlowNet on unseen cases can be improved by training on a more diverse set of initial conditions.

5.3.2 Unseen Physical Boundary Conditions

We further investigate the robustness of a trained MeshfreeFlowNet for super-resolving a low-resolution unseen dataset with physical boundary conditions different from the training datasets that the MeshfreeFlowNet has been trained on. As the use of more datasets in Sec. 5.3.1 was shown to be effective in improving the performance of MeshfreeFlowNet for super-resolution, here we use a dataset that comprises of 10 different sets of boundary conditions (*i.e.* different Rayleigh numbers of $Ra \in [2, 90] \times 10^5$, corresponding to Reynolds numbers of up to 10,000). Table 4 shows the performance of such a trained MeshfreeFlowNet for 5 different test datasets each with a different Rayleigh number boundary condition. In Table 4, MeshfreeFlowNet's performance is evaluated for a Rayleigh number within the range of boundary conditions of the training sets (*i.e.*, $Ra = 5 \times 10^6$), for Rayleigh numbers slightly below and above the range of boundary conditions of the training sets (*i.e.*, $Ra = 1 \times 10^5$ and $Ra = 1 \times 10^7$ respectively), and for Rayleigh numbers far below and above the range of boundary conditions of the training sets (*i.e.*, $Ra = 1 \times 10^4$ and $Ra = 1 \times 10^8$ respectively). As presented in Table 4, MeshfreeFlowNet achieves a good performance not only on the boundary conditions within the range of Rayleigh number boundary conditions it has been trained on, but also on the unseen boundary conditions far out of the range of Rayleigh number boundary conditions it has been trained on. This illustrates the fact that a trained MeshfreeFlowNet model can generalize well to unseen boundary conditions.

5.4 Scalability of MeshfreeFlowNet

Last but not least, we study the scalability of the MeshfreeFlowNet model to study its applicability to larger problems that require orders-of-magnitude more compute. The scaling results are presented in Fig. 7. Figs. 7a demonstrates that an almost-ideal scaling performance can be achieved for the MeshfreeFlowNet model on up to 128

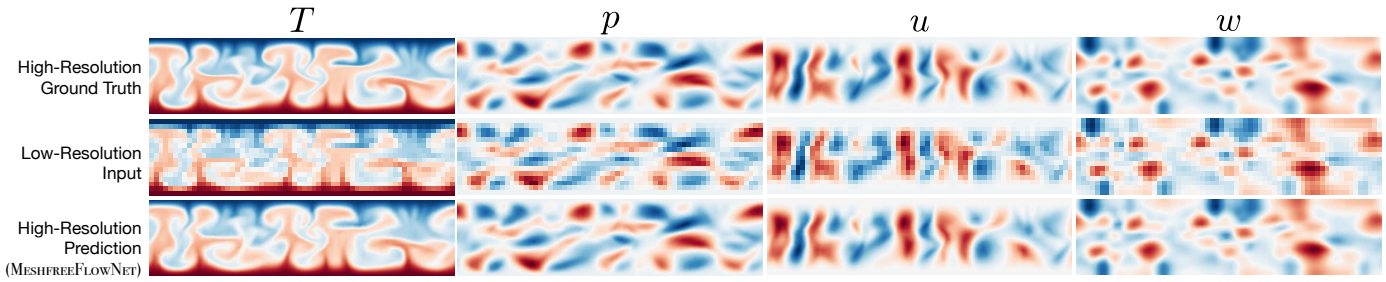
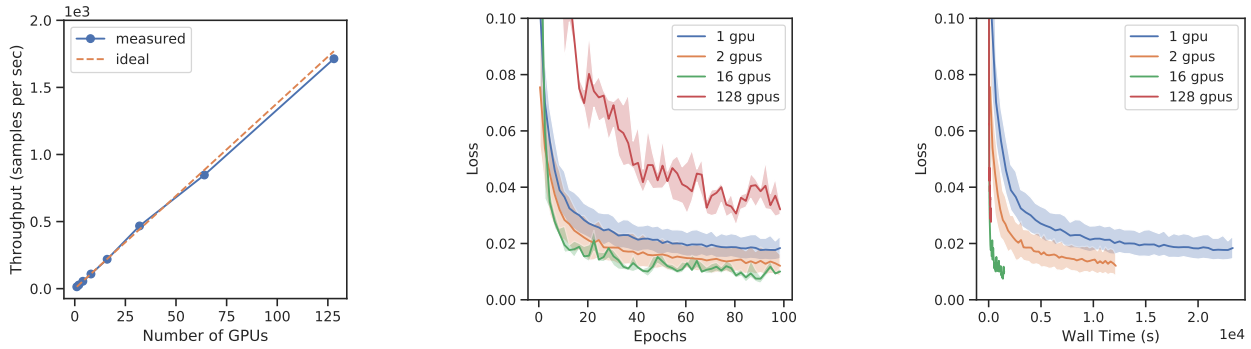


Figure 6: Sample tuples of low-resolution input data for MeshfreeFlowNet, the high-resolution super-resolved data by MeshfreeFlowNet, and the ground truth high-resolution data for the 4 physical parameters of the Rayleigh–Bénard problem, *i.e.*, T, p, u, w as the temperature, pressure, and the x and z components of the velocity. The contours correspond to the MeshfreeFlowNet model evaluation presented in Table 4 for $Ra = 1 \times 10^8$.



(a) Throughput vs. Num. of GPUs (b) Loss vs. Num. of Epochs (c) Loss vs. Wall Time

Figure 7: Scaling performance on up to 128 GPUs on NERSC Cori GPU cluster. Scaling efficiency is close to ideal performance, at 96% with all 128 GPUs. Training convergence is greatly accelerated by scaling across more GPUs. The shaded band in Fig. 7b-7c denotes the $(10\%, 90\%)$ noise interval. *Throughput* refers to the number of samples per second.

GPU workers, achieving approximately 96.80% scaling efficiency. In Fig. 7b, we show the convergence for the model training loss with respect to the number of epochs. In Fig. 7c, we show the loss convergence with respect to total wall time, where an increasing number of GPU workers leads to a drastic decrease in total training time. As the models achieve similar levels of losses after 100 epochs, yet the training throughput scales almost linearly with the number of GPUs, we see a close to an ideal level of scaling for convergence speed on up to 16 GPUs. A small anomaly regarding the loss curve for 128 GPUs where the loss does not decrease to an ideal level after 100 epochs shows that a very large batch size could lead to diminishing scalability with respect to model convergence. This has been observed in numerous machine learning scaling studies and requires further investigations within the community.

6 Conclusion and Future Work

In this work, for the first time, we presented the MeshfreeFlowNet, a physics-constrained super-resolution framework, that can produce continuous super-resolution outputs, would allow imposing arbitrary combinations of PDE constraints and could be evaluated on arbitrary-sized spatio-temporal domains due to its fully-convolutional nature. We further demonstrated that MeshfreeFlowNet can recover a wide range of important physical flow quantities (*e.g.*, including Turbulent Kinetic Energy, Kolmogorov Time and Length Scales, etc.) by accurately super-resolving turbulent flows significantly better than traditional (trilinear interpolation) and deep learning based (3D U-Net) baselines. We further illustrated the scalability of MeshfreeFlowNet to a large cluster of GPU nodes with a high speed interconnect, demonstrating its applicability to problems that require orders of magnitude more computational resources.

Future work includes exploring the applicability of MeshfreeFlowNet to other physical applications beyond 2D Rayleigh Bernard convection. One interesting direction to pursue is to explore the use of 4D spatio-temporal convolution operators such as those proposed by Choy et al. [65] to further extend this framework to 4D space-time simulations. That will open up a wide range of applications in Turbulence modeling, where statistical priors between the low-resolution simulation and subgrid-scale physics can be learned from $3 + 1$D Direct Numerical Simulations (DNS). The scalability of the model on HPC clusters will be critical in learning from such large scale datasets, where single node training will be prohibitively slow. The fully convolutional nature of the MeshfreeFlowNet framework, along with

the demonstrated scalability makes it well poised for such challenges. Moreover, due to the generalizability of our PDE constrained framework, it would be interesting to apply this framework on applications beyond turbulent flows.

Acknowledgements

This research used resources of the National Energy Research Scientific Computing Center (NERSC), a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. The research was performed at the Lawrence Berkeley National Laboratory for the U.S. Department of Energy under Contract No. DE340AC02-05CH11231. K. Kashinath is supported by the Intel Big Data Center at NERSC. K. Azizzadenesheli gratefully acknowledges the financial support of Raytheon and Amazon Web Services. A. Anandkumar is supported in part by Bren endowed chair, DARPA PAIHR00111890035 and LwLL grants, Raytheon, Microsoft, Google, and Adobe faculty fellowships. We also acknowledge the Industrial Consortium on Reservoir Simulation Research at Stanford University (SUPRI-B).

References

[1] Guoqing Hu and Dongqing Li. Multiscale phenomena in microfluidics and nanofluidics. *Chemical Engineering Science*, 62(13):3443–3454, 2007. ISSN 00092509. doi: 10.1016/j.ces.2006.11.058.

[2] James Hurrell, Gerald A. Meehl, David Bader, Thomas L. Delworth, Ben Kirtman, and Bruce Wielicki. A Unified Modeling Approach to Climate System Prediction. *Bulletin of the American Meteorological Society*, 90(12):1819–1832, dec 2009. ISSN 0003-0007. doi: 10.1175/2009BAMS2752.1.

[3] Biman Bagchi and Charusita Chakravarty. Interplay between multiple length and time scales in complex chemical systems. *Journal of Chemical Sciences*, 122(4):459–470, jul 2010. ISSN 0974-3626. doi: 10.1007/s12039-010-0081-0.

[4] Chaoqun Liu and Zhining Liu. Multiple scale simulation for transitional and turbulent flow. In *33rd Aerospace Sciences Meeting and Exhibit*, Reston, Virigina, jan 1995. American Institute of Aeronautics and Astronautics. doi: 10.2514/6.1995-777.

[5] P. Szymczak and A. J. C. Ladd. Interacting length scales in the reactive-infiltration instability. *Geophysical Research Letters*, 40(12):3036–3041, jun 2013. ISSN 00948276. doi: 10.1002/grl.50564.

[6] Zhipeng Qin, Soheil Esmaeilzadeh, Amir Riaz, and Hamdi A. Tchelepi. Two-Phase Multiscale Numerical Framework for Modeling Thin Films on Curved Solid Surfaces in Porous Media. *Journal of Computational Physics*, page 109464, Apr 2020. ISSN 00219991. doi: 10.1016/j.jcp.2020.109464.

[7] Shaoping Quan. Simulations of multiphase flows with multiple length scales using moving mesh interface tracking with adaptive meshing. *Journal of Computational Physics*, 230 (13):5430–5448, 2011. ISSN 10902716. doi: 10.1016/j.jcp.2011.03.050.

[8] A. Riaz, M. Hesse, H. A. Tchelepi, and F. M. Orr. Onset of convection in a gravitationally unstable diffusive boundary layer in porous media. *Journal of Fluid Mechanics*, 548(-1):87, feb 2006. ISSN 0022-1120. doi: 10.1017/S0022112005007494.

[9] Soheil Esmaeilzadeh, Amir Salehi, Gill Hetz, Feyisayo Olalotiti-lawal, Hamed Darabi, and David Castineira. Multiscale modeling of compartmentalized reservoirs using a hybrid clustering-based non-local approach. *Journal of Petroleum Science and Engineering*, 184(September 2019): 106485, jan 2020. ISSN 09204105. doi: 10.1016/j.petrol.2019.106485.

[10] Soheil Esmaeilzadeh, Amir Salehi, Gill Hetz, Feyisayo Olalotiti-lawal, Hamed Darabi, and David Castineira. A General Spatio-Temporal Clustering-Based Non-Local Formulation for Multiscale Modeling of Compartmentalized Reservoirs. In *SPE Western Regional Meeting*, volume 2019. Society of Petroleum Engineers, apr 2019. doi: 10.2118/195329-MS.

[11] P. Jenny, S.H. Lee, and H.A. Tchelepi. Adaptive fully implicit multi-scale finite-volume method for multi-phase flow and transport in heterogeneous porous media. *Journal of Computational Physics*, 217(2):627–641, sep 2006. ISSN 00219991. doi: 10.1016/j.jcp.2006.01.028.

[12] P. Jenny, S. H. Lee, and H. A. Tchelepi. Adaptive Multiscale Finite-Volume Method for Multiphase Flow and Transport in Porous Media. *Multiscale Modeling & Simulation*, 3(1): 50–64, jan 2005. ISSN 1540-3459. doi: 10.1137/030600795.

[13] P. Jenny, S.H Lee, and H.A Tchelepi. Multi-scale finite-volume method for elliptic problems in subsurface flow simulation. *Journal of Computational Physics*, 187(1):47–67, may 2003. ISSN 00219991. doi: 10.1016/S0021-9991(03)00075-5.

[14] F. Bramkamp, Ph Lamby, and S. Müller. An adaptive multiscale finite volume solver for unsteady and steady state flow computations. *Journal of Computational Physics*, 197 (2):460–490, 2004. ISSN 00219991. doi: 10.1016/j.jcp.2003.12.005.

[15] Michael F. Modest. Multiscale Modeling of Turbulence, Radiation, and Combustion Interactions in Turbulent Flames. *International Journal for Multiscale Computational Engineering*, 3(1):85–106, 2005. ISSN 1543-1649. doi: 10.1615/IntJMultCompEng.v3.i1.70.

[16] Bo-Wen Shen, Bron Nelson, Samson Cheung, and Wei-Kuo Tao. Improving NASA's Multiscale Modeling Framework for Tropical Cyclone Climate Study. *Computing in Science & Engineering*, 15(5):56–67, sep 2013. ISSN 1521-9615. doi: 10.1109/MCSE.2012.90.

[17] K. H. Luo, J. Xia, and E. Monaco. *Multiscale modeling of multiphase flow with complex interactions*, volume 1. 2009. ISBN 1756973709. doi: 10.1142/S1756973709000074.

[18] Hsiaotao T. Bi and Jinghai Li. Multiscale analysis and modeling of multiphase chemical reactors. *Advanced Powder Technology*, 15(6):607–627, 2004. ISSN 09218831. doi: 10.1163/1568552042456223.

[19] Matthias Bauer and Gerhart Eigenberger. Multiscale modeling of hydrodynamics, mass transfer and reaction in bubble column reactors. *Chemical Engineering Science*, 56(3):1067–1074, 2001. ISSN 00092509. doi: 10.1016/S0009-2509(00)00323-7.

[20] Mehmet T. Odman and Armistead G. Russell. Multiscale modeling of pollutant transport and chemistry. *Journal of Geophysical Research*, 96(D4):7363, 1991. ISSN 0148-0227. doi: 10.1029/91JD00387.

[21] Weiqing Ren and E. Weinan. Heterogeneous multiscale method for the modeling of complex fluids and microfluidics. *Journal of Computational Physics*, 204(1):1–26, 2005. ISSN 00219991. doi: 10.1016/j.jcp.2004.10.001.

[22] Paris Perdikaris, Leopold Grinberg, and George Em Karniadakis. Multiscale modeling and simulation of brain blood flow. *Physics of Fluids*, 28(2), 2016. ISSN 10897666. doi: 10.1063/1.4941315.

[23] Xiaoxiao Guo, Wei Li, and Francesco Iorio. Convolutional neural networks for steady flow approximation. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[24] Saakaar Bhatnagar, Yaser Afshar, Shaowu Pan, Karthik Duraisamy, and Shailendra Kaushik. Prediction of aerodynamic flow fields using convolutional neural networks. *Computational Mechanics*, 64(2):525–545, 2019.

[25] Yinhao Zhu and Nicholas Zabaras. Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification. *Journal of Computational Physics*, 2018. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2018.04.018.

[26] Weinan E and Bing Yu. The deep ritz method: A deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 3 2018. ISSN 2194-6701. doi: 10.1007/s40304-018-0127-z.

[27] Leah Bar and Nir Sochen. Unsupervised deep learning algorithm for pde-based forward and inverse problems. *arXiv preprint arXiv:1904.05417*, 2019.

[28] Zichao Long, Yiping Lu, Xianzhong Ma, and Bin Dong. Pde-net: Learning pdes from data. *arXiv preprint arXiv:1710.09668*, 2017.

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

[30] Jonathan D Smith, Kamyar Azizzadenesheli, and Zachary E Ross. Eikonet: Solving the eikonal equation with deep neural networks. *arXiv preprint arXiv:2004.00361*, 2020.

[31] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020.

[32] Chiyu Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, and Thomas Funkhouser. Local implicit grid representations for 3d scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[33] Rui Wang, Karthik Kashinath, Mustafa Mustafa, Adrian Albert, and Rose Yu. Towards physics-informed deep learning for turbulent flow prediction. *arXiv preprint arXiv:1911.08655*, 2019.

[34] Chiyu Jiang, Karthik Kashinath, Prahbat, and Philip Marcus. Enforcing physical constraints in cnns through differentiable pde layer. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020.

[35] Michal Irani and Shmuel Peleg. Improving resolution by image registration. *CVGIP: Graphical Models and Image Processing*, 53(3):231–239, may 1991. ISSN 10499652. doi: 10.1016/1049-9652(91)90045-L.

[36] R. Keys. Cubic convolution interpolation for digital image processing. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 29(6):1153–1160, dec 1981. ISSN 0096-3518. doi: 10.1109/TASSP.1981.1163711.

[37] Gilad Freedman and Raanan Fattal. Image and video upscaling from local self-examples. *ACM Transactions on Graphics*, 30(2):1–11, apr 2011. ISSN 0730-0301. doi: 10.1145/1944846.1944852.

[38] Jian Sun, Jian Sun, Zongben Xu, and Heung Yeung Shum. Image super-resolution using gradient profile prior. *26th IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2008. doi: 10.1109/CVPR.2008.4587659.

[39] Kwang In Kim and Younghee Kwon. Single-Image Super-Resolution Using Sparse Regression and Natural Image Prior. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(6):1127–1133, jun 2010. ISSN 0162-8828. doi: 10.1109/TPAMI.2010.25.

[40] Zhiwei Xiong, Xiaoyan Sun, and Feng Wu. Robust Web Image/Video Super-Resolution. *IEEE Transactions on Image Processing*, 19(8):2017–2028, aug 2010. ISSN 1057-7149. doi: 10.1109/TIP.2010.2045707.

[41] W.T. Freeman, T.R. Jones, and E.C. Pasztor. Example-based super-resolution. *IEEE Computer Graphics and Applications*, 22(2):56–65, 2002. ISSN 02721716. doi: 10.1109/38.988747.

[42] Hong Chang, Dit-Yan Yeung, and Yimin Xiong. Super-resolution through neighbor embedding. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages 275–282. IEEE, 2004. ISBN 0-7695-2158-4. doi: 10.1109/CVPR.2004.1315043.

[43] Jianchao Yang, John Wright, Thomas S. Huang, and Yi Ma. Image Super-Resolution Via Sparse Representation. *IEEE Transactions on Image Processing*, 19(11):2861–2873, nov 2010. ISSN 1057-7149. doi: 10.1109/TIP.2010.2050625.

[44] Jianchao Yang, John Wright, Thomas Huang, and Yi Ma. Image super-resolution as sparse representation of raw image patches. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, jun 2008. ISBN 978-1-4244-2242-5. doi: 10.1109/CVPR.2008.4587647.

[45] Wei-Sheng Lai, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. Deep Laplacian Pyramid Networks for Fast and Accurate Super-Resolution. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2017-Janua, pages 5835–5843. IEEE, jul 2017. ISBN 978-1-5386-0457-1. doi: 10.1109/CVPR.2017.618.

[46] Namhyuk Ahn, Byungkon Kang, and Kyung-Ah Sohn. Fast, Accurate, and Lightweight Super-Resolution with Cascading Residual Network. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 11214 LNCS, pages 256–272. 2018. ISBN 9783030012489. doi: 10.1007/978-3-030-01249-6_16.

[47] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual Losses for Real-Time Style Transfer and Super-Resolution. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9906 LNCS, pages 694–711. 2016. ISBN 9783319464749. doi: 10.1007/978-3-319-46475-6_43.

[48] Adrian Bulat and Georgios Tzimiropoulos. Super-FAN: Integrated Facial Landmark Localization and Super-Resolution of Real-World Low Resolution Faces in Arbitrary Poses with GANs. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, number c, pages 109–117. IEEE, jun 2018. ISBN 978-1-5386-6420-9. doi: 10.1109/CVPR.2018.00019.

[49] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced Deep Residual Networks for Single Image Super-Resolution. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, volume 2017-July, pages 1132–1140. IEEE, jul 2017. ISBN 978-1-5386-0733-6. doi: 10.1109/CVPRW.2017.151.

[50] Yifan Wang, Federico Perazzi, Brian McWilliams, Alexander Sorkine-Hornung, Olga Sorkine-Hornung, and Christopher Schroers. A Fully Progressive Approach to Single-Image Super-Resolution. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, volume 2018-June, pages 977–97709. IEEE, jun 2018. ISBN 978-1-5386-6100-0. doi: 10.1109/CVPRW.2018.00131.

[51] Keaton J. Burns, Geoffrey M. Vasil, Jeffrey S. Oishi, Daniel Lecoanet, and Benjamin P. Brown. Dedalus: A Flexible Framework for Numerical Simulations with Spectral Methods. 2019.

[52] Distributed communication package - torch.distributed, . URL https://pytorch.org/docs/stable/distributed.html.

[53] Nvidia collective communications library. URL https://docs.nvidia.com/deeplearning/sdk/nccl-developer-guide/docs/overview.html.

[54] Torch distributed data parallel, . URL https://pytorch.org/docs/stable/nn.html#distributeddataparallel.

[55] Cori gpu node topology. URL https://docs-dev.nersc.gov/cgpu/hardware/#node-topology.

[56] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9351, pages 234–241. 2015. ISBN 9783319245737. doi: 10.1007/978-3-319-24574-4_28.

[57] Lvmin Zhang, Yi Ji, Xin Lin, and Chunping Liu. Style transfer for anime sketches with enhanced residual u-net and auxiliary classifier GAN. *Proceedings - 4th Asian Conference on Pattern Recognition, ACPR 2017*, pages 512–517, 2018. doi: 10.1109/ACPR.2017.61.

[58] Sheng Lian, Lei Li, Guiren Lian, Xiao Xiao, Zhiming Luo, and Shaozi Li. A Global and Local Enhanced Residual U-Net for Accurate Retinal Vessel Segmentation. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 14(8):1–1, 2019. ISSN 1545-5963. doi: 10.1109/tcbb.2019.2917188.

[59] Wei Chen, Boqiang Liu, Suting Peng, Jiawei Sun, and Xu Qiao. S3D-UNet: Separable 3D U-Net for Brain Tumor Segmentation. volume 2, pages 358–368. Springer International Publishing, 2019. ISBN 9783030117252. doi: 10.1007/978-3-030-11726-9_32.

[60] Yu Sheng Chen, Yu Ching Wang, Man Hsin Kao, and Yung Yu Chuang. Deep Photo Enhancer: Unpaired Learning for Image Enhancement from Photographs with GANs. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 6306–6314, 2018. ISSN 10636919. doi: 10.1109/CVPR.2018.00660.

[61] Gang Liu, Xin Chen, and Yanzhong Hu. Anime sketch coloring with swish-gated residual U-net. *Communications in Computer and Information Science*, 986(August):190–204, 2019. ISSN 18650929. doi: 10.1007/978-981-13-6473-0_17.

[62] Ming Fang, Yu Song, Xiaoying Bai, Yushu Ren, and Shuhua Liu. A Method for Coloring Low-resolution Black and White Old Movies through Object Understanding. *Proceedings of the 31st Chinese Control and Decision Conference, CCDC 2019*, pages 5944–5949, 2019. doi: 10.1109/CCDC.2019.8833380.

[63] Patrick Esser and Ekaterina Sutter. A Variational U-Net for Conditional Appearance and Shape Generation Heidelberg Collaboratory for Image Processing. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 8857–8866, 2018.

[64] Shuaibin Zhang, Haoran Su, Tangbo Liu, and Xin Fu. Artistic Image Generation from Sketch by Using Conditional Adversarial Network and Style Feature Transform. pages 1–6.

[65] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3075–3084, 2019.