FROM COARSE TRAINING TO FINE INFERENCE IN NEURAL OPERATORS VIA RESOLUTION ADAPTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

We address this cross-resolution generalization problem using Polyphase Decomposition. Rather than adapting a pretrained neural operator to a finer discretization, we decompose the high-resolution problem into subproblems at the resolution seen during pretraining, apply the pretrained operator to each subproblem, and recombine their predictions at the target resolution. This yields a zero-shot method, Polyphase Inference, that requires no high-resolution training data. When high-resolution examples are available, the decomposition strategy can be learned and complemented with fine-tuning modules, giving Polyphase Fine-Tuning. Across several PDE benchmarks and neural architectures, Polyphase Inference consistently improves cross-resolution performance over direct evaluation on the target resolution. In the low-data setting, Polyphase Fine-Tuning substantially outperforms standard parameter-efficient adaptation methods while using the same or fewer additional parameters. These results suggest that matching the pretrained input resolution can be an effective component of cross-resolution transfer.

1 INTRODUCTION

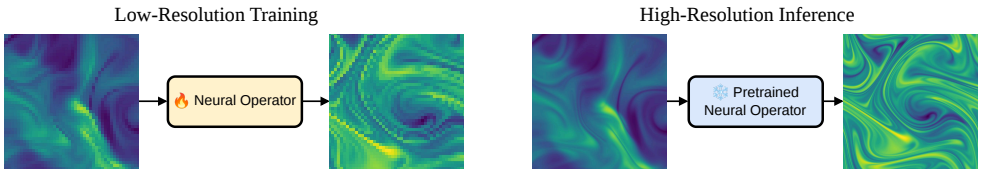

Figure 1: **Left:** Training at low resolution. **Right:** Inference at high resolution. Images represent density fields of a compressible Navier-Stokes dataset at resolutions 128^2 (left) and 1024^2 (right).

Many physical systems exhibit critical fine-scale structures that can only be captured at sufficiently high resolution (Moin & Mahesh, 1998). However, collecting or simulating high-resolution data comes at a prohibitive computational and storage cost. As a consequence, scientific machine learning operates under a severe asymmetric data regime: low-resolution trajectories are abundant, while high-resolution examples remain scarce and costly. This is precisely the setting, illustrated in Figure 1, in which one needs to pretrain a model cheaply at low resolution and then deploy it to the high-resolution task, either in a zero-shot manner (i.e., without any parameter updates), or through parameter-efficient fine-tuning (PEFT, a lightweight update of only a few parameters) Prottasha et al. (2025).

An active line of work to tackle this problem is to design models that can generalize across discretizations by construction. Within this framework, neural operators (Kovachki et al., 2023) learn mappings between functions rather than fixed-size arrays. This allows them, in principle, to be evaluated at resolutions unseen during training. Among them, Fourier Neural Operators (FNOs) (Li et al., 2021) have shown particularly strong robustness across resolution changes, making them a natural reference point for studying super-resolution transfer.

However, recent evidence shows that this zero-shot transfer of Neural Operators across resolutions can fail substantially (Gao et al., 2025; Sakarvadia et al., 2026; Colagrande et al., 2026a). Changing

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.

resolution alters the information content and statistics of the discretized fields through aliasing and the loss or emergence of spatial degrees of freedom, making it challenging for a model trained at a given resolution r to generalize to higher resolutions $R > r$. In this work, we argue that transferring a neural PDE solver from resolution r to $R > r$ should be viewed as a *structured distribution shift induced by discretization*, rather than simply evaluating the same model on a denser mesh.

In the general case, it is unrealistic to assume that a model trained at low resolution could learn the high-resolution information unseen during pretraining. Instead, a more realistic approach should assume that a pretrained model can, at most, maintain the low-resolution performance and learn the additional fine-scale information through a fine-tuning procedure.

To preserve the prediction quality learned at low resolution, a solution is to downsample the data to the pretraining resolution, perform inference with the pretrained model, and upsample to obtain a high-resolution field, as done in (Colagrande et al., 2026a). However, downsampling often implies a significant loss in information. Therefore, downsampling cannot, by design, exploit the additional information obtained from high-resolution data and it is not particularly suitable to fine-tuning either. We will refer to these methods as *compressed inference*.

In this work, we propose *decomposed inference*. The main idea is to *decompose high-resolution data into multiple fields at the pretraining resolution, apply the pretrained model to each field, and then recombine the resulting low-resolution predictions into a high-resolution prediction*. As illustrated in Figure 2, decomposed inference, similarly to compressed inference, applies the model only at the resolution used for pretraining, but, unlike compressed inference, it does not lose information by compression. This paper makes the following principal contributions:

1. **A general framework for cross-resolution adaptation via decomposition.** We introduce *Decomposed Inference*, which transfers models pretrained at low resolution to finer discretizations by decomposing high-resolution inputs into components at the pretraining resolution, applying the pretrained model independently, and recomposing the predictions. Polyphase Inference provides a parameter-free method that is naturally extendable to fine-tuning.
2. **A theoretical characterization of cross-resolution transfer.** We refine bounds on FNO discretization mismatch, characterize the irreducible error of compressed inference, and identify marginal invariance as the key condition for risk preservation under decomposition. We further decompose the remaining excess risk into *in-component* and *cross-component* terms, directly motivating separate model and representation adaptation.
3. **Broad empirical validation.** We evaluate the method across multiple PDEs, architectures, and large pretrained physics models. Polyphase improves zero-shot transfer, supports parameter-efficient finetuning, and offers favourable time–memory trade-offs at high resolution.

2 Related work

We discuss the works most closely related to cross-resolution generalization and defer a broader discussion to Appendix B. Neural operators (Kovachki et al., 2023), and FNOs in particular (Li et al., 2021), are a class of models designed to be robust across discretizations, whereas we do not rely on the pretrained model being resolution-agnostic and instead perform inference entirely at its training resolution. ReNO (Bartolucci et al., 2023) and CNO (Raonic et al., 2023) address discretisation effects through representation consistency; in contrast, our method is model-agnostic and leaves the pretrained architecture unchanged.

Gao et al. (2025) characterises discretisation mismatch and proposes CROP, which controls it through band-limited representations, while we avoid cross-resolution evaluation altogether by decomposing the fine-resolution problem into coarse-resolution components. We do not include CROP as a baseline since we found the same negative results reported in Sakarvadia et al. (2026). Sakarvadia et al. (2026) tackle unreliable zero-shot super-resolution through multi-resolution training, our setting instead assumes a model pretrained only at low resolution and requires no high-resolution data in the zero-shot case. Finally, Colagrande et al. (2026a) mitigate resolution shift through Low+Pad, an instance of compressed inference, which returns the input to the pretraining resolution at the cost of discarding information, our decomposition likewise keeps inference at the pretraining resolution but preserves the full high-resolution representation.

3 Problem Setting and Limitations of Previous Methods

3.1 Super-Resolution transfer learning in Physics

Let $\mathcal{G}_r : \mathcal{X}_r \to \mathcal{Y}_r$ be a model pretrained at spatial resolution r to approximate the discretized solution operator $\mathcal{G}_r^\dagger : \mathcal{X}_r \to \mathcal{Y}_r$ of a physical system, where $\mathcal{X}_r$ and $\mathcal{Y}_r$ are the input and output spaces induced by discretizing the continuous solution operator $\mathcal{G}^\dagger : \mathcal{X} \to \mathcal{Y}$ at resolution r.

We are interested in transferring this pretrained model to a finer discretization of the same physical system. Let $R > r$ denote the target resolution. Given an input $u_R \in \mathcal{X}_R$, the target task is therefore to approximate $v_R = \mathcal{G}_R^\dagger(u_R) \in \mathcal{Y}_R$ while reusing the knowledge already encoded in $\mathcal{G}_r$.

We refer to this problem as *super-resolution transfer learning*. Given a model $\mathcal{G}_r$ pretrained at resolution r, our goal is to construct a predictor for resolution R while requiring little or no additional high-resolution data and tuning only a small number of parameters.

We consider two problems. In the *zero-shot* problem, no high-resolution training data are available, and the pretrained model must be used without updating its parameters. In the *few-shot* problem, a small number of high-resolution examples are available and can be used to adapt the predictor.

3.2 The Limits of Direct inference of Fourier Neural Operators

A natural approach to address the asymmetry between abundant low-resolution data and scarce high-resolution data is to develop architectures that are robust to changes in resolution by construction. Neural Operators (Kovachki et al., 2023) pursue this objective, and among existing neural operator architectures, FNO (Li et al., 2021) has shown particularly strong capabilities for evaluation at resolutions beyond those used during training. Thus, FNO is the starting point of our work.

However, the super-resolution performance of FNOs generally does not match the performance obtained at the pretraining resolution (Sakarvadia et al., 2026; Gao et al., 2025; Colagrande et al., 2026a). Tables in Appendix H show the degradation quantitatively. In Proposition 1, we characterise the cross-resolution discretisation mismatch of a fixed FNO by refining Proposition 4.4 of Gao et al. (2025), using proof techniques developed in Lanthaler et al. (2024).

Proposition 1 (Cross-resolution discretization mismatch of an FNO)**.** *Let $\mathcal{G}_r$ denote an L-layer FNO pretrained at resolution r and evaluated at two resolutions $r < R = mr$. Define its aliasing discrepancy by $\Delta_{r,R}(f) := \|(I_R - I_r)f\|_{L^2(\mathbb{T}^d)}$ where I_r and I_R are trigonometric interpolation operators. We identify each discrete FNO output with its trigonometric interpolant when taking the L^2 norm. Then*

$$\|\mathcal{G}_r(u_R) - \mathcal{G}_r(u_r)\|_{L^2} \le C_{\text{in}}\, \Delta_{r,R}(\mathcal{P}(u)) + \sum_{\ell=0}^{L-1} C_\ell\, \Delta_{r,R}(\sigma_\ell(z_\ell^r)) + \Delta_{r,R}(\mathcal{Q}(v_L^r))\,, \qquad (1)$$

where $\mathcal{P}$ and $\mathcal{Q}$ denote the FNO lifting and projection maps, $u \in H^s(\mathbb{T}^d)$ the continuous function in the Sobolev space $H^s(\mathbb{T}^d)$ with regularity $s > d/2$ on the Torus $\mathbb{T}^d$ from which u_r and u_R are sampled from. z_ℓ^r and v_L^r are FNO's intermediate and final features at resolution r, and σ_ℓ the layerwise nonlinearity at layer ℓ that we assume globally lipschitz. The constants C_{in} and C_ℓ depend only on the Lipschitz constants and operator norms of the subsequent FNO layers.

Additionally, indicating with Λ_r and Λ_R the sets of Fourier indices, the aliasing discrepancy has the explicit form

$$\Delta_{r,R}(f)^2 = \sum_{k \in \Lambda_r} \|\mathcal{F}_R(f)(k) - \mathcal{F}_r(f)(k)\|^2 + \sum_{k \in \Lambda_R \setminus \Lambda_r} \|\mathcal{F}_R(f)(k)\|^2\,, \qquad (2)$$

where the first term quantifies the discrepancy due to high frequencies aliased into low-frequency modes, while the second captures frequencies represented on the finer grid but not on the coarser grid. The proof is provided in Appendix D.

While Gao et al. (2025) obtain zero discrepancy under bandlimiting assumptions at the input, our analysis shows that aliasing can arise again after nonlinear layers. A sufficient condition for the right-hand side of equation 1 to vanish is that every intermediate representation has zero aliasing discrepancy between resolutions r and R. Bandlimiting all of these quantities to modes resolved at resolution r is a sufficient condition to ensure this.

3.3 Compression as a solution to aliasing and its limits

Proposition 1 shows that bandlimiting a high-resolution input to the frequencies available at the pretraining resolution is not sufficient, in general, to eliminate discretization mismatch throughout a nonlinear FNO. This suggests a different strategy: map the high-resolution input back to the resolution on which the model was trained. In this way, the model only processes discretizations whose frequency range is compatible with that encountered during pretraining.

A natural implementation of this idea is *Compressed inference*, for which the Low+Pad method of Colagrande et al. (2026a) is an example. Given a downsampling map $D : \mathcal{X}_R \to \mathcal{X}_r$, one first compresses the high-resolution input, applies the pretrained operator $\mathcal{G}_r : \mathcal{X}_r \to \mathcal{Y}_r$, and finally maps the prediction back to the target resolution through an upsampling operator $U : \mathcal{Y}_r \to \mathcal{Y}_R$. The resulting predictor is therefore

$$\mathcal{G}^{\text{comp.}} = U \circ \mathcal{G}_r \circ D. \tag{3}$$

While compressed inference restores the resolution encountered during pretraining, it can discard high-resolution information before inference. We formalise this idea in Proposition 2.

Proposition 2 (Irreducible error of compressed inference)**.** *Let U_R be a random variable following the distribution of the inputs at high resolution and assume that $\mathcal{G}_R^\dagger(U_R) \in L^2(\mathcal{Y}_R)$. Let $D : \mathcal{X}_R \to \mathcal{X}_r$ be measurable, and let $\mathcal{C} : \mathcal{X}_r \to \mathcal{Y}_R$ be any measurable compressed predictor such that $\mathcal{C}(D(U_R)) \in L^2(\mathcal{Y}_R)$; in particular, $\mathcal{C}$ may be $U \circ \mathcal{G}_r$. Define $\overline{\mathcal{G}}_D(D(U_R)) := \mathbb{E}\Big[\mathcal{G}_R^\dagger(U_R) \mid D(U_R)\Big]$ as the conditional mean of the high-resolution target on the fibers of D, and define $\mathcal{V}_D := \mathbb{E}\left[\left\|\mathcal{G}_R^\dagger(U_R) - \overline{\mathcal{G}}_D(D(U_R))\right\|_R^2\right]$. Then*

$$\mathbb{E}\left[\left\|\mathcal{G}_R^\dagger(U_R) - \mathcal{C}(D(U_R))\right\|_R^2\right] = \mathcal{V}_D + \mathbb{E}\left[\left\|\overline{\mathcal{G}}_D(D(U_R)) - \mathcal{C}(D(U_R))\right\|_R^2\right]. \tag{4}$$

Consequently,

$$\inf_{\mathcal{C}} \mathbb{E}\left[\|\mathcal{G}_R^\dagger(U_R) - \mathcal{C}(D(U_R))\|_R^2\right] = \mathcal{V}_D, \tag{5}$$

where the infimum is over square-integrable measurable predictors based only on $D(U_R)$. Thus $\mathcal{V}_D$ is the irreducible error associated with any compressed inference method. Proof in Appendix E.2.

Motivated by these limitations, we introduce a tool to bridge the cross-resolution performance gap.

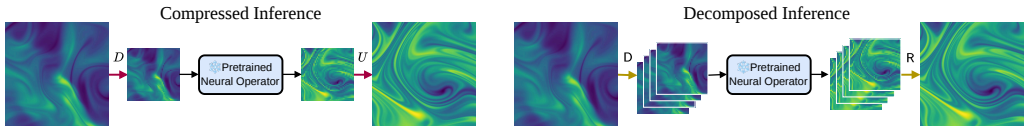

Figure 2: **Left:** Compressed inference: The input is downsampled, fed into the pretrained model and interpolated to obtain a high-resolution prediction. **Right:** Decomposed Inference that, similarly, applies the model solely at the pretraining resolution but, differently, without compression.

4 Method

4.1 Decomposed Inference

The previous results identify two distinct limitations: direct FNO inference can exhibit cross-resolution discretization mismatch due to aliasing, whereas compressed inference can lose target-relevant information through downsampling.

To avoid both sources of error, we propose a method, called *Decomposed Inference*, that works entirely at the pretraining resolution while preserving the information from the high-resolution representation via an invertible decomposition.

Let q denote the number of decomposed components. We consider a decomposition of inputs, $\mathsf{D}_{\mathcal{X}} : \mathcal{X}_R \to \mathcal{X}_r^q$, and a recomposition of outputs, $\mathsf{R}_{\mathcal{Y}} : \mathcal{Y}_r^q \to \mathcal{Y}_R$. When these maps are invertible, let $\mathsf{R}_{\mathcal{X}} := \mathsf{D}_{\mathcal{X}}^{-1}$ and $\mathsf{R}_{\mathcal{Y}} := \mathsf{D}_{\mathcal{Y}}^{-1}$. The pretrained operator $\mathcal{G}_r$ is applied independently to each input component. The resulting low-resolution predictions are then recomposed at the target resolution. Thus, Decomposed Inference has the form

$$\mathcal{G}_r^{\mathsf{D}} := \mathsf{R}_{\mathcal{Y}} \circ \mathcal{G}_r^{\times q} \circ \mathsf{D}_{\mathcal{X}}, \quad \text{with} \quad \mathcal{G}_r^{\times q}((z_1, \dots, z_q)) = (\mathcal{G}_r(z_1), \dots, \mathcal{G}_r(z_q)). \tag{6}$$

Compressed inference is contained as a special case of this general construction: one input component can be chosen as the downsampled field $D(u_R)$, while the recomposition can ignore the remaining predicted components and apply the upsampling map U to the first one. We detail this construction in Appendix E.1.

4.2 Choosing the Decomposition

Having established the motivation for Decomposed Inference, the remaining question is how to choose the decomposition and recomposition operators.

For the following risk analysis, we restrict to *exact isometric decompositions*: $\mathsf{D}_{\mathcal{X}}$ is a bimeasurable bijection, $\mathsf{D}_{\mathcal{Y}}$ is a linear isometric isomorphism, and $\mathsf{R}_{\mathcal{X}} = \mathsf{D}_{\mathcal{X}}^{-1}$, $\mathsf{R}_{\mathcal{Y}} = \mathsf{D}_{\mathcal{Y}}^{-1}$. The product output space is equipped with the normalized norm $\|(y_1, \dots, y_q)\|_{\oplus}^2 := \frac{1}{q} \sum_{k=1}^{q} \|y_k\|_r^2$,

Our goal is to transfer, via Decomposed inference, a pretrained low-resolution predictor $\mathcal{G}_r$ to the high-resolution problem without degrading the prediction risk achieved at the pretraining resolution. To this end, we analyze conditions under which $\mathcal{R}_r(\mathcal{G}_r)$ coincides with the high-resolution risk of its decomposed extension $\mathcal{R}_R(\mathcal{G}_r^{\mathsf{D}})$.

Intuitively, an ideal decomposition is one in which each component individually reproduces the data distribution observed during pretraining. We formalize this through the following notion.

Definition 1 (Component-wise marginal invariance)**.** *Let $(U_r, V_r) \sim \mu_r$ and $(U_R, V_R) \sim \mu_R$ denote the data pairs at the two resolutions. An input–output decomposition $(\mathsf{D}_{\mathcal{X}}, \mathsf{D}_{\mathcal{Y}})$ satisfies* component-wise marginal invariance *if*

$$\big(\mathsf{D}_k^{\mathcal{X}}(U_R), \mathsf{D}_k^{\mathcal{Y}}(V_R)\big) \overset{d}{=} (U_r, V_r), \qquad k = 1, \dots, q. \tag{7}$$

Proposition 3 (Risk preservation under marginal invariance)**.** *Let $(\mathsf{D}_{\mathcal{X}}, \mathsf{D}_{\mathcal{Y}})$ be an exact isometric decomposition satisfying component-wise marginal invariance. Then, for any square-integrable low-resolution predictor $\mathcal{G}_r$,*

$$\mathcal{R}_R(\mathcal{G}_r^{\mathsf{D}}) = \mathcal{R}_r(\mathcal{G}_r). \tag{8}$$

Additional details and the complete proof are given in Appendix F.2.

This suggests a practical design principle: choose a decomposition whose marginal invariance is induced by structural properties that are exactly, or approximately, satisfied by the tasks of interest.

4.3 Polyphase Decomposition

Many physical systems are translation invariant, and physical fields often exhibit spatial regularity even when exact distributional translation invariance is absent. These observations motivate the use of the *polyphase decomposition*.

Assume that the fine grid refines the coarse grid by an integer factor m along each of the d spatial dimensions,

$$R = mr, \qquad \Lambda_m := \{0, \dots, m-1\}^d, \qquad q := |\Lambda_m| = m^d. \tag{9}$$

For each $k \in \Lambda_m$, the k-th polyphase component of a high-resolution field u_R is defined as

$$(\mathsf{P}_k u_R)(j) = u_R(mj + k), \qquad j \in \{0, \dots, r-1\}^d, \tag{10}$$

Polyphase decomposition and recomposition are respectively

$$\mathsf{P}(u_R) = \big(\mathsf{P}_k u_R\big)_{k \in \Lambda_m}, \qquad \mathsf{P}^{-1}\big((u_k)_{k \in \Lambda_m}\big)(mj + k) = u_k(j). \tag{11}$$

With the normalized discrete L^2 norms, this rearrangement is an exact isometric decomposition.

Each component corresponds to a translated version of the same coarse sublattice. Translation invariance of μ_R therefore implies that all polyphase components have the same joint distribution. To conclude component-wise marginal invariance with respect to the *pretraining* distribution μ_r, one additionally needs one reference phase, k, to satisfy $(\mathsf{P}_k^{\mathcal{X}} U_R, \mathsf{P}_k^{\mathcal{Y}} V_R) \overset{d}{=} (U_r, V_r)$.

Note that when exact translation invariance does not hold, spatial Sobolev regularity controls differences between phase distributions; comparison with μ_r additionally contains the mismatch of a reference phase with the pretraining distribution. These statements are formalized in Appendix F.4.

Our decomposition should not be confused with the lifting and projection operators commonly used in FNOs. FNO lifting and projection transform features pointwise without changing the spatial discretization; our decomposition reorganizes a fine grid into multiple coarse-grid components, enabling inference at the pretraining resolution without learned parameters. Where specified, we apply a smoothing operator to the recomposed prediction. PyTorch code is provided in Appendix I.

4.4 The remaining cross-component gap and the role of finetuning

Marginal invariance removes the component-wise distribution mismatch introduced when transferring the low-resolution predictor to the decomposed high-resolution problem. However, it does not account for predictive information shared across decomposed components. The decomposition of Proposition 4 makes this distinction precise (complete proof in Appendix F.3) .

Proposition 4 (Error decomposition)**.** *Let $(\mathsf{D}_{\mathcal{X}}, \mathsf{D}_{\mathcal{Y}})$ be an exact isometric decomposition and let $\mathcal{G}_r$ be a low-resolution predictor such that $\mathcal{G}_r(\mathsf{D}_k^{\mathcal{X}}(U_R)) \in L^2(\mathcal{Y}_r)$ for every k, $V_R \in L^2(\mathcal{Y}_R)$. Then*

$$\mathcal{R}_R\big(\mathcal{G}_r^{\mathsf{D}}\big) - \mathcal{R}_R^B = \mathcal{E}_{\mathrm{in}}^2(\mathsf{D}, \mathcal{G}_r) + \mathcal{E}_{\mathrm{cross}}^2(\mathsf{D}), \tag{12}$$

where the in-component error is:

$$\mathcal{E}_{\mathrm{in}}^2(\mathsf{D}, \mathcal{G}_r) := \frac{1}{q} \sum_{k=1}^{q} \mathbb{E}\Big[\big\| \mathcal{G}_r(\mathsf{D}_k^{\mathcal{X}} U_R) - \mathbb{E}[\mathsf{D}_k^{\mathcal{Y}} V_R \mid \mathsf{D}_k^{\mathcal{X}} U_R] \big\|_r^2 \Big], \tag{13}$$

and the cross-component error is:

$$\mathcal{E}_{\mathrm{cross}}^2(\mathsf{D}) := \frac{1}{q} \sum_{k=1}^{q} \mathbb{E}\Big[\big\| \mathbb{E}[\mathsf{D}_k^{\mathcal{Y}} V_R \mid \mathsf{D}_k^{\mathcal{X}} U_R] - \mathbb{E}[\mathsf{D}_k^{\mathcal{Y}} V_R \mid U_R] \big\|_r^2 \Big]. \tag{14}$$

These two errors are fundamentally different: (i) the *in-component error* depends on the pretrained predictor and measures the average error made within each decomposed component, while (ii) the *cross-component error* doesn't depend on the model, but just on the decomposition, and quantifies predictive information about each output component that is contained in the full input but is not measurable with respect to the corresponding input component alone.

4.5 Polyphase Finetuning

The error decomposition of Proposition 4 suggests two adaptation mechanisms: PEFT of $\mathcal{G}_r$ can reduce the in-component error, while reducing the cross-component term requires exploiting interactions across components. We therefore separate model adaptation from cross-component adaptation.

For model adaptation, we use established PEFT methods such as LoRA Hu et al. (2022), Adapter (Houlsby et al., 2019), and F-Adapter (Zhang et al., 2026). For cross-component adaptation, we exploit that convolution on the fine grid induces structured interactions between polyphase components. Indeed, for $(K * u)[n] = \sum_{s \in \mathbb{Z}^d} K[s]\, u[n-s]$, the polyphase representation satisfies

$$P_k(K * u) = \sum_{\ell \in \Lambda_m} K_{k,\ell} * P_\ell u, \qquad K_{k,\ell}[q] = K[mq + k - \ell], \tag{15}$$

for every $k \in \Lambda_m$. Hence, we use lightweight depthwise convolutions before decomposition and after recomposition, initialized such that finetuning starts from Polyphase Inference (implementation details in Appendix L.

5 EXPERIMENTS

5.1 ZERO-SHOT TRANSFER AT HIGHER RESOLUTIONS WITH FNO

We consider a $1D$ time-dependent problem, Burgers (see App.G.1), a $2D$ steady state system, Darcy flow (G.2), and two $2D$ time-dependent problems, Reaction-Diffusion (G.3) and Compressible Navier-Stokes(G.4). We follow the training protocol of (Sakarvadia et al., 2026) where a pretrained FNO retains all the modes available at the pretraining resolution and the resolutions used for pretraining are $\frac{1}{2}$, $\frac{1}{4}$ and $\frac{1}{8}$ of the original dataset resolution (i.e. $m = R/r \in \{2, 4, 8\}$). We consider as a baseline Low+Pad (Colagrande et al., 2026a), an example of Compressed inference method that downsamples by Fourier truncation at the pretraining resolution, then performs the inference with FNO and finally upsamples the output by zero-padding in Fourier space.

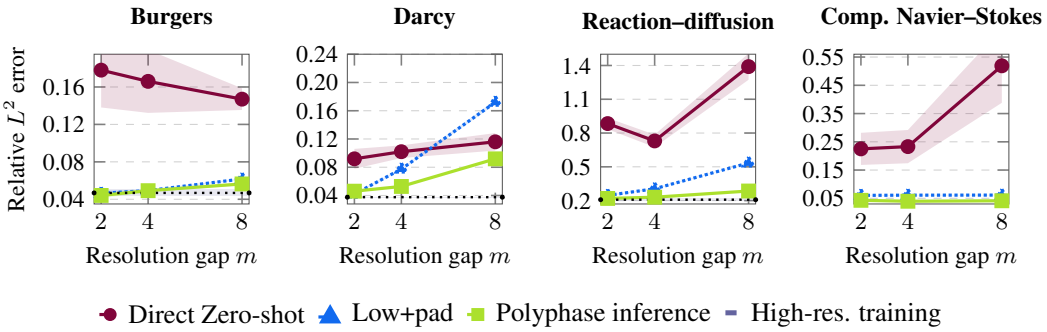

Figure 3: Test relative L^2 error versus the resolution gap $m = R/r$, where R is the fixed target resolution and r is the pretraining resolution, for zero-shot transfer. Models are pretrained at various low resolutions ($m \in \{2, 4, 8\}$) and evaluated at a fixed target resolution. **Lower is better**.

We observe in Figure 3, as also reported in (Sakarvadia et al., 2026), that FNO degrades substantially when tested directly at a resolution higher than the one seen during pretraining. Results show that Polyphase inference performs close to Low+Pad on Burgers, while becoming increasingly advantageous as the resolution gap grows on Darcy. The improvement is more pronounced on Reaction–Diffusion and compressible Navier–Stokes, where Polyphase inference consistently achieves the lowest error across all resolution gaps. Overall, these results suggest that preserving high-resolution information through decomposition is particularly beneficial for larger resolution gaps and for systems with richer fine-scale spatial structure. (Detailed result tables are in Appendix H).

5.2 FINE-TUNING AN FNO

We consider the problem where we assume access to high-resolution data amounting to 1% and 10% of the low-resolution pretraining data set. Using the same datasets as in Section 5.1, we compare three PEFT methods (LoRA (Hu et al., 2022), Adapter (Houlsby et al., 2019), F-Adapter Zhang et al. (2026)) on FNO at high resolution, applied (i) directly and (ii) together with Polyphase inference. We also evaluate polyphase inference alone (PPFT), in which we simply fine-tune the parameters of the decomposition and recomposition operators.

Figure 4 shows that PEFT methods improve FNO performance at higher resolution but do not reach the zero-shot polyphase inference strategy, which does not require data or training. When PEFT methods are used together with Polyphase inference, the results further improve, with larger gains for Darcy flow and little improvement for Reaction-Diffusion and Burgers, where Polyphase inference is already very close to the high-resolution training performance.

5.3 COMPARISON ACROSS MODEL ARCHITECTURES

Here, we train models at $1/4$ of the original resolution and test at the highest resolution, using the same four datasets described in Section 5.1. We consider four types of architectures: FNO, UNet, the transformer-based Transolver, and LaMO, a state-space model. As illustrated in Fig. 5, our results show that Polyphase inference enables effective zero-shot transfer to higher resolutions across all

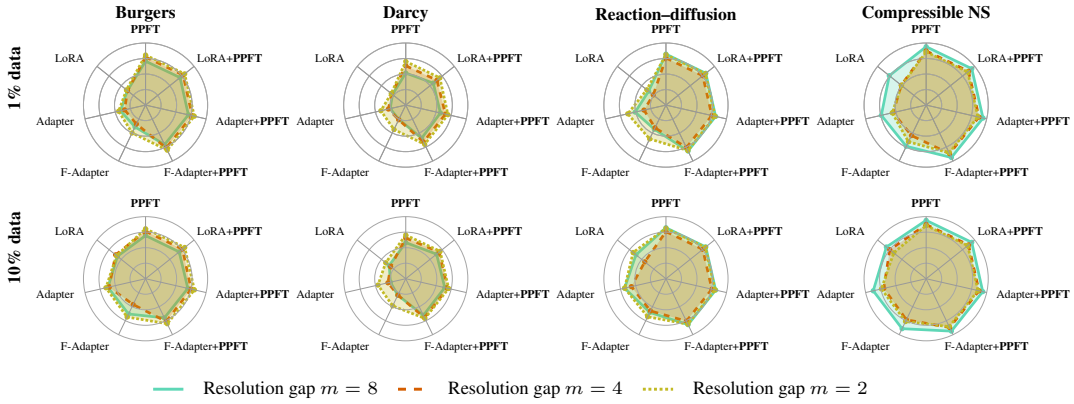

Figure 4: Relative reduction of L^2 error compared to zero-shot FNO for PEFT methods with and without Polyphase method under a budget of 1% and 10% training data. The radial scale runs from -33.3% at the center to $+100\%$ improvement at the edge, and the innermost grid circle marks no change. Colours and line styles denote resolution gaps $m = R/r$ between pretraining and test data. **Higher is better**. **Polyphase fine-tuning (PPFT) consistently outperforms LoRA, and combining polyphase decomposition with existing PEFT methods (LoRA, Adapter, F-Adapter) generally improves upon them too.**

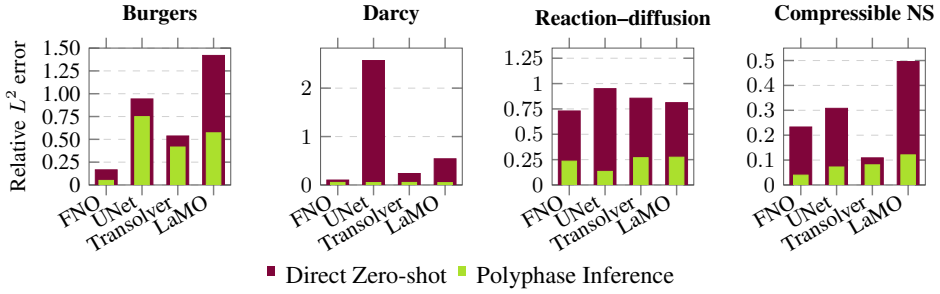

Figure 5: Relative L^2 error for direct zero-shot evaluation and Polyphase inference at a resolution 4 times higher than the pretraining one. **Lower is better. Polyphase consistently improves upon zero-shot inference for all tested architectures (FNO, Unet, Transolver, LaMO) and datasets (Burgers, Darcy, Reaction-Diffusion, Compressible Navier-Stockes)**.

datasets and architectures. Polyphase consistently improves over direct high-resolution inference. The largest gains occur for models that degrade most severely under resolution shift. For example, on Darcy Flow, direct UNet inference performs substantially worse than FNO, whereas with Polyphase its error becomes comparable to, and even slightly lower than, that of Polyphase FNO. A similar effect is observed for LaMO on compressible Navier-Stokes: while direct evaluation is markedly worse than in the case of FNO, Polyphase narrows the gap between models and brings them to a much more comparable level of performance.

5.4 Adaptation of Multi-Physics Foundation Models

We investigate how large-scale models pretrained across different kinds of physical systems generalize to resolutions beyond those seen during pretraining. We focus on DPOT Hao et al. (2024), with models ranging from 7.5 million to 1.03 billion parameters, pretrained on 12 datasets at resolution 128×128. To evaluate these models at higher resolution on a problem they know well, we use the Compressible Navier–Stokes equations as in their pretraining set and generate a 1024×1024 test set with the official solver from PDEBench Takamoto et al. (2022).

Our findings are that, despite the variety in physics seen during pretraining, and the promise of high downstream performance when fine-tuning the pretrained model on high-resolution datasets, DPOT models do not maintain their zero-shot performance at out-of-distribution resolutions.

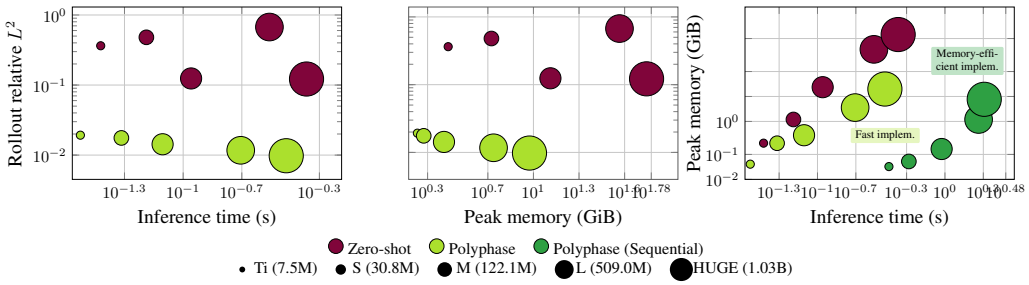

Figure 6: DPOT rollout relative L^2 error versus inference cost (Left) and vs Memory peak usage (Center), together with inference time versus peak memory (Right). Circle size denotes model size (Ti, S, M, L, HUGE). The left panel uses the parallel implementation of Polyphase inference, while the centre panel adopts the sequential implementation. **Lower left corner is better.**

Figure 6 shows that high-resolution evaluation of DPOT does not preserve the ordering induced by model size and native-resolution performance. In contrast, Polyphase inference restores this monotonic behavior: larger DPOT models, which perform better at the pretraining resolution, continue to outperform smaller variants when transferred to higher resolution through Polyphase inference.

The large gains of Polyphase inference over direct evaluation can be attributed to the larger pretraining distribution that makes marginal invariance easier to be satisfied. In other words, a component can be close to one of the distributions in the dataset used for pretraining, or a mixture of them.

5.5 Efficiency Analysis

Polyphase Inference can process the components either in parallel or sequentially, trading wall-clock time for memory usage. As illustrated in Figure 6, where we test, we evaluate the model on a single NVIDIA A100 using Compressible Navier–Stokes data generated at resolution 1024^2 with the same solver and PDE configuration as the PDEBench data (Takamoto et al., 2022) used for DPOT pretraining (Hao et al., 2024) that was done at resolution 128^2. Parallel execution is able to reduce inference time by exploiting hardware concurrency, while sequential execution reduces peak memory usage. These two alternatives provide flexibility for choosing a time–memory trade-off depending on the available infrastructure. On highly parallel hardware such as GPUs and TPUs, the independent low-resolution evaluations can be executed concurrently, allowing the wall-clock cost to approach that of a single low-resolution evaluation.

6 Conclusion and Future Work

We tackle the problem of transferring a Neural Operator at a higher resolution than during pretraining and introduce *Polyphase Inference*, a zero-shot strategy that leverages a geometric decomposition of the input to adapt the high-resolution data to the pretrained resolution. We extend it to the case when high-resolution samples are available and introduce *Polyphase fine-tuning* by making the decomposition learnable. Our results show that Polyphase can efficiently recover high-resolution inference capabilities across a range of architectures and datasets.

Limitations In this work, we focused on regular grids, and on the case where the resolution gap is an integer. Future works will address other geometries, irregular grids and point clouds. We are also interested in extending Polyphase Inference beyond physics to dense computer vision tasks, such as segmentation, as well as other structured modalities such as audio and time series.

AI USE STATEMENT

Generative AI tools were used as supporting tools for language editing, checking mathematical notation and consistency, code review and debugging assistance, and literature search. The research questions, proposed methodology, theoretical results, and experimental design were developed by the authors. Generative AI tools were not used to formulate the contributions of this work or to derive its theoretical results. All AI-assisted outputs, including text, code suggestions, and references, were manually reviewed and verified by the authors. The authors take full responsibility for the correctness and integrity of the final manuscript and its accompanying artifacts.

ETHICS STATEMENT

This work focuses on methodological and theoretical advances for scientific machine learning and does not involve human subjects, personal data, or sensitive information. The experiments rely on established scientific benchmarks and simulation data. We are not aware of specific ethical risks arising from the methods presented beyond the general considerations associated with the use of machine learning systems. We have conducted the research and reported the results with attention to reproducibility, transparency, and research integrity.

REPRODUCIBILITY STATEMENT

We have made several efforts to ensure the reproducibility of this work. The assumptions and definitions underlying our theoretical results are stated in the main text, with proofs provided in Appendices C-F. We describe the implementation of the method in Appendix I. Detailed descriptions of the datasets and data-generation procedures are given in Appendix G, while the complete quantitative results corresponding to the experiments in the main text are reported in Appendix H.

REFERENCES

Francesca Bartolucci, Emmanuel De Bézenac, Bogdan Raonic, Roberto Molinaro, Siddhartha Mishra, and Rima Alaifari. Representation equivalent neural operators: a framework for alias-free operator learning. *Advances in Neural Information Processing Systems*, 36:69661–69672, 2023.

Alex Colagrande, Paul Caillon, Eva Feillet, and Alexandre Allauzen. Linear attention with global context: A multipole attention mechanism for vision and physics. In *2025 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pp. 3130–3139. IEEE, 2025.

Alex Colagrande, Paul Caillon, Eva Feillet, and Alexandre Allauzen. Limits of resolution equivariance in fourier neural operators. *arXiv preprint arXiv:2606.00677*, 2026a.

Alex Colagrande, Paul Caillon, Eva Feillet, and Alexandre Allauzen. Higher-order fourier neural operator: Explicit mode mixer for nonlinear pdes. *arXiv preprint arXiv:2606.28122*, 2026b.

Wenhan Gao, Ruichen Xu, Yuefan Deng, and Yi Liu. Discretization-invariance? on the discretization mismatch errors in neural operators. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=J9FgrqOOni.

Zhongkai Hao, Chang Su, Songming Liu, Julius Berner, Chengyang Ying, Hang Su, Anima Anandkumar, Jian Song, and Jun Zhu. DPOT: Auto-regressive denoising operator transformer for large-scale PDE pre-training. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=X7UnDevHOM.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pp. 2790–2799. PMLR, 2019.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Jean Kossaifi, Nikola Kovachki, Kamyar Azizzadenesheli, and Anima Anandkumar. Multi-grid tensorized fourier neural operator for high-resolution pdes. *arXiv preprint arXiv:2310.00120*, 2023.

Jean Kossaifi, Nikola Kovachki, Zongyi Li, David Pitt, Miguel Liu-Schiaffini, Robert J George, Boris Bonev, Kamyar Azizzadenesheli, Julius Berner, Valentin Duruisseaux, et al. A library for learning neural operators. *Journal of Machine Learning Research*, 27(205):1–6, 2026.

Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

Samuel Lanthaler, Andrew M Stuart, and Margaret Trautner. Discretization error of fourier neural operators. *arXiv preprint arXiv:2405.02221*, 2024.

Zijie Li, Kazem Meidani, and Amir Barati Farimani. Transformer for partial differential equations' operator learning. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=EPPqt3uERT.

Zongyi Li, Nikola Borislavov 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.

Parviz Moin and Krishnan Mahesh. Direct numerical simulation: a tool in turbulence research. *Annual review of fluid mechanics*, 30(1):539–578, 1998.

Ruben Ohana, Michael McCabe, Lucas Meyer, Rudy Morel, Fruzsina J Agocs, Miguel Beneitez, Marsha Berger, Blakesley Burkhart, Stuart B Dalziel, Drummond B Fielding, et al. The well: a large-scale collection of diverse physics simulations for machine learning. *Advances in Neural Information Processing Systems*, 37:44989–45037, 2024.

Nusrat Jahan Prottasha, Upama Roy Chowdhury, Shetu Mohanto, Tasfia Nuzhat, Abdullah As Sami, Md Shamol Ali, Md Shohanur Islam Sobuj, Hafijur Raman, Md Kowsher, and Ozlem Ozmen Garibay. Peft a2z: Parameter-efficient fine-tuning survey for large language and vision models. *arXiv preprint arXiv:2504.14117*, 2025.

Bogdan Raonic, Roberto Molinaro, Tim De Ryck, Tobias Rohner, Francesca Bartolucci, Rima Alaifari, Siddhartha Mishra, and Emmanuel De Bézenac. Convolutional neural operators for robust and accurate learning of pdes. *Advances in Neural Information Processing Systems*, 36:77187–77200, 2023.

Mansi Sakarvadia, Kareem Hegazy, Amin Totounferoush, Kyle Chard, Yaoqing Yang, Ian Foster, and Michael W Mahoney. The false promise of zero-shot super-resolution in machine-learned operators. In *International Conference on Learning Representations*, volume 2026, pp. 69635–69688, 2026.

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

Karn Tiwari, Niladri Dutta, N M Anoop Krishnan, and Prathosh AP. Latent mamba operator for partial differential equations. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=Y4BDcJmb8t.

Alasdair Tran, Alexander Mathews, Lexing Xie, and Cheng Soon Ong. Factorized fourier neural operators. *arXiv preprint arXiv:2111.13802*, 2021.

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.

Zipeng Xiao, Zhongkai Hao, Bokai Lin, Zhijie Deng, and Hang Su. Improved operator learning by orthogonal attention. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 54288–54299. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/xiao24c.html.

Hangwei Zhang, Chun Kang, Yan Wang, and Difan Zou. F-adapter: Frequency-adaptive parameter-efficient fine-tuning in scientific machine learning. *Advances in Neural Information Processing Systems*, 38:111120–111162, 2026.

Appendix

A Notations **15**

B Extended Related Work **16**

C Preliminaries **17**

C.1 Resolution-dependent learning problems 17

D Discretization mismatch in FNO **18**

D.1 Preliminaries 18

D.2 Previous theoretical results 19

D.3 Layer-wise Upper bound of Discretization Mismatch of FNO 20

E Compressed and Decomposed Inference **24**

E.1 Compressed inference as a special case of decomposed inference 24

E.2 Irreducible error of compressed inference 24

F Decomposed inference: theoretical perspective **26**

F.1 Exact isometric decompositions 26

F.2 Risk preservation under marginal invariance 26

F.3 Error decomposition of decomposed inference 26

F.4 Translation invariance and Polyphase Decomposition 28

F.5 Approximate phase invariance under spatial regularity 29

G Datasets **30**

G.1 Burgers' equation (Takamoto et al., 2022) 30

G.2 Darcy flow (Takamoto et al., 2022) 30

G.3 Reaction-Diffusion (Takamoto et al., 2022) 31

G.4 Compressible Navier–Stokes (Takamoto et al., 2022) 31

H Experimental Tables **33**

H.1 Zero-Shot Transfer At Higher Resolution with FNO 33

H.2 Finetuning 35

H.3 Comparison across model architectures 39

H.4 DPOT Experiments 40

I PyTorch implementations of Polyphase inference **42**

J Experimental Details **44**

K Training and evaluation hyperparameters **44**

L Few-shot finetuning and testing **46**

A NOTATIONS

Symbol	Meaning
Ω	Spatial domain.
$\mathbb{T}^d$	d-dimensional torus, used as a periodic spatial domain in the FNO analysis.
d	Number of spatial dimensions.
$x \in \Omega$	Spatial coordinate.
$[0, T]$	Temporal domain, when time-dependent PDEs are considered.
$u(x, t)$	Physical field evaluated at (x, t).
$r,\ R$	Low (pretraining) and high (target) spatial resolutions, measured per spatial dimension.
$m := R/r$	Integer refinement factor along each spatial dimension; when used in the analysis, $R = mr$.
q	Number of decomposed components. For a d-dimensional decomposition, $q = m^d$.
Ω_{s}	Uniform grid at resolution $\mathsf{s} \in \{r, R\}$.
Λ_{s}	Set of discrete Fourier indices at resolution s in the FNO analysis.
$\Lambda_m := \{0, \dots, m-1\}^d$	Set of polyphase offsets; $\lvert\Lambda_m\rvert = q = m^d$.
$\mathcal{X}_{\mathsf{s}},\ \mathcal{Y}_{\mathsf{s}}$	Input and output spaces at resolution $\mathsf{s} \in \{r, R\}$.
$(U_{\mathsf{s}}, V_{\mathsf{s}}) \sim \mu_{\mathsf{s}}$	Random input-output pair and its joint distribution at resolution s.
$\mathcal{G}^\dagger_{\mathsf{s}}$	Exact discretized solution operator at resolution s.
$\mathcal{G}_r : \mathcal{X}_r \to \mathcal{Y}_r$	Predictor pretrained at the low resolution r.
$\mathcal{R}_{\mathsf{s}}(\mathcal{G})$	Squared prediction risk $\mathbb{E}[\lVert\mathcal{G}(U_{\mathsf{s}}) - V_{\mathsf{s}}\rVert^2_{\mathsf{s}}]$.
$\mathcal{B}_{\mathsf{s}}$	Bayes predictor $\mathcal{B}_{\mathsf{s}}(u) = \mathbb{E}[V_{\mathsf{s}} \mid U_{\mathsf{s}} = u]$.
$\mathcal{R}^B_{\mathsf{s}}$	Bayes risk $\mathcal{R}_{\mathsf{s}}(\mathcal{B}_{\mathsf{s}})$.
$\mathcal{P},\ \mathcal{Q}$	Pointwise FNO lifting and projection maps, respectively.
L	Number of FNO layers.
$v_\ell,\ z_\ell$	Post-activation and pre-activation FNO features at layer ℓ.
W_ℓ	Pointwise linear map in FNO layer ℓ.
$\mathcal{K}_\ell$	Fourier convolution operator in FNO layer ℓ.
σ_ℓ	Pointwise nonlinearity in FNO layer ℓ.
$\mathcal{M}$	the set of Fourier modes retained by the FNO
$\mathcal{F}$	Continuous Fourier transform/Fourier coefficient map on $\mathbb{T}^d$.
$\mathcal{F}_{\mathsf{s}}$	Discrete Fourier transform of samples at resolution s.
I_{s}	Trigonometric interpolation operator at resolution s.
$\Delta_{r,R}(f)$	Aliasing discrepancy $\lVert(I_R - I_r)f\rVert_{L^2}$.
s	Sobolev regularity exponent.
H^s	Sobolev space of exponent s.
$D,\ U$	Downsampling and upsampling operators used in compressed inference.
$\mathsf{D}_{\mathcal{X}},\ \mathsf{D}_{\mathcal{Y}}$	Decomposition maps on the high-resolution input and output spaces.
$\mathsf{R}_{\mathcal{X}},\ \mathsf{R}_{\mathcal{Y}}$	Corresponding recomposition maps; for an exact decomposition they are the inverses of $\mathsf{D}_{\mathcal{X}}$ and $\mathsf{D}_{\mathcal{Y}}$.
$\mathcal{G}^{\mathsf{D}}_r$	Decomposed extension $\mathsf{R}_{\mathcal{Y}} \circ \mathcal{G}_r^{\times q} \circ \mathsf{D}_{\mathcal{X}}$.
$\lVert\cdot\rVert_\oplus$	Normalized product norm $\lVert(y_k)_{k=1}^q\rVert^2_\oplus = q^{-1}\sum_{k=1}^q \lVert y_k\rVert^2_r$.
$\mathsf{P},\ \mathsf{P}_k$	Polyphase decomposition and its k-th component.
τ_a	Translation by spatial offset a; a is used for a shift so that s remains reserved for Sobolev regularity.
$\mathcal{E}^2_{\mathrm{in}},\ \mathcal{E}^2_{\mathrm{cross}}$	In-component and cross-component terms in the excess-risk decomposition of decomposed inference.

Table 1: Main mathematical notation used in this work.

B Extended Related Work

Neural operators and cross-resolution generalization. Neural operators learn mappings between function spaces and, at the level of their parameterization, can be evaluated on multiple discretizations (Kovachki et al., 2023). Fourier Neural Operators (FNOs) (Li et al., 2021), and related Fourier-based Neural Operators ((Wen et al., 2022; Tran et al., 2021; Kossaifi et al., 2023; Colagrande et al., 2026b), are a prominent instance that parameterizes global integral operators in Fourier space.Transformer-based (Li et al., 2023; Xiao et al., 2024; Colagrande et al., 2025) have been proposed with the same goal of learning mappings between function spaces. However, resolution-independent parameterization does not by itself guarantee resolution-independent numerical predictions. ReNO formalizes representation equivalence and operator aliasing (Bartolucci et al., 2023), while CNO explicitly constructs continuous–discrete equivalent convolutional architectures (Raonic et al., 2023). More recently, Gao et al. (2025) characterize discretization mismatch errors and introduce CROP, and Sakarvadia et al. (2026) provide a systematic empirical study showing that zero-shot super-resolution can fail under changes in both discretization and represented frequency content. Recent work on FNOs also shows that direct fine-grid evaluation can underperform inference at the training resolution followed by interpolation (Colagrande et al., 2026a). In contrast to methods that modify the architecture or training procedure to improve cross-resolution consistency, *our approach keeps the pretrained model unchanged and performs all model evaluations at its pretraining resolution after an exact decomposition of the fine grid*.

High-resolution operator learning and adaptation. High-resolution PDE learning has also been addressed through architectures designed explicitly for large grids. For instance, MG-TFNO combines domain decomposition with tensorized Fourier operators to reduce the memory and parameter cost of high-resolution operator learning (Kossaifi et al., 2023). Our objective is different: we target the adaptation of an already pretrained low-resolution model to a finer discretization without requiring high-resolution retraining in the zero-shot setting. When a small amount of high-resolution data is available, our setting is also related to parameter-efficient fine-tuning (PEFT), including adapters (Houlsby et al., 2019), LoRA (Hu et al., 2022), and the scientific-machine-learning-specific F-Adapter (Zhang et al., 2026). Our Polyphase Fine-Tuning instead *adapts the decomposition/recomposition interface around the pretrained operator, rather than requiring the pretrained operator itself to be evaluated outside its training resolution*.

C PRELIMINARIES

We collect here the common definitions and identities used throughout the theoretical analysis.

C.1 RESOLUTION-DEPENDENT LEARNING PROBLEMS

Let $r < R$ denote respectively the pretraining and target spatial resolutions, that will often be called, respectively, low- and high-resolution.

For each resolution $\mathsf{s} \in \{r, R\}$, let $\mathcal{X}_\mathsf{s}$ and $\mathcal{Y}_\mathsf{s}$ denote the corresponding input and output spaces, and let

$$(U_\mathsf{s}, V_\mathsf{s}) \sim \mu_\mathsf{s} \tag{16}$$

be the associated input-output random pair where U_s and V_s are random variables with joint probability distribution μ_s.

For a measurable predictor $\mathcal{G} : \mathcal{X}_\mathsf{s} \to \mathcal{Y}_\mathsf{s}$, we define its squared prediction risk at resolution s as

$$\mathcal{R}_\mathsf{s}(\mathcal{G}) := \mathbb{E}\left[\|\mathcal{G}(U_\mathsf{s}) - V_\mathsf{s}\|^2\right]. \tag{17}$$

Under the squared loss, the Bayes predictor, that is the optimal predictor in L^2-norm, is given by the conditional mean

$$\mathcal{B}_\mathsf{s}(u) := \mathbb{E}\left[V_\mathsf{s} \mid U_\mathsf{s} = u\right], \tag{18}$$

and its risk is denoted by

$$\mathcal{R}_\mathsf{s}^\mathcal{B} := \mathcal{R}_\mathsf{s}(\mathcal{B}_\mathsf{s}). \tag{19}$$

The super-resolution transfer problem considered throughout the paper consists in reusing a predictor

$$\mathcal{G}_r : \mathcal{X}_r \to \mathcal{Y}_r \tag{20}$$

learned from the low-resolution distribution μ_r to construct a predictor on the high-resolution space $\mathcal{X}_R \to \mathcal{Y}_R$.

In the theoretical analysis, we will often consider the idealized case in which the pretrained predictor is the low-resolution Bayes predictor $\mathcal{G}_r^B$. This isolates the error induced by cross-resolution transfer from the approximation and optimization errors of the learned model.

D DISCRETIZATION MISMATCH IN FNO

In order to study the behaviour of Fourier Neural Operators across resolutions, we derive a direct bound on the discrepancy between the same Fourier Neural Operator evaluated at two different spatial resolutions, R and r.

D.1 PRELIMINARIES

Throughout this section, the spatial domain is the d-dimensional torus $\mathbb{T}^d$. For $\mathsf{s} \in \mathbb{N}$, let

$$\Omega_{\mathsf{s}} := \left\{ \frac{n}{\mathsf{s}} : n \in \{0, \dots, \mathsf{s}-1\}^d \right\} \tag{21}$$

be the uniform s^d grid. We denote by $\Lambda_{\mathsf{s}} \subset \mathbb{Z}^d$ a fixed set of representatives of $\mathbb{Z}^d/\mathsf{s}\mathbb{Z}^d$, chosen such that

$$\Lambda_r \subseteq \Lambda_R \qquad \text{whenever} \qquad R = mr, \quad m \in \mathbb{N}. \tag{22}$$

For a sufficiently regular function $f : \mathbb{T}^d \to \mathbb{R}^m$, define its discrete Fourier coefficients at resolution s by

$$\mathcal{F}_{\mathsf{s}}(f)(k) := \frac{1}{s^d} \sum_{x \in \Omega_{\mathsf{s}}} f(x) e^{-2\pi i k \cdot x}, \qquad k \in \Lambda_{\mathsf{s}}, \tag{23}$$

and denote by

$$I_{\mathsf{s}} f(x) := \sum_{k \in \Lambda_{\mathsf{s}}} \mathcal{F}_{\mathsf{s}}(f)(k) e^{2\pi i k \cdot x} \tag{24}$$

its trigonometric interpolant(Lanthaler et al., 2024).

We recall the mathematical formulation of the Fourier Neural Operator used throughout the analysis. Let $\mathcal{X}$ and $\mathcal{Y}$ be suitable spaces of functions defined on a spatial domain Ω. We denote by

$$\mathcal{G}_r : \mathcal{X} \to \mathcal{Y} \tag{25}$$

a fixed FNO pretrained at spatial resolution r. The subscript r refers exclusively to the resolution used during pretraining, the same model and parameters are used when $\mathcal{G}_r$ is evaluated at any other resolution.

An L-layer FNO can be written as the composition

$$\mathcal{G}_r = Q \circ \mathcal{L}_{L-1} \circ \cdots \circ \mathcal{L}_0 \circ P, \tag{26}$$

where P and Q are pointwise lifting and projection maps, respectively, and $\mathcal{L}_\ell$ denotes the ℓ-th Fourier layer.

Given an input field $u \in \mathcal{X}$, the lifting map produces the initial latent representation

$$v_0(x) = P(u(x)). \tag{27}$$

For $\ell = 0, \dots, L-1$, the Fourier layers are defined recursively by

$$z_\ell(x) = W_\ell v_\ell(x) + \mathcal{K}_\ell v_\ell(x) + b_\ell, \tag{28}$$

$$v_{\ell+1}(x) = \sigma_\ell \left(z_\ell(x) \right), \tag{29}$$

where W_ℓ is a pointwise linear map, b_ℓ is a bias, σ_ℓ is a pointwise nonlinear activation, and $\mathcal{K}_\ell$ is a non-local Fourier convolution operator.

Let $\mathcal{M} \subset \mathbb{Z}^D$ denote the set of Fourier modes retained by the FNO. The spectral convolution is given by

$$\mathcal{K}_\ell v = \mathcal{F}^{-1} \left(R_\ell^{(k)} \, \mathcal{F}(v)(k) \right)_{k \in \mathcal{M}}, \tag{30}$$

where $R_\ell^{(k)}$ is the learned complex-valued matrix acting on the channel dimension of Fourier mode k. Frequencies outside $\mathcal{M}$ are set to zero.

Finally, the output of the network is obtained through the pointwise projection

$$\mathcal{G}_r(u)(x) = Q(v_L(x)). \tag{31}$$

Although the FNO is naturally formulated as an operator between function spaces, its numerical implementation acts on discretizations of functions. For a spatial resolution s, let Ω_{s} denote the corresponding uniform grid and let

$$u_{\mathsf{s}} := u|_{\Omega_{\mathsf{s}}} \tag{32}$$

denote the discretization of u on this grid. Evaluating the fixed model $\mathcal{G}_r$ on u_{s} produces a discrete output on the same grid, $\mathcal{G}_r(u_{\mathsf{s}})$. In particular, evaluating the same pretrained FNO at resolutions r and R gives

$$\mathcal{G}_r(u_r) \qquad \text{and} \qquad \mathcal{G}_r(u_R), \tag{33}$$

respectively. These are produced by the same model parameters, but are discretized on grids of different sizes.

To compare such outputs in a common function space, we follow the literature (Gao et al., 2025; Lanthaler et al., 2024) and for each resolution s we identify the discrete output $\mathcal{G}_r(u_{\mathsf{s}})$ with its continuous representative

$$I_{\mathsf{s}}\left[\mathcal{G}_r(u_{\mathsf{s}})\right] = I_{\mathsf{s}}\left[\mathcal{Q}_r(v_L^{\mathsf{s}})\right]. \tag{34}$$

Consequently, the discrepancy between evaluations of the same FNO at resolutions r and R is measured as

$$\|\mathcal{G}_r(u_R) - \mathcal{G}_r(u_r)\|_{L^2(\Omega)} := \|I_R\left[\mathcal{G}_r(u_R)\right] - I_r\left[\mathcal{G}_r(u_r)\right]\|_{L^2(\Omega)}. \tag{35}$$

D.2 Previous theoretical results

We first recall two theoretical results on the discretization error of Fourier Neural Operators that motivate our analysis. (Gao et al., 2025) directly compare the same FNO evaluated at two finite resolutions, while (Lanthaler et al., 2024) study the convergence of a discrete FNO toward its continuum counterpart. We refer to the original works for the corresponding proofs.

Proposition 5 (Discretization mismatch of FNOs, (Gao et al., 2025))**.** *Let $r < R$ be two spatial resolutions. Let $\mathcal{G}_r$ denotes the L-layer FNO pretrained at resolution r. Let ω denote the Lipschitz constant of the activation function. Then the discretization mismatch satisfies*

$$\|\mathcal{G}_r(u_R) - \mathcal{G}_r(u_r)\|_{L^2} \le C\omega^L \left(\prod_{\ell=1}^{L} C_\ell\right) \sum_{\xi=0}^{r-1} \left[\frac{2\pi|r-R|}{rR}\,\xi \left\|\sum_{n=1}^{r} u(\frac{n}{r}) n e^{-i2\pi\frac{n}{r}\xi}\right\| + \left\|\sum_{n=r+1}^{R} u(\frac{n}{R}) e^{-i2\pi\frac{n}{R}\xi}\right\|\right], \tag{36}$$

where

$$C_\ell = \|W_\ell\| + C_{\kappa_\ell}, \tag{37}$$

C_{κ_ℓ} depends on the convolution kernel of layer ℓ, and the constants are independent of r, R, and L.

Proposition 5 is stated in one spatial dimension. The same argument extends to d-dimensional grids by replacing the scalar Fourier index with a multi-index. We therefore view it as the finite-to-finite counterpart of the discretization problem considered in our setting.

The bound expresses the discrepancy induced by the two input discretizations and its amplification through the subsequent FNO layers.

A complementary perspective is provided by Lanthaler et al. (2024), who compare a finite-resolution FNO with its continuum counterpart.

Proposition 6 (Discretization error of FNOs, (Lanthaler et al., 2024))**.** *Let v_ℓ denote the continuum representation after layer ℓ and let v_ℓ^r denote the corresponding representation obtained by evaluating the FNO on a uniform grid of resolution r in each spatial dimension. Assume that the input belongs to $H^s(\mathbb{T}^d)$ with $\mathsf{s} > d/2$. Then, for every layer ℓ, there exists a constant C independent of r such that*

$$\frac{1}{r^{d/2}}\,\|v_\ell - v_\ell^r\|_{\ell^2} \le Cr^{-s}. \tag{38}$$

Equivalently, letting $I_r v_\ell^r$ denote the trigonometric interpolant of the discrete representation, there exists a constant C' independent of r such that

$$\|v_\ell - I_r v_\ell^r\|_{L^2(\mathbb{T}^d)} \le C' r^{-s}. \tag{39}$$

The analysis of Lanthaler et al. (2024) further shows that aliasing is not restricted to the input discretization. Pointwise nonlinearities can generate unresolved frequencies at intermediate layers, introducing new discretization errors that are subsequently propagated through the remaining network.

The two results therefore provide complementary perspectives. Proposition 5 directly compares the same FNO at two finite resolutions, but bounds the resulting discrepancy through the mismatch induced by the input discretization and its propagation through the network. Proposition 6, instead, compares a finite-resolution implementation with the continuum FNO and reveals that new aliasing errors can arise throughout the network.

In the following subsection, we combine these perspectives. We derive a more explicit bound between $\mathcal{G}_r$ and $\mathcal{G}_R$, while explicitly accounting for the mismatch generated at the input, after every nonlinear layer, and at the final projection.

D.3 Layer-wise Upper bound of Discretization Mismatch of FNO

The results in this section builds on and refine Prop. 5 in (Gao et al., 2025), that introduced the discretization mismatch error between two FNO discretizations and bounded its propagation through the network. We use the proof techniques introduced in (Lanthaler et al., 2024), which showed that, relative to the continuum FNO, aliasing acts as a new source of discretization error at every layer.

We introduce the following quantity.

Definition 2 (Aliasing discrepancy)**.** *For two resolutions $r < R$, the Aliasing discrepancy of f is*

$$\Delta_{r,R}(f) := \|(I_R - I_r)f\|_{L^2(\mathbb{T}^d)}\,. \tag{40}$$

Unlike a comparison with a continuum limit, such as the one in (Lanthaler et al., 2024), $\Delta_{r,R}(f)$ directly measures which components of f are represented differently by the two finite grids.

Lemma 1 (Spectral representation of the aliasing discrepancy)**.** *Let $R = qr$ with $q \in \mathbb{N}$, and let $f \in H^s(\mathbb{T}^d, \mathbb{R}^m)$ with $s > d/2$. Then*

$$\Delta_{r,R}(f)^2 = \sum_{k\in\Lambda_r} \|\mathcal{F}_R(f)(k) - \mathcal{F}_r(f)(k)\|^2 + \sum_{k\in\Lambda_R\setminus\Lambda_r} \|\mathcal{F}_R(f)(k)\|^2\,. \tag{41}$$

Moreover,

$$\Delta_{r,R}(f)^2 = \sum_{k\in\Lambda_r} \left\| \sum_{m\in\mathbb{Z}^d\setminus q\mathbb{Z}^d} \mathcal{F}(f)(k+rm) \right\|^2 + \sum_{k\in\Lambda_R\setminus\Lambda_r} \left\| \sum_{\ell\in\mathbb{Z}^d} \mathcal{F}(f)(k+R\ell) \right\|^2 .$$

Here $\|\cdot\|$ denotes the L^2 norm.

Proof. Since $\mathsf{s} > d/2$, the Fourier coefficients of f are absolutely summable. Indeed,

$$\sum_{k\in\mathbb{Z}^d} \|\mathcal{F}(f)(k)\| \le \left(\sum_{k\in\mathbb{Z}^d} (1+\|k\|^{2s}) \|\mathcal{F}(f)(k)\|^2 \right)^{1/2} \left(\sum_{k\in\mathbb{Z}^d} \frac{1}{1+\|k\|^{2s}} \right)^{1/2} < \infty.$$

As a consequence, we can rearrange its Fourier series.

The standard aliasing identity for the discrete Fourier transform gives, for every $k \in \Lambda_{\mathsf{s}}$,

$$\mathcal{F}_{\mathsf{s}}(f)(k) = \sum_{\ell\in\mathbb{Z}^d} \mathcal{F}(f)(k+s\ell). \tag{42}$$

Consequently,

$$I_R f - I_r f = \sum_{k\in\Lambda_r} \left(\mathcal{F}_R(f)(k) - \mathcal{F}_r(f)(k)\right) e^{2\pi i k\cdot x} + \sum_{k\in\Lambda_R\setminus\Lambda_r} \mathcal{F}_R(f)(k) e^{2\pi i k\cdot x}.$$

The Fourier modes appearing in the two sums are mutually orthogonal. Parseval's identity therefore yields

$$\|I_R f - I_r f\|_{L^2}^2 = \sum_{k\in\Lambda_r} \|\mathcal{F}_R(f)(k) - \mathcal{F}_r(f)(k)\|^2 + \sum_{k\in\Lambda_R\setminus\Lambda_r} \|\mathcal{F}_R(f)(k)\|^2,$$

which proves equation 41.

It remains to express the first term in terms of the Fourier coefficients of f. Since $R = qr$, equation 42 gives, for $k \in \Lambda_r$,

$$\mathcal{F}_r(f)(k) = \sum_{m\in\mathbb{Z}^d} \mathcal{F}(f)(k + rm),$$

whereas

$$\mathcal{F}_R(f)(k) = \sum_{\ell\in\mathbb{Z}^d} \mathcal{F}(f)(k + R\ell) = \sum_{m\in q\mathbb{Z}^d} \mathcal{F}(f)(k + rm).$$

Hence

$$\mathcal{F}_R(f)(k) - \mathcal{F}_r(f)(k) = -\sum_{m\in\mathbb{Z}^d\setminus q\mathbb{Z}^d} \mathcal{F}(f)(k + rm).$$

For $k \in \Lambda_R \setminus \Lambda_r$, we simply have

$$\mathcal{F}_R(f)(k) = \sum_{\ell\in\mathbb{Z}^d} \mathcal{F}(f)(k + R\ell).$$

Substituting these identities into equation 41 proves equation 42. □

We now propagate this pairwise discrepancy through the layers of an FNO. All Fourier multipliers use modes contained in the coarse Fourier set Λ_r, since the model was trained at resolution r. This guarantees that the same Fourier multiplier is well defined at both resolutions.

For each Fourier multiplier, define

$$\kappa_t := \|\mathcal{K}_t\|_{L^2\to L^2},$$

and let L_t and $L_{\mathcal{Q}}$ denote the Lipschitz constants of σ_t and $\mathcal{Q}$, respectively. Set

$$\lambda_t := L_t\big(\|W_t\|_2 + \kappa_t\big). \tag{43}$$

Proposition 7 (Layerwise discretization mismatch of FNOs)**.** *Let $R = qr$, $q \in \mathbb{N}$, and let $a \in H^s(\mathbb{T}^d, \mathbb{R}^{d_a})$ with $s > d/2$. Assume that the pointwise maps preserve H^s, that the activations are globally Lipschitz, and that all Fourier modes used by the FNO are resolved at resolution r.*

Then the discrepancy between the same FNO evaluated at resolutions r and R satisfies

$$\begin{aligned}\|\mathcal{G}_r(u_R) - \mathcal{G}_r(u_r)\|_{L^2} \le L_{\mathcal{Q}} &\left(\prod_{j=0}^{T-1} \lambda_j\right) \Delta_{r,R}\big(\mathcal{P}(u)\big) \\ &+ L_{\mathcal{Q}} \sum_{t=0}^{T-1} \left(\prod_{j=t+1}^{T-1} \lambda_j\right) \Delta_{r,R}\big(\sigma_t(z_t^r)\big) \\ &+ \Delta_{r,R}\big(\mathcal{Q}(v_T^r)\big).\end{aligned} \tag{44}$$

Thus, the discretization mismatch is controlled by three sources: the mismatch introduced by discretizing the lifted input, the aliasing newly generated by each nonlinear FNO layer and propagated through the subsequent layers, and the aliasing introduced by the final projection.

Proof. For $t = 0, \dots, T$, define

$$\delta_t := \big\|v_t^R - v_t^r\big\|_{L^2}.$$

We first control the initial discrepancy,

$$v_0^R - v_0^r = (I_R - I_r)\mathcal{P}(u),$$

and therefore

$$\delta_0 = \Delta_{r,R}\big(\mathcal{P}(u)\big). \tag{45}$$

Consider now one Fourier layer.

$$v_{t+1}^R - v_{t+1}^r = I_R\big[\sigma_t(z_t^R)\big] - I_r\big[\sigma_t(z_t^r)\big].$$

Adding and subtracting $I_R[\sigma_t(z_t^r)]$ yields

$$\begin{aligned} v_{t+1}^R - v_{t+1}^r = I_R\left[\sigma_t(z_t^R) - \sigma_t(z_t^r)\right] \\ + (I_R - I_r)\sigma_t(z_t^r). \end{aligned} \tag{46}$$

Taking the L^2 norm gives

$$\begin{aligned} \delta_{t+1} \le \left\| I_R\left[\sigma_t(z_t^R) - \sigma_t(z_t^r)\right]\right\|_{L^2} \\ + \Delta_{r,R}\big(\sigma_t(z_t^r)\big). \end{aligned} \tag{47}$$

We next bound the first term. For an arbitrary function h, the L^2 norm of its trigonometric interpolant on the R^d grid satisfies

$$\|I_R h\|_{L^2} = \left(\frac{1}{R^d}\sum_{x\in X_R} |h(x)|^2\right)^{1/2}.$$

Hence, using the L_t-Lipschitz continuity of σ_t,

$$\begin{aligned} &\left\| I_R\left[\sigma_t(z_t^R) - \sigma_t(z_t^r)\right]\right\|_{L^2} \\ &\quad \le L_t\left(\frac{1}{R^d}\sum_{x\in X_R} |z_t^R(x) - z_t^r(x)|^2\right)^{1/2}. \end{aligned} \tag{48}$$

The bias cancels in the difference of the preactivations, so that

$$z_t^R - z_t^r = W_t(v_t^R - v_t^r) + \mathcal{K}_t(v_t^R - v_t^r).$$

Since v_ℓ^R belongs to the trigonometric space associated with Λ_R and v_ℓ^r belongs to the subspace associated with $\Lambda_r \subseteq \Lambda_R$, the pointwise linear map W_ℓ preserves these spatial frequencies, while $\mathcal{K}_\ell$ outputs only retained modes contained in Λ_r. Hence $z_\ell^R - z_\ell^r$ is represented on the R grid. Discrete Parseval on the R^d grid gives

$$\left(\frac{1}{R^d}\sum_{x\in X_R} |z_t^R(x) - z_t^r(x)|^2\right)^{1/2} = \|z_t^R - z_t^r\|_{L^2}.$$

Consequently,

$$\begin{aligned} \|z_t^R - z_t^r\|_{L^2} &\le \|W_t(v_t^R - v_t^r)\|_{L^2} + \|\mathcal{K}_t(v_t^R - v_t^r)\|_{L^2} \\ &\le \big(\|W_t\|_2 + \kappa_t\big)\delta_t. \end{aligned} \tag{49}$$

Combining equation 47, equation 48, and equation 49, we obtain the layerwise recursion

$$\delta_{t+1} \le \lambda_t\delta_t + \Delta_{r,R}\big(\sigma_t(z_t^r)\big). \tag{50}$$

Iterating equation 50 gives

$$\delta_T \le \left(\prod_{j=0}^{T-1}\lambda_j\right)\delta_0 + \sum_{t=0}^{T-1}\left(\prod_{j=t+1}^{T-1}\lambda_j\right)\Delta_{r,R}\big(\sigma_t(z_t^r)\big). \tag{51}$$

Using equation 45,

$$\delta_T \le \left(\prod_{j=0}^{T-1}\lambda_j\right)\Delta_{r,R}\big(\mathcal{P}(u)\big) + \sum_{t=0}^{T-1}\left(\prod_{j=t+1}^{T-1}\lambda_j\right)\Delta_{r,R}\big(\sigma_t(z_t^r)\big). \tag{52}$$

It remains to control the projection layer,

$$\mathcal{G}_R(u) - \mathcal{G}_r(u) = I_R\big[\mathcal{Q}(v_T^R)\big] - I_r\big[\mathcal{Q}(v_T^r)\big].$$

Adding and subtracting $I_R[\mathcal{Q}(v_T^r)]$ gives

$$\mathcal{G}_R(u) - \mathcal{G}_r(u) = I_R\left[\mathcal{Q}(v_T^R) - \mathcal{Q}(v_T^r)\right] + (I_R - I_r)\mathcal{Q}(v_T^r). \tag{53}$$

And using the $L_{\mathcal{Q}}$-Lipschitz continuity of $\mathcal{Q}$,

$$\left\|I_R\left[\mathcal{Q}(v_T^R) - \mathcal{Q}(v_T^r)\right]\right\|_{L^2} \le L_{\mathcal{Q}}\delta_T. \tag{54}$$

Therefore,

$$\|\mathcal{G}_R(u) - \mathcal{G}_r(u)\|_{L^2} \le L_{\mathcal{Q}}\delta_T + \Delta_{r,R}\big(\mathcal{Q}(v_T^r)\big). \tag{55}$$

Substitution of equation 52 into equation 55 yields equation 44. □

E Compressed and Decomposed Inference

E.1 Compressed inference as a special case of decomposed inference

Let $\mathcal{G}_r : \mathcal{X}_r \to \mathcal{Y}_r$ and define

$$\mathcal{G}_r^{\times q}(z_1, \dots, z_q) := (\mathcal{G}_r(z_1), \dots, \mathcal{G}_r(z_q)). \tag{56}$$

Proposition 8 (Compressed inference is a special case of decomposed inference)**.** *Let $D : \mathcal{X}_R \to \mathcal{X}_r$ be a downsampling map and $U : \mathcal{Y}_r \to \mathcal{Y}_R$ an upsampling map. Assume $\mathcal{X}_r$ is nonempty and fix any $x_0 \in \mathcal{X}_r$. Then there exist maps*

$$\mathsf{D}_{\mathcal{X}} : \mathcal{X}_R \to \mathcal{X}_r^q, \qquad \mathsf{R}_{\mathcal{Y}} : \mathcal{Y}_r^q \to \mathcal{Y}_R \tag{57}$$

such that

$$\mathsf{R}_{\mathcal{Y}} \circ \mathcal{G}_r^{\times q} \circ \mathsf{D}_{\mathcal{X}} = U \circ \mathcal{G}_r \circ D. \tag{58}$$

Proof. Define

$$\mathsf{D}_{\mathcal{X}}(u) := (D(u), x_0, \dots, x_0), \tag{59}$$

and

$$\mathsf{R}_{\mathcal{Y}}(y_1, \dots, y_q) := U(y_1). \tag{60}$$

Then

$$\mathsf{R}_{\mathcal{Y}}\left(\mathcal{G}_r^{\times q}(\mathsf{D}_{\mathcal{X}}(u))\right) = \mathsf{R}_{\mathcal{Y}}\left(\mathcal{G}_r(D(u)), \mathcal{G}_r(x_0), \dots, \mathcal{G}_r(x_0)\right) \tag{61}$$

$$= U(\mathcal{G}_r(D(u))). \tag{62}$$

Thus the two predictors coincide for every $u \in \mathcal{X}_R$. □

The construction in the previous proof is interesting because makes clear the additional expressivity carried by decomposed inference. To be equivalent to a fixed compressed inference method it is sufficient to set all, except one, the decomposition and recomposition maps to the zero map. Differently, a general decomposed inference can choose these maps wisely to improve performance.

E.2 Irreducible error of compressed inference

Let U_R be a random input and set

$$Y_R^\dagger := \mathcal{G}_R^\dagger(U_R). \tag{63}$$

Assume $Y_R^\dagger \in L^2(\mathcal{Y}_R)$, where $\mathcal{Y}_R$ is a Hilbert space. Let $D : \mathcal{X}_R \to \mathcal{X}_r$ be measurable, and let $\mathcal{C} : \mathcal{X}_r \to \mathcal{Y}_R$ be measurable with $\mathcal{C}(D(U_R)) \in L^2(\mathcal{Y}_R)$. Define

$$\overline{\mathcal{G}}_D(D(U_R)) := \mathbb{E}[Y_R^\dagger \mid D(U_R)] \tag{64}$$

and

$$\mathcal{V}_D := \mathbb{E}\left[\|Y_R^\dagger - \overline{\mathcal{G}}_D(D(U_R))\|_R^2\right]. \tag{65}$$

Proposition 9 (Error decomposition for compressed inference)**.** *For every compressed predictor $\mathcal{C}$ satisfying the preceding square-integrability assumption,*

$$\mathbb{E}\left[\|Y_R^\dagger - \mathcal{C}(D(U_R))\|_R^2\right] = \mathcal{V}_D + \mathbb{E}\left[\|\overline{\mathcal{G}}_D(D(U_R)) - \mathcal{C}(D(U_R))\|_R^2\right]. \tag{66}$$

Proof. Set

$$A := Y_R^\dagger - \overline{\mathcal{G}}_D(D(U_R)), \qquad B := \overline{\mathcal{G}}_D(D(U_R)) - \mathcal{C}(D(U_R)). \tag{67}$$

Then $Y_R^\dagger - \mathcal{C}(D(U_R)) = A + B$. By definition of conditional expectation,

$$\mathbb{E}[A \mid D(U_R)] = 0, \tag{68}$$

and B is measurable with respect to $\sigma(D(U_R))$. Therefore

$$\mathbb{E}\langle A, B\rangle_R = \mathbb{E}\left[\mathbb{E}[\langle A, B\rangle_R \mid D(U_R)]\right] \tag{69}$$

$$= \mathbb{E}\left[\langle \mathbb{E}[A \mid D(U_R)], B\rangle_R\right] = 0. \tag{70}$$

The Pythagorean identity consequently gives

$$\mathbb{E}\|A + B\|_R^2 = \mathbb{E}\|A\|_R^2 + \mathbb{E}\|B\|_R^2, \tag{71}$$

and the first term equals $\mathcal{V}_D$ by definition. □

Corollary 1 (Irreducible error of compression)**.** *Over the class of measurable predictors $\mathcal{C}$ for which $\mathcal{C}(D(U_R))$ is square-integrable,*

$$\inf_{\mathcal{C}} \mathbb{E}\left[\|Y_R^\dagger - \mathcal{C}(D(U_R))\|_R^2\right] = \mathcal{V}_D. \tag{72}$$

Proof. The second term in Proposition 9 is nonnegative and vanishes for the measurable version $\mathcal{C} = \overline{\mathcal{G}}_D$. Hence the infimum is $\mathcal{V}_D$. □

In particular, $\mathcal{V}_D = 0$ if and only if $Y_R^\dagger = \mathbb{E}[Y_R^\dagger \mid D(U_R)]$ almost surely, i.e. the target contains no conditional-mean information beyond that retained by $D(U_R)$.

F DECOMPOSED INFERENCE: THEORETICAL PERSPECTIVE

F.1 EXACT ISOMETRIC DECOMPOSITIONS

Let

$$\mathsf{D}_{\mathcal{X}} : \mathcal{X}_R \to \mathcal{X}_r^q, \qquad \mathsf{D}_{\mathcal{Y}} : \mathcal{Y}_R \to \mathcal{Y}_r^q, \tag{73}$$

with components $\mathsf{D}_k^{\mathcal{X}}$ and $\mathsf{D}_k^{\mathcal{Y}}$.

Definition 3 (Exact isometric decomposition)**.** *The pair $(\mathsf{D}_{\mathcal{X}}, \mathsf{D}_{\mathcal{Y}})$ is an* exact isometric decomposition *if $\mathsf{D}_{\mathcal{X}}$ is a bimeasurable bijection and $\mathsf{D}_{\mathcal{Y}}$ is a linear isometric isomorphism when $\mathcal{Y}_r^q$ is equipped with*

$$\|(y_1, \dots, y_q)\|_{\oplus}^2 := \frac{1}{q} \sum_{k=1}^{q} \|y_k\|_r^2. \tag{74}$$

Set

$$\mathsf{R}_{\mathcal{X}} := \mathsf{D}_{\mathcal{X}}^{-1}, \qquad \mathsf{R}_{\mathcal{Y}} := \mathsf{D}_{\mathcal{Y}}^{-1}. \tag{75}$$

Thus

$$\|v - v'\|_R^2 = \frac{1}{q} \sum_{k=1}^{q} \|\mathsf{D}_k^{\mathcal{Y}} v - \mathsf{D}_k^{\mathcal{Y}} v'\|_r^2. \tag{76}$$

For $\mathcal{G}_r : \mathcal{X}_r \to \mathcal{Y}_r$, define

$$\mathcal{G}_r^{\mathsf{D}} := \mathsf{R}_{\mathcal{Y}} \circ \mathcal{G}_r^{\times q} \circ \mathsf{D}_{\mathcal{X}}. \tag{77}$$

F.2 RISK PRESERVATION UNDER MARGINAL INVARIANCE

Definition 4 (Component-wise marginal invariance)**.** *The decomposition satisfies component-wise marginal invariance with respect to μ_r if, for every $k = 1, \dots, q$,*

$$\left(\mathsf{D}_k^{\mathcal{X}}(U_R), \mathsf{D}_k^{\mathcal{Y}}(V_R)\right) \overset{d}{=} (U_r, V_r). \tag{78}$$

Proposition 10 (Risk preservation under marginal invariance)**.** *Let $(\mathsf{D}_{\mathcal{X}}, \mathsf{D}_{\mathcal{Y}})$ be an exact isometric decomposition satisfying component-wise marginal invariance. Then, for any square-integrable low-resolution predictor $\mathcal{G}_r$,*

$$\mathcal{R}_R(\mathcal{G}_r^{\mathsf{D}}) = \mathcal{R}_r(\mathcal{G}_r). \tag{79}$$

Proof. By construction,

$$\mathsf{D}_k^{\mathcal{Y}}(\mathcal{G}_r^{\mathsf{D}}(U_R)) = \mathcal{G}_r(\mathsf{D}_k^{\mathcal{X}}(U_R)). \tag{80}$$

The isometry of $\mathsf{D}_{\mathcal{Y}}$ therefore gives

$$\mathcal{R}_R(\mathcal{G}_r^{\mathsf{D}}) = \mathbb{E}\|\mathcal{G}_r^{\mathsf{D}}(U_R) - V_R\|_R^2 \tag{81}$$

$$= \frac{1}{q} \sum_{k=1}^{q} \mathbb{E}\left[\|\mathcal{G}_r(\mathsf{D}_k^{\mathcal{X}}(U_R)) - \mathsf{D}_k^{\mathcal{Y}}(V_R)\|_r^2\right]. \tag{82}$$

By component-wise marginal invariance, every summand equals

$$\mathbb{E}\|\mathcal{G}_r(U_r) - V_r\|_r^2 = \mathcal{R}_r(\mathcal{G}_r). \tag{83}$$

Averaging the q identical terms proves the claim. □

F.3 ERROR DECOMPOSITION OF DECOMPOSED INFERENCE

For each component define

$$\mathcal{B}_{\mathsf{D},k} := \mathbb{E}\left[\mathsf{D}_k^{\mathcal{Y}}(V_R) \,\middle|\, \mathsf{D}_k^{\mathcal{X}}(U_R)\right]. \tag{84}$$

The k-th decomposed component of the full high-resolution Bayes predictor is

$$\mathsf{D}_k^{\mathcal{Y}}(\mathcal{B}_R(U_R)) = \mathbb{E}\left[\mathsf{D}_k^{\mathcal{Y}}(V_R) \,\middle|\, U_R\right]. \tag{85}$$

Proposition 11 (Error decomposition for decomposed inference)**.** *Let* $(\mathsf{D}_{\mathcal{X}}, \mathsf{D}_{\mathcal{Y}})$ *be an exact isometric decomposition. Assume* $V_R \in L^2(\mathcal{Y}_R)$ *and*

$$\mathcal{G}_r(\mathsf{D}_k^{\mathcal{X}}(U_R)) \in L^2(\mathcal{Y}_r), \qquad k = 1, \dots, q. \tag{86}$$

Then

$$\mathcal{R}_R(\mathcal{G}_r^{\mathsf{D}}) - \mathcal{R}_R^B = \mathcal{E}^2_{\text{in}}(\mathsf{D}, \mathcal{G}_r) + \mathcal{E}^2_{\text{cross}}(\mathsf{D}), \tag{87}$$

where

$$\mathcal{E}^2_{\text{in}}(\mathsf{D}, \mathcal{G}_r) := \frac{1}{q} \sum_{k=1}^{q} \mathbb{E}\left[\|\mathcal{G}_r(\mathsf{D}_k^{\mathcal{X}}(U_R)) - \mathcal{B}_{\mathsf{D},k}\|_r^2\right] \tag{88}$$

and

$$\mathcal{E}^2_{\text{cross}}(\mathsf{D}) := \frac{1}{q} \sum_{k=1}^{q} \mathbb{E}\left[\|\mathcal{B}_{\mathsf{D},k} - \mathsf{D}_k^{\mathcal{Y}}(\mathcal{B}_R(U_R))\|_r^2\right]. \tag{89}$$

Proof. By the L^2-projection property of conditional expectation,, applied at resolution R,

$$\mathcal{R}_R(\mathcal{G}) - \mathcal{R}_R^B = \mathbb{E}\|\mathcal{G}(U_R) - \mathcal{B}_R(U_R)\|_R^2 \tag{90}$$

for every square-integrable predictor $\mathcal{G}$. Applying this to $\mathcal{G}_r^{\mathsf{D}}$ gives

$$\mathcal{R}_R(\mathcal{G}_r^{\mathsf{D}}) - \mathcal{R}_R^B = \mathbb{E}\|\mathcal{G}_r^{\mathsf{D}}(U_R) - \mathcal{B}_R(U_R)\|_R^2. \tag{91}$$

Because $\mathsf{D}_{\mathcal{Y}}$ is bounded and linear, it commutes with conditional expectation. Hence

$$\mathsf{D}_{\mathcal{Y}}(\mathcal{B}_R(U_R)) = \mathsf{D}_{\mathcal{Y}}\mathbb{E}[V_R \mid U_R] \tag{92}$$

$$= \mathbb{E}[\mathsf{D}_{\mathcal{Y}}(V_R) \mid U_R]. \tag{93}$$

In particular,

$$\mathsf{D}_k^{\mathcal{Y}}(\mathcal{B}_R(U_R)) = \mathbb{E}[\mathsf{D}_k^{\mathcal{Y}}(V_R) \mid U_R]. \tag{94}$$

By construction of decomposed inference,

$$\mathsf{D}_k^{\mathcal{Y}}(\mathcal{G}_r^{\mathsf{D}}(U_R)) = \mathcal{G}_r(\mathsf{D}_k^{\mathcal{X}}(U_R)). \tag{95}$$

Using the isometry in equation 76, equation equation 91 becomes

$$\mathcal{R}_R(\mathcal{G}_r^{\mathsf{D}}) - \mathcal{R}_R^B = \frac{1}{q} \sum_{k=1}^{q} \mathbb{E}\left[\left\|\mathcal{G}_r(\mathsf{D}_k^{\mathcal{X}}(U_R)) - \mathsf{D}_k^{\mathcal{Y}}(\mathcal{B}_R(U_R))\right\|_r^2\right]. \tag{96}$$

For each k, add and subtract $\mathcal{B}_{\mathsf{D},k}$. By the tower property and because $\sigma(\mathsf{D}_k^{\mathcal{X}}(U_R)) \subseteq \sigma(U_R)$,

$$\mathbb{E}\left[\mathsf{D}_k^{\mathcal{Y}}(\mathcal{B}_R(U_R)) \,\middle|\, \mathsf{D}_k^{\mathcal{X}}(U_R)\right] = \mathbb{E}\left[\mathbb{E}[\mathsf{D}_k^{\mathcal{Y}}(V_R) \mid U_R] \,\middle|\, \mathsf{D}_k^{\mathcal{X}}(U_R)\right] \tag{97}$$

$$= \mathbb{E}[\mathsf{D}_k^{\mathcal{Y}}(V_R) \mid \mathsf{D}_k^{\mathcal{X}}(U_R)] \tag{98}$$

$$= \mathcal{B}_{\mathsf{D},k}. \tag{99}$$

Therefore

$$\mathbb{E}\left[\mathsf{D}_k^{\mathcal{Y}}(\mathcal{B}_R(U_R)) - \mathcal{B}_{\mathsf{D},k} \,\middle|\, \mathsf{D}_k^{\mathcal{X}}(U_R)\right] = 0. \tag{100}$$

On the other hand,

$$\mathcal{G}_r(\mathsf{D}_k^{\mathcal{X}}(U_R)) - \mathcal{B}_{\mathsf{D},k} \tag{101}$$

is measurable with respect to $\sigma(\mathsf{D}_k^{\mathcal{X}}(U_R))$. The two differences are therefore orthogonal in $L^2(\mathcal{Y}_r)$, and

$$\mathbb{E}\left[\left\|\mathcal{G}_r(\mathsf{D}_k^{\mathcal{X}}(U_R)) - \mathsf{D}_k^{\mathcal{Y}}(\mathcal{B}_R(U_R))\right\|_r^2\right] \tag{102}$$

$$= \mathbb{E}\left[\|\mathcal{G}_r(\mathsf{D}_k^{\mathcal{X}}(U_R)) - \mathcal{B}_{\mathsf{D},k}\|_r^2\right] + \mathbb{E}\left[\|\mathcal{B}_{\mathsf{D},k} - \mathsf{D}_k^{\mathcal{Y}}(\mathcal{B}_R(U_R))\|_r^2\right]. \tag{103}$$

Averaging over k and using equation 96 proves the proposition. □

The cross-component term is independent of the learned predictor $\mathcal{G}_r$ but depends on the decomposition and on μ_R. It vanishes if and only if, for every component,

$$\mathbb{E}[\mathsf{D}_k^{\mathcal{Y}}(V_R) \mid U_R] = \mathbb{E}[\mathsf{D}_k^{\mathcal{Y}}(V_R) \mid \mathsf{D}_k^{\mathcal{X}}(U_R)] \quad \text{a.s.} \tag{104}$$

Thus it measures the conditional-mean information about the k-th output component that is available in the full high-resolution input but unavailable from the k-th input component alone.

F.4 Translation invariance and Polyphase Decomposition

Let $\tau_a^{\mathcal{X}}$ and $\tau_a^{\mathcal{Y}}$ denote translations by a spatial offset a on the high-resolution input and output grids.

Definition 5 (Translation invariance)**.** *The high-resolution distribution μ_R is translation invariant if, for every admissible grid translation a,*

$$(\tau_a^{\mathcal{X}} U_R, \tau_a^{\mathcal{Y}} V_R) \overset{d}{=} (U_R, V_R). \tag{105}$$

Assume

$$R = mr, \qquad \Lambda_m = \{0, \dots, m-1\}^d, \qquad q = m^d. \tag{106}$$

For $k \in \Lambda_m$, define

$$(\mathsf{P}_k u)(j) := u(mj + k), \tag{107}$$

and

$$\mathsf{P}(u) := (\mathsf{P}_k u)_{k \in \Lambda_m}. \tag{108}$$

The inverse rearrangement is

$$\mathsf{P}^{-1}((u_k)_{k \in \Lambda_m})(mj + k) = u_k(j). \tag{109}$$

Proposition 12 (Exact isometry of the polyphase rearrangement)**.** *Equip the high- and low-resolution grid spaces with the normalized discrete norms*

$$\|u\|_R^2 := \frac{1}{R^d} \sum_i |u_i|^2, \qquad \|z\|_r^2 := \frac{1}{r^d} \sum_j |z_j|^2. \tag{110}$$

Then P is a bijection and

$$\|u\|_R^2 = \frac{1}{q} \sum_{k \in \Lambda_m} \|\mathsf{P}_k u\|_r^2. \tag{111}$$

Hence the polyphase decomposition is exact and isometric with respect to the normalized product norm.

Proof. Bijectivity follows from the explicit inverse rearrangement. Moreover,

$$\frac{1}{q} \sum_{k \in \Lambda_m} \|\mathsf{P}_k u\|_r^2 = \frac{1}{q} \sum_{k \in \Lambda_m} \frac{1}{r^d} \sum_j |u(mj + k)|^2 \tag{112}$$

$$= \frac{1}{qr^d} \sum_i |u_i|^2 \tag{113}$$

$$= \frac{1}{R^d} \sum_i |u_i|^2 = \|u\|_R^2, \tag{114}$$

because $q = m^d$ and $R^d = m^d r^d = qr^d$. □

With $(\tau_a u)(i) := u(i + a)$,

$$\mathsf{P}_k = \mathsf{P}_0 \circ \tau_k. \tag{115}$$

Proposition 13 (Translation invariance of polyphase components)**.** *If μ_R is translation invariant, then for every $k \in \Lambda_m$,*

$$(\mathsf{P}_k^{\mathcal{X}} U_R, \mathsf{P}_k^{\mathcal{Y}} V_R) \overset{d}{=} (\mathsf{P}_0^{\mathcal{X}} U_R, \mathsf{P}_0^{\mathcal{Y}} V_R). \tag{116}$$

Proof. By equation 115,

$$(\mathsf{P}_k^{\mathcal{X}} U_R, \mathsf{P}_k^{\mathcal{Y}} V_R) = (\mathsf{P}_0^{\mathcal{X}} \tau_k^{\mathcal{X}} U_R, \mathsf{P}_0^{\mathcal{Y}} \tau_k^{\mathcal{Y}} V_R). \tag{117}$$

Translation invariance gives

$$(\tau_k^{\mathcal{X}} U_R, \tau_k^{\mathcal{Y}} V_R) \overset{d}{=} (U_R, V_R). \tag{118}$$

Applying the measurable map $(\mathsf{P}_0^{\mathcal{X}}, \mathsf{P}_0^{\mathcal{Y}})$ to both pairs proves the claim. □

Corollary 2 (Polyphase marginal invariance)**.** *Assume that μ_R is translation invariant and that*

$$(\mathsf{P}_0^{\mathcal{X}} U_R, \mathsf{P}_0^{\mathcal{Y}} V_R) \overset{d}{=} (U_r, V_r). \tag{119}$$

Then, for every $k \in \Lambda_m$,

$$(\mathsf{P}_k^{\mathcal{X}} U_R, \mathsf{P}_k^{\mathcal{Y}} V_R) \overset{d}{=} (U_r, V_r), \tag{120}$$

so polyphase decomposition satisfies component-wise marginal invariance with respect to μ_r.

Proof. Proposition 13 identifies the law of every phase with the law of the reference phase $k = 0$. The additional reference-phase assumption identifies that common law with μ_r. □

F.5 Approximate phase invariance under spatial regularity

Equip grid functions with the normalized norm

$$\|u\|_R^2 := \frac{1}{R^d} \sum_i |u_i|^2 \tag{121}$$

and define

$$\|u\|_{H_R^s}^2 := \sum_{\xi \in \mathcal{K}_R} (1 + |\xi|^2)^s |\mathcal{F}_R(u)(\xi)|^2. \tag{122}$$

Let $\alpha := \min\{s, 1\}$.

Lemma 2 (Sobolev control of translations)**.** *For $s > 0$ there exists $C_{s,d}$ such that*

$$\|\tau_a u - u\|_R \le C_{s,d} \left(\frac{|a|}{R}\right)^{\alpha} \|u\|_{H_R^s}. \tag{123}$$

Proof. The normalized discrete Fourier Transform satisfies

$$\mathcal{F}_R(\tau_a u)(\xi) = e^{2\pi i \xi \cdot a/R} \mathcal{F}_R(u)(\xi). \tag{124}$$

By Parseval and $|e^{it} - 1| \le C_\alpha |t|^\alpha$ for $0 < \alpha \le 1$,

$$\|\tau_a u - u\|_R^2 = \sum_{\xi \in \mathcal{K}_R} |e^{2\pi i \xi \cdot a/R} - 1|^2 |\mathcal{F}_R(u)(\xi)|^2 \tag{125}$$

$$\le C_{s,d} \left(\frac{|a|}{R}\right)^{2\alpha} \sum_{\xi \in \mathcal{K}_R} |\xi|^{2\alpha} |\mathcal{F}_R(u)(\xi)|^2 \tag{126}$$

$$\le C_{s,d} \left(\frac{|a|}{R}\right)^{2\alpha} \|u\|_{H_R^s}^2, \tag{127}$$

because $\alpha \le s$. □

Define

$$\nu_k := \mathrm{Law}(\mathsf{P}_k^{\mathcal{X}} U_R, \mathsf{P}_k^{\mathcal{Y}} V_R). \tag{128}$$

If

$$M_s^2 := \mathbb{E}\|U_R\|_{H_R^s}^2 + \mathbb{E}\|V_R\|_{H_R^s}^2 < \infty, \tag{129}$$

then the coupling induced by the same realization (U_R, V_R) and the preceding translation estimate yield

$$W_2(\nu_k, \nu_\ell) \le C_{s,d} \sqrt{q} \left(\frac{|k - \ell|}{R}\right)^{\alpha} M_s. \tag{130}$$

Consequently,

$$W_2(\nu_k, \mu_r) \le W_2(\nu_0, \mu_r) + C_{s,d} \sqrt{q} \left(\frac{|k|}{R}\right)^{\alpha} M_s. \tag{131}$$

Thus regularity controls the variation across phases, whereas $W_2(\nu_0, \mu_r)$ separately measures the reference-phase mismatch with pretraining.

G DATASETS

We report in this section a detailed description of the benchmarks adopted in this works.

Table 2 summarizes the datasets considered in our experiments. Each dataset is characterized by the number of trajectories and timesteps, the mesh type, the physical problem dimension, and the resolution.

Table 2: PDEBench datasets used in our experiments.

Dataset Name	**# Trajectories**	**# Timesteps**	**Mesh Type**	**Resolution**
Burgers	10000	101	Regular (1D periodic grid)	$(1024, 1)$
Darcy flow	10000	1	Regular (2D grid)	$(128 \times 128, 1)$
Reaction-diffusion	1000	101	Regular (2D grid)	$(128 \times 128, 2)$
Compressible Navier–Stokes	10000	21	Regular (2D periodic grid)	$(128 \times 128, 4)$

G.1 BURGERS' EQUATION (TAKAMOTO ET AL., 2022)

We consider the one-dimensional viscous Burgers' equation from PDEBench (Takamoto et al., 2022), a canonical nonlinear PDE combining advection and diffusion:

$$\partial_t u(t,x) + \partial_x \left(\frac{u(t,x)^2}{2} \right) = \frac{\nu}{\pi} \partial_{xx} u(t,x), \tag{132}$$

where $u(t,x)$ denotes the scalar state and $\nu > 0$ controls the strength of diffusion. The competition between the nonlinear transport term and diffusion produces increasingly sharp gradients as ν decreases, making the problem particularly sensitive to the spatial resolution. PDEBench considers periodic boundary conditions and random initial conditions constructed as superpositions of sinusoidal modes,

$$u_0(x) = \sum_{i=1}^{N} A_i \sin(k_i x + \phi_i), \tag{133}$$

with randomly sampled amplitudes, frequencies, and phases. Numerical trajectories are generated using a second-order upwind discretization for the advective term and a central-difference discretization for diffusion. Each PDEBench Burgers dataset contains $10\,000$ trajectories discretized over 1024 spatial cells and 201 stored temporal states, with separate datasets corresponding to different values of the diffusion coefficient ν. Following the experimental setup in (Sakarvadia et al., 2026), we subsample the temporal dimension by 2 and obtain 101 timestep per trajectory.

G.2 DARCY FLOW (TAKAMOTO ET AL., 2022)

The Darcy flow benchmark considers the steady-state pressure field of a fluid flowing through a heterogeneous porous medium. Given a spatially varying permeability coefficient $a(\mathbf{x})$, the solution $u(\mathbf{x})$ is determined on the unit square $\Omega = (0,1)^2$ by the elliptic equation

$$-\nabla \cdot (a(\mathbf{x}) \nabla u(\mathbf{x})) = f(\mathbf{x}), \qquad \mathbf{x} \in \Omega, \qquad u(\mathbf{x}) = 0, \quad \mathbf{x} \in \partial\Omega. \tag{134}$$

In PDEBench, the forcing is spatially constant, $f(\mathbf{x}) = \beta$, while the heterogeneous coefficient field $a(\mathbf{x})$ constitutes the input and the corresponding steady-state solution $u(\mathbf{x})$ the prediction target. Thus, in contrast to the other time-dependent benchmarks, the learning problem is directly an operator regression problem $a(\mathbf{x}) \mapsto u(\mathbf{x})$. PDEBench obtains the target solution by integrating the auxiliary parabolic equation

$$\partial_t u(\mathbf{x}, t) - \nabla \cdot (a(\mathbf{x}) \nabla u(\mathbf{x}, t)) = f(\mathbf{x}) \tag{135}$$

from a random initial field until convergence to the steady state. Each dataset contains $10\,000$ input–solution pairs represented on a 128×128 grid, making Darcy flow a useful test of resolution transfer in the presence of strongly heterogeneous spatial coefficients.

G.3 REACTION-DIFFUSION (TAKAMOTO ET AL., 2022)

We use the two-dimensional reaction–diffusion system from PDEBench (Takamoto et al., 2022), which models the coupled evolution of an activator $u(t, x, y)$ and an inhibitor $v(t, x, y)$. The dynamics follow a FitzHugh–Nagumo-type system,

$$\partial_t u = D_u \Delta u + R_u(u, v), \qquad R_u(u, v) = u - u^3 - k - v, \tag{136}$$

$$\partial_t v = D_v \Delta v + R_v(u, v), \qquad R_v(u, v) = u - v, \tag{137}$$

on $(x, y) \in (-1, 1)^2$ and $t \in (0, 5]$. PDEBench sets $D_u = 10^{-3}$, $D_v = 5 \times 10^{-3}$, and $k = 5 \times 10^{-3}$, and uses no-flow Neumann boundary conditions for both variables. The initial fields are sampled from standard Gaussian noise, producing spatially complex patterns through the nonlinear interaction between reaction and diffusion. Simulations are generated using a finite-volume spatial discretization and fourth-order Runge–Kutta time integration. The original trajectories are computed at resolution 512×512 with 501 temporal states, while PDEBench provides a 128×128, 101-step version used for model training. The dataset contains 1000 trajectories, each consisting of the two coupled fields (u, v), and is therefore a particularly relevant benchmark for cross-resolution transfer, since increasingly fine discretizations expose progressively finer spatial structure generated by the reaction–diffusion dynamics.

G.4 COMPRESSIBLE NAVIER–STOKES (TAKAMOTO ET AL., 2022)

We consider the two-dimensional compressible Navier–Stokes equations from PDEBench (Takamoto et al., 2022), which describe the evolution of a viscous compressible fluid through conservation of mass, momentum, and energy:

$$\partial_t \rho + \nabla \cdot (\rho \mathbf{v}) = 0, \tag{138}$$

$$\rho \left(\partial_t \mathbf{v} + \mathbf{v} \cdot \nabla \mathbf{v}\right) = -\nabla p + \eta \Delta \mathbf{v} + \left(\zeta + \frac{\eta}{3}\right) \nabla(\nabla \cdot \mathbf{v}), \tag{139}$$

$$\partial_t \left(\epsilon + \frac{\rho |\mathbf{v}|^2}{2}\right) + \nabla \cdot \left[\left(\epsilon + p + \frac{\rho |\mathbf{v}|^2}{2}\right) \mathbf{v} - \mathbf{v} \cdot \sigma'\right] = 0, \tag{140}$$

where ρ denotes the mass density, $\mathbf{v} = (v_x, v_y)$ the velocity, p the pressure, $\epsilon = p/(\Gamma - 1)$ the internal energy with $\Gamma = 5/3$, and η and ζ the shear and bulk viscosities, respectively. We use the PDEBench random-field configuration with periodic boundary conditions, Mach number $M = 0.1$, and viscosities $\eta = \zeta = 0.1$. The initial density, pressure, and velocity are generated from random combinations of sinusoidal modes, with the characteristic velocity determined by the prescribed Mach number. Numerical trajectories are computed using a second-order HLLC scheme with MUSCL reconstruction for the inviscid terms and central differences for the viscous terms. The dataset contains $10\,000$ trajectories on a 128×128 periodic grid with 21 stored temporal states and four predicted fields (ρ, v_x, v_y, p). The nonlinear coupling between advection, pressure, and compressibility produces multiscale flow structures, making this benchmark particularly suitable for evaluating resolution transfer in nonlinear time-dependent dynamics.

For the resolution-transfer experiments, we additionally generated high-resolution trajectories using the official PDEBench implementation rather than using a separate solver. The code was taken from the PDEBench repository, available at `foundation_models/pdebench_official/pdebench/data_gen/data_gen_NLE/CompressibleFluid/`, in particular the `CFD_multi_Hydra.py` generator and the configuration files `2D_CFD_Rand_M1_Eta001_Zeta001_{256,512,1024}.yaml`. Thus, the high-resolution data are generated with the same numerical solver and physical model as the PDEBench compressible-flow benchmark.

For these experiments, we use periodic boundary conditions, $\Gamma = 5/3$, $M_0 = 1$, and $\eta = \zeta = 10^{-2}$. The simulations evolve the solution from $t = 0$ to $t = 1$ and save states every $\Delta t_{\text{save}} = 0.05$, resulting in 21 temporal snapshots per trajectory. We generated 100 trajectories at each resolution, using independent random seeds from 2026 to 2125, on grids of size 256×256, 512×512, and 1024×1024. The generator stores density, the three velocity components, and pressure; for the two-dimensional learning task, we use the four fields (ρ, v_x, v_y, p).

The solver is implemented in JAX and uses a finite-volume formulation with four ghost-cell layers and periodic boundary conditions. The inviscid fluxes are computed using an HLLC Riemann solver with second-order reconstruction. The time integration employs a predictor-corrector update: an intermediate half-step is first computed, followed by a full-step update. The viscous contribution is then applied through directionally split finite-difference updates in the x, y, and z directions. The time step is selected adaptively from a CFL condition based on the local characteristic wave speeds, with $\mathrm{CFL} = 0.3$, and is additionally restricted by the viscous stability condition. Pressure is floored at 10^{-4} to maintain numerical stability.

The resulting files are stored as NumPy arrays with shape $(1, 21, N, N, 1)$, where $N \in \{256, 512, 1024\}$. This procedure gives resolution-matched trajectories with identical physical parameters and initial-condition generation across resolutions, enabling a controlled evaluation of zero-shot and polyphase resolution transfer.

H EXPERIMENTAL TABLES

In this section, we report the complete experimental results corresponding to the experiments presented in the main text.

H.1 ZERO-SHOT TRANSFER AT HIGHER RESOLUTION WITH FNO

Table 3: Burgers PDEBench zero-shot transfer results. Best and second-best transfer results within each training-resolution block are shown in bold and italics, respectively. Mean and standard deviation are computed across pretraining runs with three seeds.

Train res.	Test res.	Strategy	Parameters	Test MSE	Test L^2	Test rel. L^2
128	1024	zero-shot	0.3M	$6.56 \times 10^{-3} \pm 1.90 \times 10^{-3}$	$2.02 \times 10^{-2} \pm 3.45 \times 10^{-3}$	$1.47 \times 10^{-1} \pm 1.16 \times 10^{-2}$
128	1024	Low+pad (2026a)	0.3M	$\mathit{5.16 \times 10^{-4}} \pm 5.69 \times 10^{-6}$	$\mathit{6.44 \times 10^{-3}} \pm 5.42 \times 10^{-5}$	$\mathit{6.17 \times 10^{-2}} \pm 6.06 \times 10^{-4}$
128	1024	Polyphase inference	0.3M	$\mathbf{4.46 \times 10^{-4}} \pm 2.23 \times 10^{-5}$	$\mathbf{5.95 \times 10^{-3}} \pm 1.55 \times 10^{-4}$	$\mathbf{5.64 \times 10^{-2}} \pm 1.60 \times 10^{-3}$
128	128	Native train res.	0.3M	$5.20 \times 10^{-4} \pm 2.53 \times 10^{-5}$	$1.81 \times 10^{-2} \pm 4.74 \times 10^{-4}$	$6.04 \times 10^{-2} \pm 1.76 \times 10^{-3}$
256	1024	zero-shot	0.5M	$1.52 \times 10^{-2} \pm 5.68 \times 10^{-3}$	$2.67 \times 10^{-2} \pm 6.67 \times 10^{-3}$	$1.66 \times 10^{-1} \pm 3.37 \times 10^{-2}$
256	1024	Low+pad (2026a)	0.5M	$\mathit{3.70 \times 10^{-4}} \pm 2.58 \times 10^{-5}$	$\mathit{5.35 \times 10^{-3}} \pm 1.86 \times 10^{-4}$	$\mathit{4.93 \times 10^{-2}} \pm 1.09 \times 10^{-3}$
256	1024	Polyphase inference	0.5M	$\mathbf{3.65 \times 10^{-4}} \pm 2.96 \times 10^{-5}$	$\mathbf{5.31 \times 10^{-3}} \pm 2.10 \times 10^{-4}$	$\mathbf{4.92 \times 10^{-2}} \pm 1.37 \times 10^{-3}$
256	256	Native train res.	0.5M	$3.67 \times 10^{-4} \pm 2.65 \times 10^{-5}$	$1.06 \times 10^{-2} \pm 4.03 \times 10^{-4}$	$4.91 \times 10^{-2} \pm 1.39 \times 10^{-3}$
512	1024	zero-shot	1.1M	$1.09 \times 10^{-2} \pm 4.57 \times 10^{-3}$	$2.51 \times 10^{-2} \pm 6.28 \times 10^{-3}$	$1.78 \times 10^{-1} \pm 3.99 \times 10^{-2}$
512	1024	Low+pad (2026a)	1.1M	$\mathit{3.36 \times 10^{-4}} \pm 4.36 \times 10^{-5}$	$\mathit{5.05 \times 10^{-3}} \pm 3.43 \times 10^{-4}$	$\mathit{4.64 \times 10^{-2}} \pm 3.22 \times 10^{-3}$
512	1024	Polyphase inference	1.1M	$\mathbf{3.16 \times 10^{-4}} \pm 4.35 \times 10^{-5}$	$\mathbf{4.86 \times 10^{-3}} \pm 3.41 \times 10^{-4}$	$\mathbf{4.41 \times 10^{-2}} \pm 3.19 \times 10^{-3}$
512	512	Native train res.	1.1M	$3.15 \times 10^{-4} \pm 4.34 \times 10^{-5}$	$6.87 \times 10^{-3} \pm 4.81 \times 10^{-4}$	$4.40 \times 10^{-2} \pm 3.19 \times 10^{-3}$
1024	1024	Native train res.	2.1M	$3.61 \times 10^{-4} \pm 4.81 \times 10^{-5}$	$5.15 \times 10^{-3} \pm 3.82 \times 10^{-4}$	$4.68 \times 10^{-2} \pm 3.68 \times 10^{-3}$

Table 4: Darcy PDEBench zero-shot transfer results. Best and second-best transfer results within each training-resolution block are shown in bold and italics, respectively. Mean and standard deviation are computed across pretraining runs with three seeds.

Train res.	Test res.	Strategy	Parameters	Test MSE	Test L^2	Test rel. L^2
16	128	zero-shot	0.3M	$7.98 \times 10^{-4} \pm 9.76 \times 10^{-5}$	$2.24 \times 10^{-2} \pm 2.17 \times 10^{-3}$	$1.16 \times 10^{-1} \pm 1.24 \times 10^{-2}$
16	128	Low+pad (2026a)	0.3M	$1.35 \times 10^{-3} \pm 1.46 \times 10^{-5}$	$3.13 \times 10^{-2} \pm 3.97 \times 10^{-4}$	$1.72 \times 10^{-1} \pm 2.91 \times 10^{-3}$
16	128	Polyphase inference	0.3M	$5.48 \times 10^{-4} \pm 6.45 \times 10^{-6}$	$1.76 \times 10^{-2} \pm 1.77 \times 10^{-5}$	$9.19 \times 10^{-2} \pm 1.09 \times 10^{-4}$
16	16	Native train res.	0.3M	$3.77 \times 10^{-4} \pm 8.69 \times 10^{-6}$	$1.23 \times 10^{-2} \pm 1.35 \times 10^{-4}$	$5.44 \times 10^{-2} \pm 8.92 \times 10^{-4}$
32	128	zero-shot	1.2M	$6.44 \times 10^{-4} \pm 6.57 \times 10^{-5}$	$1.95 \times 10^{-2} \pm 1.56 \times 10^{-3}$	$1.02 \times 10^{-1} \pm 8.98 \times 10^{-3}$
32	128	Low+pad (2026a)	1.2M	$4.88 \times 10^{-4} \pm 5.92 \times 10^{-6}$	$1.52 \times 10^{-2} \pm 2.07 \times 10^{-4}$	$7.74 \times 10^{-2} \pm 1.32 \times 10^{-3}$
32	128	Polyphase inference	1.2M	$3.66 \times 10^{-4} \pm 7.20 \times 10^{-6}$	$1.13 \times 10^{-2} \pm 1.27 \times 10^{-4}$	$5.28 \times 10^{-2} \pm 5.20 \times 10^{-4}$
32	32	Native train res.	1.2M	$3.50 \times 10^{-4} \pm 9.56 \times 10^{-6}$	$9.49 \times 10^{-3} \pm 1.56 \times 10^{-4}$	$3.94 \times 10^{-2} \pm 8.16 \times 10^{-4}$
64	128	zero-shot	4.5M	$5.24 \times 10^{-4} \pm 1.05 \times 10^{-4}$	$1.72 \times 10^{-2} \pm 2.53 \times 10^{-3}$	$9.19 \times 10^{-2} \pm 1.41 \times 10^{-2}$
64	128	Low+pad (2026a)	4.5M	$3.44 \times 10^{-4} \pm 6.59 \times 10^{-6}$	$9.59 \times 10^{-3} \pm 1.25 \times 10^{-4}$	$4.20 \times 10^{-2} \pm 1.17 \times 10^{-3}$
64	128	Polyphase inference	4.5M	$3.58 \times 10^{-4} \pm 4.61 \times 10^{-6}$	$1.03 \times 10^{-2} \pm 3.82 \times 10^{-5}$	$4.61 \times 10^{-2} \pm 9.26 \times 10^{-5}$
64	64	Native train res.	4.5M	$3.33 \times 10^{-4} \pm 6.25 \times 10^{-6}$	$8.66 \times 10^{-3} \pm 4.79 \times 10^{-5}$	$3.46 \times 10^{-2} \pm 4.18 \times 10^{-4}$
128	128	Native train res.	17.3M	$3.53 \times 10^{-4} \pm 8.13 \times 10^{-6}$	$9.01 \times 10^{-3} \pm 9.90 \times 10^{-5}$	$3.76 \times 10^{-2} \pm 6.16 \times 10^{-4}$

Table 5: Reaction–diffusion PDEBench zero-shot transfer results. Best and second-best transfer results within each training-resolution block are shown in bold and italics, respectively. Mean and standard deviation are computed across pretraining runs with three seeds.

Train res.	Test res.	Strategy	Parameters	Test MSE	Test L^2	Test rel. L^2
16	128	zero-shot	0.3M	$1.51 \times 10^{-2} \pm 4.80 \times 10^{-3}$	$1.67 \times 10^{0} \pm 1.41 \times 10^{-1}$	$1.39 \times 10^{0} \pm 1.17 \times 10^{-1}$
16	128	Low+pad (2026a)	0.3M	$\mathit{2.24 \times 10^{-3}} \pm 1.79 \times 10^{-5}$	$\mathit{6.43 \times 10^{-1}} \pm 3.82 \times 10^{-3}$	$\mathit{5.34 \times 10^{-1}} \pm 3.15 \times 10^{-3}$
16	128	Polyphase inference	0.3M	$\mathbf{6.38 \times 10^{-4}} \pm 1.40 \times 10^{-5}$	$\mathbf{3.43 \times 10^{-1}} \pm 3.24 \times 10^{-3}$	$\mathbf{2.85 \times 10^{-1}} \pm 2.61 \times 10^{-3}$
16	16	Native train res.	0.3M	$9.28 \times 10^{-4} \pm 1.19 \times 10^{-5}$	$4.38 \times 10^{-1} \pm 2.79 \times 10^{-3}$	$3.43 \times 10^{-1} \pm 2.10 \times 10^{-3}$
32	128	zero-shot	1.2M	$4.18 \times 10^{-3} \pm 5.98 \times 10^{-4}$	$8.78 \times 10^{-1} \pm 6.61 \times 10^{-2}$	$7.30 \times 10^{-1} \pm 5.56 \times 10^{-2}$
32	128	Low+pad (2026a)	1.2M	$\mathit{7.33 \times 10^{-4}} \pm 2.11 \times 10^{-5}$	$\mathit{3.67 \times 10^{-1}} \pm 6.61 \times 10^{-3}$	$\mathit{3.05 \times 10^{-1}} \pm 5.47 \times 10^{-3}$
32	128	Polyphase inference	1.2M	$\mathbf{4.15 \times 10^{-4}} \pm 1.82 \times 10^{-5}$	$\mathbf{2.77 \times 10^{-1}} \pm 6.10 \times 10^{-3}$	$\mathbf{2.30 \times 10^{-1}} \pm 5.08 \times 10^{-3}$
32	32	Native train res.	1.2M	$4.32 \times 10^{-4} \pm 1.90 \times 10^{-5}$	$2.89 \times 10^{-1} \pm 6.36 \times 10^{-3}$	$2.34 \times 10^{-1} \pm 5.16 \times 10^{-3}$
64	128	zero-shot	4.5M	$6.14 \times 10^{-3} \pm 7.28 \times 10^{-4}$	$1.06 \times 10^{0} \pm 6.00 \times 10^{-2}$	$8.84 \times 10^{-1} \pm 4.99 \times 10^{-2}$
64	128	Low+pad (2026a)	4.5M	$\mathit{4.74 \times 10^{-4}} \pm 1.56 \times 10^{-5}$	$\mathit{2.96 \times 10^{-1}} \pm 5.21 \times 10^{-3}$	$\mathit{2.46 \times 10^{-1}} \pm 4.32 \times 10^{-3}$
64	128	Polyphase inference	4.5M	$\mathbf{3.78 \times 10^{-4}} \pm 8.89 \times 10^{-6}$	$\mathbf{2.64 \times 10^{-1}} \pm 3.12 \times 10^{-3}$	$\mathbf{2.19 \times 10^{-1}} \pm 2.58 \times 10^{-3}$
64	64	Native train res.	4.5M	$3.80 \times 10^{-4} \pm 9.67 \times 10^{-6}$	$2.67 \times 10^{-1} \pm 3.41 \times 10^{-3}$	$2.20 \times 10^{-1} \pm 2.80 \times 10^{-3}$
128	128	Native train res.	17.3M	$3.40 \times 10^{-4} \pm 3.41 \times 10^{-5}$	$2.50 \times 10^{-1} \pm 1.20 \times 10^{-2}$	$2.08 \times 10^{-1} \pm 9.92 \times 10^{-3}$

Table 6: Compressible Navier–Stokes PDEBench rollout results at 128×128. Mean and standard deviation are computed across pretraining runs with three seeds.

Train res.	Test res.	Strategy	Parameters	Test MSE	Test L^2	Test rel. L^2
16	128	zero-shot	0.3M	$2.34 \times 10^{1} \pm 1.35 \times 10^{1}$	$2.76 \times 10^{1} \pm 8.58 \times 10^{0}$	$5.19 \times 10^{-1} \pm 1.31 \times 10^{-1}$
16	128	Low+pad (2026a)	0.3M	$2.28 \times 10^{-1} \pm 9.35 \times 10^{-3}$	$2.54 \pm 5.08 \times 10^{-2}$	$6.14 \times 10^{-2} \pm 8.24 \times 10^{-4}$
16	128	Polyphase inference	0.3M	$2.30 \times 10^{-1} \pm 1.08 \times 10^{-2}$	$2.55 \pm 6.12 \times 10^{-2}$	$4.13 \times 10^{-2} \pm 1.03 \times 10^{-3}$
16	16	Native train res.	0.3M	$2.28 \times 10^{-1} \pm 1.37 \times 10^{-2}$	$2.69 \pm 8.61 \times 10^{-2}$	$4.17 \times 10^{-2} \pm 1.21 \times 10^{-3}$
32	128	zero-shot	1.2M	$3.55 \pm 2.04 \times 10^{0}$	$1.07 \times 10^{1} \pm 3.33 \times 10^{0}$	$2.33 \times 10^{-1} \pm 5.89 \times 10^{-2}$
32	128	Low+pad (2026a)	1.2M	$4.34 \times 10^{-1} \pm 9.72 \times 10^{-3}$	$5.32 \pm 4.98 \times 10^{-2}$	$6.11 \times 10^{-2} \pm 7.90 \times 10^{-4}$
32	128	Polyphase inference	1.2M	$2.31 \times 10^{-1} \pm 1.09 \times 10^{-2}$	$2.48 \pm 5.95 \times 10^{-2}$	$3.96 \times 10^{-2} \pm 9.90 \times 10^{-4}$
32	32	Native train res.	1.2M	$2.36 \times 10^{-1} \pm 1.42 \times 10^{-2}$	$2.54 \pm 8.13 \times 10^{-2}$	$4.06 \times 10^{-2} \pm 1.18 \times 10^{-3}$
64	128	zero-shot	4.5M	$1.94 \pm 1.12 \times 10^{0}$	$7.95 \pm 2.47 \times 10^{0}$	$2.25 \times 10^{-1} \pm 5.69 \times 10^{-2}$
64	128	Low+pad (2026a)	4.5M	$8.61 \times 10^{-1} \pm 1.78 \times 10^{-2}$	$6.08 \pm 6.08 \times 10^{-2}$	$6.04 \times 10^{-2} \pm 8.58 \times 10^{-4}$
64	128	Polyphase inference	4.5M	$4.06 \times 10^{-1} \pm 1.91 \times 10^{-2}$	$3.00 \pm 7.20 \times 10^{-2}$	$4.29 \times 10^{-2} \pm 1.07 \times 10^{-3}$
64	64	Native train res.	4.5M	$4.35 \times 10^{-1} \pm 2.61 \times 10^{-2}$	$3.07 \pm 9.82 \times 10^{-2}$	$4.29 \times 10^{-2} \pm 1.24 \times 10^{-3}$
128	128	Native train res.	17.3M	$3.40 \times 10^{-1} \pm 2.04 \times 10^{-2}$	$2.79 \pm 8.93 \times 10^{-2}$	$4.13 \times 10^{-2} \pm 1.20 \times 10^{-3}$

H.2 FINETUNING

Table 7: Burgers PDEBench few-shot finetuning results at target resolution 1024.

Pretrain res.	Fraction	Method	Trainable params	Test MSE	Test L^2	Test rel. L^2
128	0.10	LoRA	2,048	1.154×10^{-3}	9.443×10^{-3}	8.233×10^{-2}
128	0.10	Adapter	2,336	9.192×10^{-4}	8.384×10^{-3}	7.417×10^{-2}
128	0.10	F-Adapter	8,304	8.190×10^{-4}	7.952×10^{-3}	7.238×10^{-2}
128	0.10	PPFT	12	5.23×10^{-4}	6.43×10^{-3}	5.977×10^{-2}
128	0.10	LoRA + PPFT	2,060	5.23×10^{-4}	6.43×10^{-3}	6.010×10^{-2}
128	0.10	Adapter + PPFT	2,348	4.775×10^{-4}	6.187×10^{-3}	5.904×10^{-2}
128	0.10	F-Adapter + PPFT	8,316	$\mathbf{4.534 \times 10^{-4}}$	$\mathbf{6.047 \times 10^{-3}}$	$\mathbf{5.877 \times 10^{-2}}$
128	0.01	LoRA	2,048	4.017×10^{-3}	1.638×10^{-2}	1.226×10^{-1}
128	0.01	Adapter	2,336	2.868×10^{-3}	1.425×10^{-2}	1.149×10^{-1}
128	0.01	F-Adapter	8,304	2.929×10^{-3}	1.471×10^{-2}	1.193×10^{-1}
128	0.01	PPFT	12	4.509×10^{-4}	6.026×10^{-3}	5.744×10^{-2}
128	0.01	LoRA + PPFT	2,060	$\mathbf{4.450 \times 10^{-4}}$	$\mathbf{5.975 \times 10^{-3}}$	$\mathbf{5.668 \times 10^{-2}}$
128	0.01	Adapter + PPFT	2,348	4.555×10^{-4}	6.027×10^{-3}	5.724×10^{-2}
128	0.01	F-Adapter + PPFT	8,316	4.594×10^{-4}	6.046×10^{-3}	5.746×10^{-2}
256	0.10	LoRA	2,048	1.646×10^{-3}	1.047×10^{-2}	8.490×10^{-2}
256	0.10	Adapter	2,336	1.351×10^{-3}	9.956×10^{-3}	8.744×10^{-2}
256	0.10	F-Adapter	8,304	2.929×10^{-3}	1.471×10^{-2}	1.193×10^{-1}
256	0.10	PPFT	12	3.913×10^{-4}	5.530×10^{-3}	5.192×10^{-2}
256	0.10	LoRA + PPFT	2,060	3.979×10^{-4}	5.588×10^{-3}	5.269×10^{-2}
256	0.10	Adapter + PPFT	2,348	4.455×10^{-4}	5.881×10^{-3}	5.475×10^{-2}
256	0.10	F-Adapter + PPFT	8,316	$\mathbf{3.851 \times 10^{-4}}$	$\mathbf{5.514 \times 10^{-3}}$	$\mathbf{5.187 \times 10^{-2}}$
256	0.01	LoRA	2,048	9.446×10^{-3}	2.165×10^{-2}	1.391×10^{-1}
256	0.01	Adapter	2,336	8.112×10^{-3}	2.083×10^{-2}	1.446×10^{-1}
256	0.01	F-Adapter	8,304	7.049×10^{-3}	1.980×10^{-2}	1.462×10^{-1}
256	0.01	PPFT	12	3.681×10^{-4}	5.370×10^{-3}	4.989×10^{-2}
256	0.01	LoRA + PPFT	2,060	$\mathbf{3.638 \times 10^{-4}}$	$\mathbf{5.343 \times 10^{-3}}$	$\mathbf{4.974 \times 10^{-2}}$
256	0.01	Adapter + PPFT	2,348	3.754×10^{-4}	5.431×10^{-3}	5.077×10^{-2}
256	0.01	F-Adapter + PPFT	8,316	3.878×10^{-4}	5.524×10^{-3}	5.188×10^{-2}
512	0.10	LoRA	2,048	1.640×10^{-3}	1.081×10^{-2}	9.546×10^{-2}
512	0.10	Adapter	2,336	1.082×10^{-3}	9.085×10^{-3}	8.198×10^{-2}
512	0.10	F-Adapter	8,304	9.027×10^{-4}	8.318×10^{-3}	7.479×10^{-2}
512	0.10	PPFT	12	3.269×10^{-4}	5.003×10^{-3}	4.636×10^{-2}
512	0.10	LoRA + PPFT	2,060	3.133×10^{-4}	4.912×10^{-3}	4.576×10^{-2}
512	0.10	Adapter + PPFT	2,348	$\mathbf{3.131 \times 10^{-4}}$	$\mathbf{4.908 \times 10^{-3}}$	$\mathbf{4.573 \times 10^{-2}}$
512	0.10	F-Adapter + PPFT	8,316	3.202×10^{-4}	4.976×10^{-3}	4.648×10^{-2}
512	0.01	LoRA	2,048	7.059×10^{-3}	1.958×10^{-2}	1.440×10^{-1}
512	0.01	Adapter	2,336	4.780×10^{-3}	1.695×10^{-2}	1.318×10^{-1}
512	0.01	F-Adapter	8,304	3.553×10^{-3}	1.488×10^{-2}	1.190×10^{-1}
512	0.01	PPFT	12	3.223×10^{-4}	4.972×10^{-3}	4.605×10^{-2}
512	0.01	LoRA + PPFT	2,060	$\mathbf{3.178 \times 10^{-4}}$	$\mathbf{4.946 \times 10^{-3}}$	4.602×10^{-2}
512	0.01	Adapter + PPFT	2,348	3.205×10^{-4}	4.963×10^{-3}	4.618×10^{-2}
512	0.01	F-Adapter + PPFT	8,316	3.223×10^{-4}	4.980×10^{-3}	4.612×10^{-2}

Table 8: Reaction–diffusion few-shot finetuning results at target resolution 128.

Pretrain res.	Fraction	Method	Trainable params	Test MSE	Test L^2	Test rel. L^2
16	10%	LoRA	2,048	3.946×10^{-3}	8.526×10^{-1}	7.086×10^{-1}
16	10%	Adapter	2,336	2.717×10^{-3}	7.072×10^{-1}	5.869×10^{-1}
16	10%	F-Adapter	8,304	4.272×10^{-3}	8.877×10^{-1}	7.372×10^{-1}
16	10%	PPFT	52	$\mathbf{8.67 \times 10^{-4}}$	$\mathbf{4.00 \times 10^{-1}}$	$\mathbf{3.32 \times 10^{-1}}$
16	10%	LoRA + PPFT	2,100	$\mathbf{8.67 \times 10^{-4}}$	$\mathbf{4.00 \times 10^{-1}}$	$\mathbf{3.32 \times 10^{-1}}$
16	10%	Adapter + PPFT	2,388	$\mathbf{8.67 \times 10^{-4}}$	$\mathbf{4.00 \times 10^{-1}}$	$\mathbf{3.32 \times 10^{-1}}$
16	10%	F-Adapter + PPFT	8,356	$\mathbf{8.67 \times 10^{-4}}$	$\mathbf{4.00 \times 10^{-1}}$	$\mathbf{3.32 \times 10^{-1}}$
16	1%	LoRA	2,048	1.112×10^{-2}	1.433	1.191
16	1%	Adapter	2,336	6.502×10^{-3}	1.096	9.105×10^{-1}
16	1%	F-Adapter	8,304	8.721×10^{-3}	1.268	1.053
16	1%	PPFT	52	$\mathbf{8.67 \times 10^{-4}}$	$\mathbf{4.00 \times 10^{-1}}$	$\mathbf{3.32 \times 10^{-1}}$
16	1%	LoRA + PPFT	2,100	$\mathbf{8.67 \times 10^{-4}}$	$\mathbf{4.00 \times 10^{-1}}$	$\mathbf{3.32 \times 10^{-1}}$
16	1%	Adapter + PPFT	2,388	$\mathbf{8.67 \times 10^{-4}}$	$\mathbf{4.00 \times 10^{-1}}$	$\mathbf{3.32 \times 10^{-1}}$
16	1%	F-Adapter + PPFT	8,356	$\mathbf{8.67 \times 10^{-4}}$	$\mathbf{4.00 \times 10^{-1}}$	$\mathbf{3.32 \times 10^{-1}}$
32	10%	LoRA	2,048	2.364×10^{-3}	6.603×10^{-1}	5.487×10^{-1}
32	10%	Adapter	2,336	1.370×10^{-3}	5.024×10^{-1}	4.171×10^{-1}
32	10%	F-Adapter	8,304	1.347×10^{-3}	4.982×10^{-1}	4.136×10^{-1}
32	10%	PPFT	52	$\mathbf{4.34 \times 10^{-4}}$	$\mathbf{2.83 \times 10^{-1}}$	$\mathbf{2.35 \times 10^{-1}}$
32	10%	LoRA + PPFT	2,100	$\mathbf{4.34 \times 10^{-4}}$	$\mathbf{2.83 \times 10^{-1}}$	$\mathbf{2.35 \times 10^{-1}}$
32	10%	Adapter + PPFT	2,388	$\mathbf{4.34 \times 10^{-4}}$	$\mathbf{2.83 \times 10^{-1}}$	$\mathbf{2.35 \times 10^{-1}}$
32	10%	F-Adapter + PPFT	8,356	$\mathbf{4.34 \times 10^{-4}}$	$\mathbf{2.83 \times 10^{-1}}$	$\mathbf{2.35 \times 10^{-1}}$
32	1%	LoRA	2,048	4.044×10^{-3}	8.637×10^{-1}	7.182×10^{-1}
32	1%	Adapter	2,336	3.016×10^{-3}	7.459×10^{-1}	6.200×10^{-1}
32	1%	F-Adapter	8,304	2.632×10^{-3}	6.968×10^{-1}	5.785×10^{-1}
32	1%	PPFT	52	$\mathbf{4.34 \times 10^{-4}}$	$\mathbf{2.83 \times 10^{-1}}$	$\mathbf{2.35 \times 10^{-1}}$
32	1%	LoRA + PPFT	2,100	$\mathbf{4.34 \times 10^{-4}}$	$\mathbf{2.83 \times 10^{-1}}$	$\mathbf{2.35 \times 10^{-1}}$
32	1%	Adapter + PPFT	2,388	$\mathbf{4.34 \times 10^{-4}}$	$\mathbf{2.83 \times 10^{-1}}$	$\mathbf{2.35 \times 10^{-1}}$
32	1%	F-Adapter + PPFT	8,356	$\mathbf{4.34 \times 10^{-4}}$	$\mathbf{2.83 \times 10^{-1}}$	$\mathbf{2.35 \times 10^{-1}}$
64	10%	LoRA	2,048	1.133×10^{-3}	4.571×10^{-1}	3.798×10^{-1}
64	10%	Adapter	2,336	1.095×10^{-3}	4.496×10^{-1}	3.735×10^{-1}
64	10%	F-Adapter	8,304	1.114×10^{-3}	4.534×10^{-1}	3.767×10^{-1}
64	10%	PPFT	52	$\mathbf{3.82 \times 10^{-4}}$	$\mathbf{2.65 \times 10^{-1}}$	$\mathbf{2.21 \times 10^{-1}}$
64	10%	LoRA + PPFT	2,100	$\mathbf{3.82 \times 10^{-4}}$	$\mathbf{2.65 \times 10^{-1}}$	$\mathbf{2.21 \times 10^{-1}}$
64	10%	Adapter + PPFT	2,388	$\mathbf{3.82 \times 10^{-4}}$	$\mathbf{2.65 \times 10^{-1}}$	$\mathbf{2.21 \times 10^{-1}}$
64	10%	F-Adapter + PPFT	8,356	$\mathbf{3.82 \times 10^{-4}}$	$\mathbf{2.65 \times 10^{-1}}$	$\mathbf{2.21 \times 10^{-1}}$
64	1%	LoRA	2,048	4.000×10^{-3}	8.589×10^{-1}	7.126×10^{-1}
64	1%	Adapter	2,336	1.484×10^{-3}	5.235×10^{-1}	4.345×10^{-1}
64	1%	F-Adapter	8,304	1.701×10^{-3}	5.605×10^{-1}	4.653×10^{-1}
64	1%	PPFT	52	$\mathbf{3.82 \times 10^{-4}}$	$\mathbf{2.65 \times 10^{-1}}$	$\mathbf{2.21 \times 10^{-1}}$
64	1%	LoRA + PPFT	2,100	$\mathbf{3.82 \times 10^{-4}}$	$\mathbf{2.65 \times 10^{-1}}$	$\mathbf{2.21 \times 10^{-1}}$
64	1%	Adapter + PPFT	2,388	$\mathbf{3.82 \times 10^{-4}}$	$\mathbf{2.65 \times 10^{-1}}$	$\mathbf{2.21 \times 10^{-1}}$
64	1%	F-Adapter + PPFT	8,356	$\mathbf{3.82 \times 10^{-4}}$	$\mathbf{2.65 \times 10^{-1}}$	$\mathbf{2.21 \times 10^{-1}}$

Table 9: Darcy few-shot finetuning at target resolution 128. Best and second-best results within each pretraining-resolution and training-fraction block are shown in bold and italics.

Pretrain res.	Fraction	Method	Trainable params	Test MSE	Test L^2	Test rel. L^2
16	10%	LoRA	2,048	6.825×10^{-4}	1.980×10^{-2}	1.018×10^{-1}
16	10%	Adapter	2,336	7.138×10^{-4}	2.201×10^{-2}	1.156×10^{-1}
16	10%	F-Adapter	8,304	7.184×10^{-4}	2.262×10^{-2}	1.09×10^{-1}
16	10%	PPFT	452	4.310×10^{-4}	1.306×10^{-2}	6.466×10^{-2}
16	10%	LoRA + PPFT	2,500	3.540×10^{-4}	1.158×10^{-2}	5.640×10^{-2}
16	10%	Adapter + PPFT	2,788	*3.494×10^{-4}*	*1.140×10^{-2}*	*5.497×10^{-2}*
16	10%	F-Adapter + PPFT	8,756	**3.395×10^{-4}**	**1.107×10^{-2}**	**5.344×10^{-2}**
16	1%	LoRA	2,048	7.514×10^{-4}	2.184×10^{-2}	1.124×10^{-1}
16	1%	Adapter	2,336	8.288×10^{-4}	2.350×10^{-2}	1.15×10^{-1}
16	1%	F-Adapter	8,304	1.089×10^{-3}	2.765×10^{-2}	1.15×10^{-1}
16	1%	PPFT	452	4.700×10^{-4}	1.480×10^{-2}	7.494×10^{-2}
16	1%	LoRA + PPFT	2,500	3.836×10^{-4}	1.332×10^{-2}	6.686×10^{-2}
16	1%	Adapter + PPFT	2,788	*3.780×10^{-4}*	*1.324×10^{-2}*	*6.607×10^{-2}*
16	1%	F-Adapter + PPFT	8,756	**3.635×10^{-4}**	**1.270×10^{-2}**	**6.311×10^{-2}**
32	10%	LoRA	2,048	5.887×10^{-4}	1.812×10^{-2}	9.360×10^{-2}
32	10%	Adapter	2,336	6.018×10^{-4}	1.824×10^{-2}	9.596×10^{-2}
32	10%	F-Adapter	8,304	5.973×10^{-4}	1.832×10^{-2}	9.617×10^{-2}
32	10%	PPFT	452	**3.488×10^{-4}**	1.009×10^{-2}	4.668×10^{-2}
32	10%	LoRA + PPFT	2,500	*3.630×10^{-4}*	*9.600×10^{-3}*	4.222×10^{-2}
32	10%	Adapter + PPFT	2,788	3.730×10^{-4}	9.613×10^{-3}	*4.200×10^{-2}*
32	10%	F-Adapter + PPFT	8,756	3.759×10^{-4}	**9.568×10^{-3}**	**4.141×10^{-2}**
32	1%	LoRA	2,048	5.666×10^{-4}	1.835×10^{-2}	9.646×10^{-2}
32	1%	Adapter	2,336	5.767×10^{-4}	1.844×10^{-2}	9.699×10^{-2}
32	1%	F-Adapter	8,304	5.947×10^{-4}	1.897×10^{-2}	1.003×10^{-1}
32	1%	PPFT	452	3.767×10^{-4}	1.080×10^{-2}	4.987×10^{-2}
32	1%	LoRA + PPFT	2,500	3.625×10^{-4}	1.049×10^{-2}	4.859×10^{-2}
32	1%	Adapter + PPFT	2,788	*3.312×10^{-4}*	*1.010×10^{-2}*	*4.683×10^{-2}*
32	1%	F-Adapter + PPFT	8,756	**3.280×10^{-4}**	**1.001×10^{-2}**	**4.625×10^{-2}**
64	10%	LoRA	2,048	4.546×10^{-4}	1.442×10^{-2}	7.189×10^{-2}
64	10%	Adapter	2,336	4.438×10^{-4}	1.353×10^{-2}	6.633×10^{-2}
64	10%	F-Adapter	8,304	4.149×10^{-4}	1.289×10^{-2}	6.305×10^{-2}
64	10%	PPFT	452	**3.316×10^{-4}**	*8.746×10^{-3}*	3.627×10^{-2}
64	10%	LoRA + PPFT	2,500	3.538×10^{-4}	8.805×10^{-3}	3.575×10^{-2}
64	10%	Adapter + PPFT	2,788	3.525×10^{-4}	8.778×10^{-3}	*3.573×10^{-2}*
64	10%	F-Adapter + PPFT	8,756	*3.496×10^{-4}*	**8.735×10^{-3}**	**3.552×10^{-2}**
64	1%	LoRA	2,048	4.915×10^{-4}	1.633×10^{-2}	8.589×10^{-2}
64	1%	Adapter	2,336	4.239×10^{-4}	1.431×10^{-2}	7.216×10^{-2}
64	1%	F-Adapter	8,304	3.925×10^{-4}	1.361×10^{-2}	6.902×10^{-2}
64	1%	PPFT	452	3.306×10^{-4}	8.768×10^{-3}	**3.635×10^{-2}**
64	1%	LoRA + PPFT	2,500	3.275×10^{-4}	8.752×10^{-3}	*3.640×10^{-2}*
64	1%	Adapter + PPFT	2,788	*3.128×10^{-4}*	*8.751×10^{-3}*	3.714×10^{-2}
64	1%	F-Adapter + PPFT	8,756	**3.123×10^{-4}**	**8.663×10^{-3}**	3.650×10^{-2}

Table 10: Compressible Navier–Stokes few-shot finetuning at target resolution 128. Best and second-best results for each pretraining resolution and training fraction are shown in bold and italics.

Pretrain res.	Fraction	Method	Trainable params	Test n	Test MSE	Test L^2	Test rel. L^2
16	10%	LoRA	2,048	1000	1.065×10^{0}	6.025×10^{0}	1.251×10^{-1}
16	10%	Adapter	2,336	1000	5.283×10^{-1}	4.114×10^{0}	8.713×10^{-2}
16	10%	F-Adapter	8,304	1000	4.747×10^{-1}	3.872×10^{0}	7.409×10^{-2}
16	10%	PPFT	452	1000	$\mathbf{2.289 \times 10^{-1}}$	$\mathbf{2.548 \times 10^{0}}$	$\mathbf{4.124 \times 10^{-2}}$
16	10%	LoRA + PPFT	2,500	1000	$\textit{2.345} \times 10^{-1}$	$\textit{2.592} \times 10^{0}$	$\textit{4.190} \times 10^{-2}$
16	10%	Adapter + PPFT	2,788	1000	2.428×10^{-1}	2.648×10^{0}	4.476×10^{-2}
16	10%	F-Adapter + PPFT	8,756	1000	2.477×10^{-1}	2.670×10^{0}	4.366×10^{-2}
16	1%	LoRA	2,048	1000	1.791×10^{0}	7.833×10^{0}	1.666×10^{-1}
16	1%	Adapter	2,336	1000	1.296×10^{0}	6.836×10^{0}	1.795×10^{-1}
16	1%	F-Adapter	8,304	1000	1.431×10^{0}	7.240×10^{0}	1.821×10^{-1}
16	1%	PPFT	452	1000	$\mathbf{2.290 \times 10^{-1}}$	$\textit{2.549} \times 10^{0}$	$\mathbf{4.109 \times 10^{-2}}$
16	1%	LoRA + PPFT	2,500	1000	$\textit{2.290} \times 10^{-1}$	$\mathbf{2.548 \times 10^{0}}$	$\textit{4.149} \times 10^{-2}$
16	1%	Adapter + PPFT	2,788	1000	2.315×10^{-1}	2.568×10^{0}	4.192×10^{-2}
16	1%	F-Adapter + PPFT	8,756	1000	2.372×10^{-1}	2.608×10^{0}	4.319×10^{-2}
32	10%	LoRA	2,048	1000	6.219×10^{-1}	4.356×10^{0}	7.784×10^{-2}
32	10%	Adapter	2,336	1000	6.480×10^{-1}	4.530×10^{0}	9.325×10^{-2}
32	10%	F-Adapter	8,304	1000	5.738×10^{-1}	4.294×10^{0}	8.268×10^{-2}
32	10%	PPFT	452	1000	2.369×10^{-1}	2.488×10^{0}	3.951×10^{-2}
32	10%	LoRA + PPFT	2,500	1000	2.355×10^{-1}	$\mathbf{2.476 \times 10^{0}}$	$\mathbf{3.942 \times 10^{-2}}$
32	10%	Adapter + PPFT	2,788	1000	$\mathbf{2.350 \times 10^{-1}}$	$\textit{2.477} \times 10^{0}$	$\textit{3.947} \times 10^{-2}$
32	10%	F-Adapter + PPFT	8,756	1000	$\textit{2.355} \times 10^{-1}$	2.481×10^{0}	3.980×10^{-2}
32	1%	LoRA	2,048	1000	1.480×10^{0}	6.942×10^{0}	1.514×10^{-1}
32	1%	Adapter	2,336	1000	1.173×10^{0}	6.270×10^{0}	1.413×10^{-1}
32	1%	F-Adapter	8,304	1000	1.117×10^{0}	6.170×10^{0}	1.414×10^{-1}
32	1%	PPFT	452	1000	$\textit{2.365} \times 10^{-1}$	$\textit{2.483} \times 10^{0}$	$\textit{3.962} \times 10^{-2}$
32	1%	LoRA + PPFT	2,500	1000	$\mathbf{2.361 \times 10^{-1}}$	$\mathbf{2.480 \times 10^{0}}$	$\mathbf{3.945 \times 10^{-2}}$
32	1%	Adapter + PPFT	2,788	1000	2.369×10^{-1}	2.490×10^{0}	3.972×10^{-2}
32	1%	F-Adapter + PPFT	8,756	1000	2.403×10^{-1}	2.516×10^{0}	4.096×10^{-2}
64	10%	LoRA	2,048	1000	9.520×10^{-1}	5.258×10^{0}	9.944×10^{-2}
64	10%	Adapter	2,336	1000	6.150×10^{-1}	4.106×10^{0}	7.826×10^{-2}
64	10%	F-Adapter	8,304	1000	5.765×10^{-1}	3.899×10^{0}	7.032×10^{-2}
64	10%	PPFT	452	1000	4.329×10^{-1}	3.036×10^{0}	$\mathbf{4.291 \times 10^{-2}}$
64	10%	LoRA + PPFT	2,500	1000	4.335×10^{-1}	3.037×10^{0}	4.330×10^{-2}
64	10%	Adapter + PPFT	2,788	1000	$\textit{4.305} \times 10^{-1}$	$\textit{3.029} \times 10^{0}$	$\textit{4.291} \times 10^{-2}$
64	10%	F-Adapter + PPFT	8,756	1000	$\mathbf{4.231 \times 10^{-1}}$	$\mathbf{3.017 \times 10^{0}}$	4.305×10^{-2}
64	1%	LoRA	2,048	1000	1.295×10^{0}	6.513×10^{0}	1.479×10^{-1}
64	1%	Adapter	2,336	1000	9.768×10^{-1}	5.521×10^{0}	1.316×10^{-1}
64	1%	F-Adapter	8,304	1000	8.674×10^{-1}	4.968×10^{0}	1.030×10^{-1}
64	1%	PPFT	452	1000	4.332×10^{-1}	3.037×10^{0}	$\mathbf{4.304 \times 10^{-2}}$
64	1%	LoRA + PPFT	2,500	1000	4.328×10^{-1}	$\mathbf{3.036 \times 10^{0}}$	$\textit{4.309} \times 10^{-2}$
64	1%	Adapter + PPFT	2,788	1000	$\textit{4.319} \times 10^{-1}$	3.038×10^{0}	4.342×10^{-2}
64	1%	F-Adapter + PPFT	8,756	1000	$\mathbf{4.272 \times 10^{-1}}$	$\textit{3.037} \times 10^{0}$	4.390×10^{-2}

H.3 COMPARISON ACROSS MODEL ARCHITECTURES

Table 11: Zero-shot and polyphase Darcy results at 32×32 native training resolution and 128×128 transfer resolution. Mean and standard deviation are computed across pretraining runs with three seeds.

Model	Train res.	Test res.	Strategy	Parameters	Test MSE	Test L^2	Test rel. L^2
UNetClassic2D	32	32	Native train res.	1.3M	$3.33 \times 10^{-4} \pm 1.73 \times 10^{-5}$	$9.15 \times 10^{-3} \pm 1.90 \times 10^{-4}$	$3.65 \times 10^{-2} \pm 8.20 \times 10^{-4}$
UNetClassic2D	32	128	Zero-shot	1.3M	$1.60 \times 10^{-1} \pm 1.25 \times 10^{-2}$	$3.99 \times 10^{-1} \pm 2.11 \times 10^{-2}$	$2.57 \pm 2.18 \times 10^{-1}$
UNetClassic2D	32	128	Polyphase inference	1.3M	$3.39 \times 10^{-4} \pm 2.01 \times 10^{-5}$	$1.079 \times 10^{-2} \pm 1.21 \times 10^{-4}$	$4.940 \times 10^{-2} \pm 4.51 \times 10^{-4}$
Transolver	32	32	Native train res.	1.0M	$4.10 \times 10^{-4} \pm 2.13 \times 10^{-5}$	$1.38 \times 10^{-2} \pm 2.87 \times 10^{-4}$	$6.76 \times 10^{-2} \pm 1.52 \times 10^{-3}$
Transolver	32	128	Zero-shot	1.0M	$2.35 \times 10^{-3} \pm 1.28 \times 10^{-3}$	$4.27 \times 10^{-2} \pm 8.77 \times 10^{-3}$	$2.37 \times 10^{-1} \pm 4.14 \times 10^{-2}$
Transolver	32	128	Polyphase inference	1.0M	$3.28 \times 10^{-4} \pm 1.48 \times 10^{-5}$	$1.11 \times 10^{-2} \pm 1.38 \times 10^{-4}$	$5.25 \times 10^{-2} \pm 1.54 \times 10^{-3}$
LaMO	32	32	Native train res.	1.0M	$4.09 \times 10^{-4} \pm 2.12 \times 10^{-5}$	$1.36 \times 10^{-2} \pm 2.82 \times 10^{-4}$	$6.51 \times 10^{-2} \pm 1.46 \times 10^{-3}$
LaMO	32	128	Zero-shot	1.0M	$9.04 \times 10^{-3} \pm 6.36 \times 10^{-3}$	$9.06 \times 10^{-2} \pm 2.29 \times 10^{-2}$	$5.41 \times 10^{-1} \pm 1.23 \times 10^{-1}$
LaMO	32	128	Polyphase inference	1.0M	$3.08 \times 10^{-4} \pm 2.11 \times 10^{-6}$	$1.031 \times 10^{-2} \pm 7.80 \times 10^{-5}$	$4.808 \times 10^{-2} \pm 4.87 \times 10^{-4}$
FNO	32	32	Native train res.	1.2M	$3.50 \times 10^{-4} \pm 9.56 \times 10^{-6}$	$9.49 \times 10^{-3} \pm 1.56 \times 10^{-4}$	$3.94 \times 10^{-2} \pm 8.16 \times 10^{-4}$
FNO	32	128	Zero-shot	1.2M	$6.44 \times 10^{-4} \pm 6.57 \times 10^{-5}$	$1.95 \times 10^{-2} \pm 1.56 \times 10^{-3}$	$1.02 \times 10^{-1} \pm 8.98 \times 10^{-3}$
FNO	32	128	Polyphase inference	1.2M	$3.66 \times 10^{-4} \pm 7.20 \times 10^{-6}$	$1.13 \times 10^{-2} \pm 1.27 \times 10^{-4}$	$5.28 \times 10^{-2} \pm 5.20 \times 10^{-4}$

Table 12: Zero-shot and polyphase Burgers results. Models were trained at resolution 256; all transferred results use resolution 1024. Mean and standard deviation are computed across pretraining runs with three seeds.

Model	Train res.	Test res.	Strategy	Parameters	Test MSE	Test L^2	Test rel. L^2
UNetClassic1D	256	256	Native train res.	0.5M	$1.25 \times 10^{-1} \pm 9.03 \times 10^{-3}$	$9.79 \times 10^{-2} \pm 3.72 \times 10^{-3}$	$9.43 \times 10^{-1} \pm 2.67 \times 10^{-2}$
UNetClassic1D	256	1024	Zero-shot	0.5M	$1.25 \times 10^{-1} \pm 4.67 \times 10^{-2}$	$9.79 \times 10^{-2} \pm 2.45 \times 10^{-2}$	$9.43 \times 10^{-1} \pm 1.91 \times 10^{-1}$
UNetClassic1D	256	1024	Polyphase inference	0.5M	$8.24 \times 10^{-2} \pm 6.68 \times 10^{-3}$	$7.84 \times 10^{-2} \pm 3.10 \times 10^{-3}$	$7.49 \times 10^{-1} \pm 2.09 \times 10^{-2}$
Transolver1D	256	256	Native train res.	0.6M	$9.40 \times 10^{-2} \pm 6.79 \times 10^{-3}$	$1.16 \times 10^{-1} \pm 4.41 \times 10^{-3}$	$4.69 \times 10^{-1} \pm 1.33 \times 10^{-2}$
Transolver1D	256	1024	Zero-shot	0.6M	$8.35 \times 10^{-2} \pm 3.12 \times 10^{-2}$	$6.96 \times 10^{-2} \pm 1.74 \times 10^{-2}$	$5.37 \times 10^{-1} \pm 1.09 \times 10^{-1}$
Transolver1D	256	1024	Polyphase inference	0.6M	$6.67 \times 10^{-2} \pm 5.41 \times 10^{-3}$	$5.12 \times 10^{-2} \pm 2.02 \times 10^{-3}$	$4.16 \times 10^{-1} \pm 1.16 \times 10^{-2}$
FNO	256	256	Native train res.	0.5M	$3.79 \times 10^{-4} \pm 2.74 \times 10^{-5}$	$1.09 \times 10^{-2} \pm 4.14 \times 10^{-4}$	$5.00 \times 10^{-2} \pm 1.42 \times 10^{-3}$
FNO	256	1024	Zero-shot	0.5M	$1.52 \times 10^{-2} \pm 5.68 \times 10^{-3}$	$2.67 \times 10^{-2} \pm 6.67 \times 10^{-3}$	$1.66 \times 10^{-1} \pm 3.37 \times 10^{-2}$
FNO	256	1024	Polyphase inference	0.5M	$3.79 \times 10^{-4} \pm 3.07 \times 10^{-5}$	$5.42 \times 10^{-3} \pm 2.14 \times 10^{-4}$	$5.01 \times 10^{-2} \pm 1.40 \times 10^{-3}$
LaMO	256	1024	Zero-shot	0.6M	$2.98 \times 10^{-1} \pm 1.11 \times 10^{-1}$	$1.54 \times 10^{-1} \pm 3.85 \times 10^{-2}$	$1.42 \pm 2.88 \times 10^{-1}$
LaMO	256	1024	Polyphase inference	0.6M	$8.23 \times 10^{-2} \pm 6.67 \times 10^{-3}$	$7.27 \times 10^{-2} \pm 2.88 \times 10^{-3}$	$5.72 \times 10^{-1} \pm 1.59 \times 10^{-2}$

Table 13: Zero-shot and polyphase reaction–diffusion results. Models were trained at 32×32 and transferred to 128×128. Mean and standard deviation are computed across pretraining runs with three seeds.

Model	Train res.	Test res.	Strategy	Parameters	Test MSE	Test L^2	Test rel. L^2
UNetClassic2D	32	32	Native train res.	1.3M	$5.82 \times 10^{-5} \pm 2.56 \times 10^{-6}$	$1.06 \times 10^{-1} \pm 2.33 \times 10^{-3}$	$8.58 \times 10^{-2} \pm 1.89 \times 10^{-3}$
UNetClassic2D	32	128	Zero-shot	1.3M	$7.08 \times 10^{-3} \pm 1.01 \times 10^{-3}$	$1.14 \pm 8.58 \times 10^{-2}$	$9.50 \times 10^{-1} \pm 7.24 \times 10^{-2}$
UNetClassic2D	32	128	Polyphase inference	1.3M	$1.42 \times 10^{-4} \pm 6.23 \times 10^{-6}$	$1.61 \times 10^{-1} \pm 3.55 \times 10^{-3}$	$1.34 \times 10^{-1} \pm 2.96 \times 10^{-3}$
FNO	32	32	Native train res.	1.2M	$4.55 \times 10^{-4} \pm 2.00 \times 10^{-5}$	$2.96 \times 10^{-1} \pm 6.51 \times 10^{-3}$	$2.40 \times 10^{-1} \pm 5.29 \times 10^{-3}$
FNO	32	128	Zero-shot	1.2M	$4.18 \times 10^{-3} \pm 5.98 \times 10^{-4}$	$8.78 \times 10^{-1} \pm 6.61 \times 10^{-2}$	$7.30 \times 10^{-1} \pm 5.56 \times 10^{-2}$
FNO	32	128	Polyphase inference	1.2M	$4.34 \times 10^{-4} \pm 1.90 \times 10^{-5}$	$2.82 \times 10^{-1} \pm 6.21 \times 10^{-3}$	$2.35 \times 10^{-1} \pm 5.19 \times 10^{-3}$
Transolver	32	32	Native train res.	1.0M	$5.10 \times 10^{-4} \pm 2.30 \times 10^{-5}$	$3.05 \times 10^{-1} \pm 6.71 \times 10^{-3}$	$2.47 \times 10^{-1} \pm 5.43 \times 10^{-3}$
Transolver	32	128	Zero-shot	1.0M	$5.74 \times 10^{-3} \pm 8.21 \times 10^{-4}$	$1.03 \pm 7.75 \times 10^{-2}$	$8.56 \times 10^{-1} \pm 6.52 \times 10^{-2}$
Transolver	32	128	Polyphase inference	1.0M	$6.84 \times 10^{-3} \pm 3.00 \times 10^{-4}$	$1.12 \pm 2.47 \times 10^{-2}$	$2.69 \times 10^{-1} \pm 5.94 \times 10^{-3}$
LaMO	32	32	Native train res.	0.98M	$5.80 \times 10^{-4} \pm 2.61 \times 10^{-5}$	$3.10 \times 10^{-1} \pm 6.82 \times 10^{-3}$	$2.60 \times 10^{-1} \pm 5.72 \times 10^{-3}$
LaMO	32	128	Zero-shot	0.98M	$1.19 \times 10^{-2} \pm 1.70 \times 10^{-3}$	$1.48 \pm 1.11 \times 10^{-1}$	$8.12 \times 10^{-1} \pm 6.18 \times 10^{-2}$
LaMO	32	128	Polyphase inference	0.98M	$1.14 \times 10^{-2} \pm 5.00 \times 10^{-4}$	$1.45 \pm 3.19 \times 10^{-2}$	$2.74 \times 10^{-1} \pm 6.05 \times 10^{-3}$

Table 14: Zero-shot and polyphase compressible Navier–Stokes results at 128×128. Mean and standard deviation are computed across pretraining runs with three seeds.

Model	Train res.	Test res.	Strategy	Parameters	Test MSE	Test L^2	Test rel. L^2
UNetClassic2D	32	32	Native train res.	1.3M	$1.31 \pm 7.88 \times 10^{-2}$	$5.87 \pm 1.88 \times 10^{-1}$	$7.23 \times 10^{-2} \pm 2.10 \times 10^{-3}$
UNetClassic2D	32	128	Zero-shot	1.3M	$1.28 \times 10^{1} \pm 7.36 \times 10^{0}$	$2.01 \times 10^{1} \pm 6.26 \times 10^{0}$	$3.08 \times 10^{-1} \pm 7.79 \times 10^{-2}$
UNetClassic2D	32	128	Polyphase inference	1.3M	$1.40 \pm 6.61 \times 10^{-2}$	$5.82 \pm 1.40 \times 10^{-1}$	$7.23 \times 10^{-2} \pm 1.81 \times 10^{-3}$
Transolver	32	32	Native train res.	1.0M	$3.00 \times 10^{-1} \pm 1.80 \times 10^{-2}$	$2.70 \pm 8.10 \times 10^{-2}$	$4.50 \times 10^{-2} \pm 1.31 \times 10^{-3}$
Transolver	32	128	Zero-shot	1.0M	$2.78 \pm 1.60 \times 10^{0}$	$8.96 \pm 2.79 \times 10^{0}$	$1.09 \times 10^{-1} \pm 2.76 \times 10^{-2}$
Transolver	32	128	Polyphase inference	1.0M	$1.49 \pm 7.03 \times 10^{-2}$	$6.52 \pm 1.56 \times 10^{-1}$	$8.10 \times 10^{-2} \pm 2.02 \times 10^{-3}$
FNO	32	32	Native train res.	1.2M	$2.36 \times 10^{-1} \pm 1.42 \times 10^{-2}$	$2.54 \pm 8.13 \times 10^{-2}$	$4.06 \times 10^{-2} \pm 1.18 \times 10^{-3}$
FNO	32	128	Zero-shot	1.2M	$3.55 \pm 2.04 \times 10^{0}$	$1.07 \times 10^{1} \pm 3.33 \times 10^{0}$	$2.33 \times 10^{-1} \pm 5.89 \times 10^{-2}$
FNO	32	128	Polyphase inference	1.2M	$2.31 \times 10^{-1} \pm 1.09 \times 10^{-2}$	$2.48 \pm 5.95 \times 10^{-2}$	$3.96 \times 10^{-2} \pm 9.90 \times 10^{-4}$
LaMO	32	32	Native train res.	0.99M	$3.40 \times 10^{-1} \pm 2.04 \times 10^{-2}$	$2.85 \pm 8.55 \times 10^{-2}$	$5.00 \times 10^{-2} \pm 1.45 \times 10^{-3}$
LaMO	32	128	Zero-shot	0.99M	$2.27 \times 10^{1} \pm 1.30 \times 10^{1}$	$2.43 \times 10^{1} \pm 7.56 \times 10^{0}$	$4.96 \times 10^{-1} \pm 1.25 \times 10^{-1}$
LaMO	32	128	Polyphase inference	0.99M	$4.06 \pm 1.92 \times 10^{-1}$	$1.04 \times 10^{1} \pm 2.50 \times 10^{-1}$	$1.21 \times 10^{-1} \pm 3.02 \times 10^{-3}$

H.4 DPOT Experiments

In our experiments with DPOT, every polyphase component is processed by the model at the same resolution as the pretraining resolution and therefore uses the same learned positional embedding. In contrast, in direct zero-shot evaluation with high-resolution data, DPOT interpolates the positional embedding to the target resolution. Hence a larger positional-embedding tensor, which explains the slightly higher parameter-count of zero-shot compared to polyphase inference in Tables 15,16 and 17.

Table 15: DPOT evaluation on compressible Navier–Stokes trajectories at resolution 256^2. Mean and standard deviation are computed over the dataset.

Model	Method	Parameters	TF rel. L^2	Rollout rel. L^2	1-step time (s)	1-step memory (GiB)	Rollout time (s)	Rollout memory (GiB)
Tiny	Zero-shot	7.9M	0.05683 ± 0.02380	0.1660 ± 0.0795	0.0153 ± 0.0004	0.157	0.074 ± 0.002	0.18
Tiny	Polyphase (Parallel implementation)	7.5M	0.01245 ± 0.00831	0.01725 ± 0.01276	0.0068 ± 0.0002	0.167	0.066 ± 0.002	0.19
Small	Zero-shot	31.6M	0.03923 ± 0.01504	0.08673 ± 0.03856	0.0149 ± 0.0000	0.321	0.099 ± 0.000	0.35
Small	Polyphase (Parallel implementation)	30.8M	0.01091 ± 0.00708	0.01491 ± 0.00931	0.0101 ± 0.0000	0.330	0.100 ± 0.000	0.36
Medium	Zero-shot	122.9M	0.02619 ± 0.01117	0.04528 ± 0.02042	0.0200 ± 0.0000	0.641	0.170 ± 0.000	0.69
Medium	Polyphase (Parallel implementation)	122.1M	0.00812 ± 0.00552	0.01121 ± 0.00781	0.0164 ± 0.0000	0.649	0.174 ± 0.000	0.70
Large	Zero-shot	510.2M	0.02987 ± 0.00930	0.05247 ± 0.01624	0.0367 ± 0.0000	2.162	0.370 ± 0.000	2.25
Large	Polyphase (Parallel implementation)	509.0M	0.00756 ± 0.00514	0.01018 ± 0.00683	0.0327 ± 0.0001	2.162	0.356 ± 0.001	2.25
Huge	Zero-shot	1.033B	0.02132 ± 0.00813	0.04863 ± 0.02238	0.0588 ± 0.0001	4.214	0.627 ± 0.001	4.32
Huge	Polyphase (Parallel implementation)	1.031B	0.00572 ± 0.00404	0.00885 ± 0.00725	0.0560 ± 0.0001	4.223	0.617 ± 0.001	4.33

Table 16: DPOT evaluation on compressible Navier–Stokes trajectories at resolution 512^2. Mean and standard deviation are computed over the dataset.

Model	Method	Parameters	TF rel. L^2	Rollout rel. L^2	1-step time (s)	1-step memory (GiB)	Rollout time (s)	Rollout memory (GiB)
Tiny	Zero-shot	9.5M	0.1618 ± 0.0831	0.2846 ± 0.1490	0.0200 ± 0.0006	0.497	0.097 ± 0.003	0.57
Tiny	Polyphase (Parallel implementation)	7.5M	0.01374 ± 0.00891	0.01837 ± 0.01315	0.0097 ± 0.0003	0.528	0.095 ± 0.003	0.60
Small	Zero-shot	34.7M	0.3095 ± 0.4477	0.4609 ± 0.6383	0.0358 ± 0.0000	0.852	0.238 ± 0.000	0.93
Small	Polyphase (Parallel implementation)	30.8M	0.01235 ± 0.00769	0.01619 ± 0.00976	0.0188 ± 0.0184	0.880	0.187 ± 0.183	0.96
Medium	Zero-shot	126.1M	0.09457 ± 0.03883	0.1249 ± 0.0574	0.0529 ± 0.0000	1.179	0.451 ± 0.000	1.27
Medium	Polyphase (Parallel implementation)	122.1M	0.00970 ± 0.00631	0.01264 ± 0.00839	0.0343 ± 0.0000	1.206	0.363 ± 0.000	1.30
Large	Zero-shot	514.9M	1.0545 ± 0.4940	0.6090 ± 0.3683	0.1375 ± 0.0000	2.893	1.385 ± 0.000	3.01
Large	Polyphase (Parallel implementation)	509.0M	0.00921 ± 0.00599	0.01171 ± 0.00745	0.0985 ± 0.0002	2.912	1.071 ± 0.002	3.03
Huge	Zero-shot	1.039B	0.09773 ± 0.03636	0.1534 ± 0.1183	0.2185 ± 0.0002	5.141	2.329 ± 0.002	5.27
Huge	Polyphase (Parallel implementation)	1.031B	0.00752 ± 0.00498	0.01050 ± 0.00779	0.1789 ± 0.0003	5.150	1.973 ± 0.003	5.28

Table 17: DPOT evaluation on compressible Navier–Stokes trajectories at resolution 1024^2. Mean and standard deviation are computed over the dataset.

Model	Method	Parameters	TF rel. L^2	Rollout rel. L^2	1-step time (s)	1-step memory (GiB)	Rollout time (s)	Rollout memory (GiB)
Tiny	Zero-shot	15.8M	0.1989 ± 0.1064	0.36371 ± 0.1524	0.03776 ± 0.2539	2.735	0.339 ± 0.000	2.735
Tiny	Polyphase (Parallel implementation)	7.5M	0.01434 ± 0.00802	0.01924 ± 0.01324	0.02970 ± 0.0089	1.526	0.269 ± 0.000	2.68
Tiny	Polyphase (Sequential Implementation)	7.5M	0.01434 ± 0.00802	0.01924 ± 0.01324	0.36350 ± 0.0066	1.427	4.288 ± 0.071	1.713
Small	Zero-shot	47.3M	0.3201 ± 0.4997	0.48007 ± 0.6835	0.06476 ± 0.3469	5.280	0.851 ± 0.000	5.280
Small	Polyphase (Parallel implementation)	30.8M	0.01341 ± 0.00695	0.01758 ± 0.00897	0.04820 ± 0.0111	2.729	0.681 ± 0.000	3.85
Small	Polyphase (Sequential Implementation)	30.8M	0.01341 ± 0.00695	0.01758 ± 0.00897	0.52093 ± 0.0128	1.645	5.677 ± 0.062	1.886
Medium	Zero-shot	138.6M	0.09778 ± 0.04232	0.12418 ± 0.0600	0.10982 ± 0.2851	12.945	1.615 ± 0.001	12.945
Medium	Polyphase (Parallel implementation)	122.1M	0.01053 ± 0.00585	0.01428 ± 0.00816	0.07844 ± 0.0128	3.409	1.328 ± 0.019	4.54
Medium	Polyphase (Sequential Implementation)	122.1M	0.01053 ± 0.00585	0.01428 ± 0.00816	0.94184 ± 0.0181	2.325	10.629 ± 0.189	2.566
Large	Zero-shot	533.8M	0.5809 ± 0.2930	0.67304 ± 0.4446	0.27712 ± 0.2747	37.006	4.934 ± 0.009	37.006
Large	Polyphase (Parallel implementation)	509.0M	0.01022 ± 0.00555	0.01171 ± 0.00674	0.19744 ± 0.0347	7.346	4.131 ± 0.009	8.44
Large	Polyphase (Sequential Implementation)	509.0M	0.01022 ± 0.00555	0.01171 ± 0.00674	1.83683 ± 0.083	5.279	20.233 ± 0.349	5.453
Huge	Zero-shot	1.064B	0.09863 ± 0.03783	0.12189 ± 0.1207	0.42947 ± 0.1990	55.848	8.208 ± 0.024	55.848
Huge	Polyphase (Parallel implementation)	1.031B	0.008669 ± 0.005378	0.00979 ± 0.00807	0.33797 ± 0.0324	12.297	7.652 ± 0.050	13.36
Huge	Polyphase (Sequential Implementation)	1.031B	0.008669 ± 0.005378	0.00979 ± 0.00807	2.03344 ± 0.051	9.245	23.025 ± 0.358	9.417

I PyTorch implementations of Polyphase inference

The provide the sequential and the parallel implementation of Polyphase Inference. Note that the parallel implementation is mathematically equivalent to sequential component-wise evaluation provided that the pretrained operator acts independently on samples along the batch dimension, as is the case for the models considered here.

Listing 1: PyTorch implementation of parallel polyphase inference.

```
import torch
import torch.nn as nn

class ParallelPolyphaseInference(nn.Module):
    """
    Apply G_r independently to all q = m^2 polyphase
    components in a single batched forward pass.

    Input u_R: [B, T, R, R, C]
    Output v_R: [B, R, R, C]
    """

    def __init__(self, G_r, r=128):
        super().__init__()
        self.G_r = G_r
        self.r = r

    def forward(self, u_R):
        B, _, R, _, C = u_R.shape
        m = R // self.r
        v_R = u_R.new_empty(
            B, R, R, C
        )

        # P_k u_R for k in Lambda_m = {0, ..., m-1}^2.
        u_components = [
            u_R[:, :, k_y::m, k_x::m, :]
            for k_y in range(m)
            for k_x in range(m)
        ]

        # Evaluate G_r on all q components in parallel.
        u_batch = torch.cat(u_components, dim=0)
        v_batch = self.G_r(u_batch)

        k = 0
        for k_y in range(m):
            for k_x in range(m):
                start = k * B
                stop = (k + 1) * B

                v_R[:, k_y::m, k_x::m, :] = (
                    v_batch[start:stop]
                )
                k += 1

        return v_R
```

Listing 2: PyTorch implementation of sequential polyphase inference.

```
class SequentialPolyphaseInference(nn.Module):
    """
    Apply G_r to each polyphase component sequentially and
    recompose the predictions at resolution R.

    Input u_R: [B, T, R, R, C]
```

```
    Output v_R: [B, R, R, C]
    """

    def __init__(self, G_r, r=128):
        super().__init__()
        self.G_r = G_r
        self.r = r

    def forward(self, u_R):
        B, _, R, _, C = u_R.shape
        m = R // self.r
        v_R = u_R.new_empty(
            B, R, R, C
        )

        # Evaluate G_r(P_k u_R) and apply P^{-1} componentwise.
        for k_y in range(m):
            for k_x in range(m):
                u_k = u_R[
                    :, :, k_y::m, k_x::m, :
                ]
                v_k = self.G_r(u_k)

                v_R[:, k_y::m, k_x::m, :] = v_k

        return v_R
```

```
# Parallel implementation used for high-throughput inference.
parallel_model = ParallelPolyphaseInference(
    G_r=pretrained_operator,
    r=128,
)

# Sequential implementation used to reduce peak activation memory.
sequential_model = SequentialPolyphaseInference(
    G_r=pretrained_operator,
    r=128,
)

# u_R has shape [B, T, R, R, C].
with torch.inference_mode():
    v_R_parallel = parallel_model(u_R)
    v_R_sequential = sequential_model(u_R)
```

J EXPERIMENTAL DETAILS

We detail here the experimental setting followed in this work. We followed the setup of (Sakarvadia et al., 2026) and used their code, that is available at `https://github.com/msakarvadia/operator_aliasing`. We detail the hyperparameters of the pretrainings and the finetunings in the following tables.

K TRAINING AND EVALUATION HYPERPARAMETERS

We report the data, model, and optimization settings for the pretraining and few-shot adaptation experiments. Separate models are trained at each listed resolution. The best pretrained checkpoint minimizes relative L^2 at its native resolution. Unless stated otherwise, pretraining uses AdamW, normalized MSE, cosine annealing over 150 epochs.

For Darcy Flow and Compressible Navier–Stokes, we found it beneficial to apply a smoothing to the predictions after Polyphase recomposition, implemented as a non-learnable convolution with a 5×5 uniform averaging kernel. No smoothing was applied for Burgers or Reaction–Diffusion.

For FNO, we directly import and use the implementation from the Neural Operator library(Kossaifi et al., 2026). For Transolver and LaMO (Tiwari et al., 2025), we use the code provided by the authors and for UNet we adopt the modern implementation provided in (Ohana et al., 2024)

Note that our polyphase decomposition does not require any modification of the coordinate or positional encoding used by the pretrained neural operator. Each polyphase component is treated as an input on the original r-resolution grid and is processed independently by exactly the same pretrained operator $\mathcal{G}_r$. So, if $\mathcal{G}_r$ internally constructs or uses a positional encoding, the same positional encoding is applied to every polyphase component, exactly as for a standard r-resolution input. The polyphase decomposition and recomposition therefore operate only on the input and output fields and are agnostic to the particular positional encoding, if any, used within $\mathcal{G}_r$.

Table 18: Optimization settings shared by the FNO, UNetClassic, Transolver, and LaMO pretraining runs. Batch size, weight decay, minimum learning rate, and teacher forcing vary by run and are listed in the model tables.

Hyperparameter	**Value**
Optimizer	AdamW
Initial learning rate	10^{-3}
Learning-rate scheduler	Cosine annealing
Cosine schedule $T_{\max}$	150 epochs
Configured training epochs	150
Training loss	MSE on normalized fields
Input and target normalization	Channel-wise, from training statistics
Checkpoint selection	Minimum native-grid test relative L^2

Table 19: FNO pretraining settings. Here R denotes spatial points per axis.

Hyperparameter	**Burgers FNO1D**	**Darcy FNO2D**	**Reaction–diffusion FNO2D**	**CNS FNO2D**
Training resolutions	128, 256, 512, 1,024	16, 32, 64, 128	16, 32, 64, 128	16, 32, 64, 128
Hidden width	32	32	32	32
Latent size	32	32	32	32
Activation	GELU	GELU	GELU	GELU
Fourier modes	$R/2$	$R/2$ per axis	$R/2$ per axis	$R/2$ per axis
Batch size	64	128	128	64
Initial learning rate	10^{-3}	10^{-3}	10^{-3}	10^{-3}
Weight decay	10^{-5}	10^{-5}	10^{-4}	10^{-5}
Minimum learning rate	10^{-4}	10^{-4}	5×10^{-5}	10^{-4}
Cosine schedule $T_{\max}$	150	150	150	150
Configured epochs	150	150	150	150
Teacher forcing during training	No	Not applicable	Yes	No
Input and target normalization	Channel-wise	Channel-wise	Channel-wise	Channel-wise

Table 20: UNetClassic pretraining.

Setting	**Burgers**	**Darcy**	**Reaction–diffusion**	**CNS**
Architecture	UNetClassic1D	UNetClassic2D	UNetClassic2D	UNetClassic2D
Training resolution	256	32	32	32
Initial features	14	13	13	13
Batch size	64	64	128	64
Weight decay	10^{-4}	10^{-5}	10^{-4}	10^{-5}
Minimum learning rate	10^{-5}	10^{-4}	5×10^{-5}	10^{-4}
Teacher forcing during training	No	Not applicable	Yes	No

Table 21: Transolver pretraining.

Setting	**Burgers**	**Darcy**	**Reaction–diffusion**	**CNS**
Architecture	Transolver1D	Transolver2D	Transolver2D	Transolver2D
Training resolution	256	32	32	32
Hidden width	96	96	96	96
Layers	4	4	4	4
Attention heads	8	8	8	8
Slices	64	64	64	64
MLP ratio	4	4	4	4
Dropout	0	0	0	0
Activation	GELU	GELU	GELU	GELU
Batch size	64	64	128	64
Weight decay	10^{-4}	10^{-5}	10^{-4}	10^{-5}
Minimum learning rate	10^{-5}	10^{-4}	5×10^{-5}	10^{-4}
Teacher forcing during training	Yes	Not applicable	Yes	No

Table 22: LaMO pretraining.

Setting	Burgers	Darcy	Reaction–diffusion	CNS
Architecture	LaMO1D	LaMO2D	LaMO2D	LaMO2D
Training resolution	256	32	32	32
Hidden width	96	96	96	96
Layers	4	4	4	4
Patch size	2	2	2	2
MLP ratio	4	4	4	4
Dropout	0	0	0	0
Batch size	64	64	128	64
Weight decay	10^{-4}	10^{-4}	10^{-4}	10^{-5}
Minimum learning rate	10^{-5}	5×10^{-5}	5×10^{-5}	10^{-4}
Teacher forcing during training	Yes	Not applicable	Yes	No

L FEW-SHOT FINETUNING AND TESTING

The following settings describe the FNO adapter runs in the results tables. The pretrained FNO weights are frozen for LoRA, Adapter, F-Adapter and PPFT; only the added parameters are optimized. A method combined with PPFT trains its own added parameters together with the phase mixers. All runs use AdamW, MSE for training, 100 configured epochs, cosine annealing. Evaluation uses the checkpoint's pretraining normalization statistics.

Table 23: FNO few-shot finetuning dataset and optimizer settings.

Hyperparameter	Burgers	Darcy	Reaction–diffusion	CNS
Pretrained FNO resolutions	128, 256, 512	16, 32, 64	16, 32, 64	16, 32, 64
Finetuning resolution	1,024	128×128	128×128	128×128
Training fraction	1% or 10%	1% or 10%	1% or 10%	1% or 10%
Training batch size	128	128	64	64
Test batch size	128	64	64	32
Initial learning rate	10^{-3}	10^{-3}	10^{-3}	10^{-3}
Weight decay	10^{-6}	10^{-6}	10^{-6}	10^{-6}
Minimum learning rate	10^{-4}	10^{-4}	10^{-4}	10^{-4}
Cosine schedule $T_{\max}$	100	100	100	100
Adapter and LoRA rank	4	4	4	4
F-Adapter frequency bands	4	4	4	4
Adapter activation	GELU	GELU	GELU	GELU
Adapter initialization	Zero	Zero	Zero	Zero
Teacher forcing during training	No	Not applicable	Yes	No
Test mode	Autoregressive	One-step	Autoregressive	Autoregressive

Additionally the convolution before Polyphase decomposition and the convolution after Polyphase recomposition have the following hyperparameters,

Table 24: Input and output convolution kernel sizes and number of kernels for each dataset.

Dataset	Number of kernels	Input kernel size	Output kernel size
Burgers	1	5	5
Reaction–diffusion	2	5×5	5×5
Darcy	1	15×15	15×15
Compressible Navier–Stokes	4	15×15	15×15

The input convolution is initialized as the identity while the output convolution is initialized as described for Polyphase inference in the previous section.