COMPACT CONDITIONAL NEURAL FIELDS FOR PDE SURROGATES WITH DEPTH-ADAPTED SHARED OPERATORS

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning PDE surrogates requires compact models that retain both global sample information and spatially localized context. We introduce a compact conditional neural field architecture in which a task encoder produces distributed query context together with a pooled sample condition, and a coordinate-conditioned decoder reconstructs the physical field. To reduce parameter storage without removing iterative refinement, repeated transformations share a base operator while using lightweight depth-specific low-rank adaptations. This design preserves node- or grid-level information paths while allowing a compact global condition to modulate the field decoder. Across standard PDE benchmarks, the resulting models achieve competitive accuracy with substantially fewer parameters: 0.00572 relative L_2 on Elasticity using 0.624M parameters, and 0.00523, 0.00338, and 0.00548 on Airfoil, Pipe, and Darcy using fewer than 0.875M parameters each. Relative to published Transolver-profile parameter counts, our models reduce parameter storage by approximately 54%–84%, while maintaining errors close to strong published baselines under the reported protocols. Ablations show that compact global conditioning is most valuable when distributed communication is restricted, whereas full token or convolutional context can already carry substantial sample-specific information. The results support a parameter-efficient conditional-field route for PDE surrogate modeling that avoids forcing the solution through a single global bottleneck.

1 INTRODUCTION

Learning a PDE surrogate requires both communication across the input domain and reconstruction of a spatially varying output. These requirements are easy to conflate with model size. Increasing communication depth often introduces additional parameter matrices, while reducing the state to a small pooled vector can discard information needed for localized predictions. A compact parameterization need not impose such a global information bottleneck.

This work examines a narrower design question: *can a conditional field predictor reuse its transformations across depth while preserving the distributed context needed for accurate reconstruction?* We build an encoder that produces a feature at each output location and a pooled representation of the sample. A coordinate-conditioned decoder uses both signals. Its repeated blocks share their main weights but have small, step-specific low-rank updates. For the graph encoder, the same construction is used inside local message passing and node–token communication. For structured grids, a multiscale convolutional encoder supplies the context, and sharing is applied to the decoder. Models are trained separately for each dataset.

Unlike a classical low-rank reconstruction, our output is a nonlinear function of coordinates, distributed features, and a global condition. Spatial context bypasses the pooled vector: this is not a strict reduced-order model and uses no POD teacher. Residual updates refer to neural blocks, not a separately supervised correction to a POD field.

The simplified-input Elasticity model obtains 0.005723 validation relative L_2 with $624{,}068$ parameters; grid variants use approximately 0.874 million parameters. These operating points motivate a controlled accuracy–size study, not a superiority claim under unmatched benchmark protocols.

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.

Replacing sample-dependent z with a learned constant does not degrade the token-enabled Elasticity model, but does degrade a controlled token-free model. This contrast motivates separating the value of pooled conditioning from the communication already carried by distributed context.

We organize the study around three contributions with distinct evidence requirements:

1. A compact conditional-field architecture that combines distributed context with depth-adapted shared transformations, without POD supervision.
2. An explicit separation of communication and parameterization controls: sample versus constant conditioning, token exchange and local depth, and shared versus adapted versus step-independent linear maps.
3. An empirical account of accuracy, parameter storage, and measured cost, including the regimes where sample conditioning does not improve accuracy.

2 Related Work

Operator learning and coordinate-conditioned fields. DeepONet uses separate branch and trunk representations for operator prediction (Lu et al., 2019), whereas Fourier neural operators perform learned transformations in a spectral representation (Li et al., 2021). CORAL uses coordinate-based representations for operator learning (Serrano et al., 2023). Our decoder belongs to this broad family of conditioned field representations; coordinate conditioning itself is not a novelty claim. Unlike a decoder driven only by a sample code, ours also consumes spatially distributed context.

Communication through latent tokens. Transolver aggregates mesh features into learned slices and computes attention over the resulting tokens (Wu et al., 2024). Our graph variant uses learned node aggregation followed by node-to-token attention to return context to nodes. These operations are related latent-communication mechanisms, but are not identical implementations of Physics-Attention. A final pooled vector and a set of communicating tokens also have different roles. Their respective ablation outcomes must not be conflated.

Sharing and conditioning. Universal Transformers reuse transformations across computational depth (Dehghani et al., 2019). LoRA parameterizes weight updates with low-rank factors (Hu et al., 2022); Relaxed Recursive Transformers combine recursive sharing and depth-specific LoRA (Bae et al., 2025). The latter is a direct architectural precedent, so we do not claim that sharing plus low-rank specialization is new. We study its use in PDE field reconstruction, including message, token, and decoder maps, and train the base and updates jointly from scratch. Decoder conditioning follows the feature-wise modulation principle of FiLM (Perez et al., 2018). The intended contribution is the particular field-prediction design and its demonstrated tradeoffs, not ownership of these established building blocks.

3 Method

3.1 Prediction interface and information paths

For sample s, let $\mathcal{X}_s = \{(x_{s,i}, a_{s,i})\}_{i=1}^{N_s}$ contain coordinates and available input attributes, with graph or structured-grid connectivity $\mathcal{T}_s$. Here s indexes samples, i indexes their N_s nodes, $x_{s,i} \in \mathbb{R}^{d_x}$ is a spatial coordinate, and $a_{s,i} \in \mathbb{R}^{d_a}$ contains the task's observed attributes. The target $u_s(x_{s,i}) \in \mathbb{R}^{d_u}$ has d_u physical output channels. The attributes may include an input coefficient field, boundary data, or input history, depending on the physical task. They do not include the target solution. Reference coordinates $\bar{x}_i$, when used, are part of the adapter specification rather than assumed universally available. An encoder produces

$$(H_s, z_s) = E_\eta(\mathcal{X}_s, \mathcal{T}_s), \qquad H_s \in \mathbb{R}^{N_s \times d_h}, \quad z_s \in \mathbb{R}^{d_z}. \tag{1}$$

At an encoded query node, the prediction is

$$\widehat{u}_s(x_{s,i}) = D_\theta(\psi_{s,i}, h_{s,i}, z_s), \tag{2}$$

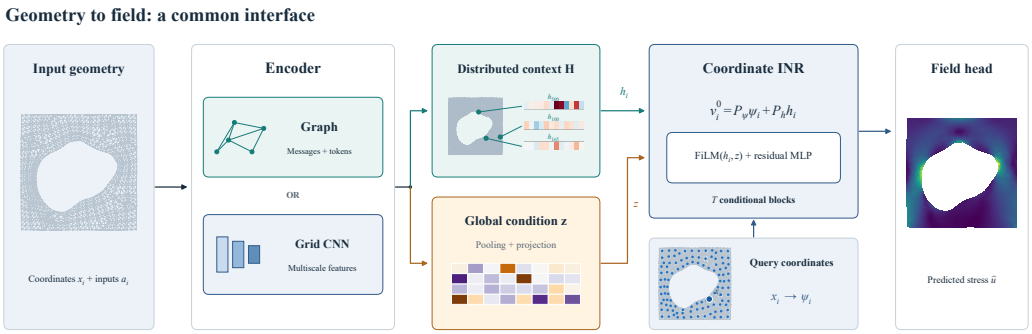

Figure 1: Common prediction architecture. Either input adapter produces distributed context H and pooled condition z; H bypasses z, and both condition the coordinate decoder. Reference geometry, when used, is specified by the task protocol. The decoder produces one field, not a sum of separately supervised global and residual fields. Insets show a fixed validation example (sample 1000, epoch-443 adapted checkpoint). Node-linked feature strips show individual rows of H; each query uses its corresponding h_i. The H and z colors encode features, not physical fields; the output is a real predicted stress field. Display triangulation is illustrative, not the reference kNN computation graph.

Here η and θ are the trainable encoder and decoder parameters, $h_{s,i} \in \mathbb{R}^{d_h}$ is row i of H_s, and d_z is the pooled-code width. The vector $\psi_{s,i} \in \mathbb{R}^{d_\psi}$ contains the configured coordinate encoding and allowed query attributes; it contains no target-field values. The reference-based implementations encode both current and reference coordinates. H_s is a *distributed context*, not necessarily a purely local feature: token exchange and multiscale convolution can already give it nonlocal information.

The interface does not by itself define $h_s(q)$ at a previously unseen query coordinate q. Such evaluation requires a specified context-interpolation or query-encoding procedure. We therefore evaluate the native query sets here and make no resolution-transfer or arbitrary-mesh claim based solely on the presence of coordinates in Eq. (2).

3.2 Depth-adapted shared linear maps

Each repeated affine map is parameterized as

$$L_t(v) = Wv + b + \frac{\alpha}{r} U_t V_t v, \quad V_t \in \mathbb{R}^{r \times d_{\text{in}}}, \quad U_t \in \mathbb{R}^{d_{\text{out}} \times r}. \tag{3}$$

For an input $v \in \mathbb{R}^{d_{\text{in}}}$, the base matrix $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ and bias $b \in \mathbb{R}^{d_{\text{out}}}$ are shared over steps $t = 0, \ldots, T-1$. The positive integer r is the adapter rank, α its scaling constant, and T the number of applications of this particular repeated map. The low-rank factors specialize each step. All factors and the base are trained jointly. The implementation initializes $U_t = 0$ and randomly initializes V_t, so all steps begin with the same affine function. Our principal configurations use $r = 4$ and $\alpha = 4$. This is a parameter-space factorization, not a rank-r decomposition of the predicted field.

Ignoring bias, one adapted map stores

$$P_{\text{adapted}} = d_{\text{out}} d_{\text{in}} + Tr(d_{\text{in}} + d_{\text{out}}), \qquad P_{\text{independent}} = T d_{\text{out}} d_{\text{in}}. \tag{4}$$

Thus its matrix-parameter count is smaller when $Tr(d_{\text{in}} + d_{\text{out}}) < (T-1) d_{\text{out}} d_{\text{in}}$. This accounting excludes other network components and proves no accuracy or wall-time improvement. The base map is still evaluated at every step, with additional low-rank operations. In particular, parameter sharing does not remove the sequential computation associated with depth.

3.3 Graph and structured-grid encoders

Graph adapter. Node features are initialized from coordinate encodings and permitted attributes. A message step forms gated messages from source and destination features and encoded edges, averages incoming messages, and applies a residual node update with LayerNorm. Reference-simple

Elasticity retains reference kNN connectivity and current/reference offsets and lengths. It masks the seven engineered node channels and three derived edge channels; it does not remove reference geometry or node correspondence.

When token exchange is enabled, let $H \in \mathbb{R}^{N\times d_h}$ denote the features after the local message update of a single sample (the sample index is suppressed). With m tokens and $\overline{H} = \mathrm{LN}(H)$, the implemented aggregation and return path are

$$A_{ki} = \mathrm{softmax}_i\left([L_t^{\mathrm{assign}}(\overline{h}_i)]_k\right), \qquad S = AH, \tag{5}$$

$$B = \mathrm{softmax}_k\left(\frac{\mathcal{Q}_t(\overline{H})\mathcal{K}_t(S)^\top}{\sqrt{d_h}}\right), \qquad M = B\mathcal{V}_t(S). \tag{6}$$

Here $k = 1, \ldots, m$ indexes tokens, $A \in \mathbb{R}^{m\times N}$ aggregates nodes, $S \in \mathbb{R}^{m\times d_h}$ contains token features, $B \in \mathbb{R}^{N\times m}$ returns information, and $M \in \mathbb{R}^{N\times d_h}$ is the node message. L_t^{assign} maps d_h features to m logits; $\mathcal{Q}_t, \mathcal{K}_t, \mathcal{V}_t$ are learned d_h-to-d_h query, key, and value maps, distinct from the low-rank matrix factors. A residual map sends M back to nodes. Assignments are normalized across nodes for each token; return attention is normalized across tokens for each node. There is no token–token self-attention in this implementation. For fixed m and width, aggregation and return attention scale linearly with N, although dense projections and graph operations remain additional costs. The message and token projections use Eq. (3). Learned step gates interpolate between the incoming and updated features:

$$H^{t+1} = H^t + 2\,\mathrm{sigmoid}(\rho_t^{\mathrm{enc}})(\widetilde{H}^{t+1} - H^t). \tag{7}$$

Here $\widetilde{H}^{t+1}$ is the candidate block output and ρ_t^{enc} is a learned scalar gate, for $0 \le t < T_e$. The default Elasticity encoder has width 96, $T_e = 4$ steps, and $m = 8$ tokens. Figure 2 visualizes the learned aggregation on a fixed validation geometry. Unlike a hard partition, token supports can overlap. Four learned attention-pooling summaries, the nodewise mean, and the nodewise maximum are concatenated and projected to $z \in \mathbb{R}^{32}$.

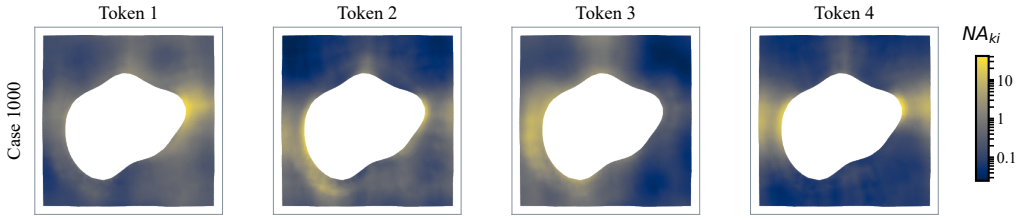

Figure 2: Learned node-to-token aggregation, Elasticity case 1000, encoder step 4 of the epoch-443 adapted checkpoint. The first four token indices are shown without selection by appearance. Colors give NA_{ki} on a shared log scale; 1 denotes uniform aggregation. These are feature-aggregation weights, not stress fields or POD modes. All eight tokens on two cases appear in Figure 4.

Grid adapter. On structured grids, a residual convolutional encoder uses channel widths $32, 48, 64, 96$, average-pooling downsampling, and bilinear upsampling with skip concatenation. A final projection produces 96 context channels on the native grid. The pooled representation is read from the coarsest features. Convolution acts in grid-index space; curvilinear geometry is conveyed by coordinate features, not by treating index distance as physical distance. The CNN has its own untied convolutional weights. For this variant the proposed sharing control concerns the decoder, not the entire encoder.

3.4 Conditioned iterative decoder

Let $v_i^0 = P_\psi(\psi_i) + P_h(h_i) \in \mathbb{R}^{d_v}$ and $c_i = \mathrm{LN}([z_s, h_i]) \in \mathbb{R}^{d_z+d_h}$. P_ψ and P_h are learned input projections, and $[\cdot,\cdot]$ denotes feature concatenation. At decoder step $0 \le t < T_d$,

$$[\gamma_i^t, \beta_i^t] = L_t^{\mathrm{mod}}(c_i), \tag{8}$$

$$\widetilde{v}_i^t = \mathrm{LN}(v_i^t) \odot (1 + 0.1\tanh\gamma_i^t) + 0.1\beta_i^t, \tag{9}$$

$$v_i^{t+1} = v_i^t + g_t L_t^{\mathrm{out}}\big(\mathrm{SiLU}(L_t^{\mathrm{in}}(\widetilde{v}_i^t))\big), \qquad g_t = 2\,\mathrm{sigmoid}(\rho_t^{\mathrm{dec}}). \tag{10}$$

The vectors $\gamma_i^t, \beta_i^t \in \mathbb{R}^{d_v}$ provide featurewise scaling and shifting, $\odot$ denotes elementwise multiplication, and ρ_t^{dec} is a learned scalar independent of the encoder gates. LN is LayerNorm, $\mathrm{sigmoid}(a) = (1 + e^{-a})^{-1}$, and $\mathrm{SiLU}(a) = a\,\mathrm{sigmoid}(a)$, applied elementwise. The default decoder has width $d_v = 160$ and $T_d = 5$ steps. The modulation and update maps share their bases and have separate low-rank updates per step. The output head maps $\mathrm{LN}(v_i^{T_d})$ through SiLU and a learned affine map to d_u normalized output channels; inverse target normalization gives the physical prediction in Eq. (2). The formula omits training dropout for clarity. There is a single final field prediction, not separately supervised global and residual output fields.

3.5 Training and controlled variants

For the static runs in Table 5, the objective combines physical-space per-sample relative error and normalized field MSE:

$$\mathcal{L} = \frac{1}{|\mathcal{B}|} \sum_{s \in \mathcal{B}} \left[\frac{\|\widehat{u}_s - u_s\|_2}{\max(\|u_s\|_2, \epsilon)} + \frac{\lambda}{N_s d_u} \|\mathcal{N}(\widehat{u}_s) - \mathcal{N}(u_s)\|_2^2 \right]. \tag{11}$$

Here $\mathcal{B}$ is a minibatch, $\lambda = 0.1$, $\epsilon = 10^{-12}$, and each norm acts on all $N_s d_u$ flattened field entries of one sample. The affine target standardizer $\mathcal{N}$ uses only training-set statistics; its centering and scale are frozen for validation and testing. The reported relative error is computed after inverse normalization. No POD coefficient target, basis fit, or spectral residual constraint is used. Other physical tasks have separately specified objectives; their losses are not numerically interchangeable with this metric.

Three linear-map controls preserve width and step count: a shared base without updates, a shared base with Eq. (3), and independent full linear maps per step. The independent control duplicates only the selected affine maps, not every normalization or auxiliary module. It is an equal-depth, equal-width comparison, not a parameter-matched comparison. Separately, sample-dependent z_s can be replaced with a trainable constant z_0. Both conditions are trained from scratch; this avoids interpreting an out-of-distribution test-time removal as the performance of a retrained model.

4 Experiments

4.1 Evaluation scope

We report validation results for the PDE field-prediction tasks, frozen holdout evaluations for design tasks, and published reference results for context. Table 1 places all six standard tasks in a common overview; Table 3 reports the completed ShapeNetCar holdout. Literature baselines are attributed to their sources, and values are interpreted under their stated protocols. The static-task runs use seed 42, a 500-epoch budget, batch size 4, AdamW with initial learning rate 5×10^{-4} and weight decay 10^{-5}, cosine decay to 10^{-5}, and gradient clipping at 1. Checkpoints are selected using validation error. These runs use FP32 without AMP or TF32. The sharing study reports the completed seed-42 controls.

The accuracy overview and parameter comparison in Table 2 establish sub-million-parameter operating points. The checkpoint ledger is provided separately in Appendix Table 5. Airfoil and Pipe use 800 training cases plus 200 validation cases. Darcy's training and validation rows come from different source files, so we retain the source-file qualified split definition. Published results provide reference context for each task under their own public protocols.

4.2 Parameter and computational efficiency

Reduced parameter storage. Table 2 makes the storage advantage explicit: our standard-task models use 0.624–0.874 million parameters, 54.2–83.6% fewer than the Transolver configurations profiled by LNO. They are also smaller than the reported LNO configurations on five of six tasks; Darcy is the exception (0.874 versus 0.760 million). This is a configuration-specific parameter comparison, not a claim of matched accuracy or the smallest possible model. The AirfRANS and ShapeNetCar models use 0.617 and 0.619 million parameters.

Table 1: Six-task relative L_2 overview (lower is better). Published rows are from Transolver Table 2 and the LNO/Transolver++ main tables. Our row reports the validation protocol used for our compact conditional-field runs. †: trajectory-level NS evaluation. ‡: Plasticity uses a protocol-specific trajectory metric and is reported as auxiliary evidence. Values are reported under their respective published or stated protocols.

Method	Elasticity	Plasticity	Airfoil	Pipe	N–S	Darcy
FNO	–	–	–	–	0.1556	0.0108
WMT	0.0359	0.0076	0.0075	0.0077	0.1541	0.0082
U-FNO	0.0239	0.0039	0.0269	0.0056	0.2231	0.0183
Geo-FNO	0.0229	0.0074	0.0138	0.0067	0.1556	0.0108
U-NO	0.0258	0.0034	0.0078	0.0100	0.1713	0.0113
F-FNO	0.0263	0.0047	0.0078	0.0070	0.2322	0.0077
LSM	0.0218	0.0025	0.0059	0.0050	0.1535	0.0065
Galerkin	0.0240	0.0120	0.0118	0.0098	0.1401	0.0084
HT-Net	–	0.0333	0.0065	0.0059	0.1847	0.0079
OFormer	0.0183	0.0017	0.0183	0.0168	0.1705	0.0124
GNOT	0.0086	0.0336	0.0076	0.0047	0.1380	0.0105
FactFormer	–	0.0312	0.0071	0.0060	0.1214	0.0109
ONO	0.0118	0.0048	0.0061	0.0052	0.1195	0.0076
Transolver (Wu et al., 2024)	0.0064	0.0012	0.0053	0.0033	0.0900	0.0057
LNO (Wang & Wang, 2024)	0.0052	0.0029	0.0051	0.0026	0.0845	0.0049
Transolver++ (Luo et al., 2025)	0.0052	0.0011	0.0048	0.0027	0.0719	0.0049
Ours (protocols above)	0.00572	0.00263‡	0.00523	0.00338	0.05370†	0.00548

Table 2: Parameter storage and forward computation. (A) Parameters in millions: published configurations profiled by LNO, Table 4 (Wang & Wang, 2024), versus our allocated models. Bold identifies the smallest count in each column. (B) Our native-mesh supported-kernel counts, batch one; multiply–add counts as two FLOPs. Counted operators and preprocessing conventions are specified in Appendix C.

(A) Cross-dataset parameter comparison

Model	Elasticity	Plasticity	Airfoil	Pipe	N–S	Darcy
Transolver (LNO profile)	1.91	1.91	1.91	1.91	5.33	1.91
LNO	1.42	1.36	1.36	1.36	5.08	**0.76**
Ours (unified)	**0.624**	**0.874**	**0.874**	**0.874**	**0.874**	0.874
Plasticity trajectory	–	1.422	–	–	–	–
Reduction vs. Transolver	67.3%	54.2%	54.3%	54.3%	83.6%	54.3%

(B) Our checkpoint-level computation and storage

Dataset	Query nodes	FP32 weights (MiB)	Counted GFLOPs
Elasticity	972	2.38	15.69
Airfoil	11,271	3.33	20.57
Pipe	16,641	3.33	30.36
Darcy	7,225	3.33	13.19

Computation and measured runtime. The native-query forward counts in panel B expose computation separately from storage. Reused weights are still applied at every depth and node; missing published FLOPs are not inferred from parameter counts. On an RTX 5090, at batch size one and 32,000 nodes, ShapeNetCar's GPU-resident FP32 forward pass takes 39–40 ms with about 1.70 GiB peak allocated memory. Warm-cache CPU preprocessing takes another 98–105 ms, excluding transfer. These measurements characterize the implementation used in this study. Timing and counting details are in Appendices C and C.1.

Table 3: ShapeNetCar official 100-case holdout. Literature rows are from Transolver, Table 3. Volume is velocity relative L_2 on non-surface nodes; surface is pressure relative L_2. e_D is mean relative drag-coefficient error and ρ_D is Spearman rank correlation. Our frozen checkpoint uses 710 training and 79 validation cases, while the published setup uses a 789-case fit.

Method	Velocity ↓	Pressure ↓	e_D ↓	ρ_D ↑
MLP	0.0512	0.1304	0.0307	0.9496
GraphSAGE	0.0461	0.1050	0.0270	0.9695
PointNet	0.0494	0.1104	0.0298	0.9583
Graph U-Net	0.0471	0.1102	0.0226	0.9725
MeshGraphNet	0.0354	0.0781	0.0168	0.9840
GNO	0.0383	0.0815	0.0172	0.9834
Galerkin	0.0339	0.0878	0.0179	0.9764
Geo-FNO	0.1670	0.2378	0.0664	0.8280
GNOT	0.0329	0.0798	0.0178	0.9833
GINO	0.0386	0.0810	0.0184	0.9826
3D-GeoCA	0.0319	0.0779	0.0159	0.9842
Transolver (Wu et al., 2024)	0.0207	0.0745	0.0103	0.9935
Ours (710 training cases)	0.02507	0.07683	0.01604	0.98883

4.3 Prediction fields and design-task holdouts

Figure 3 brings seven datasets into one display: four static validation examples, two design holdouts, and a protocol-specific Plasticity trajectory example. All use frozen checkpoints; the static cases correspond to Table 5. True and predicted fields share color limits. Absolute-error maps expose spatially concentrated discrepancies that can be obscured by an aggregate relative error: near the hole in Elasticity, along the Airfoil shock, and in localized regions of the Darcy solution. These examples are selected by split order and complement dataset-level statistics.

The Plasticity trajectory evaluation obtains 0.002629 four-channel relative L_2 over 80 cases, with 1.422M parameters and 900 training cases. It is reported as protocol-specific auxiliary evidence in Table 8; Figure 8 shows its time evolution.

The ShapeNetCar checkpoint is selected on 79 validation cases, frozen, and evaluated on all 100 official param0 holdout cases with frozen statistics. It has $618{,}945$ parameters. Its surface-pressure error is approximately 3.1% above the published Transolver result, while velocity and drag errors are 21.1% and 55.7% higher, respectively. Thus the present model is competitive with several published alternatives on surface pressure while leaving room for improved force prediction. Figure 3(e–f) shows both design-task predictions; the AirfRANS metric protocol is detailed in Appendix D. AirfRANS's completed 200-case evaluation gives 0.02134 velocity relative L_2 and 0.03818 mean relative lift error, but 2.65067 relative drag error. We report both force outcomes and list physical field metrics separately from the public implementation's normalized MSE.

4.4 Parameterization and communication ablations

Table 4 groups nine controls by protocol: weight reuse (A), token exchange and conditioning (B), and CNN conditioning (C).

When does sample conditioning help? Constant conditioning slightly improves validation error in both the token-enabled Elasticity and Airfoil CNN pairs. Sample-dependent pooling therefore provides no measured advantage when these alternative communication paths remain. Without tokens, constant z increases both training and validation error, consistent with a communication or optimization limitation rather than overfitting alone. Sample z reduces that error to 0.007983, but does not recover the token-enabled 0.0053 operating point. This supports a conditional role for z, not lossless replacement of token communication or an expressivity lower bound. Airfoil's constant-vector implementation allocates 32 additional parameters, included in the table.

Weight reuse and specialization. The adapted parameterization uses approximately 64.7% fewer parameters than the independent-map Elasticity control and 38.6% fewer for Darcy. These are stor-

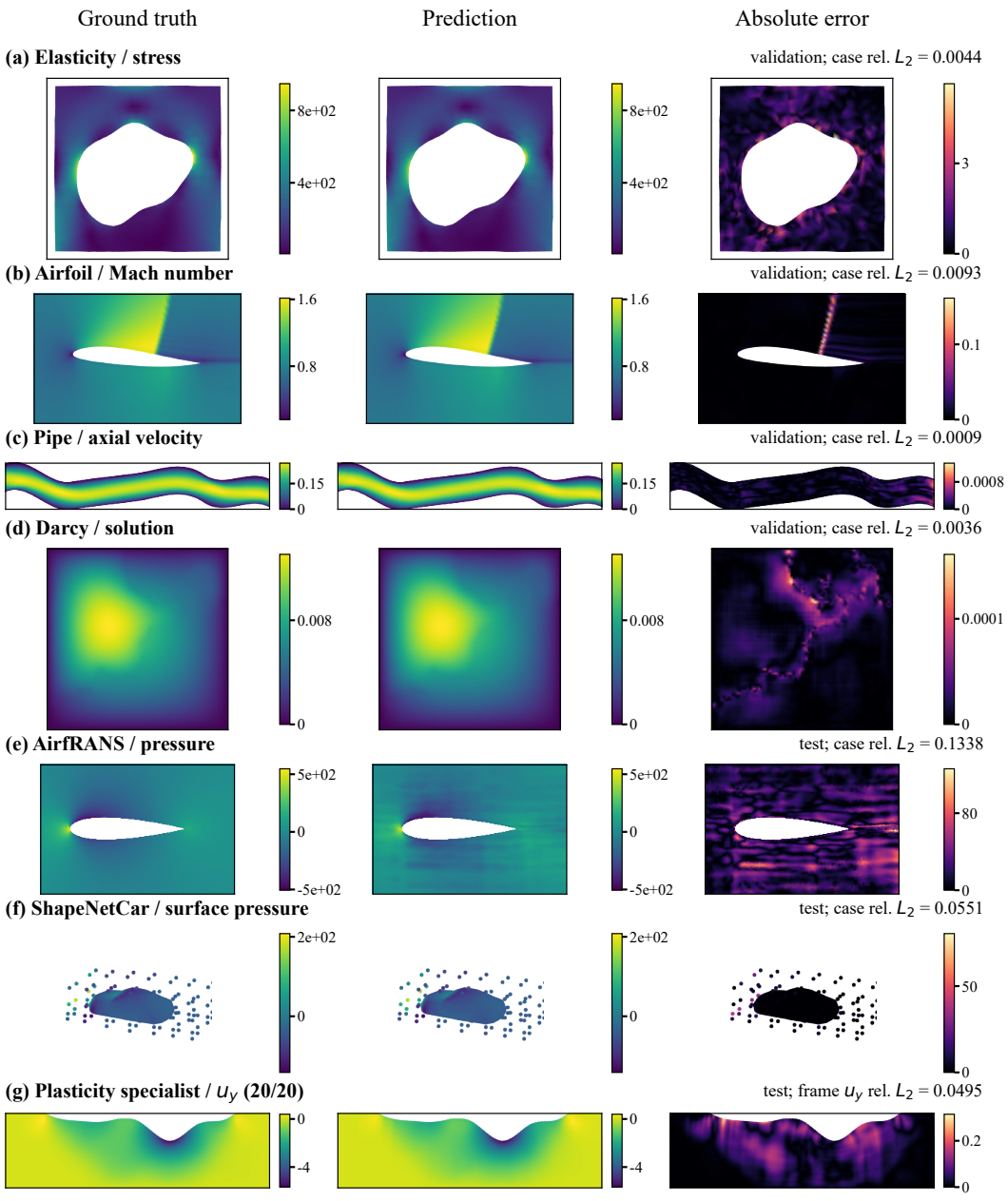

Figure 3: Ground truth, frozen-checkpoint predictions, and absolute errors on seven datasets. Each is the first example in its labeled split, not selected by error. Truth and prediction share a color scale; errors use separate scales. Airfoil/AirfRANS displays are cropped around the body; annotated case errors use the complete native target field. Holes are masked, Pipe retains its physical aspect ratio, and Car shows native surface nodes. Display interpolation does not modify evaluated nodal values. The Plasticity panel shows frame-20 vertical displacement from the protocol-specific trajectory evaluation; its annotated frame/channel error is distinct from the trajectory score.

age reductions at fixed depth and width. Appendix C.1 provides the initialization controls and full learning curves.

For the completed Elasticity seed (Figure 7), adaptation lowers error by approximately 2.5% compared with pure sharing, but uses 14.5% more parameters. Compared with independent maps, it

Table 4: Joint summary of parameterization and communication ablations, seed 42. Comparisons are within each labeled block and input protocol. "Indep." duplicates step-dependent affine maps; normalization remains shared. Token-disabled models retain allocated token parameters. Errors are validation-selected relative L_2, not test results or multi-seed means.

Variant	Local steps	Tokens	z	Maps	Params (M)	Rel. L_2
A. Elasticity, reference-simple: parameter reuse						
Shared base	4	8	Sample	Shared	0.54495	0.005907
Depth-adapted	4	8	Sample	Adapted	0.62407	0.005759
Independent	4	8	Sample	Indep.	1.76558	0.004613
B. Elasticity, legacy inputs: communication and conditioning						
Token-enabled	4	8	Sample	Adapted	0.62410	0.005345
Token-enabled	4	8	Constant	Adapted	0.62410	0.005258
Token-disabled	4	Off	Sample	Adapted	0.62410	0.007983
Token-disabled	4	Off	Constant	Adapted	0.62410	0.100291
C. Airfoil, geometry CNN: conditioning with multiscale context						
Geometry CNN	–	–	Sample	Adapted	0.87356	0.005231
Geometry CNN	–	–	Constant	Adapted	0.87359	0.005093

uses 64.7% fewer parameters with approximately 24.8% higher error. This provides a concrete size–accuracy tradeoff for the depth-adapted parameterization.

5 Discussion

What is unified? Unified variants use distributed context and pooled conditioning with a depth-adapted coordinate decoder. Graph/grid encoders, temporal inputs, and output channels remain task dependent; models are fitted separately. Reference-based variants assume node correspondence and CNN variants assume native grid ordering. This is an explicit adapter family, not one identical network or input budget across all tasks. Plasticity is included as protocol-specific trajectory evidence while the method description remains the same compact conditional-field template.

What does compactness mean? Sharing reduces duplicated weights, not sequential computation or distributed activations. Adapter rank r, code width d_z, token count, and mesh size are distinct quantities; none alone measures the solution's effective dimension.

Scope and extensions. The experiments focus on parameter-efficient field reconstruction under the dataset protocols reported in Section 4. Broader statistical sweeps, conservation diagnostics, and out-of-distribution geometry transfer are natural extensions. The current study emphasizes the accuracy–efficiency tradeoff of shared depth-adapted conditional fields.

6 Conclusion

We present a conditional neural-field design that reuses operators across depth while retaining distributed spatial context. The reported results demonstrate accurate sub-million-parameter configurations on several PDE datasets. The conditioning controls show that pooled sample information can be important when token exchange is absent and partly redundant when alternative communication paths remain. This supports evaluating parameter reuse and information flow as separate design choices for compact PDE surrogates.

AI use statement

Generative AI tools were used to aid and polish writing, and to assist with code editing during manuscript preparation. All scientific claims, experiments, and reported results were reviewed by the authors.

Reproducibility statement

The method equations specify the implemented information paths and shared parameterization. Appendix A records protocols, ablation definitions, and evaluation scope. Reported quantities are linked to saved configurations, split manifests, normalization statistics, and checkpoint records.

References

Sangmin Bae, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Seungyeon Kim, and Tal Schuster. Relaxed recursive transformers: Effective parameter sharing with layer-wise LoRA. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.20672.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019. URL https://arxiv.org/abs/1807.03819.

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://arxiv.org/abs/2106.09685.

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://arxiv.org/abs/2010.08895.

Lu Lu, Pengzhan Jin, and George Em Karniadakis. DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019. URL https://arxiv.org/abs/1910.03193.

Huakun Luo, Haixu Wu, Hang Zhou, Lanxiang Xing, Yichen Di, Jianmin Wang, and Mingsheng Long. Transolver++: An accurate neural solver for PDEs on million-scale geometries. In *International Conference on Machine Learning*, 2025. URL https://arxiv.org/abs/2502.02414.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018. URL https://arxiv.org/abs/1709.07871.

Louis Serrano, Lise Le Boudec, Armand Kassaï Koupaï, Thomas X Wang, Yuan Yin, Jean-Noël Vittaut, and Patrick Gallinari. Operator learning with neural fields: Tackling PDEs on general geometries. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2306.07266.

Tian Wang and Chuang Wang. Latent neural operator for solving forward and inverse PDE problems. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2406.03923.

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*, 2024. URL https://proceedings.mlr.press/v235/wu24r.html.

A Protocol and Evidence Scope

A.1 Inputs are part of the method

For reference-simple Elasticity, the model retains current and training-derived reference coordinates, their Fourier encodings, reference kNN topology, and current/reference edge offsets and lengths.

Seven legacy node-feature slots and three derived edge-feature slots are zeroed without changing module shapes or LayerNorm dimensions. The reduction is an input-information ablation, not removal of all preprocessing or correspondence assumptions. Geometric differences from a reference mesh are not ground-truth mechanical displacement measurements. They must not be described as supplied solution fields.

The Airfoil geometry CNN uses standardized current-minus-reference coordinate differences together with coordinate encodings. Reference and scaling statistics are fitted on the training samples. Pipe and Darcy use a fixed structured-grid adapter; Darcy supplies its spatial input coefficient field. The adapter retains padded feature slots to reuse the interface, and all allocated parameters count toward the reported size. These choices require input-matched baseline controls, even if the added quantities are deterministic functions of admissible coordinates.

A.2 Splits and evaluations

Table 5: Checkpoint and split ledger for the static-task results. Each row uses seed 42 and records the selected validation checkpoint.

Dataset / adapter	Train / val	Parameters	Best epoch	Val. rel. L_2
Elasticity / reference-simple graph	$1000/200$	$624{,}068$	494	0.005723
Airfoil / geometry CNN	$800/200$	$873{,}556$	460	0.005231
Pipe / grid CNN	$800/200$	$873{,}588$	499	0.003381
Darcy / grid CNN	$1000/200$	$873{,}588$	486	0.005481

Table 6: Training/validation splits and evaluation scope. Static validation, trajectory, and holdout protocols use their stated metrics.

Dataset	Train	Validation	Evaluation scope
Elasticity	1000	200	Validation-selected static field
Airfoil	800	200	Validation-selected static field
Pipe	800	200	Validation-selected static field
Darcy	1000	200	Validation; source-qualified split
Navier–Stokes	900	100	Free-rollout trajectory evaluation
Plasticity	800	100	Trajectory evaluation
AirfRANS	720	80	200-case holdout
ShapeNetCar	710	79	Completed 100-case test and drag evaluation

For Darcy, training uses rows $[0, 1000)$ from the first smooth-data source, validation uses rows $[200, 400)$ from the second source, and test uses rows $[0, 200)$ from that second source. Dataset identity includes the source file. All reported holdout evaluations use frozen input processing, normalization, and checkpoints. Validation and holdout quantities are labeled by protocol.

A.3 Ablation interpretation

The constant-z control learns one vector shared across all samples. It does not remove the input dependence of distributed context. Node features may already incorporate global information through tokens or the convolutional hierarchy. Consequently, an accurate constant-z model does not mean that a constant vector encodes multiple geometries; the geometries reach the decoder through other paths. Conversely, poor token-free constant-z performance does not prove that a particular latent size is information-theoretically minimal.

Token-disabled controls may retain unused token parameters in the allocated model. Such variants are communication ablations, not automatically compact deployment implementations. Report allocated and actually pruned parameter counts separately. The per-step-independent sharing control also retains shared normalization and non-step operators. A claim about fully independent blocks or parameter-matched superiority requires different controls.

A.4 Evaluation conventions

Published baseline results are attributed to their sources. Inputs, training splits, targets, and aggregation are reported so that cross-paper comparisons can be read with their protocol context. Timing uses isolated FP32 inference with GPU synchronization, with preprocessing reported separately. Repetitions on three cases characterize the timing measurements used in the cost figure.

A.5 Nearest methodological controls

The primary mechanistic comparison is shared versus depth-adapted versus independent linear maps, with fixed information, depth, width, and training budget. These seed-42 controls quantify the measured size–accuracy tradeoff for the shared depth-adapted parameterization.

B Additional Visualizations

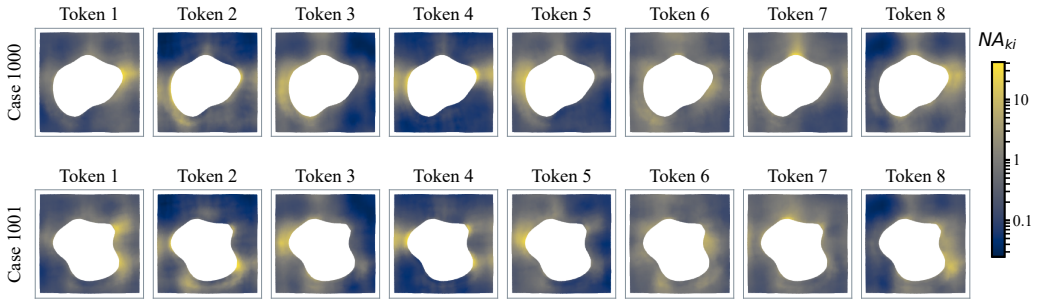

Figure 4: All eight node-to-token aggregation maps at encoder step 4, epoch-443 adapted checkpoint, on the first two validation cases. The shared log scale displays NA_{ki}, with uniform aggregation equal to 1. Token indices are unchanged across cases. No target labels or error-based selection are used; spatially overlapping supports are allowed.

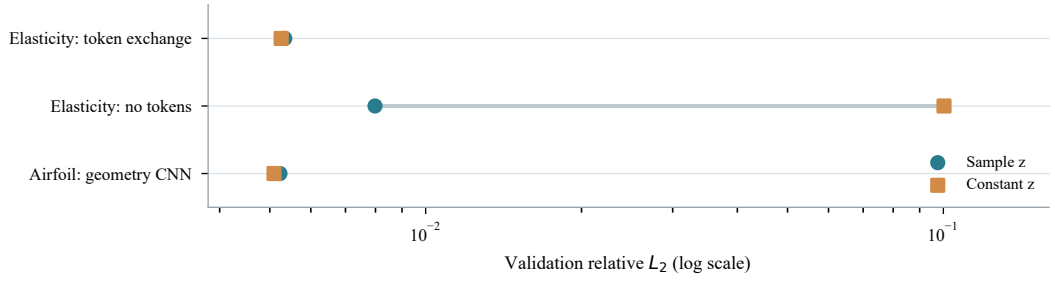

Figure 5: Retrained sample/constant-z controls, seed 42, validation-selected checkpoints. The two Elasticity pairs use legacy geometry features, whereas Airfoil uses the geometry CNN. Comparisons are within each pair, not across datasets. The horizontal axis is logarithmic. Sample conditioning helps strongly in the tested token-free setting but does not improve either token-enabled Elasticity or the Airfoil pair. No multi-seed uncertainty is available.

C Notation and Computational Accounting

Cost is not parameter count. For n input vectors, a shared dense map with a rank-r update performs approximately $2n[d_{\text{in}}d_{\text{out}} + r(d_{\text{in}} + d_{\text{out}})]$ floating-point operations, excluding biases and activations. The base matrix is stored once but applied at every refinement step. With E graph

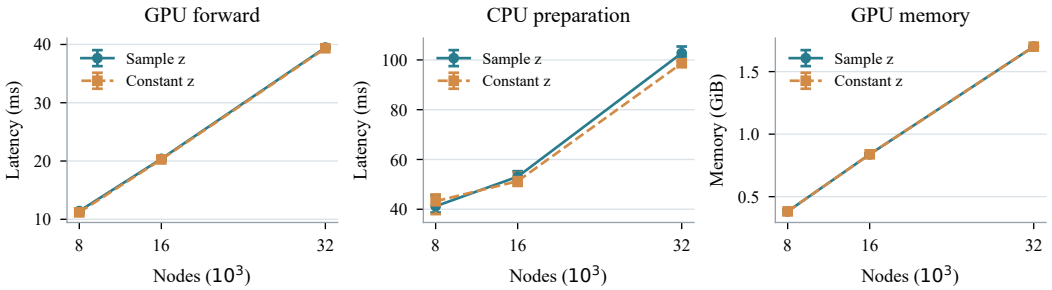

Figure 6: Isolated ShapeNetCar cost measurements on an RTX 5090, batch one, FP32. Points are means of three case-level measurements; bars show their minimum–maximum range, not confidence intervals. Each case-level latency is a median of repeated measurements. Forward timing uses GPU-resident inputs and excludes host-to-device transfer; CPU preprocessing is timed separately with warm caches. Both arms are unpruned, including pooled-z computation in the constant arm.

Table 7: Notation used in the method. Widths and ranks describe different objects.

Symbol	Definition
s, i, k, t	Sample, spatial node, token, and refinement-step indices.
N_s, d_x, d_a, d_u	Node count; coordinate, input-attribute, and output-channel dimensions.
$\mathcal{T}_s$	Admissible graph or native-grid connectivity for sample s.
$H_s, h_{s,i}, z_s$	Distributed context, its node row, and the pooled sample condition.
d_h, d_z, d_ψ, d_v	Context, pooled-code, query-encoding, and decoder-state widths.
$\psi_{s,i}, \bar{x}_i$	Query encoding and reference coordinate, where the protocol supplies one.
η, θ	Encoder and decoder trainable parameters.
T_e, T_d, m	Encoder depth, decoder depth, and token count.
$W, b, U_t, V_t, r, \alpha$	Shared affine map, per-step low-rank factors, rank, and scaling constant.
A, S, B, M	Node-to-token weights, token features, token-to-node weights, and returned messages.
γ, β, ρ, g	Featurewise modulation, learned gate logit, and its bounded step size.
$\mathcal{B}, \mathcal{N}, \lambda, \epsilon$	Minibatch, target standardizer, MSE weight, and denominator floor.

edges, encoder width d_h, N nodes, and m tokens, dense edge-message maps cost $O(T_e E d_h^2)$; token aggregation/return adds $O(T_e N m d_h)$, in addition to dense node/token projections. The grid encoder costs $O(\sum_\ell H_\ell W_\ell k_\ell^2 C_{\ell-1} C_\ell)$ across convolution layers ℓ with height H_ℓ, width W_ℓ, kernel size k_ℓ, and channels C_ℓ. For fixed widths, the coordinate decoder costs $O(T_d N[d_v^2 + (d_h + d_z)d_v])$ plus its rank-update operations. No quadratic node–node attention is introduced, but substantial linear-in-node work remains. FLOP counting excludes backward passes and must specify resolution, batch size, and supported operators.

Table 2 in the main results combines the published parameter profiles and our measured counts. The profiler counts matrix products, convolutions, and supported elementwise addition/multiplication; normalization, scatter, activations, softmax, and preprocessing are not fully counted. These are partial forward FLOPs, not total computational cost. FP32 weight storage counts four bytes per allocated parameter; it excludes buffers and activation memory. Our Darcy model is larger than the cited LNO configuration, so the parameter advantage is not universal across baselines.

C.1 Sharing controls and timing checks

All three sharing controls are constructed from the same seeded model, and independent maps copy the shared base. Zero-initialized adaptation makes their initial functions agree. Unit tests verify replacement behavior, independent map gradients, and unchanged CNN parameters in Darcy. Normalizations and non-step modules are not duplicated. Darcy's allocated counts are 845,428, 873,588, and 1,422,068 for shared, adapted, and independent maps, respectively. The completed Elasticity conditioning pairs have train-evaluation errors of $0.003445/0.003567$ (sample/constant with tokens)

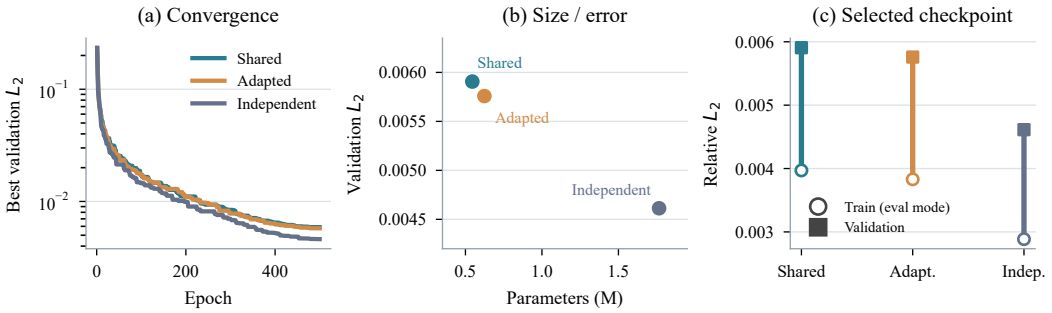

Figure 7: Elasticity sharing controls, seed 42, 500 epochs, reference-simple inputs. Left: cumulative minimum of observed validation error (no smoothing). Middle: parameter count versus selected-checkpoint validation error. Right: train-evaluation and validation errors at those checkpoints. Checkpoint replacement uses the configured minimum-improvement threshold, so the selected error may differ slightly from the lowest logged value.

and $0.004478/0.079938$ (sample/constant without tokens). Thus the poor token-free constant model also fails to fit the training data.

A completed ShapeNetCar validation evaluation reproduces both trained sample/constant-z checkpoints over 79 cases, with aggregate metric differences below 2×10^{-9} relative to their recorded summaries. Each model allocates $618{,}945$ parameters. Sample conditioning gives velocity and surface-pressure relative errors of 0.028005 and 0.077977; constant conditioning gives 0.027615 and 0.075984. Official-test metrics are reported separately in Table 3. Both unpruned models are timed in isolation, in FP32 at batch size one, using three validation cases per node cap, ten warmups, and thirty synchronized repetitions per case. Timings exclude host-to-device transfer; warm-cache CPU preprocessing is measured separately. The constant arm still computes and discards pooled z.

D Temporal and Design-Task Results

Table 8: Temporal and protocol-specific trajectory results. Targets and evaluated splits are distinct.

Task	Evaluated quantity	Epoch	Relative L_2
Navier–Stokes	Full 10-step free rollout	415	0.053698
Navier–Stokes	Last rollout frame only	415	0.096968
Plasticity	Coordinate-inclusive target	140	0.006598
Plasticity	Displacement target	140	0.156307
Plasticity trajectory	Full trajectory (80 test cases)	265	0.002629

The first two tasks evaluate 100 validation trajectories; the Plasticity trajectory row reports the 80-case trajectory protocol used for the auxiliary evidence. The NS full-rollout norm and the final-frame norm quantify different errors. For Plasticity, coordinate-inclusive and displacement targets are kept as separate metrics.

AirfRANS evaluation protocol. The frozen sample-conditioned model was fitted to 720 training cases, with 80 separate validation cases. Its 200-case holdout evaluation uses repeated uniform 32,000-node draws until every native node is covered, averages overlapping predictions, and enforces zero wall velocity and turbulent viscosity after averaging. Target normalization is frozen from training. Physical relative L_2 and normalized per-channel MSE are recorded separately. The paper's Table 3 labels field errors as relative L_2, whereas the public AirfRANS evaluation implementation reports normalized MSE. We therefore report the two field quantities separately. Force coefficients and correlations use the published physical postprocessing functions. All 200 holdout cases completed. The dataset is a locally archived public mirror used with the published force postprocessing.

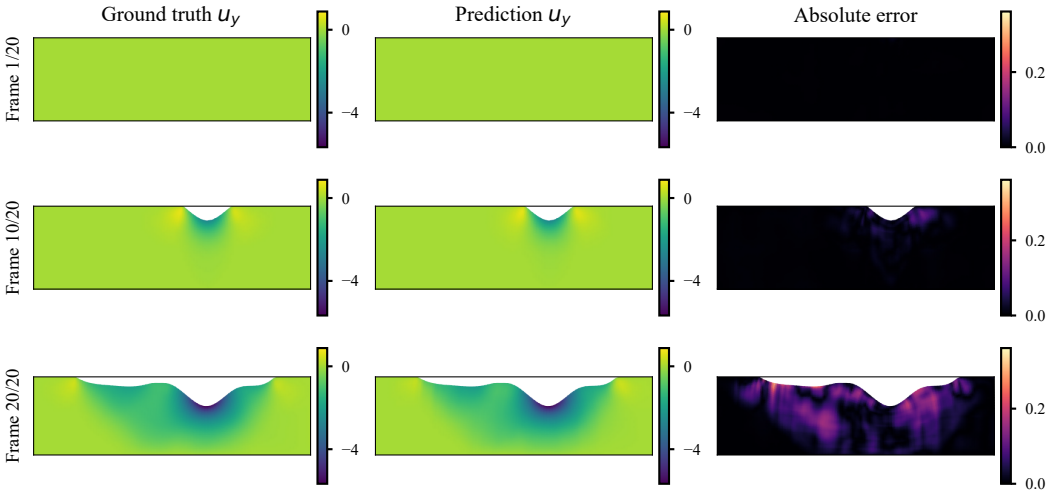

Figure 8: Plasticity trajectory example, first test case 907, epoch-265 checkpoint. Rows show frames 1, 10, and 20; columns show true vertical displacement, prediction, and absolute error on the same ground-truth positions. Color limits are fixed across times. Its four-channel trajectory error for this case is 0.002498; the saved 80-case mean is 0.002629.

Table 9: Published AirfRANS references from Transolver, Table 3 (Wu et al., 2024). Field-column definitions follow the metric note above. e_L is the reported relative lift error and ρ_L the lift Spearman correlation. Our last row evaluates all 200 cases with the published force postprocessing. Field entries are listed separately because physical relative L_2 and normalized MSE are different quantities.

Method	Volume ↓	Surface ↓	e_L ↓	ρ_L ↑
MLP	0.0081	0.0200	0.2108	0.9932
GraphSAGE	0.0087	0.0184	0.1476	0.9964
PointNet	0.0253	0.0996	0.1973	0.9919
Graph U-Net	0.0076	0.0144	0.1677	0.9949
MeshGraphNet	0.0214	0.0387	0.2252	0.9945
GNOT	0.0049	0.0152	0.1992	0.9942
Transolver	0.0037	0.0142	0.1030	0.9978
Ours (200-case holdout)	–	–	0.03818	0.99939

The lift result is strong, while drag remains more challenging. This pattern suggests that force components sensitive to surface gradients and small denominators deserve targeted modeling and diagnostics.

The AirfRANS and ShapeNetCar example fields are displayed with the four static tasks in the main-text Figure 3.

Table 10: AirfRANS frozen-checkpoint holdout metrics on all 200 cases. All means are over cases. Normalized MSE and physical relative L_2 are distinct metrics and are not compared numerically across rows. Drag is included even though the published comparison focuses on lift.

Quantity	Value
Velocity, physical relative L_2	0.021342
Volume pressure, physical relative L_2	0.047735
Surface pressure, physical relative L_2	0.038579
Volume u_x, normalized MSE	0.001295
Volume u_y, normalized MSE	0.000915
Volume pressure, normalized MSE	0.001494
Volume turbulent viscosity, normalized MSE	0.006486
Surface pressure, normalized MSE	0.006646
Lift coefficient, mean relative error	0.038176
Lift coefficient, Spearman correlation	0.999389
Drag coefficient, mean relative error	2.650673
Drag coefficient, Spearman correlation	0.496548