NEURAL SUPER-RESOLUTION FOR MESH-BASED SIMULATIONS UNDER SCARCE SUPERVISION

Anonymous authors
Paper under double-blind review

ABSTRACT

Mesh-based simulations provide high-fidelity solutions to partial differential equations (PDEs), but achieving such accuracy typically requires fine meshes, leading to substantial computational overhead. Super-resolution techniques aim to mitigate this cost by reconstructing high-resolution (HR), high-fidelity solutions from low-cost, low-resolution (LR) counterparts. However, training neural networks for super-resolution often demands large amounts of expensive HR supervision data, posing a major practical limitation. To address this challenge, we propose SuperMeshNet, an HR data-efficient super-resolution framework for mesh-based simulations aided by message passing neural networks (MPNNs). As its core, SuperMeshNet introduces **complementary learning** that effectively leverages both a small amount of paired LR–HR data and abundant unpaired LR data via two jointly trained, complementary MPNN-based models. Additionally, our model is enriched by **inductive biases**, which is empirically shown to further improve super-resolution performance. Extensive experiments demonstrate that SuperMeshNet—an MPNN-based model with inductive biases trained on a dataset with 10% paired LR–HR data and 90% unpaired LR data—achieves an even lower root mean square error (RMSE) than the same MPNN without inductive biases trained on 100% of LR–HR pairs, while in turn requiring 90% less HR data. The source code and datasets are available at https://anonymous.4open.science/r/SuperMeshNet/README.md.

1 INTRODUCTION

Mesh-based simulations—such as the finite element method (FEM), finite volume method (FVM), or computational fluid dynamics (CFD)—are widely used to obtain high-fidelity solutions to partial differential equations (PDEs) across a range of scientific and engineering domains. In mesh-based simulations, the mesh size is carefully chosen to balance computational costs against solution fidelity: finer meshes offer higher fidelity but incur significantly greater computational expenses (Obiols-Sales et al., 2024). Super-resolution techniques are developed to alleviate this trade-off by predicting high-resolution (HR) simulation results from low-resolution (LR) counterparts, thereby aiming to deliver high-fidelity solutions at a reduced cost (Barwey et al., 2024; Obiols-Sales et al., 2024). However, training super-resolution models via conventional fully supervised learning demands substantial quantities of computationally expensive HR data, making data collection a significant bottleneck (Obiols-Sales et al., 2024). In this context, improving the HR data efficiency of super-resolution model training is of paramount importance in reality.

As summarized in Table 1, several unsupervised learning approaches tackled this challenge but pose their own limitations. For example, PhysRNet (Arora, 2022) performs super-resolution without any HR data by incorporating PDEs and constraints into its loss function; however, PhysRNet uses a finite-difference scheme for derivative calculations, which limits its applicability to irregular meshes. MAgNet (Boussif et al., 2022) offers an alternative with zero-shot super-resolution through an interpolator trained on LR data; yet, the prediction error of MAgNet is much larger than that of supervised methods (see Appendix H). To the best of our knowledge, *semi-supervised learning* has not been applied to the super-resolution task for mesh-based simulations. This may be partly due to the limited exploration of semi-supervised regression methods, especially those compatible with message passing neural networks (MPNNs), compared to semi-supervised classification. Refer to Appendix B for detailed explanations and limitations of existing semi-supervised regression approaches.

Table 1: Comparison between prior studies and our work. Here, $r = \frac{\text{number of HR data samples}}{\text{number of LR data samples}}$.

Reference	Learning method	Model
Li and McComb (2022), Yonekura et al. (2023), Obiols-Sales et al. (2024)	fully supervised ($r = 1$)	CNN
de Avila Belbute-Peres et al. (2020), Barwey et al. (2024)	fully supervised ($r = 1$)	MPNN
Arora (2022)	unsupervised ($r = 0$)	CNN
Boussif et al. (2022)	unsupervised ($r = 0$)	MPNN
SuperMeshNet (ours)	**semi-supervised** ($0 < r \ll 1$) **(complementary)**	MPNN **(inductive biases)**

Many related studies (Yonekura et al., 2023; Arora, 2022; Li and McComb, 2022; Obiols-Sales et al., 2024) on super-resolution for mesh-based simulations rely heavily on convolutional neural networks (CNNs), which cannot directly handle irregular mesh structures. CNNs require interpolating irregular mesh-based data onto a regular grid, which often necessitates a significantly larger number of nodes to achieve the same fidelity as an irregular mesh, leading to relatively lower computational efficiency. Some studies have adopted MPNNs, such as graph convolutional networks (GCNs) (de Avila Belbute-Peres et al., 2020) or SRGNN (Barwey et al., 2024), which can directly handle irregular mesh data. However, the impact of inductive biases on enhancing mesh-based super-resolution performance remains largely underexplored.

To address these limitations, we propose SuperMeshNet, an HR data-efficient super-resolution framework tailored for mesh-based simulations under *very scarce HR supervision*, which basically differs from the supervised and unsupervised settings. To the best of our knowledge, this is the first general framework that can be applied across diverse MPNN architectures for the super-resolution task. To be specific, SuperMeshNet introduces two key components: **complementary learning** and **inductive biases for MPNNs.** First, the **complementary learning** is a *semi-supervised learning* method that exploits a small amount of paired LR–HR data for supervised learning, while judiciously leveraging a large pool of unpaired LR data in an unsupervised manner. Our complementary learning is built upon two models; an MPNN-based primary model predicts HR solutions from LR counterparts, while an MPNN-based auxiliary model predicts the ***difference*** between two HR solutions corresponding to two LR counterparts. The predictions from each model are utilized to calculate *pseudo-ground truths*, which serve as ground truth for the other, enabling *mutual supervision*. Since conventional semi-supervised methods typically employ two identical models, they often produce highly similar pseudo-ground truths, making them less informative. On the other hand, owing to distinct but interrelated input–output configurations of our complementary learning, the auxiliary model captures intra-resolution relation while the primary model focuses on inter-resolution relation. This division of roles fosters synergies in mutual supervision, enhancing super-resolution performance while reducing training time compared to prior semi-supervised strategies.

Second, to improve the performance of mesh-based super-resolution, we introduce **inductive biases for MPNNs**, guided by our empirical observation. Specifically, we employ two MPNN-architecture-agnostic inductive biases: **node-level centering** and **message-level centering**. The node-level centering centers each node embedding by subtracting the global mean of all node embeddings from each node embedding, while the message-level centering performs a similar centering operation over aggregated messages.

We carry out extensive experiments to validate the effectiveness of our two components in SuperMeshNet. Our results demonstrate that, even with only a small portion (*e.g.,* 10%) of paired LR–HR data, SuperMeshNet surpasses a fully supervised (*e.g.,* 100% paired) benchmark method lacking inductive biases in terms of the root mean square error (RMSE). We also prove that the injected inductive biases consistently reduce the RMSE across six different MPNN architectures, underscoring their general applicability. Finally, our main contributions are summarized as follows:

- **MPNN-agnostic applicability.** SuperMeshNet provides a general super-resolution framework for mesh-based simulations, applicable to various MPNNs, under very scarce HR supervision scenarios.
- **Complementary learning.** To the best of our knowledge, this is the first attempt to incorporate semi-supervised learning compatible with MPNNs into super-resolution for the mesh-based simulations.
- **Inductive biases.** We introduce node-level centering and message-level centering, which can substantially enhance super-resolution performance across different MPNN types.

2 METHODOLOGY[1] AND SUMMARIZED IN AP

2.1 PROBLEM DEFINITION

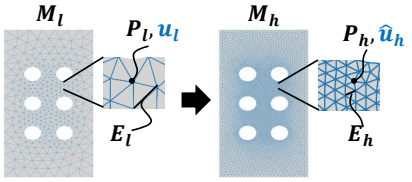

Figure 1: Problem setting. We aim to make prediction $\hat{u}_h$ on HR mesh M_h, containing nodes at positions P_h and edges E_h, from LR data sample u_l defined on LR mesh M_l, comprising nodes at positions P_l and edges E_l.

Briefly, we aim to predict an HR solution $\hat{u}_h$ from an LR solution u_l of the same PDE, while relying on as few HR solutions u_h as possible for training. Formally, let $\Omega \subset \mathbb{R}^D$ be the computational domain on which the PDE is solved. Here, D denotes the spatial dimension. A parameter μ represents all variations of PDE instances, such as material coefficients, domain geometry, or boundary conditions. For example, μ could be an angle of an applied force or an aspect ratio of an elliptical hole (see Figures 14–16 in Appendix G). Each choice of μ defines a different PDE instance. We discretize $\mathbf{\Omega}$ with an LR mesh $M_l = (P_l, E_l)$ and an HR mesh $M_h = (P_h, E_h)$, where $P_l \in \mathbb{R}^{n_l \times D}$ and $P_h \in \mathbb{R}^{n_h \times D}$ are the positions of the nodes on M_l and M_h, respectively, and E_l and E_h are edges on M_l and M_h, respectively. Running a PDE solver on these meshes yields: LR and HR solutions, which we regard as an LR data sample u_l and an HR data sample u_h, defined on the nodes on M_l and M_h, respectively. Our objective is to predict $\hat{u}_h \in \mathbb{R}^{n_h \times d}$ from u_l as closely as possible to u_h, while minimizing the amount of HR data u_h required for training, where d denotes the dimension of the solution field.

2.2 COMPLEMENTARY LEARNING

2.2.1 DATASET SETTING

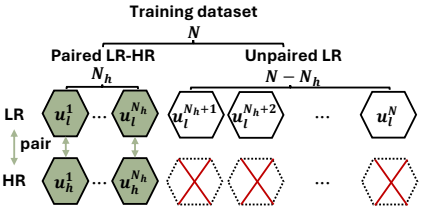

Figure 2: Dataset setting. Complementary learning utilizes a paired LR–HR training dataset, including N_h paired data samples (green hexagons), and an unpaired LR training dataset, containing $N - N_h$ unpaired LR data samples (white hexagons). In total, complementary learning can reduce $N - N_h$ HR data samples compared to the case of fully supervised learning.

As depicted in Figure 2, the complementary learning in SuperMeshNet leverages both a paired LR–HR training dataset $\mathcal{D}_a = \{(u_l^q, u_h^q) \mid q = 1, 2, \cdots, N_h\}$ having N_h LR–HR data pairs (u_l^q, u_h^q) and an unpaired LR training dataset $\mathcal{D}_b = \{u_l^q \mid q = N_{h+1}, N_{h+2}, \cdots, N\}$ having $N - N_h$ LR data samples. The total number of LR data samples is N, among which only N_h have HR counterparts, with $N_h \ll N$ in practice. In other words, $N - N_h$ fewer HR data samples are required compared to fully supervised learning. Here, the superscript q is simply an index to distinguish different samples corresponding to different parameters μ. For instance, if μ is the angle of an applied force, then (u_l^1, u_h^1) corresponds to one angle μ^1, and (u_l^2, u_h^2) corresponds to another μ^2.

2.2.2 THE TWO MODELS: F_θ AND G_ϕ

To fully exploit unpaired LR data, the complementary learning utilizes mutual supervision between two models, namely, F_θ and G_ϕ, trained jointly under different roles. The primary model F_θ, which is used for inference only, predicts an HR solution $\hat{u}_h^q$ from its LR counterpart u_l^q:

$$F_\theta(u_l^q) = \hat{u}_h^q \quad (\text{ground truth} : u_h^q). \tag{1}$$

On the other hand, the auxiliary model G_ϕ, which is used only during training, predicts the ***difference between two HR*** solutions $\hat{u}_h^{rs}$ corresponding to two LR input samples u_l^r, u_l^s to further utilize intra-resolution relations. Here, r and s indicate two LR samples corresponding to different parameter μ's. Since computational geometry may vary across samples with different μ, HR samples u_h^r and u_h^s may be defined on different positions P_h^r and P_h^s. Thus, direct subtraction is not possible. To resolve this, we apply k-nearest neighbor (kNN) interpolation (Qi et al., 2017) to project solutions defined on P_h^s onto P_h^r:

$$G_\phi(u_l^r, u_l^s) = \hat{u}_h^{rs} \quad (\text{ground truth} : u_h^r - kNN(u_h^s; P_h^s \to P_h^r)). \tag{2}$$

[1]All notations are defined upon their first appearance in each subsection

In kNN interpolation, each target point is assigned a distance-weighted average of its k nearest source points. The detailed calculation is provided in Appendix D.

2.2.3 Learning Procedure

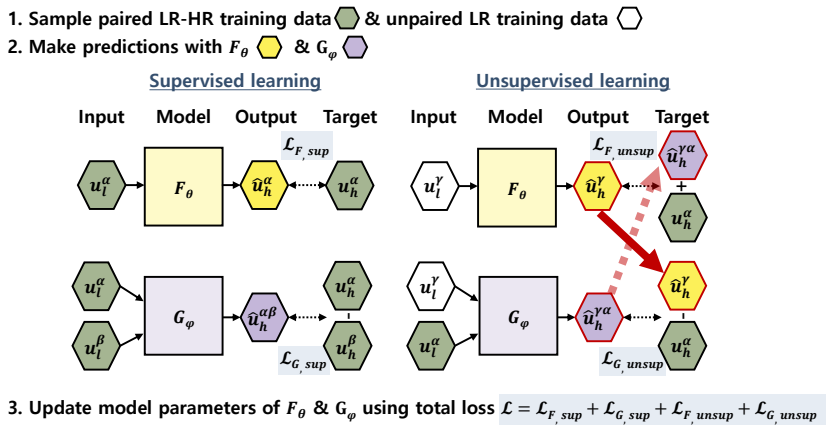

Figure 3: A schematic overview of complementary learning in SuperMeshNet. It first samples paired LR–HR data (u_l^α, u_h^α), (u_l^β, u_h^β) and unpaired LR data u_l^γ. It leverages both supervised and unsupervised learning to jointly train two neural network models, F_θ and G_ϕ. F_θ predicts an HR solution from its LR counterpart, while G_ϕ predicts the difference between two HR solutions from two LR counterparts to enable synergistic mutual supervision. More specifically, for supervised learning, F_θ and G_ϕ are trained with pairs of LR–HR data (green hexagons). In unsupervised learning, the prediction of one model (yellow and purple hexagons predicted by F_θ and G_ϕ, respectively) is used to calculate a pseudo-ground truth that serves as the target for training another model (as depicted by solid and dotted red arrows). The model parameters of F_θ and G_ϕ are updated using the total loss, including both supervised and unsupervised losses based on predictions from both F_θ and G_ϕ.

As depicted in Figure 3, each training step combines supervised and unsupervised learning of the two models F_θ and G_ϕ. In other words, the loss functions for training F_θ and G_ϕ, denoted by $\mathcal{L}_F$ and $\mathcal{L}_G$, respectively, are expressed as:

$$\mathcal{L}_F = \mathcal{L}_{F,sup} + \mathcal{L}_{F,unsup} \tag{3}$$

$$\mathcal{L}_G = \mathcal{L}_{G,sup} + \mathcal{L}_{G,unsup}, \tag{4}$$

where the subscripts sup and $unsup$ represent supervised and unsupervised learning, respectively. To this end, three samples are randomly sampled: 1) two paired LR samples u_l^α and u_l^β from the paired LR–HR dataset $\mathcal{D}_a$ for supervised learning and 2) one additional unpaired LR sample u_l^γ from the unpaired LR dataset $\mathcal{D}_b$ for unsupervised learning. Here, α, β, and γ are the indices referring to distinct parameters μ.

In supervised learning, an HR data sample u_h^α is available. As depicted in Figure 3, F_θ is trained to reduce the MSE between its prediction $\hat{u}_h^\alpha$ and its target, which is the ground truth u_h^α. An analogous procedure is applied to β, thus resulting in:

$$L_{F,sup} = \ell(\hat{u}_h^\alpha, u_h^\alpha) + \ell(\hat{u}_h^\beta, u_h^\beta), \tag{5}$$

where $\ell(\cdot,\cdot)$ denotes the MSE. Similarly, as expressed in Eq. (2), G_ϕ is trained to reduce the MSE between its predictions $\hat{u}_h^{\alpha\beta}$ and its target, which is the ground truth difference between u_h^α and u_h^β alongside the following loss:

$$\mathcal{L}_{G,sup} = \ell(\hat{u}_h^{\alpha\beta}, u_h^\alpha - kNN(u_h^\beta; P_h^\beta \to P_h^\alpha)). \tag{6}$$

In unsupervised learning, the ground truth HR data sample u_h^γ is unavailable. Under this circumstance, we leverage *mutual supervision* between F_θ and G_ϕ. For example, as depicted in Figure 3, if $G_\phi(u_l^\gamma, u_l^\alpha)$ predicts $\hat{u}_h^{\gamma\alpha}$ that approximates the difference between two HR samples $u_h^\gamma - u_h^\alpha$, then adding this to the known u_h^α yields an estimate of u_h^γ. This pseudo-ground truth can serve as a target for $F_\theta(u_l^\gamma)$. Similarly, if $F_\theta(u_l^\gamma)$ produces $\hat{u}_h^\gamma$ close to u_h^γ, then subtracting known u_h^α from $\hat{u}_h^\gamma$ provides an approximation of $u_h^\gamma - u_h^\alpha$, which can serve as a target for $G_\phi(u_l^\gamma, u_l^\alpha)$, accordingly. An analogous procedure is applied to the pair (β, γ). It should be noted that a thorough treatment of kNN interpolation is also required to effectively handle mesh mismatches. For example, in Eq. (7), $u_h^\beta - \hat{u}_h^{\beta\gamma}$ can be used to approximate u_h^γ. However, the values are defined on position P_h^β, whereas the prediction $\hat{u}_h^\gamma$ is defined on position P_h^γ. To reconcile this discrepancy, kNN interpolation is employed to project the values from P_h^γ onto P_h^β. The resultant loss functions are:

$$\mathcal{L}_{F,unsup} = \ell(\hat{u}_h^\gamma, \hat{u}_h^{\gamma\alpha} + kNN(u_h^\alpha; P_h^\alpha \to P_h^\gamma)) + \ell(\hat{u}_h^\gamma, kNN(u_h^\beta - \hat{u}_h^{\beta\gamma}; P_h^\beta \to P_h^\gamma)) \quad (7)$$

$$\mathcal{L}_{G,unsup} = \ell(\hat{u}_h^{\gamma\alpha}, \hat{u}_h^\gamma - kNN(u_h^\alpha; P_h^\alpha \to P_h^\gamma)) + \ell(\hat{u}_h^{\beta\gamma}, u_h^\beta - kNN(\hat{u}_h^\gamma; P_h^\gamma \to P_h^\beta)). \quad (8)$$

A pseudo-code for the complementary learning mechanism can be found in Appendix C.

2.3 Model Architecture

The architecture of the primary model F_θ, illustrated in Figure 4, is built upon SRGNN (Barwey et al., 2024). The role of F_θ is to transform LR data into HR data, which is conducted by the lowermost upsampler in Figure 4. To surpass the performance of kNN interpolation by the lowermost upsampler, we introduce additional upsampling in latent space. Specifically, an encoder maps the physical quantities into high-dimensional latent space. The LR processor applies message passing to refine LR representations, which are then upsampled to HR latent embeddings. Subsequently, the HR processor applies additional message passing to further enhance the HR representations. Finally, a decoder maps the latent embeddings back to the physical space. The final HR output is obtained by adding the two upsampled HR fields: one from the kNN-based upsampler and the other from the latent-space upsampling pathway.

The auxiliary model G_ϕ, visualized in Figure 5, is responsible for predicting the difference between two HR samples corresponding two LR inputs. It extends F_θ to accommodate two input samples, maintaining a comparable structure. A notable architectural feature is the use of a shared feature extractor between F_θ and G_ϕ, which helps reduce computational costs during training. First,latent embeddings are extracted from the two LR inputs using the shared feature extractor. Then, one embedding is subtracted from another and the result is decoded to predict the HR difference. The final HR output is obtained by adding the two upsampled HR fields: one from the kNN-based upsampler and the other from the latent-space upsampling pathway. The interpolators in the Figure 13 serve only to address mesh mismatches when the two LR samples are defined on different meshes.

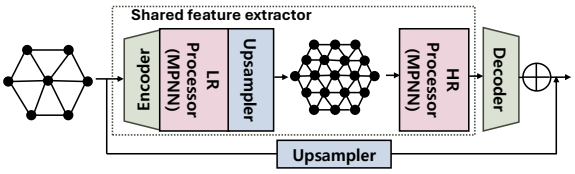

Figure 4: Model architecture of F_θ

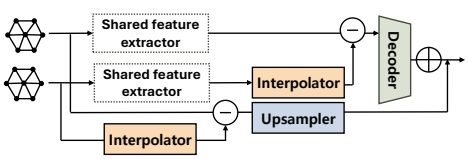

Figure 5: Model architecture of G_ϕ

2.4 Inductive Biases for MPNNs

The key difference between the two models, F_θ and G_ϕ, and SRGNN (Barwey et al., 2024) is that the MPNNs in the LR and HR processors are enriched with inductive biases: **node-level centering** and **message-level centering**. We illustrate these inductive biases using an MPNN based on MeshGraphNet (MGN) (Pfaff et al., 2021), which serves as the default MPNN architecture throughout our experiments. Implementation details for various types of MPNNs (incluing MGN) are provided in Appendix F.

In an MPNN layer, a target node i receives messages msg_{ij} from its neighboring nodes $j \in \mathcal{N}(i)$. In MGN, the message is defined as the edge embedding e_{ij}, where e_{ij} is computed from the node embeddings x_i and x_j of nodes i and j using a learnable multilayer perceptron (MLP) MLP_e:

$$\begin{aligned} e_{ij} &\leftarrow MLP_e(x_i, x_j, e_{ij}), \\ msg_{ij} &= e_{ij}, \end{aligned} \tag{9}$$

where the exact definition of msg_{ij} depends on the MPNN types.

Each node i then aggregates messages msg_{ij} from its neighbors j, and the aggregated message agg_i is used to update its node embedding by employing a learnable MLP MLP_x:

$$\begin{aligned} agg_i &= \sum_{j \in \mathcal{N}(i)} msg_{ij}, \\ x_i &\leftarrow MLP_x(x_i, agg_i). \end{aligned} \tag{10}$$

We empirically find that injecting the following two inductive biases into the message-passing mechanism substantially improves super-resolution performance. The first inductive bias, **node-level centering**, subtracts the mean of all node embeddings x_i from each individual x_i after the node embedding update in Eq. (10):

$$x_i \leftarrow x_i - \frac{1}{n} \sum_{i=1}^{n} x_i, \tag{11}$$

where n denotes the number of nodes in the LR mesh M_l or the HR mesh M_h. For MPNNs that explicitly compute aggregated messages agg_i, such as MGN, **message-level centering** subtracts the mean of agg_i from each individual aggregated message between the message aggregation and the node embedding update in Eq. (10):

$$agg_i \leftarrow agg_i - \frac{1}{n} \sum_{i=1}^{n} agg_i. \tag{12}$$

The centerings tend to smooth the loss landscape, similar to the effect reported for batch normalization (Santurkar et al., 2018), which may facilitate optimization. However, because centering removes global mean information, it is beneficial only for tasks that do not rely heavily on such information (*e.g.*, super-resolution). Experimental validations of these arguments are provided in Appendix T.

3 Experimental Results and Analyses

3.1 Datasets[2]

FEM Datasets. FEM datasets used in our experiments are generated by solving PDEs utilizing FEniCSx, an open-source computing platform for FEM. Table 3 summarizes three datasets, detailing their governing PDEs, the quantities derived from solving these equations, the parameters that vary across the samples, and the number of nodes in LR and HR meshes. Dirichlet boundary conditions are utilized for all datasets.

Table 2: Summary of FEM datasets.

Dataset	Equation	Solution	Parameter	LR nodes	HR nodes
1	Linear elasticity	von Mises stress	Force angle	333	4,053
2	Linear elasticity	von Mises stress	Hole shape	329–387	3,959–4,157
3	Poisson equation	Electric field	Hole shape	324–388	3,959–4,154

CFD Datasets. To validate the applicability of SuperMeshNet to time-dependent PDEs and complex real-world geometry, we adopt three additional datasets. The time-dependent PDE dataset 1 is obtained by solving the incompressible Navier–Stokes equations for flow around a cylinder. The time-dependent PDE dataset 2 is generated following (Kochkov et al., 2021a) to construct a dataset in which the HR vorticity differ substantially from the LR vorticity. Meanwhile, the real-world geometry dataset is constructed by solving the Laplace equation for potential flow around a motorbike with a rider. For the time-dependent PDE dataset 1, LR data samples are generated by downsampling HR data onto LR meshes. For all other datasets, LR data samples are obtained by independently solving the governing PDEs on LR meshes. Detailed descriptions of each dataset are provided in Appendix G. We refer to Section 3.9 to see results on CFD datasets and their relevant discussions.

[2]A detailed description of the datasets and the complete set of experiments is provided in Appendices G–Q.

Table 3: Summary of CFD datasets.

Dataset	Equation	Solution	Parameter	LR nodes	HR nodes
Time-dependent PDE 1	Incompressible Navier-Stokes	Speed	Time	576	7,440
Time-dependent PDE 2	Incompressible Navier-Stokes	Vorticity	Time	1,024	1,048,576
Real-world geometry	Laplace	Pressure	Angle of attack	10,897	46,004

3.2 Experimental Setup

We evaluate our methodology using six representative MPNNs, including GCN (Kipf and Welling, 2017), GraphSAGE (SAGE) (Hamilton et al., 2017), GAT (Veličković et al., 2018), Graph Transformer (GTR) (Shi et al., 2021), GIN (Xu et al., 2019), and MGN (Pfaff et al., 2021). Each MPNN consists of three layers for processing in LR and additional three layers for processing in HR, and hidden dimension of each layer is 30. Throughout the experiments, we adopt Adam optimizer with learning rate of 1×10^{-3} and PyTorch's automatic mixed precision training to improve computational efficiency. All experiments are carried out on a machine with Intel (R) Core (TM) i9-10920X CPUs@3.50 GHz and an NVIDIA RTX A6000 GPU. The RMSE is used as a metric where lower values indicate better performance.

3.3 Comparison with Full Supervision

Table 4: The RMSE of SuperMeshNet (with inductive biases) and SuperMeshNet-O (without inductive biases) trained with $N_h = 20$ HR data samples and $N = 200$ LR data samples across six MPNNs and three datasets, in comparison with two fully supervised MPNNs including 1) $N_h = N = 20$ and 2) $N_h = N = 200$. The best performer is highlighted as **bold**.

	Method	N_h, N	MPNN					
			GCN	SAGE	GAT	GTR	GIN	MGN
Dataset 1	Fully supervised	20, 20	0.0874	0.0876	0.0826	0.0758	0.0819	0.0655
	Fully supervised	200, 200	0.0575	0.0544	0.0512	0.0450	0.0381	0.0228
	SuperMeshNet–O	20, 200	0.0613	0.0589	0.0544	0.0451	0.0404	0.0269
	SuperMeshNet	20, 200	**0.0431**	**0.0450**	**0.0457**	**0.0385**	**0.0277**	**0.0226**
Dataset 2	Fully supervised	20, 20	0.0972	0.1025	0.0983	0.0983	0.0775	0.0730
	Fully supervised	200, 200	0.0624	0.0633	0.0637	**0.0572**	**0.0534**	**0.0461**
	SuperMeshNet–O	20, 200	0.0636	0.0664	0.0680	0.0631	0.0569	0.0514
	SuperMeshNet	20, 200	**0.0574**	**0.0624**	**0.0634**	0.0600	0.0537	0.0507
Dataset 3	Fully supervised	20, 20	0.0587	0.0611	0.0616	0.0513	0.0569	0.0523
	Fully supervised	200, 200	0.0370	0.0340	0.0374	0.0329	0.0317	**0.0243**
	SuperMeshNet–O	20, 200	0.0380	0.0366	0.0375	0.0363	0.0316	0.0281
	SuperMeshNet	20, 200	**0.0297**	**0.0297**	**0.0310**	**0.0294**	**0.0258**	0.0245

Table 4 compares the RMSE of each MPNN integrated with our framework SuperMeshNet, and its variant without inductive biases, SuperMeshNet-O, against two fully supervised baselines—the same type of MPNNs but trained with full supervision without inductive biases. SuperMeshNet-O trained with 20 HR data samples (*i.e.*, $N_h = 20$) and 200 LR data samples (*i.e.*, $N = 200$) achieves a significantly lower RMSE compared to the MPNNs trained exclusively on 20 paired LR–HR samples (*i.e.*, $N_h = N = 20$). The improvement is attributed to complementary learning, which is inherently designed to effectively leverage the 180 unpaired LR samples that fully supervised learning cannot utilize. Remarkably, despite being trained only with 20 HR data samples, SuperMeshNet-O achieves RMSE values that are on par with the second fully supervised baseline trained with the entire 200 HR data samples (*i.e.*, $N_h = N = 200$). SuperMeshNet, enriched by inductive biases, surpasses the second fully supervised baseline ($N_h = N = 200$) in most cases, highlighting the efficacy of the proposed inductive biases tailored for super-resolution in improving performance. This implies the potential to reduce up to 90% of the effort required to generate HR data. It is worth noting that the main advantage of our approach lies in reducing HR data requirements by 90% while maintaining the accuracy of full supervision, rather than in improving absolute RMSE

itself. Furthermore, our findings consistently demonstrate the improvement of SuperMeshNet in terms of HR data efficiency across all six MPNNs. This underscores its versatility and effectiveness in enhancing super-resolution performance, regardless of types of underlying MPNN architectures. Additional experimental analyses, including the effect of strategies for selecting 20 HR samples and an ablation study disentangling the contributions of complementary learning and inductive biases, are presented in Appendices U and W, respectively.

3.4 Comparison with a Super-Resolution Competitor

Although the primary objective of SuperMeshNet is to improve super-resolution performance across a wide range of MPNNs rather than to outperform a specific state-of-the-art method, we compare a special case of SuperMeshNet using MGN with the most recent and relevant benchmarks, SRGNN (Barwey et al., 2024) and MAgNet (Boussif et al., 2022), to further validate its effectiveness. The results in Table 5 signify that SuperMeshNet, even when trained with only 20 HR data samples, outperforms SRGNN (Barwey et al., 2024) trained with 200 HR data samples, underscoring its superior data efficiency. Furthermore, the results demonstrate that SuperMeshNet significantly outperforms the unsupervised baseline, MAgNet (Boussif et al., 2022), even under minimal HR supervision (N_h=5).

Table 5: The RMSE of MGN-based SuperMeshNet trained with varying numbers of HR data samples N_h and a fixed N=200 LR data samples for Dataset 1 in comparison with SRGNN with full supervision (N=N_h=200), and MAgNet with no supervision (N=200, N_h=0).

Methods	N_h	RMSE
SuperMeshNet	5	0.0447
	10	0.0280
	20	0.0226
	40	0.0191
SRGNN	200	0.0247
MAgNet	0	0.0979

3.5 Comparison with Benchmark Semi-Supervised Regression Methods

Table 6: Comparison with benchmark semi-supervised regression methods in terms of the RMSE and training time (in second). Here, MGN is employed as an MPNN for each method. Training is conducted when $N_h = 20$ and $N = 200$ for Dataset 1. The best performer is highlighted as **bold**.

Methods	RMSE	Training time (s)
mean-teacher (Tarvainen and Valpola, 2017)	0.0325	693.84
TNNR (Wetzel et al., 2022)	0.0624	477.48
UCVME (Dai et al., 2023)	0.0293	1122.62
SuperMeshNet-O	0.0269	503.2
SuperMeshNet	**0.0226**	**421**

Table 6 compares complementary learning in SuperMeshNet against benchmark semi-supervised regression methods on Dataset 1, using MGN as an MPNN architecture for each method. As presented in Table 6, SuperMeshNet achieves the lowest RMSE while also exhibiting the shortest training time among all benchmark semi-supervised regression methods. The performance improvements achieved by SuperMeshNet likely stem from its inherent characteristics of using two complementary models. Mean-Teacher (Tarvainen and Valpola, 2017) and UCVME (Dai et al., 2023) employ two models to predict the same target, *i.e.*, an HR data sample. Similarly, TNNR (Wetzel et al., 2022) uses one twin neural network to predict the difference between two HR data. On the other hand, SuperMeshNet employs the *complementary learning* mechanism that leverages two distinct yet cooperative models: primary model F_θ, which learns to predict an HR data sample, and auxiliary G_ϕ, which learns to predict the difference between two HR data samples, as formulated in Eq. (2). While F_θ operates on single LR input, G_ϕ utilizes two LR data samples along with one HR data sample, enabling the two models to make predictions from distinct informational viewpoints. This architectural design promotes diversity into the learning process (See Table 15 in Appendix L) .

3.6 Ablation Studies on Inductive Biases

Table 6 presents ablation results on the two inductive biases in SuperMeshNet, demonstrating their effect on super-resolution performance in terms of the RMSE across six different MPNNs[3]. For all MPNNs, the incorporation of node-level centering (N) and message-level centering (M) into the MPNN architecture leads to substantial improvements in super-resolution performance (*i.e.*, a lower RMSE) compared to MPNNs without inductive biases (O). We provide further analysis on the underlying factors contributing to the performance improvement in Appendix T.

Figure 6: Ablation studies on inductive biases. The RMSE of SuperMeshNet across six MPNNs under four inductive bias conditions (O: without inductive biases, N: node-level centering, M : message-level centering, and N+M: both node-level and message-level centerings) trained with $N_h = 20$ and $N = 200$ for Dataset 1 is compared. For each MPNN, the lowest RMSE value among the four inductive bias conditions is highlighted as **bold**.

MPNN	RMSE			
	O	N	M	N + M
GCN	0.0613	**0.0431**	-	-
SAGE	0.0589	0.0493	0.0528	**0.0450**
GAT	0.0544	**0.0457**	-	-
GTR	0.0451	0.0405	0.0438	**0.0385**
GIN	0.0404	0.0290	0.0281	**0.0277**
MGN	0.0269	0.0237	0.0247	**0.0226**

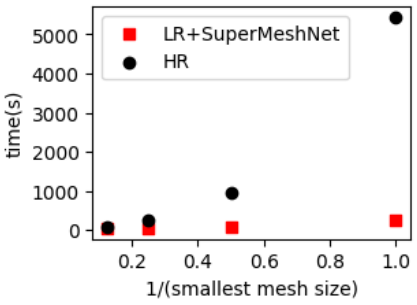

Figure 7: Comparison of the HR simulation time with the combined computational cost of LR simulation and subsequent SuperMeshNet's inference. Each value is measured over 1,000 data samples.

3.7 Inference Time Analysis

Figure 7 demonstrates that SuperMeshNet substantially reduces the computational cost of HR simulation. Specifically, the combined computational cost of LR simulation and subsequent SuperMeshNet inference is significantly lower than the cost of HR simulation. The time saving becomes even more pronounced for small mesh sizes, where the computational cost required for HR simulation grows rapidly. For details on the relative computational cost of kNN interpolation and the training time of SuperMeshNet, we refer to Appendices V and P, respectively.

3.8 Effect of Our Shared Feature Extractor

As addressed in subsection 2.3, in SuperMeshNet, the two models F_θ and G_ϕ use a shared feature extractor. To assess its effect, we compare our default shared-extractor implementation with a variant that employs separate extractors for the two models. As shown in Table 7, using separate extractors improves accuracy, but at a substantially higher computational cost. This is because each extractor can specialize more strongly for the distinct roles of the two models. However, since the roles of F_θ and G_ϕ are not entirely different, the performance gain remains modest. In contrast, sharing the extractor reduces training time by more than a factor of three by avoiding redundant feature extraction on the same input under complementary learning. Given this trade-off between accuracy and computational complexity, we adopt the shared feature extractor in SuperMeshNet.

Table 7: Comparison of shared feature extractor with separate feature extractors in terms of RMSE and training time. Here, MGN is employed as an MPNN for each method. Training is conducted when N_h=20 and N=200 for Dataset 1.

Methods	RMSE	Training time (s)
Shared feature extractor	0.0192	962.85
Separate feature extractors	0.0226	263.72

[3]Note that, in MPNNs such as GCN and GAT, the message-level centering cannot be employed independently since the message aggregation in Eq. (10) and the node embedding updates are integrated into a single step.

3.9 Application to Time-Dependent PDEs and Real-World Geometry

Figure 8 presents the applicability of SuperMeshNet to real-world geometry , when MGN is employed as an MPNN. Figure 8 shows the error distribution on the real-world geometry dataset, where darker blue indicates higher error. Figure 8 apparently demonstrates that SuperMeshNet, trained with only 20 HR data samples, exhibits even smaller errors than the case of the fully supervised model trained with 200 HR data samples, especially around the *front region of the motorbike*. Additionally, Figure 9 displays the qualitative comparisons of prediction on the time-depndent PDE dataset 1. The prediction by SuperMeshNet, trained with only 20 HR data samples, is even closer to the ground truth than the case of the fully supervised model trained with 200 HR data samples, particularly around the *blue wake region* behind the cylinder. Furthermore, as visualized in Figure 10, SuperMeshNet successfully predicts the HR counterpart even when the HR data differ substantially from the LR input, whereas conventional fully supervised learning fails despite having access to significantly more HR data. We refer to Appendix N for quantitative results for the three datasets as well as conditions and additional visualizations for the time-dependent PDE dataset 2.

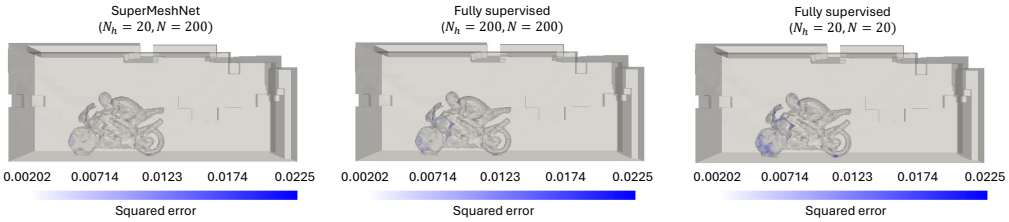

Figure 8: Comparison of squared error fields between SuperMeshNet and fully supervised baselines on real-world geometry dataset. Here, N_h and N represent the number of HR and LR data samples, respectively. For all cases, MGN is utilized as the underlying MPNN.

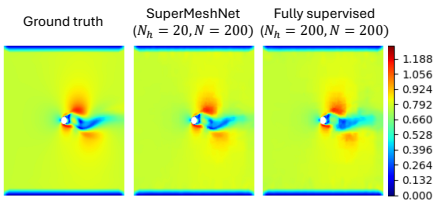

Figure 9: Comparison of ground-truth and predicted fluid flow speeds on the time-dependent PDE dataset 1. Here, N_h and N represent the number of HR and LR data samples, respectively. For all cases, MGN is utilized as the underlying MPNN.

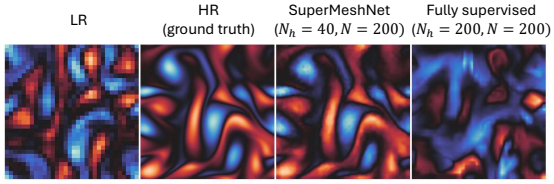

Figure 10: Comparison of LR input, ground-truth HR, and two vorticity predictions produced by SuperMeshNet and full supervision on the time-dependent PDE dataset 2. Here, N_h and N represent the number of HR and LR data samples, respectively. For all cases, MGN is utilized as the underlying MPNN.

Additional experimental analyses—including comparisons with standard normalization, scalability evaluation, training stability anlayses, singular point analysis, application to a CNN-based architecture, and comparison with a physics-informed neural network (PINN)—are provided in Appendices O, Q, R, S, X, and Y, respectively.

4 Conclusions and Limitations

In this paper, we explored the open problem of super-resolution for mesh-based simulations by presenting SuperMeshNet, which judiciously harnesses complementary learning and inductive biases to achieve remarkable HR data efficiency. The complementary learning enabled effective utilization of unpaired LR data, while the inductive biases further enhanced performance across a variety of MPNN architectures. We expect that this improvement can facilitate the broader adoption of simulations across various engineering disciplines, potentially accelerating innovation by lowering the barriers to conducting complex simulations. However, while our complementary learning mechanism achieves shorter training time compared to benchmark semi-supervised learning methods, it still remains slower than the case of full supervision. Further reducing the computational cost of complementary learning is a potential avenue of our future research.

REFERENCES

Martin S. Alnaes, Anders Logg, Kristian B. Ølgaard, Marie E. Rognes, and Garth N. Wells. Unified form language: A domain-specific language for weak formulations of partial differential equations. *ACM Transactions on Mathematical Software*, 40, 2014. doi: 10.1145/2566630.

Rajat Arora. Physrnet: Physics informed super-resolution network for application in computational solid mechanics. In *2022 IEEE/ACM International Workshop on Artificial Intelligence and Machine Learning for Scientific Applications (AI4S)*, pages 13–18, 2022. doi: 10.1109/AI4S56813.2022.00008.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Igor A. Baratta, Joseph P. Dean, Jørgen S. Dokken, Michal Habera, Jack S. Hale, Chris N. Richardson, Marie E. Rognes, Matthew W. Scroggs, Nathan Sime, and Garth N. Wells. DOLFINx: the next generation FEniCS problem solving environment. preprint, 2023.

Shivam Barwey, Pinaki Pal, Saumil Patel, Riccardo Balin, Bethany Lusch, Venkatram Vishwanath, Romit Maulik, and Ramesh Balakrishnan. Mesh-based super-resolution of fluid flows with multiscale graph neural networks, 2024. URL https://arxiv.org/abs/2409.07769.

Oussama Boussif, Yoshua Bengio, Loubna Benabbou, and Dan Assouline. MAgnet: Mesh agnostic neural PDE solver. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=bx2roi8hca8.

Ulf Brefeld, Thomas Gärtner, Tobias Scheffer, and Stefan Wrobel. Efficient co-regularised least squares regression. *Proceedings of the 23rd international conference on Machine learning*, 2006. URL https://api.semanticscholar.org/CorpusID:2025415.

Yadi Cao, Menglei Chai, Minchen Li, and Chenfanfu Jiang. Efficient learning of mesh-based physical simulation with bi-stride multi-scale graph neural network. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 3541–3558. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/cao23a.html.

Yutian Chen, Max Welling, and Alex Smola. Super-samples from kernel herding. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, UAI'10, page 109–116, Arlington, Virginia, USA, 2010. AUAI Press. ISBN 9780974903965.

Weihang Dai, Xiaomeng Li, and Kwang-Ting Cheng. Semi-supervised deep regression with uncertainty consistency and variational model ensembling via bayesian neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(6):7304–7313, 2023.

Filipe de Avila Belbute-Peres, Thomas D. Economon, and J. Zico Kolter. Combining differentiable pde solvers and graph neural networks for fluid flow prediction. In *International Conference on Machine Learning*, 2020. URL https://api.semanticscholar.org/CorpusID:220424832.

Huayu Deng, Xiangming Zhu, Yunbo Wang, and Xiaokang Yang. Discovering message passing hierarchies for mesh-based physics simulation, 2024. URL https://arxiv.org/abs/2410.03779.

Meire Fortunato, Tobias Pfaff, Peter Wirnsberger, Alexander Pritzel, and Peter W. Battaglia. Multiscale meshgraphnets. *ArXiv*, abs/2210.00612, 2022. URL https://api.semanticscholar.org/CorpusID:251011532.

Rini Jasmine Gladstone, Helia Rahmani, Vishvas Samuel Suryakumar, Hadi Meidani, Marta D'Elia, and Ahmad Zareei. Mesh-based gnn surrogates for time-independent pdes. *Scientific Reports*, 14, 2024. URL https://api.semanticscholar.org/CorpusID:267579608.

Yanan Guo, Junqiang Song, Xiaoqun Cao, Chuanfeng Zhao, and Hongze Leng. Physics field super-resolution reconstruction via enhanced diffusion model and fourier neural operator. *Theoretical and Applied Mechanics Letters*, 15(5):100604, 2025. ISSN 2095-0349. doi: https://doi.org/10.1016/j.taml.2025.100604. URL https://www.sciencedirect.com/science/article/pii/S2095034925000364.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

Pin-Yen Huang, Szu-Wei Fu, and Yu Tsao. Rankup: Boosting semi-supervised regression with an auxiliary ranking classifier. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=d2lPM1Aczs.

Sergey Ioffe and Christian Szegedy. Batch normalization: accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 448–456. JMLR.org, 2015.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Dmitrii Kochkov, Jamie A. Smith, Ayya Alieva, Qing Wang, Michael P. Brenner, and Stephan Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21):e2101784118, 2021a. doi: 10.1073/pnas.2101784118. URL https://www.pnas.org/doi/abs/10.1073/pnas.2101784118.

Dmitrii Kochkov, Jamie A. Smith, Ayya Alieva, Qing Wang, Michael P. Brenner, and Stephan Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21), 2021b. ISSN 0027-8424. doi: 10.1073/pnas.2101784118. URL https://www.pnas.org/content/118/21/e2101784118.

Georgios Kostopoulos, Stamatis Karlos, Sotiris Kotsiantis, Omiros Ragos, Shailesh Tiwari, Munesh Trivedi, and Mohan L. Kohle. Semi-supervised regression: A recent review. *J. Intell. Fuzzy Syst.*, 35(2):1483–1500, January 2018. ISSN 1064-1246. doi: 10.3233/JIFS-169689. URL https://doi.org/10.3233/JIFS-169689.

Matthew Li and Christopher McComb. Using physics-informed generative adversarial networks to perform super-resolution for multiphase fluid simulations. *Journal of Computing and Information Science in Engineering*, 22(4):044501, 02 2022. ISSN 1530-9827. doi: 10.1115/1.4053671. URL https://doi.org/10.1115/1.4053671.

Octavi Obiols-Sales, Abhinav Vishnu, Nicholas P. Malaya, and Aparna Chandramowlishwaran. Surfnet: Super-resolution of turbulent flows with transfer learning using small datasets. In *Proceedings of the 30th International Conference on Parallel Architectures and Compilation Techniques*, PACT '21, page 331–344. IEEE Press, 2024. ISBN 9781665442787. doi: 10.1109/PACT52795.2021.00031. URL https://doi.org/10.1109/PACT52795.2021.00031.

OpenCFD. Openfoam v2412. 2024. URL https://openfoam.com.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=roNqYL0_XP.

Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: deep hierarchical feature learning on point sets in a metric space. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 5105–5114, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2018.10.045. URL https://www.sciencedirect.com/science/article/pii/S0021999118307125.

Bruno Alves Ribeiro, João Alves Ribeiro, Faez Ahmed, Hugo Penedones, Jorge Belinha, Luís Sarmento, Miguel Anibal Bessa, and Sérgio Tavares. Simustruct: Simulated structural plate with holes dataset with machine learning applications. In *Workshop on "Machine Learning for Materials" ICLR 2023*, 2023. URL https://openreview.net/forum?id=s3tOuyR1vM7.

Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Mądry. How does batch normalization help optimization? In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 2488–2498, Red Hook, NY, USA, 2018. Curran Associates Inc.

Matthew W. Scroggs, Igor A. Baratta, Chris N. Richardson, and Garth N. Wells. Basix: a runtime finite element basis evaluation library. *Journal of Open Source Software*, 7(73):3982, 2022a. doi: 10.21105/joss.03982.

Matthew W. Scroggs, Jørgen S. Dokken, Chris N. Richardson, and Garth N. Wells. Construction of arbitrary order finite element degree-of-freedom maps on polygonal and polyhedral cell meshes. *ACM Transactions on Mathematical Software*, 48(2):18:1–18:23, 2022b. doi: 10.1145/3524456.

Yunsheng Shi, Zhengjie Huang, shikun feng, Hui Zhong, Wenjin Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification, 2021. URL https://openreview.net/forum?id=B9t708KMr9d.

Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 1195–1204, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

Sebastian J Wetzel, Roger G Melko, and Isaac Tamblyn. Twin neural network regression is a semi-supervised regression algorithm. *Machine Learning: Science and Technology*, 3(4): 045007, oct 2022. doi: 10.1088/2632-2153/ac9885. URL https://dx.doi.org/10.1088/2632-2153/ac9885.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryGs6iA5Km.

Kazuo Yonekura, Kento Maruoka, Kyoku Tyou, and Katsuyuki Suzuki. Super-resolving 2d stress tensor field conserving equilibrium constraints using physics-informed u-net. *Finite Elements in Analysis and Design*, 213:103852, 2023. ISSN 0168-874X. doi: https://doi.org/10.1016/j.finel.2022.103852. URL https://www.sciencedirect.com/science/article/pii/S0168874X22001251.

Zhi-Hua Zhou and Ming Li. Semi-supervised regression with co-training. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, IJCAI'05, page 908–913, San Francisco, CA, USA, 2005. Morgan Kaufmann Publishers Inc.

Table of Contents

This table provides a mapping between the appendix and the corresponding sections in the main manuscript.

A NOTATIONS

Table 8: Summary of notations

Notation	Description
μ	PDE parameters
F_θ, G_ϕ	neural network models
u_l, u_l^q, u_l^r, u_l^s, u_l^α, u_l^β, u_l^γ	LR data samples
u_h, u_h^q, u_h^r, u_h^s, u_h^α, u_h^β, u_h^γ	HR data samples
$\hat{u}_h$, $\hat{u}_h^\alpha$, $\hat{u}_h^\beta$, $\hat{u}_h^\gamma$	prediction by F_θ
$\hat{u}_h^{\alpha\beta}$, $\hat{u}_h^{\beta\gamma}$, $\hat{u}_h^{\gamma\alpha}$	prediction by G_ϕ
M_l	LR mesh
M_h	HR mesh
P_l	nodal positions of LR mesh
P_h	nodal positions of HR mesh
E_l	edges of LR mesh
E_h	edges of HR mesh
n_l	number of nodes in LR mesh
n_h	number of nodes in HR mesh
n	number of nodes in graph
$\mathcal{D}_a$	paired LR–HR training dataset
$\mathcal{D}_b$	unpaired LR training dataset
N	total number of data samples = number of LR data samples
N_h	number of paired LR–HR data samples = number of HR data
$\mathcal{L}_F$	loss function for training F_θ
$\mathcal{L}_G$	loss function for training G_ϕ
ℓ	mean squared error
kNN	k-nearest neighbor interpolation
g_l	input graph
${}^0X_l^q$	node feature
${}^0x_l^q$, ${}^{H_l}x_l^q$, ${}^0x_h^q$, ${}^{H_h}x_h^q$	node embeddings
x_i	node embedding of node i
x_j	node embedding of node j
e_{ij}	edge embedding between nodes i and j
msg_{ij}	message between nodes i and j
agg_i	aggregated message of node i
f_m	message function
f_x	node embedding update function
$\mathcal{N}(i)$	set of neighboring nodes of i

Table 8 summarizes the notations used throughout the paper.

B Related Work

MPNNs for mesh-based simulations. Conventional methods for solving PDEs, such as the FEM, typically rely on mesh-based approaches. Nevertheless, CNNs, which are not well-suited for irregular mesh-based data, have gained popularity as surrogate models, due to their simplicity and efficiency (Pfaff et al., 2021). Among notable advancements, MeshGraphNet (MGN) (Pfaff et al., 2021) represents a significant breakthrough, demonstrating that MPNNs can outperform CNN-based models for mesh-based simulation tasks. However, MGN, like other MPNNs, is constrained by a limited interaction range. Since MPNNs exchange information with immediate neighbors, extending interactions to nodes farther away requires stacking additional message passing layers. Most MPNNs suffer from over-smoothing, a phenomenon where output node embeddings become overly uniform as the number of stacked message passing layers increases, leading to performance degradation. While MGN has been reported to exhibit robustness against over-smoothing, an increased number of message passing layers still results in significantly higher computational costs. To address this limitation, researchers (Cao et al., 2023; Fortunato et al., 2022; Gladstone et al., 2024) proposed incorporating additional coarse meshes alongside the original fine mesh. This approach enables messages to propagate more efficiently across the coarse meshes while allowing the fine mesh to finely adjust node embeddings. Moreover, an attention mechanism was integrated into MGN to further refine the aggregation function, enhancing the neural network's ability to adaptively prioritize relevant information during message passing (Deng et al., 2024).

Super-resolution for simulations. Similarly as in surrogate models for simulations, early super-resolution models for simulations predominantly employed CNN-based image super-resolution architectures, such as SRGAN (Li and McComb, 2022) and UNet (Yonekura et al., 2023). As pioneering work, CFD-GCN (de Avila Belbute-Peres et al., 2020) introduced GCNs for the super-resolution of computational fluid dynamics (CFD) simulations. This method demonstrated both improved generalization to unseen data and enhanced cost efficiency. More recently, advanced MPNN architectures like SRGNN was applied to the super-resolution of fluid flows (Barwey et al., 2024). Despite these advancements, inductive biases tailored for MPNNs in the context of super-resolution for mesh-based simulations are largely underexplored.

Semi-supervised regression. Semi-supervised regression involves predicting real-valued output using both labeled and unlabeled datasets. Compared to semi-supervised classification, semi-supervised regression remains largely underexplored (Kostopoulos et al., 2018). A co-training approach typically splits input features into groups, with each group used to train a separate model (Brefeld et al., 2006). However, in scenarios with limited features, such as FEM-relevant data including only nodal positions and nodal values, splitting features can lead to insufficient information for accurate predictions. As an alternative, CoREG (Zhou and Li, 2005) was presented to eliminate the need for feature splitting by using two k-nearest neighbor (kNN) regressors with different distance metrics. Unfortunately, this approach is restricted to kNN regressors and is unsuitable for predicting values at the node-level. A recent method, Rankup (Huang et al., 2024), reformulated regression tasks into classification tasks to leverage a rich set of methodologies developed for semi-supervised classification. However, this technique lacks generalizability for regression tasks involving mesh-based graph data. The Mean-Teacher framework (Tarvainen and Valpola, 2017), though not originally designed for mesh-based data, has potential for dealing with mesh-based graph data. It involves teacher and student models with teacher's weights updated as the exponential moving average of the student's weights. In contrast, the recently proposed UCVME framework (Dai et al., 2023) has demonstrated superior performance over the Mean-Teacher (Tarvainen and Valpola, 2017) by incorporating uncertainty consistency and utilizing a variational model ensemble. However, because Mean-Teacher (Tarvainen and Valpola, 2017) and UCVME (Dai et al., 2023) both employ identically structured models predicting the same target, they exhibit reduced psuedo-label diversity during training. This uniformity diminishes synergy between the models and consequently hinders learning efficiency. Additionally, Twin Neural Network Regression (TNNR) (Wetzel et al., 2022) is applicable to mesh-based predictions when an appropriate model architecture is involved. However, it involves only a single twin neural network, thus lacking the synergistic benefits of mutual supervision.

C Pseudo-code for Complementary Learning

Algorithm 1 Complementary learning

Input: paired LR–HR dataset: $\mathcal{D}_a = \{(I^q, u_h^q) \mid q = 1, 2, \cdots, N_h\}$, unpaired LR dataset: $\mathcal{D}_b = \{I^q \mid q = N_{h+1}, N_{h+2}, \cdots, N\}$, neural network models: feature extractor E_c, F_θ's decoder D_F, and G_ϕ's decoder D_G, maximum epoch: *ep*, learning rate: η, early stopping criterion
Output: Trained neural network models: E_c, D_F, and D_G
for *epoch* $\leftarrow 1$ to *ep* **do**
  for *step* $\leftarrow 1$ to N **do**
    Sample paired LR–HR data $(u_l^\alpha, u_h^\alpha), (u_l^\beta, u_h^\beta) \in \mathcal{D}_a$
    Sample unpaired LR data $u_l^\gamma \in \mathcal{D}_b$
    Compute node embeddings by E_c:

$$x_\alpha \leftarrow E_c(u_l^\alpha),\ x_\beta \leftarrow E_c(u_l^\beta),\ x_\gamma \leftarrow E_c(u_l^\gamma)$$

    Compute prediction by D_F:

$$\hat{u}_h^\alpha \leftarrow D_F(x^\alpha),\ \hat{u}_h^\beta \leftarrow D_F(x^\beta),\ \hat{u}_h^\gamma \leftarrow D_F(x^\gamma)$$

    Compute prediction by D_G:

$$\hat{u}_h^{\alpha\beta} \leftarrow D_G(x^\alpha, x^\beta),\ \hat{u}_h^{\beta\gamma} \leftarrow D_G(x^\beta, x^\gamma),\ \hat{u}_h^{\gamma\alpha} \leftarrow D_G(x^\gamma, x^\alpha)$$

    Compute loss for F_θ:

$$\begin{aligned}\mathcal{L}_F =& \ell(\hat{u}_h^\alpha, u_h^\alpha) + \ell(\hat{u}_h^\beta, u_h^\beta) + \ell\big(\hat{u}_h^\gamma, \hat{u}_h^{\gamma\alpha} + kNN(u_h^\alpha; P_h^\alpha \to P_h^\gamma)\big) \\ &+ \ell\big(\hat{u}_h^\gamma, kNN(u_h^\beta - \hat{u}_h^{\beta\gamma}; P_h^\beta \to P_h^\gamma)\big)\end{aligned}$$

    Compute loss for G_ϕ:

$$\begin{aligned}\mathcal{L}_G =& \ell\big(\hat{u}_h^{\alpha\beta}, u_h^\alpha - kNN(u_h^\beta; P_h^\beta \to P_h^\alpha)\big) + \ell\big(\hat{u}_h^{\gamma\alpha}, \hat{u}_h^\gamma - kNN(u_h^\alpha; P_h^\alpha \to P_h^\gamma)\big) \\ &+ \ell\big(\hat{u}_h^{\beta\gamma}, u_h^\beta - kNN(\hat{u}_h^\gamma; P_h^\gamma \to P_h^\beta)\big)\end{aligned}$$

    Compute total loss:

$$\mathcal{L} = \mathcal{L}_F + \mathcal{L}_G$$

    Compute gradients: $\nabla_\psi \mathcal{L}$ where ψ is the parameters of E_c, D_F, and D_G
    Update weights:

$$\psi \leftarrow \psi - \eta \nabla_\psi \mathcal{L}$$

  end for
  if early stopping criterion is met **then**
    Break
  end if
end for
return E_c, D_F, and D_G

We present the pseudo-code of our complementary learning mechanism in SuperMeshNet.

D kNN INTERPOLATION IN SUPERMESHNET

We provide a brief explanation of the kNN interpolation procedure in SuperMeshNet, which projects values defined on nodes of a source mesh onto nodes of a target mesh.

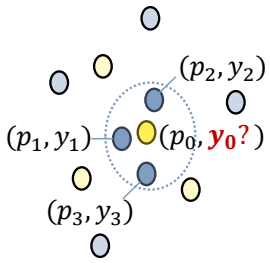

Figure 11: Schematic illustration of kNN interpolation with $k = 3$. Yellow nodes belong to the target mesh, blue nodes to the source mesh, and the darker blue nodes indicate the k nearest neighbors of the darker yellow node. Given the positions of the k nearest source nodes p_i $(1 \leq i \leq k)$, their corresponding values y_i, and the target node position p_0, the value at the target node y_0 can be estimated via weighted averaging.

1. **Find k nearest neighbors (kNN).** For each node in the target mesh, identify the k closest nodes in the source mesh. For example, as illustrated in Figure 11, the darker blue nodes represent the three nearest neighbors of the darker yellow node.
2. **Known information.** The nodal positions of the k nearest source nodes p_i $(1 \leq i \leq k)$, their values y_i, and the target node position p_0.
3. **Unknown quantity.** The value at the target node, denoted by y_0.
4. **Compute the target node value via weighted averaging.** The interpolation weight for each neighbor is defined as the inverse squared distance from the target node:

$$w_i = \frac{1}{d(p_0, p_i)^2}.$$

The interpolated value at the target node y_0 is then obtained as

$$y_0 = \frac{\sum_{i=1}^{k} w_i y_i}{\sum_{i=1}^{k} w_i}.$$

E MODEL ARCHITECTURES

The architecture of F_θ, illustrated in Figure 12, is basically built upon SRGNN (Barwey et al., 2024), with the key difference that the MPNNs in the LR and HR processors are enriched with our proposed inductive biases. The G_ϕ, visualized in Figure 13, extends F_θ to accommodate two input samples, maintaining a comparable structure. A notable architectural feature is the use of a shared feature extractor between F_θ and G_ϕ, which helps reduce computational costs during training. A detailed description of the model architectures follows.

E.1 MODEL ARCHITECTURE OF F_θ

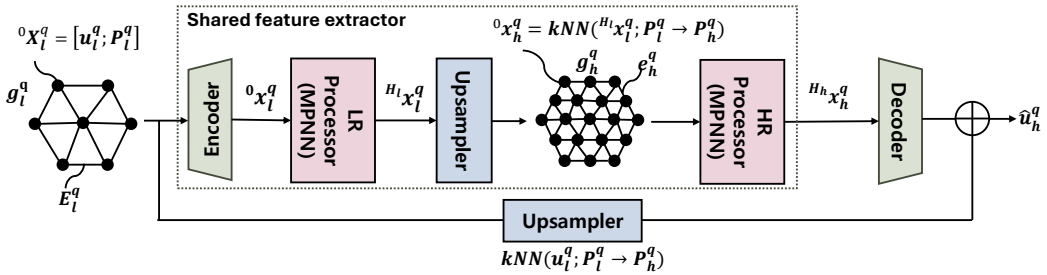

Figure 12: The schematic overview of the primary model F_θ.The F_θ aims to predict $\hat{u}_h^q$ targeting HR data sample u_h^q from LR data sample u_l^q. The LR data sample u_l^q is input to the F_θ as a part of node feature ${}^0X_l^q$ of an input graph g_l^q.

The role of F_θ is to transform LR data into HR data, which is conducted by the lowermost upsampler in Figure 12. To surpass the performance of kNN interpolation by the lowermost upsampler, we have introduced additional upsampling in latent space. Specifically, an encoder maps the physical quantities into high-dimensional latent space. The LR processor applies message passing to refine LR representations, which are then upsampled to HR latent embeddings. Subsequently, the HR processor applies additional message passing to further enhance the HR representations. Finally, a decoder maps the latent embeddings back to the physical space. The final HR output is obtained by adding the two upsampled HR fields: one from the kNN-based upsampler and the other from the latent-space upsampling pathway.

More precisely, F_θ is designed to make prediction $\hat{u}_h^q$ from an LR data sample u_l^q. The LR data sample u_l^q is input to the F_θ as a form of an input graph g_l^q. More specifically, input graph g_l^q's node feature ${}^0X_l^q$ is the concatenation of LR data sample u_l^q and node position P_l^q. Depending on MPNN types used in the LR and HR processors (refer to Appendix F), the g_l^q may further include edge feature, which is the concatenation of positions of source and target nodes of the edge E_l^q. The F_θ comprises an encoder, an LR processor, upsamplers, an HR processor, and a decoder. The encoder can be an multi-layer perceptron (MLP) that can convert low-dimensional ${}^0X_l^q$ to high-dimensional node embeddings ${}^0x_l^q$. The LR processor updates the node embedding ${}^0x_l^q$ to ${}^{H_l}x_l^q$ through stacked H_l MPNN layers enriched by our inductive biases. Here, the prescript 0 and H_l indicate the index of the MPNN layers. The node embedding ${}^{H_l}x_l^q$ defined on nodes located at P_l^q is upsampled onto nodes of g_h^q positioned at P_h^q by using kNN interpolation. The HR processor similarly updates ${}^0x_h^q$ to ${}^{H_h}x_h^q$ through stacked H_h MPNN layers equipped with our inductive biases. Then, the decoder, which is an MLP, predicts low-dimensional output from ${}^{H_h}x_h^q$. Finally, the LR data sample u_l^q upsampled onto P_h^q by kNN interpolation is added to the output of the decoder to obtain the final prediction $\hat{u}_h^q$. The upsampled LR data sample serves as a rough estimation of the prediction, enabling the super-resolution model to focus on learning the finer details, thereby simplifying the learning task.

E.2 MODEL ARCHITECTURE OF G_ϕ

The model G_ϕ is responsible for predicting the difference between two HR samples corresponding two LR inputs. Again, to go beyond simple kNN-based upsampling by upsampler in Figure 13, we

have further performed latent-space processing. We have extracted latent embeddings from the two LR inputs using a shared encoder. The shared encoder is the one used for the model F_θ depicted in Figure 12. Then, we subtracted the embeddings, and decoded the result to predict the HR difference. Here, we incorporated subtraction because the goal is to predict the difference between two HR samples. The final HR output is obtained by adding the two upsampled HR fields: one from the kNN-based upsampler and the other from the latent-space upsampling pathway. The interpolators in the Figure 13 serve only to address mesh mismatches when the two LR samples are defined on different meshes. Since the underlying computational domain geometry may vary across samples, direct point-wise operations, such as subtraction or addition, are generally infeasible. To overcome this, we apply kNN interpolation to project one mesh onto another, enabling consistent alignment between mesh structures.

More precisely, the auxiliary model G_ϕ is designed to make prediction $\hat{u}_h^{rs}$ from a pair of LR data samples u_l^r and u_l^s. More specifically, The two input LR data samples u_l^r and u_l^s are fed into the G_ϕ as parts of node features of two input graphs g_l^r and g_l^s, respectively. In order to reduce computational costs, F_θ and G_ϕ share a feature extractor comprising the encoder, the LR processor, the interpolator, and the HR processor. The feature extractor returns node embeddings x_h^r and x_h^s from input graphs g_l^r and g_l^s, respectively. Then, x_h^s is subtracted from x_h^s to yield x_h^{rs}. Here, kNN interpolator is used to enable subtraction operation between two node embeddings x_h^r and x_h^s defined at different nodal positions. The x_h^{rs} is fed into the decoder, and the upsampled difference between u_l^r and u_l^s through kNN interpolation is added to the decoder's output. Here, the interpolated difference between u_l^r and u_l^s also serves as a rough estimation of the prediction $\hat{u}_h^{rs}$. Again, an kNN interpolator is used to enable subtraction operation between u_l^r and u_l^s defined on different nodal positions.

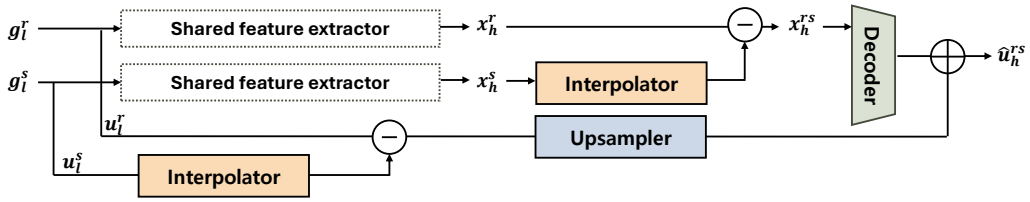

Figure 13: The schematic overview of the auxiliary model G_ϕ. The G_ϕ aims to predict $\hat{u}_h^{rs}$ targeting difference between two input LR data samples u_l^r and u_l^s. The two input LR data samples u_l^r and u_l^s are fed into the G_ϕ as parts of node features of two input graphs g_l^r and g_l^s, respectively. In order to reduce computational cost, F_θ and G_ϕ share a feature extractor in Figure 12 consisting of an encoder, an LR processor, an upsampler, and an HR processor.

F INCORPORATION OF INDUCTIVE BIASES INTO MPNN ARCHITECTURES

This section describes how inductive biases are incorporated into each of MPNN models.

F.1 INDUCTIVE BIASES-ENRICHED GCN (KIPF AND WELLING, 2017)

$$
\begin{aligned}
&msg_{ij} = \Theta^T e_{ji} x_j \\
&x_i = agg_i = \sum_{j \in \mathcal{N}(i)} msg_{ij} \\
&x_i \leftarrow x_i - \frac{1}{n} \sum_{i=1}^{n} x_i,
\end{aligned}
\tag{13}
$$

where Θ is a learnable parameter.

F.2 INDUCTIVE BIASES-ENRICHED GRAPHSAGE (SAGE) (HAMILTON ET AL., 2017)

$$
\begin{aligned}
&msg_{ij} = x_j \\
&agg_i = \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} msg_{ij} \\
&agg_i \leftarrow agg_i - \frac{1}{n} \sum_{i=1}^{n} agg_i \\
&x_i \leftarrow W_1 x_i + W_2 agg_i \\
&x_i \leftarrow x_i - \frac{1}{n} \sum_{i=1}^{n} x_i,
\end{aligned}
\tag{14}
$$

where W_1 and W_2 are learnable parameters.

F.3 INDUCTIVE BIASES-ENRICHED GAT (VELIČKOVIĆ ET AL., 2018)

$$
\begin{aligned}
&\alpha_{ij} = \frac{exp(LeakyReLU(a_s^T \Theta_s x_i + a_t^T \Theta_t x_j)}{\sum_{k \in \mathcal{N}(i)} exp(LeakyReLU(a_s^T \Theta_s x_i + a_t^T \Theta_t x_k))} \\
&msg_{ij} = \alpha_{ij} \Theta_t x_j \\
&x_i = agg_i = \sum_{j \in \mathcal{N}(i)} msg_{ij} \\
&x_i \leftarrow x_i - \frac{1}{n} \sum_{i=1}^{n} x_i,
\end{aligned}
\tag{15}
$$

where a_s, s_t Θ_s, and Θ_t are learnable parameters.

F.4 INDUCTIVE BIASES-ENRICHED GRAPH TRANSFORMER (GTR) (SHI ET AL., 2021)

$$
\begin{aligned}
&\alpha_{ij} = softmax((W_3 x_i)^T (W_4 x_j)) \\
&msg_{ij} = \alpha_{ij} W_2 x_j \\
&agg_i = \sum_{j \in \mathcal{N}(i)} msg_{ij} \\
&agg_i \leftarrow agg_i - \frac{1}{n} \sum_{i=1}^{n} agg_i \\
&x_i \leftarrow W_1 x_i + agg_i \\
&x_i \leftarrow x_i - \frac{1}{n} \sum_{i=1}^{n} x_i,
\end{aligned}
\tag{16}
$$

where W_1, W_2, W_3 and W_4 are learnable parameters.

F.5 INDUCTIVE BIASES-ENRICHED GIN (XU ET AL., 2019)

$$
\begin{aligned}
&msg_{ij} = x_j \\
&agg_i = \sum_{j \in \mathcal{N}(i)} msg_{ij} \\
&agg_i \leftarrow agg_i - \frac{1}{n} \sum_{i=1}^{n} agg_i \\
&x_i \leftarrow MLP_\Theta((1+\epsilon) x_i + agg_i) \\
&x_i \leftarrow x_i - \frac{1}{n} \sum_{i=1}^{n} x_i,
\end{aligned}
\tag{17}
$$

where MLP_θ is a learnable MLP and ϵ is a learnable parameter.

F.6 INDUCTIVE BIASES-ENRICHED MESHGRAPHNET (MGN) (PFAFF ET AL., 2021)

$$
\begin{aligned}
&e_{ij} \leftarrow MLP_e(x_i, x_j, e_{ij}) \\
&msg_{ij} = e_{ij} \\
&agg_i = \sum_{j \in \mathcal{N}(i)} msg_{ij} \\
&agg_i \leftarrow agg_i - \frac{1}{n} \sum_{i=1}^{n} agg_i \\
&x_i \leftarrow MLP_x(x_i, agg_i) \\
&x_i \leftarrow x_i - \frac{1}{n} \sum_{i=1}^{n} x_i,
\end{aligned}
\tag{18}
$$

where MLP_e and MLP_x are learnable MLPs.

G Datasets for Experimental Evaluations

G.1 Dataset 1

The first dataset is inspired by simustruct (Ribeiro et al., 2023), the dataset for machine learning-based methods in structural analysis. Examples of HR and LR data samples from Dataset 1 are visualized in Figure 14. As depicted in the figure, the computational domain is a rectangle measuring 0.25×0.5 in the x- and y-directions, containing six circular holes, each with a diameter of 0.05. For the HR mesh, the mesh size around outer four sides is 10×10^{-3}, while the mesh size around the circular holes is set to be 4×10^{-3}. For the LR mesh, the mesh size around the outer sides is 40×10^{-3}, and the mesh size around the circular holes is 16×10^{-3}.

On the computation domain, the following linear elasticity equation is solved:

$$\begin{aligned} -\nabla \cdot \sigma(u) &= 0 \\ \sigma(u) &= \lambda \operatorname{tr}(\epsilon(u)) I + 2\mu\epsilon(u), \\ \epsilon(u) &= \frac{1}{2}\left(\nabla u + (\nabla u)^T\right), \end{aligned} \tag{19}$$

where $\sigma(u)$ is the stress tensor, λ and μ are Lamé's elasticity parameters for the material, I is the identity tensor, tr is the trace operator on a tensor, $\epsilon(u)$ is the symmetric strain tensor (symmetric gradient), and u is the displacement vector field.

A force of 1×10^8 is applied to the top side of the rectangle in angles between $40°$ and $140°$ relative to the x-axis, while the bottom side of the rectangle is fixed to zero displacement. Lamé's first and second parameters are 1.25, and 80.8×10^9, respectively. Von Mises stress is evaluated at each node of the meshes. In order to solve the equation for each dataset, we leverage FEniCSx (Baratta et al., 2023; Scroggs et al., 2022b;a; Alnaes et al., 2014), an open-source computing platform for solving PDEs with the FEM.

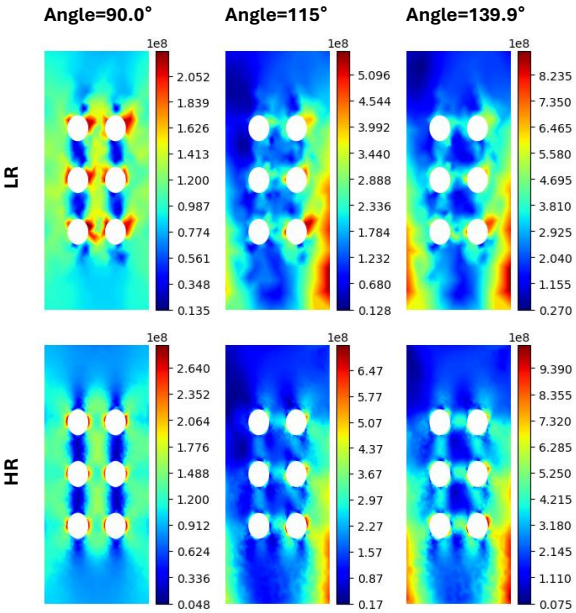

Figure 14: Examples of LR and HR data samples with various angles of applied force relative to the x-axis from Dataset 1.

G.2 DATASET 2

The geometry of the second dataset resembles that of the first dataset, with the primary difference being the shapes of the holes. Specifically, the holes in the second dataset are elliptical, with varying ratios between the lengths of the major and minor axes. The mesh sizes remain the same as those in Dataset 1. Similarly as in Dataset 1, the linear elasticity equation in Eq. (19) is solved. The applied force is directed along the y-axis, while all other conditions and constants remain identical to those in Dataset 1. Examples of HR and LR data samples from Dataset 2 are visualized in Figure 15.

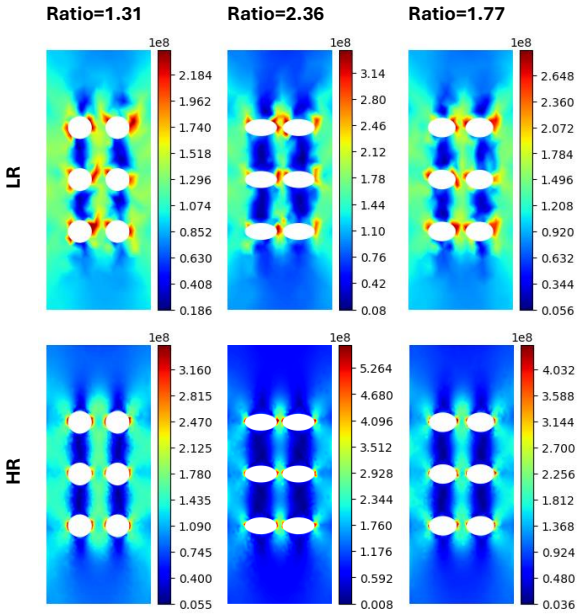

Figure 15: Examples of LR and HR data samples with various ratios between the lengths of the major and minor axes from Dataset 2.

G.3 DATASET 3

The geometry and mesh sizes of Dataset 3 are identical to those of Dataset 2. However, instead of solving the linear elasticity equation, the following Poisson equation is solved.

$$\nabla^2 u = 0, \tag{20}$$

where u is an electrical potential.

The boundary conditions are defined as follows: the four outer sides are set to 0 V, while the elliptical holes have alternating boundary values. Specifically, the holes centered at (0.08, 0.15), (0.17, 0.25), and (0.08, 0.35) are assigned a value of -1 V, whereas the holes centered at (0.17, 0.15), (0.08, 0.25), and (0.17, 0.35) are assigned a value of 1 V. The magnitude of the electric field is calculated at each node of the mesh. Examples of HR and LR data samples from the Dataset 3 are visualized in Figure 16.

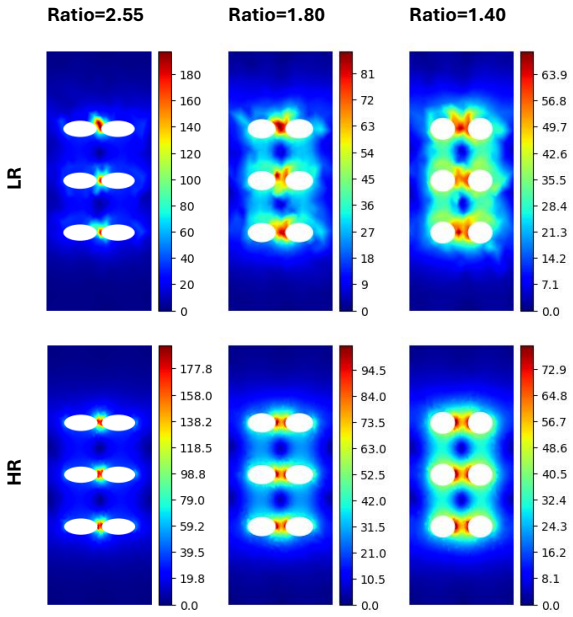

Figure 16: Examples of LR and HR data samples with various ratios between the lengths of the major and minor axes from Dataset 3.

G.4 TIME-DEPENDENT PDE DATASET 1

Examples of HR and LR data samples from the time-dependent PDE dataset 1 are visualized in Figure 17. As depicted in the figure, the computational domain is a square measuring 2×2 in the x- and y-directions, containing one cylinder at the center of the domain with a diameter of 0.05. The mesh size is set to be finer around the cylinder.

On the computation domain, the following incompressible Navier-Stokes equation is solved:

$$\nabla \cdot \mathbf{v} = 0, \tag{21}$$

$$\rho \left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right) = -\nabla p + \mu \nabla^2 \mathbf{u}, \tag{22}$$

where u is velocity, p is pressure, and ρ is density, μ is dynamic viscosity. The velocity at the left side of the square varies from 1 to 10 as time proceeds from 0 to 5. The density and the dynamic viscosity are set to be 1 and 10^{-5}, respectively. A speed, a magnitude of velocity, is evaluated at each node of the meshes. In order to solve the equation, we leverage OpenFOAM(OpenCFD, 2024), an open-source CFD toolbox.

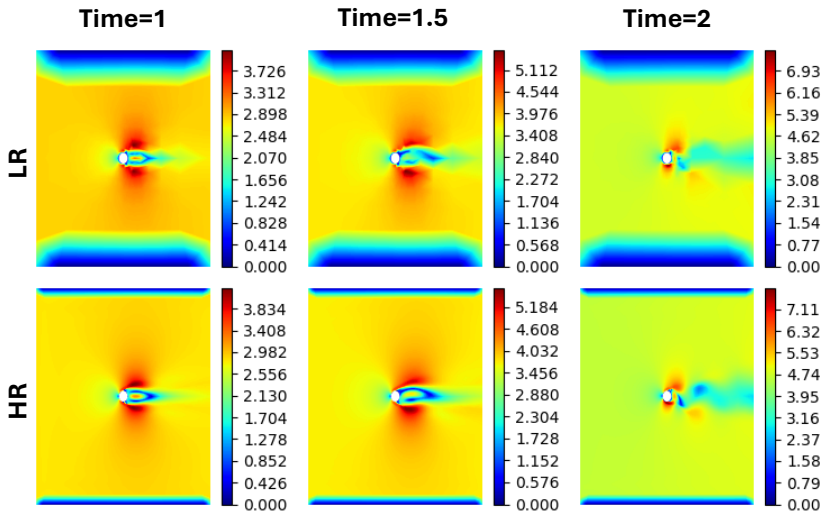

Figure 17: Examples of LR and HR data samples corresponding to multiple timestamps from the time-dependent PDE dataset 1.

G.5 TIME-DEPENDENT PDE DATASET 2

The time-dependent PDE dataset 2 is generated using the JAX-CFD library (Kochkov et al., 2021b), following the Kolmogorov-type forced 2D incompressible Navier–Stokes dynamics. Examples of HR and LR vorticity fields are visualized in Figure 18. In contrast to the time-dependent PDE dataset 1, this dataset is designed so that the HR and LR trajectories exhibit markedly different flow structures.

The computation domain is a periodic square domain $(0, 2\pi) \times (0, 2\pi)$. On this domain, we solve the 2D incompressible Navier–Stokes equations under external forcing:

$$\nabla \cdot \mathbf{u} = 0, \tag{23}$$

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{f}, \tag{24}$$

where $\mathbf{u}$ is velocity, p is pressure, and ν is the kinematic viscosity. We set $\nu = 10^{-2}$ and maximum velocity to be 7.

The HR simulation uses a 1024×1024 spectral grid, while the LR simulation uses a 32×32 grid. The HR time step is determined using the stability condition provided in JAX-CFD, and the LR time step is scaled proportionally to the grid resolution ratio. Both simulations are advanced using the Crank–Nicolson RK4 integrator implemented in the JAX-CFD spectral module.

The LR initial condition is downsampled in the Fourier space. We truncate the HR Fourier coefficients to retain only the lowest 32×32 modes and rescale the amplitudes to preserve energy. We evaluate the vorticity, $\omega = \nabla \times \mathbf{u}$, at every grid point for all time steps. A total of 300 frames are collected for both HR and LR simulations.

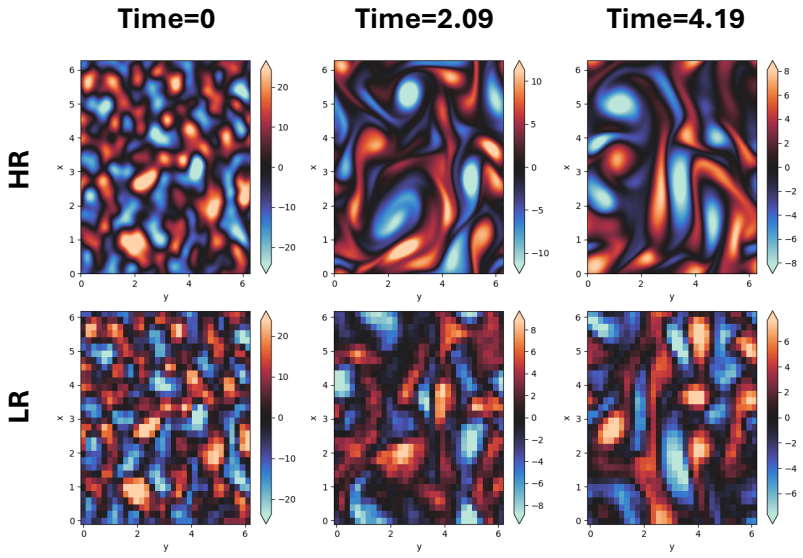

Figure 18: Examples of LR and HR data samples corresponding to multiple timestamps from the time-dependent PDE dataset 2.

G.6 REAL-WORLD GEOMETRY DATASET

An Example of HR and LR data samples from the real-world geometry dataset are visualized in Figure 19. As depicted in the figure, the computational domain is a rectangular box of size $20 \times 8 \times 8$ in the x-, y- and z-directions, containing a rider on a motorbike. The mesh size is set to be finer around the rider and the motorbike. The dataset is built upon a bike tutorial of OpenFOAM (OpenCFD, 2024) by varying angle of attack from 0° to -90°.

On the computation domain, the following Laplacian equation is solved:

$$\nabla^2 \phi = 0, \tag{25}$$

$$u = \nabla \phi, \tag{26}$$

where u is velocity, and ϕ is velocity potential.

Then, pressure is calculated utilizing the following Bernoulli equation:

$$p = p_{ref} + \frac{1}{2}(|u_{ref}|^2 - |u|^2), \tag{27}$$

where p_{ref} and u_{ref} are the pressure and velocity at a reference location, respectively. The speed of fluid at the left side of the rectangular box is set to be 20. In order to solve the equation, we leverage OpenFOAM (OpenCFD, 2024).

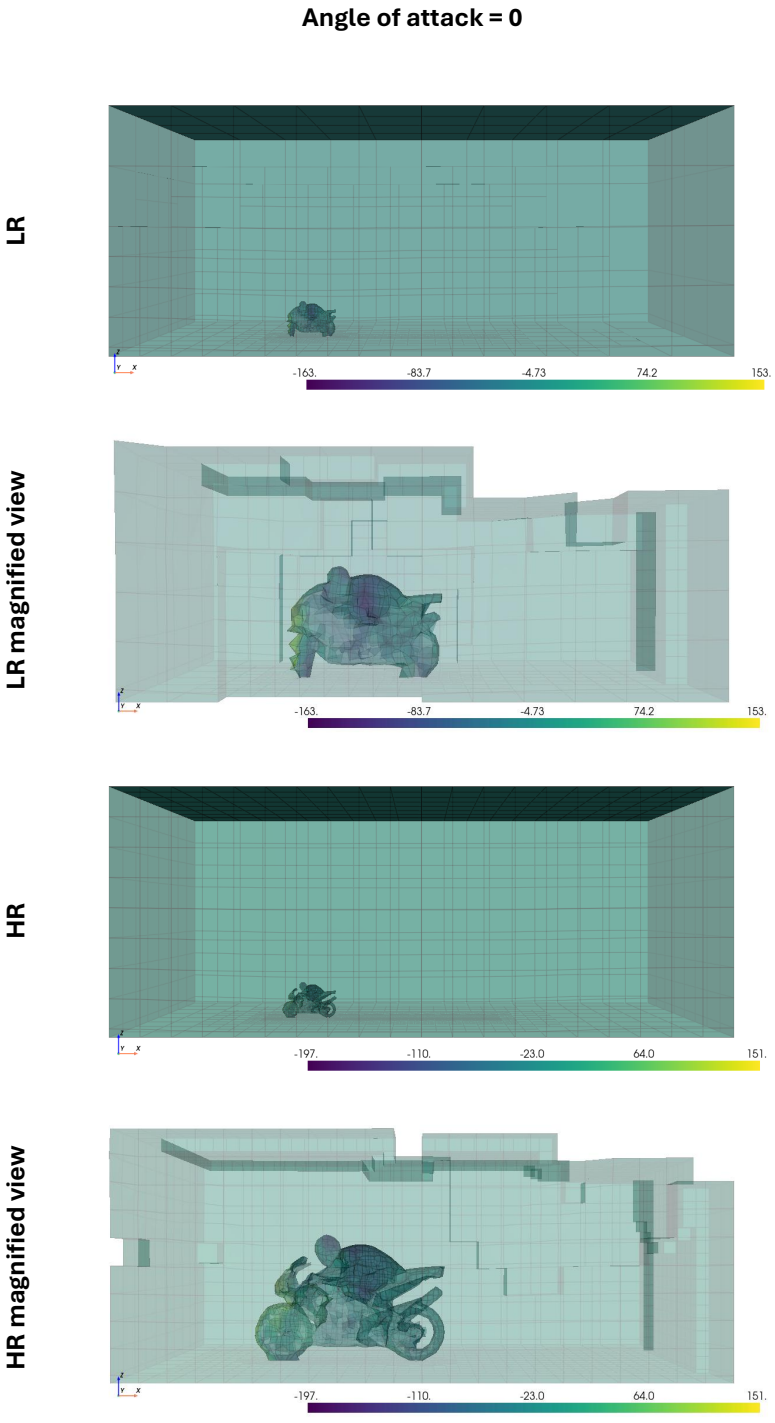

Figure 19: An example of LR and HR data samples corresponding to an angle of attack of 0° from the real-world geometry dataset.

H COMPARISON OF MAGNET, FULLY SUPERVISED BASELINE, AND SUPERMESHNET

Table 9: RMSE comparison of MAgNet (Boussif et al., 2022), the baseline using fully supervised learning, and SuperMeshNet. SuperMeshNet is MGN with inductive biases trained with $N_h = 20$ and $N = 200$, and the baseline is MGN without inductive biases trained with $N_h = N = 200$. MAgNet is a zero-shot super-resolution method trained with $N_h = 0$ (*i.e.*, no HR data) and $N = 200$.

	Dataset 1	Dataset 2	Dataset 3
MAgNet	0.0979± 0.0009	0.1305± 0.0007	0.0754±0.0014
Fully supervised learning	0.0228±0.0015	0.0461±0.0004	0.0243±0.0017
SuperMeshNet	0.0226±0.0007	0.0507± 0.0011	0.0245±0.0005

We empirically show the effect of HR data over the case using zero-shot super-resolution (Boussif et al., 2022). Table 9 demonstrates that MAgNet's RMSE is far apart from that of the fully supervised baseline and SuperMeshNet across all three datasets, while being approximately up to four times higher.

I COMPARISON WITH FULL SUPERVISION

Table 10: The RMSE of SuperMeshNet (with inductive biases) and SuperMeshNet-O (without inductive biases) trained with $N_h = 20$ HR data samples and $N = 200$ LR data samples across six MPNNs and three datasets, in comparison with two fully supervised MPNNs including 1) $N_h = N = 20$ and 2) $N_h = N = 200$. The best performer is highlighted as **bold**.

	Method	MPNN					
	(N_h, N)	GCN	SAGE	GAT	GTR	GIN	MGN
Dataset 1	Fully supervised (20, 20)	0.0874 ± 0.0039	0.0876 ± 0.0015	0.0826 ± 0.0042	0.0758 ± 0.0068	0.0819 ± 0.0047	0.0655 ± 0.0030
	Fully supervised (200, 200)	0.0575 ± 0.0035	0.0544 ± 0.0025	0.0512 ± 0.0016	0.0450 ± 0.0023	0.0381 ± 0.0027	0.0228 ± 0.0015
	SuperMeshNet–O (20, 200)	0.0613 ± 0.0020	0.0589 ± 0.0021	0.0544 ± 0.0008	0.0451 ± 0.0020	0.0404 ± 0.0028	0.0269 ± 0.0019
	SuperMeshNet (20, 200)	**0.0431** ± 0.0009	**0.0450** ± 0.0010	**0.0457** ± 0.0016	**0.0385** ± 0.0029	**0.0277** ± 0.0006	**0.0226** ± 0.0007
Dataset 2	Fully supervised (20, 20)	0.0972 ± 0.0082	0.1025 ± 0.0052	0.0983 ± 0.0026	0.0983 ± 0.0016	0.0775 ± 0.0073	0.0730 ± 0.0075
	Fully supervised (200, 200)	0.0624 ± 0.0022	0.0633 ± 0.0032	0.0637 ± 0.0013	**0.0572** ± 0.0016	**0.0534** ±0.0009	**0.0461** ± 0.0004
	SuperMeshNet–O (20, 200)	0.0636 ± 0.0013	0.0664 ± 0.0032	0.0680 ± 0.0023	0.0631 ± 0.0018	0.0569 ± 0.0023	0.0514 ± 0.0003
	SuperMeshNet (20, 200)	**0.0574** ± 0.0003	**0.0624** ± 0.0006	**0.0634** ± 0.0018	0.0600 ± 0.0012	0.0537 ± 0.0015	0.0507 ± 0.0011
Dataset 3	Fully supervised (20, 20)	0.0587 ± 0.0038	0.0611 ± 0.0043	0.0616 ± 0.0050	0.0513 ± 0.0052	0.0569 ± 0.0016	0.0523 ± 0.0055
	Fully supervised (200, 200)	0.0370 ± 0.0029	0.0340 ± 0.0015	0.0374 ± 0.0012	0.0329 ± 0.0021	0.0317 ± 0.0022	**0.0243** ± 0.0017
	SuperMeshNet–O (20, 200)	0.0380 ± 0.0018	0.0366 ± 0.0021	0.0375 ± 0.0009	0.0363 ± 0.0023	0.0316 ± 0.0010	0.0281 ± 0.0006
	SuperMeshNet (20, 200)	**0.0297** ± 0.0008	**0.0297** ± 0.0014	**0.0310** ± 0.0012	**0.0294** ± 0.0011	**0.0258** ± 0.0008	0.0245 ± 0.0005

Table 11: The RMSE and its standard deviation of SuperMeshNet trained with $N_h = 40$ HR data samples and $N = 200$ LR data samples, in comparison with a fully supervised MPNN trained with $N_h = N = 200$. Experiments are conducted only for cases where using $N_h = 20$ is insufficient to outperform the fully supervised baseline.

Dataset	Method	N_h	N	MPNN	RMSE
2	SuperMeshNet	40	200	GTR	0.0568 ± 0.0015
2	Fully supervised	200	200	GTR	0.0572 ± 0.0016
2	SuperMeshNet	40	200	GIN	0.0501 ± 0.0004
2	Fully supervised	200	200	GIN	0.0534 ± 0.0009
2	SuperMeshNet	40	200	MGN	0.0461 ± 0.0011
2	Fully supervised	200	200	MGN	0.0461 ± 0.0004
3	SuperMeshNet	40	200	MGN	0.0225 ± 0.0003
3	Fully supervised	200	200	MGN	0.0243 ± 0.0017

In Table 10, we present the full version of Table 4 from the main manuscript, now including the standard deviation of RMSE values. In some cases, SuperMeshNet (N_h=20, N=200) yields higher RMSEs than the fully supervised baselines (N_h=N=200). In such cases, Table 11 provides additional results using SuperMeshNet with N_h=40 and $N = 200$, where N_h is set to a slightly larger value than our default setting (*i.e.*, $N_h = 20$) but still significantly smaller than 200, in comparison with the fully supervised (N_h=N=200) baselines. These results demonstrates that using only 20% of HR data samples (*i.e.*, $N_h = 40$) in SuperMeshNet is sufficient to outperform the fully supervised baseline.

J Comparison with a Super-Resolution Competitor

Table 12: The RMSE and its standard deviation of MGN-based SuperMeshNet trained with varying numbers of HR data samples N_h and a fixed N=200 LR data samples in comparison with SRGNN with full supervision (N=N_h=200).

Dataset	Methods	N_h	RMSE
1	SuperMeshNet	5	0.0447 ± 0.0010
		10	0.0280 ± 0.0013
		20	0.0226 ± 0.0007
		40	0.0191 ± 0.0021
	SRGNN (Barwey et al., 2024)	200	0.0247 ± 0.0013
2	SuperMeshNet	5	0.0723 ± 0.0018
		10	0.0645 ± 0.0022
		20	0.0507 ± 0.0011
		40	0.0461 ± 0.0011
	SRGNN (Barwey et al., 2024)	200	0.0487 ± 0.0011
3	SuperMeshNet	5	0.0353 ± 0.0015
		10	0.0294 ± 0.0010
		20	0.0245 ± 0.0005
		40	0.0225 ± 0.0003
	SRGNN (Barwey et al., 2024)	200	0.0254 ± 0.0011

In Table 12, we present the full version of Table 5 from the main manuscript, now including the mean and standard deviation of RMSE values for all datasets. Across all datasets, SuperMeshNet achieves a lower RMSE with a significantly smaller number of HR data samples, N_h, compared to the most recent and relevant benchmark super-resolution method, SRGNN (Barwey et al., 2024).

K Comparison with Benchmark Semi-supervised Regression Methods

Table 13: Comparison with benchmark semi-supervised regression methods in terms of the RMSE and training time (in second). Here, MGN is employed as an MPNN architecture for each method. Training is conducted when $N_h = 20$ and $N = 200$ for each dataset. The best performer is highlighted as **bold**.

Dataset	Methods	RMSE	Training time (s)
1	mean-teacher (Tarvainen and Valpola, 2017)	0.0325 ± 0.0016	694 ± 50
	TNNR (Wetzel et al., 2022)	0.0624 ± 0.0202	477 ± 333
	UCVME (Dai et al., 2023)	0.0293 ± 0.0012	1123 ± 102
	SuperMeshNet-O	0.0269 ± 0.0019	503 ± 64
	SuperMeshNet	**0.0226** ± 0.0007	**421** ± 93
2	mean-teacher (Tarvainen and Valpola, 2017)	0.0499± 0.0007	402 ± 28
	TNNR (Wetzel et al., 2022)	0.0823 ± 0.0055	350 ± 90
	UCVME (Dai et al., 2023)	**0.0484** ± 0.0006	739 ± 59
	SuperMeshNet-O	0.0514 ± 0.0003	306 ± 16
	SuperMeshNet	0.0507 ± 0.0011	**250** ± 9
3	mean-teacher (Tarvainen and Valpola, 2017)	0.0270 ± 0.0003	446 ± 33
	TNNR (Wetzel et al., 2022)	0.0393 ± 0.0027	437 ± 83
	UCVME (Dai et al., 2023)	0.0281± 0.0013	700 ± 89
	SuperMeshNet-O	0.0281 ± 0.0006	345 ± 10
	SuperMeshNet	**0.0245** ± 0.0005	**286** ± 18

Table 14: Comparison with benchmark semi-supervised regression methods in terms of the RMSE and training time (in second). Here, MGN is employed as an MPNN architecture for each method. Training is conducted when $N_h = 40$ and $N = 200$ for each dataset. The best performer is highlighted as **bold**.

Dataset	Methods	RMSE	Training time (s)
2	mean-teacher (Tarvainen and Valpola, 2017)	0.0474 ± 0.0007	427 ± 37
	TNNR (Wetzel et al., 2022)	0.0807 ± 0.0074	409 ± 143
	UCVME (Dai et al., 2023)	0.0474 ± 0.0013	780 ± 89
	SuperMeshNet-O	0.0479 ± 0.0005	348 ± 15
	SuperMeshNet	**0.0461** ± 0.0011	**292** ± 25

Table 13 presents the full version of Table 6 from the main manuscript, now including the mean and standard deviation of RMSE and training time for all datasets under the setting of $N_h = 20$ HR data samples and $N = 200$ LR data samples. SuperMeshNet consistently results in the shortest training time across all datasets, in comparison with benchmark semi-supervised regression methods. It also achieves the lowest RMSE on Datasets 1 and 3. For Dataset 2, while SuperMeshNet yields a slightly higher RMSE, it significantly reduces training time compared to all the benchmarks. Overall, SuperMeshNet is shown to reveal strong potential in terms of both predictive accuracy and training efficiency. In Table 14, we additionally report that, for Dataset 2 with $N_h = 40$ and $N = 200$, SuperMeshNet still exhibits both the lowest RMSE and the shortest training time among all evaluated methods.

L COMPARISON OF PREDICTION DIVERSITY BETWEEN UCVME AND SUPERMESHNET

Table 15: Comparison of prediction diversity between UCVME and SuperMeshNet. The prediction diversity is quantified as the root mean square of the difference between two predictions made by MGN model pairs trained by each method with N_h=20 and N=200 on each dataset. The dropout probability in UCVME is set to 0.1.

Dataset	Methods	Prediction diversity
1	UCVME (Dai et al., 2023)	0.0017 ± 0.0001
	SuperMeshNet	0.0200 ± 0.0014
2	UCVME (Dai et al., 2023)	0.0024 ± 0.0002
	SuperMeshNet	0.0514 ± 0.0011
3	UCVME (Dai et al., 2023)	0.0014 ± 0.0001
	SuperMeshNet	0.0294 ± 0.0004

The results in Table 15 demonstrate that SuperMeshNet exhibits consistently greater prediction diversity across all datasets, compared to the case of UCVME (Dai et al., 2023). The prediction diversity is quantified as the root mean square of the difference between two predictions made by MGN model pairs, each trained with $N_h = 20$ and $N = 200$ on each dataset. This implies that our architectural design in complementary learning apparently more promotes diversity into the learning process than the case of UCVME.

M Ablation Studies on Inductive Biases

Table 16: Ablation studies on inductive biases. The RMSE of SuperMeshNet across six MPNNs under four inductive bias conditions (O: without inductive biases, N: node-level centering, M : message-level centering, and N+M: both node-level and message-level centerings) trained with $N_h = 20$ and $N = 200$ for each dataset is compared. For each MPNN, the lowest RMSE value among the four inductive bias conditions is highlighted as **bold**.

Dataset	MPNN	RMSE			
		O	N	M	N + M
1	GCN	0.0613 ± 0.0020	**0.0431** ± 0.0009	-	-
	SAGE	0.0589 ± 0.0021	0.0493 ± 0.0024	0.0528 ± 0.0018	**0.0450** ± 0.0010
	GAT	0.0544 ± 0.0008	**0.0457** ± 0.0016	-	-
	GTR	0.0451 ± 0.0020	0.0405 ± 0.0025	0.0438 ± 0.0010	**0.0385** ± 0.0029
	GIN	0.0404 ± 0.0028	0.0290 ± 0.0026	0.0281 ± 0.0015	**0.0277** ± 0.0006
	MGN	0.0269 ± 0.0019	0.0237 ± 0.0010	0.0247 ± 0.0014	**0.0226** ± 0.0007
2	GCN	0.0636 ± 0.0013	**0.0574** ± 0.0003	-	-
	SAGE	0.0664 ± 0.0032	**0.0623** ± 0.0005	0.0652 ± 0.0009	0.0624 ± 0.0006
	GAT	0.0680 ± 0.0023	**0.0634** ± 0.0018	-	-
	GTR	0.0631 ± 0.0018	0.0607 ± 0.0009	0.0617 ± 0.0025	**0.0600** ± 0.0012
	GIN	0.0569 ± 0.0023	**0.0523** ± 0.0009	0.0549 ± 0.0008	0.0537 ± 0.0015
	MGN	0.0514 ± 0.0003	**0.0488** ± 0.0013	0.0509 ± 0.0004	0.0507 ± 0.0011
3	GCN	0.0380 ± 0.0018	**0.0297** ± 0.0008	-	-
	SAGE	0.0366 ± 0.0021	0.0309 ± 0.0018	0.0346 ± 0.0018	**0.0297** ± 0.0014
	GAT	0.0375 ± 0.0009	**0.0310** ± 0.0012	-	-
	GTR	0.0363 ± 0.0023	0.0312 ± 0.0008	0.0327 ± 0.0006	**0.0294** ± 0.0011
	GIN	0.0316 ± 0.0010	0.0261 ± 0.0002	0.0268 ± 0.0007	**0.0258** ± 0.0008
	MGN	0.0281 ± 0.0006	**0.0245** ± 0.0003	0.0246 ± 0.0006	**0.0245** ± 0.0005

Table 17: Additional ablation studies on inductive biases. The RMSE of SuperMeshNet across six MPNNs under four inductive bias conditions (O: without inductive biases, N: node-level centering, M : message-level centering, and N+M: both node-level and message-level centerings) trained with $N_h = 80$ and $N = 200$ for Dataset 2 is compared. For each MPNN, the lowest RMSE value among the four inductive bias conditions is highlighted as **bold**.

Dataset	MPNN	RMSE			
		O	N	M	N + M
2	GCN	0.0630 ± 0.0017	**0.0556** ± 0.0006	-	-
	SAGE	0.0626 ± 0.0023	**0.0606** ± 0.0016	0.0628 ± 0.0026	**0.0606** ± 0.0011
	GAT	0.0637 ± 0.0005	**0.0606** ± 0.0016	-	-
	GTR	0.0606 ± 0.0011	0.0560 ± 0.0010	0.0574 ± 0.0012	**0.0554** ± 0.0013
	GIN	0.0528 ± 0.0011	0.0481 ± 0.0007	0.0470 ± 0.0006	**0.0468** ± 0.0006
	MGN	0.0463 ± 0.0005	0.0447 ± 0.0009	0.0448 ± 0.0011	**0.0432** ± 0.0003

Table 16 presents the full version of Table 6 from the main manuscript, now including the mean and standard deviation of RMSE values for all datasets. On Dataset 2, the combination of both inductive biases (N+M) occasionally results in a higher RMSE than using a single inductive bias (N). To further investigate this, Table 17 reports additional results on Dataset 2 using SuperMeshNet with $N_h = 80$ and $N = 200$, where N_h is set to a slightly larger value than our default setting (*i.e.*, $N_h = 20$) yet still significantly smaller than 200. With the larger N_h, the incorporation of both inductive biases consistently yields the lowest RMSE among the four inductive bias settings. These findings reveal the existence of a dataset-dependent threshold for N_h above which leveraging both inductive biases becomes beneficial.

Table 18: Effect of noise on the RMSE of SuperMeshNet (with inductive biases), denoted as N+M, and SuperMeshNet-O (without inductive biases), denoted as O. Noise is added to node embeddings x_i and aggregated messages agg_i in MGN, trained with $N_h = 20$ and $N = 200$ for each dataset.

Dataset	noise condition	O	N+M
1	without noise	0.0263 ± 0.0013	0.0214 ± 0.0015
	noise in mean	0.0300 ± 0.0014	0.0214 ± 0.0015
	noise in deviation	0.0285 ± 0.0012	0.0328 ± 0.0017
2	without noise	0.0508 ± 0.0010	0.0500 ± 0.0011
	noise in mean	0.0518 ± 0.0012	0.0500 ± 0.0011
	noise in deviation	0.0512 ± 0.0010	0.0513 ± 0.0011
3	without noise	0.0280 ± 0.0007	0.0243 ± 0.0005
	noise in mean	0.0292 ± 0.0009	0.0243 ± 0.0005
	noise in deviation	0.0283 ± 0.0007	0.0259 ± 0.0005

Table 18 consistently presents that addition of noise to the mean component of x_i and agg_i significantly impact the RMSE of SuperMeshNet-O (without inductive biases), denoted as O in Table 18. while perturbations in the deviation component has relatively little effect. It indicates a strong reliance on global mean information, which is undesirable for super-resolution tasks. In contrast, SuperMeshNet (with inductive biases), denoted as N+M in Table 18, demonstrates robustness to mean perturbations but exhibits increased sensitivity to deviations. This suggests that the model with inductive biases prioritizes local deviation-related information. These findings justifies our claim that, regardless of datasets, the proposed inductive biases effectively guide the model's attention to local deviations from the global mean, which are more pertinent to high-resolution reconstruction.

N Application to Time-Dependent PDEs and Real-World Geometry

Due to the large grid size of the time-dependent PDE dataset 2, we increased the hidden dimension to 150, whereas all other experiments use a hidden dimension of 30. In addition, because the dataset is defined on a regular grid, we used standard bilinear interpolation instead of kNN interpolation. The visualization below shows that SuperMeshNet successfully predicts the HR counterpart even when the HR data differ substantially from the LR input, whereas conventional fully supervised learning fails despite having access to more HR data.

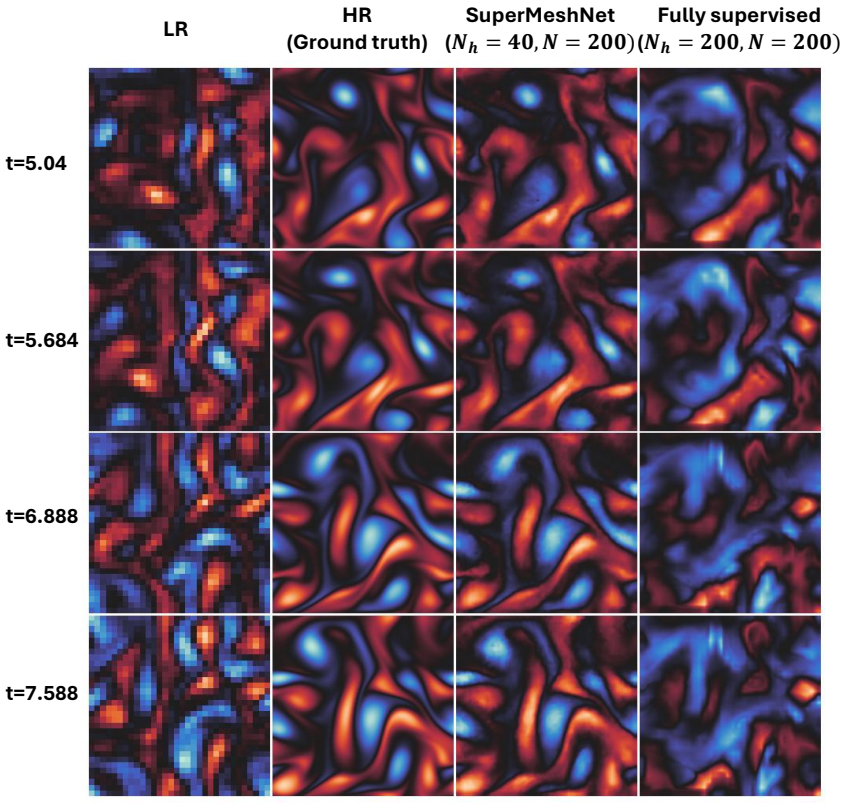

Figure 20: LR input, ground truth HR, and two vorticity predictions produced by SuperMeshNet and full supervision for the time-dependent PDE dataset 2 over multiple timestamps. Here, N_h and N represent the number of HR and LR data samples, respectively. For all cases, MGN is utilized as the underlying MPNN.

Table 19: The RMSE of MGN-based SuperMeshNet trained with $N_h = 40$ HR data samples and $N = 200$ LR data samples for the time-dependent PDE and real-world geometry datasets, in comparison with two fully supervised MGNs including 1) $N_h = N = 40$ and 2) $N_h = N = 200$. The best-performing result in each case is highlighted in **bold**.

Dataset	Methoddy	N_h	N	RMSE
Time-dependent PDE 1	SuperMeshNet	40	200	**0.0310**
	fully supervised	200	200	0.0318
	fully supervised	40	40	0.0599
Time-dependent PDE 2	SuperMeshNet	40	200	**0.0543**
	fully supervised	200	200	0.1613
	fully supervised	40	40	0.1870
Real-World Geometry	SuperMeshNet	40	200	**0.0185**
	fully supervised	200	200	0.0189
	fully supervised	40	40	0.0323

We further quantitatively evaluate the applicability of SuperMeshNet to time-dependent PDE and real-world geometry datasets. The results in Table 19 show that SuperMeshNet, trained with only 20 HR data samples, achieves even a lower error than the case of the fully supervised model trained with 200 HR data samples, demonstrating its effectiveness on to time-dependent PDE and real-world geometry datasets.

O Comparison with Standard Normalization

Table 20: Comparison of our inductive biases in SuperMeshNet with layer normalization (Ba et al., 2016) and batch normalization (Ioffe and Szegedy, 2015). All models are based on the MGN architecture and trained using complementary learning with $N_h = 20$ and $N = 200$. The best-performing result in each case is highlighted in **bold**.

Dataset	Methods	RMSE
1	layer normalization (Ba et al., 2016)	0.0310 ± 0.0014
	batch normalization (Ioffe and Szegedy, 2015)	$\mathbf{0.0165} \pm 0.0009$
	SuperMeshNet	0.0226 ± 0.0007
2	layer normalization (Ba et al., 2016)	0.0587 ± 0.0008
	batch normalization (Ioffe and Szegedy, 2015)	0.0508 ± 0.0507
	SuperMeshNet	$\mathbf{0.0507} \pm 0.0011$
3	layer normalization (Ba et al., 2016)	0.0297 ± 0.0013
	batch normalization (Ioffe and Szegedy, 2015)	0.0250 ± 0.0003
	SuperMeshNet	$\mathbf{0.0245} \pm 0.0005$

Similarly as in our inductive biases, existing normalization techniques such as layer normalization (Ba et al., 2016) and batch normalization (Ioffe and Szegedy, 2015) also involve mean subtraction. In the context of MPNNs, layer normalization computes the mean across features within each node embedding, whereas batch normalization computes the mean across node embeddings, which is also leveraged in our inductive biases. Thus, rather than layer normalization (Ba et al., 2016), our inductive biases are closer to batch normalization (Ioffe and Szegedy, 2015). Standard normalization techniques typically include additional operations such as scaling by the standard deviation and the use of learnable shift and scale parameters. The results in Table 20 demonstrate that, without these additional components, SuperMeshNet consistently outperforms layer normalization and, in some cases, even surpasses batch normalization. This suggests that centering alone, namely mean subtraction, acts as a crucial inductive bias for the super-resolution task.

Although our inductive biases are rather simple and closely related to conventional normalization methods, they offer several key insights. First, computing the mean *across* nodes (as in batch normalization and SuperMeshNet) is proven to be more effective than computing the mean *within* each node (as in layer normalization), likely because super-resolution benefits more from local deviations from the global mean. Second, the additional components of standard normalization, such as division by the standard deviation and learnable affine parameters, do not necessarily yield performance gains. This underscores that centering itself is the most essential component for super-resolution tasks.

P TIME COMPLEXITY

SuperMeshNet alleviates the reliance on expensive HR data samples but incurs a longer training time compared to a fully supervised baseline. Figures 21–23 display how the training time increase and data generation time decrease—resulting from the introduction of SuperMeshNet ($N_h = 20$, $N = 200$)—scale with mesh size, relative to fully supervised learning ($N_h = N = 200$). To estimate the data generation time decrease, we measure the time required to generate 180 HR data samples by solving the PDE using a direct solver on an Intel(R) Core(TM) i7-9700K CPU @ 3.60GHz. For the training time increase, we compute the difference between the training times of MGN under fully supervised learning and SuperMeshNet, using an Intel (R) Core (TM) i9-10920X CPUs@3.50 GHz and an NVIDIA RTX A6000 GPU. The slopes in each figure are computed using the least square method. Comparison of two slopes, representing the sensitivity of training time increase and data generation time decrease to mesh size, respectively, reveals that data generation time grows more rapidly as mesh size decreases. This trend is particularly important in our target regime, where fine meshes lead to significant computational cost. We expect that for sufficiently fine meshes, the savings in data generation time will outweigh the additional training time, resulting in an overall reduction in computational costs.

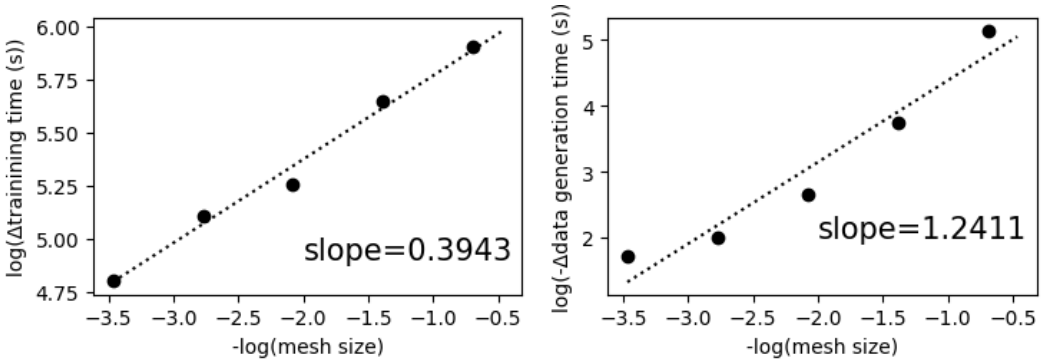

Figure 21: Training time increase (left) and data generation time decrease (right), resulting from the introduction of SuperMeshNet ($N_h = 20$, $N = 200$), relative to fully supervised learning ($N_h = N = 200$) on Dataset 1 and its mesh-size variants. All experiments use MGN as the underlying MPNN architecture.

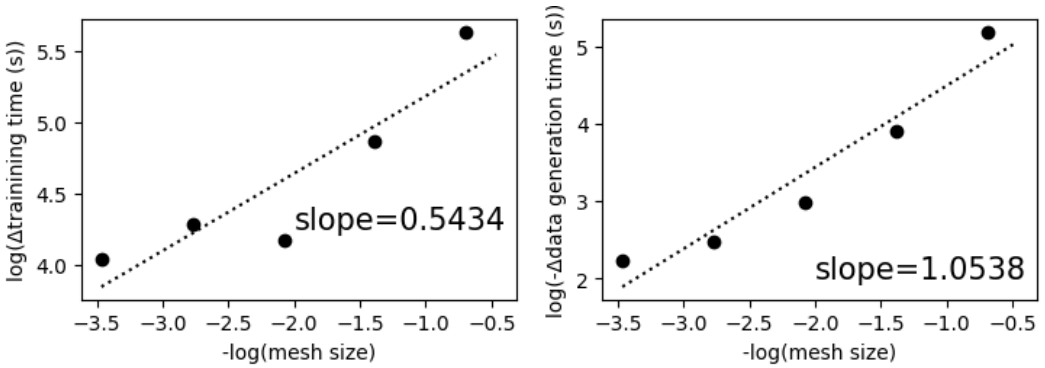

Figure 22: Training time increase (left) and data generation time decrease (right), resulting from the introduction of SuperMeshNet ($N_h = 20$, $N = 200$), relative to fully supervised learning ($N_h = N = 200$) on Dataset 2 and its mesh-size variants. All experiments use MGN as the underlying MPNN architecture.

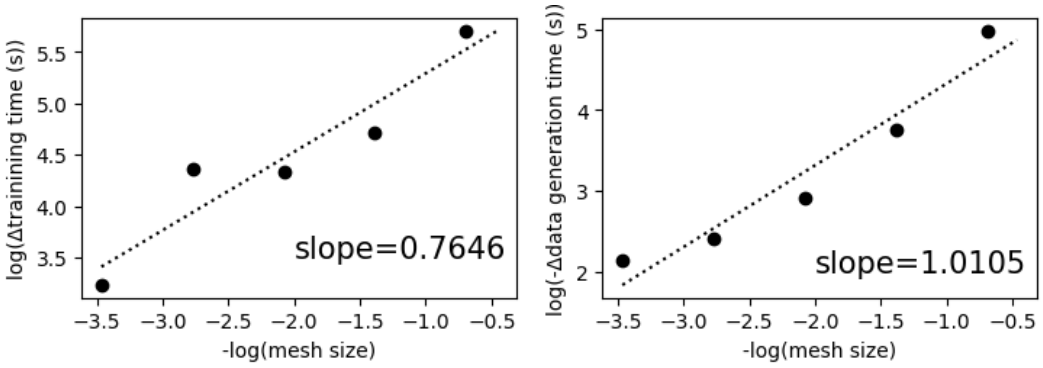

Figure 23: Training time increase (left) and data generation time decrease (right), resulting from the introduction of SuperMeshNet ($N_h = 20$, $N = 200$), relative to fully supervised learning ($N_h = N = 200$) on Dataset 3 and its mesh-size variants. All experiments use MGN as the underlying MPNN architecture.

Q SCALABILITY

Table 21: Training time (in second) and RMSE of MGN-based SuperMeshNet trained with 40 HR and 200 LR data samples from Dataset 1 as a function of mesh size while fixing the magnification ratio.

Smallest mesh size	Training time (s)	RMSE
0.016	383.28	0.0112
0.008	443.34	0.0158
0.004	587.50	0.0194
0.002	710.96	0.0121

Table 22: RMSE of MGN-based SuperMeshNet trained with 40 HR and 200 LR data samples from Dataset 1 as a function of magnification ratio (HR mesh size) while fixing the LR mesh size.

Magnification ratio	RMSE
2×2	0.0099
4×4	0.0112
8×8	0.0689
16×16	0.0796

In Table 21, we evaluate the scalability of our framework SuperMeshNet by measuring the training time and RMSE as a function of mesh size (while fixing the magnification ratio). These experiments have been conducted using MGN, trained with 40 HR and 200 LR data samples from Dataset 1. According to the results, the training time increases moderately as the mesh becomes finer. The RMSE tends to increase and then decrease again as the mesh becomes extremely fine. This is because we fixed the magnification ratio (the ratio between the LR mesh size and the HR mesh size) while varying both the LR mesh size and the HR mesh size. When the mesh becomes finer, the HR data samples contain more detailed features, making the LR-to-HR transformation more challenging. However, once the mesh resolution exceeds the characteristic length scale of the domain geometry, no additional details can be represented in the HR mesh. At the same time, the LR data samples already capture most of the meaningful features, and the LR-to-HR transformation becomes easier again. For comparison, we have provided Table 22, presenting experimental results where the LR mesh size is fixed while the magnification ratio (the HR mesh size) varies. In this case, the RMSE tends to monotonically increases as the HR mesh becomes finer since the LR mesh size is fixed.

R TRAINING STABILITY

To assess training stability, we plot the loss curves over epochs for five different random seeds. As shown in Figure 24, the loss consistently decreases in all runs without exhibiting divergence.

Although a pseudo-label generated by one model may contain potential errors, such errors do not lead to catastrophic mutual reinforcement. This is because every loss computation judiciously incorporates both pseudo-label-based mutual supervision and ground truth–based supervision. The presence of true labels at every iteration constrains error propagation, preventing any amplification caused by imperfect pseudo-labels. Consequently, the complementary learning in our SuperMeshNet framework maintains stable optimization dynamics.

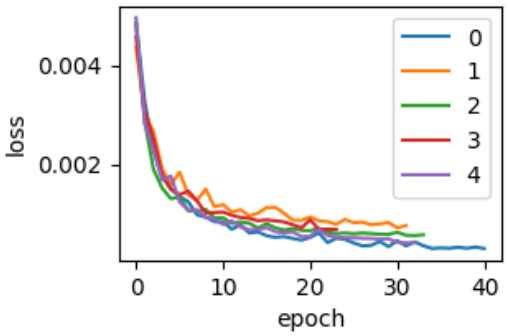

Figure 24: Loss curves for five different random seeds, each shown in a different color. For all cases, the MGN-based SuperMeshNet is trained using $N_h = 20$ and $N = 200$ from Dataset 1.

S SINGULAR POINT ANALYSIS

To evaluate the effect of geometric singularity on super-resolution performance, we additionally generate a dataset that explicitly includes a singular point. Representative HR and LR samples from this dataset are shown in Figure 25.

The computational domain is defined by six verticies: (0, -0.25), (-0.25, -0.25), (-0.25, 0.25), (0.25, 0.25), (0.25,0), and (x_0, y_0), where (x_0, y_0) denotes the location of the singular point. Both x_0 and y_0 are independently sampled from a uniform distribution on [-0.125, 0.125]. The default mesh sizes for LR and HR meshes are set to 0.04 and 0.01, respectively. However, within the vicinity of the singular point, the mesh is refined by a factor of four for each resolution.

On this domain, the Poisson equation below is solved :

$$\nabla^2 u = ((x - x_0)^2 + (y - y_0)^2)^{-1}, \tag{28}$$

where u denotes the electrical potential. The boundary condition is fixed at 0V. After solving the equation, the magnitude of the electric field is calculated at each node of the mesh. Examples of LR and HR fields are visualized in Figure 25.

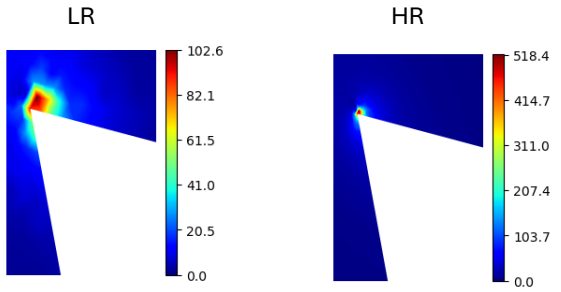

Figure 25: Examples of LR and HR data samples from the dataset including a singular point.

We compare the HR fields predicted by SuperMeshNet with those obtained via pure kNN interpolation without neural network-based prediction in Figure 26. Both methods exhibit relatively high errors at the singular point. Nevertheless, the MPNN in SuperMeshNet noticeably mitigates the severity of these errors: both the magnitude and spatial extent of the high error region are reduced compared to the case of pure interpolation. Specifically, the maximum error near the singularity is 4.18 for pure kNN interpolation, whereas SuperMeshNet reduces this peak error to 1.57.

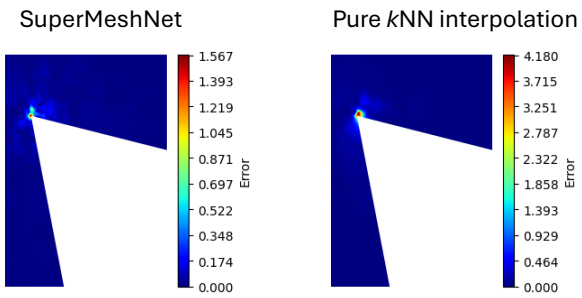

Figure 26: Comparison of predictions by SuperMeshNet and pure kNN interpolation.

T Analysis on Inductive Biases

We explore why centering (*i.e.*, subtracting the mean over nodes at each layer) is beneficial for tasks in which the global mean is relatively unimportant (*e.g.*, super-resolution), but less effective for tasks where the mean level plays a central role (*e.g.*, norm prediction).To this end, we carry out a comparative study on Dataset 1 across two representative tasks:

- Super-resolution: The goal is to predict HR fields from LR fields;
- Norm prediction: The goal is to predict the (normalized) norm of LR input fields, which is constant across nodes.

T.1 Effect of Global Mean

T.1.1 Super-resolution

As summarized in Table 23, subtracting the mean from LR data leads to only a minor change in RMSE, whereas providing only the global mean significantly degrades RMSE performance. This implies that the global mean contributes relatively little to prediction accuracy. Super-resolution primarily requires learning high-frequency discrepancies between LR and HR fields (Guo et al., 2025). Furthermore, our architecture (Figures 12 and 13) produces HR predictions by refining a kNN-interpolated coarse estimate, which may further reduce reliance on modeling the global mean.

Table 23: RMSE comparison of MGN-based SuperMeshNet trained for super-resolution task on Dataset 1, using the following three cases as input: the LR data sample, its deviation from the mean, and the mean.

input	RMSE
LR data	0.0228
Deviation of LR data	0.0235
Mean of LR data	0.0617

T.1.2 Norm prediction

In contrast, for norm prediction, the global mean appears to contain essential information. As summarized in Table 24, when only the mean is provided, performance gets improved, whereas providing only the deviation significantly degrades performance.

Table 24: RMSE comparison of MGN-based SuperMeshNet trained for norm prediction task on Dataset 1, using the following three cases as input: the LR data sample, its deviation from the mean, and the mean.

input	RMSE
LR data	0.00269
Deviation of LR data	0.00378
Mean of LR data	0.00149

T.1.3 Further Clarification

We further clarify that whether mean information matters for a task is determined by how strongly the ground-truth output is influenced by the mean of the input. In Figure 27, we compare the relationship between the input mean and the ground truth output for two representative tasks: super-resolution and norm prediction. As illustrated in Figure 27, the norm prediction task exhibits a clear dependency between the input mean and the target output (the norm), indicating that the mean indeed carries essential information; in contrast, the super-resolution task reveals no such dependency, demonstrating that global mean information is relatively unimportant for this task.

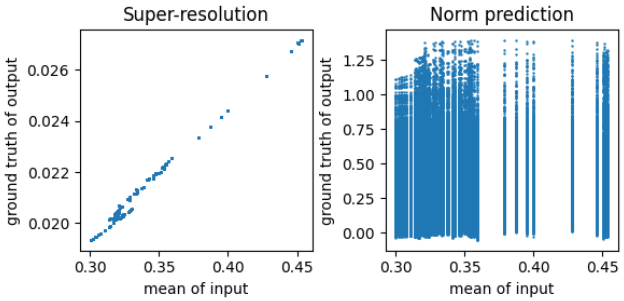

Figure 27: Relationship between the mean of the input and the ground truth output for super-resolution and norm prediction tasks.

Below, we would like to additionally provide concrete examples of tasks where the mean is typically less important:

1) Scientific tasks

- Vorticity prediction: the vorticity field depends on the local rotational structure, not the mean velocity.
- Force prediction: forces depend on relative distances, not absolute positions.

2) General graph learning tasks

- Node classification: predictions rely mainly on relational structures, not absolute feature scales.
- Link prediction: predictions depend on similarity between features rather than the absolute feature scale.

It is worth noting that these inductive biases may also be useful for solving the tasks mentioned above.

T.2 Loss Landscape

As reported in prior work on batch normalization (Santurkar et al., 2018), a smoother loss landscape stabilizes and improves optimization.

To measure the loss landscape, following the prior work in (Santurkar et al., 2018), we perturb the model parameters by taking a step along the gradient direction and evaluate the loss at the perturbed point. The perturbation magnitude is set to a fixed multiple of the learning rate (four times the learning rate in our experiments).

As illustrated in Figures 28 and 29, centering (*i.e.*, the case with inductive biases) yields a smoother loss landscape over training iterations in super-resolution, while it leads to a noticeably rougher loss landscape in norm prediction.

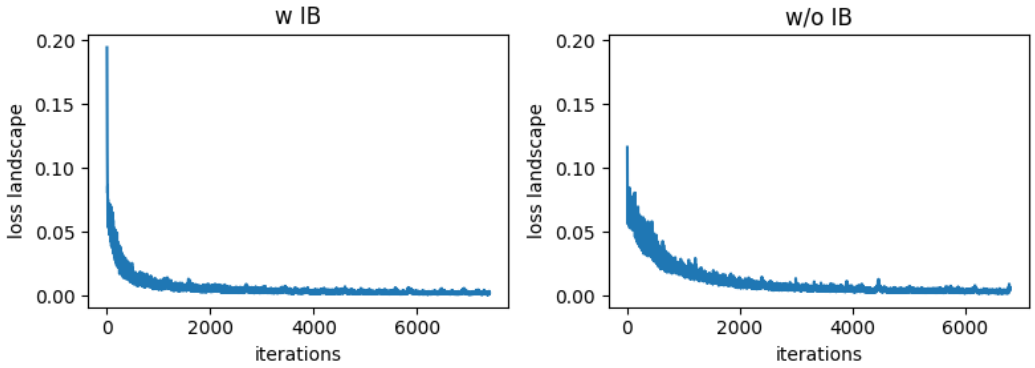

Figure 28: Effect of centering on the loss landscape for the super-resolution task when SuperMeshNet with and without inductive biases (corresponding to w IB and w/o IB, respectively) is used. Here, MGN is employed as an MPNN for each method and is trained when N_h=20 and N=200 for Dataset 1.

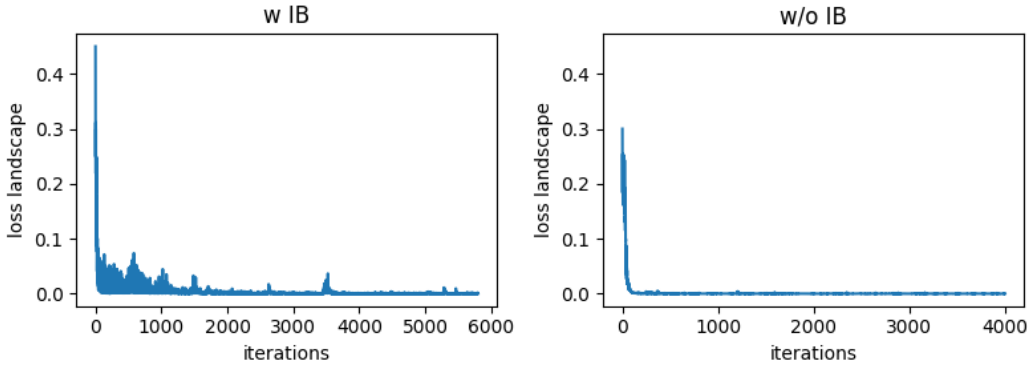

Figure 29: Effect of centering on the loss landscape for the norm prediction task when SuperMeshNet with and without inductive biases (corresponding to w IB and w/o IB, respectively) is used. Here, MGN is employed as an MPNN for each method and is trained when N_h=20 and N=200 for Dataset 1.

T.3 RMSE IMPROVEMENT

As summarized in Table 25, centering empirically improves RMSE performance in super-resolution, whereas it significantly reduces accuracy for norm prediction.

Table 25: Effect of inductive biases on the performance of super-resolution and norm prediction tasks. All experiments are conducted using MGN-based SuperMeshNet and Dataset 1.

Task	Inductive biases	RMSE
Super-resolution	O	0.0226
Super-resolution	X	0.0269
Norm prediction	O	0.00924
Norm prediction	X	0.00647

T.4 OVERALL EFFECT

Our experiments indicate a task-dependent behavior of centering:

- For tasks where the global mean information is less important (*e.g.*, super-resolution), centering tends to smooth the optimization landscape and can improve RMSE performance.
- For tasks where the global mean is important (*e.g.*, norm prediction), centering removes useful information and may hinder optimization, leading to degraded performance.

U Analysis on HR Data Sampling Strategies

We empirically analyze how HR data sampling strategies affect performance on RMSE. In the experiments, we only use Dataset 1 for simplicity.

U.1 Impact of Different 20 HR Subsets

Table 26 shows that using different subsets of 20 HR samples (*i.e.*, $N_h = 20$) introduces a moderate variance: the RMSE standard deviation (STD) is about 6.7% of the mean RMSE and is roughly twice as large as in the case of using the same 20 HR samples. In the same-samples setting, all five training runs use the same 20 HR samples drawn once from a uniform distribution, whereas, in the different-samples setting, each run uses a distinct set of 20 HR samples independently drawn from the same distribution. For both settings, we measure the RMSE from each training run and report the mean and STD across the five runs.

These results indicate that, under a fixed sampling strategy, the variability induced by using different subsets of 20 HR samples remains moderate.

Table 26: Effect of different HR subsets on the mean and STD of RMSE for MGN-based SuperMeshNet trained with $N_h = 20$ and $N = 200$ on Dataset 1.

Strategy	RMSE mean	RMSE STD
same random samples	0.0226	0.0007
different random samples	0.0208	0.0014

U.2 Impact of Distribution Mismatch

We further examine how the distributional mismatch between two cases using all 200 LR samples and the 20 LR samples corresponding to selected HR samples affects performance. The mismatch is quantified using maximum mean discrepancy (MMD). As summarized in Table 27, MMD strongly correlates with RMSE: subsets that better match the overall LR distribution yield lower RMSE.

To study this, we compare three sampling strategies. MMD-minimizing greedy sampling (kernel herding (Chen et al., 2010)) begins by randomly selecting one HR sample from the pool of 200 candidates and then iteratively chooses the next HR sample whose corresponding LR counterpart most reduces the MMD between the selected subset and the full LR distribution. This procedure ensures that the selected HR samples closely represent the overall dataset. In contrast, MMD-maximizing greedy sampling intentionally selects the next point that most increases the MMD, pushing the subset away from the full distribution. As demonstrated below, the selected sampling strategy has a significant impact on the RMSE of SuperMeshNet. Designing more effective sampling strategies could be an interesting direction for future work.

Table 27: Impact of HR data selection strategies on RMSE for MGN-based SuperMeshNet trained with $N_h = 20$ and $N = 200$ on Dataset 1.

Strategy	RMSE	MMD
same random samples	0.0226	0.147
MMD-minimizing greedy sampling	0.0185	0.058
MMD-maximizing greedy sampling	0.0259	0.330

V RELATIVE COMPUTATIONAL COST OF kNN INTERPOLATION

We acknowledge that kNN interpolation introduces additional overhead, particularly for very small mesh sizes. As expected, the relative interpolation cost indeed grows as the task difficulty increases, which is investigated below.

Table 28: Relative overhead of kNN interpolation across different mesh sizes. All experiments are conducted using the MGN-based SuperMeshNet trained with $N_h = 20$ and $N = 200$ on Dataset 1.

Mesh size	(kNN interpolation time) / (total inference time)
8	17.0%
4	18.6%
2	29.4%
1	71.2%

However, even after including kNN interpolation, the total inference cost remains far lower than the cost of running the HR simulation for all mesh sizes, as indicated in Figure 7. Thus, while kNN interpolation may act as a localized bottleneck (particularly for very small mesh sizes), it does not diminish the substantial overall speed-up achieved by SuperMeshNet. We plan to explore lighter-weight or learned interpolation methods as part of future work.

W Ablation Study on Core Components

To disentangle the individual contributions of complementary learning (CL) and inductive biases (IB), we evaluate the performance of all four combinations (no CL + no IB (20, 200), CL only, IB only, and CL + IB) under the same data configuration (*i.e.*., $N_h = 20$, $N = 200$). For comparison, the fully supervised baseline is assumed to use all 200 HR–LR pairs (*i.e.*, $N_h = N = 200$).

Table 29: Ablation study on complementary learning (CL) and inductive biases (IB). All experiments use the MGN-based SuperMeshNet and Dataset 1.

Methodology	IB	CL	N_h	N	RMSE
no CL + no IB (20,20)	X	X	20	20	0.0655
no CL + no IB (20, 200)	X	X	20	200	0.0454
CL only	X	O	20	200	0.0269
IB only	O	X	20	200	0.0371
CL + IB (SuperMeshNet)	O	O	20	200	0.0226
baseline: fully supervised	X	X	200	200	0.0228

The results demonstrate that:

- both CL and IB indeed contribute to improved performance relative to the case of no CL + no IB (20, 200).
- Both components are necessary for SuperMeshNet to outperform the fully supervised baseline trained with 180 additional HR samples.

In particular, in Regard to the settings where CL is not used ((i) no CL + no IB (20, 200) and (ii) IB only), we pretrain models to reconstruct LR data from noised LR data (via denoising reconstruction), and fine-tune the models on the 20 paired LR–HR samples. This approach provides a stronger control than the case of no CL + no IB (20, 20) involving training only on the 20 paired samples, as reflected in the lower RMSE.

X APPLICATION TO A CNN-BASED ARCHITECTURE

To test whether our complementary learning can be generalized beyond MPNN architectures, we additionally apply the complementary learning to a CNN-based architecture. Specifically, we replace both the LR and HR MPNN processors with ResNet blocks. For regular-grid data, we also replace kNN interpolation with bilinear interpolation for computational efficiency. On the regular-grid setting for the time-dependent PDE dataset 2, complementary learning not only improves HR-data efficiency but also even outperforms the fully supervised baseline.

Table 30: Application of complementary learning to a CNN-based architecture.

Methodology	N_h	N	RMSE
complementary learning + ResNet	40	200	0.0339
full supervision + ResNet	200	200	0.0417

This result indicates that the complementary learning is architecture-agnostic and can be flexibly extended beyond MPNNs.

Y COMPARISON WITH A PINN

As summarized in the table 31, we compare a physics-informed neural network (PINN) (Raissi et al., 2019) and SuperMeshNet in terms of computational time and RMSE performance.

First, we describe the experimental setting. we conduct evaluations on the Poisson equation:

$$\nabla^2 u = \sqrt{x^2 + y^2},$$

where u is the solution and (x, y) denotes the spatial coordinate. The computational domain is an L-shaped region with six vertices:

$$(0, -0.25),\ (-0.25, -0.25),\ (-0.25, 0.25),\ (0.25, 0.25),\ (0.25, 0),\ (0, 0),$$

and homogeneous Dirichlet boundary conditions ($u = 0$) are applied on all boundaries.

For training SuperMeshNet, we construct a dataset consisting of 20 HR and 200 LR samples. The domain is the same L-shape, except that the final vertex is replaced by a variable point (x_0, y_0), where both x_0 and y_0 are independently sampled from a uniform distribution on $[-0.125, 0.125]$. The default sizes for LR and HR meshes are set to 0.04 and 0.01, respectively; however, within the vicinity of the singular point, the mesh is refined by a factor of four for each resolution. On this domain, to generate LR and HR data, we solve the following Poisson equation:

$$\nabla^2 u = \sqrt{(x - x_0)^2 + (y - y_0)^2}.$$

Next, we show experimental results, which exhibit a clear trade-off: SuperMeshNet achieves substantially lower RMSE, while PINN attains much lower total computation time per instance.

However, these results are not conclusive. Once SuperMeshNet is trained, it can be applied to any new instance without retraining, whereas a PINN must be retrained for every new instance. Thus, when a large number of parameterized PDE instances must be solved, the initial training cost of SuperMeshNet becomes amortized. Likewise, the RMSE of a PINN can potentially be improved by recent advanced optimization techniques. A more extensive comparison is left for future work.

Table 31: Comparison between PINN and MGN-based SuperMeshNet.

Methodology	Data preparation time + training time (s)	Inference time (s)	RMSE
PINN	5.8	0.0003	8.0×10^{-4}
SuperMeshNet	16467.3	0.0061	1.0×10^{-4}

Z USE OF LARGE LANGUAGE MODELS

The writing of this paper was refined with the assistance of a Large Language Model (LLM).