GMNO ON MAXWELLBENCH: SCALING PRETRAINED GENERAL MESH NEURAL OPERATOR ACROSS INDUSTRIAL-STANDARD ELECTROMAGNETISM SIMULATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Engineers routinely solve partial differential equations (PDEs) numerically to simulate physical phenomena and evaluate industrial designs. To alleviate the high computational cost of these simulations, neural operators have become a promising class of surrogate models for fast inference. While recent works have advanced neural operator architectures and begun to explore their scaling properties, most studies rely on heavily simplified PDE data that fail to reflect the structural complexity of industrial simulations. In practice, industrial simulation data have complex structure, require substantial domain expertise to interpret, and remain largely inaccessible to the broader machine learning community. We argue that a true foundation neural operator should generalize across industrial simulation scenarios that go far beyond simple parameter variations, while remaining governed by the same underlying PDEs. To bridge this gap, we introduce MaxwellBench, the first open-source industrial-standard benchmark comprising 14 datasets from distinct electromagnetism applications, and propose the General Mesh Neural Operator (GMNO), a neural operator with novel embedding architectures for handling complex multi-modal PDE data defined on meshes. We also develop a heterogeneous parallel training paradigm that enables large scale training on industrial-standard datasets. Extensive experiments show that GMNO consistently outperforms existing neural operator baselines on industrial-standard datasets. Furthermore, large scale pretraining yields strong out-of-distribution generalization, highlighting the promise of scaling neural operators on industrial applications.

1 INTRODUCTION

Partial Differential Equations (PDEs) are fundamental descriptors developed by human beings to model various physical phenomenons. However, it is impossible to obtain analytical solutions to PDEs in real world 2D or 3D problems. Therefore, the common practice in industry is to solve PDEs numerically via certain discretization schemes such as finite element method (FEM) (Dhatt et al., 2012), finite volume method (Eymard et al., 2000), finite difference method (Zhou, 1993), boundary element method (Hall, 1994), and other particle based methods (Müller et al., 2003). Across various discretization schemes, FEM has been the most prevailing numerical method across a wide range of industrial simulation scenarios including electromagnetics (Bensalem & Abdelkrim, 2016), thermal dynamics (Alzahrani et al., 2020), structural mechanics (Duarte et al., 2000), acoustics (Thompson, 2006), and fluid dynamics (Dick, 2009). Although FEM can accurately simulate real world phenomenons upon proper calibration, it requires near $O(n^3)$ (with respect to mesh size) computation complexity which prohibits high-throughput design evaluation on complex problems.

A potential solution to reduce the high computation cost of numerical simulations is to train neural operators as surrogate models to learn the projection from input to solution function spaces so that the inference time can be reduced by orders of magnitudes (Lu et al., 2021; Li et al., 2020b; Hao et al., 2023). However, neural operators are yet not broadly applied in industries due to two major bottlenecks: First, operator learning literatures assume point cloud field representation with concatenated physics channels (one channel per input modality), whereas industrial simulation

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

data are typically represented on meshes, with complex multi-modal physical information defined on mesh entities of different dimensions. This mismatch between simplified representations and industrial FEM data structures remains one of the major obstacles for deploying neural operators in real applications. Second, current operator learning paradigm has low generalization capability where parameter shifts are partially allowed (Goswami et al., 2022; Liu et al., 2025) but completely reparameterized geometries and simulations settings have to be learned from scratch. In other words, neural operators are only transferring the computation cost from inference to training, where the benefit is relatively marginal. The first bottleneck has been underexplored due to lack of industrial-standard open source datasets. Although recent works (Wu et al., 2024; Luo et al., 2025) have addressed industrial-size datasets (Elrefaie et al., 2024), these datasets do not fully capture the structural complexity of industrial-standard simulations. Meanwhile, some preliminary attempts were made to explore the second bottleneck via large scale operator pretraining under simplified scenarios (Subramanian et al., 2023; McCabe et al., 2024; Herde et al., 2024; Hao et al., 2024). Although clear scaling effect can be observed from these works, extreme simplifications on data structures are made by taking the pixel space and assuming concatenated physics channels, making them more of scalability study on image or video like data rather than physical simulations.

To narrow the gap between neural operators and industrial simulation data structures, and to explore the possibility of generalization across diverse industrial scenarios, we propose the General Mesh Neural Operator (GMNO) and introduce MaxwellBench, an industrial-standard benchmark designed to evaluate the expressiveness and scalability of GMNO. MaxwellBench contains more than 140k simulation files from 14 industrial electromagnetism applications, each treated as a separate dataset. Although all applications are governed by Maxwell's equations, they differ substantially in topology, geometry, and simulation settings, going far beyond simple parameter variations. To faithfully reflect industrial FEM data formats, MaxwellBench preserves mesh-based field representations together with multi-modal inputs, including sources, material properties, and boundary conditions. To model such complex data, we design novel embedding and fusion modules in GMNO for precise aggregation and processing of multi-modal physical fields defined on complex mesh structures. GMNO consistently outperforms existing neural operators on all MaxwellBench datasets under in-distribution (ID) training and evaluation. To further assess its scalability and generalization ability, we pretrain a 0.54B parameter GMNO on 10 MaxwellBench datasets to approximate the solution operators induced by Maxwell's equations. The pretrained GMNO yields additional gains on ID tasks and, more importantly, demonstrates strong transfer performance on out-of-distribution (OOD) datasets under scarce data finetuning. The contributions of this work can be summarized as follows:

- We introduce **MaxwellBench**, an industrial-standard benchmark comprising 14 datasets from distinct 2D electromagnetism applications, with mesh-based field representations, multi-modal physical inputs, and diverse simulation settings, including static and frequency-domain problems in Cartesian and cylindrical coordinates. MaxwellBench captures the structural complexity of industrial simulation data through heterogeneous meshes, wide field dynamic ranges, and varied application scenarios.
- We propose **GMNO**, a scalable general-purpose neural operator designed for industrial-standard mesh simulation data. GMNO incorporates dedicated physics encoders for multi-modal inputs defined on mesh entities and employs exact linear attention to avoid the ambiguities introduced by padded computation. It achieves state-of-the-art (SOTA) performance across all MaxwellBench datasets.
- We develop a distributed large-scale training pipeline and pretrain a 0.54B parameter GMNO across heterogeneous PDE datasets for the first time. The pretrained model delivers further gains on ID tasks and strong transfer to OOD datasets under scarce data finetuning, revealing a clear scaling effect on industrial-standard PDE data.

2 BACKGROUND

In this section, we explain the problem setup, related literature, and introduce MaxwellBench which is designed to mirror industrial electromagnetic simulations across a broad and diverse set of scenarios. Please refer to Appendix A.2 for a complete notation list.

2.1 PROBLEM SETUP

Consider a PDE problem defined on a domain $\Omega \subset \mathbb{R}^d$. The domain is partitioned into a few subdomains $\{\Omega_k\}_{k=1}^K$. This setup reflects practical industrial simulations, where inputs are rarely specified as continuously varying random fields (e.g. Gaussians), but instead arise from piecewise-defined material regions and sources. We then define the dimension-one skeleton $\Gamma = \cup_{k=1}^K \partial\Omega_k$ which collects all $d-1$ dimensional boundary and subdomain interface entities (embedded in $\mathbb{R}^d$). Our goal is to train a model to approximate the operator $\mathcal{G} : \mathcal{V}_\Omega \times \mathcal{W}_\Gamma \to \mathcal{H}_\Omega$. Here, $\mathcal{V}_\Omega$ denotes the input function space over Ω (e.g. material, source, geometry), $\mathcal{W}_\Gamma$ denotes the input function space defined on Γ (e.g. boundary condition, interface shapes), and $\mathcal{H}_\Omega$ denotes the solution function space.

In this work, we hope to explore the scaling potential of GMNO on industrial-standard simulation data, and the possibility of generalization towards OOD data given the same set of governing PDEs. Therefore, we construct MaxwellBench, a suite of 14 datasets all governed by Maxwell's equations as shown in Eq. 1, where $\mathbf{B}$, $\mathbf{J}$, $\mathbf{H}$, $\mathbf{E}$, $\mathbf{D}$ represent magnetic flux density, current density, magnetic field, electric field, and electric flux density. Due to limited computation resource, this work only considers 2D electromagnetic simulations with $\Omega \subset \mathbb{R}^2$.

$$\begin{aligned} \nabla \times \mathbf{E} &= -\frac{\partial \mathbf{B}}{\partial t}, & \nabla \times \mathbf{H} &= \mathbf{J} + \frac{\partial \mathbf{D}}{\partial t}, \\ \nabla \cdot \mathbf{D} &= \rho, & \nabla \cdot \mathbf{B} &= 0. \end{aligned} \tag{1}$$

Although governed by the same set of PDEs, the 14 datasets are constructed based on different industrial applications including inductor, transformer, and electromagnets. Each application further includes different analysis types, coordinate systems, geometry parameterization, mesh configurations, and field magnitudes. More details including notations, simulation configurations, file formats, dataset statistics, prediction tasks, and visualizations can be found in Appendix B.

2.2 MESH DISCRETIZATION

As it is impossible to obtain analytical solutions to the complex PDE problems in industry, the input and solution physical fields are oftentimes defined on discretized meshes and solved with numerical methods (e.g. finite element method). In this work, we consider the 2D triangular mesh discretization of Ω, which typically includes one node coordinate matrix and multiple connectivity matrices: the nodes are stored in $\mathbf{X} \in \mathbb{R}^{N_n \times 2}$, written either as $\mathbf{X}_{i,:} = (x_i, y_i)$ in Cartesian coordinates or as $\mathbf{X}_{i,:} = (r_i, z_i)$ in cylindrical coordinates; 2D triangular elements are given by the connectivity matrix $\mathbf{C}^T \in \{1, \ldots, N_n\}^{N_e \times 3}$ whose row $\mathbf{C}^T_{e,:} = (n_{e1}, n_{e2}, n_{e3})$ lists the global node indices of triangle e, while 1D line elements on subdomain boundaries and interfaces are given by $\mathbf{C}^\Gamma \in \{1, \ldots, N_n\}^{N_\Gamma \times 2}$ (this includes domain boundaries $\partial\Omega \subset \Gamma$), with each row containing the two endpoint node indices of the corresponding segment.

Most existing PDE datasets reduce meshes into point clouds by interpolating field variables onto nodes or element centers. While this simplified data handling, it blurs the distinction between different topological dimensions (faces, edges, and nodes) and discards geometric measures (area and length) and fine-grained boundary details, which are crucial in complex problem settings. Instead, we explicitly associate each physical quantity with its corresponding mesh entity and construct datasets that more faithfully reflect industrial simulation workflows. Specifically for 2D electromagnetism problems, material properties, current excitation source, and magnetic fields are considered as 2D fields $\{(\cdot)_e\}_{e=1}^{N_e}$ embedded on triangular elements, while boundary conditions are considered as 1D fields $\{(\cdot)_l\}_{l=1}^{N_\Gamma}$ embedded on line elements.

2.3 RELATED WORKS

Neural Operators: Neural operators have emerged in the past few years as fast surrogate models of industrial physical simulations in various fields including fluid dynamics (Bonnet et al., 2022), thermodynamics (Chen et al., 2025), structural mechanics (Koric et al., 2024), electromagnetism (Augenstein et al., 2023), etc. Operator learning research centers on how to represent and manipulate field variables as functions, rather than treating them as ordinary numeric feature vectors as in most

other deep learning settings. As the pioneer work in operator learning, DeepONet (Lu et al., 2021) first introduced the format of branch network for input fields and trunk network for querying output fields. Fourier Neural Operator (Li et al., 2020b) then proposes to project fields in frequency domain given a regular grid structure. A few followup works such as Geo-FNO (Li et al., 2023a), NUNO (Liu et al., 2023), GINO (Li et al., 2023b) extend FNO to geometries with irregular grids.

With the rise of transformer architectures (Vaswani et al., 2017), researchers start to apply various forms of attention architectures as backbones of neural operators. Cao (2021) first introduced linearized attention layers for operator learning tasks with high computation efficiency. OFormer (Li et al., 2022) introduced a new framework allowing flexible input and output fields probed at arbitrary coordinates. Further progress was made by GNOT (Hao et al., 2023) which proposed heterogeneous attention mechanism for aggregating multi-modal field inputs. Transolver (Wu et al., 2024) and LNO (Wang & Wang, 2024) proposed different mechanisms to perform attention on compressed token space and improve computation efficiency on PDE problems with large point clouds. Another line of work compresses token number via graph neural operator layers (Li et al., 2020a) as demonstrated in GAOT (Wen et al., 2025), where neighbors are selected within predefined balls.

Large-Scale Pretraining for Neural Operators: Large-scale pretraining has been proven highly successful in natural language processing (Achiam et al., 2023) and computer vision (Kirillov et al., 2023). Similar practices are also applied in scientific machine learning fields including AlphaFold (Jumper et al., 2021), GNoME (Merchant et al., 2023), Fourcastnet (Bonev et al., 2025), etc. Research in these fields show that pretraining at large-scale has the potential of strong OOD generalization which is valuable for operator learning problems where collecting PDE solution data on new tasks is extremely expensive.

A few large-scale operator pretraining attempts have been made these years focusing on autoregressive tasks on pixel space (for instance PDEBench (Takamoto et al., 2022)), including MPP (McCabe et al., 2024), Poseidon (Herde et al., 2024), DPOT (Hao et al., 2024), etc. Unlike the operator $\mathcal{G}$ defined above which directly models the projection between inputs and outputs of industrial simulations, an autoregressive operator assumes the form of $\mathcal{G}_{AR} : \mathcal{H}_\Omega^m \to \mathcal{H}_\Omega$ where the model learns to infer dynamics from historical frames and rollout future frames. Such autoregressive task configuration and pixel space representation significantly simplifies the structure of numerical simulations, but has minor significance in industrial scenarios. Although these abovementioned work show clear indication of scaling, they should be treated more as extensions of video generation modeling rather than scaling of neural operators. There remains a huge gap to be explored in scaling of neural operators on industrial standard simulation data.

3 METHOD

3.1 MAXWELLBENCH

We introduce MaxwellBench to address the lack of industrial-standard simulation data in the existing literature. MaxwellBench comprises 14 datasets, each derived from a distinct power electronics application with its own geometry parameterization. This design gives MaxwellBench substantial diversity and complexity, covering a wide range of simulation configurations, mesh scales, element sizes, and magnetic field magnitudes, as summarized in Appendix B. All datasets in MaxwellBench are constructed and solved using COMSOL FEM (Multiphysics, 1998), and the complete industrial-standard input fields, output fields, and mesh structures (described in Section A.3) are exported for training. Compared with existing datasets, MaxwellBench is distinguished by its mesh-based field representation, extreme geometric irregularity, multi-modal input fields, heterogeneous data distributions, and wide value ranges. Additional details of MaxwellBench are explained in Appendix B with a few sample data provided in Supplementary.

3.2 MODEL ARCHITECTURE

Modern deep learning architectures in natural language processing and computer vision follow a common paradigm: a tokenizer first lifts the input into a set of tokens, which are then processed by a token mixer, most often attention layers. Existing neural operators largely adopt a similar strategy, but usually rely on simplified field representations in pixel space or point clouds. In this formulation,

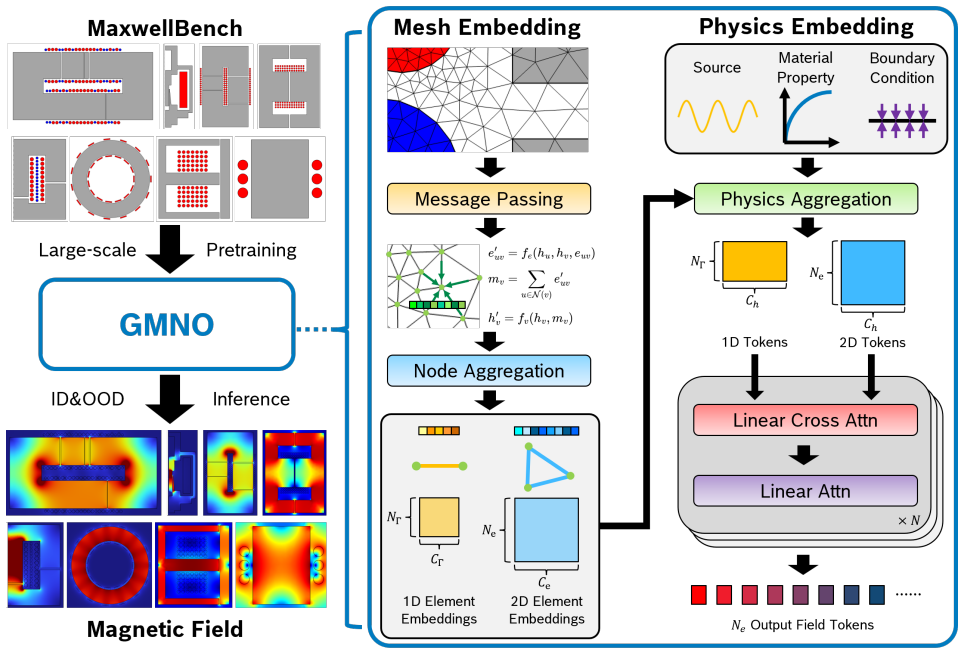

Figure 1: An illustration of GMNO architecture. GMNO is designed for large-scale training on PDE datasets with industrial-standard structures, such as MaxwellBench. GMNO first performs message passing to obtain node embeddings, which are aggregated into element embeddings. The input field is then tokenized by assigning input conditions onto corresponding element embeddings. A few linear attention layers then process the input tokens to predict output physical fields.

input fields are usually encoded as channels concatenated with pixel or point coordinates, after which token compression and token mixing modules are applied. However, such a representation obscures geometric details and makes strong assumptions about input modalities that are often unrealistic in complex industrial simulation settings. To address this limitation, we instead retain the full mesh information and introduce a multi-modal input field fusion strategy at the tokenizer stage, as illustrated in Fig. 1. Specifically, each input modality is fused with and tokenized together with its corresponding geometric entity, either the d dimensional computational domain or the $d-1$ dimensional skeleton.

Mesh Embedding. Given the mesh information (node coordinates $\mathbf{X}$, triangular element connectivity $\mathbf{C}^T$, and line element connectivity $\mathbf{C}^\Gamma$) as input, we first compute node embeddings $\mathbf{Node}^{emb}$ using several message passing layers, where node neighbors are directly derived from the connectivity matrices. We then construct 2D triangular element embeddings $\mathbf{Tri}^{emb} \in \mathbb{R}^{N_e \times C_e}$ and 1D line element embeddings $\mathbf{Lin}^{emb} \in \mathbb{R}^{N_\Gamma \times C_\Gamma}$ by aggregating and averaging the embeddings of their associated nodes in a permutation invariant manner as shown in Eq. 2. This design enables each element embedding to capture rich geometric information, including spatial location, shape, size, and local neighborhood structure.

$$\begin{aligned} \mathbf{Node}^{\text{emb}} &= \text{MessagePassing}\big(\mathbf{X}^+, \mathcal{N}(\mathbf{X})\big)\,, \\ \mathbf{Tri}_e^{\text{emb}} &= \sum_{n \in \mathbf{C}^T_{e,:}} \mathbf{Node}_n^{\text{emb}}, \qquad \mathbf{Lin}_l^{\text{emb}} = \sum_{n \in \mathbf{C}^\Gamma_{l,:}} \mathbf{Node}_n^{\text{emb}}. \end{aligned} \tag{2}$$

Multi-Modal Physics Embedding. Industrial simulations often involve input physics information from multiple modalities that cannot be naively concatenated. We therefore encode each modality separately and fuse them at a later stage. In typical industrial simulation settings, the major input modalities include sources, material properties, and boundary conditions, each of which is lifted into the embedding space using an encoder tailored to its structure. For MaxwellBench, global parameter embeddings ξ^{emb}, 2D area embeddings $\mathbf{A}^{emb}$, 1D length embeddings $\mathbf{L}^{emb}$, and source embeddings $\mathbf{S}^{emb}$ are generated by an MLP; material embeddings $\mathbf{M}^{emb}$ are obtained by first encoding the

BH curve with a Transformer encoder and then linearly modulating the resulting representation with additional material parameters; boundary condition embeddings $\mathbf{P}^{emb}$ are constructed by concatenating encoded boundary condition types, represented through a learnable lookup table, with boundary normals encoded by an MLP.

Aggregation. Once properly lifted, the physics embeddings should be aggregated onto their mesh embeddings of corresponding geometric dimension via index selection. For MaxwellBench, 2D element embeddings would aggregate corresponding area embeddings, source embeddings, and material embeddings to generate tokenized 2D input fields $\mathcal{V}_\Omega$; while 1D element embeddings would aggregate corresponding length embeddings, boundary condition embeddings, and material embeddings on their neighbor 2D elements to generate tokenized 1D input fields $\mathcal{W}_\Gamma$ as shown in Eq. 3. Please refer to Appendix B for more details.

$$\begin{aligned} \mathcal{V}_\Omega^{token} &= MLP_\Omega(\mathbf{Tri}^{emb} \parallel \mathbf{M}^{emb} \parallel \mathbf{S}^{emb} \parallel \mathbf{A}^{emb}) \\ \mathcal{W}_\Gamma^{token} &= MLP_\Gamma(\mathbf{Lin}^{emb} \parallel \mathbf{M}^{emb} \parallel \mathbf{P}^{emb} \parallel \mathbf{L}^{emb}) \end{aligned} \tag{3}$$

Token Mixer. Two major approaches to token compression have been explored in the literature: learnable linear/nonlinear projection on sequence dimension (Wu et al., 2024; Wang & Wang, 2024), and graph-based projection (Li et al., 2023b; Wen et al., 2025). Although both have shown strong performance on various open-source datasets, they remain inadequate for meshes with large variations in element count, strong density heterogeneity, varying domain scales, and multiple input fields. We therefore retain all tokens and adopt the heterogeneous linear attention layers of GNOT (Hao et al., 2023) to process the tokenized input fields at different dimensions (Eq. 4). However, vanilla linear attention relies on zero-padded inputs for batched computation, which can introduce ambiguity when training at scale on complex and heterogeneous datasets, while also incurring substantial computation overhead when sample lengths vary by orders of magnitude. To eliminate this ambiguity and maximize memory efficiency, we process each sample independently within the linear attention module, and recover overall efficiency through distributed training and bucketed sampling.

$$\begin{aligned} \mathcal{Q}^{token} &= MLP_Q(\mathbf{Tri}^{emb} \parallel \mathbf{A}^{emb} \parallel \xi^{emb}) \\ pred &= MLP_{out}(LinearAttnBlocks(\mathcal{Q}^{token}, \mathcal{V}_\Omega^{token}, \mathcal{W}_\Gamma^{token})) \end{aligned} \tag{4}$$

3.3 Pretraining and Finetuning

To explore the scalability of GMNO, we first perform dense pretraining on 10 datasets from MaxwellBench, each containing 9600 simulation samples. A 0.54B parameter GMNO is trained on 8 Nvidia H100 GPUs using a bucketed distributed sampler and gradient accumulation, resulting in an effective batch size of 128. Additional parallel training, data augmentation, optimizer, and scheduler details are provided in Appendix C.3. For ID finetuning, we update all parameters for 10 epochs using 9600 samples per dataset. For OOD finetuning, we use 240 samples for 100 epochs and update only projection layers and LoRA modules, while freezing the mesh embedding, physics embedding, and pretrained attention parameters. In all cases, the finetuned models are evaluated on 960 held-out samples. Relative L_2 norm (Eq. 5) is used as the loss function for all training and evaluation, where $\hat{B}$ denotes the normalized magnetic flux density and D is the spatial dimension. To achieve generalization across MaxwellBench, we adopt a log normalization scheme as detailed in Appendix C.

$$Loss = \frac{1}{D}\sum_{d=1}^{D} \frac{\sqrt{\sum_{e=1}^{N_e}(GMNO_d(\cdot)_e - ||\hat{B}_d||_e)^2}}{\sqrt{\sum_{e=1}^{N_e}||\hat{B}_d||_e^2}} \tag{5}$$

4 Experiments

In this section, we first compare GMNO with representative neural operator baselines on individual MaxwellBench datasets, reporting the mean and 1-sigma error range over three random seeds. We

then pretrain GMNO across multiple MaxwellBench datasets to investigate its scaling behavior and OOD generalization on industrial-standard simulation data. Finally, we present ablation studies to analyze the key architectural components of GMNO.

4.1 Baseline Comparison

We compare GMNO against several representative neural operator baselines that are applicable to irregular grids, including GNOT (Hao et al., 2023), Transolver (Wu et al., 2024), LNO (Wang & Wang, 2024), and GAOT (Wen et al., 2025). Since all existing baselines assume point cloud field representations with concatenated physics channels, we adapt MaxwellBench accordingly as detailed in Appendix C. The comparison is conducted on 10 MaxwellBench datasets, each with 9600 training samples and 960 test samples. To ensure fairness, all models are matched to approximately 10$\sim$11M parameters and trained with the same optimizer and scheduler settings.

The first part of Table 1 reports the relative L_2 test errors of GMNO and the baseline models when trained separately on each of the 10 MaxwellBench datasets. GMNO consistently achieves SOTA performance, substantially outperforming all baseline models with a comparable number of trainable parameters. These results highlight the importance of preserving complete mesh information and introducing a dedicated fusion strategy for multi-modal physical inputs when modeling industrial-standard simulation data. The conventional point cloud representation may be adequate when mesh density is nearly uniform in space or consistent across samples, but it fails to capture complex geometric features at different scales once element shapes and measure information are discarded. Likewise, simple input concatenation implicitly assumes that values from different physical modalities can be treated as equivalent channels, which is inconsistent with the structure of numerical PDE solvers. Although such assumptions are common in the machine learning literature, they significantly limit the expressive power of neural operators.

Table 1: Comparison of GMNO with existing neural operator baselines under per-dataset training (1-sigma error range included), large-scale pretraining, and finetuning. Large-scale pretraining is performed jointly across all 10 datasets, while per-dataset training and finetuning are carried out separately on each dataset. All results are reported in units of $\times 10^{-2}$. The best results are highlighted in bold. Results with less than 5% relative difference are regarded as equivalent.

Dataset	GNOT 11.1M	Transolver 10.2M	LNO 10.3M	GAOT 11.0M	GMNO 10.4M	GMNO 0.54B	GMNO 0.54B
	Per-dataset training					Large-scale pretraining	Finetuning
Trans UU	12.6±0.4	8.0±0.3	20.3±8.4	10.7±0.8	3.8±0.5	**2.2**	**2.1**
Trans PQ	10.4±0.4	6.9±0.3	17.1±6.1	10.5±0.5	3.4±0.1	**2.0**	**1.9**
Induc UU	17.7±0.2	11.4±0.6	16.3±0.2	13.7±0.6	5.5±0.1	**3.5**	**3.4**
Induc I Gap	5.7±0.9	2.7±0.2	9.0±3.2	5.8±0.0	1.1±0.1	**0.7**	**0.7**
Induc EI Gap	8.8±0.0	5.9±0.4	8.5±0.5	9.8±0.2	3.8±0.1	**2.5**	**2.5**
Induc EE	3.5±0.1	2.3±0.1	2.8±0.1	5.5±0.6	1.3±0.0	**0.8**	**0.8**
Induc Sm Gap	13.5±0.1	11.7±0.5	42.9±1.8	14.9±0.2	6.6±0.1	**3.5**	**3.4**
Induc Lg	3.7±0.2	2.0±0.1	7.2±1.1	5.5±0.6	1.0±0.1	**0.8**	**0.7**
EM	2.3±0.1	1.7±0.0	2.6±0.1	4.7±0.3	**1.3**±0.0	**1.2**	**1.2**
EM C Wire	7.5±0.2	5.1±0.1	7.3±0.1	13.0±0.2	2.3±0.1	**1.4**	**1.3**

4.2 Scaling and Generalization

A practical foundation neural operator should generalize across diverse industrial scenarios governed by the same underlying PDEs. Such scenarios may differ not only in parameter values, but also in scale, topology, geometry, material properties, sources, boundary conditions, and even mesh configurations. To investigate the scalability and generalization ability of GMNO under these substantial variations, we pretrain a 0.54B parameter GMNO on 10 datasets from MaxwellBench and evaluate its ID and OOD performance under scarce-data finetuning.

Table 2: Results of scarce-data finetuning on OOD datasets. Pretrained models use the same 10 datasets in Table 1. All results are reported in units of $\times 10^{-2}$.

Model	Params	*Trans L*	*Induc EI*	*Induc Lg Gap*	*EM C Chunk*
Per-dataset training, $N = 240$					
GMNO	10.4M	33.1	20.1	29.8	10.2
OOD finetuning, $N = 240$					
Transolver	0.10B	14.8	4.3	8.7	3.8
GMNO	0.10B	4.7	3.4	7.3	3.6
GMNO	0.54B	**3.5**	**3.0**	**5.9**	**3.3**

The second part of Table 1 reports the relative L_2 test errors of the 0.54B GMNO pretrained on 96000 samples from the 10 MaxwellBench datasets. The large pretrained GMNO substantially outperforms all baseline models as well as the smaller GMNO trained separately on individual datasets, demonstrating the scaling potential of GMNO for extracting shared physical knowledge across industrial-standard simulation datasets. We also observe that the scaling gains are less pronounced on datasets with relatively simple geometries and physics, such as *Induc Lg* and *EM*, where 9600 samples already appear sufficient to train an accurate operator. The third part of Table 1 further reports the performance of the 0.54B GMNO after ID finetuning. ID finetuning yields nearly identical performance to the pretrained model, with only marginal improvements on a few datasets. This suggests that GMNO is able to learn a strong shared representation of the underlying electromagnetism during joint pretraining, leaving only limited room for further task-specific calibration and specialization on seen distributions.

The most important criterion for a foundation neural operator is its ability to generalize on OOD simulation data. Table 2 compares GMNO on four unseen MaxwellBench datasets under two settings: training from scratch and finetuning from the pretrained model, each with only 240 training samples. As detailed in Appendix B, these four datasets are held out from pretraining and differ in geometry or configuration from the pretraining applications. The results show that 240 samples are insufficient to train an accurate GMNO from scratch, but are enough to effectively adapt the pretrained 0.54B GMNO, yielding nearly an order of magnitude lower error. A finetuning strategy ablation in Appendix C.3.1 shows that a better transfer requires freezing most of the pretrained parameters, including the mesh embedding, physics embedding, and linear attention blocks, while updating only the projection layers together with a small number of LoRA modules injected into the linear attention blocks. This strategy helps preserve the shared Maxwell prior acquired during pretraining while allowing efficient adaptation to new applications.

For a matched-scale comparison, we also pretrain Transolver (the best baseline) and GMNO at approximately 0.1B parameters on the same 10 datasets and use the same finetuning strategy. GMNO obtains lower errors on all four OOD datasets, with the largest gap on *Trans L*.

4.3 Ablation Studies

We conduct a series of ablation studies by training and testing small scale models individually on 4 datasets in MaxwellBench, to explore the effectiveness of different modules in GMNO.

Message Passing in Mesh Embedding. Message passing plays a central role in mesh embedding, as it allows each node to incorporate information from the connectivity and geometry of its local neighborhood. To assess its importance, we replace the message passing layers with a larger MLP that directly projects node coordinates into the embedding space, without explicitly modeling neighborhood interactions. The first row of Table 3 reports the resulting performance degradation, highlighting the importance of local neighbor awareness at the node embedding stage.

Input Fusion Strategy. Unlike baseline neural operators, GMNO encodes each physical input modality separately and then fuses these modality specific embeddings with mesh embeddings of the corresponding dimensionality through concatenation followed by projection. The baseline

Table 3: Results of small scale ablation studies on the influences of GMNO modules. All results are reported in units of $\times 10^{-2}$.

Ablations	*Trans UU*	*Induc EI Gap*	*Induc Lg*	*EM C Wire*
No Message Passing	6.2	4.6	1.5	4.0
Add Fusion	4.8	**3.9**	**1.0**	2.8
Transolver backbone	6.1	4.2	**1.0**	**2.4**
Zero-padding linear attention	5.6	**4.0**	**1.1**	2.6
Standard GMNO	**3.8**	**3.8**	**1.0**	**2.3**

comparison in Table 1 already suggests that different physical modalities should be processed independently before fusion, rather than being naively concatenated at the input stage. The second row of Table 3 further evaluates this design by comparing our fusion strategy against an alternative based on summation followed by projection. The results show a consistent performance drop, especially on the most challenging dataset, *Trans UU*. This suggests that although both strategies produce embeddings of the same dimension after projection, summation makes it easier for the model to entangle or confuse information from different input modalities.

Backbone selection. Besides the embedding layers, we also conduct ablation tests on alternative model backbones (token mixers) including transolver backbone which learns a weighting strategy for token compression, and vanilla GNOT backbone which performs zero padding. Notice that the ViT style backbone of GAOT requires regular latent grid which is not applicable here. The third and fourth rows of Table 3 show the performance of alternative backbone options. On simpler datasets such as *Induc Lg*, all backbones show similar performance, while on challenging datasets such as *Trans UU*, GMNO's linear attention backbone clearly outperforms vanilla linear attention or Transolver. Although linear attention is not the most efficient token processing technique in operator learning, we believe it's most expressive for industrial-standard datasets such as MaxwellBench when input fields are embedded on highly irregular meshes. On the other hand, the zero-padding strategy adopted in vanilla linear attention can generate ambiguity and waste GPU memory when sequence lengths of training data vary by orders of magnitudes.

5 Conclusion and Limitation

In this paper, we propose GMNO as a scalable neural operator architecture for generalizing across industrial-standard simulation data from diverse scenarios governed by the same underlying PDEs. To evaluate its effectiveness and scalability, we introduce MaxwellBench, the first industrial-standard FEM benchmark comprising 14 datasets from distinct electromagnetism applications, covering a wide range of heterogeneity and variation far beyond simple parameter changes. Baseline comparisons with existing SOTA neural operators demonstrate the effectiveness of GMNO's mesh embedding and multi-modal input fusion designs on industrial-standard datasets. We further pretrain a 0.54B GMNO on 10 MaxwellBench datasets to study its scaling behavior. The results show that transferring the pretrained GMNO to scarce-data OOD datasets yields nearly an order of magnitude lower error than training from scratch. To the best of our knowledge, this is the first work to introduce a versatile industrial-standard benchmark and demonstrate both the scaling potential and the OOD generalization capability of neural operators on industrial-standard datasets.

At the same time, we acknowledge several limitations of this study. First, MaxwellBench does not yet include extremely large meshes with millions of elements, due to limited computational resources. Second, while existing token compression techniques such as those used in Transolver or GAOT tend to lose accuracy on highly heterogeneous industrial-standard datasets, the current GMNO backbone is less efficient when mesh sizes become very large. Third, although we include a broad range of electromagnetism scenarios in MaxwellBench, some input modalities and physical phenomena are inevitably omitted, such as magnetic hysteresis and skin effects. Despite these limitations, we believe that GMNO and MaxwellBench provide a strong foundation for a new scaling paradigm in operator learning that more faithfully reflects industrial simulation practice.

AI USE STATEMENT

Generative AI tools assisted with language editing, manuscript revision suggestions, and LaTeX formatting.

REPRODUCIBILITY STATEMENT

We provide anonymized source code and representative MaxwellBench data samples in the supplementary material. The dataset representation and simulation settings are described in Appendix B, and the training settings, data processing, and hyperparameters are provided in Appendix C.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Mussad Alzahrani, Hasan Baig, Katie Shanks, and Tapas Mallick. Estimation of the performance limits of a concentrator solar cell coupled with a micro heat sink based on a finite element simulation. *Applied Thermal Engineering*, 176:115315, 2020.

Yannick Augenstein, Taavi Repan, and Carsten Rockstuhl. Neural operator-based surrogate solver for free-form electromagnetic inverse design. *Acs Photonics*, 10(5):1547–1557, 2023.

Yemna Bensalem and Mohamed Naceur Abdelkrim. Modeling and simulation of induction motor based on finite element analysis. *International Journal of Power Electronics and Drive Systems*, 7 (4), 2016.

Boris Bonev, Thorsten Kurth, Ankur Mahesh, Mauro Bisson, Jean Kossaifi, Karthik Kashinath, Anima Anandkumar, William D Collins, Michael S Pritchard, and Alexander Keller. Fourcastnet 3: A geometric approach to probabilistic machine-learning weather forecasting at scale. *arXiv preprint arXiv:2507.12144*, 2025.

Florent Bonnet, Jocelyn Mazari, Paola Cinnella, and Patrick Gallinari. Airfrans: High fidelity computational fluid dynamics dataset for approximating reynolds-averaged navier–stokes solutions. *Advances in Neural Information Processing Systems*, 35:23463–23478, 2022.

Shuhao Cao. Choose a transformer: Fourier or galerkin. *Advances in neural information processing systems*, 34:24924–24940, 2021.

Zhonghao Chen, Haochen Mu, Zhao Zhang, Lei Yuan, Donghong Ding, Hongtao Zhu, Ninshu Ma, and Zengxi Pan. Graph neural operator: A deeponet-based framework for learning thermo-mechanical distortion in metallic additive manufacturing. *Additive Manufacturing*, pp. 105063, 2025.

Gouri Dhatt, Emmanuel Lefrançois, and Gilbert Touzot. *Finite element method*. John Wiley & Sons, 2012.

E Dick. Introduction to finite element methods in computational fluid dynamics. In *Computational fluid dynamics*, pp. 235–274. Springer, 2009.

C Armando Duarte, Ivo Babuška, and J Tinsley Oden. Generalized finite element methods for three-dimensional structural mechanics problems. *Computers & Structures*, 77(2):215–232, 2000.

Mohamed Elrefaie, Florin Morar, Angela Dai, and Faez Ahmed. Drivaernet++: A large-scale multi-modal car dataset with computational fluid dynamics simulations and deep learning benchmarks. *Advances in Neural Information Processing Systems*, 37:499–536, 2024.

Robert Eymard, Thierry Gallouët, and Raphaèle Herbin. Finite volume methods. *Handbook of numerical analysis*, 7:713–1018, 2000.

Somdatta Goswami, Katiana Kontolati, Michael D Shields, and George Em Karniadakis. Deep transfer operator learning for partial differential equations under conditional shift. *Nature Machine Intelligence*, 4(12):1155–1164, 2022.

William S Hall. Boundary element method. In *The boundary element method*, pp. 61–83. Springer, 1994.

Zhongkai Hao, Zhengyi Wang, Hang Su, Chengyang Ying, Yinpeng Dong, Songming Liu, Ze Cheng, Jian Song, and Jun Zhu. Gnot: A general neural operator transformer for operator learning. In *International conference on machine learning*, pp. 12556–12569. PMLR, 2023.

Zhongkai Hao, Chang Su, Songming Liu, Julius Berner, Chengyang Ying, Hang Su, Anima Anandkumar, Jian Song, and Jun Zhu. Dpot: Auto-regressive denoising operator transformer for large-scale pde pre-training. *arXiv preprint arXiv:2403.03542*, 2024.

Maximilian Herde, Bogdan Raonic, Tobias Rohner, Roger Käppeli, Roberto Molinaro, Emmanuel De Bezenac, and Siddhartha Mishra. Poseidon: Efficient foundation models for pdes. *Advances in Neural Information Processing Systems*, 37:72525–72624, 2024.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589, 2021.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4015–4026, 2023.

Seid Koric, Asha Viswantah, Diab W Abueidda, Nahil A Sobh, and Kamran Khan. Deep learning operator network for plastic deformation with variable loads and material properties. *Engineering with Computers*, 40(2):917–929, 2024.

Zijie Li, Kazem Meidani, and Amir Barati Farimani. Transformer for partial differential equations' operator learning. *arXiv preprint arXiv:2205.13671*, 2022.

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

Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, Anima Anandkumar, et al. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2020b.

Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. Fourier neural operator with learned deformations for pdes on general geometries. *Journal of Machine Learning Research*, 24(388):1–26, 2023a.

Zongyi Li, Nikola Kovachki, Chris Choy, Boyi Li, Jean Kossaifi, Shourya Otta, Mohammad Amin Nabian, Maximilian Stadler, Christian Hundt, Kamyar Azizzadenesheli, et al. Geometry-informed neural operator for large-scale 3d pdes. *Advances in Neural Information Processing Systems*, 36: 35836–35854, 2023b.

Huafeng Liu, Yiran Fu, Jingyue Shi, Liping Jing, and Jian Yu. Learning ood robust neural operator with risk-averse stochastic optimization. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 1799–1810, 2025.

Songming Liu, Zhongkai Hao, Chengyang Ying, Hang Su, Ze Cheng, and Jun Zhu. Nuno: A general framework for learning parametric pdes with non-uniform data. In *International Conference on Machine Learning*, pp. 21658–21671. PMLR, 2023.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021.

Huakun Luo, Haixu Wu, Hang Zhou, Lanxiang Xing, Yichen Di, Jianmin Wang, and Mingsheng Long. Transolver++: An accurate neural solver for pdes on million-scale geometries. *arXiv preprint arXiv:2502.02414*, 2025.

Michael McCabe, Bruno Régaldo-Saint Blancard, Liam Parker, Ruben Ohana, Miles Cranmer, Alberto Bietti, Michael Eickenberg, Siavash Golkar, Geraud Krawezik, Francois Lanusse, et al. Multiple physics pretraining for spatiotemporal surrogate models. *Advances in Neural Information Processing Systems*, 37:119301–119335, 2024.

Amil Merchant, Simon Batzner, Samuel S Schoenholz, Muratahan Aykol, Gowoon Cheon, and Ekin Dogus Cubuk. Scaling deep learning for materials discovery. *Nature*, 624(7990):80–85, 2023.

Matthias Müller, David Charypar, and Markus Gross. Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 154–159, 2003.

COMSOL Multiphysics. Introduction to comsol multiphysics®. *COMSOL Multiphysics, Burlington, MA, accessed Feb*, 9(2018):32, 1998.

Shashank Subramanian, Peter Harrington, Kurt Keutzer, Wahid Bhimji, Dmitriy Morozov, Michael W Mahoney, and Amir Gholami. Towards foundation models for scientific machine learning: Characterizing scaling and transfer behavior. *Advances in Neural Information Processing Systems*, 36: 71242–71262, 2023.

Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Daniel MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. Pdebench: An extensive benchmark for scientific machine learning. *Advances in neural information processing systems*, 35:1596–1611, 2022.

Lonny L Thompson. A review of finite-element methods for time-harmonic acoustics. *The Journal of the Acoustical Society of America*, 119(3):1315–1330, 2006.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Tian Wang and Chuang Wang. Latent neural operator for solving forward and inverse pde problems. *Advances in Neural Information Processing Systems*, 37:33085–33107, 2024.

Shizheng Wen, Arsh Kumbhat, Levi Lingsch, Sepehr Mousavi, Yizhou Zhao, Praveen Chandrashekar, and Siddhartha Mishra. Geometry aware operator transformer as an efficient and accurate neural surrogate for pdes on arbitrary domains. *arXiv preprint arXiv:2505.18781*, 2025.

Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. Transolver: A fast transformer solver for pdes on general geometries. In *International Conference on Machine Learning*, pp. 53681–53705. PMLR, 2024.

Pei-bai Zhou. Finite difference method. In *Numerical analysis of electromagnetic fields*, pp. 63–94. Springer, 1993.

A PRELIMINARIES

In this section, we introduce the preliminaries for understanding the motivation, representation format, underlying physics, scope, statistics, and other characteristics of our electromagnetism dataset.

A.1 CHALLENGES

Despite the extensive research in operator learning for accelerating PDE solving, the benchmark datasets are oftentimes considered toy problems. This "toyness" doesn't come from the nonlinearity of PDE, but the extreme simplification on causality, geometry, and physical field representation, which hinges the practical usage of neural operators. To mitigate this gap, we attempt to provide a collection of electromagnetism datasets with maximum complexity to best approximate real industrial simulation files, which includes the following aspects:

Mesh representation rather than point clouds. In industrial practice, physical simulations are performed on computation domains discretized by meshes. Yet much of the operator learning literature represents fields on point clouds, often relying on interpolation. While this simplifies data handling, it blurs the distinction between mesh entities of different topological dimensions (faces, edges, and nodes) and discards geometric measures (e.g., area/volume) and fine-grained boundary details that are critical in complex problem settings. Instead, we explicitly associate each physical quantity with its corresponding mesh entity and construct datasets that more faithfully reflect industrial simulation workflows.

Extreme geometry irregularity. Our electromagnetics datasets exhibit geometric features spanning multiple length scales, mirroring the challenges encountered in industrial settings. For example, there exist large air and iron regions on the order of $10 \sim 10^2$ mm, copper winding bundles at $1 \sim 10$ mm, and also tiny iron core gaps as small as $10^{-2} \sim 10^{-1}$ mm. In addition, each data has a different domain size and is discretized with Comsol's adaptive meshing (dataset statistics in Table. 4). Together, these factors produce extremely irregular meshes with a wide range of element sizes, making many point cloud based processing techniques commonly used in operator learning literature inadequate.

Multi-modal input fields. Much of the operator learning literature represents inputs as concatenated feature channels on point clouds, which rarely reflects industrial practice. In real simulation pipelines, input files are hierarchical and multi-modal, with attributes defined on mesh entities of different topological dimensions (faces, edges, and nodes). For instance, material properties are oftentimes defined by nonlinear curves (or even hysteresis) rather than a few parameters. Consequently, flattening all inputs into concatenated channels on point samples obscures this structure and is not representative of industrial simulation scenarios.

Heterogeneous datasets and value ranges. Industrial simulation files vary substantially in size due to differences in mesh resolution, which complicates large scale operator training. Moreover, the underlying governing PDE can also differ across data based on coordinate systems and analysis types. Besides the dataset structural heterogeneity, the target fields can span wide dynamic ranges: variations at small magnitudes ($10^{-3} \sim 10^{-2}$) can be just as important as variations at larger magnitudes ($1 \sim 10$) in practical applications. Under such conditions, naive normalization can suppress informative low amplitude structure, and thus failing to preserve meaningful variations when target field is on the order of $10^{-3} \sim 10^{-2}$.

A.2 NOTATION

Ω	The entire computation domain.
$\partial\Omega$	The boundary of computation domain.
Ω_k	The kth subdomain of Ω, with $\Omega = \cup_{k=1}^{K}\Omega_k$.
Γ	All subdomain boundaries and interfaces, with $\Gamma = \cup_{k=1}^{K}\partial\Omega_k$.
$\mathbf{n}$	Unit normal vector.
$\mathbf{B}$	Magnetic flux density.
$\mathbf{H}$	Magnetic field.
$\mathbf{E}$	Electric field.
$\mathbf{D}$	Electric flux density.
$\mathbf{J}$	Total current density.
$\mathbf{J}_s$	Prescribed current density.
ρ	Charge density.
$\mathcal{M}$	An abstract variable that collects all material parameters.
$\mathcal{S}$	An abstract variable that collects all current density parameters.
$\mathcal{P}$	An abstract variable that collects all boundary condition parameters.
$\mathcal{V}_\Omega$	The 2D input function space.
$\mathcal{W}_\Gamma$	The 1D input function space.
$\mathcal{H}_\Omega$	The 2D solution function space.
μ	Permeability for linear magnetic material.
$\mathcal{B}(\cdot)$	BH curve for nonlinear magnetic material.
ε	Permittivity.
σ	Conductivity.
$\tilde{J}_s$	Magnitude of the prescribed current density.
φ	Phase angle of the prescribed current density.
$\|\| \cdot \|\|$	Magnitude of a complex value, or absolute of a real value.
$Real(\cdot)$	Real part of a complex value.
$Imag(\cdot)$	Imaginary part of a complex value.
$Type(\cdot)$	Type/Category of a discrete variable.
j	Unit imaginary number: $\sqrt{-1}$.
f	Frequency.
ω	Angular frequency.
ξ	Global parameters of a simulation.
N_n	Total number of nodes in the mesh.
N_e	Total number of 2D triangular elements.
N_Γ	Total number of 1D line elements.
$\mathbf{X}$	Node coordinate matrix.
$\mathbf{X}^+$	Augmented node coordinate matrix.
$\mathbf{C}^T$	Connectivity matrix for all 2D triangular elements in the mesh.
A	Area of a triangular element.
$\mathbf{C}^\Gamma$	Connectivity matrix for all 1D interface elements in the mesh.
L	Length of a line element.
$\mathbf{Node}^{emb}$	Node embedding matrix.
$\mathbf{Tri}^{emb}$	2D triangular element embedding matrix.
$\mathbf{Lin}^{emb}$	1D line element embedding matrix.
$\mathbf{A}^{emb}$	Area embedding matrix.
$\mathbf{L}^{emb}$	Length embedding matrix.
$\mathbf{S}^{emb}$	Source embedding matrix.
$\mathbf{M}^{emb}$	Material embedding matrix.
$\mathbf{P}^{emb}$	Boundary condition embedding matrix.
ξ^{emb}	Global parameter embedding.
$(\cdot)^{token}$	Tokenized field.
$\hat{(\cdot)}$	Normalized quantity.
$\{(\cdot)_e\}_{e=1}^{N_e}$	Field variable defined on the triangular element faces.
$\{(\cdot)_l\}_{l=1}^{N_\Gamma}$	Field variable defined on the line elements.

A.3 MESH

As it is impossible to obtain analytical solutions to the complex PDE problems in industry, the input and solution physical fields are oftentimes defined on discretized meshes and solved with numerical methods (e.g. finite element method).

In this work, we consider the 2D triangular mesh discretization of Ω, which typically includes one node coordinate matrix and multiple connectivity matrices: the nodes are stored in $\mathbf{X} \in \mathbb{R}^{N_n \times 2}$, written either as $\mathbf{X}_{i,:} = (x_i, y_i)$ in Cartesian coordinates or as $\mathbf{X}_{i,:} = (r_i, z_i)$ in cylindrical coordinates; 2D triangular elements are given by the connectivity matrix $\mathbf{C}^T \in \{1, \dots, N_n\}^{N_e \times 3}$ whose row $\mathbf{C}^T_{e,:} = (n_{e1}, n_{e2}, n_{e3})$ lists the global node indices of triangle e, while 1D line elements on subdomain boundaries and interfaces are given by $\mathbf{C}^\Gamma \in \{1, \dots, N_n\}^{N_\Gamma \times 2}$ (this includes domain boundaries $\partial\Omega \subset \Gamma$), with each row containing the two endpoint node indices of the corresponding segment.

Based on such mesh discretization, we can represent 2D fields as $\{(\cdot)_e\}_{e=1}^{N_e}$ and 1D fields as $\{(\cdot)_l\}_{l=1}^{N_\Gamma}$. Notice that for 2D electromagnetism problems, most 2D fields such as material, current, and magnetic fields are defined on elements rather than nodes.

A.4 ELECTROMAGNETISM IN 2D

To achieve the best model generalization performance, we hope to establish a dataset that spans across a wide range of electromagnetism applications in industry. As a result, we include steady state results both in time domain (static) and in frequency domain. On the other hand, transient electromagnetism responses are less important in industrial simulations. Maxwell's equations in time domain takes the following form:

$$\begin{aligned} \nabla \times \mathbf{E} &= -\frac{\partial \mathbf{B}}{\partial t}, & \nabla \times \mathbf{H} &= \mathbf{J} + \frac{\partial \mathbf{D}}{\partial t}, \\ \nabla \cdot \mathbf{D} &= \rho, & \nabla \cdot \mathbf{B} &= 0. \end{aligned} \tag{6}$$

While Maxwell's equations in frequency domain takes the form of:

$$\begin{aligned} \nabla \times \mathbf{E} &= -j\omega \mathbf{B}, & \nabla \times \mathbf{H} &= \mathbf{J} + j\omega \mathbf{D}, \\ \nabla \cdot \mathbf{D} &= \rho, & \nabla \cdot \mathbf{B} &= 0. \end{aligned} \tag{7}$$

We include both the cartesian (x, y, z) and cylindrical (r, ϕ, z) coordinate systems in the datasets as common industrial practices. Due to restricted computation resources, we focus only on 2D Maxwell problems in this study, where we assume $B_z = 0$, $J_x = J_y = 0$ in 2D cartesian coordinate, and $B_\phi = 0$, $J_r = J_z = 0$ in 2D cylindrical coordinate.

The constitutive relations in industrial settings take the following form:

$$\mathbf{D} = \varepsilon \mathbf{E}, \quad \mathbf{B} = \mathcal{B}(\mathbf{H}), \quad \mathbf{J} = \sigma \mathbf{E} + \mathbf{J}_s \tag{8}$$

where materials are assumed to be isotropic with linear electrical properties and nonlinear magnetic properties. Notice that major operator learning literatures adopt linear material models $\mathcal{B}(\mathbf{H}) = \mu \mathbf{H}$ so that the input material distribution field takes the simplified form $\{\mu \parallel \varepsilon \parallel \sigma\}$ with concatenated properties. However, industrial electromagnetism simulations typically adopt nonlinear magnetic properties with the form of a B-H curve as shown in Fig. 2. Therefore, we denote the material property variable as $\mathcal{M} = \{\mathcal{B}(\cdot), \varepsilon, \sigma\}$, which should be processed by a specific property fusion block in the model rather than simple concatenation. Moreover, we assume that $\mathcal{B}(\cdot)$ varies across data.

The prescribed current excitation takes the complex form of $J_s = \tilde{J}_s e^{j(\omega t + \varphi)}$ (scalar form as current density is always out of plane), and is therefore denoted as variable $\mathcal{S} = \{\tilde{J}_s, \omega, \varphi\}$.

To construct a more general data format, we define boundary condition on all 1D line entities Γ, including the computation domain boundary $\partial\Omega \subset \Gamma$ and subdomain interfaces. The dataset includes three types of boundary conditions: magnetic insulation $\mathbf{n} \cdot \mathbf{B} = 0$, axial symmetry $B_r = 0$, and the null condition. Magnetic insulation condition is only applied on $\partial\Omega$; the axial symmetry is applied on the symmetry axis $\partial\Omega \cap \{r = 0\}$ when using a cylindrical coordinate system; and the null

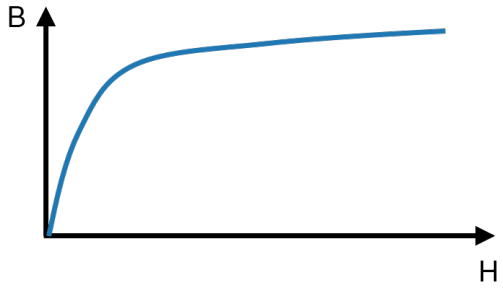

Figure 2: A nonlinear B-H curve of a typical magnetic material.

condition is for all subdomain interfaces. Accordingly, we represent the boundary condition variable as $\mathcal{P} = \{Type(\Gamma), \mathbf{n}\}$ where $\mathbf{n} = (0, 0)$ when $Type(\Gamma)$ is null.

Given the material distribution, prescribed current sources, and boundary conditions, we use Comsol FEM to solve Maxwell's equations to obtain $||B_{x/r}||$ and $||B_{y/z}||$ as the prediction target for our model. Notice that we choose the magnitude field as the prediction task rather than the complete complex magnetic, as engineers only need magnitudes to compute downstream power loss fields. For static scenarios, the magnitude reduces to the absolute value.

B MaxwellBench

B.1 Dataset Format

Different from existing open-source PDE datasets, our datasets adopts the structure in Fig. 3 to approximate industrial simulation files.

To numerically solve and learn the 2D Maxwell's equations, all physical field variables are discretized via the mesh representation introduced in section A.3. We use a multi-modal physics encoder to process all input information including global parameters, mesh information, material information, source information, and boundary information. The processed input information is then organized into a 2D input field $\mathcal{V}_\Omega$ approximated as $\{\mathbf{X}(\mathbf{C}^T_{e,:}), A_e, \mathcal{M}_e, \mathcal{S}_e\}_{e=1}^{N_e}$ and a 1D input field $\mathcal{W}_\Gamma$ approximated as $\{\mathbf{X}(\mathbf{C}^\Gamma_{l,:}), L_l, \mathcal{M}_{l,left}, \mathcal{M}_{l,right}, \mathcal{P}_l\}_{l=1}^{N_\Gamma}$. Where $\mathcal{M}_{l,left}$ and $\mathcal{M}_{l,right}$ denote the material assignment on two sides of line element l. The target PDE solution $\mathcal{H}_\Omega$ is approximated as $\{||B_{x/r}||_e, ||B_{y/z}||_e\}_{e=1}^{N_e}$. The model backbone then learns to approximate the following operator task:

$$\mathcal{H}_\Omega = \mathcal{G}(\mathcal{V}_\Omega, \mathcal{W}_\Gamma, \xi) \tag{9}$$

We also summarize some statistical indicators of the datasets as shown in Table. 4, where we can clearly observe the mesh irregularity, data structure heterogeneity, and wide dynamic value range.

- Data file
  - Global parameters
    - Analysis type (Static/Frequency domain)
    - Coordinate system
  - Mesh information
    - Node coordinate matrix $\mathbf{X}$ of shape $N_n \times 2$
    - 2D triangular element connectivity $\mathbf{C}^T$ of shape $N_e \times 3$
    - 1D line element connectivity $\mathbf{C}^\Gamma$ of shape $N_\Gamma \times 2$
  - Material information
    - Material 0
      - BH curve $\mathcal{B}(\cdot)$ of shape 10×2
      - Permittivity ε, scalar
      - Conductivity σ, scalar
      - Index array for triangular elements with material 0
      - Index array for line elements surrounding material 0
    - Material 1
      - ...
    - ...
  - Source information
    - Source 0
      - Magnitude of prescribed current density $\tilde{J}_s$, scalar
      - Frequency of current excitation f, scalar
      - Phase angle of prescribed current density φ, scalar
      - Index array for triangular elements with source 0
    - Source 1
      - ...
    - ...
  - Boundary information
    - Boundary 0
      - Boundary condition type (Null/Magnetic insulation/Symmetry axis)
      - Boundary normal vector $\mathbf{n}$
    - Boundary 1
      - ...
    - ...
  - Magnetic field information
    - Real component of magnetic flux density field in the first dimension $\{Real(B_{x/r})_e\}_{e=1}^{N_e}$ of shape $N_e \times 1$
    - Imaginary component of magnetic flux density field in the first dimension $\{Imag(B_{x/r})_e\}_{e=1}^{N_e}$ of shape $N_e \times 1$
    - Real component of magnetic flux density field in the second dimension $\{Real(B_{y/z})_e\}_{e=1}^{N_e}$ of shape $N_e \times 1$
    - Imaginary component of magnetic flux density field in the second dimension $\{Imag(B_{y/z})_e\}_{e=1}^{N_e}$ of shape $N_e \times 1$

Figure 3: Illustrative layout of a single data file.

Table 4: Statistical indicators of all the pretraining and unseen (OOD) datasets.

	Datasets	Number	N_e range	Element size (mm^2)	Mean $\|\|B\|\|$	Std $\|\|B\|\|$
Pretrained	*Trans UU*	11000	15 k $\sim$ 60 k	$7.1e{-7} \sim 3.0e2$	$6.1e{-2}$	$1.4e{-1}$
	Trans PQ	11000	7 k $\sim$ 56 k	$7.9e{-8} \sim 1.6e2$	$8.5e{-2}$	$2.6e{-1}$
	Induc UU	11000	15 k $\sim$ 51 k	$1.7e{-7} \sim 3.0e2$	$2.4e{-1}$	$4.8e{-1}$
	Induc I Gap	11000	3 k $\sim$ 39 k	$1.6e{-5} \sim 3.6e1$	$1.4e{-2}$	$1.4e{-2}$
	Induc EI Gap	11000	5 k $\sim$ 36 k	$4.3e{-7} \sim 3.5e1$	$9.1e{-2}$	$2.8e{-1}$
	Induc EE	11000	3 k $\sim$ 43 k	$1.4e{-3} \sim 3.6e1$	$4.1e{-2}$	$1.1e{-1}$
	Induc Sm Gap	11000	5 k $\sim$ 68 k	$2.3e{-8} \sim 2.5e1$	$2.0e{-1}$	$5.0e{-1}$
	Induc Lg	11000	3 k $\sim$ 38 k	$6.3e{-3} \sim 8.6e2$	$8.5e{-1}$	$1.0e0$
	EM	11000	2 k $\sim$ 7 k	$4.5e{-7} \sim 9.6e1$	$2.4e{-1}$	$4.5e{-1}$
	EM C Wire	11000	5 k $\sim$ 61 k	$2.7e{-3} \sim 2.3e2$	$8.4e{-2}$	$2.1e{-1}$
OOD	*Trans L*	11000	9 k $\sim$ 38 k	$4.2e{-3} \sim 1.6e2$	$2.5e{-2}$	$2.9e{-2}$
	Induc EI	11000	5 k $\sim$ 60 k	$7.0e{-7} \sim 3.6e1$	$7.7e{-2}$	$2.1e{-1}$
	Induc Lg Gap	11000	2 k $\sim$ 30 k	$1.3e{-5} \sim 8.8e2$	$4.2e{-1}$	$6.0e{-1}$
	EM C Chunk	11000	1 k $\sim$ 4 k	$3.4e{-1} \sim 2.2e2$	$2.0e{-1}$	$3.3e{-1}$

B.2 Dataset Visualization

In this section, we provide visualization of all 14 electromagnetism datasets, which include material distribution, source distribution, material interface, and boundaries as input fields, and $\{||B_x||_e\}_{e=1}^{N_e}$, $\{||B_y||_e\}_{e=1}^{N_e}$ as output fields.

Transformer UU. Fig. 4 visualizes three data samples of a power electronic transformer application with UU shape iron core. This is a challenging dataset as it includes iron core gaps and two AC coil sets (each in two directions) applied in randomly placed copper wires. The abbreviation for this dataset is *Trans UU*.

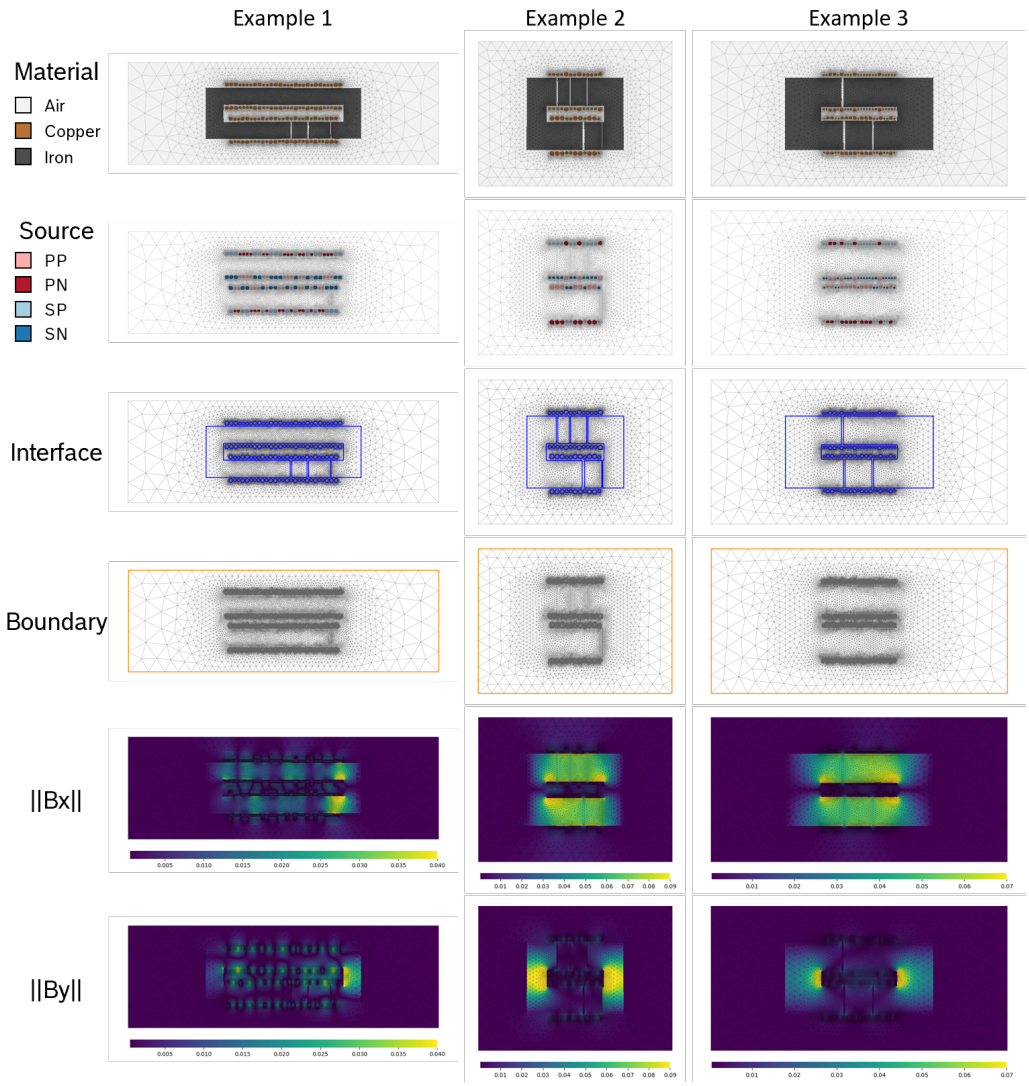

Figure 4: Visualization of three examples in the Transformer UU dataset.

Transformer PQ. Fig. 5 visualizes three data samples of a power electronic transformer application with PQ shape iron core. This dataset lies in cylindrical coordinates with two AC coil sets applied in clustered copper wires. Iron core gaps are also included. The abbreviation for this dataset is *Trans PQ*.

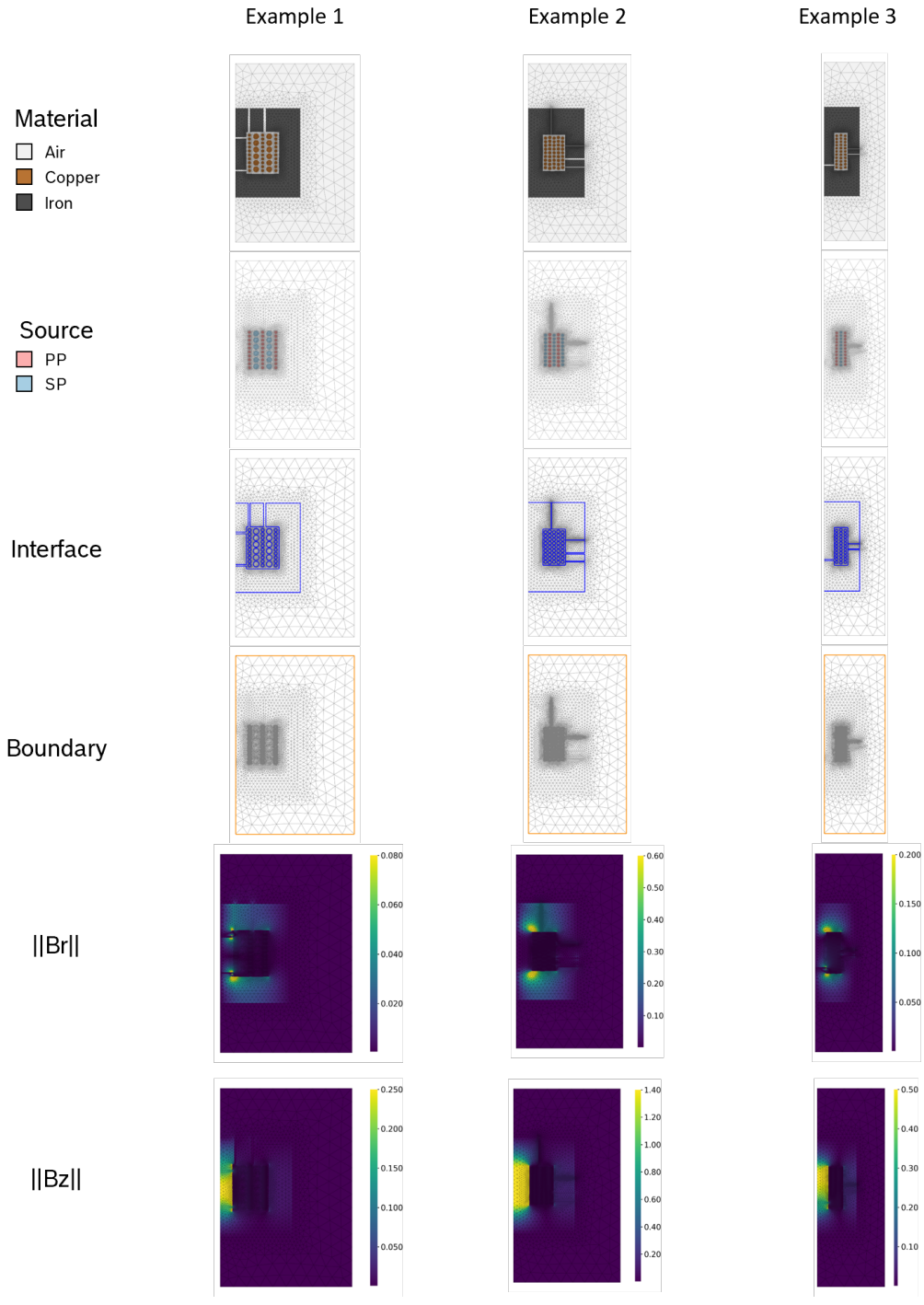

Figure 5: Visualization of three examples in the Transformer PQ dataset.

Inductor UU. Fig. 6 visualizes three data samples of a power electronic inductor application with UU shape iron core. This dataset includes iron core gaps and two directions of one AC coil set applied in randomly placed copper wires. The abbreviation for this dataset is *Induc UU*.

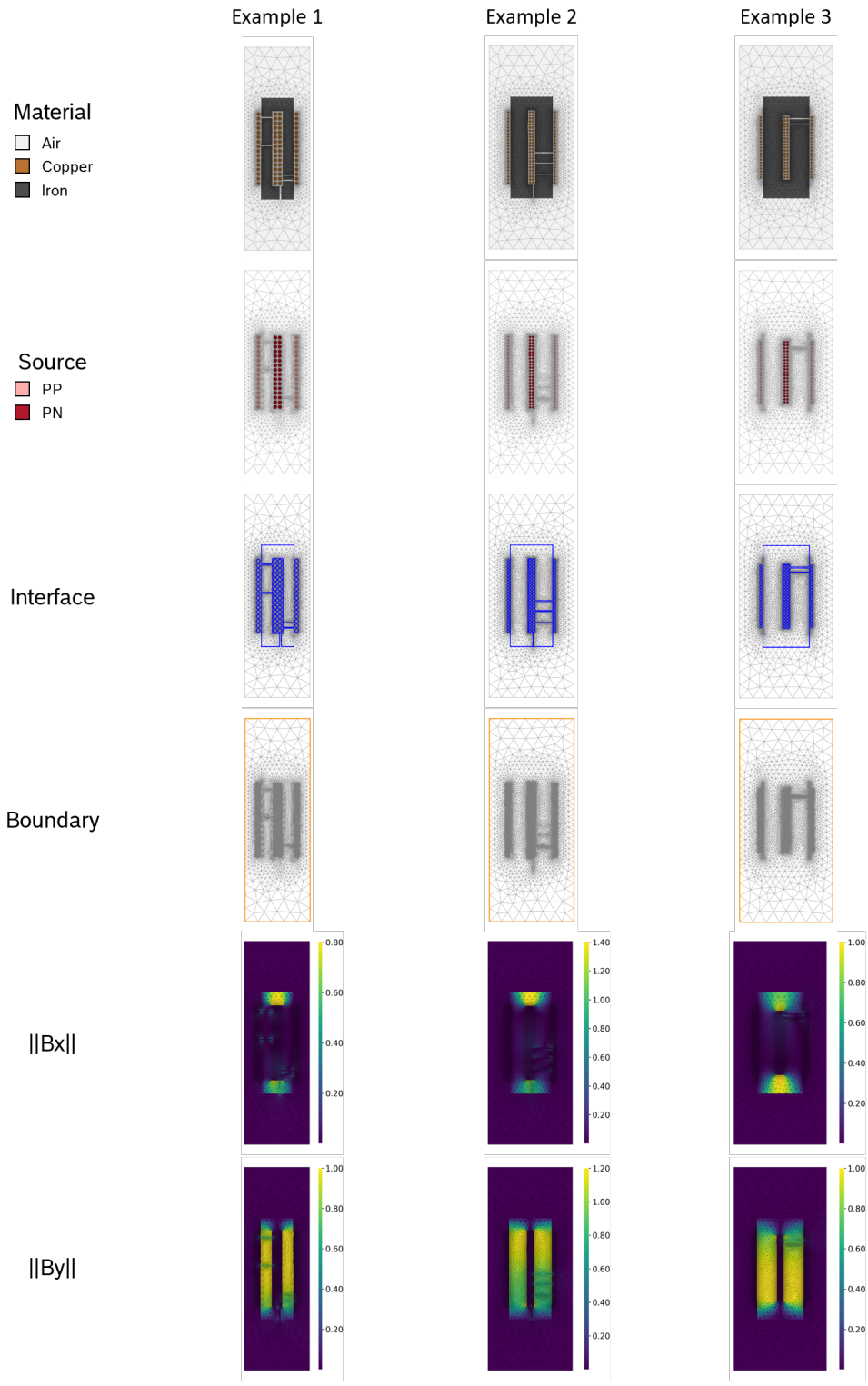

Figure 6: Visualization of three examples in the Inductor UU dataset.

Inductor I. Fig. 7 visualizes three data samples of a power electronic inductor application with I shape iron core. This is a simpler dataset that includes iron core gaps and two directions of one AC coil set applied in clustered copper wires. The abbreviation for this dataset is *Induc I*.

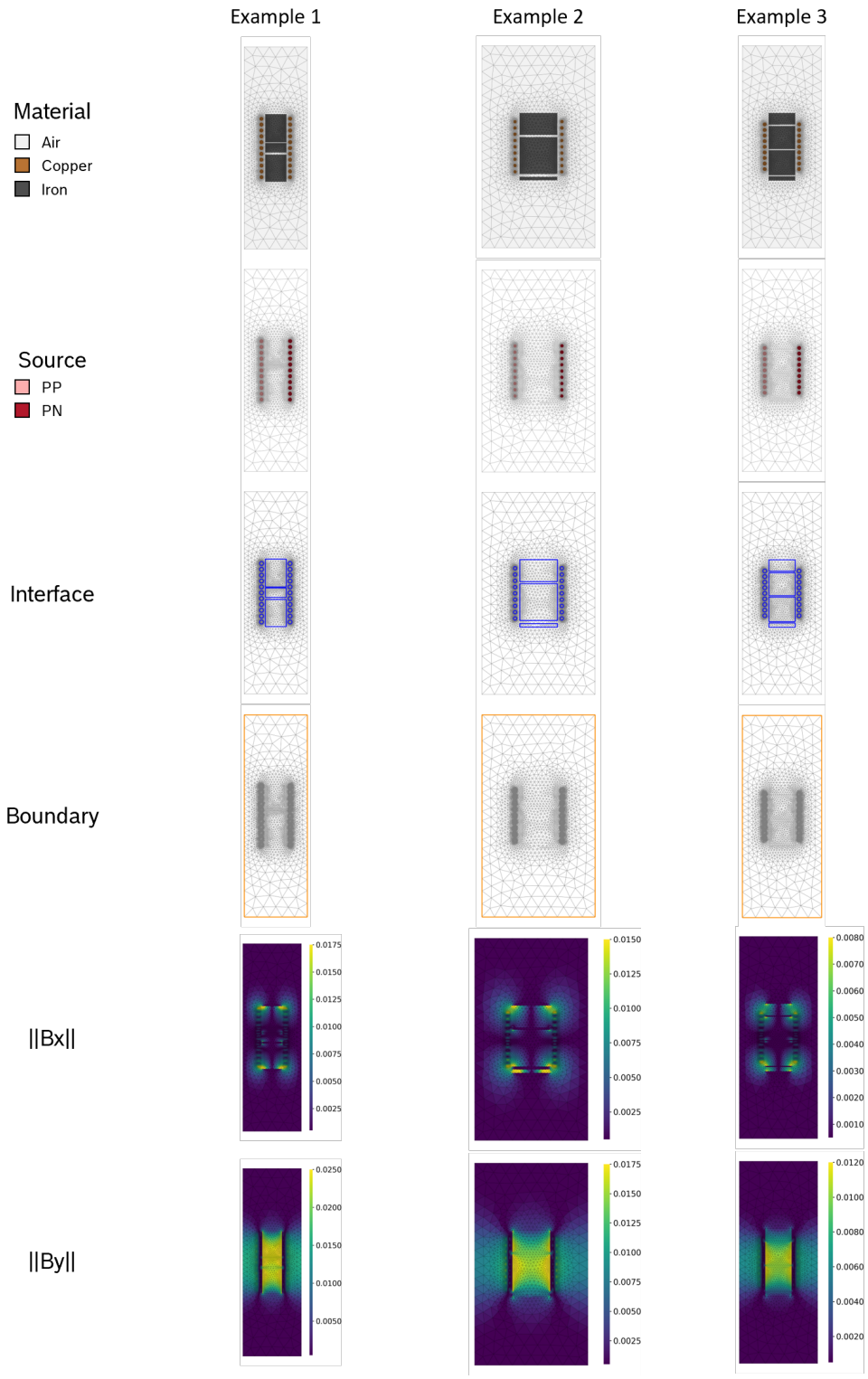

Figure 7: Visualization of three examples in the Inductor I dataset.

Inductor EI Gap. Fig. 8 visualizes three data samples of a power electronic inductor application with EI shape iron core. This dataset includes iron core gaps and two directions of one AC coil set applied in clustered copper wires. The abbreviation for this dataset is *Induc EI Gap*.

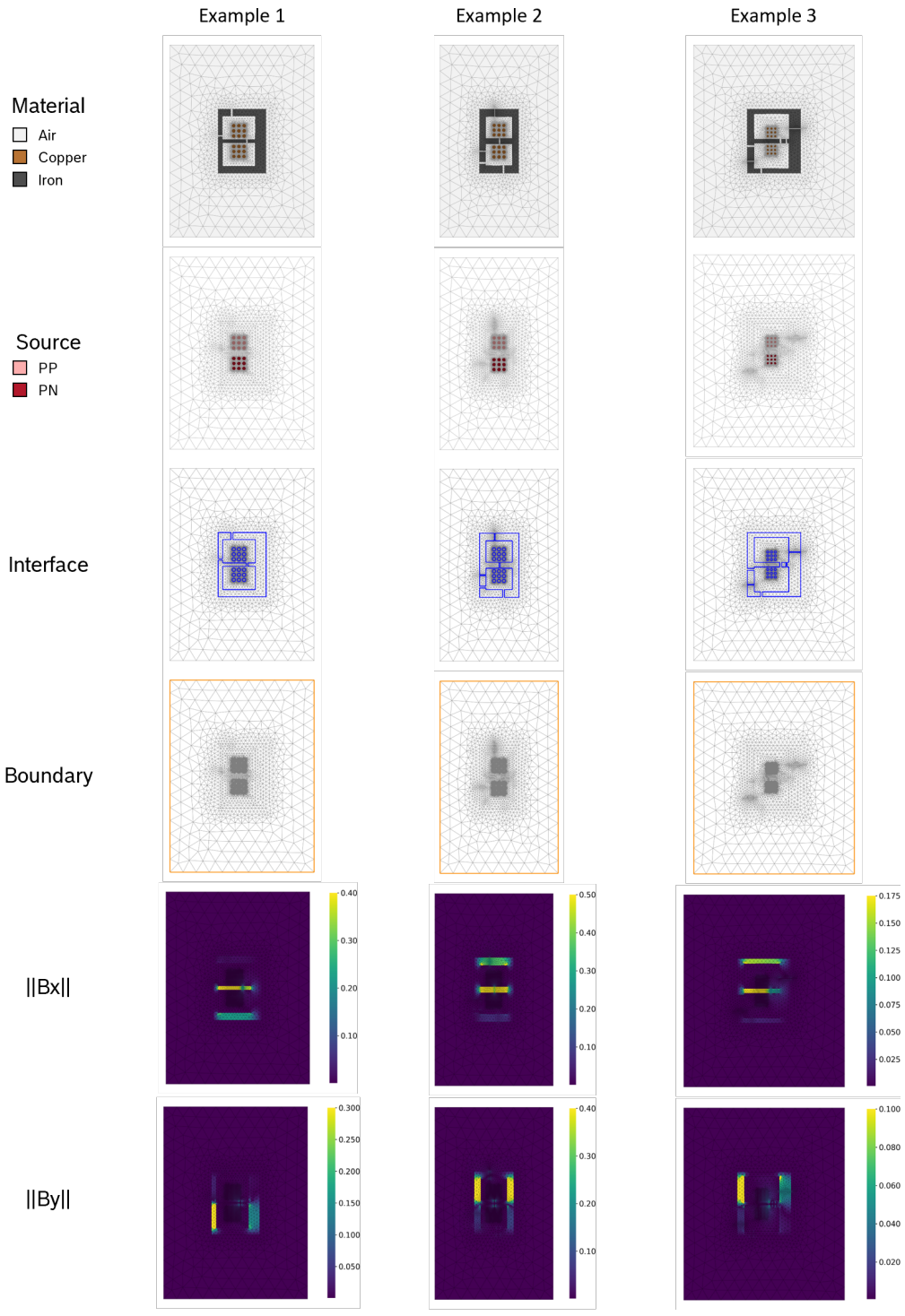

Figure 8: Visualization of three examples in the Inductor EI Gap dataset.

Inductor EE. Fig. 9 visualizes three data samples of a power electronic inductor application with EE shape iron core. This dataset has centered gaps and two directions of one AC coil set applied in clustered copper wires. The abbreviation for this dataset is *Induc EE*.

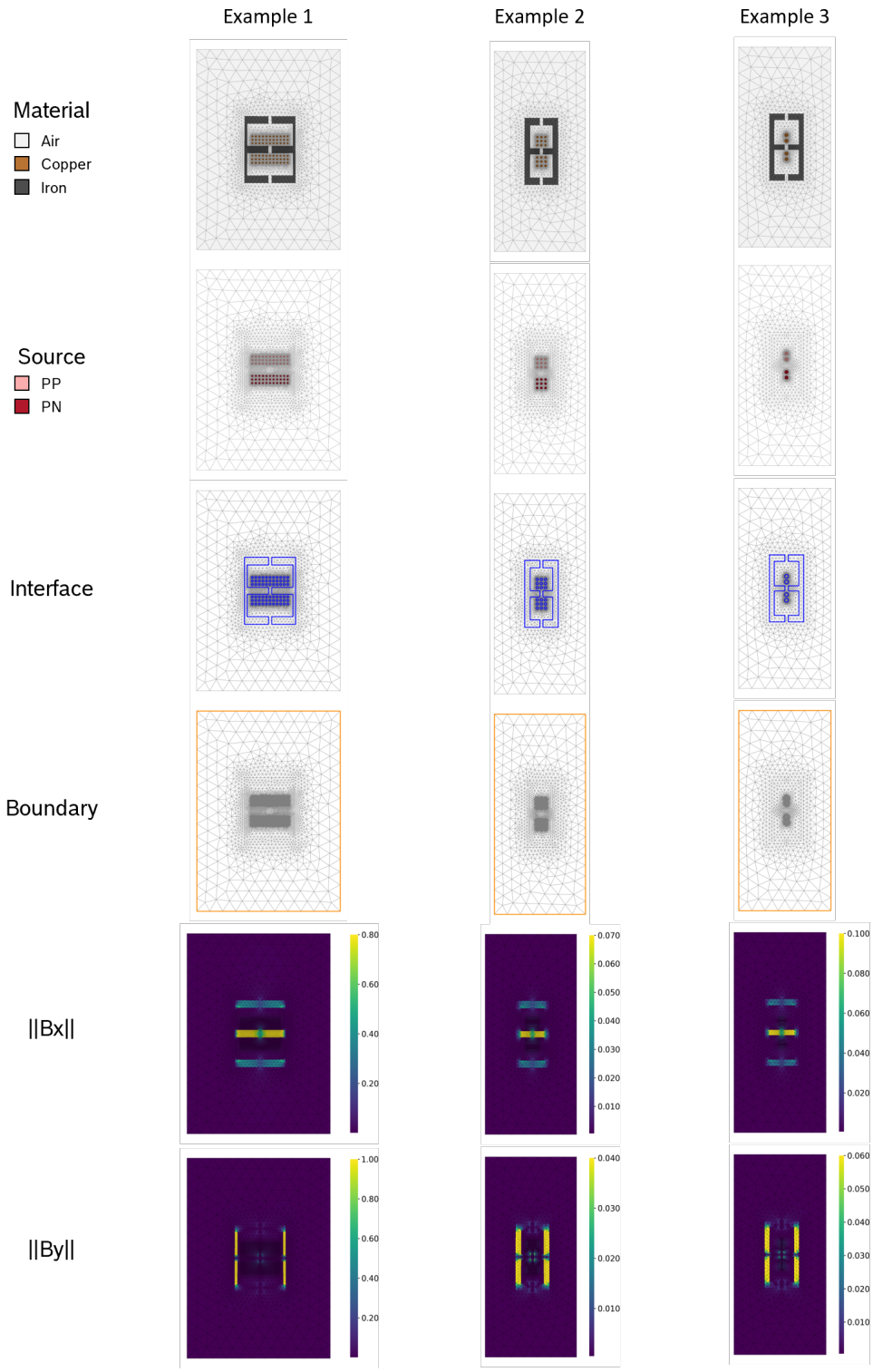

Figure 9: Visualization of three examples in the Inductor EE dataset.

Inductor Small Gap. Fig. 10 visualizes three data samples of a power electronic inductor application with small ring shape iron core. This dataset has random iron core gaps and two directions of one AC coil set applied in clustered copper wires. The abbreviation for this dataset is *Induc Sm Gap*.

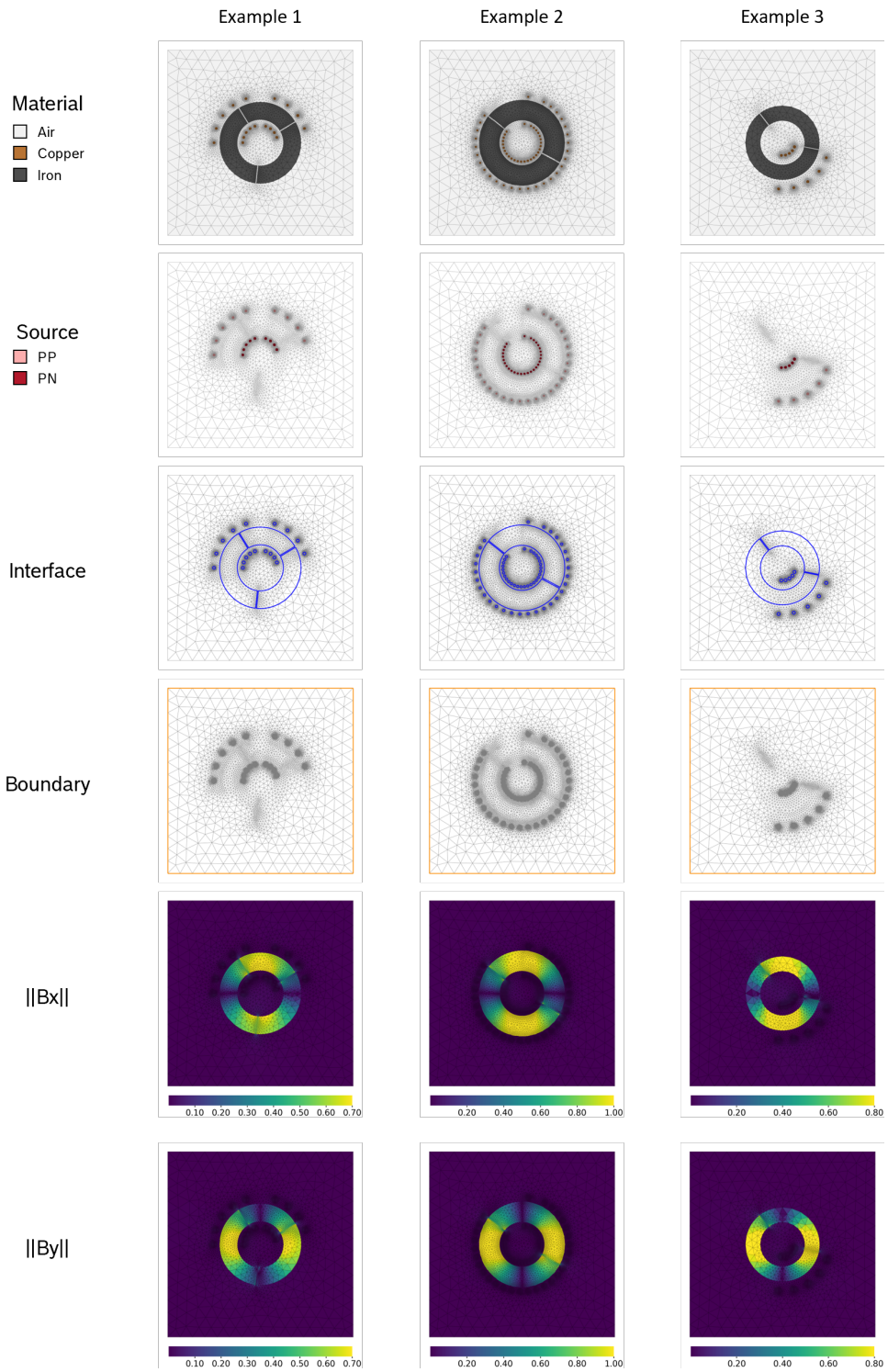

Figure 10: Visualization of three examples in the Inductor Small Gap dataset.

Inductor Large. Fig. 11 visualizes three data samples of a power electronic inductor application with large ring shape iron core. This is a relatively simpler dataset with no iron core gaps. The two directions of one AC coil set are applied in clustered copper wires. The abbreviation for this dataset is *Induc Lg*.

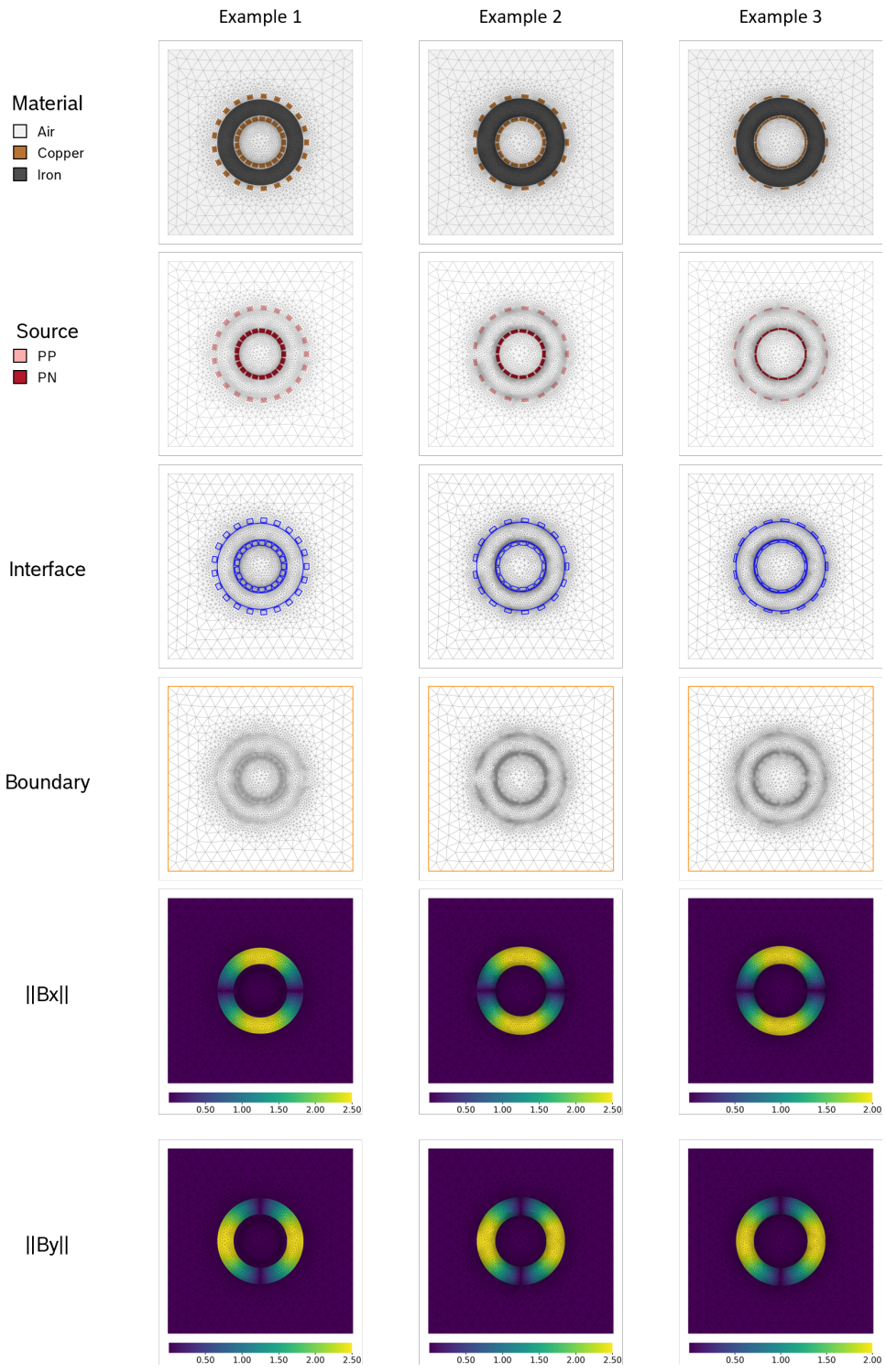

Figure 11: Visualization of three examples in the Inductor Large dataset.

Electromagnet. Fig. 12 visualizes three data samples of an electromagnet application with complex iron housing shape. This dataset lies in cylindrical coordinates with one DC coil set applied in a copper chunk. The abbreviation for this dataset is *EM*.

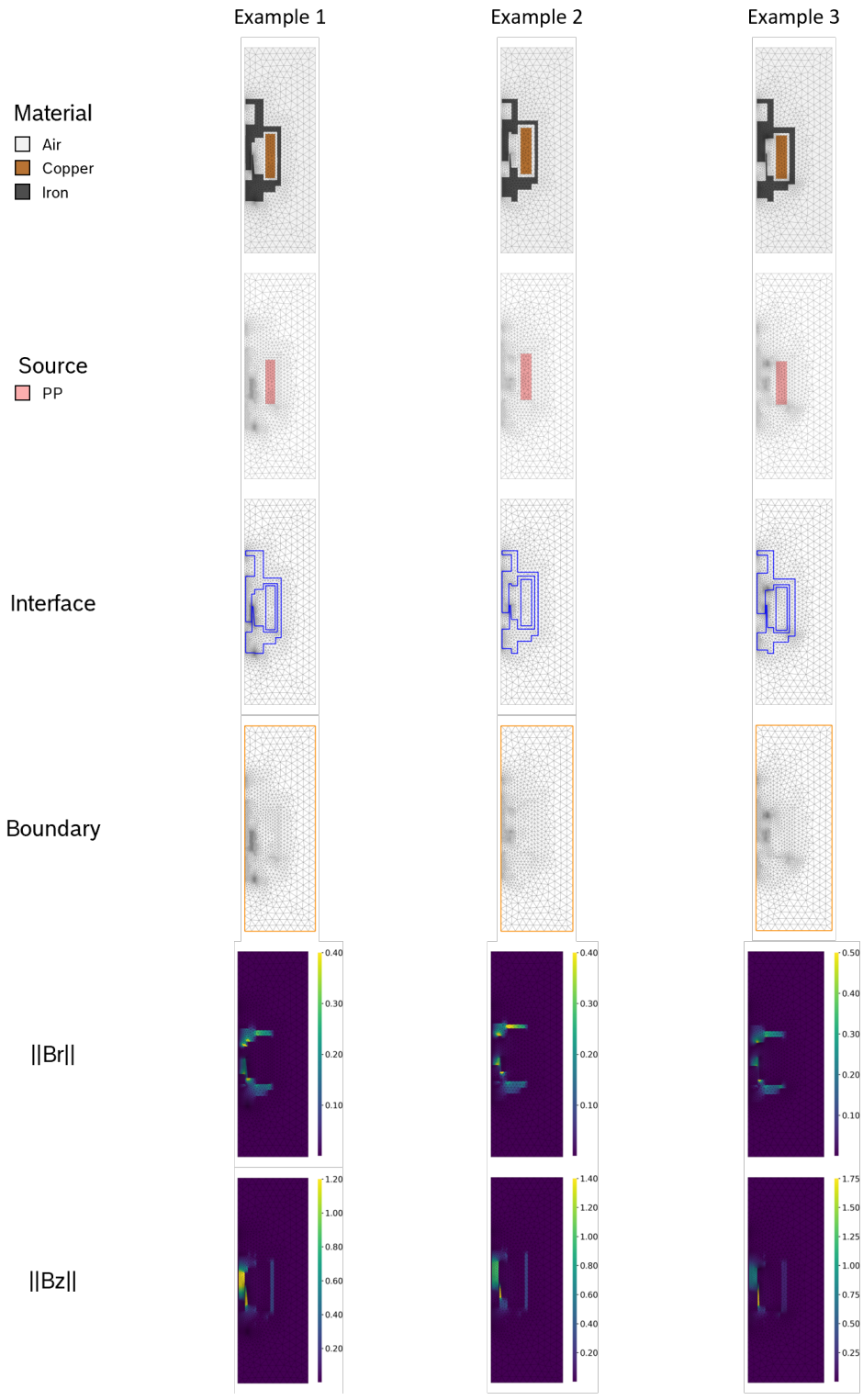

Figure 12: Visualization of three examples in the Electromagnet dataset.

Electromagnet C Wire. Fig. 13 visualizes three data samples of an electromagnet application with C shape iron housing. This dataset lies in rectangular coordinates with two directions of one DC coil set applied in clustered copper wires. The abbreviation for this dataset is *EM C Wire*.

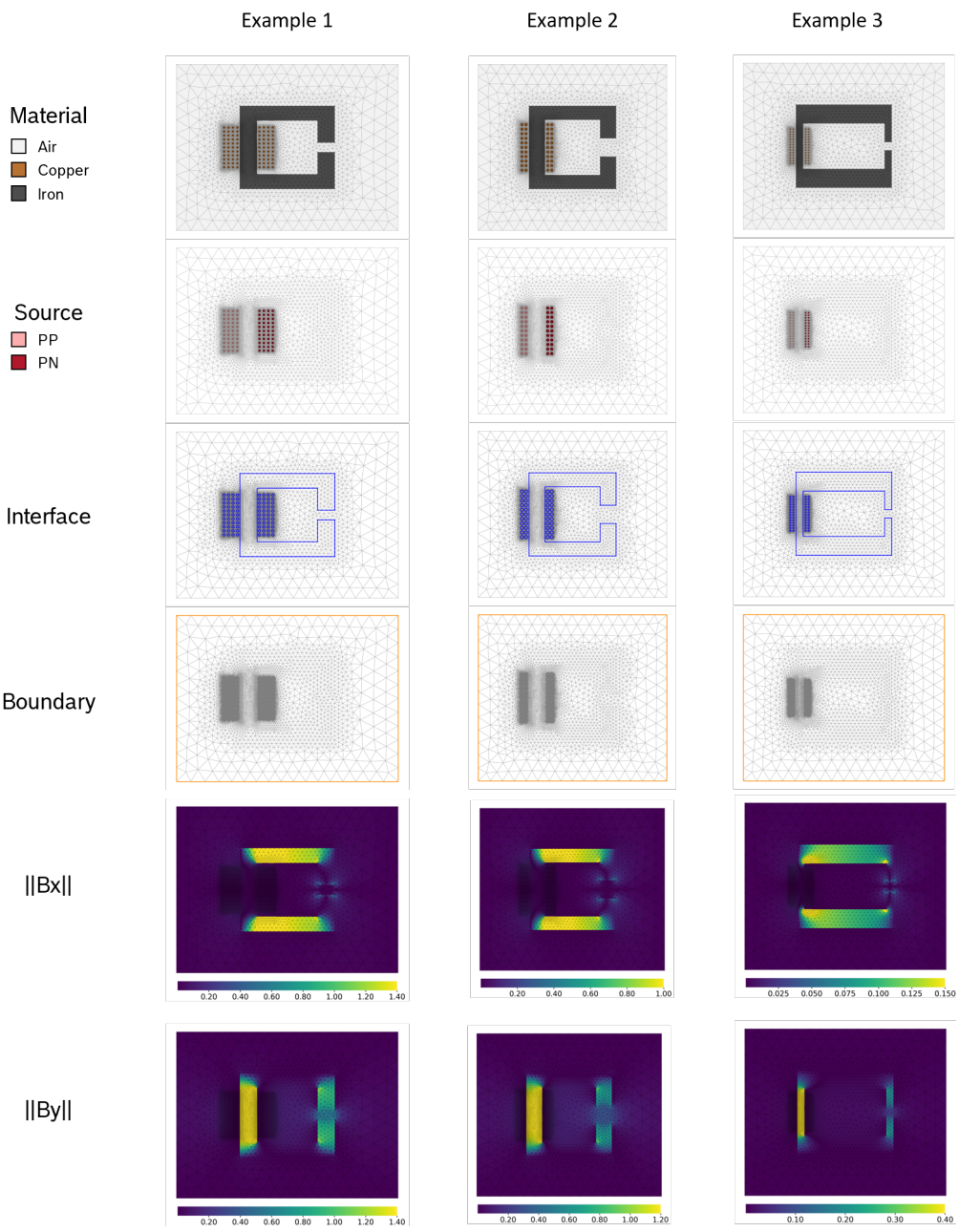

Figure 13: Visualization of three examples in the Electromagnet C Wire dataset.

Transformer L. Fig. 14 visualizes three data samples of a power electronic transformer application with L shape iron core. This dataset contains four directions of two AC coil sets applied in clustered copper wires. The abbreviation for this dataset is *Trans L*.

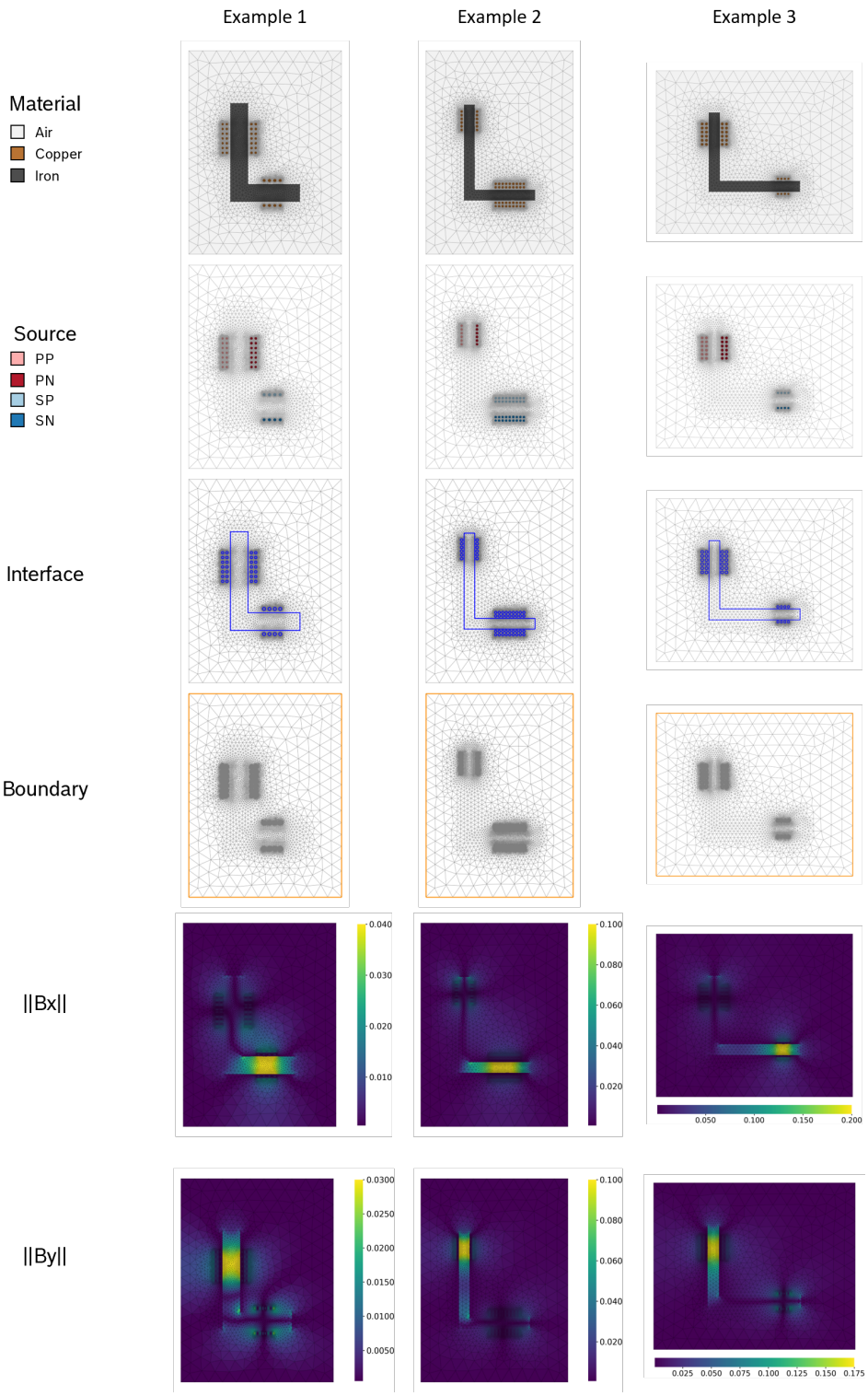

Figure 14: Visualization of three examples in the Transformer L dataset.

Inductor EI. Fig. 15 visualizes three data samples of a power electronic inductor application with EI shape iron core. This dataset has a fixed core gap location and contains two directions of one AC coil set applied in clustered copper wires. Therefore, this dataset is generated from a special sub design space of *Induc EI Gap*. The abbreviation for this dataset is *Induc EI*.

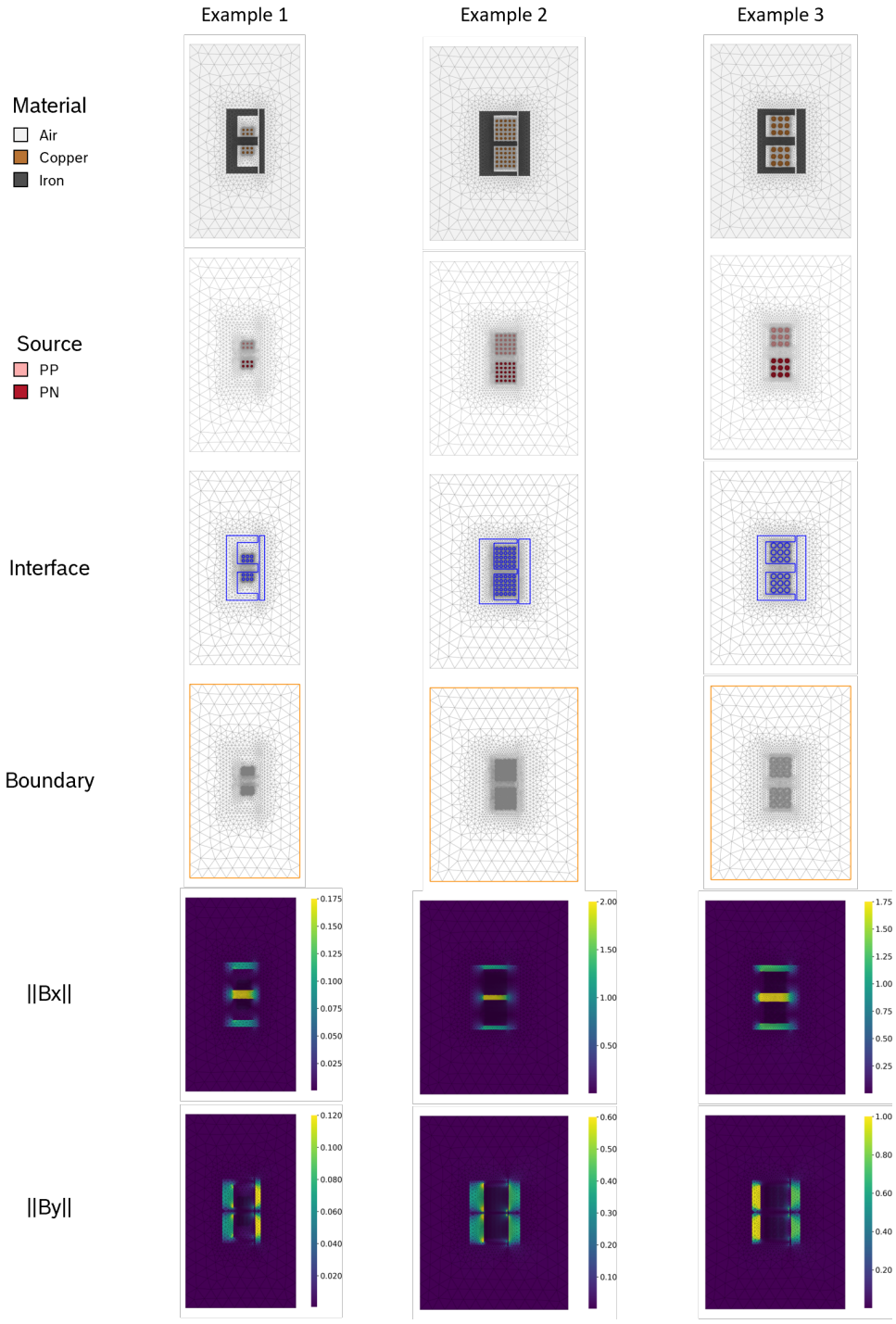

Figure 15: Visualization of three examples in the Inductor EI dataset.

Inductor Large Gap. Fig. 16 visualizes three data samples of a power electronic inductor application with large ring shape iron core. This dataset includes random iron core gap location and contains two directions of one AC coil set applied in clustered flat copper wires. Therefore, this dataset is generated from the super design space of *Induc Lg*. The abbreviation for this dataset is *Induc Lg Gap*.

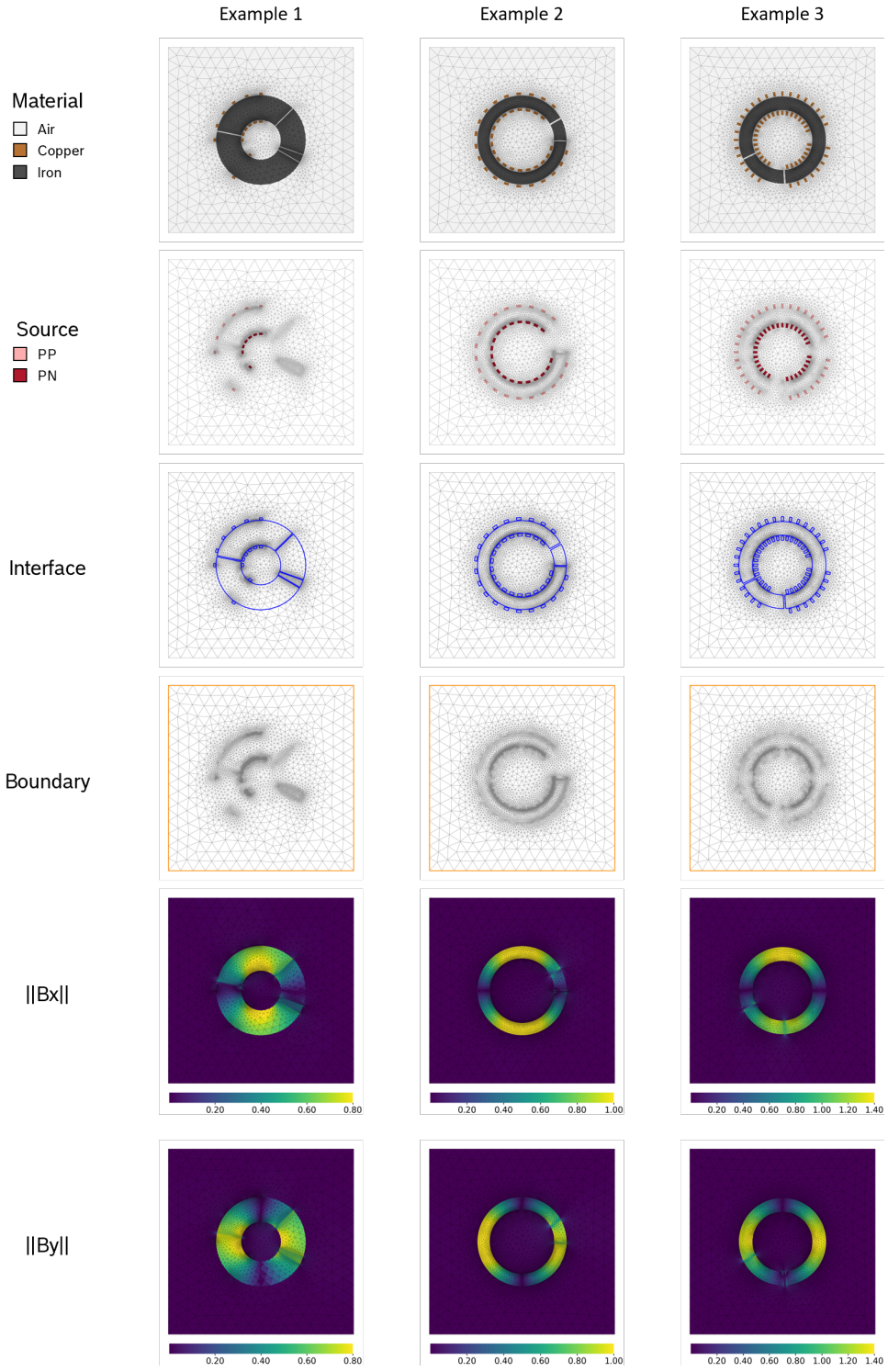

Figure 16: Visualization of three examples in the Inductor Large Gap dataset.

Electromagnet C Chunk. Fig. 17 visualizes three data samples of an electromagnet application with C shape iron housing. This dataset lies in rectangular coordinates with two directions of one DC coil set applied in copper chunks. The abbreviation for this dataset is *EM C Chunk*.

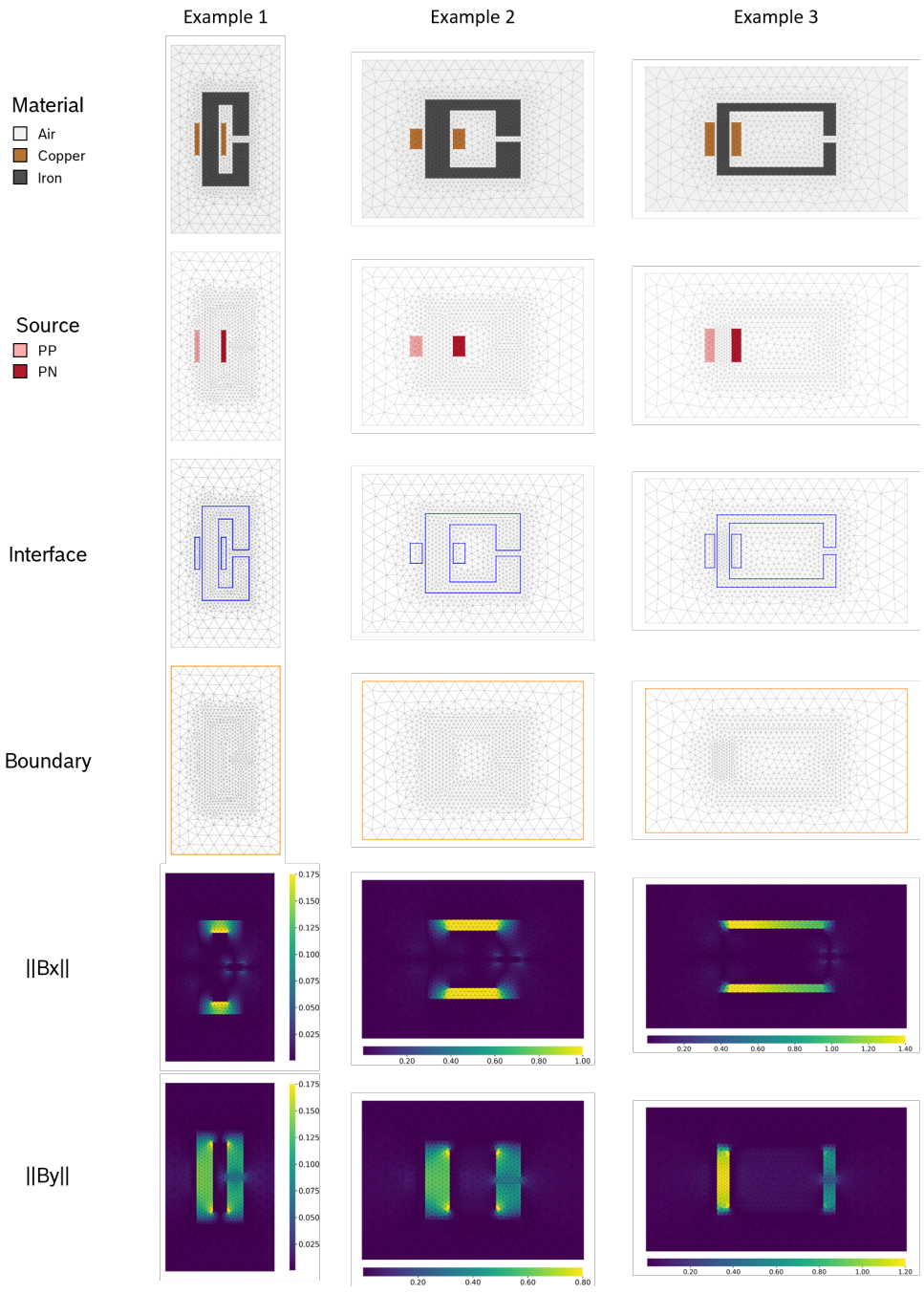

Figure 17: Visualization of three examples in the Electromagnet C Chunk dataset.

C HETEROGENEOUS PARALLEL TRAINING

In this section, we describe our parallel training implementation for handling the heterogeneous data structures and variable sequence lengths in MaxwellBench. This includes the customized dataset, dataloader, collate function, data batching strategy, distributed sampler and training recipe.

C.1 DATASET AND DATALOADER

As individual samples vary drastically in size, with larger ones containing tens of thousands of elements, we adopt a scattered data loading strategy. To avoid ambiguity, we use the term "vertices" to refer to graph vertices, while reserving "nodes" for mesh nodes.

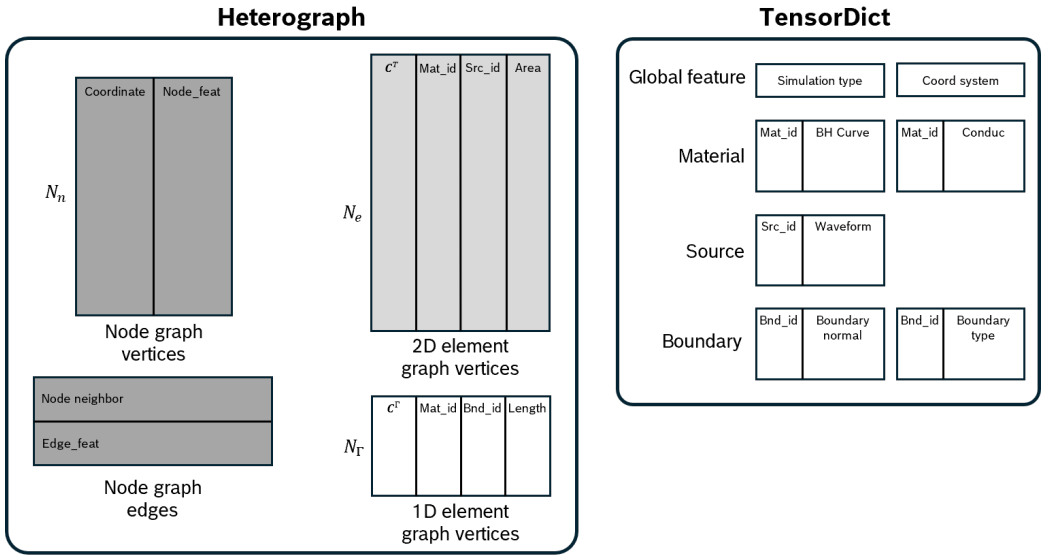

Figure 18: PyTorch representation of a single loaded sample.

Datasets for GMNO. To support both faithful representation of industrial FEM data and efficient batching during training, we design each sample as a tuple consisting of a heterograph and a tensor dictionary, as illustrated in Fig. 18. The key idea is to separate irregular mesh topology from reusable physical attributes: all physical input information is stored in the tensor dictionary and referenced by indices, rather than being explicitly expanded into field form, while all mesh information, together with its correspondence to the physical inputs, is stored in the heterograph. For example, if a simulation contains three materials, the corresponding three BH curves are stacked in the tensor dictionary and accessed through their material indices. This representation preserves the complete physical and geometric structure of an industrial FEM simulation, while also making heterogeneous samples easier to batch.

The heterograph contains three vertex sets. The first stores mesh node coordinates and augmented node features, and therefore has size N_n. The second stores triangular element connectivity, element areas, material indices, and source indices corresponding to entries in the tensor dictionary, and therefore has size N_e. The third stores line element connectivity, element lengths, material indices, and boundary condition indices, and therefore has size $N_b + N_\Gamma$. In addition, we store the neighborhood relations of mesh nodes as an edge index matrix in the heterograph for message passing, as described in Section 3.2.

Batching. This tuple based data representation also naturally enables an efficient batching strategy for PDE field data with highly unequal lengths. Existing operator learning practice mainly relies on three approaches: using a batch size of one, downsampling, or zero padding. Batch size one is prohibitively expensive for large scale pretraining, whereas downsampling and zero padding are both unsuitable for industrial-standard datasets such as MaxwellBench, where sample heterogeneity is

substantial. In our design, mesh information from different samples can be merged into a single large batched heterograph, while the physical inputs, referenced through indices, can be stacked normally in the tensor dictionary. As a result, most GMNO operations, including linear projections, message passing, aggregation, layer normalization, and MLPs, can be executed over the full batch in parallel. The only exception is linear attention, which is applied independently to each sample in order to avoid zero padding and preserve exact sample boundaries.

Normalization. Existing large scale pretraining studies have been focusing on autoregressive tasks where every dataset can be normalized separately as the magnitude information is self-contained. However, to study scalability and generalization of a real solution operator, a universal normalization scheme should be adopted across all datasets. We summarize some key statistical values in Table 4 and found that electromagnetism problems governed by Maxwell's equations have the following characteristics: current frequency is either zero or at large magnitudes; most other physical quantities are positive but can vary by orders of magnitudes; the magnetic flux density usually caps at 3 Tesla due to magnetic saturation; the pattern of magnetic flux density field is important in industrial scenarios even when the magnitude is small; the mesh element sizes can vary by orders of magnitudes. Based on these consideration, we adopt log normalization strategies across the entire MaxwellBench as below:

- The prediction target, magnetic flux density magnitudes $\{||B_{x/r}||_e, ||B_{y/z}||_e\}$, are normalized via $||\hat{B}|| = log(||B||).clamp_min(5 \times 10^{-4})/1.2 + 3.2$.
- Triangular element areas and line element lengths are normalized via $\hat{A} = log(A.clamp_min(1 \times 10^{-4}))$, $\hat{L} = log(L.clamp_min(1 \times 10^{-4}))$.
- Source current density is normalized via $\hat{\tilde{J}}_s = \frac{\tilde{J}_s}{3 \times 10^6}$, frequency is normalized via $\hat{f} = log_{10}(f.clamp_min(0.1))$.
- For BH curves, only the H values are normalized via $\hat{H} = log_{10}(H.clamp_min(0.1))$, as B values naturally saturate. Material conductivity is normalized via $\hat{\sigma} = log_{10}(\sigma.clamp_min(1 \times 10^{-5}))/2$.
- For Cartesian coordinates, all simulation data are normalized to center at origin with coordinates divided by 30, while cylindrical coordinates are directly divide by 30.

Augmentation. To improve robustness across different coordinate systems, we augment the raw node coordinates with coordinate aware geometric features at graph construction (before message passing). For all simulation data in MaxwellBench, each node coordinate is either in Cartesian coordinates (x, y) or in cylindrical coordinates (r, z). We append a coordinate system indicator to distinguish Cartesian from cylindrical samples, together with additional geometry features that encode the local metric structure. In the cylindrical case, these include the radial measure r, as well as $\log(r)$ and $\log(1/r)$, which help the model capture the nonuniform geometric scaling and near-axis behavior inherent to axisymmetric problems. For Cartesian data, the corresponding channels are filled with constant placeholders so that the feature dimensionality remains consistent across datasets. Edge attributes are constructed from the relative displacement between connected nodes, including the normalized edge direction, the logarithm of the edge length, and the average radial coordinate along each edge in cylindrical settings. Through this augmentation, the graph explicitly exposes both Euclidean geometry and coordinate dependent metric information, enabling a unified model to better handle both Cartesian and cylindrical simulations.

In addition to coordinate augmentation, we apply random geometric and physical transformations during pretraining of 0.54B parameter GMNO to further improve generalization. Specifically, we consider translation, rotation, and source flipping augmentations. Translation perturbs the mesh coordinates by a random offset; for Cartesian samples, both spatial directions are translated, whereas for cylindrical samples only the axial coordinate is shifted so that the radial coordinate remains physically meaningful. Rotation rotates the mesh about the origin by a randomly selected angle and is applied only to Cartesian samples. To preserve physical consistency, the target magnetic field and boundary normals are rotated accordingly. Finally, source flipping changes the sign of the excitation current density, and the corresponding magnetic field target is simultaneously negated. Each transformation is applied stochastically during training, and multiple transformations can be

composed within the same sample. These augmentations encourage GMNO to learn the underlying physical symmetries and invariances of the problem, rather than overfitting to specific coordinate realizations or excitation orientations.

Datasets for baselines. Since all baseline neural operators are designed for point cloud inputs, we convert our customized dataset into a point cloud representation for fair comparison. Because the 2D physical quantities in MaxwellBench are defined on elements rather than mesh nodes, we use the element centroids, $\{\frac{1}{3}\sum_{n\in\mathbf{C}^T_{e,:}}\mathbf{X}_n\}_{e=1}^{N_e}$, as the point cloud coordinates. All physical information, including material properties, current sources, boundary conditions, and global parameters, are concatenated with these coordinates to form the input channels. The normalization and coordinate augmentation schemes are kept identical to those used for GMNO. For Transolver, GAOT, and LNO, only 2D information is provided, since their architectures do not support heterogeneous field inputs. For GAOT, we follow its standard recipe and construct point cloud adjacency using constant volume ball search and nearest-neighbor search. For GNOT, we additionally include 1D information, as its heterogeneous cross attention layer can accommodate such inputs. However, for baseline comparison, we use the vanilla zero padded linear attention in GNOT.

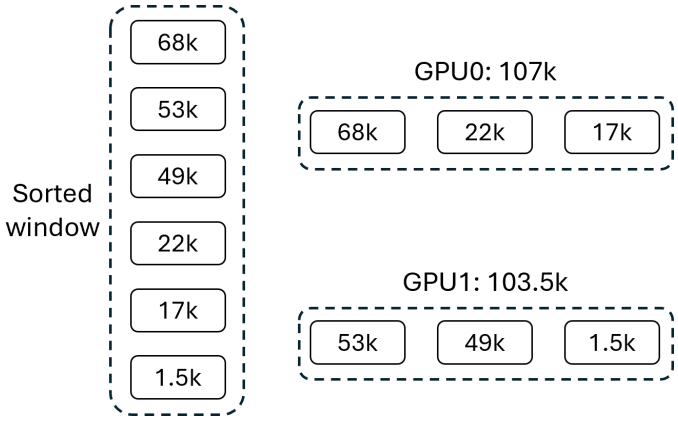

Figure 19: Illustration of the windowed greedy bucketed sampler. Samples are first globally shuffled and grouped into fixed size windows. Within each window, samples are sorted by the number of body elements, greedily packed into global microbatches with balanced total mesh size, and then split across ranks to balance per-GPU workload.

C.2 Heterogeneous Data Sampler.

As mesh sizes vary significantly across MaxwellBench, with up to two orders of magnitude difference as shown in Table 4, naive padding and distributed data sampling can introduce substantial computational overhead and severe load imbalance across GPUs. We therefore adopt a greedy bucketed distributed batch sampler as illustrated in Fig. 19. Specifically, all training samples are first globally shuffled at the beginning of each epoch, so that data from different datasets are naturally mixed. The shuffled index list is then divided into windows whose sizes are chosen as a multiple of the global batch size. Within each window, samples are sorted in descending order by the number of 2D triangular elements N_e, which serves as a proxy for computational cost. We then greedily assign samples to a set of global microbatches by minimizing the accumulated N_e in each bin while enforcing a fixed number of samples per microbatch. This produces global batches with approximately balanced total mesh size.

Each global microbatch is subsequently partitioned across distributed ranks using the same greedy balancing strategy. In this second stage, samples within the microbatch are assigned to different GPUs such that each rank receives the same number of samples while $\sum N_e$ per rank is approximately balanced. As a result, every training step processes a fixed per-rank batch size, while the computational load across devices is better aligned, reducing stragglers caused by exceptionally large meshes. Since bucket construction is performed only within local windows rather than over the entire dataset, the

Table 5: Hyperparameters for GMNO and other neural operator baselines under different training modes. In the layer counts a+b, a and b denote the numbers of cross-attention and self-attention layers, respectively.

	GNOT	Transolver	LNO	GAOT	GMNO	GMNO	GMNO	GMNO
Training mode	Per dataset training					Pretraining	ID FT	OOD FT
Hidden dim	256	256	256	256	280	1344		
Layers	5+5	8	12	8	4+4	10+10		
Heads	8	8	8	8	8	21		
Params	11.1M	10.2M	10.3M	11.0M	10.4M	0.54B		
Data	9600	9600	9600	9600	9600	96000	9600	240
Batch size	64	64	64	64	64	128	64	16
Augment	0%	0%	0%	0%	0%	40%	0%	0%
Window size	/				4800	4800	4800	120
Epochs	100	100	100	100	100	60	10	100
Warmup	10	10	10	10	10	8	1	10
Learning rate	1e-3	1e-3	1e-3	1e-3	1e-3	5e-4	2e-5	1e-3
Beta1	0.9							
Beta2	0.95							
Weight decay	1e-4							
Weight dtype	bf16							

sampler preserves sufficient stochasticity while substantially reducing padding waste and improving hardware utilization during large scale training.

C.3 Hyperparameters and Training Recipe.

We summarize the main hyperparameters for GMNO pretraining, finetuning, and baseline comparisons in Table 5.

To explore the scalability of GMNO, we first perform dense pretraining on 10 datasets from MaxwellBench, each containing 9600 simulation samples. A 0.54B parameter GMNO is trained on 8 Nvidia H100 GPUs using a bucketed distributed sampler and gradient accumulation, resulting in an effective batch size of 128. We use AdamW with a learning rate 5×10^{-4}, weight decay of 1×10^{-4}, and betas of $(0.9, 0.95)$. The learning rate follows a half-cycle cosine schedule: it is linearly warmed up over the first 8 epochs and then decayed to 1×10^{-6} over the remaining 52 epochs. We also conduct small scale training for ablation studies and baseline comparisons with other neural operators, using models of approximately 10M parameters. In addition, we apply data augmentation by randomly translating and rotating the mesh, while transforming the magnetic fields and boundary normals accordingly. We perform in-distribution (ID) testing on the 960 held-out samples from each of the 10 datasets.

Finetuning is performed on the pretrained 0.54B GMNO for both ID and OOD datasets. For each ID dataset, all parameters of the pretrained GMNO are finetuned for 10 epochs on 9600 training samples using AdamW with a reduced learning rate of 2×10^{-5}. The learning rate is warmed up for the first epoch and then decayed over the remaining 9 epochs. For each OOD dataset, we evaluate the pretrained GMNO under a scarce-data setting with only 240 training samples, in order to assess its generalization capability. In this regime, we freeze mesh embedding, physics embedding, and all linear attention blocks and update only the projection layers and a few LoRA modules. The active parameters are trained for 100 epochs with an effective batch size of 16, a learning rate of 1×10^{-3}, and 10 warmup epochs. In all cases, the finetuned models are evaluated on 960 held-out samples. Relative L_2 norm (Eq. 5) is used as the loss function for all training and evaluation, where $\hat{B}$ denotes the normalized magnetic flux density and D is the spatial dimension. To achieve generalization across MaxwellBench, we adopt a log normalization scheme as described under Normalization above.

C.3.1 FINETUNING STRATEGY ABLATION

We compare four finetuning strategies for the pretrained 0.54B GMNO using 240 training samples per OOD dataset and 100 finetuning epochs under the same reported recipe. Table 6 reports relative L_2 errors and the fraction of trainable parameters. Projection+LoRA achieves the lowest reported error on *Induc Lg Gap* and *EM C Chunk*; full finetuning is best on *Trans L*, and LoRA-only is slightly better on *Induc EI*. Projection+LoRA therefore offers a competitive accuracy-parameter trade-off.

Table 6: OOD finetuning strategy ablation for the 0.54B GMNO. All strategies use 240 training samples and 100 epochs. Relative L_2 errors are in units of $\times 10^{-2}$; bold marks the lowest error per dataset. Trainable fractions refer to the full 0.54B model.

OOD dataset	Full	LoRA-only	Projection-only	Proj+LoRA
Trans L	**3.2**	4.2	6.6	3.5
Induc EI	9.2	**2.9**	3.4	**3.0**
Induc Lg Gap	10.3	6.3	6.9	**5.9**
EM C Chunk	6.9	**3.4**	3.7	**3.3**
Trainable parameters	100%	0.7%	3.2%	3.9%

D COMPUTE RESOURCES

Due to the heterogeneity of MaxwellBench, sequence lengths (i.e., mesh sizes) vary substantially both across and within datasets. As a result, memory consumption and training time depend not only on the model architecture, but also strongly on factors such as GPU configuration, distributed training strategy, batching scheme, gradient accumulation, and activation checkpointing. It is therefore difficult to summarize the computational requirements with a single number. Instead, we provide the following representative examples to offer an intuitive sense of the resource demands. Please refer to the code for the detailed configuration of each training setting.

- Training 11.0M GAOT on *Trans UU*: 2 $\times$ H100, 31GB/GPU, 240s/epoch.
- Training 10.4M GMNO on *Trans UU*: 2 $\times$ H100, 15GB/GPU, 500s/epoch.
- Pretraining 0.54B GMNO: 8 $\times$ H100, 28GB/GPU, 9600s/epoch.
- ID finetuning 0.54B GMNO on *Trans UU*: 8 $\times$ H100, 30GB/GPU, 2700s/epoch.
- Training 10.4M GMNO on scarce *Trans L*: 2 $\times$ H100, 8GB/GPU, 6s/epoch.
- OOD finetuning 0.54B GMNO on scarce *Trans L*: 2 $\times$ H100, 32GB/GPU, 40s/epoch.

E BROADER IMPACTS

This work introduces an industrial-standard simulation benchmark and proposes a novel model architecture and training strategy for learning, scaling, and generalizing over complex heterogeneous simulation data. We anticipate positive impact in both research and industry domains where partial differential equations are used to model and simulate the physical world. At the same time, we do not foresee clear avenues for harmful misuse of our methods, nor do we anticipate negative societal impacts arising directly from this work.