Learning to Generate Structured Meshes with In-Context: Toward Generalization in Mesh Generation

Jing Xiao[1,2,3], **Xinhai Chen**[1,2,3*], **Jiaming Peng**[1,2,3], **Jie Liu**[1,2,3]

[1]Laboratory of Digitizing Software for Frontier Equipment, National University of Defense Technology
[2]National Key Laboratory of Parallel and Distributed Computing, National University of Defense Technology
[3]College of Computer Science and Technology, National University of Defense Technology, Changsha 410073, China
{xiaoj10, chenxinhai16, pengjiaming, liujie}@nudt.edu.cn

Abstract

Structured mesh generation serves as a crucial preprocessing step in numerical simulations and can be formulated as a mapping problem from geometry to structured mesh. Existing approaches typically establish an isolated mapping for each geometry. This geometry-specific paradigm fails to capture and leverage commonalities across geometries, inevitably requiring recomputation or costly retraining for new geometries. To overcome this limitation, we propose ICL-Mesh, a meta-learning framework based on in-context learning (ICL) for structured mesh generation. It treats learning one mapping as one task and trains a single neural network to extract commonalities across tasks and learn from in-context examples within each task, enabling rapid generalization to unseen tasks without parameter updates. Experimental results demonstrate that ICL-Mesh effectively generalizes to diverse geometries with only a few context examples, and even without examples. It also exhibits robustness to in-context example order sensitivity and can be extended to various mesh generation scenarios, including mesh refinement and coarsening.

Introduction

Mesh generation is a fundamental prerequisite in various scientific and engineering disciplines, such as computational fluid dynamics (Wang et al. 2025), electromagnetics (Ulaby 2025), and aerospace engineering (Wittenborg et al. 2025). This process involves discretizing continuous geometric domains into meshes composed of finite computational cells (Wang et al. 2024), which constitute the foundation of numerical methods (Adler et al. 2025), greatly affecting the accuracy and efficiency of numerical simulations.

Among various mesh types, structured meshes are widely used in high-order numerical methods for accurate simulations (Lei et al. 2023). The process of structured mesh generation can be viewed as learning a mapping f from geometry to structured mesh, as depicted in Figure 1 A. With the advancement of deep learning, neural network-based methods have attracted significantly attention for their effectiveness in learning such mappings (Chen et al. 2024). These methods leverage the universal approximation capability of neural networks (Gurney 2018) to approximate the mapping

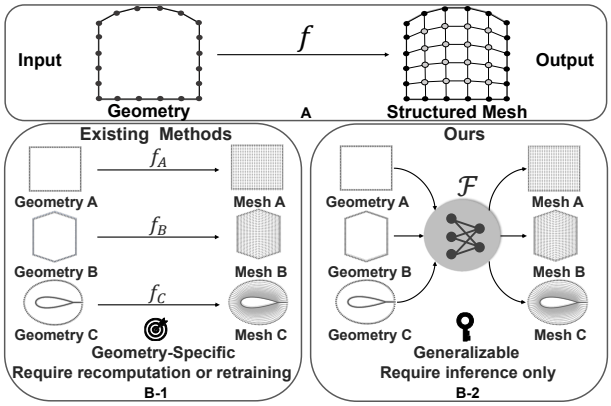

Figure 1: **A**: Illustration of the structured mesh generation. **B**: Comparison between existing methods (**B-1**) and our proposed method (**B-2**).

f, thereby avoiding interpolation errors and iterative solving commonly encountered in traditional methods (Shragge 2006; Allen 2008). Consequently, they remarkably enhance meshing efficiency and mesh quality, showcasing substantial potential for advancing mesh generation.

One representative class of methods formulates structured mesh generation as solving a partial differential equation (PDE) system (Shragge 2006). In this formulation, the governing equations and geometry conditions are embedded into the loss function to guide the meshing process (Chen et al. 2022a,b; Peng et al. 2024; Peng, Chen, and Liu 2025; Zhang et al. 2025). However, since the training process is inherently tied to the specific geometry conditions incorporated in the loss function, any variation in the geometry will necessitate retraining to accommodate the new configuration. To overcome the limited generalization to unseen geometries, data-driven approaches have been proposed to learn a generalizable mapping f from input-output pairs (Xiao et al. 2025). This enables the model to accommodate small variations in the geometry; however, more substantial or topologically changes still necessitate reconstructing the input-output pairs and retraining the neural network.

Overall, existing approaches significantly improve meshing efficiency while maintaining mesh quality, but they fundamentally lack generalization capabilities. This limitation

*Corresponding Author

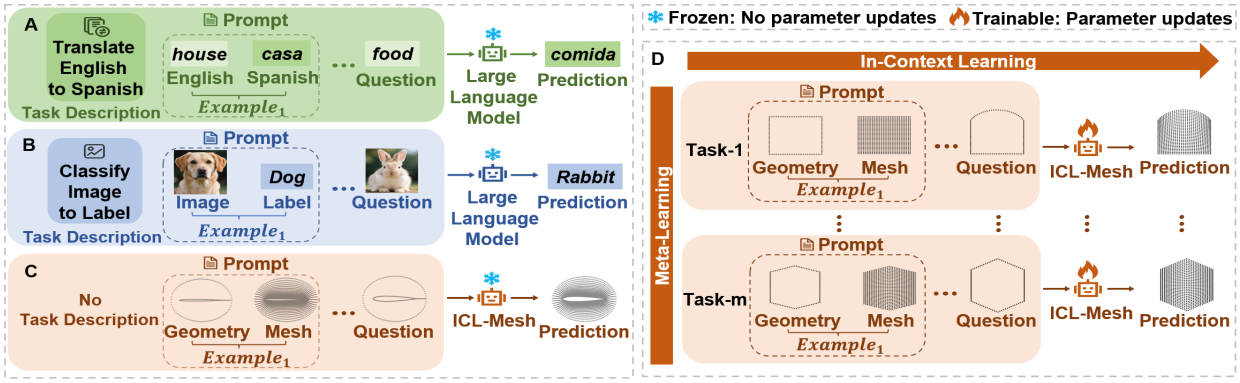

Figure 2: Illustration of in-context learning across domains: **(A)** natural language processing, **(B)** computer vision, and **(C)** structured mesh generation. **(D)** Illustration of the training process for the ICL-Mesh shown in **(C)**.

can be attributed to the fact that these methods typically establish an isolated mapping for each geometry, as illustrated in Figure 1B-1. As a result, this geometry-specific paradigm fails to capture and leverage commonalities across geometries, inevitably requiring recomputation or costly retraining when applied to new geometries.

To address this limitation, we propose ICL-Mesh, a meta-learning framework for structured mesh generation, where each task corresponds to learning a geometry-specific mapping, aiming to generalize across diverse geometries (Figure 1B-2). Moreover, to further enhance generalization, we introduce in-context learning—a paradigm that has shown strong adaptability to unseen tasks with few examples, as demonstrated in large language models such as GPT (Liu et al. 2021) (Figures 2A and B). Specifically, ICL-Mesh treats learning one mapping as one task and trains a single neural network to extract commonalities across tasks and learn from the in-context examples within each task, enabling rapid generalization to unseen tasks without any parameter updates. During training, ICL-Mesh is conditioned on a few geometry-mesh pairs (prompts) for each task and updates its parameters based on the predicted outputs (Figure 2D). At inference, it predicts directly from the given geometry-mesh pairs without retraining (Figure 2C).

Our main contributions are summarized as follows:

- We propose ICL-Mesh, a novel meta-learning framework for structured mesh generation that learns a universal mapping across diverse geometries, overcoming the limitation of existing methods which typically learn geometry-specific mappings.
- We are the first to introduce the in-context learning paradigm into mesh generation, enabling generalization to unseen geometries by leveraging only a few in-context examples during inference, thereby avoiding recomputation and costly retraining.
- We conduct extensive experiments showing that ICL-Mesh generalizes well to unseen geometries with only a few examples, and even without any example. It also exhibits robustness to in-context example order sensitivity and can be extended to various mesh generation scenarios, such as mesh refinement and coarsening.

Related Work

Structured Mesh Generation

Structured mesh generation has been widely studied. One representative method, MGNet (Chen et al. 2022a), employs physics-informed neural networks (Raissi, Perdikaris, and Karniadakis 2019) to learn the mapping f, improving mesh quality by optimizing loss functions and utilizing coarse meshes and auxiliary data (Chen et al. 2022b, 2023). Additionally, this approach has also been extended to 3D (Peng, Chen, and Liu 2025), but slow training convergence remains an issue. To address this, recent works (Peng et al. 2024; Zhang et al. 2025) replace MLPs with Kolmogorov-Arnold networks (KAN) (Liu et al. 2024) to accelerate training. Despite these advancements, such methods remain geometry-specific, lacking generalization capabilities. In response, MeshONet (Xiao et al. 2025) constructs a dataset comprising multiple geometry-mesh pairs and learns a generalized mapping through training on this dataset. Nevertheless, substantial changes in geometry still necessitate reconstructing input-output pairs and retraining the neural network. Therefore, our work is dedicated to addressing the challenge of generalization across geometries.

In-Context Learning

In-context learning was first demonstrated by GPT-3 (Brown et al. 2020), showing that large language models can perform new tasks using task descriptions and a few examples, without updating model parameters (Dong et al. 2022). Since GPT-3 is based on the Transformer, subsequent studies investigated the intrinsic capabilities of Transformer in supporting ICL, which can be theoretically explained via Bayesian inference (Dai et al. 2022). Moreover, multi-task pretraining and meta-learning have been employed to further enhance ICL's ability to generalize across tasks (Min et al. 2021; Chen et al. 2021; Gharoun et al. 2024). This paradigm of "learning to learn"(Hospedales et al. 2021) was recently applied to scientific computing (Yin et al. 2021; Kirchmeyer et al. 2022; Zhang et al. 2023; Yang et al. 2023; Yang, Liu, and Osher 2025). However, this paradigm has not yet been introduced to structured mesh generation.

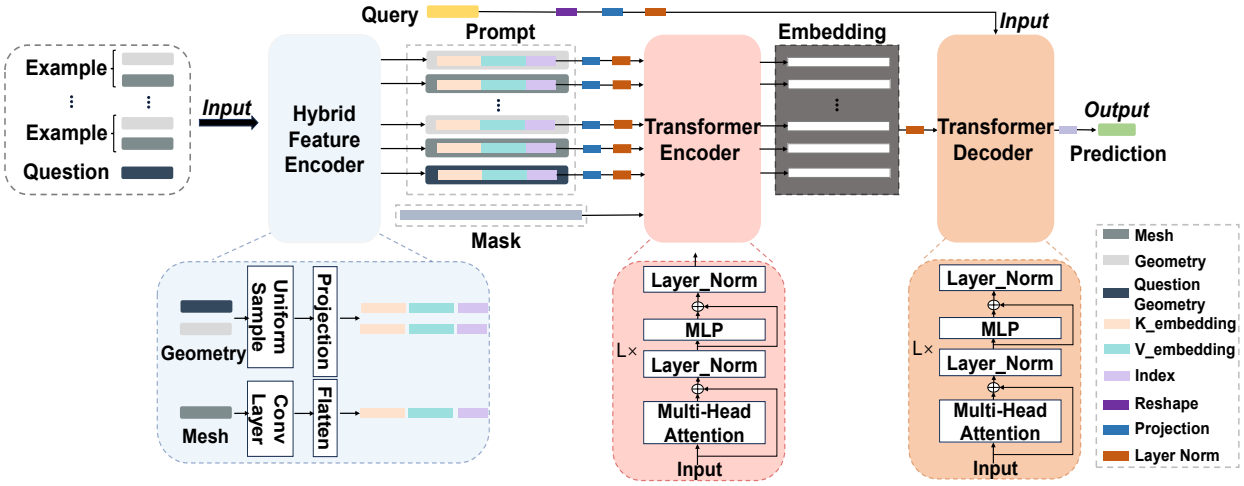

Figure 3: The architecture of ICL-Mesh.

Methodology

Problem Formulation

Structured mesh generation can be formulated as learning a mapping $f : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X}$ represents a geometry and $\mathcal{Y}$ its corresponding mesh. We propose a meta-learning framework based on in-context learning, where each task corresponding to learning a mapping f. The goal of this framework is to learn a universal mapping $\mathcal{F}$ that leverages in-context examples for rapid generalization to unseen geometries without the need for parameter updates.

To further describe our method, we introduce some notations. Specifically, for each task i, we define a set of geometry–mesh pairs (x_j^i, y_j^i), where j indexes the context examples within task i; $x_j^i \in \mathcal{X}$ represents the geometry input, and $y_j^i \in \mathcal{Y}$ denotes the corresponding mesh output. For the i-th task, the dataset $\mathcal{D}_i$ contains N_i samples:

$$\mathcal{D}_i = \{(x_j^i, y_j^i)\}_{j=1}^{N_i}. \tag{1}$$

From $\mathcal{D}_i$, we randomly sample $M_i \sim \text{Uniform}(1, M)$ examples to form a candidate set:

$$\mathcal{C}_i = \{(x_j^i, y_j^i)\}_{j \in \mathcal{M}_i}, \tag{2}$$

where $\mathcal{M}_i \subset \{1, \ldots, N_i\}$ and M is the maximum number of candidates. Then, one example

$$\mathbf{q}_i = (x_q^i, y_q^i) \tag{3}$$

is randomly selected from $\mathcal{C}_i$ as the question, with x_q^i as the input geometry and y_q^i as the corresponding mesh. The remaining examples form the context set:

$$\mathcal{K}_i = \mathcal{C}_i \setminus \{\mathbf{q}_i\}. \tag{4}$$

The model is provided with a context set $\{(x_j^i, y_j^i)\}_{j \in \mathcal{K}_i}$ and a question input x_q^i. The model predicts $\hat{y}_q^i$ as follows:

$$\mathcal{F} : \left(x_q^i, \{(x_j^i, y_j^i)\}_{j \in \mathcal{K}_i}\right) \mapsto \hat{y}_q^i. \tag{5}$$

Here, $\mathcal{F}$ serves as a meta-learned model that derives a task-specific mapping f_i from the provided context examples, enabling the prediction to approximate the behavior of $f_i(x_q^i)$:

$$\mathcal{F}(x_q^i, \{(x_j^i, y_j^i)\}_{j \in \mathcal{K}_i}) \approx f_i(x_q^i). \tag{6}$$

Prompt Construction

During the in-context learning process, geometry-mesh pairs are organized into a prompt. To effectively structure this prompt data, we introduce two coordinate spaces: the computational domain and the physical domain. The computational domain is defined as a regular grid over $[0, 1] \times [0, 1]$, where each grid point serves as an index, referred to as a key. The physical domain corresponds to the actual geometric region to be meshed. There exists a one-to-one mapping from the computational domain to the physical domain, associating each key with a specific physical coordinate, referred to as the value. This key–value mapping facilitates querying the physical domain via a computational coordinate (referred to as the query Q) to efficiently retrieve the corresponding mesh point. Typically, the query Q represents the entire computational domain and is omitted in (5) and (6). Moreover, it is important to note that this key–value mapping differs from the geometry-mesh mapping f; the former is used to structure data, whereas the latter describes structured mesh generation.

According to the above definitions, the prompt construction is described as follows. For each task i, the prompt is constructed from the corresponding context set $\mathcal{K}_i$ and question $\mathbf{q}_i$. Additionally, to enable the model to effectively distinguish between geometry and mesh while preserving the positional index of each example, we introduce signed one-hot index encodings e_j and $-e_j$ for the j-th example, where e_j represents the geometry and $-e_j$ corresponds to the associated mesh. We define two types of feature embeddings: $k_{\text{embedding}}(\cdot)$ for key representations and $v_{\text{embedding}}(\cdot)$ for value representations. These embeddings are generated by the hybrid encoder, which will be introduced in the following subsection. The prompt is organized as a two-dimensional tensor, with rows alternating between geometry and mesh for each example, and columns corresponding to the embeddings and index encodings. Notably, to support batch processing, we apply zero-padding to maintain consistent prompt sequence lengths and introduce a masking mechanism to eliminate the interference of padding.

Formally, for each task i, we concatenate along the feature dimension the key embedding $k_{\text{embedding}}(\cdot)$, value embedding $v_{\text{embedding}}(\cdot)$, and a signed index encoding e_j or $-e_j$ for each paired input (x_j^i, y_j^i), resulting in a prompt matrix prompt_i (with padding applied) of the form:

$$\text{prompt}_i = \begin{bmatrix} k_{\text{embedding}}(x_1^i) & v_{\text{embedding}}(x_1^i) & e_1 \\ k_{\text{embedding}}(y_1^i) & v_{\text{embedding}}(y_1^i) & -e_1 \\ k_{\text{embedding}}(x_2^i) & v_{\text{embedding}}(x_2^i) & e_2 \\ k_{\text{embedding}}(y_2^i) & v_{\text{embedding}}(y_2^i) & -e_2 \\ \vdots & \vdots & \vdots \\ k_{\text{embedding}}(x_M^i) & v_{\text{embedding}}(x_M^i) & e_M \\ k_{\text{embedding}}(y_M^i) & v_{\text{embedding}}(y_M^i) & -e_M \end{bmatrix}. \tag{7}$$

This structured prompt design enables the model to effectively capture contextual relationships across examples. With these structured data representations, the forward pass of our neural network $\mathcal{F}_\theta$, can be further expressed as

$$\mathcal{F}_\theta\left(\text{prompt}_i, Q\right) \mapsto \hat{y}_q^i, \quad \text{where} \quad \text{prompt}_i = (x_q^i, \mathcal{K}_i). \tag{8}$$

Neural Network Architecture

We adopt transformer (Vaswani et al. 2017) as backbone. To fully leverage the potential of the transformer in structured mesh generation, we designed a Hybrid Feature Encoder layer to encode the input data. Specifically, for geometry data, we first apply uniform sampling to both the key and value, followed by a projection to generate the corresponding $k_{\text{embedding}}$ and $v_{\text{embedding}}$. For mesh data, the key and value are passed through convolutional layers, after which they are flattened to obtain the corresponding $k_{\text{embedding}}$ and $v_{\text{embedding}}$. Finally, all embeddings are concatenated with the corresponding index encodings to construct the prompt.

After constructing the prompt, we apply a shared linear projection followed by layer normalization (Ba, Kiros, and Hinton 2016) to align the dimensionality of the input. These representations are used as the query, key, and value inputs to the transformer encoder, generating the final prompt embedding. The encoder consists of multiple layers, each with an identical structure, including a multi-head self-attention mechanism and a feedforward neural network with a GELU activation function (Hendrycks and Gimpel 2016). Each sub-layer is equipped with residual connections (He et al. 2016) and layer normalization.

The embedding produced by the encoder is subsequently normalized and used as the key and value inputs to the decoder. Meanwhile, the query vector is independently passed through a linear transformation to ensure dimensional compatibility with the decoder. This transformed query is then normalized and fed into the decoder as the query input.

Especially, to better suit this task, the decoder adopts a simplified design by removing the self-attention layer and retaining only the multi-head cross-attention and a feedforward network with GELU activation. Each sub-layer incorporates residual connections and layer normalization for stability. This structure ensures that each output depends solely on its query, enabling flexible mesh generation at arbitrary locations. Finally, the output of the decoder is projected into a space with dimensions consistent with the target mesh through linear projection, generating the predicted mesh corresponding to the question geometry.

Algorithm 1: Training Phase of ICL-Mesh

Input: Training set $\{\mathcal{D}_i\}_{i=1}^C$, Model $\mathcal{F}_\theta$, Batch size B, Max candidate example count M, Optimizer, Number of epochs E, Query Q, Training steps $T = \frac{C}{B}$
Output: Trained model parameters θ
for epoch = 1 to E **do**
  for step = 1 to T **do**
    Initialize batch loss: $\mathcal{L}_{batch} \leftarrow 0$
    for $b = 1$ to B **do**
      Get dataset $\mathcal{D}_b = \{(x_j^b, y_j^b)\}_{j=1}^{N_b}$
      Select the number of candidate examples: $M_b \sim \text{Uniform}(1, M)$
      Randomly select candidate indices $\mathcal{M}_b \subset \{1, \ldots, N_b\}$ with $|\mathcal{M}_b| = M_b$
      Build candidate set $\mathcal{C}_b \leftarrow \{(x_j^b, y_j^b)\}_{j \in \mathcal{M}_b}$
      Randomly select question index $q_b \in \mathcal{M}_b$
      Extract question example $\mathbf{q}_b = (x_{q_b}^b, y_{q_b}^b)$
      Build context set $\mathcal{K}_b \leftarrow \mathcal{C}_b \setminus \{\mathbf{q}_b\}$
      Forward pass: $\hat{y}_{q_b}^b \leftarrow \mathcal{F}_\theta\left(x_{q_b}^b, \mathcal{K}_b, Q\right)$
      Compute loss: $\mathcal{L}_b = \text{MSE}(\hat{y}_{q_b}^b, y_{q_b}^b)$
      Accumulate loss: $\mathcal{L}_{batch} \leftarrow \mathcal{L}_{batch} + \mathcal{L}_b$
    end for
    Compute average batch loss: $\mathcal{L}_{avg} = \mathcal{L}_{batch}/B$
    Update model: $\theta \leftarrow \text{Optimizer}(\theta, \nabla_\theta \mathcal{L}_{avg})$
  end for
end for

Algorithm 2: Inference Phase of ICL-Mesh

Input: Trained model $\mathcal{F}_\theta$, User-provided context set $\mathcal{K}$, User-specified question geometry x_q, Query Q
Output: Predicted mesh $\hat{y}_q$ for question geometry x_q
// No parameter update during inference
Forward pass: $\hat{y}_q \leftarrow \mathcal{F}_\theta\left(x_q, \mathcal{K}, Q\right)$
Output: Predicted mesh $\hat{y}_q$

Training and Inference

The training process is outlined in Algorithm 1. In each iteration, a batch of tasks is sampled. For each selected task, a candidate set of examples is randomly drawn. One example is designated as the question, and the remaining examples form the context set. The model takes the question geometry and its context as a prompt, along with the query as input, and predicts the corresponding mesh. The prediction is then compared against the ground truth using the mean squared error loss, which is used to update the model parameters.

The inference process is described in Algorithm 2. After training, the model generates the corresponding mesh for an unseen geometry by leveraging the given in-context examples as guidance, without requiring any parameter updates.

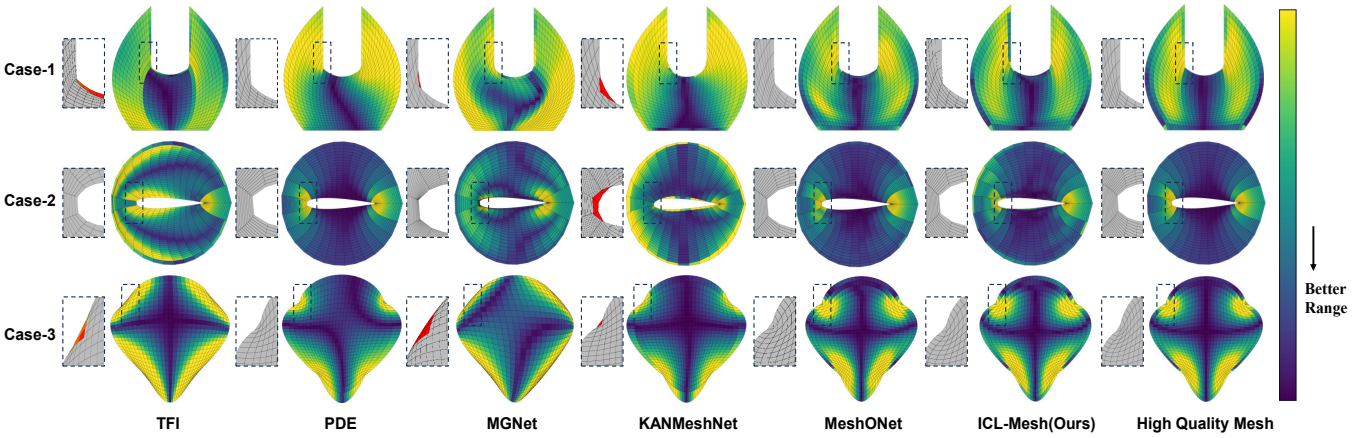

Figure 4: Comparison of mesh quality among various structured mesh generation methods. Invalid mesh elements are shown in red in the zoomed-in views, indicating that the mesh is unusable for numerical simulations.

Experimental Results

Experimental Setup

Our method was evaluated on a server with two NVIDIA A100 (80GB) SXM4 GPUs, while other methods were tested on a single A100 GPU. The software environment includes Python 3.11.11, PyTorch 2.6.0, and CUDA 12.2.

Datasets

Building upon the previous works (Chen et al. 2022a; Peng, Chen, and Liu 2025; Xiao et al. 2025), we further constructed expanded 2D and 3D datasets using the mesh generation software Pointwise (Pointwise 2018). Each dataset includes a large number of geometrically diverse samples, each containing 18 to 20 geometry–mesh pairs. These pairs serve as in-context examples within each sample.

Metrics

We evaluate the mesh quality of different methods using the following metrics (Wang et al. 2022):

- **Mesh Validity**: Checks whether the mesh contains any degenerate or self-intersecting elements that render it invalid.
- **Max Included Angle**: The largest angle between two adjacent edges in a mesh element; smaller values indicate better mesh quality.
- **Min Included Angle**: The smallest angle between two adjacent edges; larger values are preferred for higher mesh quality.
- **Equiangle Skewness**: Measures the deviation from an equilateral element; lower values correspond to better mesh quality.
- **Aspect Ratio**: The ratio of the longest edge to the shortest edge of an element; values closer to 1 indicate higher mesh quality.

Results

The experiments consist of five parts. First, we compare the mesh quality with existing approaches. Next, we investigate the method's capabilities in mesh refinement and coarsening, followed by an evaluation of its scalability in three-dimensional scenarios. Then, we analyze the impact of the ordering of in-context examples, and finally, evaluate its few-shot learning ability.

Mesh Quality Evaluation We conducted experiments on diverse geometries, splitting the training and testing sets to ensure test geometries were unseen during training. To evaluate the quality of meshes generated by our method, We selected several baselines for comparison, including traditional methods such as TFI (Allen 2008) and PDE-based approaches (Shragge 2006), as well as neural network-based methods including MGNet (Chen et al. 2022a), KANMeshNet (Peng et al. 2024), and MeshONet (Xiao et al. 2025). Since these baselines lack generalization capability, traditional methods were executed separately for each test case, while neural network-based methods were trained individually on specific cases to produce respective results.

As shown in Figure 4, our method achieves strong performance across all test cases in terms of the Max Included Angle. The quantitative results in Table 1 further demonstrate that our approach is comparable to existing methods across multiple metrics and even achieves state-of-the-art results in Case-1. By comparison, TFI suffers from distortions and self-intersections near complex boundaries due to the limitations of its interpolation-based formulation. KANMeshNet and MGNet exhibit reduced performance near complex boundaries, which may be attributed to the lack of explicit boundary processing in their architectures. In contrast, MeshONet leverages a dual-branch design that decouples boundary and domain features, allowing it to better capture complex boundary conditions. These findings suggest that although our method emphasizes generalization, it still achieves competitive mesh quality by learning shared structural patterns across geometries.

Test Case	Method	Mesh Validity	Max Included Angle ↓	Min Included Angle ↑	Equiangle Skewness ↓	Aspect Ratio ↓
Case-1	TFI	Invalid	120.829	59.743	0.344	1.924
	PDE	**Valid**	120.560	60.248	0.344	**1.734**
	MGNet	**Valid**	131.946	49.027	0.468	1.881
	KANMeshNet	Invalid	120.054	60.836	0.339	1.712
	MeshONet	**Valid**	120.631	60.238	0.343	1.925
	ICL-Mesh	**Valid**	**119.896**	**60.925**	**0.336**	1.941
Case-2	TFI	Invalid	109.731	70.391	0.224	6.462
	PDE	**Valid**	104.180	76.232	0.161	6.635
	MGNet	**Valid**	104.531	75.844	0.165	6.930
	KANMeshNet	Invalid	105.116	75.112	0.173	5.364
	MeshONet	**Valid**	**100.002**	**81.002**	**0.111**	**5.811**
	ICL-Mesh	**Valid**	100.742	80.035	0.121	5.816
Case-3	TFI	Invalid	123.615	58.449	0.373	1.191
	PDE	**Valid**	**116.253**	**64.973**	**0.294**	**1.196**
	MGNet	Invalid	135.326	46.056	0.504	1.276
	KANMeshNet	Invalid	112.387	69.021	0.251	1.435
	MeshONet	**Valid**	117.371	64.495	0.307	1.432
	ICL-Mesh	**Valid**	117.585	64.256	0.309	1.446

Table 1: A quantitative comparison of different mesh generation methods on Case-1, Case-2, and Case-3. "↑" / "↓" indicate better directions. Bold and underlined values indicate best results among valid entries only.

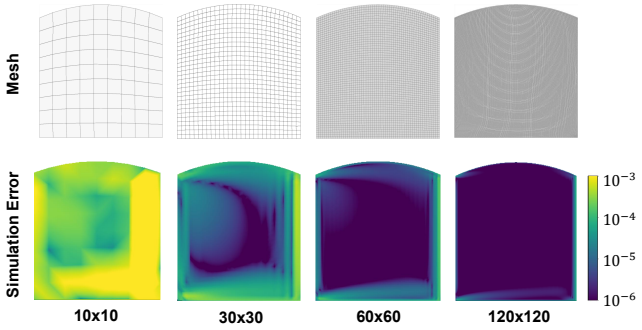

Figure 5: Experimental results on mesh refinement and coarsening, supporting cross-resolution generation without retraining. The upper part shows the generated meshes, while the lower part presents the simulation errors.

Mesh Refinement and Coarsening In practical applications, mesh resolution is often adjusted to suit specific problems. Therefore, a robust mesh generation method must be capable of handling both refinement and coarsening scenarios. Specifically, the model was trained on 30×30 meshes and directly generated meshes at 10×10, 60×60, and 120×120 resolutions without retraining.

Besides, to evaluate the practical applicability of the generated meshes, we conducted computational fluid dynamics simulations using the icoFOAM solver in OpenFOAM (Jasak et al. 2007). The setup modeled incompressible laminar flow through a 2D channel, with the left boundary as a velocity inlet, the right boundary as a pressure outlet, and the top/bottom walls as no-slip boundaries. The primary quantity of interest was the x-component of velocity.

As illustrated in Figure 5, our method performs well in both mesh coarsening and refinement, with the simulation error gradually decreasing as the mesh resolution increases, demonstrating the effectiveness of our approach.

Scalability to 3D Structured Mesh Generation While the experiments conducted thus far have primarily focused on 2D, this does not imply that our model is constrained to 2D scenarios. To extend the model to 3D structured mesh generation, only the Hybrid Feature Encoder needs modification. Specifically, we replace uniform sampling with convolutional layers followed by flattening to handle boundaries. Additionally, 2D convolutions are replaced with 3D convolutions for mesh processing.

Figure 6 presents the 3D structured meshes generated by our proposed method, clearly demonstrating its scalability and effectiveness in handling 3D configurations.

Robustness to Example Order Variations This section investigates the model's sensitivity to the order of provided examples, which is helpful for prompt construction. We randomly selected a diverse set of geometries as test samples. Each sample includes 20 geometry-mesh example pairs (labeled 1–20), where higher indices indicate increased complexity. To eliminate the influence of the number of examples, we fixed the number of training examples to five and defined seven groups with different example orders:

- **Groups A–B** represent in-distribution scenarios.
  - Group A: randomly sampled examples centered around index 10, arranged in ascending order.

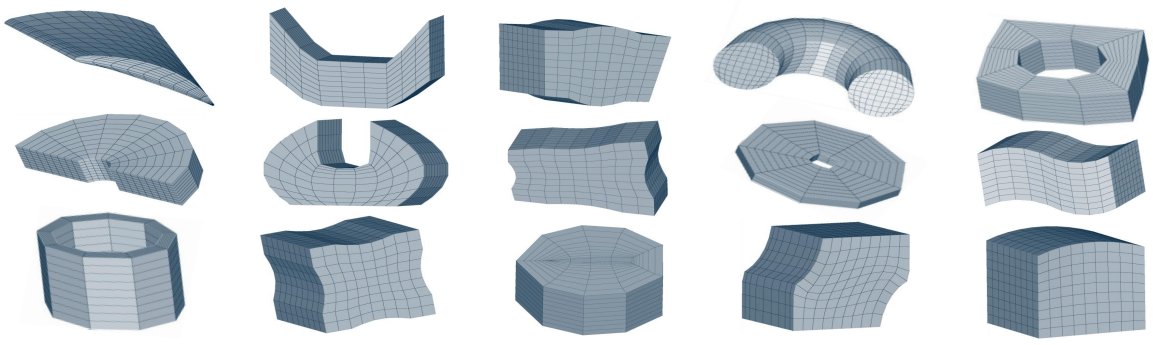

Figure 6: Illustration of 3D structured mesh generation under various geometries.

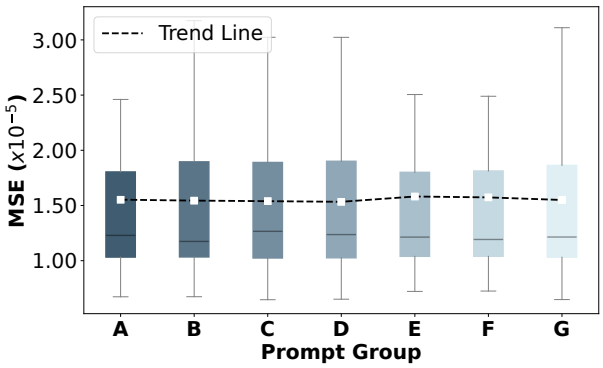

Figure 7: Comparison of model performance across seven example order groups (A–G).

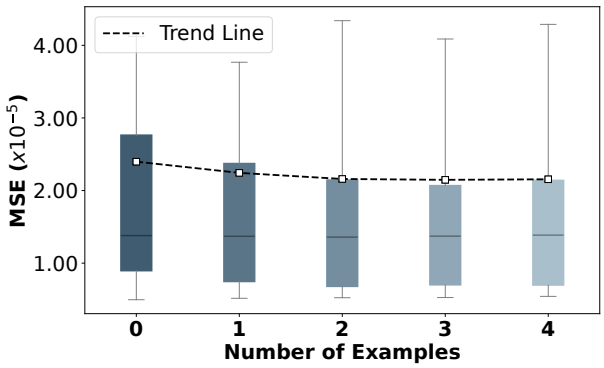

Figure 8: Comparison of model performance with varying numbers of examples.

  - Group B: same as Group A, but arranged in descending order.
- **Groups C–F** represent out-of-distribution scenarios.
  - Groups C–D: randomly sampled from indices less than 10, sorted in ascending/descending order respectively.
  - Groups E–F: randomly sampled from indices greater than 10, sorted in ascending/descending order respectively.
- **Group G** is a random combination, with index 10 fixed and others randomly selected.

For each test sample, each group was evaluated over 100 repeated trials. As shown in Figure 7, results were consistently similar across all groups, indicating that the model is robust to input order and capable of generalizing well across varied prompt sequences.

Few-Shot Learning Capability Analysis In this section, we investigate the model's few-shot learning ability by analyzing how its performance varies with the number of examples provided during inference. We randomly select a diverse set of geometries as test samples, evaluating each sample 100 times with different numbers of examples.

As shown in Figure 8, the MSE decreases as the number of examples increases, indicating that our method effectively leverages in-context information. Notably, it demonstrates strong few-shot learning capability, achieving reasonable performance even in one-shot and zero-shot settings.

Conclusion

In this paper, we propose ICL-Mesh, a meta-learning framework based on in-context learning for structured mesh generation. Our method overcomes the limitations of existing approaches that require recomputation or costly retraining when adapting to new geometries. Experimental results demonstrate that ICL-Mesh effectively generalizes to diverse geometries with only a few context examples, and even without any examples. It also shows robustness to the order of in-context examples and can be extended to various mesh generation tasks, such as mesh refinement and coarsening.

Currently, the scale of our experiments is constrained by computational resources. In future work, we aim to increase the scale of mesh data, and enhance the diversity of geometries. One promising direction is to incorporate more efficient attention mechanisms. Another potential direction is to implement a chunking strategy, dividing large-scale data into smaller portions that interact with each other. These modifications may reduce the heavy dependence on computational resources and further enhance the practical applicability of our approach.

Acknowledgments

This research was partially supported by the National Natural Science Foundation of China (12402349), the Natural Science Foundation of Hunan Province (2024JJ6468), the Innovation Reserch Foundation of National University of Defense Technology (ZK2023-11), and the National Key Research and Development Program of China (2021YFB0300101).

References

Adler, J. H.; De Sterck, H.; MacLachlan, S.; and Olson, L. 2025. Numerical partial differential equations.

Allen, C. 2008. Towards automatic structured multiblock mesh generation using improved transfinite interpolation. *International Journal for Numerical Methods in Engineering*, 74(5): 697–733.

Ba, J. L.; Kiros, J. R.; and Hinton, G. E. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901.

Chen, X.; Li, T.; Wan, Q.; He, X.; Gong, C.; Pang, Y.; and Liu, J. 2022a. MGNet: a novel differential mesh generation method based on unsupervised neural networks. *Engineering with Computers*, 38(5): 4409–4421.

Chen, X.; Liu, J.; Yan, J.; Wang, Z.; and Gong, C. 2022b. An improved structured mesh generation method based on physics-informed neural networks. *arXiv preprint arXiv:2210.09546*.

Chen, X.; Liu, J.; Zhang, Q.; Liu, J.; Wang, Q.; Deng, L.; and Pang, Y. 2023. Developing a novel structured mesh generation method based on deep neural networks. *Physics of Fluids*, 35(9).

Chen, X.; Wang, Z.; Deng, L.; Yan, J.; Gong, C.; Yang, B.; Wang, Q.; Zhang, Q.; Yang, L.; Pang, Y.; et al. 2024. Towards a new paradigm in intelligence-driven computational fluid dynamics simulations. *Engineering Applications of Computational Fluid Mechanics*, 18(1): 2407005.

Chen, Y.; Zhong, R.; Zha, S.; Karypis, G.; and He, H. 2021. Meta-learning via language model in-context tuning. *arXiv preprint arXiv:2110.07814*.

Dai, D.; Sun, Y.; Dong, L.; Hao, Y.; Ma, S.; Sui, Z.; and Wei, F. 2022. Why can gpt learn in-context? language models implicitly perform gradient descent as meta-optimizers. *arXiv preprint arXiv:2212.10559*.

Dong, Q.; Li, L.; Dai, D.; Zheng, C.; Ma, J.; Li, R.; Xia, H.; Xu, J.; Wu, Z.; Liu, T.; et al. 2022. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*.

Gharoun, H.; Momenifar, F.; Chen, F.; and Gandomi, A. H. 2024. Meta-learning approaches for few-shot learning: A survey of recent advances. *ACM Computing Surveys*, 56(12): 1–41.

Gurney, K. 2018. *An introduction to neural networks*. CRC press.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hendrycks, D.; and Gimpel, K. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

Hospedales, T.; Antoniou, A.; Micaelli, P.; and Storkey, A. 2021. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(9): 5149–5169.

Jasak, H.; Jemcov, A.; Tukovic, Z.; et al. 2007. OpenFOAM: A C++ library for complex physics simulations. In *International workshop on coupled methods in numerical dynamics*, volume 1000, 1–20. Dubrovnik, Croatia).

Kirchmeyer, M.; Yin, Y.; Donà, J.; Baskiotis, N.; Rakotomamonjy, A.; and Gallinari, P. 2022. Generalizing to new physical systems via context-informed dynamics model. In *International Conference on Machine Learning*, 11283–11301. PMLR.

Lei, N.; Li, Z.; Xu, Z.; Li, Y.; and Gu, X. 2023. What's the situation with intelligent mesh generation: A survey and perspectives. *IEEE transactions on visualization and computer graphics*.

Liu, J.; Shen, D.; Zhang, Y.; Dolan, B.; Carin, L.; and Chen, W. 2021. What Makes Good In-Context Examples for GPT-3? *arXiv preprint arXiv:2101.06804*.

Liu, Z.; Wang, Y.; Vaidya, S.; Ruehle, F.; Halverson, J.; Soljačić, M.; Hou, T. Y.; and Tegmark, M. 2024. Kan: Kolmogorov-arnold networks. *arXiv preprint arXiv:2404.19756*.

Min, S.; Lewis, M.; Zettlemoyer, L.; and Hajishirzi, H. 2021. Metaicl: Learning to learn in context. *arXiv preprint arXiv:2110.15943*.

Peng, J.; Chen, X.; and Liu, J. 2025. 3DMeshNet: a three-dimensional differential neural network for structured mesh generation. *Graphical Models*, 139: 101257.

Peng, J.; Chen, X.; Zhang, Q.; Deng, L.; Shen, L.; and Liu, J. 2024. A data-free Kolmogorov–Arnold Network-based method for structured mesh generation. *Physics of Fluids*, 36(11).

Pointwise, I. 2018. Pointwise: A grid generation software for computational fluid dynamics, Version 18.1 R2.

Raissi, M.; Perdikaris, P.; and Karniadakis, G. E. 2019. 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.

Shragge, J. 2006. Differential mesh generation. In *SEG International Exposition and Annual Meeting*, SEG–2006. SEG.

Ulaby, F. 2025. *Electromagnetics for engineers*. Michigan Publishing Services.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Wang, K.; Li, Z.; Zhang, R.; Ma, R.; Huang, L.; Wang, Z.; and Jiang, X. 2025. Computational fluid dynamics-based ship energy-saving technologies: A comprehensive review. *Renewable and Sustainable Energy Reviews*, 207: 114896.

Wang, Z.; Chen, X.; Gong, C.; Yang, B.; Deng, L.; Sun, Y.; Pang, Y.; and Liu, J. 2024. GNNRL-Smoothing: A Prior-Free Reinforcement Learning Model for Mesh Smoothing. *arXiv preprint arXiv:2410.19834*.

Wang, Z.; Chen, X.; Li, T.; Gong, C.; Pang, Y.; and Liu, J. 2022. Evaluating mesh quality with graph neural networks. *Engineering with Computers*, 38(5): 4663–4673.

Wittenborg, T.; Baimuratov, I.; Franzén, L. K.; Staack, I.; Römer, U.; and Auer, S. 2025. Knowledge-Based Aerospace Engineering–A Systematic Literature Review. *arXiv preprint arXiv:2505.10142*.

Xiao, J.; Chen, X.; Wang, Q.; and Liu, J. 2025. MeshONet: A Generalizable and Efficient Operator Learning Method for Structured Mesh Generation. *arXiv preprint arXiv:2501.11937*.

Yang, L.; Liu, S.; Meng, T.; and Osher, S. J. 2023. In-context operator learning with data prompts for differential equation problems. *Proceedings of the National Academy of Sciences*, 120(39): e2310142120.

Yang, L.; Liu, S.; and Osher, S. J. 2025. Fine-tune language models as multi-modal differential equation solvers. *Neural Networks*, 107455.

Yin, Y.; Ayed, I.; de Bézenac, E.; Baskiotis, N.; and Gallinari, P. 2021. Leads: Learning dynamical systems that generalize across environments. *Advances in Neural Information Processing Systems*, 34: 7561–7573.

Zhang, H.; Wang, M.; Li, H.; and Li, N. 2025. MeshKINN: A self-supervised mesh generation model based on Kolmogorov–Arnold-Informed neural network. *Expert Systems with Applications*, 275: 126959.

Zhang, L.; You, H.; Gao, T.; Yu, M.; Lee, C.-H.; and Yu, Y. 2023. Metano: How to transfer your knowledge on learning hidden physics. *Computer Methods in Applied Mechanics and Engineering*, 417: 116280.