FUNCTION-SPACE TRANSFORMER WITH ADAPTIVE ANCHORS

Anonymous authors
Paper under double-blind review

ABSTRACT

Many forms of data, including physical fields, geometric shapes, and visual signals, are naturally described by functions over continuous domains but are observed through discrete samples. Representing these functions on fixed uniform grids imposes a trade-off between resolving localized variation and increasing computation across the domain. Neural operators address this mismatch by learning mappings between functions, while latent-attention architectures provide flexible processing of sampled observations. We introduce the Function-Space Transformer (FST), a framework for learning from functions through a spatially adaptive continuous latent representation. FST stores features at anchors whose locations are predicted from the input observations and recursively refines these anchor features through function-space interactions. This allows the representation to adapt its spatial organization to each input rather than inherit that of the observation grid, while supporting both spatially resolved and finite-dimensional outputs. On PDE solution prediction using PDEBench Burgers and Darcy flow, FST substantially outperforms the Perceiver IO baseline, whose latent representation lacks explicit spatial organization, and is highly competitive with the Fourier Neural Operator. On ImageNet-1K, FST achieves higher classification accuracy than the Vision Transformer baseline, with fewer parameters across these comparisons. Ablations further support the benefits of function-space updates and recursive refinement. Together, these results highlight the potential of adaptive continuous representations for both scientific prediction and visual recognition.

1 INTRODUCTION

Applying transformers to functions over continuous domains requires representing the input through a finite collection of tokens. In common grid- and patch-based formulations, the spatial organization of these tokens follows the input discretization (Dosovitskiy et al., 2021). This couples representational resolution to computational cost: finer tokenizations can preserve smaller-scale variation, but increasing the number of tokens raises the quadratic cost of dense self-attention (Jaegle et al., 2021). Coarser tokenizations reduce this cost but may obscure localized structure needed for prediction. This trade-off is especially restrictive when spatial variation is nonuniform. Resolving a narrow transition or fine boundary through uniform refinement increases computation throughout the domain, including regions that require little additional detail.

The discretization used to observe a function, however, need not determine how a model represents it internally. Even when the input is sampled on a fixed grid, latent features need not follow the same layout or allocate equal capacity to equally sized regions. Instead, their spatial organization can adapt to the structure of each input. The challenge is therefore not simply to process a fixed set of tokens more efficiently, but to learn how to distribute a finite representation across a continuous domain. This motivates a continuous latent field whose spatial allocation is learned from the observations rather than prescribed by their sampling grid.

Neural operators approach learning from discretely observed functions by modeling mappings between function spaces. The Fourier Neural Operator (FNO; Li et al., 2021), for example, models global interactions through spectral transformations, while attention can also be formulated over continuous domains (Calvello et al., 2025). Latent-attention architectures such as Perceiver (Jaegle et al., 2021) and Perceiver IO (Jaegle et al., 2022) offer a complementary approach: they encode

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.

sampled observations into a smaller latent array, reducing the cost of processing large inputs and supporting task-dependent outputs. However, their latent arrays lack an explicit spatial organization that defines a continuous field. Spatial representations such as Equivariant Neural Fields (ENFs; Wessels et al., 2025) associate latent features with locations and use Gaussian windows to define a field between them. These perspectives motivate combining compact latent processing with a continuous representation whose spatial organization adapts to each input.

We introduce the *Function-Space Transformer* (FST), which represents each input through a continuous latent field defined by input-adaptive *anchors*. Each anchor pairs a coordinate predicted from the observations with a latent feature, and a weighted blend of the anchor features defines the field throughout the domain. This construction allows the model to learn where to store latent features rather than inherit their placement from the observation grid. The resulting representation supports both spatially resolved outputs, such as PDE solution fields, and finite-dimensional outputs, such as class predictions.

FST refines this representation through recursive function-space interactions. At each refinement step, the anchors first attend to the input observations. A function-space layer then evaluates the latent field at sampled points, applies global self-attention among the evaluated features, and writes the processed information back to the anchors through cross-attention. This separates the coordinates used to store the representation from those used to compute its global interactions. Rather than applying global attention directly to stored spatial latents (Knigge et al., 2024; Alkin et al., 2024; 2025) or modal tokens (Li et al., 2026), FST processes sampled evaluations of the continuous field and uses them to update its stored features. Parameters are shared across refinement steps, similar to the weight-sharing approaches used by Dehghani et al. (2019) and Jolicoeur-Martineau (2025). The input-predicted anchor coordinates remain fixed during refinement, while the anchor features and the field they define evolve. Our contributions are:

- **Adaptive continuous representation.** We introduce FST, which constructs a continuous latent field from features stored at input-predicted anchors. This allows the spatial organization of the representation to adapt to each input rather than follow the observation grid, while supporting both spatially resolved and finite-dimensional outputs.
- **Function-space interactions and recursive refinement.** We propose a shared function-space layer that applies global self-attention to sampled evaluations of the latent field and transfers the resulting information back to its anchors. This separates representation storage from interaction sampling and recursively refines the field with repeated access to the observations.
- **Experimental evaluation.** FST substantially outperforms the Perceiver IO baseline on Burgers and Darcy solution prediction and achieves accuracy comparable to FNO on all-viscosity Burgers and Darcy (Takamoto et al., 2022; Jaegle et al., 2022; Li et al., 2021). On ImageNet-1K, it achieves higher classification accuracy than the ViT baseline (Dosovitskiy et al., 2021). FST uses fewer parameters across these comparisons. Ablations examine adaptive anchors and support the benefits of function-space updates and recursive refinement.

2 Related Work

Attention-based learning in function space. Continuum attention defines attention as an operator between function spaces and studies its discrete approximation from sampled values (Calvello et al., 2025). Perceiver aggregates observations into a fixed number of latent features and applies self-attention among them (Jaegle et al., 2021); Perceiver IO adds output queries, separating latent computation from the output structure (Jaegle et al., 2022). For operator learning, the Inducing Point Operator Transformer and Universal Physics Transformers (UPT) use this separation to process latent features and predict function values at specified coordinates (Lee & Oh, 2024; Alkin et al., 2024). FST inherits this separation of latent processing and output queries; its difference lies in how the latent representation is organized and where interactions are computed.

Learning with spatially structured representations. LIIF, Convolutional Occupancy Networks, and GINO decode from or connect to latent features on fixed regular grids (Chen et al., 2021; Peng et al., 2020; Li et al., 2023); deformable attention adapts sampling locations while retaining a regular output grid (Xia et al., 2022). ENFs use Gaussian windows to localize the contributions of spatial latent features and fit these features and their geometry to each signal's observations (Wes-

Table 1: Where each architecture stores its latent representation and where it computes global interaction.

Model	Latent representation	Global interaction
Perceiver IO / UPT	Unstructured latent tokens	Self-attention among latent tokens
ENF	Latent features optimized for each input with Gaussian windows	Message passing among latents (ENF-PDE)
GPO	Predicted Gaussian geometry	Attention among modal tokens
AB-UPT	Spatial anchor tokens; separate query tokens	Self-attention among anchors; queries only read anchors
FST (ours)	Anchors predicted from the input, normalized RBFs	Attention among sampled field evaluations, written back

sels et al., 2025). Related continuous representations also appear in CViT (Wang et al., 2025b) and AROMA (Serrano et al., 2024). FST instead predicts input-adaptive anchor coordinates in a single forward pass and recursively refines their features for the target task.

These spatial representations also differ in how they exchange information (Table 1). For PDE forecasting, ENFs use message passing among pose–feature pairs (Knigge et al., 2024). GPO predicts Gaussian geometry, pools the representation into modal tokens, applies attention among them, and scatters the result back to spatial locations (Li et al., 2026). FST applies attention directly to features evaluated at sampled points, whose number can be chosen separately from the anchor count. AB-UPT applies self-attention among anchor tokens, while query tokens read from the anchors to produce independent predictions (Alkin et al., 2025). In FST, self-attention mixes the evaluated features, and cross-attention writes the result back to the anchor features before output decoding.

Recursive refinement. Perceiver repeatedly attends to the observations because a fixed latent can omit needed detail (Jaegle et al., 2021). Universal Transformers and recent recursive models refine latent states with a shared network (Dehghani et al., 2019; Wang et al., 2025a; Jolicoeur-Martineau, 2025), increasing depth through parameter sharing. FST uses this recursive approach to refine the anchor features.

3 METHOD

Let $\Omega \subseteq \mathbb{R}^{d_x}$ be a domain. We learn a map $\mathcal{G} : \mathcal{F}(\Omega; \mathbb{R}^{d_f}) \to \mathcal{Y}$, where $\mathcal{Y}$ is a function space $\mathcal{F}(\Omega; \mathbb{R}^{d_g})$ or $\mathbb{R}^C$. The model accesses the input f through N observations: $\mathcal{O} = \{(\mathbf{x}_i, \mathbf{v}_i)\}_{i=1}^N$, with values $\mathbf{v}_i \in \mathbb{R}^{d_v}$ at $\mathbf{x}_i \in \Omega$. For a function output $g = \mathcal{G}(f)$, the model predicts $\widehat{g}(\mathbf{q}_k) = \mathcal{G}_\theta(\mathcal{O})(\mathbf{q}_k)$ at output queries $\mathcal{Q} = \{\mathbf{q}_k\}_{k=1}^Q$; otherwise it returns $\widehat{\mathbf{y}} = \mathcal{G}_\theta(\mathcal{O}) \in \mathbb{R}^C$.

FST constructs a continuous latent field from input-adaptive anchors (Figure 1). A shared function-space layer refines this field R times. Each step attends to the observations, mixes features evaluated at sampled points, and writes the result back to the anchor features. The final anchor features are decoded at output queries for function prediction or pooled to produce a vector output.

FST uses three coordinate sets: M anchors at $\mathbf{X}_A$ store features $\mathbf{Z}$; S evaluation points at $\mathbf{X}_E$ specify where self-attention operates; and Q output queries in $\mathcal{Q}$ specify where function predictions are requested. These sets share a domain and may overlap.

Anchor Representation. Different inputs, such as PDE solutions with fronts at different locations, call for different spatial allocations. FST therefore predicts anchor coordinates from the observations. The coordinate predictor runs once per input and also supplies length scales when these are adaptive. Coordinates remain fixed during refinement (Appendix A.1).

Each anchor pairs a coordinate $\mathbf{a}_j \in \Omega$ with an anchor feature $\mathbf{z}_j \in \mathbb{R}^{d_z}$, giving the anchor set $\mathcal{A} = \{(\mathbf{a}_j, \mathbf{z}_j)\}_{j=1}^M$. The anchor features are initialized from a coordinate embedding γ and an optional global condition $\mathbf{c}$ as $\mathbf{z}_j^{(0)} = C_\theta(\gamma(\mathbf{a}_j), \mathbf{c})$.

Field construction. FST defines a continuous latent field using normalized Gaussian radial basis functions (RBFs), which construct continuous approximations from scattered coordinates (Buh-

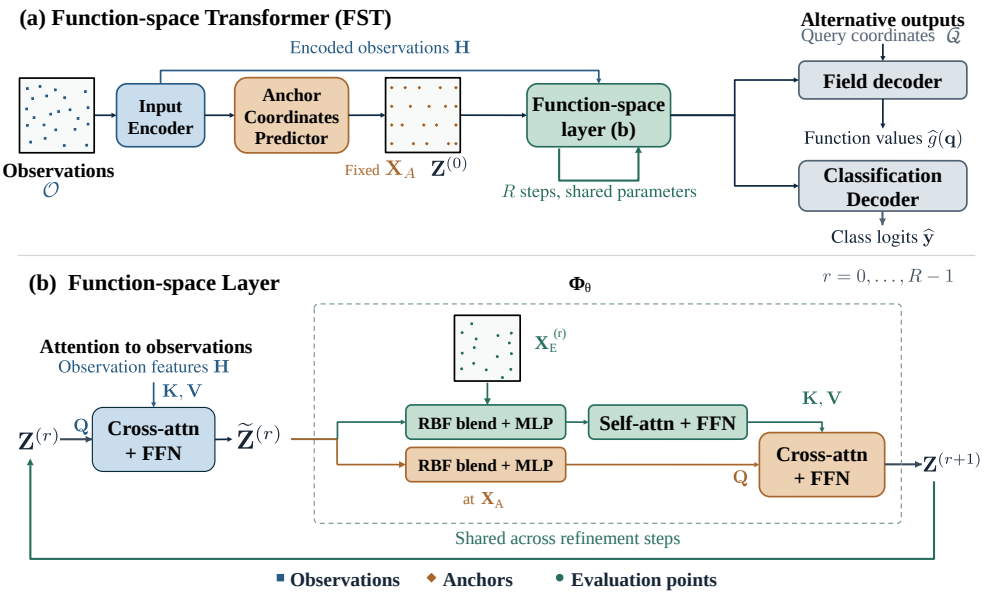

Figure 1: Function-Space Transformer (FST). (a) A shared layer refines anchor features for R steps at fixed coordinates $\mathbf{X}_A$, predicted once from the input. The decoder queries the field at $\mathcal{Q}$ or pools anchor features. (b) Each step attends to observations, evaluates RBF blends through a pointwise MLP at $\mathbf{X}_E^{(r)}$, and applies self-attention. Cross-attention updates anchors using queries from the same RBF blend and MLP at $\mathbf{X}_A$.

mann, 2003):

$$w_j(\mathbf{x}) = \frac{\exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{a}_j)^\top \Lambda_j^{-1}(\mathbf{x}-\mathbf{a}_j)\right)}{\sum_{m=1}^{M}\exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{a}_m)^\top \Lambda_m^{-1}(\mathbf{x}-\mathbf{a}_m)\right)}, \qquad \bar{\mathbf{z}}(\mathbf{x}) = \sum_{j=1}^{M} w_j(\mathbf{x})\mathbf{z}_j. \tag{1}$$

We call this RBF-weighted average of anchor features an *RBF blend*. Here, $\Lambda_j = \mathrm{diag}(\ell_{j,1}^2, \ldots, \ell_{j,d_x}^2)$ contains the RBF length scales of anchor j. They set the spatial spread of each Gaussian (Cavoretto, 2021, Section 3.2); setting $\Lambda_j = \ell_j^2\mathbf{I}$ gives isotropic weights. RBF length scales are fixed, predicted from the observations, or learned as input-independent model parameters. We denote the weights used during refinement by w_j^{ref} and their length scales by Λ^{ref}.

3.1 Function-Space Layer

The function-space layer updates the anchor features and hence the latent field while keeping anchor coordinates fixed.

Let $\mathbf{X}_A \in \mathbb{R}^{M\times d_x}$ and $\mathbf{Z}^{(r)} \in \mathbb{R}^{M\times d_z}$ denote the anchor coordinates and the anchor features at refinement step r, and let $\mathbf{H} \in \mathbb{R}^{N\times d_z}$ denote the encoded observations. Each refinement step has three stages.

(i) Attention to observations. As in Perceiver (Jaegle et al., 2021), anchors first attend to observations. An optional bias $\mathbf{B}^{\mathrm{obs}} \in \mathbb{R}^{M\times N}$ depends on anchor and observation coordinates:

$$\begin{aligned}\mathbf{U}^{(r)} &= \mathbf{Z}^{(r)} + \mathrm{MHA}_{\mathbf{B}^{\mathrm{obs}}}\Big(\mathrm{LN}_A(\mathbf{Z}^{(r)}), \mathrm{LN}_H(\mathbf{H}), \mathrm{LN}_H(\mathbf{H})\Big),\\ \widetilde{\mathbf{Z}}^{(r)} &= \mathbf{U}^{(r)} + \mathrm{FFN}\Big(\mathrm{LN}_2(\mathbf{U}^{(r)})\Big),\end{aligned} \tag{2}$$

with layer normalization (LN), a feed-forward network (FFN), and multi-head attention $\mathrm{MHA}_{\mathbf{B}}$ whose logits are $\mathbf{QK}^\top/\sqrt{d} + \mathbf{B}$, where d is the key dimension per head.

(ii) Evaluate and mix. Let $\mathbf{X}_E^{(r)} = [\mathbf{e}_1; \ldots; \mathbf{e}_S] \in \mathbb{R}^{S\times d_x}$ contain the evaluation coordinates for step r. Evaluation points can cover the domain uniformly or follow the anchor distribution to

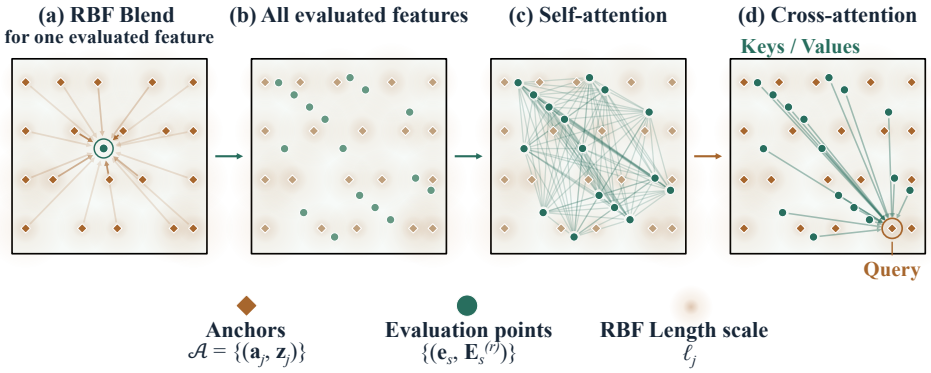

Figure 2: Evaluate, mix, and write-back. (a) RBF weights blend anchor features; a pointwise MLP, omitted for clarity, maps the blend and coordinate embedding to an evaluated feature. (b) Evaluation points may include anchor coordinates. (c) Self-attention mixes evaluated features. (d) Cross-attention incorporates information into anchor features. Its queries use the same RBF blend and MLP at anchor coordinates. Anchor coordinates stay fixed; one receiving anchor is highlighted.

emphasize regions with denser anchors. We combine these choices for PDE prediction. The points may be fixed or resampled across different steps.

We blend the anchor features at these coordinates and apply a pointwise MLP:

$$\begin{aligned} W_{sj}^{(r)} &= w_j^{\mathrm{ref}}(\mathbf{e}_s), \\ \mathbf{E}^{(r)} &= F_\theta\Big(W^{(r)}\widetilde{\mathbf{Z}}^{(r)}, \gamma(\mathbf{X}_E^{(r)}), \mathbf{c}\Big) \in \mathbb{R}^{S\times d_z}, \end{aligned} \tag{3}$$

where F_θ maps each RBF blend and coordinate embedding to an evaluated feature, with an optional global condition $\mathbf{c}$. Self-attention then mixes the S evaluated features:

$$\begin{aligned} \mathbf{G}^{(r)} &= \mathbf{E}^{(r)} + \mathrm{MHA}\Big(\mathrm{LN}_E(\mathbf{E}^{(r)}), \mathrm{LN}_E(\mathbf{E}^{(r)}), \mathrm{LN}_E(\mathbf{E}^{(r)})\Big), \\ \widetilde{\mathbf{G}}^{(r)} &= \mathbf{G}^{(r)} + \mathrm{FFN}\Big(\mathrm{LN}_G(\mathbf{G}^{(r)})\Big). \end{aligned} \tag{4}$$

(iii) Write back. Cross-attention writes information from the mixed features back to the anchor features. To form the queries, we apply the same RBF blending and pointwise MLP at the anchor coordinates. Even at $\mathbf{a}_j$, the blend combines features from all anchors, so $\bar{\mathbf{z}}(\mathbf{a}_j) \neq \mathbf{z}_j$ in general. With $W_{jm}^A = w_m^{\mathrm{ref}}(\mathbf{a}_j)$,

$$\begin{aligned} \mathbf{Q}_A^{(r)} &= F_\theta\Big(W^A\widetilde{\mathbf{Z}}^{(r)}, \gamma(\mathbf{X}_A), \mathbf{c}\Big), \\ \mathbf{V}^{(r)} &= \widetilde{\mathbf{Z}}^{(r)} + \mathrm{MHA}\Big(\mathrm{LN}_Q(\mathbf{Q}_A^{(r)}), \mathrm{LN}_E(\widetilde{\mathbf{G}}^{(r)}), \mathrm{LN}_E(\widetilde{\mathbf{G}}^{(r)})\Big), \\ \mathbf{Z}^{(r+1)} &= \mathbf{V}^{(r)} + \mathrm{FFN}\Big(\mathrm{LN}_V(\mathbf{V}^{(r)})\Big). \end{aligned} \tag{5}$$

The residual is added to the anchor features $\widetilde{\mathbf{Z}}^{(r)}$. The keys and values reuse LN_E from the self-attention step. Equations 2–5 together define the map $\mathbf{Z}^{(r+1)} = \Phi_\theta(\mathbf{Z}^{(r)}; \mathbf{H}, \mathbf{X}_A, \Lambda^{\mathrm{ref}}, \mathbf{c}, \mathbf{X}_E^{(r)})$, applied for $r = 0, \ldots, R-1$ with shared parameters (Dehghani et al., 2019). Note that only $\mathbf{X}_E^{(r)}$ and $\mathbf{Z}^{(r)}$ change across steps. Figure 2 illustrates stages (ii) and (iii).

3.2 Output

For field prediction, the output decoder blends the final anchor features at each output query. It uses its own length scales Λ^{dec} and corresponding weights w_j^{dec} from Equation 1, giving $\bar{\mathbf{z}}_{\mathrm{dec}}^{(R)}(\mathbf{q}) = \sum_j w_j^{\mathrm{dec}}(\mathbf{q})\mathbf{z}_j^{(R)}$. A pointwise MLP maps this blend to an output value:

$$\widehat{g}(\mathbf{q}) = D_\theta\Big(\mathbf{q}, \mathbf{X}_A, \mathbf{Z}^{(R)}, \Lambda^{\mathrm{dec}}, \mathbf{c}\Big) = \mathrm{MLP}_\theta\Big(\bar{\mathbf{z}}_{\mathrm{dec}}^{(R)}(\mathbf{q}), \gamma(\mathbf{q}), \mathbf{c}\Big), \tag{6}$$

where $\mathbf{c}$ is the optional global condition used during refinement. For vector outputs, we pool the final anchor features and apply an output head.

3.3 Design Properties

FST has the three properties below.

Permutation invariance. Reordering encoded observation pairs leaves the prediction unchanged. Reordering evaluation points also leaves the anchor update unchanged. These follow from the set symmetries of attention (Jaegle et al., 2021), allowing spatial samples to be processed independently of their storage order.

Spatial localization. Spatial localization allows each output query to approximate the full prediction using only nearby anchors, whose features already contain global information from refinement. This reduces the number of features combined per query, with approximation error controlled by the omitted weight. Section 4.4 evaluates this approximation on Burgers and Darcy.

Query consistency. Once the final anchor features are computed, adding, removing, or reordering output queries leaves predictions at shared coordinates unchanged. Queries can therefore be decoded in batches or added in regions of interest while reusing the refined representation.

At fixed feature width and anchor count, self-attention scales quadratically with the number of evaluation points, while output decoding scales linearly with the number of queries. Appendix A.3 gives the full computational cost.

4 Experiments

We evaluate FST on Burgers and Darcy flow for field prediction and on ImageNet-1K for classification. Burgers ablations examine anchor coordinates and length scales, function-space updates, and recursive refinement. Each reported result comes from one final training run.

4.1 Burgers Equation

Setup. Each trajectory in the one-dimensional PDEBench Burgers dataset (Takamoto et al., 2022) solves

$$\partial_t u(t,x) + \partial_x \left(\frac{u(t,x)^2}{2} \right) = \frac{\nu}{\pi} \partial_{xx} u(t,x), \qquad x \in (0,1), \quad t \in (0,2],$$

with periodic boundary conditions. Each solution is stored on a 201×1024 time-space grid. The dataset covers 12 viscosities, with 24,000 training, 1,200 validation, and 1,000 test solutions. All models predict solution fields from the same initial conditions (ICs) and boundary conditions (BCs), which together comprise 1,424 values. Appendix B gives the experiment details. We evaluate predictions using the mean relative L_2 error across solutions:

$$\overline{e}_{\mathrm{rel}} = \frac{1}{n} \sum_{i=1}^{n} \|\widehat{\mathbf{u}}_i - \mathbf{u}_i\|_2 / \|\mathbf{u}_i\|_2, \tag{7}$$

where $\mathbf{u}_i$ and $\widehat{\mathbf{u}}_i$ contain reference and predicted grid values for each of the n solutions.

We train on all 12 viscosities with viscosity as an additional input, and separately on $\nu = 0.01$ with 2,000 training, 100 validation, and 84 test solutions. Both FST models use four refinement steps.

Comparisons. We compare FST with FNO (Li et al., 2021) and Perceiver IO (Jaegle et al., 2022). The single-viscosity experiment also includes Transolver (Wu et al., 2024) and DeepONet (Lu et al., 2021). FST and Perceiver IO use 2,500 supervised points per solution at each training step, while FNO uses the full grid. All use a batch size of 48. For each model, we select the checkpoint with the lowest validation error and evaluate it on the test set.

All-viscosity results. Across 12 viscosities, FST substantially outperforms Perceiver IO and achieves error comparable to that of FNO (0.018180 versus 0.018528) at similar parameter counts (Table 2). Figure 3 shows that FST's error is 7.1–14.4% lower than FNO's at the five lowest viscosities (0.001–0.02; Table 19). Lower viscosity produces sharper structures, which Takamoto et al. (2022) associate with increased difficulty for FNO. At $\nu = 0.001$, FST captures front locations and

Table 2: Burgers results across all 12 viscosities. Stop step: end of training. Selected steps (lowest validation error): FST 450k, FNO 400k, Perceiver IO 895k.

Model	Supervision per step	Parameters	Stop step	Relative L_2
FST	2,500 points	2.901M	500,000	**0.018180**
FNO	full grid	2.978M	400,000	0.018528
Perceiver IO	2,500 points	2.995M	900,000	0.073293

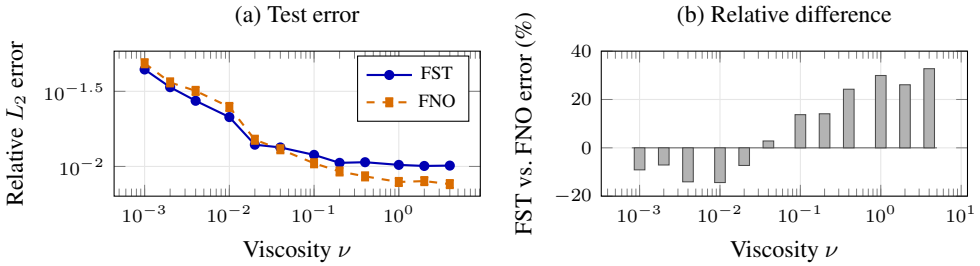

Figure 3: Per-viscosity test error on PDEBench Burgers (all-viscosity models). (a) Relative L_2 error. (b) $100\,(e_{\mathrm{FST}} - e_{\mathrm{FNO}})/e_{\mathrm{FNO}}$; negative values favor FST. All 12 viscosities are shown.

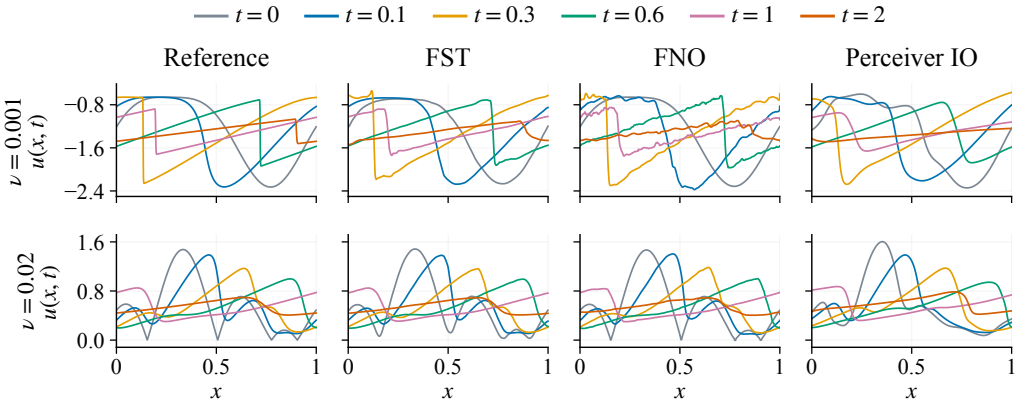

Figure 4: Burgers profiles at $\nu = 0.001$ (top) and $\nu = 0.02$ (bottom). Colors indicate time; each row shares axis limits. FST captures fronts with fewer spurious oscillations (top) but rounds initial-state minima (bottom).

Table 3: Test relative L_2 error on Burgers ($\nu = 0.01$) at four output grids. See Appendix B for detailed inputs, outputs, and evaluation.

Model	Parameters	Supervision per step	32×32	64×64	128×128	201×1024
FST	2.831M	2,500 points	0.04118	0.04159	0.04171	0.04185
FNO	2.978M	full grid	**0.02854**	**0.02950**	**0.02963**	**0.02974**
Perceiver IO	2.937M	2,500 points	0.25634	0.25465	0.25396	0.25381
Transolver	2.956M	2,500 points	0.17709	0.07151	0.06569	0.06154
DeepONet	2.998M	2,500 points	0.24712	0.25204	0.25467	0.25581

steepness with fewer spurious oscillations than FNO, while Perceiver IO broadens or displaces transitions (Figure 4). At $\nu = 0.02$, FST follows the later-time profiles but rounds sharp minima of the (observed) initial state, which FNO reconstructs more faithfully. Appendix B.5 shows more cases.

Single-viscosity results. Across all four output grids, FST ranks second to FNO (Table 3). FST's error stays near 0.042 across grids with fixed input observations. FST, Perceiver IO, and DeepONet evaluate the same predicted function at different points.

Table 4: Ablation on anchor coordinates and length scales.

Coordinates	Length scales	32^2 anchors	16^2 anchors
Uniform	Fixed	0.04604	<u>0.04571</u>
Adaptive	Fixed	**0.04069**	0.05784
Uniform	Adaptive	0.05482	0.05094
Adaptive	Adaptive	<u>0.04185</u>	**0.04480**

Table 5: Ablation on the evaluation, mixing, and write-back process.

Update	Error
Evaluation, mixing, and write-back	**0.04069**
Two self-attentions on anchors	<u>0.05212</u>

Table 6: Ablation on recursive refinement.

Refinement steps	Attention to observations	Error
1	Every step	0.06209
2	Every step	<u>0.04442</u>
4	Every step	**0.04069**
8	Every step	0.06496
4	First step only	0.05971

4.2 Burgers Ablations

All ablations use the single-viscosity Burgers task ($\nu = 0.01$). Unless stated otherwise, they use 32×32 anchors, adaptive coordinates, RBF length scales fixed at 0.03, four refinement steps, and the sampling settings in Appendix B.2. The 32×32 model with adaptive coordinates and scales is the FST checkpoint in Table 3. All the compared models are evaluated on the full grid.

Anchor coordinates and length scales. The best configuration at each anchor count uses adaptive coordinates. At 16×16, adapting coordinates and scales together gives the lowest error (0.04480), compared with 0.05784 for adaptive coordinates alone and 0.04571 for uniform coordinates and fixed scales. At 32×32, learning with adaptive coordinates perform better.

Evaluation, mixing, and write-back. We replace the evaluation, mixing, and write-back process with two self-attention blocks on the anchor features to test whether computing global interactions on the function domain gives better results. Table 5 shows that this evaluation, mixing, and write-back process reduces error by 21.9%, from 0.05212 to 0.04069.

Recursive refinement. Table 6 shows that increasing the number of refinement steps can improve performance: four steps give the lowest error, 34.5% below that of one step. However, increasing to eight steps harms performance. Attending to observations at every step reduces error by 31.9% compared with attending only at the first step, showing its necessity in the refinement process.

4.3 Darcy Flow

Setup. We use 2,000 training, 100 validation, and 1,000 test solutions from the NeuralOperator Darcy dataset (NeuralOperator Team, 2021). Models map coefficient fields to solution fields on a 128×128 grid with full-grid supervision. We select checkpoints using a 100×100 validation subset and report test relative L_2 error on the full grid.

Comparisons. We compare FST with FNO, Perceiver IO, Transolver, and DeepONet using the same tuning budget and convergence rule. Appendix C gives the model and training settings.

Table 7: Darcy flow: test relative L_2 error on the full 128×128 grid.

Model	Params.	Sel. step	Rel. L_2
FST	2.831M	58k	<u>0.03173</u>
FNO	2.911M	42k	**0.03167**
Perceiver IO	2.937M	58k	0.10102
Transolver	2.956M	36k	0.03671
DeepONet	2.993M	18k	0.14678

Table 8: ImageNet-1K validation accuracy at the selected checkpoints.

Model	Params.	Top-1 (%)	Top-5 (%)
FST	2,959,929	**69.530**	**89.042**
ViT	3,095,614	67.804	88.126
CNN-ViT	3,158,150	61.052	83.478

Results. Table 7 shows that FST achieves comparable error to FNO (0.03173 versus 0.03167) at a similar parameter count and outperforms the other baselines. Figure 10 in Appendix C shows the predicted anchor coordinates and decoding length scales.

4.4 Decoding with Nearby Anchors

Spatial localization allows each query to approximate the full prediction using fewer nearby anchors. We retain the k nearest anchors by Euclidean distance in the normalized domain during decoding and renormalize their RBF weights; refinement uses all 1,024 anchors. Figure 5 shows the error approaching that of full decoding as the neighborhood grows. Burgers reaches nearly the same error with 16 neighbors, while Darcy requires a larger neighborhood. Evaluation uses 84 Burgers and 1,000 Darcy test cases. These results suggest that decoding from fewer anchors could reduce decoding cost.

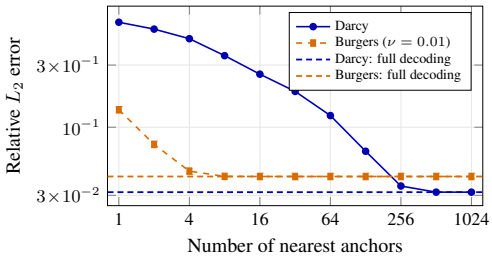

Figure 5: Full-grid relative L_2 error using different numbers of nearby anchors.

4.5 Image Classification

Viewing images as color functions over space, we evaluate FST on image classification to test its effectiveness beyond physical fields.

Setup. ImageNet-1K (Russakovsky et al., 2015) contains 1,281,167 training images and 50,000 validation images. Each image produces 196 non-overlapping 16×16 patches. For FST, each encoded patch is paired with its center coordinate. We report top-1 and top-5 accuracy.

Comparisons. We train FST, ViT (Dosovitskiy et al., 2021), and CNN-ViT from scratch. FST and CNN-ViT use the same residual convolutional patch encoder; ViT uses a linear projection. All three train to convergence under the same stopping rule. Appendix D gives their training settings.

Results. FST reaches 69.530% top-1 accuracy, 1.73 points above ViT, with fewer parameters (Table 8). With the same patch encoder, CNN-ViT reaches 61.052%. Appendix D.4 shows how the model distributes anchors across the image plane, with more anchors concentrating on relevant objects.

5 Conclusion

We proposed FST, which represents each input as a continuous latent field defined by adaptive anchors. Its function-space layer separates where features are stored from where global interactions are computed. Self-attention mixes features evaluated at sampled points, and cross-attention uses this information to refine the anchor features. Experiments on Burgers and Darcy flow demonstrate FST's effectiveness for function prediction. Its performance on ImageNet-1K further demonstrates its ability to map input functions to finite-dimensional outputs. Ablations support the benefits of function-space updates and recursive refinement. The refined representation also supports queries at arbitrary coordinates, while spatial localization allows approximate decoding from fewer nearby anchors. Our experiments focus on rectangular domains. We expect to extend FST to irregular geometries by adapting anchor prediction and evaluation-point sampling to the domain.

AI use statement

We used AI tools to polish the writing, help organize the paper, search for related work, and implement and run parts of the experimental code. We reviewed the AI-assisted work, checked the implementation, ran tests, and compared outputs with reference results. The authors take responsibility for the final paper, code, and reported results.

REFERENCES

Benedikt Alkin, Andreas Fürst, Simon Schmid, Lukas Gruber, Markus Holzleitner, and Johannes Brandstetter. Universal physics transformers: A framework for efficiently scaling neural operators. In *Advances in Neural Information Processing Systems*, volume 37, pp. 25152–25194, 2024. doi: 10.52202/079017-0793. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/2cd36d327f33d47b372d4711edd08de0-Abstract-Conference.html.

Benedikt Alkin, Maurits Bleeker, Richard Kurle, Tobias Kronlachner, Reinhard Sonnleitner, Matthias Dorfer, and Johannes Brandstetter. AB-UPT: Scaling neural CFD surrogates for high-fidelity automotive aerodynamics simulations via anchored-branched universal physics transformers. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=nwQ8nitlTZ.

Martin D. Buhmann. Radial basis functions on scattered data. In *Radial Basis Functions: Theory and Implementations*, pp. 99–146. Cambridge University Press, 2003. doi: 10.1017/CBO9780511543241.006.

Edoardo Calvello, Nikola B. Kovachki, Matthew E. Levine, and Andrew M. Stuart. Continuum attention for neural operators. *Journal of Machine Learning Research*, 26(300):1–52, 2025. URL https://www.jmlr.org/papers/v26/24-0879.html.

Roberto Cavoretto. Adaptive radial basis function partition of unity interpolation: A bivariate algorithm for unstructured data. *Journal of Scientific Computing*, 87:41, 2021. doi: 10.1007/s10915-021-01432-z.

Yinbo Chen, Sifei Liu, and Xiaolong Wang. Learning continuous image representation with local implicit image function. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8628–8638, 2021. URL https://openaccess.thecvf.com/content/CVPR2021/papers/Chen_Learning_Continuous_Image_Representation_With_Local_Implicit_Image_Function_CVPR_2021_paper.pdf.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyzdRiR9Y7.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4651–4664. PMLR, 2021. URL https://proceedings.mlr.press/v139/jaegle21a.html.

Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, Olivier J. Henaff, Matthew Botvinick, Andrew Zisserman, Oriol Vinyals, and Joao Carreira. Perceiver IO: A general architecture for structured inputs and outputs. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=fILj7WpI-g.

Alexia Jolicoeur-Martineau. Less is more: Recursive reasoning with tiny networks. *arXiv preprint arXiv:2510.04871*, 2025. URL https://arxiv.org/abs/2510.04871.

David M. Knigge, David R. Wessels, Riccardo Valperga, Samuele Papa, Jan-Jakob Sonke, Efstratios Gavves, and Erik J. Bekkers. Space-time continuous PDE forecasting using equivariant neural fields. In *Advances in Neural Information Processing*

Systems, volume 37, pp. 76553–76577, 2024. doi: 10.52202/079017-2438. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/8c2de4155634a20d903c2ab0b1784886-Abstract-Conference.html.

Seungjun Lee and Taeil Oh. Inducing point operator transformer: A flexible and scalable architecture for solving PDEs. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(1):153–161, 2024. doi: 10.1609/aaai.v38i1.27766. URL https://ojs.aaai.org/index.php/AAAI/article/view/27766.

Zhihao Li, Yu Feng, Zhilu Lai, and Wei Wang. From basis to basis: Gaussian particle representation for interpretable PDE operators. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=TYswUgruqt.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=c8P9NQVtmnO.

Zongyi Li, Nikola Kovachki, Chris Choy, Boyi Li, Jean Kossaifi, Shourya Otta, Mohammad Amin Nabian, Maximilian Stadler, Christian Hundt, Kamyar Azizzadenesheli, and Animashree Anandkumar. Geometry-informed neural operator for large-scale 3d PDEs. In *Advances in Neural Information Processing Systems*, volume 36, pp. 35836–35854, 2023. doi: 10.52202/075280-1556. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/70518ea42831f02afc3a2828993935ad-Abstract-Conference.html.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3:218–229, 2021. doi: 10.1038/s42256-021-00302-5.

NeuralOperator Team. Darcy flow dataset, 2021. URL https://doi.org/10.5281/zenodo.12784353.

Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *Computer Vision – ECCV 2020*, volume 12348 of *Lecture Notes in Computer Science*, pp. 523–540. Springer, 2020. doi: 10.1007/978-3-030-58580-8_31.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Louis Serrano, Thomas X. Wang, Etienne Le Naour, Jean-Noël Vittaut, and Patrick Gallinari. AROMA: Preserving spatial structure for latent PDE modeling with local neural fields. In *Advances in Neural Information Processing Systems*, volume 37, pp. 13489–13521, 2024. doi: 10.52202/079017-0431.

Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Daniel MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. PDEBench: An extensive benchmark for scientific machine learning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 1596–1611, 2022. doi: 10.52202/068431-0117. URL https://proceedings.nips.cc/paper_files/paper/2022/hash/0a9747136d411fb83f0cf81820d44afb-Abstract-Datasets_and_Benchmarks.html.

Hugo Touvron, Matthieu Cord, and Hervé Jégou. Deit iii: Revenge of the vit. In *European conference on computer vision*, pp. 516–533. Springer, 2022.

Guan Wang, Jin Li, Yuhao Sun, Xing Chen, Changling Liu, Yue Wu, Meng Lu, Sen Song, and Yasin Abbasi Yadkori. Hierarchical reasoning model. *arXiv preprint arXiv:2506.21734*, 2025a. URL https://arxiv.org/abs/2506.21734.

Sifan Wang, Jacob H. Seidman, Shyam Sankaran, Hanwen Wang, George J. Pappas, and Paris Perdikaris. CViT: Continuous vision transformer for operator learning. In *International Conference on Learning Representations*, 2025b. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/3bf4b55960aaa23553cd2a6bdc6e1b57-Abstract-Conference.html.

David R. Wessels, David M. Knigge, Riccardo Valperga, Samuele Papa, Sharvaree Vadgama, Efstratios Gavves, and Erik J. Bekkers. Grounding continuous representations in geometry: Equivariant neural fields. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=A4eCzSohhx.

Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. Transolver: A fast transformer solver for PDEs on general geometries. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 53681–53705. PMLR, 2024. URL https://proceedings.mlr.press/v235/wu24r.html.

Zhuofan Xia, Xuran Pan, Shiji Song, Li Erran Li, and Gao Huang. Vision transformer with deformable attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4794–4803, 2022. URL https://openaccess.thecvf.com/content/CVPR2022/papers/Xia_Vision_Transformer_With_Deformable_Attention_CVPR_2022_paper.pdf.

A Architecture Details

A.1 Anchor Coordinates and Length Scales

We parameterize $M = P(J+1)$ anchors on P fixed rows, with ordered, input-adaptive positions along each row. A transformer decoder predicts the spacing between anchors on each row. Its J learned query tokens attend to the encoded observations, each predicting one gap score for each of the P rows. We randomly initialize the tokens and learn them with the rest of the model.

For each row, we convert the scores α_j into positive gaps and normalize them so their sum equals the domain length:

$$\widetilde{\Delta}_j = \frac{1}{2}\left(\alpha_j + \sqrt{\alpha_j^2 + 4}\right) + \varepsilon_g, \qquad \Delta_j = (x_{\max} - x_{\min})\frac{\widetilde{\Delta}_j}{\sum_{m=1}^{J}\widetilde{\Delta}_m}. \tag{8}$$

Starting at $x_{\min}$, we add the gaps successively to obtain the $J+1$ anchor coordinates:

$$\chi_0 = x_{\min}, \qquad \chi_j = x_{\min} + \sum_{m=1}^{j}\Delta_m, \quad j = 1, \ldots, J. \tag{9}$$

The coordinates stay ordered, with the first and last anchors at the domain endpoints. Figure 6 illustrates this construction.

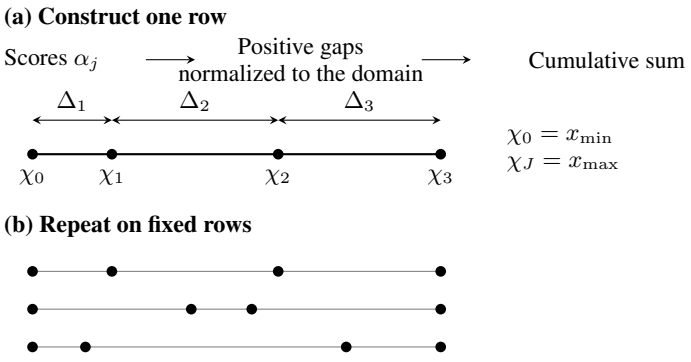

Figure 6: Converting predicted gaps to anchor coordinates. Each row has its own predicted spacing.

RBF length scales. For Burgers and Darcy, refinement and output decoding use separate learned base length scales for each anchor. Both use the same length-scale adjustments predicted from the input. For ImageNet, FST learns one scalar RBF length scale per anchor, shared across inputs. Coordinates and scales stay fixed across refinement.

A.2 Model Configurations

Tables 9 and 10 give the FST dimensions and inputs to the coordinate predictor for each task. Sobol sequences are scrambled. The Burgers sampling counts use 32^2 anchors; Appendix B.2 gives the 16^2 ablation.

Table 9: FST architecture settings for the reported experiments.

Setting	Burgers	Darcy	ImageNet-1K
Anchors M	$32 \times 32 = 1{,}024$	$32 \times 32 = 1{,}024$	$32 \times 32 = 1{,}024$
Anchor feature dimension	176	176	200
Main attention heads	8	8	8
Observation-attention bias	Zero	Zero	Eq. 11
Refinement steps R	4	4	4
Evaluation points S	512	512	512
Evaluation points: anchor / Sobol	256 / 256	256 / 256	0 / 512
Evaluation-point resampling	Every step	Every step	None
Adaptive coordinate	Spatial	One spatial axis	Horizontal
Fixed rows P	32 (time)	32	32 (image)
Predictor query tokens J	31	31	31
Coordinate predictor width	144	144	104
Predictor attention heads / blocks	4 / 2	4 / 2	4 / 2
Positive-gap offset ε_g	0.10	0.10	0.10
Output-head weight / bias initialization	0 / 0	0 / 0	0 / 0
RBF length-scale form	Axis-wise	Axis-wise	Scalar
RBF length-scale source	Input-predicted	Input-predicted	Learned per anchor
RBF length-scale initialization	0.03	0.03	0.03
RBF length-scale bounds	$[0.015, 0.30]$	$[0.015, 0.30]$	$[0.015, 0.30]$
Separate decoder length scales	Yes	Yes	n/a (pooled head)

The fixed-scale Burgers ablations use 0.03.

Table 10: Coordinate predictor encodings and observation inputs.

Component	Burgers	Darcy	ImageNet-1K
Coordinate predictor	Fourier coordinate embeddings	Fourier coordinate embeddings	Fourier embeddings of patch-center coordinates
Predictor observation inputs	Coordinates, values, viscosity	Coordinates, coefficient values	Patch centers, patch features

A.3 Computational Cost

At fixed feature width, the leading-order cost of a dense implementation is

$$O\big(C_{\text{enc}} + C_{\text{coord}} + R\,(MN + MS + S^2 + M^2) + MQ\big)\,.$$

C_{enc} and C_{coord} denote the costs of the input encoder and coordinate predictor. The predictor's J query tokens attend to the N observations and predict PJ gap scores once per input, giving $C_{\text{coord}} = O(JN + J^2 + PJ)$ at fixed width and depth.

Per refinement step, the attention to observations costs $O(MN)$; evaluating the field at S points (Equation 3) and the write-back cross-attention each cost $O(MS)$; self-attention among evaluated features costs $O(S^2)$; and forming the write-back queries $\mathbf{Q}_A^{(r)}$ for all anchors costs $O(M^2)$.

Decoding Q queries costs $O(MQ)$ and reuses the final anchor representation, so for fixed M the decoding cost grows linearly in Q.

B BURGERS EXPERIMENT DETAILS

B.1 DATA AND MODEL INPUTS

The all-viscosity experiment uses all 12 viscosities; the single-viscosity experiment uses $\nu = 0.01$. All models receive the same ICs and BCs. The ICs contain the solution at $t = 0$ on all 1,024 spatial grid points. The BCs contain the solution values at the two spatial boundary grid points at each of the 200 later time steps. Together, they provide $1{,}024 + 2 \times 200 = 1{,}424$ input values. Table 11 summarizes the datasets and input sampling.

Table 11: Burgers datasets and shared observation grid.

Setting	All viscosities	$\nu = 0.01$
Training samples	24,000	2,000
Validation samples	1,200	100
Test samples	1,000	84
Original grid (time × space)	201×1024	
IC values	1,024	
BC values	$2 \times 200 = 400$	

B.2 MODEL CONFIGURATIONS

FST output decoder. For each query, we blend the final anchor features with RBF weights, then use pointwise MLPs to predict the solution value. Query embeddings condition the prediction; the all-viscosity model also uses a viscosity embedding (Figure 7). The output decoder has its own RBF length scales and MLP parameters.

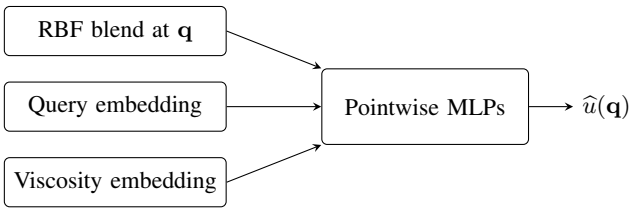

Figure 7: Burgers decoding at one query coordinate.

All-viscosity inputs. For the all-viscosity experiment, all models need to condition on the viscosity ν. FST and Perceiver IO embed it as a global condition; FNO broadcasts it as a constant input channel. In FST, a learned viscosity embedding is also added to the initial anchor features:

$$\mathbf{z}_j^{(0)} = C_a(\gamma(\mathbf{a}_j)) + C_c(\mathbf{c}).$$

Table 12 compares the inputs and supervision.

Table 12: Model inputs and output supervision for the Burgers comparison.

Model	Input	Supervision per training step
FST	ICs and BCs + ν	2,500 sampled points
Perceiver IO	ICs and BCs + ν	2,500 sampled points
FNO	Values + mask + ν on the 201×1024 grid	Full grid (205,824 points)

Baseline architectures. FNO and Perceiver IO use the same widths and depths in both Burgers experiments (Table 13). DeepONet uses branch widths $1424 \to 918 \to 918$ and trunk widths $2 \to 918 \to 918$, with ReLU activations and Glorot-normal initialization.

Table 13: Burgers baseline architecture settings. Fourier modes follow the time-space axis order.

Model	Width	Layers	Heads	Latents/slices	Modes
FNO	48	4	-	-	10×30
Perceiver IO	240	6	8	256	-
Transolver	328	5	8	32	-

Single-viscosity inputs and outputs. FNO and Transolver receive the same 1,424 IC and BC observations, without viscosity conditioning. We use the official FNO and irregular-mesh Transolver implementations with the following input and output adapters.

FNO. We place the observed values on the original 201×1024 time-space grid and fill unobserved entries with zeros. A second channel marks observed entries with one and unobserved entries with zero. FNO predicts the full grid and receives full-grid supervision. Its predictions at each evaluation grid are extracted from the original-grid prediction (Appendix B.4).

Transolver. We combine the observations $\mathcal{O} = \{(\mathbf{x}_i, \mathbf{v}_i)\}_{i=1}^{N}$ and queries $\mathcal{Q} = \{\mathbf{q}_k\}_{k=1}^{Q}$ into one point set. Each observation is represented as $(\mathbf{x}_i, \mathbf{v}_i, 1)$, where $\mathbf{v}_i$ is its standardized IC and BC value; each query is represented as $(\mathbf{q}_k, \mathbf{0}, 0)$. The final entry is a mask indicating whether the value is observed. Transolver predicts at all supplied coordinates; we retain and supervise only the query outputs.

FST configurations and ablations. Table 14 gives the main FST configurations and the default ablation settings. Coordinate and length-scale ablations vary the coordinates and length scales as shown in Table 4. For both main Burgers models, the effective length scale of anchor j along axis d is

$$\ell_{j,d}^{s} = \mathrm{clip}\big(\exp\big(b_j^s + \delta_{j,d}(\mathcal{O})\big), 0.015, 0.30\big), \qquad s \in \{\mathrm{ref}, \mathrm{dec}\}. \tag{10}$$

Here b_j^s is a learned log base scale, separate for refinement and decoding. The predictor supplies the shared axis-wise adjustment $\delta_{j,d} = \log(4)\tanh(h_{j,d})$. Interior anchors average the outputs of the two adjacent gap tokens before this transformation; endpoints use the nearest token. The two anchor counts use different numbers of sampled anchors: the 32×32 model samples 256 anchors (25%) and 256 Sobol points per step, whereas the 16×16 model samples 128 anchors (50%) and 384 Sobol points.

Table 14: FST configurations for the Burgers comparisons and the default ablation. The default settings are varied in Tables 4-6.

Experiment	Anchors	Coordinates	Length scales	R
All viscosities	32^2	Adaptive	Adaptive	4
Single viscosity	32^2	Adaptive	Adaptive	4
Default ablation	32^2	Adaptive	0.03	4

For the update ablation, two self-attention blocks on the anchor features replace stages (ii)-(iii) of the function-space layer. The replacement changes the features entering attention, the number of interaction tokens, and the write-back together, so the comparison supports the update block as a whole rather than isolating one of these factors.

B.3 Training and Checkpoint Selection

All Burgers models minimize the mean squared per-solution relative L_2 error in the original output units. Input values and prediction targets are standardized with shared scalar mean and standard deviation. The all-viscosity runs use -0.012048 and 0.621534; the single-viscosity runs compute these statistics from their training targets on the 100×100 grid.

Both Burgers FST comparisons sample 2,500 training queries per solution and training step from a 100×100 subset of the original grid. We select the checkpoint with the lowest validation relative L_2 error and evaluate it once on the test set.

All-viscosity training. The all-viscosity comparison uses a shared convergence rule. After a minimum training period, each model continues training until its best validation relative L_2 error improves by less than 1% over 50,000 steps. Table 15 gives the training settings.

Table 15: Shared training settings for the all-viscosity Burgers comparison.

Setting	Value
Optimizer	Adam
Global batch size	48
Forward / backward precision	bfloat16 (BF16)
Loss accumulation precision	32-bit floating point (FP32)
Minimum training steps	250,000
Maximum training steps	1,000,000

All-viscosity FST uses learning rates 10^{-3} for the main network and 10^{-5} for the coordinate predictor. Both remain constant through step 200,000, decay by a cosine schedule to 5% of their initial values at step 250,000, and remain constant thereafter. FNO uses the same schedule from 5×10^{-3}; Perceiver IO uses a constant 10^{-5}.

All-viscosity FST selects step 450,000, with validation error 0.018014, and stops at step 500,000. FNO selects and stops at step 400,000; Perceiver IO selects step 895,000 and stops at 900,000.

Single-viscosity training. FST and the four baselines use batch size 48, 1,000 warmup steps, validation every 2,000 steps, and at most 250,000 steps. The initial learning rate is 10^{-3} for FST, 3×10^{-3} for FNO, and 3×10^{-4} for Perceiver IO, Transolver, and DeepONet. FST uses the same learning rate for its predictor and main network. If the best validation error improves by less than 1% over 6,000 steps, the learning rate is divided by ten, up to two times. At the final learning rate, training stops when improvement is below 0.5% over 12,000 steps, after at least 20,000 total steps.

Table 16: Checkpoints for the single-viscosity models in Table 3.

Model	Selected step	Stop step
FST	92,000	98,000
FNO	72,000	72,000
Perceiver IO	34,000	50,000
Transolver	80,000	82,000
DeepONet	32,000	48,000

Ablation checkpoints. Each ablation setting is trained separately. Table 17 lists the selected steps for the settings in Table 4.

Table 17: Selected training steps for the Burgers ablations; k denotes 1,000 steps. Left: coordinate and length-scale ablations at two anchor counts. Right: update and refinement ablations with 32^2 anchors.

Coordinates	Scales	32^2	16^2
Uniform	Fixed	82k	116k
Adaptive	Fixed	86k	80k
Uniform	Adaptive	130k	112k
Adaptive	Adaptive	92k	78k

Variant	Selected step
Anchor self-attention	108k
$R = 1$	76k
$R = 2$	74k
$R = 4$	86k
$R = 8$	74k
Attention to observations	
First step only	108k

Training cost. Table 18 reports the hardware, training time, and memory for the all-viscosity runs.

Table 18: All-viscosity training on NVIDIA A100 80GB GPUs. Time sums the recorded training segments, including validation. Memory is the peak allocated memory per GPU.

Model	GPUs	Time (h)	Memory (GiB)
FST	1	30.5	8.80
FNO	3	32.9	15.21
Perceiver IO	3	18.6	2.32

B.4 Evaluation

Validation grids. For checkpoint selection in the single-viscosity comparison, FST and FNO use a 100×100 subset of the original grid; Perceiver IO, Transolver, and DeepONet use 64×64.

Test grids and error. The all-viscosity FST checkpoint is evaluated in FP32 on the full 201×1024 grid of all 1,000 test solutions. The single-viscosity comparison uses the four grids in Table 3: 32×32, 64×64, 128×128, and the original 201×1024 grid. The ablations use the full original grid. For the single-viscosity comparison and ablations, Equation 7 is computed per solution over the evaluation grid and averaged over the 84 test solutions.

Each output grid is a Cartesian product of original-grid subsets. For n points on an axis of size N, we round $\mathrm{linspace}(0, N-1, n)$ to integer indices and normalize them by $N-1$.

Model evaluation. FST, Perceiver IO, and DeepONet are queried directly at these coordinates. FNO predictions are extracted at the corresponding indices on the original grid. For Transolver, we permute the query indices with seed 2026091601 and partition them into groups of at most 2,500, each accompanied by all 1,424 observations. The four grids require 1, 2, 7, and 83 groups, respectively. Validation uses the same grouping procedure. All reported test evaluations use FP32. At each refinement step, FST draws 256 anchor indices without replacement and 256 scrambled Sobol points. Before each test-grid evaluation, we reset the anchor-selection seed to 2026091322 and the Sobol seed to 2026091323, keeping the sample order and batch size (8) fixed. This reproduces the same evaluation points across the four output grids. All query batches decode the resulting final anchor features.

B.5 Additional Results

Per-viscosity results. Table 19 reports the relative L_2 error within each viscosity group of the all-viscosity test set; Figure 3 plots the FST and FNO columns. FST has lower error than FNO at the five lowest viscosities, from 0.001 through 0.02; FNO has lower error at the seven higher viscosities. Perceiver IO has the highest error at every viscosity.

Table 19: Per-viscosity relative L_2 error on PDEBench Burgers. Best in bold; second best underlined.

ν	FST	FNO	Perceiver IO
0.001	**0.044195**	<u>0.048623</u>	0.147162
0.002	**0.033692**	<u>0.036266</u>	0.120887
0.004	**0.027352**	<u>0.031828</u>	0.112725
0.01	**0.021308**	<u>0.024882</u>	0.088623
0.02	**0.013957**	<u>0.015056</u>	0.063128
0.04	<u>0.013382</u>	**0.013013**	0.061419
0.1	<u>0.011940</u>	**0.010499**	0.059270
0.2	<u>0.010567</u>	**0.009262**	0.043572
0.4	<u>0.010670</u>	**0.008588**	0.044528
1	<u>0.010242</u>	**0.007883**	0.044719
2	<u>0.010076</u>	**0.007991**	0.041947
4	<u>0.010131</u>	**0.007632**	0.049411

Solution profiles. We extend Figure 4 with profiles at additional viscosities and with different initial conditions. All examples come from the held-out test set and use the reported all-viscosity checkpoints. The four initial-condition examples are selected by profile shape before inference.

Models share the observations, six plotted times, and 1,024 spatial coordinates on the original grid. Curves show predictions, including at $t = 0$, in the original units. All three figures use the same column order and colors for each time. Within each row, the reference and all predictions share axis limits.

Profiles at selected viscosities. Figure 8 compares selected trajectories at the five lowest viscosities. FST and FNO recover the main wave patterns across these examples, with local differences in front shape, spurious oscillations, and reconstruction of sharp initial-state features. Perceiver IO produces broader transitions and larger errors in oscillation amplitudes in several cases.

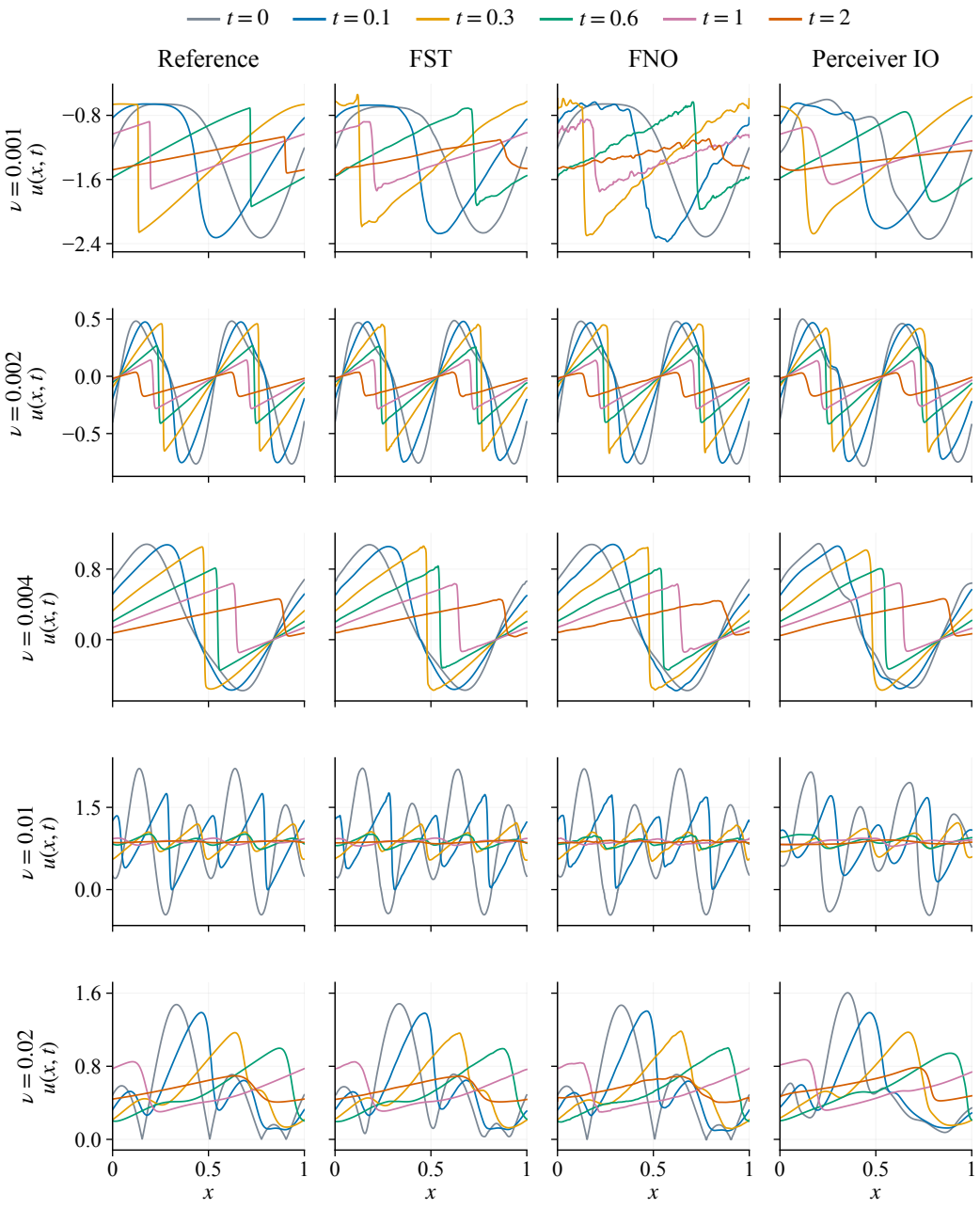

Figure 8: Additional Burgers solution profiles. From top to bottom, rows show selected trajectories at $\nu = 0.001$, 0.002, 0.004, 0.01, and 0.02. Columns show the reference solution and predictions from FST, FNO, and Perceiver IO. Colors indicate $t \in \{0, 0.1, 0.3, 0.6, 1, 2\}$; axis limits are shared across columns within each row. The first and last rows are the examples in Figure 4. Different rows have different initial conditions.

Different initial conditions. Figure 9 compares four initial conditions at $\nu = 0.002$. FST and FNO recover the principal evolving fronts and oscillatory structures in these examples. FST exhibits fewer small fluctuations along several smooth segments, although local artifacts remain, including an overshoot near the $t = 0.3$ front in the single-wave example. Perceiver IO broadens several transitions and distorts some oscillation amplitudes. The example with localized oscillations shows that FST can preserve rapidly varying initial-state structure, even though it rounds some corner-like minima in the example of Figure 4.

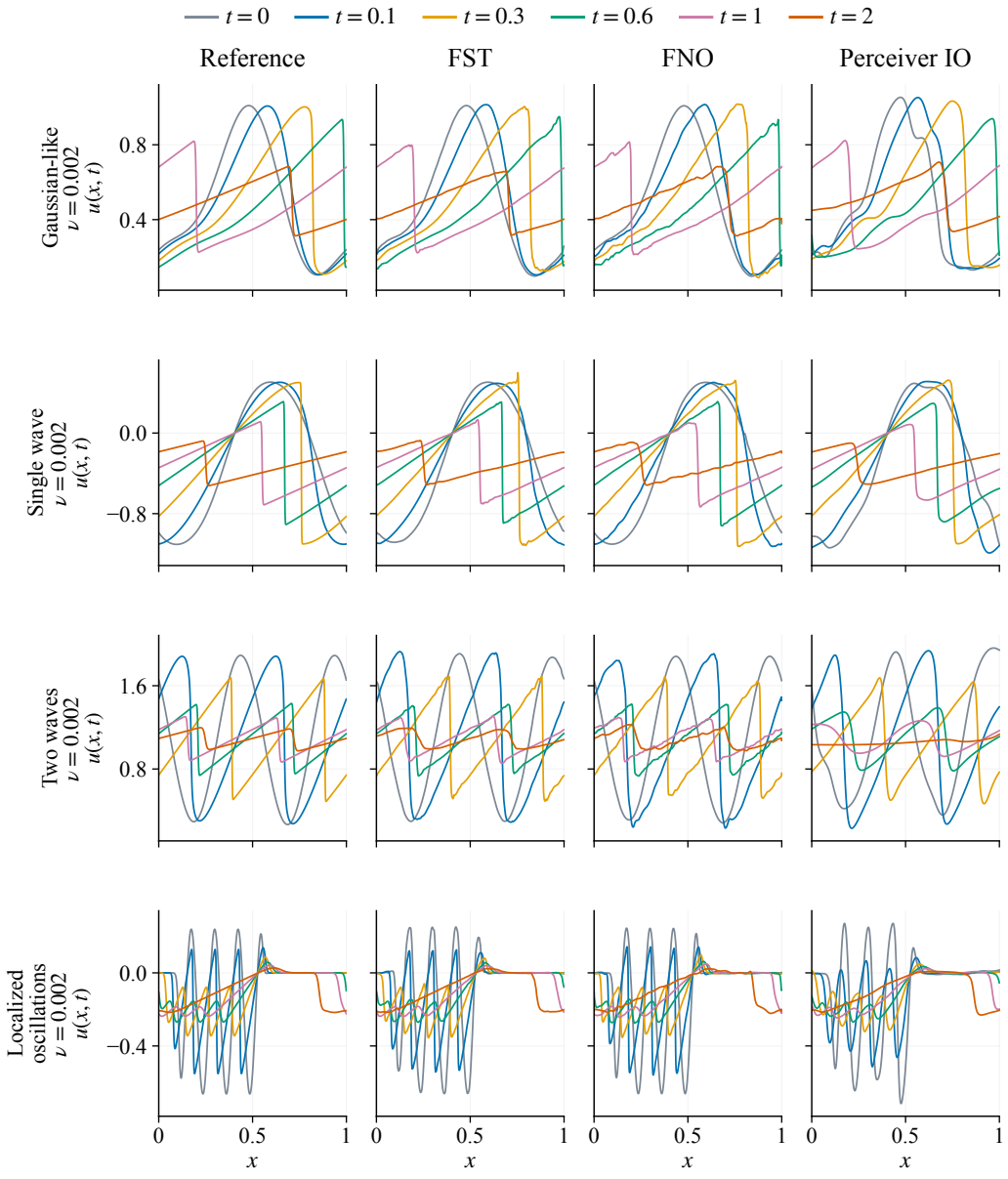

Figure 9: Burgers solution profiles for different initial conditions at fixed viscosity $\nu = 0.002$. From top to bottom, the initial profiles are Gaussian-like, single wave, two waves, and localized oscillations. Columns show the reference solution and predictions from FST, FNO, and Perceiver IO. Colors indicate $t \in \{0, 0.1, 0.3, 0.6, 1, 2\}$; axis limits are shared across columns within each row. The examples contrast broad spatial features, repeated wave structures, and spatially concentrated oscillations.

C DARCY EXPERIMENT DETAILS

We use the NeuralOperator Darcy dataset (NeuralOperator Team, 2021). Table 20 summarizes the data and training settings. We draw the training and validation splits from the 5,000-sample training file using seed 2026092001 and use all 1,000 samples in the test file. Input and output normalization statistics come from the training split.

Table 20: Darcy data and training settings.

Split	Samples	Grid	Precision
Training	2,000	128×128	BF16
Validation	100	100×100	FP32
Test	1,000	128×128	FP32

Shared setting	Value
Optimizer	Adam
Adam ϵ	10^{-8}
Weight decay	0
Gradient clipping	1
Batch size	48
Warmup (steps)	1,000
Validation interval (steps)	2,000

Model	Selected learning rate
FST	0.001
FNO	0.005
Perceiver IO	0.0003
Transolver	0.001
DeepONet	0.001

FST uses the architecture settings in Table 9. Its length scales follow Equation 10, with input-predicted adjustments along the two spatial axes.

Perceiver IO and Transolver use the widths, depths, and attention settings in Table 13. Darcy FNO uses four layers, width 50, and 16×16 Fourier modes. DeepONet uses branch widths $16384 \to 160 \to 256 \to 256 \to 256$ and trunk widths $2 \to 256 \to 256 \to 256 \to 256$, with ReLU activations and Glorot-normal initialization.

FNO and Transolver process the coefficient and solution fields at the same grid locations. DeepONet takes the flattened coefficient field as its branch input and query coordinates as its trunk input.

The loss is the mean squared per-solution relative L_2 error in original output units. FNO uses FP32 Fourier transforms during training.

For each model, we compare four learning rates over 4,000 steps, select the rate with the lowest mean of the last three validation errors, and train for an additional 8,000 steps to confirm the learning rate before training from scratch with seed 2026091311.

During final training, we smooth validation errors over three checks and halve the learning rate when the best smoothed error improves by less than 1% over 4,000 steps, down to 1% of the initial learning rate. At the minimum learning rate, training stops when improvement remains below 0.5% over 12,000 steps.

Validation uses 100 approximately evenly spaced indices on the original grid along each axis. We select the checkpoint with the lowest unsmoothed validation error for testing. Test error is computed on the full 128×128 grid. At each refinement step, FST draws 256 anchor indices without replacement and 256 scrambled Sobol points. At the start of each evaluation batch, it resets the anchor-selection seed to 2026391325 and the Sobol seed to 2026491325. The final anchor features serve all output queries.

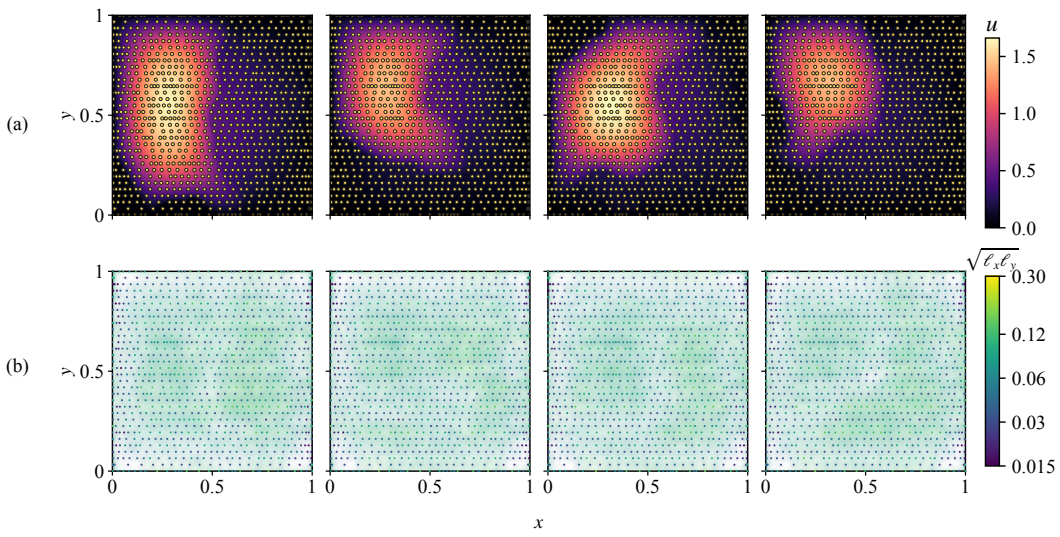

Figure 10: Darcy anchor coordinates and decoding RBF length scales for four test samples. (a) Anchors over reference solutions. (b) Soft ellipses with horizontal and vertical scales proportional to ℓ_x and ℓ_y, displayed at $0.35\times$ scale; color indicates $\sqrt{\ell_x \ell_y}$.

D IMAGENET-1K EXPERIMENT DETAILS

FST and ViT train from scratch with the same data and optimization settings for the comparison in Section 4.5.

D.1 DATASET AND DATA PREPROCESSING

Table 21: ImageNet-1K and data preprocessing.

Setting	Value
Training / validation images	1,281,167 / 50,000
Training crop	Random resized, 224×224
Training augmentations	Horizontal flips, 3-Augment (Touvron et al., 2022)
Color jitter	0.3
Random erasing probability	0.25
Validation resize (short side)	256
Validation center crop	224×224
Validation augmentation	None
Non-overlapping patches	196 of 16×16

D.2 MODEL CONFIGURATIONS

FST maps each patch to an observation token with a residual convolutional encoder (Table 22). The resulting tokens enter the backbone specified in Table 9.

Table 22: FST patch encoder, in execution order. Spatial convolutions use 3×3 kernels and the encoder uses GELU activations.

Stage	Operation	Setting
1	Convolution	64 channels
2	Residual blocks	2 blocks, 2 convolutions each
3	Pointwise projection	Width 200
4	Average pooling	Within each patch
5	Add positional embedding	Fourier embeddings of patch bounds + MLP
6	Residual feed-forward network	-

Patch observations. Figure 11 shows the patch observation $(\mathbf{c}_i, \mathbf{h}_i)$: a feature vector paired with its center coordinate. Patch bounds are normalized to $[0, 1]^2$.

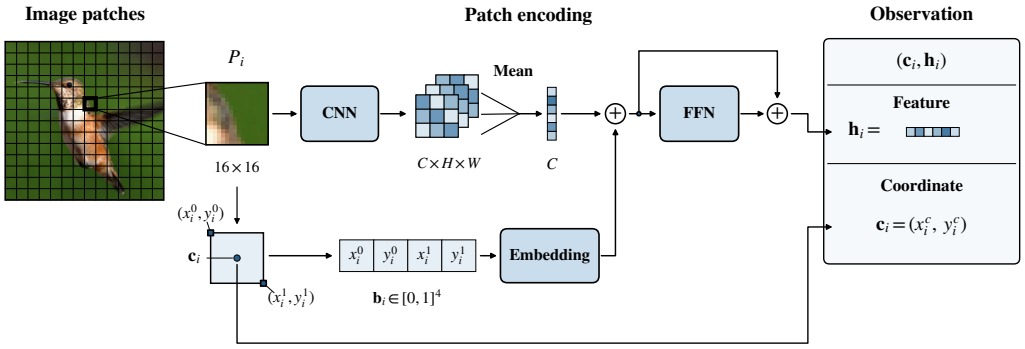

Figure 11: ImageNet patch observations. A CNN encodes each 16×16 patch. Spatial pooling, a patch-bound embedding, and a residual FFN produce the feature $\mathbf{h}_i$, paired with the patch center $\mathbf{c}_i = ((x_i^0 + x_i^1)/2, (y_i^0 + y_i^1)/2)$.

Attention bias. When anchors attend to patch features at the start of each refinement step, we add

$$B_{ji} = -\frac{\|\mathbf{a}_j - \mathbf{c}_i\|^2 + (w_i^2 + h_i^2)/12}{2\sigma_{\text{in}}^2} \tag{11}$$

to the query-key logit in every attention head for anchor j and patch i, where w_i, h_i are the normalized patch width and height, and σ_{in} is a shared learned scalar, initialized to 0.20 and bounded in $[0.03, 0.60]$. The numerator is the expected squared distance from the anchor to a uniformly sampled point within the patch. Because all patches have the same size, the size term adds the same constant to every logit and leaves the softmax weights unchanged. Attention weights depend jointly on this distance penalty and the learned query-key similarity.

Evaluation points. FST uses 512 fixed Sobol points in the image domain, independent of the patch grid.

Classification head. We apply layer normalization to each final anchor feature, average over anchors, and use a linear classifier to predict the class logits.

ViT baseline. ViT uses a linear patch projection and the transformer configuration in Table 23.

Table 23: ViT baseline configuration.

Setting	Value
Patch projection	Linear, 16×16 patches
Feature width	176
Transformer layers	10
Attention heads	4
Feed-forward hidden width	423
Trainable parameters	3,095,614

CNN-ViT. FST and CNN-ViT use the same residual convolutional patch encoder (stages 1-4 in Table 22). CNN-ViT projects its output from width 200 to 176 and retains ViT's transformer blocks, positional embeddings, and classifier. It has 3,158,150 parameters.

D.3 Training and Evaluation

We train all models to convergence using tuned hyperparameters. Training stops when the best validation error improves by less than 1% over the preceding 50,000 steps. The selected configuration and the learning-rate schedule until the convergence step are listed below.

FST and ViT use the optimizer settings in Table 24 and the staged learning-rate schedule in Table 25. FST's coordinate predictor uses one hundredth of the main learning rate throughout training. CNN-ViT uses the same data preprocessing and optimizer settings. Its learning rate warms up to 10^{-3} over 25,023 steps, then drops to 5×10^{-5} and follows a cosine decay toward 2.5×10^{-6} at step 500,000.

Table 24: Shared ImageNet-1K optimization settings.

Setting	Value
Optimizer	AdamW
Weight decay	0.05
Effective batch size	256
Computation precision	BF16
Gradient norm clipping	1.0
Label smoothing	0.1
Base seed	20260820

Table 25: Learning-rate schedule for FST and ViT.

Stage	Start	End	Learning rate
Warmup	0	25,023	Peak 10^{-3}
Cosine decay	25,023	500,456	$10^{-3} \rightarrow 5 \times 10^{-5}$
Cosine tail	500,456	550,456	$5 \times 10^{-5} \rightarrow 0$
Constant learning rate	550,456	600,456	2.5×10^{-6}

D.4 Additional Anchor Layouts

Figure 12 shows anchor layouts and RBF length scales for eight validation images. The layouts vary across inputs.

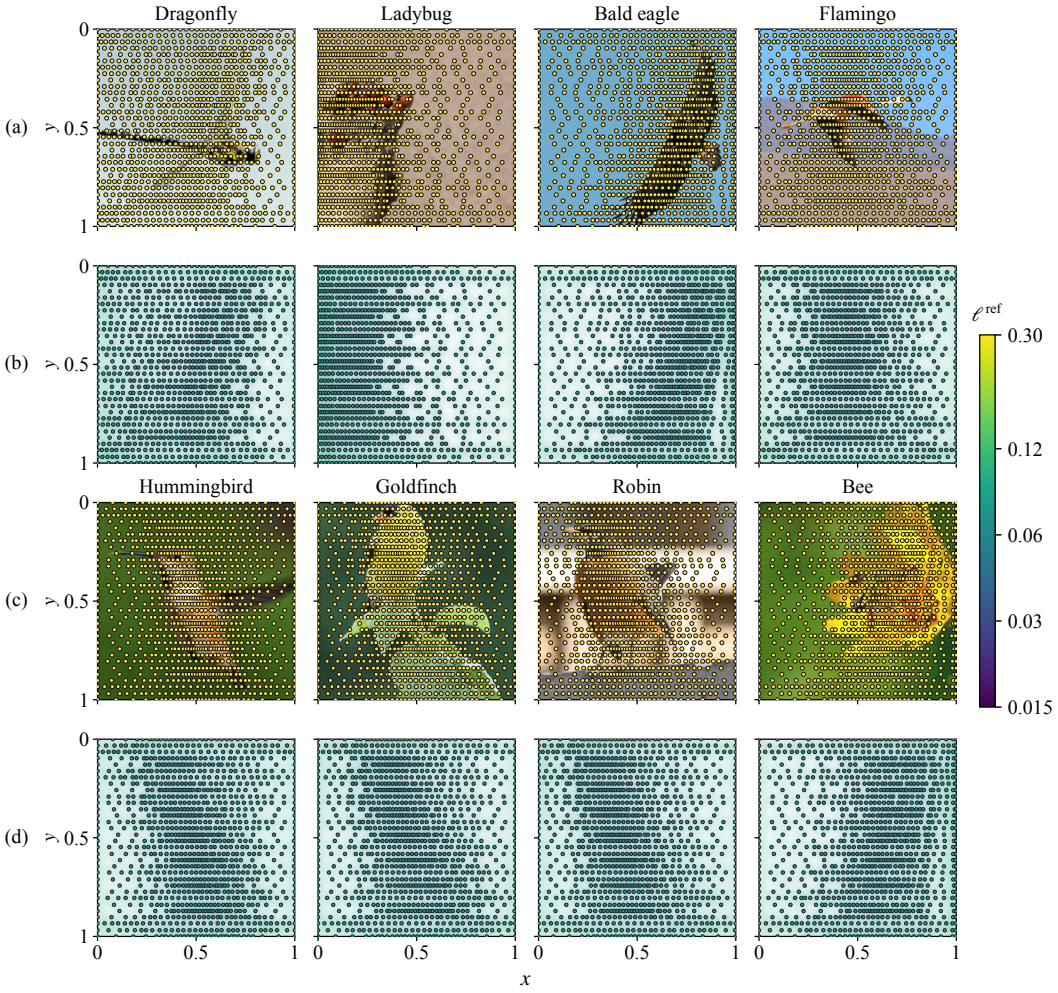

Figure 12: Anchor coordinates and RBF length scales on ImageNet-1K validation images. (a, c) All 1,024 anchors overlaid on the images. (b, d) RBF length scales ℓ^{ref}, shown by color and soft circles displayed at $0.35\times$ scale. Horizontal coordinates adapt to each input; vertical rows are fixed.