CAFE+FNO: FOURIER KERNEL GENERATION VIA MULTIPLICATIVE FEATURE COMPOSITION

Anonymous authors
Paper under double-blind review

ABSTRACT

The Fourier Neural Operator (FNO) learns solution operators of partial differential equations (PDEs) through Fourier-space kernel parameterization, but frequency truncation can limit the learning of high-frequency variations. AM-FNO and SirenFNO generate kernels for all grid modes from spectral coordinates using shared networks, making coordinate encoding and generator design important. Recent work on implicit neural representations (INRs) has proposed constructing frequency interactions through explicit feature composition rather than relying on subsequent MLPs to form them implicitly. Building on this approach, we propose CAFE+FNO, which incorporates Content-Aware Frequency Encoding+ (CAFE+) into Fourier kernel generation. CAFE+ combines Fourier–Chebyshev features through parallel affine branches and a Hadamard product, forming interactions within and across the two feature families. A kernel MLP maps the resulting representation of each normalized spectral coordinate to a complex channel-mixing matrix. Each layer shares its generator across all stored modes, making the number of trainable parameters independent of the number of modes for a fixed architecture. We compare CAFE+FNO with existing FNO variants on five PDE benchmarks and conduct ablation studies on basis configuration, multiplicative composition, and bandwidth learnability. Code and experimental configurations are available at `https://anonymous.4open.science/r/CAFEPlusFNO-70D2`.

1 INTRODUCTION

Partial differential equations (PDEs) describe the spatiotemporal behavior of physical systems, including fluid flow, heat transfer, and elastic deformation. Traditional numerical methods, such as the finite difference method (FDM) and the finite element method (FEM), provide reliable solutions but incur substantial costs in repeated analyses since the PDE must be solved again for each new initial or boundary condition. Neural operators approximate the mapping from an input function to a solution function, enabling much faster prediction than conventional numerical methods without being restricted to a particular data discretization. In particular, the Fourier Neural Operator (FNO) efficiently models global interactions by computing kernel integrals in the Fourier domain (Li et al., 2021).

FNO assigns an independent channel-mixing matrix to each Fourier mode. Parameterizing all modes increases kernel parameter counts and memory costs with the number of modes and the problem dimension. Standard implementations therefore retain a prescribed number of low-frequency modes and discard the others (George et al., 2024).

Amortized Fourier Neural Operator (AM-FNO) and SirenFNO address this frequency truncation by generating full-frequency Fourier kernels with neural networks that take spectral coordinates as input, instead of storing each kernel coefficient independently (Xiao et al., 2024; Shi et al., 2026). In this amortized parameterization, each layer shares its kernel generator across all modes, keeping the number of learnable parameters independent of grid resolution. Consequently, kernel coefficients of each mode are outputs of a shared function $R(\boldsymbol{\xi}_k)$ rather than independent parameters.

This shifts the focus from storing mode-specific kernel coefficients to approximating a frequency-to-kernel function shared across all modes. Full-frequency kernel learning thus requires considering both mode coverage and the representation used to generate mode-dependent complex kernels from spectral coordinates.

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.

To improve this representation, we propose CAFE+FNO, which incorporates the multiplicative feature composition of Content-Aware Frequency Encoding+ (CAFE+) (Ke et al., 2026) into the amortized Fourier kernel generation of AM-FNO. CAFE+FNO constructs learnable interactions among spectral-coordinate features before kernel prediction and uses them to generate a complex channel-mixing kernel for each Fourier mode. It retains amortized parameterization over all stored modes without additional frequency truncation while introducing a learnable feature-composition stage before kernel prediction.

CAFE+FNO achieves the lowest mean error on four of five PDE benchmarks, reducing error relative to SirenFNO by 18.4–60.2% on these tasks; SirenFNO performs better on ReacDiff.

1. We propose CAFE+FNO, which introduces the Fourier–Chebyshev features and multiplicative composition of CAFE+ into spectral-coordinate-to-kernel mapping.
2. We formulate CAFE+ based kernel generation using both dense parameterization and functional tensor parameterizations based on CP, TT, and Tucker decompositions.
3. We compare against existing FNO variants under the same dataset-specific training and evaluation conditions and analyze the empirical effects of the architecture through ablations on the basis configuration, composition block, and bandwidth learning.

2 Related Work

Neural operators. Neural operators learn mappings between function spaces (Bhattacharya et al., 2021; Kovachki et al., 2024). DeepONet (Lu et al., 2021) combines branch and trunk networks, while graph-based neural operators learn kernels between irregularly sampled points (Li et al., 2020a;b). Attention-based methods, including Galerkin Transformer (Cao, 2021), OFormer (Li et al., 2023), GNOT (Hao et al., 2023), and Transolver (Wu et al., 2024), model interactions among discretized points or latent representations. FNO (Li et al., 2021) implements global convolution through mode-wise Fourier transformations, retaining a prescribed set of low-frequency modes. Subsequent work improves multiscale representations (Rahman et al., 2023); U-FNO (Wen et al., 2022) adds a U-Net path to complement high-frequency representations, while F-FNO (Tran et al., 2023) and tensorized FNOs (Kossaifi et al., 2024) reduce costs by factorizing spectral operations and kernel tensors, respectively. AM-FNO (Xiao et al., 2024) and SirenFNO (Shi et al., 2026) generate kernels for all stored Fourier modes using shared networks conditioned on spectral coordinates. For fixed architectures, their parameter counts are independent of the number of modes. AM-FNO employs a KAN (Liu et al., 2025) or a Chebyshev-basis MLP, whereas SirenFNO uses a SIREN with learnable random Fourier features (RFFs). Our work introduces CAFE+ feature composition into this shared kernel parameterization.

Coordinate encodings for implicit neural representations. Coordinate-based implicit neural representations (INRs) map coordinates to continuous signals, but conventional MLPs tend to learn low-frequency components first (Rahaman et al., 2019; Xu et al., 2024). Multiplicative Filter Networks (MFNs) multiply coordinate-dependent sinusoidal or Gabor features with linearly transformed hidden features at each layer (Fathony et al., 2021). BACON constrains output bandwidths (Lindell et al., 2022), while Residual MFNs combine skip connections with frequency initialization for multiscale learning (Shekarforoush et al., 2022). SIREN uses sinusoidal activations (Sitzmann et al., 2020), whereas Fourier feature mapping provides sinusoidal input embeddings (Tancik et al., 2020). Fourier encodings can be sensitive to frequency selection and bandwidth (Hertz et al., 2021; Benbarka et al., 2022; Ma et al., 2025). Nonlinear layers can generate harmonics and combinations of the encoded frequencies (Yüce et al., 2022). CAFE introduces multiplicative feature composition, and CAFE+ adds Chebyshev features for more stable low-frequency representations (Ke et al., 2026). We use this encoding to generate Fourier kernels rather than directly reconstruct signals.

3 Preliminaries

3.1 Fourier Integral Operator

We consider the problem of approximating a solution operator between function spaces from a finite collection of input–output function pairs (Nelsen & Stuart, 2021). Let $\Omega \subset \mathbb{R}^{d_s}$ be a bounded open

computational domain, let $\mathcal{A}$ denote the admissible set of input functions $a\colon \Omega \to \mathbb{R}^{d_a}$, and let $\mathcal{U}$ be a Banach space containing the solution functions $u\colon \Omega \to \mathbb{R}^{d_u}$. Here, $\mathcal{A}$ is a subset of a suitable Banach function space. Assuming that each input $a \in \mathcal{A}$ uniquely determines a solution $u \in \mathcal{U}$, we define the corresponding solution operator as

$$\mathcal{G}^\star\colon \mathcal{A} \to \mathcal{U}, \qquad u = \mathcal{G}^\star(a). \tag{1}$$

Suppose that N input–output function pairs (a_i, u_i), $i = 1, \ldots, N$, are available, where u_i denotes observations of the solution corresponding to input a_i. Our goal is to use these observations to construct an operator $\mathcal{G}\colon \mathcal{A} \to \mathcal{U}$ that approximates $\mathcal{G}^\star$. To this end, we parameterize $\mathcal{G}$ using a neural network with finitely many trainable parameters and learn these parameters by reducing the discrepancy between the predictions $\mathcal{G}(a_i)$ and observations u_i.

To construct this approximation, FNO first maps the input function to a hidden function and then repeatedly applies layers consisting of an integral operator and a pointwise transformation.

Let $\mathcal{V} = L^2(\Omega; \mathbb{R}^{d_v})$ be the hidden function space, and denote the input to layer ℓ by $v^{(\ell)} \in \mathcal{V}$. Here, d_v is the dimension of the hidden representation, which remains the same across all layers.

The Fourier representation below describes convolution on a periodic extension of a rectangular computational domain. This computational periodic representation is distinct from the physical boundary conditions of the underlying PDE. For an integrable matrix-valued kernel κ_ℓ, the Fourier integral operator is given by

$$\begin{aligned}(K_\ell v^{(\ell)})(x) &= \int_\Omega \kappa_\ell(x-y) v^{(\ell)}(y)\,\mathrm{d}y \\ &= \mathcal{F}^{-1}\Big(R_\ell \cdot \mathcal{F}[v^{(\ell)}]\Big)(x),\end{aligned} \tag{2}$$

where $\mathcal{F}$ and $\mathcal{F}^{-1}$ denote the Fourier transform and inverse Fourier transform, respectively, and $R_\ell(k) = \mathcal{F}[\kappa_\ell](k) \in \mathbb{C}^{d_v \times d_v}$ is the complex-valued kernel matrix associated with Fourier mode k.

In the discrete implementation considered hereafter, we denote the finite index set of Fourier modes represented by the computational grid by $E \subset \mathbb{Z}^{d_s}$.

3.2 CAFE+ Encoding

CAFE introduces learnable composition of Fourier features; CAFE+ extends this encoding with Chebyshev features. The concatenated features pass through parallel affine branches, whose outputs are multiplied elementwise before entering a backbone MLP. This constructs interactions within and across the two feature families. Originally designed for coordinate-based signal reconstruction, CAFE+ is used here to encode spectral coordinates for Fourier kernel generation. Section 4 defines the corresponding feature maps and composition. For an input coordinate $\mathbf{x} \in \mathbb{R}^D$ and M Fourier frequency vectors $\{\boldsymbol{\omega}_i\}_{i=1}^M$, where $\boldsymbol{\omega}_i \in \mathbb{R}^D$, the Fourier features are defined as

$$\Phi_{\mathrm{FF}}(\mathbf{x}) = \left[\sin(2\pi\boldsymbol{\omega}_i^\top \mathbf{x}),\, \cos(2\pi\boldsymbol{\omega}_i^\top \mathbf{x})\right]_{i=1}^M. \tag{3}$$

Here, M is the number of Fourier frequency vectors, and $\boldsymbol{\omega}_i$ is a Fourier frequency vector.

Alongside the Fourier features, CAFE+ uses J Chebyshev polynomials of the first kind $T_0, \ldots, T_{J-1}$ along each coordinate axis to construct the Chebyshev features:

$$\Phi_{\mathrm{CF}}(\mathbf{x}) = [T_j(x_d)]_{d=1,\ldots,D;\ j=0,\ldots,J-1}. \tag{4}$$

CAFE+ concatenates the two sets of features, feeds them into N_b linear branches, and combines the branch outputs using a Hadamard product:

$$\Psi(\mathbf{x}) = \bigodot_{i=1}^{N_b} \left\{\mathbf{W}_i \left[\Phi_{\mathrm{FF}}(\mathbf{x}),\, \Phi_{\mathrm{CF}}(\mathbf{x})\right] + b_i\right\}, \tag{5}$$

where $\bigodot$ denotes Hadamard product.

This multiplicative composition establishes interactions between Fourier and Chebyshev bases at the encoding stage, and the resulting $\Psi(\mathbf{x})$ is subsequently used as the input to the backbone MLP.

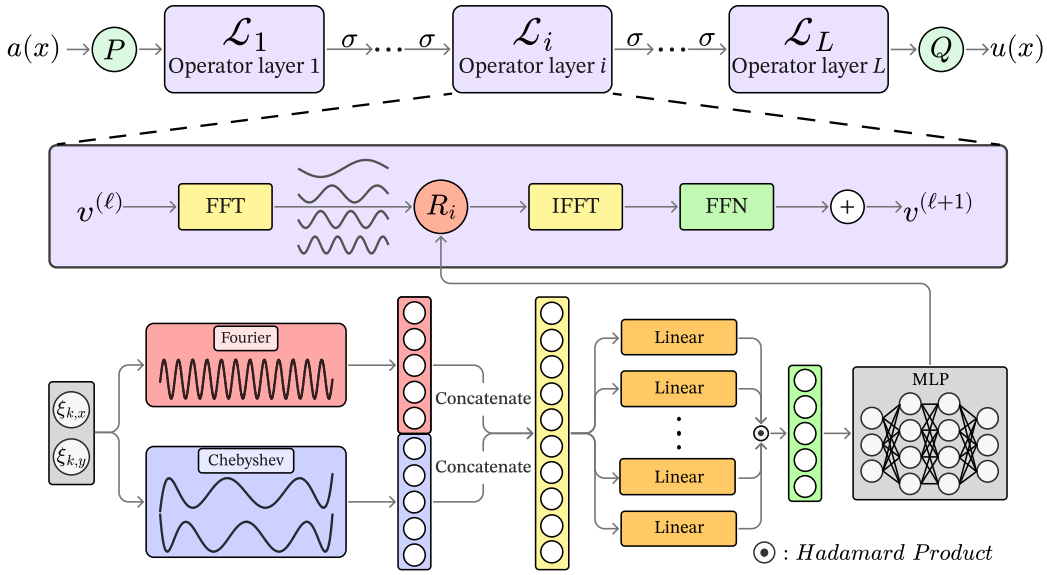

Figure 1: Architecture of CAFE+FNO in two dimensions. The input is lifted by P, processed by L operator layers, and projected by Q to the predicted solution. Within each layer, random Fourier features and Chebyshev features of the normalized spectral coordinate $\boldsymbol{\xi}_k$ are concatenated and passed through parallel linear branches. Their outputs are combined via a Hadamard product, and a kernel MLP generates the mode-specific complex channel-mixing matrix used in the spectral convolution.

4 Method

We propose CAFE+FNO, which uses the multiplicative feature composition of CAFE+ as the input representation for a Fourier kernel generator. Whereas the original CAFE+ constructs a representation for reconstructing signal values from coordinates, our approach generates the kernel of each Fourier mode from normalized spectral coordinates. We first describe the dense kernel generator; extensions based on CP, TT, and Tucker factorizations are presented in Appendix C.

4.1 Fourier–Chebyshev Feature Encoding

Let $\boldsymbol{\xi}_k \in [-1,1]^{d_s}$ denote the normalized spectral coordinate corresponding to each Fourier mode $k \in E$. The coordinate $\boldsymbol{\xi}_k$ is the input to the kernel generator; the coordinate normalization is specified in Appendix B. In each layer ℓ, we independently sample the entries of a Gaussian matrix G_ℓ once from the standard normal distribution to map this coordinate to m pairs of Fourier features:

$$G_\ell \in \mathbb{R}^{d_s \times m}, \qquad (G_\ell)_{ij} \sim \mathcal{N}(0,1). \tag{6}$$

Here, $d_s \in \{1,2\}$ is the spatial dimension. The sampled G_ℓ remains fixed during training, and a Fourier projection matrix $B_\ell = \rho_\ell G_\ell$ is constructed using $\rho_\ell > 0$. The matrix $B_\ell \in \mathbb{R}^{d_s \times m}$ maps the normalized spectral coordinate to RFF phases in the ℓth operator layer.

Our implementation defines the RFF phase as $\pi B_\ell^\top \boldsymbol{\xi}_k$, with the frequency scale incorporated into B_ℓ.

The RFF encoding of the spectral coordinate in layer ℓ is defined as

$$\Phi_{\mathrm{FF}}^{(\ell)}(\boldsymbol{\xi}_k; \rho_\ell) = \begin{bmatrix} \cos(\pi B_\ell^\top \boldsymbol{\xi}_k) \\ \sin(\pi B_\ell^\top \boldsymbol{\xi}_k) \end{bmatrix} \in \mathbb{R}^{2m}. \tag{7}$$

Sine and cosine are applied elementwise, and their outputs are concatenated into a $2m$-dimensional feature vector. The parameter ρ_ℓ determines the frequency scale of the spectral-coordinate encoding by controlling the magnitude of the Gaussian projection.

Since CAFE+ uses Chebyshev features alongside Fourier features, we also compute n first-kind Chebyshev basis functions for each coordinate axis. Writing $t_n(s) = [T_0(s), \dots, T_{n-1}(s)]^\top \in \mathbb{R}^n$, we obtain

$$\Phi_{\mathrm{CF}}(\boldsymbol{\xi}_k) = \left[t_n(\xi_{k,1})^\top, \dots, t_n(\xi_{k,d_s})^\top\right]^\top \in \mathbb{R}^{d_s n}. \tag{8}$$

The Fourier and Chebyshev features are then concatenated:

$$\boldsymbol{\gamma}_\ell(\boldsymbol{\xi}_k) = \left[\Phi_{\mathrm{CF}}(\boldsymbol{\xi}_k)^\top,\ \Phi^{(\ell)}_{\mathrm{FF}}(\boldsymbol{\xi}_k; \rho_\ell)^\top\right]^\top. \tag{9}$$

The resulting feature is $\boldsymbol{\gamma}_\ell(\boldsymbol{\xi}_k) \in \mathbb{R}^{2m+d_s n}$, with embedding dimension $2m + d_s n$.

4.2 CAFE+ Multiplicative Frequency Composition

CAFE+FNO feeds $\boldsymbol{\gamma}_\ell(\boldsymbol{\xi}_k)$ into N_b parallel branches before passing the resulting representation to the kernel MLP. Each branch applies an affine transformation with a bias, referred to hereafter as a linear branch:

$$h_{\ell,r}(\boldsymbol{\xi}_k) = \widetilde{W}_{\ell,r}\boldsymbol{\gamma}_\ell(\boldsymbol{\xi}_k) + \widetilde{b}_{\ell,r}, \qquad r = 1, \dots, N_b, \tag{10}$$

where $\widetilde{W}_{\ell,r} \in \mathbb{R}^{d_z \times (2m+d_s n)}$ and $\widetilde{b}_{\ell,r} \in \mathbb{R}^{d_z}$. The branch outputs are combined by a Hadamard product:

$$\mathbf{z}_\ell(\boldsymbol{\xi}_k) = \bigodot_{r=1}^{N_b} h_{\ell,r}(\boldsymbol{\xi}_k). \tag{11}$$

Since each branch is an affine transformation of $\boldsymbol{\gamma}_\ell(\boldsymbol{\xi}_k)$, each component of $\mathbf{z}_\ell(\boldsymbol{\xi}_k)$ can be expressed as a polynomial of degree at most N_b in the input feature components. In particular, two branches yield quadratic interactions together with linear and constant terms, whose coefficients are parameterized by products of branch weights and biases.

To examine the frequency structure of these interactions, we fix layer ℓ and write the Fourier projection matrix as $B_\ell = [\boldsymbol{\omega}_{\ell,1}, \dots, \boldsymbol{\omega}_{\ell,m}]$. Here, $\boldsymbol{\omega}_{\ell,i}$ is an encoding frequency vector for the spectral coordinate $\boldsymbol{\xi}_k$, distinct from the Fourier mode k of the hidden function. We define the phase associated with each column vector as $\theta_{\ell,i} = \pi\boldsymbol{\omega}_{\ell,i}^\top\boldsymbol{\xi}_k$ for $i = 1, \dots, m$. For this example, we abbreviate $\theta_i := \theta_{\ell,i}$ and $\theta_j := \theta_{\ell,j}$. We simplify the same component of the two branch output vectors as

$$h_1 = \alpha_1 \cos\theta_i + \beta_1 \sin\theta_i, \qquad h_2 = \alpha_2 \cos\theta_j + \beta_2 \sin\theta_j. \tag{12}$$

Applying the product-to-sum identities gives

$$\begin{aligned} h_1 h_2 = \tfrac{1}{2}\big[&(\alpha_1\alpha_2 + \beta_1\beta_2)\cos(\theta_i - \theta_j) + (\alpha_1\alpha_2 - \beta_1\beta_2)\cos(\theta_i + \theta_j) \\ &+ (\beta_1\alpha_2 - \alpha_1\beta_2)\sin(\theta_i - \theta_j) + (\beta_1\alpha_2 + \alpha_1\beta_2)\sin(\theta_i + \theta_j)\big]. \end{aligned} \tag{13}$$

Since

$$\theta_i \pm \theta_j = \pi(\boldsymbol{\omega}_{\ell,i} \pm \boldsymbol{\omega}_{\ell,j})^\top\boldsymbol{\xi}_k, \tag{14}$$

the representation after the Hadamard product can include, in addition to the initial RFF basis, basis frequencies corresponding to the sums $\boldsymbol{\omega}_{\ell,i} + \boldsymbol{\omega}_{\ell,j}$ and differences $\boldsymbol{\omega}_{\ell,i} - \boldsymbol{\omega}_{\ell,j}$. The coefficient of each component is determined by the learnable linear-branch weights $\alpha_1, \beta_1, \alpha_2, \beta_2$, allowing the network to selectively adjust the frequency interactions needed for kernel generation.

Products of Chebyshev components similarly satisfy

$$T_p(x)T_q(x) = \tfrac{1}{2}\big[T_{p+q}(x) + T_{|p-q|}(x)\big], \qquad 0 \le p, q \le n-1. \tag{15}$$

Thus, multiplying Chebyshev features along the same coordinate axis can construct higher-order polynomial components absent from the initial encoding, while also recombining lower-order components.

Furthermore, since both feature types enter the same branches, products of branch outputs can contain Fourier–Chebyshev cross terms such as $T_p(\xi_{k,a})\cos\theta_{\ell,i}$ and $T_p(\xi_{k,a})\sin\theta_{\ell,i}$. These terms form sinusoidal features whose amplitudes are modulated by polynomials. Consequently, $\mathbf{z}_\ell(\boldsymbol{\xi}_k)$ represents both interactions within each basis and cross interactions between the two bases. The kernel MLP in the next subsection uses this representation to generate a complex-valued kernel matrix for each mode.

The frequencies composed here are encoding frequencies with respect to the spectral coordinate $\boldsymbol{\xi}_k$. This operation neither directly multiplies different Fourier coefficients of the PDE solution nor adds new Fourier modes to the computational grid. The purpose of the operation is to provide basis interactions for representing the mapping from $\boldsymbol{\xi}_k$ to the kernel matrix. Its effect on solution prediction is evaluated in the experiments in Section 5.

4.3 Complex Fourier Kernel Generation

Let $v^{(\ell)} : \Omega \to \mathbb{R}^{d_v}$ be the hidden representation at layer ℓ. The CAFE+ mode representation $\mathbf{z}_\ell(\boldsymbol{\xi}_k) \in \mathbb{R}^{d_z}$ constructed above is passed to a complex matrix-valued kernel generator $\mathcal{M}_\ell : \mathbb{R}^{d_z} \to \mathbb{C}^{d_v \times d_v}$. In practice, we use a single real-valued MLP $f_\ell : \mathbb{R}^{d_z} \to \mathbb{R}^{2d_v^2}$. Its first and last d_v^2 output components are split into real and imaginary parts, respectively, and reshaped into $d_v \times d_v$ matrices:

$$\begin{bmatrix} r_\ell^{\mathrm{Re}}(k) \\ r_\ell^{\mathrm{Im}}(k) \end{bmatrix} = f_\ell\big(\mathbf{z}_\ell(\boldsymbol{\xi}_k)\big), \qquad r_\ell^{\mathrm{Re}}(k), r_\ell^{\mathrm{Im}}(k) \in \mathbb{R}^{d_v^2}. \tag{16}$$

The kernel corresponding to Fourier mode k is therefore defined as

$$R_\ell(k) = \mathcal{M}_\ell\big(\mathbf{z}_\ell(\boldsymbol{\xi}_k)\big) = R_\ell^{\mathrm{Re}}(k) + \mathrm{i} R_\ell^{\mathrm{Im}}(k). \tag{17}$$

The same $\mathcal{M}_\ell$ and encoding parameters are shared across all Fourier modes within layer ℓ, whereas different layers use independent parameters.

The kernel generator takes only spectral coordinates as input and is not conditioned on an individual input function a. Thus, after training, the kernel generated for a given mode on the same grid and in the same layer is identical across input samples. Here, content-aware composition means that end-to-end training through the PDE solution prediction error forms feature combinations suited to the data. It does not imply supervision with separate ground-truth kernels or the generation of a different kernel for each input sample.

Consequently, when the generator architecture and feature dimensions are fixed, the number of trainable parameters is independent of the size of the mode set E. However, the cost of evaluating and storing kernels over all modes increases with the grid size. We apply a Hermitian projection to the generated complex kernels to maintain consistency with the inverse Fourier transform of a real-valued hidden function:

$$\widetilde{R}_\ell = \Pi_{\mathcal{H}}(R_\ell), \qquad \widetilde{R}_\ell(-k) = \overline{\widetilde{R}_\ell(k)}. \tag{18}$$

This symmetry condition applies to the completed spectrum. The specific projection used for the rFFT storage array is described in Appendix B.

Finally, $\widetilde{R}_\ell(k)$ acts linearly on the Fourier coefficients of the hidden representation at each mode:

$$\mathcal{F}[K_\ell v^{(\ell)}](k) = (\widetilde{R}_\ell \cdot \widehat{v}^{(\ell)})(k), \qquad \widehat{v}^{(\ell)} = \mathcal{F}[v^{(\ell)}]. \tag{19}$$

4.4 CAFE+FNO Architecture

CAFE+FNO follows the basic operator architecture of AM-FNO, replacing only the kernel generator in each Fourier layer with the CAFE+ based parameterization defined above. The complete model is written as

$$\mathcal{G} := Q \circ \mathcal{L}_L \circ \mathcal{L}_{L-1} \circ \cdots \circ \mathcal{L}_1 \circ P, \tag{20}$$

with $v^{(1)} = P(a)$ and $\widehat{u} = Q\big(v^{(L+1)}\big)$. The lifting operator $P : \mathcal{A} \to \mathcal{V}$ constructs a d_v-dimensional hidden function from the values of the input function and the computational coordinates. Each operator layer $\mathcal{L}_\ell$ updates the hidden function, and the projection operator $Q : \mathcal{V} \to \mathcal{U}$ maps the final hidden function to the predicted solution function.

Let $\widetilde{R}_\ell(k)$ denote the generated complex kernel after Hermitian projection. The ℓth layer is computed as

$$\begin{aligned} \widetilde{v}^{(\ell)} &= \mathcal{F}^{-1}\Big(\widetilde{R}_\ell \cdot \mathcal{F}[v^{(\ell)}]\Big), \\ v^{(\ell+1)}(x) &= \sigma\Big(v^{(\ell)}(x) + \mathrm{FFN}_\ell\big(\widetilde{v}^{(\ell)}(x)\big)\Big). \end{aligned} \tag{21}$$

Here, $\mathrm{FFN}_\ell(y) = \widehat{W}_\ell^{(2)}\sigma(\widehat{W}_\ell^{(1)}y + b_\ell^{(1)}) + b_\ell^{(2)}$, and σ is the activation function. In the final operator layer, only the activation after the residual addition is omitted; the activation within the FFN is retained. The final hidden representation is mapped to the output through the same projection.

Table 1: Description of the benchmark datasets.

Dataset	Input	Target	Resolution	Train/test	Source
Darcy	$a(x)$	$u(x)$	128×128	1000/200	NeuralOperator
NS	$u(x, t_{\text{in}})$	$u(x, t_{\text{out}})$	128×128	1000/200	NeuralOperator
Burgers	$[u(x, t_j)]_{j=0}^{9}$	$[u(x, t_j)]_{j=10}^{19}$	1024	1000/200	PDEBench
Airfoil	(X, Y)	$u(x)$	221×51	1000/200	Geo-FNO
ReacDiff	$[u(x, t_j)]_{j=0}^{9}$	$[u(x, t_j)]_{j=10}^{19}$	1024	1000/200	PDEBench

5 EXPERIMENTS

This section presents the experimental results. All experiments were conducted on a desktop computer equipped with an AMD Ryzen 9800XD 8-Core CPU, an NVIDIA GeForce RTX 5080 GPU, and 64GB of Samsung DDR5-5600 RAM (16GB x 4).

5.1 BENCHMARKS AND BASELINES

We compare CAFE+FNO with FNO and several established variants: U-FNO, TFNO-CP with CP decomposition, AM-FNO with an MLP, and SirenFNO with a SIREN hypernetwork. We also compare CP-, TT-, and Tucker-based variants of both SirenFNO and CAFE+FNO.

Table 1 lists the benchmark datasets. Darcy flow (NeuralOperator Team, 2021a) and Navier–Stokes (NeuralOperator Team, 2021b) are obtained from the NeuralOperator library; Burgers and reaction–diffusion are from PDEBench (Takamoto et al., 2024; 2022); and Airfoil is from the Geo-FNO benchmark (Li et al., 2024).

For Airfoil, the publicly available materials were insufficient to reconstruct the corresponding U-FNO implementation consistently, and the fixed baseline implementation required additional compatibility modifications. We therefore exclude U-FNO from this benchmark and indicate the corresponding entry with a dash in Table 2.

Appendix A provides dataset-specific preprocessing, prediction protocols, training losses, and evaluation settings.

5.2 EXPERIMENTAL SETUP

For a fair comparison, we retrained FNO, U-FNO, TFNO-CP, AM-FNO, SirenFNO, and CAFE+FNO using the same data splits, preprocessing, and training pipeline. Each baseline architecture follows the dataset-specific settings of its public implementation. For Airfoil, which is not included in the original SirenFNO benchmarks, we used a separately specified architecture configuration.

All models were trained for 500 epochs using AdamW, with an initial learning rate of 10^{-3} and weight decay of 10^{-4}. The learning rate was reduced using cosine annealing. We used neither validation-based model selection nor early stopping; all results were evaluated using the final-epoch model. The main evaluation metric is the mean sample-wise relative L_2 error,

$$\mathcal{L}_{\text{rel}} = \frac{1}{N}\sum_{i=1}^{N}\frac{\|\mathcal{G}(a_i) - u_i\|_2}{\|u_i\|_2}. \tag{22}$$

Here, N is the number of evaluation samples, and u_i and $\mathcal{G}(a_i)$ are the targets and predictions expressed in the dataset-specific evaluation space. For Burgers and ReacDiff, sample-wise relative L_2 errors are computed by concatenating the temporal and spatial values over the entire prediction rollout. The other benchmarks are evaluated on the predicted target fields.

CAFE+FNO, SirenFNO, and AM-FNO generate Fourier kernels for all stored FFT modes, while FNO and TFNO-CP use the frequency truncation of their respective public configurations. Each CAFE+FNO layer uses an independent Gaussian matrix G_ℓ. For Table 2, ρ_ℓ was fixed to one

Table 2: Final test relative L_2 errors ($\times 10^{-3}$) at epoch 500: mean $\pm$ sample standard deviation over five seeds. Bold marks the lowest mean per dataset. AM-FNO uses the MLP variant; U-FNO was excluded a priori on Airfoil (—). Evaluation details appear in Appendix A.

Model	**Darcy** 128×128	**NS** 128×128	**Burgers** 1024	**Airfoil** 221×51	**ReacDiff** 1024
FNO	65.40 ± 4.18	56.29 ± 0.93	11.30 ± 0.08	6.17 ± 0.17	4.18 ± 0.15
U-FNO	47.05 ± 1.83	57.69 ± 0.78	10.02 ± 0.21	—	9.13 ± 5.77
TFNO-CP	47.34 ± 1.28	35.14 ± 0.96	10.27 ± 0.49	7.06 ± 0.48	4.07 ± 0.26
AM-FNO(MLP)	62.02 ± 12.90	29.94 ± 1.83	18.73 ± 3.20	6.29 ± 0.68	1758.09 ± 3843.63
SirenFNO	24.34 ± 1.24	43.24 ± 2.02	5.86 ± 0.41	5.76 ± 0.24	$\mathbf{2.36} \pm 0.78$
CAFE+FNO	$\mathbf{13.70} \pm 0.13$	$\mathbf{17.20} \pm 0.32$	$\mathbf{4.13} \pm 0.37$	$\mathbf{4.70} \pm 0.12$	3.80 ± 0.27
Factorized SirenFNO					
CP	46.09 ± 21.71	51.46 ± 7.29	8.91 ± 0.29	8.12 ± 1.68	4.37 ± 1.70
TT	178.78 ± 257.76	36.04 ± 5.53	7.72 ± 0.21	6.02 ± 0.26	2.63 ± 0.23
Tucker	64.06 ± 27.05	46.70 ± 12.24	8.11 ± 0.43	8.18 ± 3.81	2.92 ± 0.09
Factorized CAFE+FNO					
CP	30.05 ± 2.57	68.91 ± 26.94	8.73 ± 0.84	8.68 ± 2.52	4.15 ± 0.48
TT	27.77 ± 1.82	42.50 ± 12.94	8.48 ± 0.45	7.06 ± 0.61	4.09 ± 0.42
Tucker	27.96 ± 2.22	39.58 ± 5.92	8.97 ± 0.47	8.13 ± 1.22	3.72 ± 0.54

throughout training. The effect of learning the bandwidth is examined in Section 5.4, and the CP, TT, and Tucker results are reported in Section 5.3.

Each experiment was repeated with five random seeds, $\{0, 42, 73, 108, 202\}$. We report the mean $\pm$ standard deviation of the final test errors. Dataset-specific batch sizes, preprocessing, training losses, and evaluation protocols are provided in Appendix A, CAFE+FNO architecture settings in Appendix D, and complete configurations in the accompanying code.

5.3 Main results

Table 2 shows that dense CAFE+FNO achieves the lowest mean error on Darcy, NS, Burgers, and Airfoil. Relative to SirenFNO, errors decrease by 43.7%, 60.2%, 29.5%, and 18.4%, respectively. On ReacDiff, SirenFNO performs better (2.36 versus 3.80, $\times 10^{-3}$). CAFE+FNO also improves on FNO and AM-FNO across all five datasets, although AM-FNO's ReacDiff results show substantial variability.

Relative to dense CAFE+FNO, CP, TT, and Tucker variants reduce parameter counts by 76.7–86.1%, 68.7–74.7%, and 0.5–78.0%, respectively (Table 6). Factorized CAFE+FNO variants have higher mean errors than the dense model, except for Tucker on ReacDiff. All three outperform their SirenFNO counterparts on Darcy; results elsewhere are mixed. To target comparable parameter budgets, we use a 12-dimensional input to each CAFE+ factor-generator MLP, compared with 16 or 32 in the corresponding SirenFNO models. Relative to dense CAFE+FNO, both the initial basis counts and the MLP input dimension are reduced. These changes may also affect accuracy; their effects are not isolated from tensor factorization in this comparison.

5.4 Ablation experiments

We analyze how the combination of Fourier and Chebyshev features, multiplicative feature composition through linear branches and a Hadamard product, and RFF bandwidth learnability affect prediction accuracy. Results are presented in Table 3. Since removing components may change feature dimensions and parameter counts, each comparison is interpreted as the empirical effect of the corresponding overall configuration.

The combined Fourier–Chebyshev encoding yields lower mean errors than either single-basis variant on all five datasets, supporting its effectiveness for Fourier kernel generation.

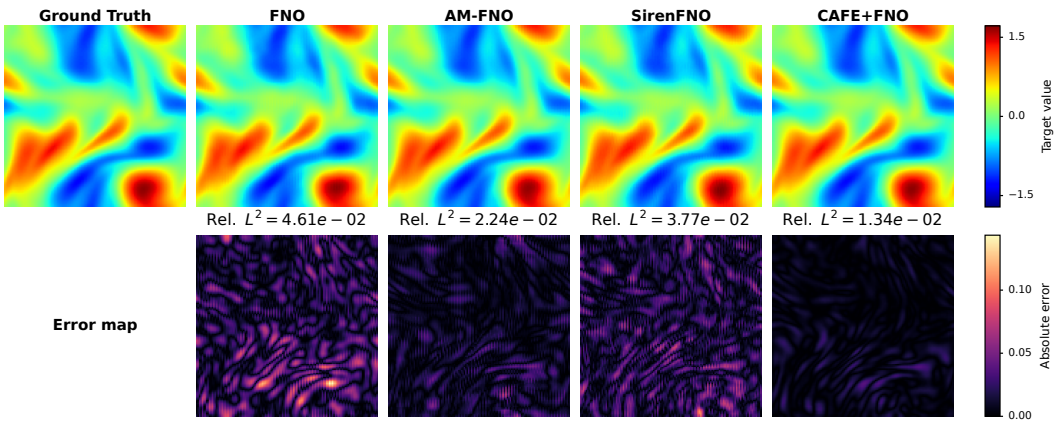

Figure 2: Comparison of FNO, AM-FNO, SirenFNO, and CAFE+FNO on a Navier–Stokes (NS) sample using trained checkpoints. The top row shows the ground-truth vorticity and model predictions; the bottom row shows pointwise absolute errors. Relative L_2 errors are reported below the predictions.

Table 3: CAFE+FNO ablations: relative L_2 errors ($\times 10^{-3}$). Evaluation and reporting conventions follow Table 2. Bold marks the lowest unrounded mean per dataset. *: no composition block, with the kernel MLP input dimension matched to the default model (16 for ReacDiff and 32 otherwise). RFF/Cheb: single-basis variants retaining the composition block. †: learnable bandwidth with a fixed Gaussian matrix.

Model	**Darcy** 128×128	**NS** 128×128	**Burgers** 1024	**Airfoil** 221×51	**ReacDiff** 1024
CAFE+FNO	**13.63** $\pm$ 0.12	17.25 $\pm$ 0.29	**4.11** $\pm$ 0.37	**4.69** $\pm$ 0.10	3.80 $\pm$ 0.27
CAFE+FNO*	27.28 $\pm$ 2.70	38.53 $\pm$ 1.46	5.14 $\pm$ 0.34	17.58 $\pm$ 11.38	3.87 $\pm$ 0.28
CAFE+FNO$_{\text{RFF}}$	15.07 $\pm$ 0.71	19.76 $\pm$ 0.20	4.91 $\pm$ 0.39	5.36 $\pm$ 0.22	4.48 $\pm$ 1.14
CAFE+FNO$_{\text{Cheb}}$	14.46 $\pm$ 0.71	17.81 $\pm$ 0.14	4.16 $\pm$ 0.24	4.87 $\pm$ 0.30	4.00 $\pm$ 0.24
CAFE+FNO†	13.69 $\pm$ 0.14	**17.22** $\pm$ 0.26	4.13 $\pm$ 0.37	4.69 $\pm$ 0.12	**3.65** $\pm$ 0.29

With matched kernel MLP input dimensions (16 for ReacDiff and 32 otherwise), the default model achieves lower mean errors than CAFE+FNO*, which directly uses Fourier–Chebyshev features. This supports the utility of CAFE+ encoding at the same MLP input dimension. Finally, CAFE+FNO†, with learnable RFF bandwidth ρ_ℓ, yields lower mean errors than the fixed-bandwidth default on NS and ReacDiff, but higher errors on Darcy and Burgers. Airfoil errors are equal at the reported precision.

6 Conclusion

We proposed CAFE+FNO, which introduces explicit, learnable feature composition into Fourier kernel generation. Its Fourier–Chebyshev encoding uses parallel affine branches and a Hadamard product to construct interactions within and across feature families before the kernel MLP. The dense model achieves the lowest mean error on four of five benchmarks, while SirenFNO performs better on ReacDiff. Ablations favor the combined basis over either basis alone and show lower mean errors with composition than with direct features at matched kernel MLP input dimensions. Together, these results support explicit spectral-coordinate feature composition as an effective design choice for Fourier kernel generation. Factorized variants use fewer parameters than dense CAFE+FNO, generally at some cost in accuracy. However, this comparison does not isolate tensor factorization from the accompanying reductions in basis counts and MLP input dimensions. Future work will focus on extending CAFE+FNO to a broader range of PDE problems and higher-dimensional spatial domains.

AI USE STATEMENT

We used GPT-6 Astra to assist with manuscript translation, grammar correction, and LaTeX syntax. The authors take full responsibility for the final content of the manuscript.

REFERENCES

Nuri Benbarka, Timon Höfer, Hamd ul-Moqeet Riaz, and Andreas Zell. Seeing Implicit Neural Representations as Fourier Series. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 2041–2050, 2022. URL https://openaccess.thecvf.com/content/WACV2022/html/Benbarka_Seeing_Implicit_Neural_Representations_As_Fourier_Series_WACV_2022_paper.html.

Kaushik Bhattacharya, Bamdad Hosseini, Nikola B. Kovachki, and Andrew M. Stuart. Model Reduction and Neural Networks for Parametric PDEs. *The SMAI Journal of Computational Mathematics*, 7:121–157, 2021. doi: 10.5802/smai-jcm.74.

Shuhao Cao. Choose a Transformer: Fourier or Galerkin. In *Advances in Neural Information Processing Systems*, volume 34, pp. 24924–24940, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/d0921d442ee91b896ad95059d13df618-Abstract.html.

Rizal Fathony, Anit Kumar Sahu, Devin Willmott, and J. Zico Kolter. Multiplicative Filter Networks. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=OmtmcPkkhT.

Robert Joseph George, Jiawei Zhao, Jean Kossaifi, Zongyi Li, and Anima Anandkumar. Incremental Spatial and Spectral Learning of Neural Operators for Solving Large-Scale PDEs. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=xI6cPQObp0.

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 *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 12556–12569, 2023. URL https://proceedings.mlr.press/v202/hao23c.html.

Amir Hertz, Or Perel, Raja Giryes, Olga Sorkine-Hornung, and Daniel Cohen-Or. SAPE: Spatially-Adaptive Progressive Encoding for Neural Optimization. In *Advances in Neural Information Processing Systems*, volume 34, pp. 8820–8832, 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/hash/4a06d868d044c50af0cf9bc82d2fc19f-Abstract.html.

Junbo Ke, Yangyang Xu, Chao Wang, and You-Wei Wen. Content-Aware Frequency Encoding for Implicit Neural Representations with Fourier-Chebyshev Features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3646–3655, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Ke_Content-Aware_Frequency_Encoding_for_Implicit_Neural_Representations_with_Fourier-Chebyshev_Features_CVPR_2026_paper.html.

Tamara G. Kolda and Brett W. Bader. Tensor Decompositions and Applications. *SIAM Review*, 51 (3):455–500, 2009. doi: 10.1137/07070111X.

Jean Kossaifi, Nikola B. Kovachki, Kamyar Azizzadenesheli, and Anima Anandkumar. Multi-Grid Tensorized Fourier Neural Operator for High-Resolution PDEs. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=AWiDlO63bH.

Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural Operator: Learning Maps Between Function Spaces. *arXiv preprint arXiv:2108.08481v6*, 2024. URL https://arxiv.org/abs/2108.08481v6.

Zijie Li, Kazem Meidani, and Amir Barati Farimani. Transformer for Partial Differential Equations' Operator Learning. *Transactions on Machine Learning Research*, 2023. URL https://openreview.net/forum?id=EPPqt3uERT.

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. URL https://arxiv.org/abs/2003.03485.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Multipole Graph Neural Operator for Parametric Partial Differential Equations. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6755–6766, 2020b. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/4b21cf96d4cf612f239a6c322b10c8fe-Paper.pdf.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier Neural Operator for Parametric Partial Differential Equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=c8P9NQVtmnO.

Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. Fourier Neural Operator with Learned Deformations for PDEs on General Geometries. *arXiv preprint arXiv:2207.05209v2*, 2024. URL https://arxiv.org/abs/2207.05209v2. Code and datasets: https://github.com/neuraloperator/Geo-FNO#datasets.

David B. Lindell, Dave Van Veen, Jeong Joon Park, and Gordon Wetzstein. BACON: Band-Limited Coordinate Networks for Multiscale Scene Representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16252–16262, 2022. URL https://openaccess.thecvf.com/content/CVPR2022/html/Lindell_BACON_Band-Limited_Coordinate_Networks_for_Multiscale_Scene_Representation_CVPR_2022_paper.html.

Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y. Hou, and Max Tegmark. KAN: Kolmogorov–Arnold Networks. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Ozo7qJ5vZi.

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. doi: 10.1038/s42256-021-00302-5.

Mingze Ma, Qingtian Zhu, Yifan Zhan, Zhengwei Yin, Hongjun Wang, and Yinqiang Zheng. Robustifying Fourier Features Embeddings for Implicit Neural Representations. *arXiv preprint arXiv:2502.05482*, 2025. URL https://arxiv.org/abs/2502.05482.

Nicholas H. Nelsen and Andrew M. Stuart. The Random Feature Model for Input-Output Maps between Banach Spaces. *SIAM Journal on Scientific Computing*, 43(5):A3212–A3243, 2021. doi: 10.1137/20M133957X.

NeuralOperator Team. Darcy Flow Dataset. Zenodo, 2021a. URL https://doi.org/10.5281/zenodo.12784353. Dataset, version v2.

NeuralOperator Team. Navier-Stokes Dataset. Zenodo, 2021b. URL https://doi.org/10.5281/zenodo.12825163. Dataset, version v2.

I. V. Oseledets. Tensor-Train Decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011. doi: 10.1137/090752286.

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the Spectral Bias of Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pp. 5301–5310, 2019. URL https://proceedings.mlr.press/v97/rahaman19a.html.

Md Ashiqur Rahman, Zachary E. Ross, and Kamyar Azizzadenesheli. U-NO: U-shaped Neural Operators. *Transactions on Machine Learning Research*, 2023. URL https://openreview.net/forum?id=j3oQF9coJd.

Shayan Shekarforoush, David B. Lindell, David J. Fleet, and Marcus A. Brubaker. Residual Multiplicative Filter Networks for Multiscale Reconstruction. In *Advances in Neural Information Processing Systems*, volume 35, pp. 8550–8563, 2022. doi: 10.52202/068431-0622.

Pengqing Shi, Jie Yin, Stephen Tierney, and Junbin Gao. SirenFNO: Efficient and Full Frequency Learning of Fourier Neural Operators. In *Proceedings of the Thirty-Fifth International Joint Conference on Artificial Intelligence*, pp. 4804–4812, 2026. doi: 10.24963/ijcai.2026/535.

Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit Neural Representations with Periodic Activation Functions. In *Advances in Neural Information Processing Systems*, volume 33, pp. 7462–7473, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/53c04118df112c13a8c34b38343b9c10-Abstract.html.

Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Dan MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. PDEBench Datasets. DaRUS, 2022. URL https://doi.org/10.18419/DARUS-2986. Dataset.

Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Dan MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. PDEBench: An Extensive Benchmark for Scientific Machine Learning. *arXiv preprint arXiv:2210.07182v7*, 2024. URL https://arxiv.org/abs/2210.07182v7.

Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. In *Advances in Neural Information Processing Systems*, volume 33, pp. 7537–7547, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/55053683268957697aa39fba6f231c68-Abstract.html.

Alasdair Tran, Alexander Mathews, Lexing Xie, and Cheng Soon Ong. Factorized Fourier Neural Operators. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=tmIiMPl4IPa.

Gege Wen, Zongyi Li, Kamyar Azizzadenesheli, Anima Anandkumar, and Sally M. Benson. U-FNO—An enhanced Fourier neural operator-based deep-learning model for multiphase flow. *Advances in Water Resources*, 163:104180, 2022. doi: 10.1016/j.advwatres.2022.104180.

Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. Transolver: A Fast Transformer Solver for PDEs on General Geometries. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 53681–53705, 2024. URL https://proceedings.mlr.press/v235/wu24r.html.

Zipeng Xiao, Siqi Kou, Zhongkai Hao, Bokai Lin, and Zhijie Deng. Amortized Fourier Neural Operators. In *Advances in Neural Information Processing Systems*, volume 37, pp. 115001–115020, 2024. doi: 10.52202/079017-3651.

Zhi-Qin John Xu, Yaoyu Zhang, Tao Luo, Yanyang Xiao, and Zheng Ma. Frequency Principle: Fourier Analysis Sheds Light on Deep Neural Networks. *arXiv preprint arXiv:1901.06523v7*, 2024. URL https://arxiv.org/abs/1901.06523v7.

Gizem Yüce, Guillermo Ortiz-Jiménez, Beril Besbinar, and Pascal Frossard. A Structured Dictionary Perspective on Implicit Neural Representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19228–19238, 2022. URL https://openaccess.thecvf.com/content/CVPR2022/html/Yuce_A_Structured_Dictionary_Perspective_on_Implicit_Neural_Representations_CVPR_2022_paper.html.

A Data Preprocessing, Training, and Evaluation

A.1 Data Splits and Preprocessing

The data sources, splits, and input–output definitions follow Table 1 in the main text. For Darcy, only the outputs are standardized using training-set statistics. For Navier–Stokes, the inputs and outputs are standardized separately using their respective training-set statistics. In both cases, the output normalization is inverted for evaluation. For Burgers, the global mean and population standard deviation are computed over all temporal and spatial values of 200 training trajectories selected separately for each seed. These statistics are used to normalize both inputs and outputs, and errors are computed in the normalized space.

Navier–Stokes. For Navier–Stokes, we use the input–output pairs in `nsforcing_train_128.pt` and `nsforcing_test_128.pt` provided by NeuralOperator. Each input and output is a single-channel vorticity field at a different time, and the corresponding output vorticity field is predicted directly from the input vorticity field. We do not use autoregressive rollout with multiple input snapshots in this experiment.

Airfoil. For Airfoil, the X, Y coordinates of a 221×51 structured mesh are used as inputs to predict the Mach number at each grid point. The prediction target is the fifth component of the original flow-field array. No additional value normalization is applied to the input coordinates or output values.

Burgers and ReacDiff. For both benchmarks, we use the first 20 snapshots of each trajectory without temporal or spatial subsampling. Denoting the stored time points in the original data by t_j, the 10 snapshots at $t_0, \ldots, t_9$ are used as inputs, and the 10 snapshots at $t_{10}, \ldots, t_{19}$ are predicted autoregressively. Thus, the input interval is $[t_0, t_9]$ and the prediction interval is $[t_{10}, t_{19}]$, with the time step determined by the storage interval of the original data.

A.2 Training and Evaluation Settings

Table 4: Training and evaluation settings.

Dataset	Batch size	Prediction protocol	Evaluation space
Darcy	32	Direct prediction of the provided input–output field pairs	After output denormalization
Navier–Stokes	32	Direct prediction of the provided single input–output field pairs	After output denormalization
Burgers	32	Autoregressive prediction of 10 future snapshots from 10 input snapshots	Normalized space
Airfoil	8	Direct prediction of the target field from coordinates	Original value space
ReacDiff	32	Autoregressive prediction of 10 future snapshots from 10 input snapshots	Original value space

Training loss. For Darcy, Navier–Stokes, and Airfoil, training uses sample-wise relative L_2 errors on the directly predicted target fields. For Burgers and ReacDiff, the training loss is the sum of the relative L_2 errors over the 10 prediction steps. At each step, the per-sample relative errors are summed over the batch for Burgers and averaged over the batch for ReacDiff. For both benchmarks, previous predictions are incorporated into the input to the next step, and the computational graph is retained over the entire rollout for backpropagation during training. The Burgers loss is computed in the normalized space, whereas the ReacDiff loss is computed without value normalization.

B Frequency Coordinate Normalization

We use the Fourier mode k and normalized spectral coordinate $\boldsymbol{\xi}_k$ defined in the main text. Let L_a denote the length of the ath FFT axis after padding. The one-dimensional rFFT in the dense model stores nonnegative Fourier modes, with normalized coordinates given by

$$\boldsymbol{\xi}_k = \frac{2k}{L_1}, \qquad k = 0, \ldots, \left\lfloor \frac{L_1}{2} \right\rfloor. \tag{23}$$

Let $p = 0, \ldots, L_1 - 1$ and $q = 0, \ldots, \lfloor L_2/2 \rfloor$ denote the integer storage indices of a two-dimensional rFFT. The components of the Fourier mode $k = (k_1, k_2)$ corresponding to storage position (p, q) are defined as

$$k_1 = \begin{cases} p, & 0 \le p \le \lfloor (L_1 - 1)/2 \rfloor, \\ p - L_1, & \lfloor (L_1 - 1)/2 \rfloor < p < L_1, \end{cases} \qquad k_2 = q. \tag{24}$$

The first axis is therefore converted to signed frequency indices, while the last axis uses nonnegative frequency indices. The normalized coordinates are then given by

$$\boldsymbol{\xi}_k = \left(\frac{2k_1}{L_1}, \frac{2k_2}{L_2} \right)^{\top}. \tag{25}$$

For the CP, TT, and Tucker models, midpoint coordinates are used for each input-component, output-component, and frequency-storage axis. For a factor axis of length I with storage index ν, the input coordinate to its factor generator is

$$s_I(\nu) = \frac{2(\nu + 1/2)}{I} - 1, \qquad \nu = 0, \ldots, I - 1. \tag{26}$$

Even on a frequency axis, ν follows the storage order and is therefore distinct from the signed frequency index used by the dense model. Hermitian projection is applied according to the rFFT storage convention. In one dimension, the kernels at the DC component and, for an even axis length, the Nyquist component are made real. In two dimensions, for the DC column of the last frequency axis and its Nyquist column when that axis has even length, each kernel along the first axis is averaged with the complex conjugate of the kernel at the opposite frequency. The remaining stored modes are unchanged, and the unstored modes are determined by conjugate symmetry. This operation is applied elementwise to each kernel matrix, without transposing the input- and output-component axes.

C CP, TT, and Tucker Tensor Decompositions

Following the decomposition structures in Eqs. (9)–(11) of Section 4.2 of SirenFNO (Shi et al., 2026), we apply CAFE+ encoding to the factor generator for each axis. Let d_s denote the spatial dimension, and arrange the $n_{\mathrm{ax}} = d_s + 2$ axes in the order of the input components, the d_s frequency axes, and the output components. The real and imaginary parts, indexed by $c \in \{\mathrm{R}, \mathrm{I}\}$, are generated independently and combined as $U = U^{(\mathrm{R})} + iU^{(\mathrm{I})}$:

$$\text{CP:} \quad U^{(c)} = \sum_{\alpha=1}^{r} \mathbf{f}_{1,\alpha}^{(c)} \otimes \cdots \otimes \mathbf{f}_{n_{\mathrm{ax}},\alpha}^{(c)}, \tag{27}$$

$$\text{TT:} \quad U^{(c)}_{\nu_1,\ldots,\nu_{n_{\mathrm{ax}}}} = A_1^{(c)}(\nu_1) \cdots A_{n_{\mathrm{ax}}}^{(c)}(\nu_{n_{\mathrm{ax}}}), \tag{28}$$

$$\text{Tucker:} \quad U^{(c)} = \mathcal{C}^{(c)} \times_1 F_1^{(c)} \cdots \times_{n_{\mathrm{ax}}} F_{n_{\mathrm{ax}}}^{(c)}. \tag{29}$$

For tensor-axis lengths I_a, we have $U^{(c)} \in \mathbb{R}^{I_1 \times \cdots \times I_{n_{\mathrm{ax}}}}$. The CP factors satisfy $\mathbf{f}_{a,\alpha}^{(c)} \in \mathbb{R}^{I_a}$, the Tucker factor matrices satisfy $F_a^{(c)} \in \mathbb{R}^{I_a \times r}$, and the trainable Tucker core satisfies $\mathcal{C}^{(c)} \in \mathbb{R}^{r \times \cdots \times r}$. Each TT core slice has dimensions

$$A_a^{(c)}(\nu_a) \in \mathbb{R}^{r_{a-1} \times r_a}, \qquad r_0 = r_{n_{\mathrm{ax}}} = 1, \qquad r_a = r \quad (1 \le a < n_{\mathrm{ax}}). \tag{30}$$

Here, $\otimes$ and $\times_a$ denote the outer product and the mode-a tensor–matrix product, respectively. CP represents a tensor as a sum of rank-one tensors, whereas Tucker combines a core tensor with axis-specific factor matrices (Kolda & Bader, 2009). In this work, CP uses r components, and Tucker uses rank r along each axis with a core containing $r^{n_{\mathrm{ax}}}$ entries. TT represents a tensor as a sequential product of cores (Oseledets, 2011); we set both boundary ranks to 1 and all internal ranks to r. The CP and Tucker generators output r values per coordinate, while TT outputs r values at the two boundary axes and r^2 values at internal axes. These ranks specify the decompositions of the individual real-valued tensors and do not denote the exact rank of the final complex-valued kernel.

An independent CAFE+ factor generator is used for each axis and each real or imaginary component. For CP and Tucker, the midpoint coordinate $s_{I_a}(\nu_a)$ from Appendix B is supplied as input, and the generator outputs r values corresponding to one row of the factor matrix. For TT, the generator similarly outputs $r_{a-1}r_a$ values, which are reshaped into a core slice. The reconstructed tensor, whose axes are ordered as input components, frequency-storage axes, and output components, is rearranged into an output–input matrix for each mode and used for spectral mixing.

D Model Settings

CAFE+FNO uses a hidden channel width of 32, four operator layers, and two linear branches. The dense model uses $m = 32$ Fourier frequency vectors in $\mathbb{R}^{d_s}$ and $n = 32$ Chebyshev polynomials along each coordinate axis, with a kernel MLP containing one hidden layer. The dataset-specific branch output dimensions, kernel MLP hidden widths, and decomposition ranks are listed in Table 5.

Each factor generator in the decomposed models uses 16 Fourier frequency vectors and eight Chebyshev polynomials, with a branch output dimension of 12. The factor MLP hidden widths are 16, 24, and 16 for CP, TT, and Tucker, respectively. For the factorized SirenFNO baselines, the backbone input dimension is 16 for all ReacDiff variants and for Tucker on NS, and 32 otherwise.

Table 5: Dataset-specific architectural settings for CAFE+FNO. The branch output dimension and MLP hidden width refer to the dense model; the common settings for the decomposed models are described in the text.

Dataset	**Dense branch output dimension**	**Dense MLP hidden width**	**CP rank**	**TT rank**	**Tucker per-axis rank**
Darcy	32	32	8	8	10
Navier–Stokes	32	64	16	16	16
Burgers	32	32	8	8	8
Airfoil	32	32	8	8	8
ReacDiff	16	32	8	8	8

E Per-Seed Test Errors

Tables 7–11 provide the individual-seed test relative L_2 errors for the models and datasets compared in Table 2, using random seeds $\{0, 42, 73, 108, 202\}$. Each result is obtained from the final-epoch model after 500 training epochs, following the same evaluation protocols as the main comparison. Errors are reported in scientific notation with three decimal places in the mantissa.

Table 6: Trainable parameter counts in real scalars, including lifting, operator layers, and projection. Each complex parameter counts as two real scalars; shared parameters are counted once. Fixed buffers are excluded, and kernel-generator outputs are not counted as additional parameters. The dash denotes an omitted model–dataset combination.

Model	**Darcy**	**NS**	**Burgers**	**Airfoil**	**ReacDiff**
Resolution	128×128	128×128	1024	221×51	1024
FNO	1,192,801	4,469,601	4,216,161	2,372,513	4,216,161
U-FNO	5,931,201	990,753	1,883,073	—	1,892,161
TFNO-CP	72,913	237,505	226,369	131,993	226,369
AM-FNO (MLP)	385,473	4,443,137	823,073	1,136,673	823,073
SirenFNO	308,865	579,457	308,993	308,897	304,833
CAFE+FNO	345,601	611,969	337,665	345,633	323,201
CP-SirenFNO	80,513	138,881	70,145	80,545	57,665
TT-SirenFNO	109,185	211,585	84,481	109,217	72,001
Tucker-SirenFNO	162,561	596,353	74,241	113,313	61,761
CAFE+FNO (CP)	80,517	84,869	70,149	80,549	70,149
CAFE+FNO (TT)	108,293	188,293	85,381	108,325	85,381
CAFE+FNO (Tucker)	161,605	609,157	74,245	113,317	74,245

Table 7: Test relative L_2 errors for seed 0 after 500 training epochs. The dash denotes an omitted model–dataset combination.

Model	Darcy	NS	Burgers	Airfoil	ReacDiff
Resolution	128×128	128×128	1024	221×51	1024
FNO	$6.200\text{e}{-2}$	$5.610\text{e}{-2}$	$1.135\text{e}{-2}$	$6.047\text{e}{-3}$	$4.252\text{e}{-3}$
U-FNO	$4.790\text{e}{-2}$	$5.824\text{e}{-2}$	$1.005\text{e}{-2}$	—	$7.578\text{e}{-3}$
TFNO-CP	$4.928\text{e}{-2}$	$3.394\text{e}{-2}$	$9.423\text{e}{-3}$	$7.026\text{e}{-3}$	$4.185\text{e}{-3}$
AM-FNO (MLP)	$7.497\text{e}{-2}$	$3.113\text{e}{-2}$	$2.338\text{e}{-2}$	$6.519\text{e}{-3}$	$1.001\text{e}{-2}$
SirenFNO	$2.587\text{e}{-2}$	$4.460\text{e}{-2}$	$5.392\text{e}{-3}$	$5.691\text{e}{-3}$	$2.148\text{e}{-3}$
CAFE+FNO	$1.384\text{e}{-2}$	$1.756\text{e}{-2}$	$4.205\text{e}{-3}$	$4.707\text{e}{-3}$	$3.843\text{e}{-3}$
CP-SirenFNO	$3.091\text{e}{-2}$	$4.714\text{e}{-2}$	$8.479\text{e}{-3}$	$6.910\text{e}{-3}$	$2.880\text{e}{-3}$
TT-SirenFNO	$1.428\text{e}{-1}$	$3.924\text{e}{-2}$	$7.403\text{e}{-3}$	$5.902\text{e}{-3}$	$2.531\text{e}{-3}$
Tucker-SirenFNO	$3.744\text{e}{-2}$	$4.890\text{e}{-2}$	$8.293\text{e}{-3}$	$7.876\text{e}{-3}$	$2.992\text{e}{-3}$
CAFE+FNO (CP)	$3.025\text{e}{-2}$	$9.429\text{e}{-2}$	$8.656\text{e}{-3}$	$8.307\text{e}{-3}$	$4.243\text{e}{-3}$
CAFE+FNO (TT)	$2.652\text{e}{-2}$	$6.126\text{e}{-2}$	$8.234\text{e}{-3}$	$7.194\text{e}{-3}$	$3.988\text{e}{-3}$
CAFE+FNO (Tucker)	$2.781\text{e}{-2}$	$4.893\text{e}{-2}$	$8.911\text{e}{-3}$	$6.724\text{e}{-3}$	$4.077\text{e}{-3}$

Table 8: Test relative L_2 errors for seed 42 after 500 training epochs. The dash denotes an omitted model–dataset combination.

Model	Darcy	NS	Burgers	Airfoil	ReacDiff
Resolution	128×128	128×128	1024	221×51	1024
FNO	7.201e−2	5.585e−2	1.121e−2	6.130e−3	4.125e−3
U-FNO	4.904e−2	5.822e−2	9.668e−3	—	4.914e−3
TFNO-CP	4.734e−2	3.525e−2	1.054e−2	6.670e−3	3.930e−3
AM-FNO (MLP)	4.344e−2	2.858e−2	2.010e−2	5.792e−3	1.052e−1
SirenFNO	2.285e−2	4.578e−2	5.617e−3	6.068e−3	2.384e−3
CAFE+FNO	1.382e−2	1.695e−2	3.718e−3	4.582e−3	3.763e−3
CP-SirenFNO	3.122e−2	5.328e−2	8.780e−3	9.428e−3	7.126e−3
TT-SirenFNO	2.904e−2	4.263e−2	7.820e−3	6.409e−3	2.697e−3
Tucker-SirenFNO	5.288e−2	3.904e−2	8.214e−3	6.280e−3	2.784e−3
CAFE+FNO (CP)	3.373e−2	4.288e−2	8.076e−3	7.653e−3	4.391e−3
CAFE+FNO (TT)	2.777e−2	3.364e−2	8.554e−3	6.576e−3	4.449e−3
CAFE+FNO (Tucker)	3.034e−2	3.540e−2	8.844e−3	7.880e−3	4.367e−3

Table 9: Test relative L_2 errors for seed 73 after 500 training epochs. The dash denotes an omitted model–dataset combination.

Model	Darcy	NS	Burgers	Airfoil	ReacDiff
Resolution	128×128	128×128	1024	221×51	1024
FNO	6.389e−2	5.573e−2	1.134e−2	6.310e−3	3.944e−3
U-FNO	4.521e−2	5.775e−2	1.011e−2	—	1.927e−2
TFNO-CP	4.770e−2	3.488e−2	1.030e−2	6.828e−3	3.715e−3
AM-FNO (MLP)	6.956e−2	2.806e−2	1.490e−2	7.247e−3	8.633e0
SirenFNO	2.517e−2	4.055e−2	6.347e−3	5.532e−3	3.682e−3
CAFE+FNO	1.362e−2	1.694e−2	4.514e−3	4.785e−3	4.203e−3
CP-SirenFNO	4.091e−2	5.295e−2	9.258e−3	7.934e−3	3.116e−3
TT-SirenFNO	4.308e−2	3.731e−2	7.753e−3	5.915e−3	3.006e−3
Tucker-SirenFNO	7.154e−2	4.141e−2	8.643e−3	1.482e−2	3.007e−3
CAFE+FNO (CP)	2.715e−2	5.776e−2	8.819e−3	7.006e−3	3.579e−3
CAFE+FNO (TT)	2.944e−2	5.075e−2	8.419e−3	8.021e−3	3.453e−3
CAFE+FNO (Tucker)	2.702e−2	3.660e−2	9.431e−3	9.674e−3	3.735e−3

Table 10: Test relative L_2 errors for seed 108 after 500 training epochs. The dash denotes an omitted model–dataset combination.

Model	Darcy	NS	Burgers	Airfoil	ReacDiff
Resolution	128×128	128×128	1024	221×51	1024
FNO	6.690e−2	5.584e−2	1.122e−2	6.000e−3	4.239e−3
U-FNO	4.812e−2	5.636e−2	1.008e−2	—	7.520e−3
TFNO-CP	4.607e−2	3.504e−2	1.046e−2	7.891e−3	4.407e−3
AM-FNO (MLP)	5.421e−2	3.245e−2	1.719e−2	5.511e−3	1.133e−2
SirenFNO	2.442e−2	4.266e−2	5.710e−3	5.936e−3	1.789e−3
CAFE+FNO	1.353e−2	1.753e−2	3.760e−3	4.588e−3	3.452e−3
CP-SirenFNO	4.386e−2	4.228e−2	9.003e−3	6.144e−3	4.083e−3
TT-SirenFNO	6.327e−1	3.225e−2	7.656e−3	5.744e−3	2.533e−3
Tucker-SirenFNO	1.073e−1	6.707e−2	7.504e−3	5.547e−3	2.928e−3
CAFE+FNO (CP)	3.098e−2	1.011e−1	8.024e−3	1.312e−2	4.767e−3
CAFE+FNO (TT)	2.546e−2	3.159e−2	7.997e−3	7.036e−3	4.481e−3
CAFE+FNO (Tucker)	2.979e−2	3.501e−2	9.368e−3	7.324e−3	3.454e−3

Table 11: Test relative L_2 errors for seed 202 after 500 training epochs. The dash denotes an omitted model–dataset combination.

Model	Darcy	NS	Burgers	Airfoil	ReacDiff
Resolution	128×128	128×128	1024	221×51	1024
FNO	6.220e−2	5.794e−2	1.138e−2	6.387e−3	4.347e−3
U-FNO	4.499e−2	5.789e−2	1.020e−2	—	6.383e−3
TFNO-CP	4.631e−2	3.660e−2	1.064e−2	6.896e−3	4.123e−3
AM-FNO (MLP)	6.792e−2	2.946e−2	1.809e−2	6.375e−3	3.048e−2
SirenFNO	2.341e−2	4.262e−2	6.246e−3	5.552e−3	1.774e−3
CAFE+FNO	1.370e−2	1.701e−2	4.440e−3	4.856e−3	3.731e−3
CP-SirenFNO	8.353e−2	6.165e−2	9.014e−3	1.018e−2	4.658e−3
TT-SirenFNO	4.625e−2	2.878e−2	7.972e−3	6.147e−3	2.401e−3
Tucker-SirenFNO	5.115e−2	3.709e−2	7.878e−3	6.363e−3	2.904e−3
CAFE+FNO (CP)	2.814e−2	4.846e−2	1.010e−2	7.332e−3	3.755e−3
CAFE+FNO (TT)	2.965e−2	3.525e−2	9.191e−3	6.499e−3	4.065e−3
CAFE+FNO (Tucker)	2.481e−2	4.197e−2	8.275e−3	9.050e−3	2.981e−3

F Qualitative Comparisons on Random Test Samples

We qualitatively compare FNO, AM-FNO, SirenFNO, and CAFE+FNO on Darcy flow, Navier–Stokes, and Airfoil. For each dataset, three test samples were selected at random, and all four models were evaluated on the same selected samples using checkpoints trained with seed 0. The results are organized into three sample sets, each containing one test sample from each dataset.

In all figures, rows correspond to Darcy flow, Navier–Stokes, and Airfoil, from top to bottom. Prediction figures show the ground truth followed by predictions from FNO, AM-FNO, SirenFNO, and CAFE+FNO, from left to right. The corresponding error maps show the pointwise absolute error $|\hat{u} - u|$ for the four models in the same order, where $\hat{u}$ and u denote the prediction and ground truth. For each figure, all panels within a dataset row share the same color scale. Values below the model panels report sample-wise relative L_2 errors. Airfoil panels show a zoomed view around the airfoil.

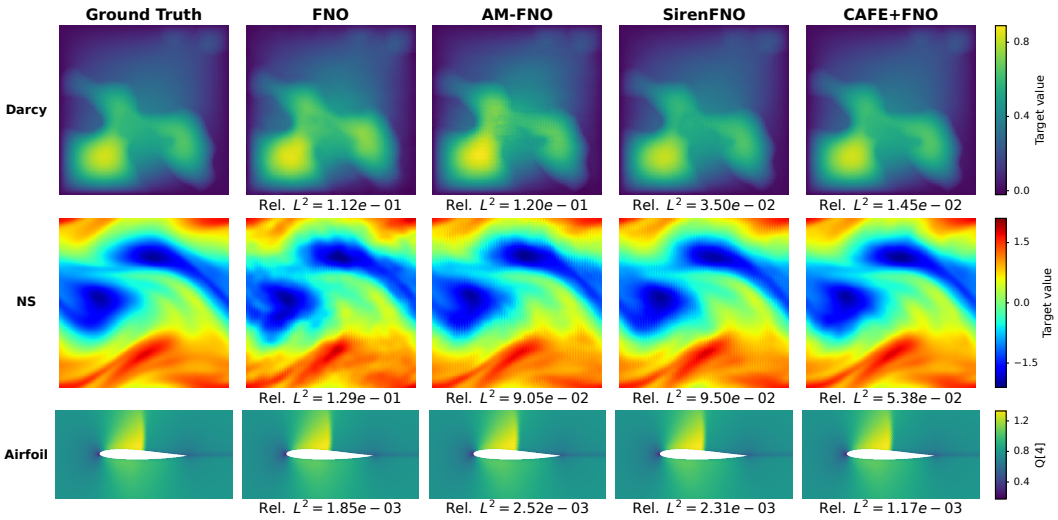

Figure 3: Ground truth and model predictions for random test sample set 1.

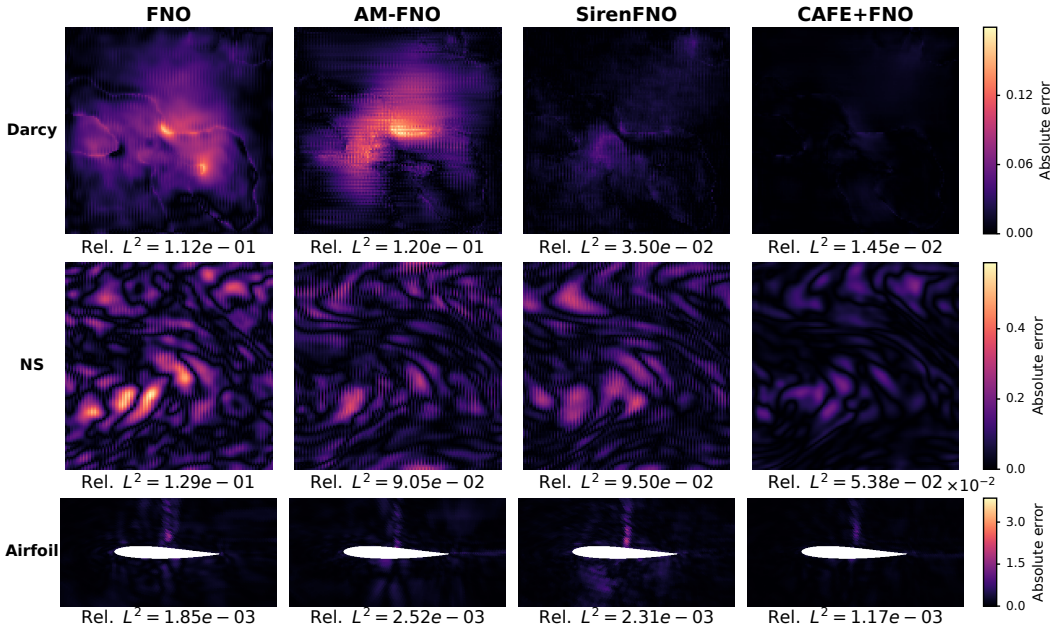

Figure 4: Absolute error maps for random test sample set 1.

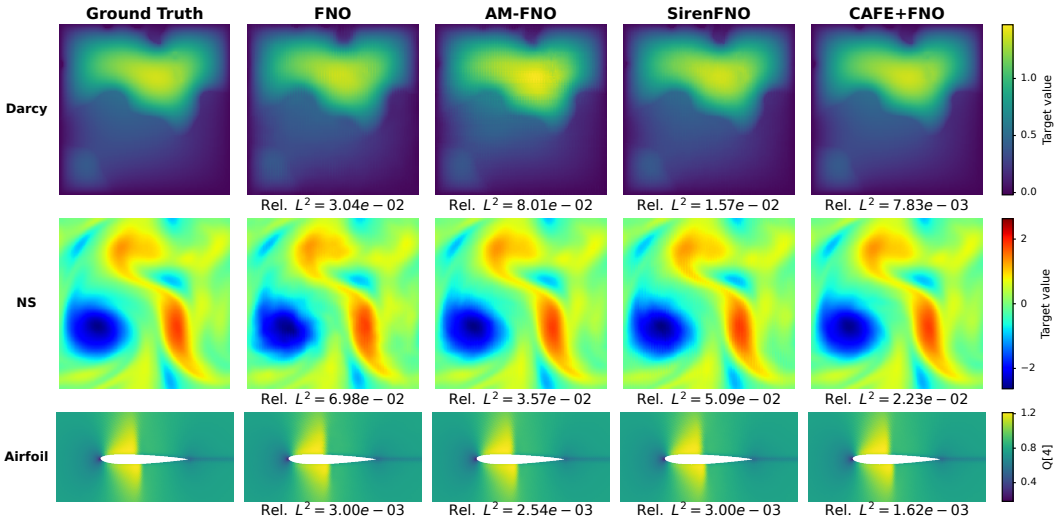

Figure 5: Ground truth and model predictions for random test sample set 2.

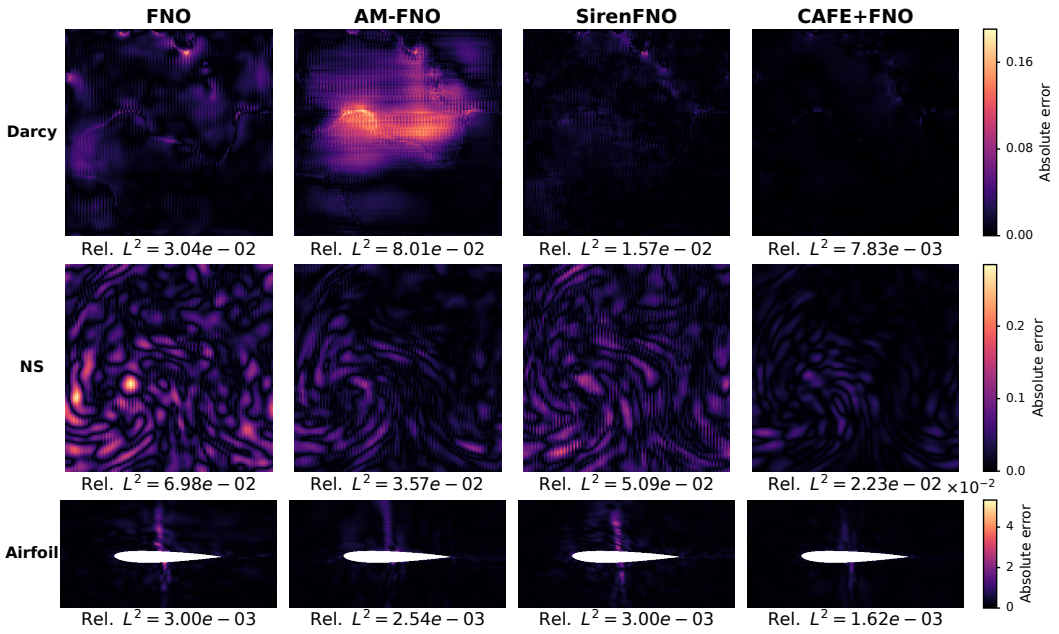

Figure 6: Absolute error maps for random test sample set 2.

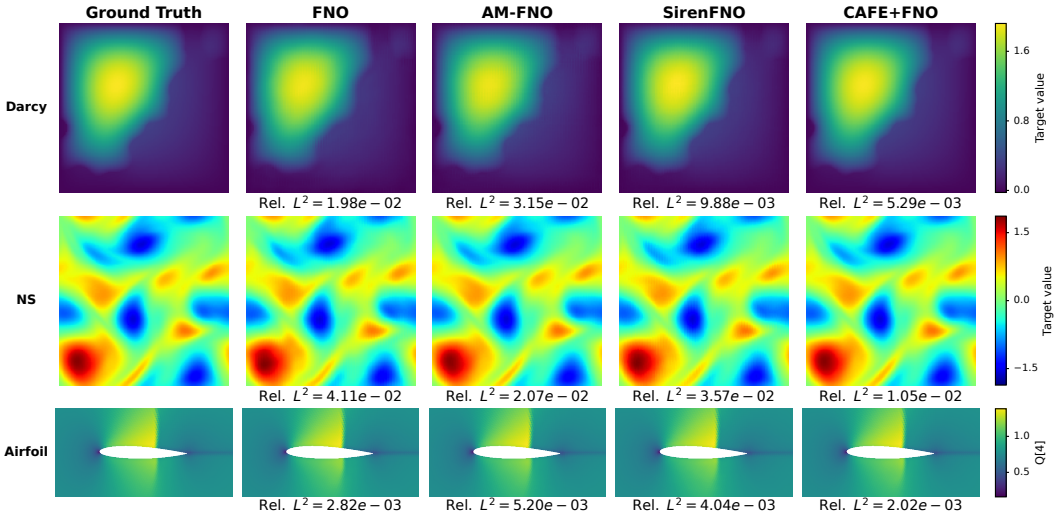

Figure 7: Ground truth and model predictions for random test sample set 3.

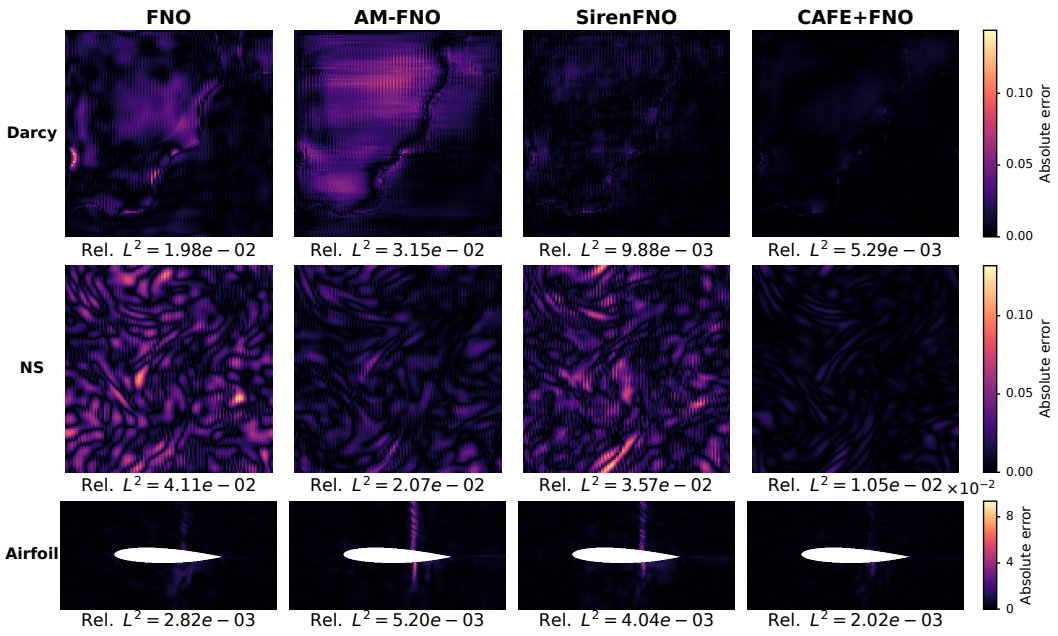

Figure 8: Absolute error maps for random test sample set 3.