mPOD-DeepONet: POD-DeepONet for multiple outputs

Chieh-An Chou & Lu-Hung Chen
Department of Applied Mathematics
National Chung Hsing University
Taichung 402, Taiwan
d114053001@mail.nchu.edu.tw, luhung@email.nchu.edu.tw

Abstract

Many scientific systems require predicting multiple coupled fields simultaneously, yet naïve strategies for extending DeepONet to multi-output settings—such as adding output channels or training separate trunks—yield inconsistent and often poor performance across problems. We propose two POD-based alternatives: cPOD-DeepONet, which performs POD independently per output field and predicts coefficients channel-wise, and mPOD-DeepONet, which leverages multivariate functional PCA to produce a shared basis across all fields so that the branch network predicts a single compact latent vector, significantly reducing model size. Additionally, we explore mKPCA-DeepONet, a kernel PCA extension that captures non-linear cross-channel correlations while maintaining the same compact architecture as mPOD-DeepONet. Extensive evaluations on four multiphysics benchmarks show that all POD-based architectures dramatically improve over naïve DeepONet extensions and deliver consistently robust results, outperforming FNO on half of the benchmarks while remaining competitive on the others—all with substantially faster inference. Between the two linear variants, mPOD-DeepONet achieves comparable accuracy to cPOD-DeepONet with a notably smaller model footprint. Both models remain robust across a wide range of basis sizes, simplifying practical deployment.

1 Introduction

Neural Operators (NOs) have emerged as a powerful paradigm in Scientific Machine Learning, learning mappings between infinite-dimensional function spaces to bypass the computational cost of classical PDE solvers. Among them, DeepONet (Lu et al., 2021) approximates operators via a branch-trunk dot product, while POD-DeepONet (Lu et al., 2022) replaces the learned trunk with a data-driven POD basis, improving learning efficiency and physical consistency. Extensions such as MIONet (Jin et al., 2022) further handle multi-input scenarios.

However, most existing work focuses on single-output operators, whereas many scientific systems require predicting multiple coupled fields simultaneously. Naïve strategies such as adding output channels to the shared trunk (mDeepONet) or training separate trunks for each variable (cDeepONet) yield inconsistent results: on some benchmarks, they perform adequately, yet on others—particularly those with stronger cross-variable coupling—they degrade severely compared to FNO, as they fail to model shared latent structure across fields. Our experiments confirm this instability: naïve DeepONet extensions exhibit order-of-magnitude increases in error on the Electro-Fluid and MHD benchmarks, motivating principled multi-output architectures with more reliable performance.

We propose two primary POD-DeepONet variants, alongside a non-linear extension, for the multi-output setting. The first, cPOD-DeepONet, performs POD independently on each output field and predicts coefficients per channel. This approach dramatically improves robustness over naïve DeepONet extensions and, on several benchmarks, outperforms FNO—though its model size grows with the number of output variables. To obtain a more compact architecture, we introduce mPOD-DeepONet, which leverages multivariate functional PCA (mFPCA) (Happ & Greven, 2018)—a joint

eigen-decomposition of cross-covariance operators—to produce a shared basis across all fields. The branch network then predicts a single compact latent vector instead of separate coefficients per field, significantly reducing model size while maintaining comparable accuracy. Furthermore, we explore mKPCA-DeepONet, a kernel extension that captures non-linear cross-channel correlations while preserving this exact compact footprint. Experiments on four multiphysics benchmarks show that POD-based architectures deliver consistently competitive results, outperforming FNO on half of the benchmarks and remaining close on the others, with substantially faster inference. In contrast, naïve DeepONet extensions suffer from erratic performance across problems. These results establish POD-based architectures as robust and efficient alternatives to FNO for multi-output operator learning, with the mPOD and mKPCA variants offering a notably smaller model footprint.

2 METHODOLOGY

The standard DeepONet approximates an operator $G : \mathcal{U} \to \mathcal{V}$ via the inner product of a branch network $\{b_k(u)\}_{k=1}^p$ and a trunk network $\{t_k(y)\}_{k=1}^p$:

$$G(u)(y) \approx \sum_{k=1}^{p} b_k(u) t_k(y) \tag{1}$$

POD-DeepONet modifies this architecture by replacing the learned trunk functions with p precomputed basis functions $\{\phi_k(y)\}_{k=1}^p$ obtained via Proper Orthogonal Decomposition (POD):

$$G(u)(y) \approx \sum_{k=1}^{p} b_k(u) \phi_k(y) \tag{2}$$

In this formulation, the branch network predicts the spectral coefficients $b_k(u)$ corresponding to the physical modes $\phi_k(y)$ of the system.

For a discretized domain with N_y spatial points, the output field can be written in matrix form. Let $\mathbf{v} \in \mathbb{R}^{N_y}$ denote the discretized output and $\mathbf{\Phi} \in \mathbb{R}^{N_y \times p}$ the POD basis matrix whose columns correspond to ϕ_k. Then

$$\mathbf{v} \approx \mathbf{\Phi} \mathcal{B}(u), \tag{3}$$

where $\mathcal{B}(u) \in \mathbb{R}^p$ is the branch network output. This matrix formulation serves as the common foundation for extending POD-DeepONet to multivariate operator learning.

2.1 CHANNEL-WISE POD-DEEPONET (CPOD-DEEPONET)

We first consider a direct extension of POD-DeepONet to multivariate outputs by treating each physical variable as an independent field, referred to as Channel-wise POD-DeepONet (cPOD-DeepONet).

Consider a multivariate output $\mathbf{V} = (\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(C)})$ with C physical channels. For each channel c, we perform an independent POD on the corresponding training snapshots, yielding a channel-specific basis matrix $\mathbf{\Phi}^{(c)} \in \mathbb{R}^{N_y \times p_c}$. The branch network $\mathcal{B}$ maps the input u directly to a concatenated vector $\mathbf{S} = [\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(C)}]^\top \in \mathbb{R}^{\sum p_c}$. Each field $\mathbf{v}^{(c)}$ is reconstructed by extracting the corresponding segment of the network output, $\mathbf{s}^{(c)} = [\mathcal{B}(u)]^{(c)}$:

$$\mathbf{v}^{(c)} = \mathbf{\Phi}^{(c)} [\mathcal{B}(u)]^{(c)} \tag{4}$$

This decoupled strategy offers conceptual simplicity and modularity, though the output dimension of the branch network grows with the number of channels.

2.2 MULTIVARIATE FUNCTIONAL PCA (MFPCA)

To obtain a more compact latent representation, we adopt the hierarchical mFPCA framework (Happ & Greven, 2018), which compresses the concatenated coefficient space via a two-stage process:

Stage I: Univariate Channel Decomposition. This stage corresponds exactly to the basis extraction used in cPOD-DeepONet. For each channel, we approximate the field as $\mathbf{v}^{(c)} \approx \mathbf{\Phi}^{(c)} \mathbf{s}^{(c)}$, where $\mathbf{s}^{(c)}$ are the univariate scores.

Stage II: Joint Latent Projection. The univariate coefficient vectors are aggregated into a single vector $\mathbf{S} = [\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(C)}]^\top$. A secondary PCA is then performed on $\mathbf{S}$, yielding a joint eigenvector matrix $\mathbf{\Psi}_{p_{\text{joint}}} \in \mathbb{R}^{(\sum p_c) \times p_{\text{joint}}}$ and a reduced joint latent vector $\mathbf{z} \in \mathbb{R}^{p_{\text{joint}}}$ such that $\mathbf{S} \approx \mathbf{\Psi}_{p_{\text{joint}}}\mathbf{z}$. For a given channel c, the corresponding univariate coefficients are recovered as:

$$\mathbf{v}^{(c)} \approx \mathbf{\Phi}^{(c)}[\mathbf{\Psi}_{p_{\text{joint}}}]^{(c)}\mathbf{z} \tag{5}$$

where $[\mathbf{\Psi}_{p_{\text{joint}}}]^{(c)}$ denotes the sub-matrix of associated with channel c such that $\mathbf{s}^{(c)} \approx [\mathbf{\Psi}_{p_{\text{joint}}}]^{(c)}\mathbf{z}$.

2.3 Multi-channel POD-DeepONet (mPOD-DeepONet)

Building on the full two-stage mFPCA analysis, we introduce Multi-channel POD-DeepONet (mPOD-DeepONet). Rather than predicting channel-wise coefficients directly, the branch network $\mathcal{B}$ is trained to output the compact joint latent vector $\mathbf{z} \in \mathbb{R}^{p_{\text{joint}}}$. The reconstruction for each physical variable $\mathbf{v}^{(c)}$ is obtained by sequentially projecting $\mathbf{z}$ through the joint eigenvectors $\mathbf{\Psi}_{p_{\text{joint}}}$ and the channel-specific POD bases $\mathbf{\Phi}$:

$$\mathbf{v}^{(c)} \approx \mathbf{\Phi}^{(c)}[\mathbf{\Psi}_{p_{\text{joint}}}]^{(c)}\mathcal{B}(u) \tag{6}$$

Since $p_{\text{joint}} \ll \sum p_c$ in practice, this yields a significantly smaller branch output layer compared to cPOD-DeepONet, reducing model size while maintaining comparable accuracy. To capture nonlinear cross-channel correlations, we further propose an kernel-based extension (Eivazi et al., 2024), Multi-channel KPCA-DeepONet (mKPCA-DeepONet), with full mathematical details provided in Appendix A.

3 Numerical Experiments

We evaluate the proposed cPOD-DeepONet, mPOD-DeepONet and mKPCA-DeepONet architectures on four multi-physics benchmarks from the MPBench collection (Yang et al., 2025): Electro-Thermal, Thermo-Fluid, Electro-Fluid, and Magneto-Hydrodynamics (MHD). These benchmarks span a broad spectrum of coupling mechanisms, ranging from temperature-dependent material properties to strongly bidirectional electromagnetic-fluid interactions. A concise overview of the input-output mappings and governing physical processes is provided in Table 1, while detailed governing equations are reported in Appendix B.

Table 1: Summary of Multiphysics Datasets from MPBench

Dataset	**Input**	**Output**	**Physical Description**
Electro-Thermal	σ/κ	$\Re(E_z), \Im(E_z), T$	Bidirectional coupling between the **wave equation** and **heat conduction**. Electrical conductivity σ depends on temperature, while Joule heating $\frac{1}{2}\sigma\lvert\mathbf{E}\rvert^2$ acts as a thermal source.
Thermo-Fluid	Q	u_x, u_y, T	Coupled **Navier–Stokes** and **heat balance** equations, where temperature gradients induce buoyancy effects that influence the velocity field.
Electro-Fluid	σ	u_x, u_y, V	Interaction between **Navier–Stokes** and the **current continuity equation**, modeling electro-osmotic flow driven by electric potential gradients.
MHD	B_z	u_x, u_y, J_x, J_y, J_z	**Magneto-hydrodynamics** with strong bidirectional coupling through the **Lorentz force** $\mathbf{J} \times \mathbf{B}$, involving Ampère's law and charge conservation for a five-channel output.

Model performance is assessed using the Channel-wise Integrated Squared Error (CISE), with Total-CISE serving as the primary quantitative metric. The results, summarized in Table 2, compare our proposed frameworks against the Fourier Neural Operator (FNO) and two naïve DeepONet extensions: cDeepONet (independent trunks for each output variable) and mDeepONet (multiple output channels in the shared trunk). Full experimental details are provided in Appendix C, while Appendix D contains extended performance tables, error maps, and sensitivity analyses for p.

Quantitative Analysis Across all multiphysics benchmarks, POD-based configurations consistently deliver competitive results. Our models outperform FNO on two of the four benchmarks (Electro-Thermal and MHD) and remain competitive on Thermo-Fluid, though a notable gap persists on

the Electro-Fluid problem. In contrast, the two naïve multi-output DeepONet extensions exhibit highly inconsistent behavior: while they perform reasonably on Electro-Thermal, they suffer severe degradation on other benchmarks—most strikingly cDeepONet on MHD and both on Electro-Fluid, where CISE values are orders of magnitude higher than those of the POD-based alternatives.

Both cPOD-DeepONet and mPOD-DeepONet achieve nearly identical predictive accuracy and are insensitive to the basis size p, with variations in CISE remaining within statistical uncertainty across the tested range. These results underscore that principled basis-based architectures are essential for reliable multi-output operator learning where standard extensions fail. The mKPCA extension only improves in which linear POD already outperforms FNO. However, it is more sensitive to the values of the basis size p and the hyperparameters of the kernel.

Visual Quality and Artifacts Qualitative analysis reveals distinct error topologies across the datasets. In fluid-dominated tasks, FNO consistently produces the smoothest fields, whereas the standard DeepONet baselines exhibit diffuse, concentric ripple patterns radiating across the domain. All POD-based models (including the mKPCA extension) suppress these distributed ripples but introduce a sharp, localized ring-shaped artifact immediately surrounding geometric discontinuities. Conversely, in the Electro-Thermal and MHD benchmarks, FNO displays widespread grid-like textures, while standard DeepONets suffer from extensive scattered graininess. In these regimes, the POD-based architectures yield the cleanest visual results, effectively avoiding both the grid textures and the scattered artifacts observed in the baselines.

Computational Efficiency The primary distinction between the two proposed configurations lies in their computational trade-offs (see Table 3 in Appendix C.1). The mPOD- and mKPCA-DeepONet models maintain a more compact parameter footprint with respect to the basis size p compared to the linear growth of cPOD-DeepONet. However, this compression incurs a slight overhead in training and inference times due to the increased complexity of the projection mechanism. Despite these internal differences, all POD-based frameworks maintain a significant efficiency advantage over the FNO baseline, achieving an approximate 2–3$\times$ speedup in training and inference.

Table 2: Model performance comparison across datasets (Total CISE, $\times 10^3$). For DeepONet-series models, results reflect the optimal configuration across all tested basis sizes p. Bold and underlined values indicate the best and second-best results for each dataset, respectively.

Dataset	FNO	cDeepONet	mDeepONet	cPOD-DeepONet	mPOD-DeepONet	mKPCA-DeepONet
Thermo-Fluid	$\mathbf{0.14 \pm 0.09}$	2.30 ± 0.90	2.22 ± 0.87	0.33 ± 0.14	$\underline{0.28 \pm 0.14}$	0.51 ± 0.23
Electro-Thermal	45.39 ± 42.36	1.02 ± 1.07	0.85 ± 0.85	$\underline{0.56 \pm 0.42}$	0.69 ± 0.50	$\mathbf{0.39 \pm 0.32}$
Electro-Fluid	$\mathbf{0.45 \pm 0.58}$	99.49 ± 11.44	100.97 ± 11.68	56.00 ± 8.25	$\underline{53.80 \pm 7.93}$	66.75 ± 9.59
MHD	31.60 ± 23.72	815.13 ± 12.61	34.17 ± 21.73	$\underline{25.15 \pm 23.59}$	25.34 ± 23.74	$\mathbf{24.41 \pm 22.71}$

4 Conclusion and discussion

We proposed cPOD-DeepONet and mPOD-DeepONet, alongside a non-linear kernelized extension (mKPCA-DeepONet), for multi-output operator learning. These architectures dramatically improve over naïve multi-output DeepONet modifications and deliver consistently robust performance across all four benchmarks. Compared to FNO, the POD-based variants outperform it on half of the benchmarks (Electro-Thermal and MHD) while remaining competitive on Thermo-Fluid. A gap persists on the Electro-Fluid problem; our exploration of the non-linear mKPCA extension revealed that while it further improves accuracy on the benchmarks where linear POD already excels, it is insufficient to close the gap on strongly coupled multi-field systems. Crucially, all architectures achieve these results at substantially faster inference speed. Between the linear variants, mPOD-DeepONet attains similar accuracy with a notably smaller model footprint by leveraging a shared mFPCA basis, while both remain insensitive to the choice of truncation rank.

Future work includes exploiting the mFPCA framework of (Happ & Greven, 2018) to extend the architecture beyond co-located uniform grids, incorporating physics-informed constraints, and scaling to systems with many more output channels as further steps toward robust, production-level multivariate operator learning.

REFERENCES

Hamidreza Eivazi, Stefan Wittek, and Andreas Rausch. Nonlinear model reduction for operator learning. In *The Second Tiny Papers Track at ICLR 2024*, 2024. URL https://openreview.net/forum?id=Jw6TUpB7Rw.

Clara Happ and Sonja Greven. Multivariate functional principal component analysis for data observed on different (dimensional) domains. *Journal of the American Statistical Association*, 113 (522):649–659, 2018.

Pengzhan Jin, Shuai Meng, and Lu Lu. Mionet: Learning multiple-input operators via tensor product. *SIAM Journal on Scientific Computing*, 44(6), 2022.

Jean Kossaifi, Nikola Kovachki, Zongyi Li, David Pitt, Miguel Liu-Schiaffini, Valentin Duruisseaux, Robert Joseph George, Boris Bonev, Kamyar Azizzadenesheli, Julius Berner, and Anima Anandkumar. A library for learning neural operators. *arXiv preprint arXiv:2412.10354*, 2025.

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.

Lu Lu, Xuhui Meng, Shengze Cai, Zhiping Mao, Somdatta Goswami, Zhongqiang Zhang, and George Em Karniadakis. A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data. *Computer Methods in Applied Mechanics and Engineering*, 393:114778, 2022.

Changfan Yang, Lichen Bai, Yinpeng Wang, Shufei Zhang, and Zeke Xie. Multiphysics bench: Benchmarking and investigating scientific machine learning for multiphysics pdes. *arXiv preprint arXiv:2505.17575*, 2025.

A NON-LINEAR EXTENSION: MULTI-CHANNEL KPCA-DEEPONET (MKPCA-DEEPONET)

To capture non-linear correlations among the channel-wise univariate coefficients, we employ Kernel PCA (KPCA) with a Radial Basis Function (RBF) kernel for the joint latent projection (Eivazi et al., 2024). This extracts the top p_{joint} joint kernel eigenvectors, denoted as $\mathbf{\Psi}_{\text{KPCA}}$, and a linear pre-image mapper matrix $\mathbf{M}$ is learned via ridge regression to invert the non-linear mapping.

In mKPCA-DeepONet, the branch network predicts the compact latent vector $\mathcal{B}(u) \in \mathbb{R}^{p_{\text{joint}}}$. During inference, the reconstruction for each physical channel c strictly reduces to efficient linear matrix multiplications:

$$\mathbf{v}^{(c)} \approx \mathbf{\Phi}^{(c)} \left[\mathbf{M}^\top \mathbf{\Psi}_{\text{KPCA}}\right]^{(c)} \mathcal{B}(u) \tag{7}$$

However, a key limitation of this approach is its sensitivity to hyperparameter selection; optimal performance depends heavily on the choice of kernel function and careful tuning of parameters, such as the scale parameter γ in the RBF kernel.

B DATASET DESCRIPTIONS AND CONFIGURATION

We evaluate our proposed models on four diverse multiphysics datasets from the MPBench collection (Yang et al., 2025): Thermo-Fluid, Electro-Thermal, Electro-Fluid, and Magneto-Hydrodynamics. All physical fields in these datasets are discretized on a 128×128 spatial grid. The standard configuration for this collection provides $10,000$ training and $1,000$ testing samples per system. For Electro-Thermal, we re-partition the training set into $9,000$ training and $1,000$ testing samples due to missing independent test data in the original repository. The governing equations and specific input-output configurations for each system are detailed below.

Thermo-Fluid System This dataset involves the Navier-Stokes Equations and the Heat Balance Equation. It describes a fluid flow coupled with heat transfer, where the velocity field $\mathbf{u}$ and pressure p are affected by temperature gradients T.

$$\rho(\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla p + \mu \nabla^2 \mathbf{u} + \rho \mathbf{g}, \quad \nabla \cdot (\rho \mathbf{u}) = 0, \quad \rho C_p \mathbf{u} \cdot \nabla T - \nabla \cdot (\kappa \nabla T) = Q.$$

The model utilizes the heat source term Q as the input function to predict 3 output channels: the velocity components (u_x, u_y) and the temperature field T.

Electro-Thermal System This system models the interaction between the Wave Equation and the Heat Conduction Equation. The electric field $\mathbf{E}$ generates heat via Joule heating, which in turn influences the temperature T.

$$\nabla^2\mathbf{E} + k_0^2\mu_r\varepsilon_r\left(1 + \frac{\sigma}{j\omega\varepsilon_0}\right)\mathbf{E} = \mathbf{0}, \quad \nabla\cdot(\kappa\nabla T) + \frac{1}{2}\sigma|\mathbf{E}|^2 = 0.$$

For this dataset, the model takes the ratio of electrical to thermal conductivity σ/κ as input and predicts 3 output channels consisting of the complex electric field component E_z (represented by its real and imaginary parts) and the temperature field T.

Electro-Fluid System This system represents the coupling of the Navier-Stokes Equations and the Current Continuity Equation. In this microfluidic context, the fluid motion $\mathbf{u}$ is driven by both a pressure gradient and the electric potential V.

$$\nabla\cdot\mathbf{u} = 0, \quad \nabla\cdot(\sigma\nabla V) = 0, \quad \mathbf{u} = -\frac{\epsilon_p a^2}{8\mu}\nabla p + \frac{\epsilon_p\epsilon_w\zeta}{\mu}\nabla V.$$

The input for this system is the electrical conductivity σ, and the model predicts 3 output channels: the velocity components (u_x, u_y) and the electric potential V.

Magneto-Hydrodynamics This system involves the complex bidirectional coupling of Ampère's Law, the Continuity Equation, the Navier-Stokes Equations, and the Lorentz Force. The current density $\mathbf{J}$ is calculated in three directions (J_x, J_y, J_z), while the Lorentz force $(\mathbf{J}\times\mathbf{B})$ modifies the fluid velocity.

$$\mathbf{J} = \sigma(-\nabla V + \mathbf{u}\times\mathbf{B}), \quad \nabla\cdot\mathbf{J} = 0, \quad \nabla\times\mathbf{B} = \mu_s\mathbf{J}, \quad \rho(\mathbf{u}\cdot\nabla)\mathbf{u} = -\nabla p + \mu\nabla^2\mathbf{u} + \mathbf{J}\times\mathbf{B}.$$

In this setup, the model takes the out-of-plane magnetic field B_z as input to predict 5 output channels: the velocity components (u_x, u_y) and the three-dimensional current density components (J_x, J_y, J_z).

C Experimental Details

C.1 Model Architectures and Computational Efficiency

To provide a comprehensive evaluation, we benchmark our proposed methods against established baselines. We compare a total of six operator learning frameworks: the Fourier Neural Operator (FNO), cDeepONet, mDeepONet, cPOD-DeepONet, mPOD-DeepONet, and mKPCA-DeepONet. Our code for replicating the experiments is available at `https://github.com/Chieh997/mPOD-DeepONet`.

Due to the high computational cost associated with training FNO, the model was evaluated using its default optimal configurations from NeuralOperator (Kossaifi et al., 2025). In contrast, for the detailed analysis of basis function impact, we conducted a sensitivity sweep across basis dimensions $p \in \{256, 512, 1024, 2048, 4096\}$ for the DeepONet-series architectures (cDeepONet, mDeepONet, cPOD, mPOD, mKPCA). These models share a four-layer convolutional branch network to process input fields. The specific configurations are as follows:

- **FNO:** Implemented via the NeuralOperator library, this model utilizes spectral convolutions to parameterize the integral operator in the frequency domain. We adopt the default architecture settings provided in the library.
- **cDeepONet:** This variant employs a bank of C independent trunk networks, where each trunk is dedicated to a specific output variable to learn unique spatial basis functions of dimension p per channel.
- **mDeepONet:** This baseline uses multiple output channels in the shared trunk network that project spatial coordinates into a concatenated basis space of dimension $p\times C$ for all output variables simultaneously.

- **cPOD-DeepONet:** This model utilizes precomputed univariate POD basis functions for each channel independently. We set the number of basis per channel $p_c = p$, and the branch network predicts the concatenated coefficients.
- **mPOD-DeepONet:** This multi-channel approach first extracts high-fidelity univariate bases with $p_c = N_y/2$ (8192 modes for the 128×128 grid) before applying a secondary joint PCA. The branch network predicts a compressed latent vector of dimension $p_{\text{joint}} = p$.
- **mKPCA-DeepONet:** As a non-linear extension of mPOD-DeepONet, this variant retains identical dimension settings ($p_c = N_y/2$ and $p_{\text{joint}} = p$). The RBF kernel scale parameter γ is optimized via grid search over $\{10^{-6}, 10^{-5}, \ldots, 10^{-2}\}$.

Table 3: Summary of trainable parameters (n_{param}), average training time per epoch (t_{train}, in seconds), and average inference time (t_{test}, in seconds) across different basis sizes p on the Thermo-Fluid system.

	mDeepONet			**cDeepONet**			**cPOD-DeepONet**			**mPOD-DeepONet**			**mKPCA-DeepONet**		
p	n_{param}	t_{train}	t_{test}	n_{param}	t_{train}	t_{test}	n_{param}	t_{train}	t_{test}	n_{param}	t_{train}	t_{test}	n_{param}	t_{train}	t_{test}
4096	28.9M	10.02	0.35	29.2M	10.02	0.35	22.5M	3.84	0.13	14.1M	4.68	0.18	14.1M	4.74	0.21
2048	19.5M	6.52	0.23	19.7M	6.52	0.23	16.2M	3.24	0.11	12.0M	4.40	0.16	12.0M	4.39	0.20
1024	14.8M	4.77	0.17	15.0M	4.77	0.17	13.1M	2.95	0.10	11.0M	4.24	0.16	11.0M	4.55	0.22
512	12.4M	3.92	0.14	12.7M	3.92	0.14	11.5M	2.79	0.10	10.4M	4.18	0.16	10.4M	4.20	0.19
256	11.2M	3.49	0.12	11.5M	3.49	0.12	10.7M	2.68	0.10	10.2M	4.13	0.15	10.2M	4.17	0.19
FNO	Baseline: 0.7 M params, 12.72s training time, 0.51s testing time.														

Computational Efficiency Analysis: Table 3 summarizes the computational costs associated with each framework. While the FNO baseline maintains a low parameter count, it incurs the highest computational overhead (≈ 12.72 s/epoch) and slowest inference speeds. The standard DeepONet benchmarks (cDeepONet and mDeepONet) improve upon this latency but suffer from significant parameter bloat and higher training costs, scaling up to ≈ 10 s/epoch and nearly 30 M parameters at high resolutions ($p = 4096$). In contrast, the proposed POD-based architectures demonstrate superior efficiency. cPOD-DeepONet achieves the fastest raw training speeds but exhibits rapid parameter growth as the basis size increases. mPOD-DeepONet offers the optimal balance for high-fidelity modeling; although its joint projection incurs a slight, stable overhead (≈ 4 s/epoch), it demonstrates remarkable scalability, maintaining a significantly more compact parameter footprint (14.1 M vs. 22.5 M for cPOD at $p = 4096$) while significantly outperforming the standard DeepONet baselines. Lastly, the non-linear extension, mKPCA-DeepONet, preserves the exact parameter count of mPOD-DeepONet, incurring only a marginal computational overhead for the kernelized projection.

C.2 Training Configuration

All models were trained using the Adam optimizer with an initial learning rate of 5×10^{-4} and no weight decay. We employed a ReduceLROnPlateau scheduler that scales the learning rate by a factor of 0.8 if the validation loss does not improve for 3 consecutive epochs. The models were trained for a total of 500 epochs with a batch size of 32. All implementations were executed in PyTorch on a single NVIDIA GeForce RTX 5080 GPU.

C.3 Performance Metrics

To evaluate model accuracy, we utilize the Channel-wise Integrated Squared Error (CISE). For a given sample i and channel c, the CISE is computed in discrete form by averaging the squared error over the N_y spatial points:

$$\text{CISE}_{i,c} = \frac{1}{N_y} \sum_{j=1}^{N_y} \left(v_{i,j}^{(c)} - \hat{v}_{i,j}^{(c)} \right)^2 \tag{8}$$

Based on this foundation, we report the following metrics:

- **Global Metric (Total-CISE):** This serves as our primary performance indicator and training objective. Defined as the sum of CISE across all channels per sample, $S_i = \sum_{c=1}^{C} \text{CISE}_{i,c}$. We compare models using the mean and standard deviation (SD) of S_i across the test set.
- **Channel-wise Analysis:** For detailed comparisons, we report the mean and SD of $\text{CISE}_{i,c}$ calculated independently for each channel c across the sample population.

D RESULTS

Table 4: Performance comparisons across four multiphysics benchmarks. Values represent the mean and standard deviation of the CISE ($\times 10^3$) on the test set. For all DeepONet-series models, results correspond to the optimal configuration across hyperparameters basis size p and RBF scale γ.

(a) Thermo-Fluid

Model	Total	$\mathbf{u}_u$	$\mathbf{u}_v$	T
FNO	0.14± 0.09	0.05± 0.03	0.04± 0.02	0.05± 0.05
cDeepONet	2.30± 0.90	1.14± 0.43	0.88± 0.34	0.29± 0.16
mDeepONet	2.22± 0.87	1.05± 0.41	0.93± 0.35	0.25± 0.13
cPOD-DeepONet	0.33± 0.14	0.15± 0.06	0.13± 0.05	0.05± 0.04
mPOD-DeepONet	0.28± 0.14	0.13± 0.06	0.11± 0.05	0.04± 0.05
mKPCA-DeepONet	0.51± 0.23	0.22± 0.10	0.20± 0.09	0.08± 0.06

(b) Electro-Thermal

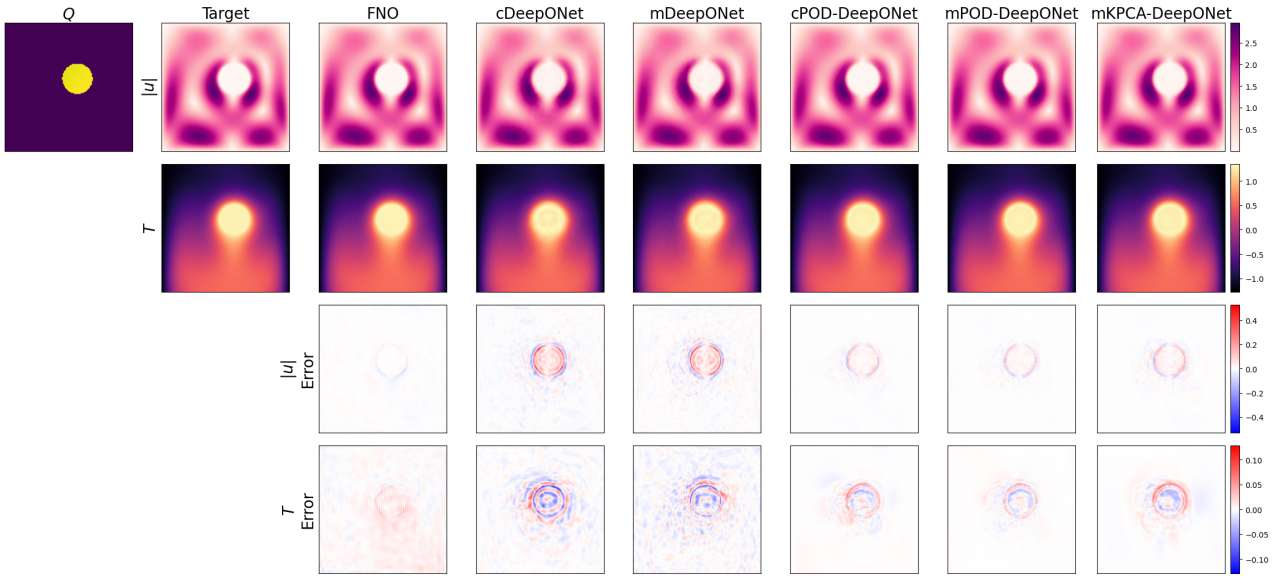
Model	Total	Re($\mathbf{E}_z$)	Im($\mathbf{E}_z$)	T
FNO	45.39±42.36	19.88±19.66	19.44±19.11	6.07±10.17
cDeepONet	1.02± 1.07	0.48± 0.54	0.47± 0.63	0.06± 0.07
mDeepONet	0.85± 0.85	0.40± 0.43	0.41± 0.52	0.04± 0.03
cPOD-DeepONet	0.56± 0.42	0.24± 0.22	0.25± 0.20	0.06± 0.08
mPOD-DeepONet	0.69± 0.50	0.31± 0.26	0.32± 0.25	0.07± 0.09
mKPCA-DeepONet	0.39± 0.32	0.17± 0.14	0.17± 0.15	0.05± 0.06

(c) Electro-Fluid

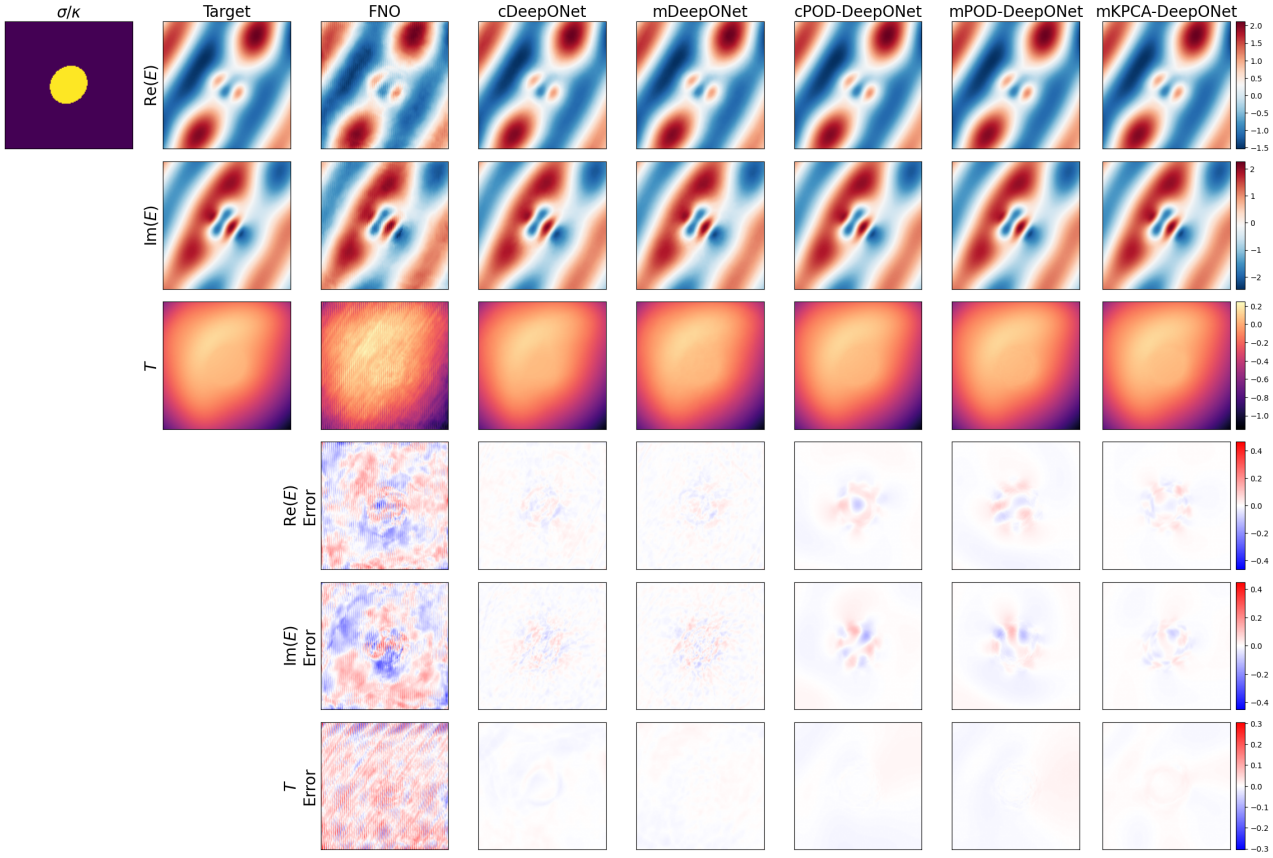
Model	Total	V	$\mathbf{u}_u$	$\mathbf{u}_v$
FNO	0.45± 0.58	0.15± 0.24	0.12± 0.18	0.18± 0.18
cDeepONet	99.49±11.44	0.23± 0.16	44.65± 4.97	54.61± 7.43
mDeepONet	100.97±11.68	0.27± 0.23	45.41± 4.69	55.29± 7.94
cPOD-DeepONet	56.00± 8.25	0.33± 0.39	24.58± 3.74	31.09± 5.22
mPOD-DeepONet	53.80± 7.93	0.35± 0.46	23.70± 3.61	29.74± 5.00
mKPCA-DeepONet	66.75± 9.59	0.54± 0.67	29.36± 4.26	36.85± 6.08

(d) Magneto-Hydrodynamic

Model	Total	$\mathbf{J}_x$	$\mathbf{J}_y$	$\mathbf{J}_z$	$\mathbf{u}_u$	$\mathbf{u}_v$
FNO	31.60±23.72	0.98±0.21	1.03±0.11	27.23±23.71	0.90±0.65	1.47±0.78
cDeepONet	815.13±12.61	0.50±0.13	0.67±0.06	813.76±12.56	0.05±0.05	0.15±0.18
mDeepONet	34.17±21.73	1.10±0.06	1.38±0.09	31.00±21.84	0.22±0.22	0.46±0.34
cPOD-DeepONet	25.15±23.59	0.26±0.18	0.19±0.19	24.59±23.32	0.02±0.07	0.08±0.24
mPOD-DeepONet	25.34±23.74	0.26±0.17	0.19±0.19	24.82±23.47	0.01±0.03	0.05±0.13
mKPCA-DeepONet	24.41±22.71	0.25±0.18	0.18±0.18	23.58±22.46	0.13±0.33	0.27±0.62

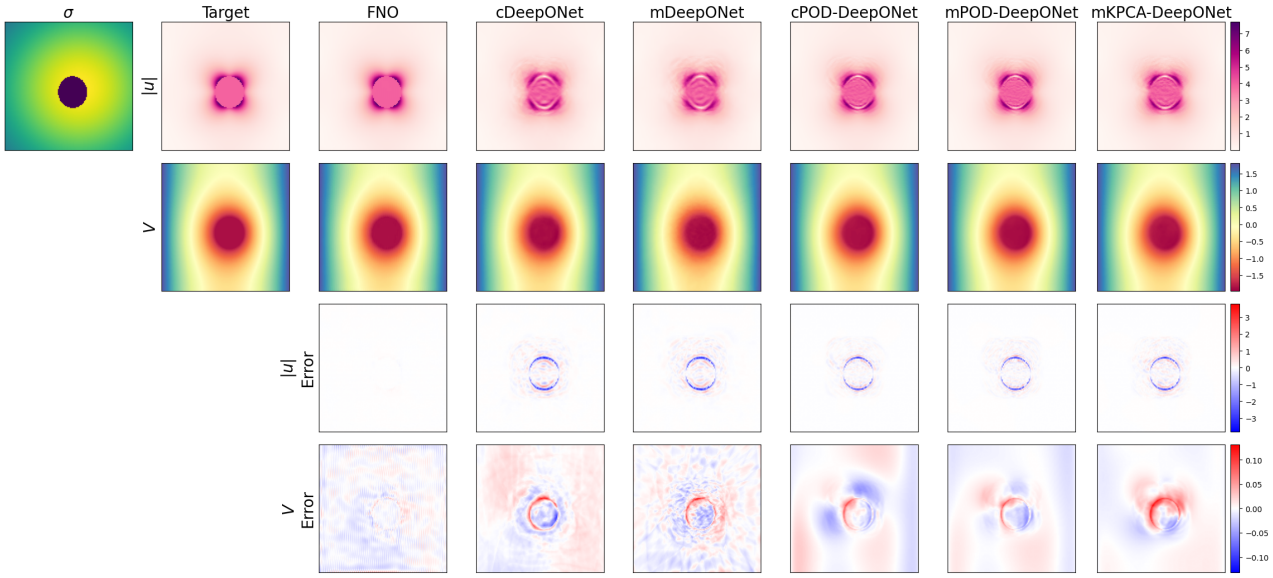


(a) Thermo-Fluid

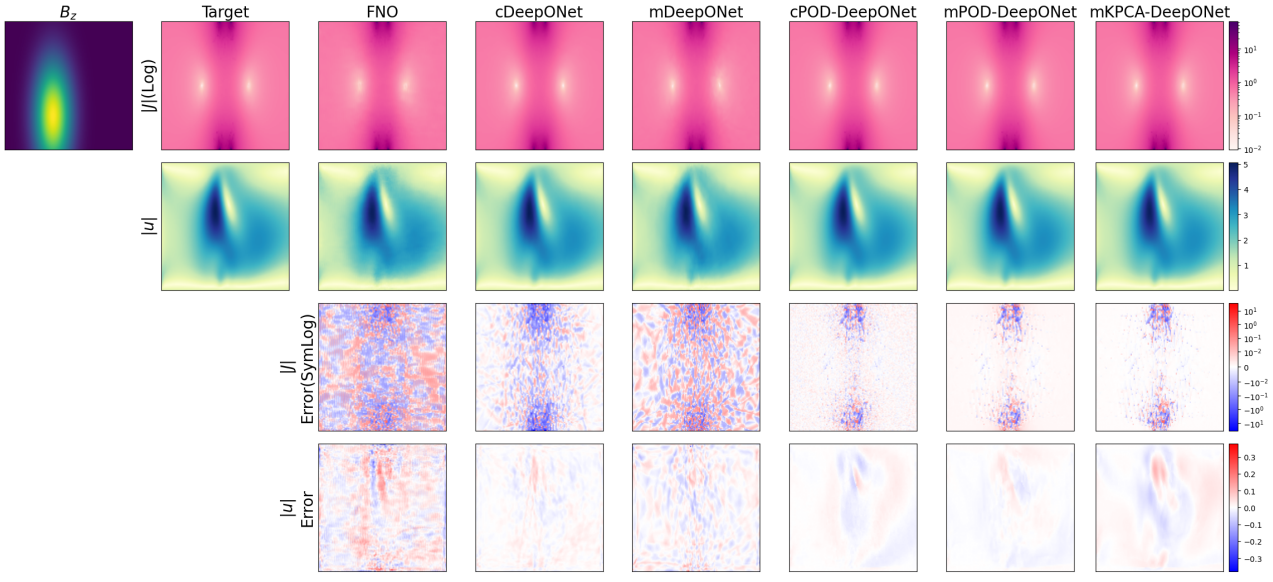


(b) Electro-Thermal

Figure 1: Qualitative comparison on representative test samples from the Thermo-Fluid and Electro-Thermal benchmarks. Subplots display the ground truth, predictions, and error maps for FNO, cDeepONet, mDeepONet, cPOD-DeepONet, mPOD-DeepONet, and mKPCA-DeepONet. For all DeepONet-series models, results correspond to the optimal configuration across hyperparameters basis size p and RBF scale γ.

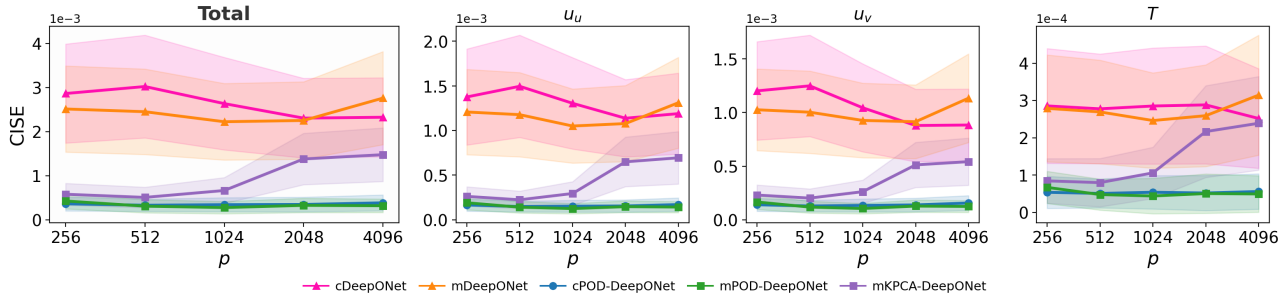


(a) Electro-Fluid

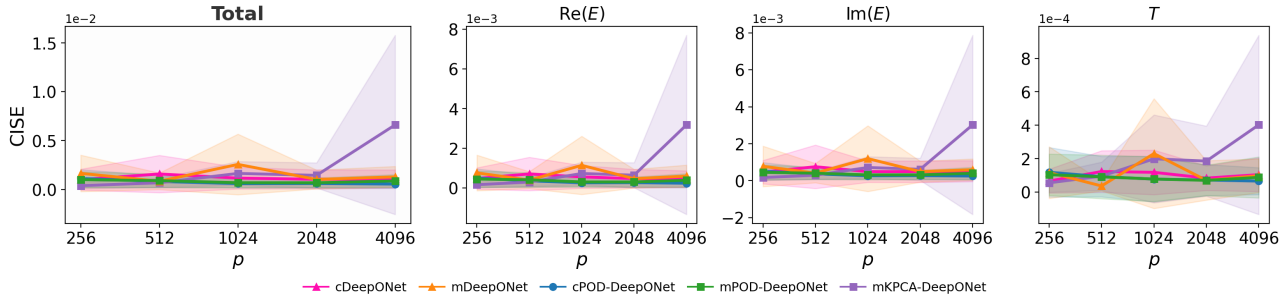


(b) Magneto-Hydrodynamic

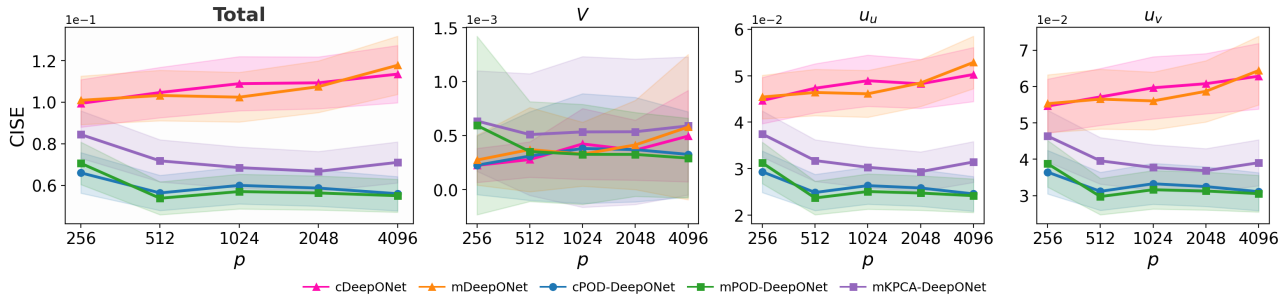
Figure 2: Qualitative comparison on representative test samples from the Electro-Fluid and Magneto-Hydrodynamic. Subplots display the ground truth, predictions, and error maps for FNO, cDeepONet, mDeepONet, cPOD-DeepONet, mPOD-DeepONet, and mKPCA-DeepONet. For all DeepONet-series models, results correspond to the optimal configuration across hyperparameters basis size p and RBF scale γ.



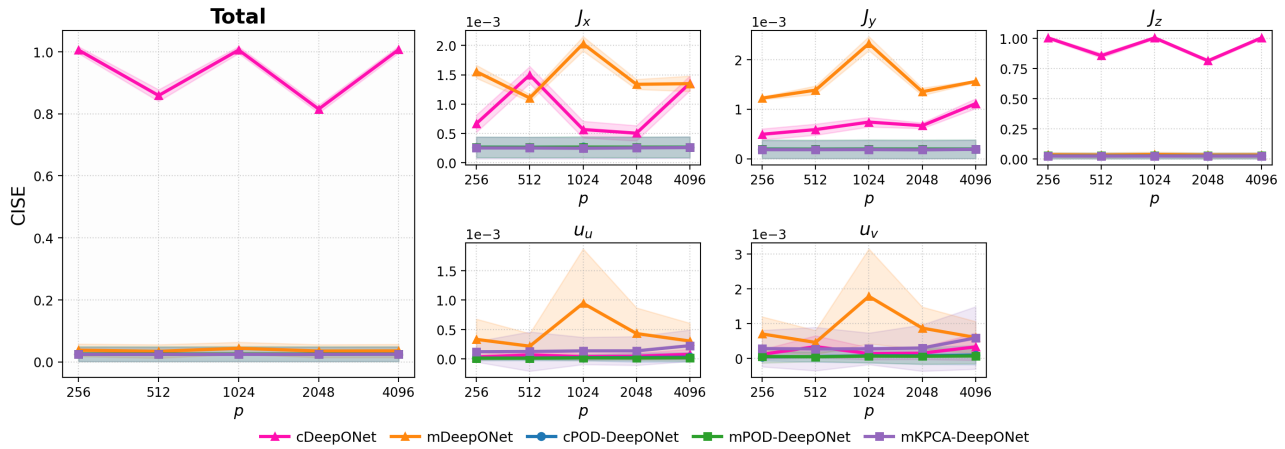
(a) Thermo-Fluid



(b) Electro-Thermal



(c) Electro-Fluid



(d) Magneto-Hydrodynamic

Figure 3: Quantitative performance analysis of cDeepONet, mDeepONet, cPOD-DeepONet, and mPOD-DeepONet with respect to p. The plots display the mean CISE and standard deviation for the Total system error and individual output channels across the multiphysics benchmarks.