FLARE: Fast Low-rank Attention Routing Engine

Anonymous authors
Paper under double-blind review

Abstract

The quadratic complexity of self-attention limits the scalability of transformers on long sequences. We introduce *Fast Low-rank Attention Routing Engine (FLARE)*, a token-mixing operator that realizes low-rank attention by routing information through a small set of latent tokens. Each layer induces an input-input token mixing matrix of rank at most M via a minimal encode-decode factorization implemented using only two standard scaled dot-product attention (SDPA) calls. Because the dominant $\mathcal{O}(NM)$ computation is expressed purely in terms of standard SDPA, FLARE is compatible with fused attention kernels and avoids materializing $M \times N$ projection matrices. FLARE further assigns disjoint latent slices to each attention head, yielding a mixture of head-specific low-rank pathways. Empirically, FLARE delivers strong results across PDE surrogate benchmarks, and performs competitively on the Long Range Arena suite. We also demonstrate training on one-million-point unstructured meshes on a single GPU, and introduce a large-scale additive manufacturing benchmark dataset.

1 Introduction

High-fidelity simulations of physical systems are often too costly for multi-query applications such as design optimization and uncertainty quantification. Neural surrogate models offer a promising alternative by learning mappings from simulation inputs to outputs, enabling rapid evaluation once trained.

Transformers (Vaswani et al., 2017) have demonstrated strong scalability and generalization across domains including natural language processing (Devlin et al., 2019) and computer vision (Dosovitskiy et al., 2020). This success has motivated their adoption for spatially distributed scientific data such as point clouds and unstructured meshes, where each mesh point is treated as a token with geometric and physical features. However, standard self-attention scales as $\mathcal{O}(N^2)$ in both time and memory for N tokens, which becomes prohibitive for high-resolution meshes containing hundreds of thousands to millions of points.

A common strategy for reducing attention cost is to introduce a low-dimensional latent bottleneck of size $M \ll N$, reducing complexity to $\mathcal{O}(NM)$. Low-rank attention methods such as Linformer (Wang et al., 2020) assume that attention matrices are approximately low-rank and learn explicit $M \times N$ projection matrices. While effective at moderate sequence lengths, this approach requires $\mathcal{O}(NM)$ parameters per layer and implicitly assumes a fixed token ordering, making it impractical for very long or unordered sequences such as unstructured meshes. Latent-token architectures such as Perceiver and PerceiverIO (Jaegle et al., 2021b;a), Transolver Wu et al. (2024), and Latent Neural Operator (LNO) Wang & Wang (2024a) instead use attention projectors to map inputs to a fixed latent array, followed by deep latent-space self-attention. Notably, most latent-token methods introduce additional transformations inside the latent space, making the overall attention behavior a composition of projections, latent mixing, and nonlinear refinements rather than a single, interpretable operator.

We introduce **Fast Low-rank Attention Routing Engine (FLARE)**, a token-mixing layer that realizes a *flexible low-rank self-attention operator* by routing information through a small set of latent tokens. Concretely, each layer induces an input-input token mixing matrix of rank at most M via an *encode-decode* factorization implemented *only* with two standard scaled dot-product attention (SDPA) calls; no additional token-mixing is performed. This formulation makes the induced matrix-operator directly analyzable (e.g., via spectra) and

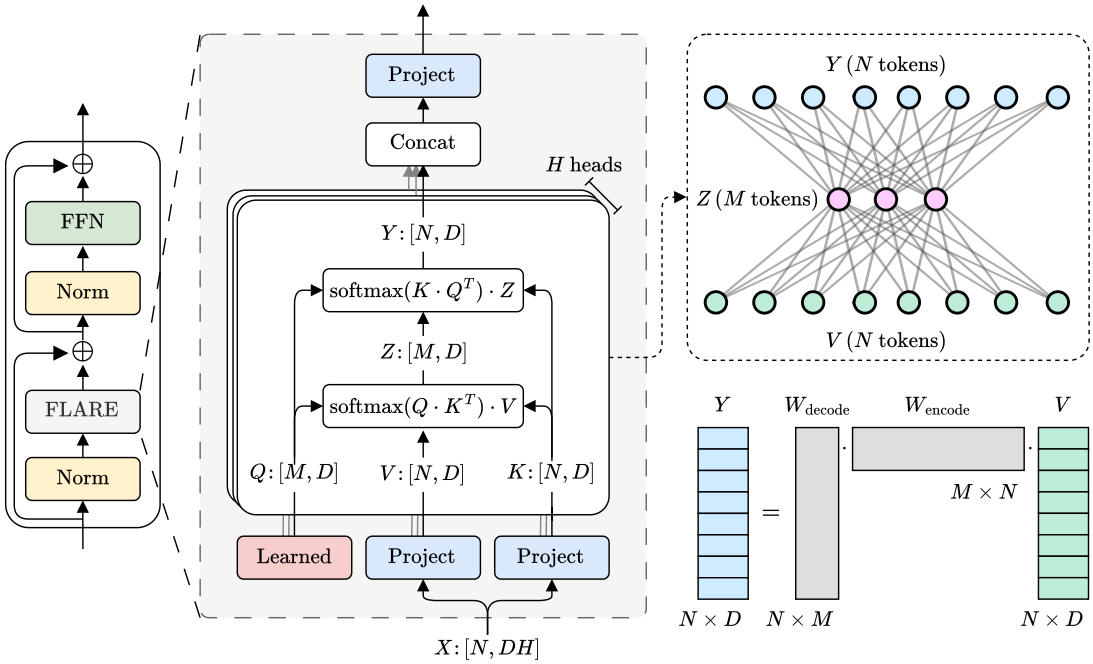


Figure 1: **FLARE block.** Each head encodes N input tokens into M latent tokens with cross-attention $W_{\text{enc}} = \text{softmax}(Q \cdot K^T)$ and decodes back with $W_{\text{dec}} = \text{softmax}(K \cdot Q^T)$, inducing an explicit rank-$\leq M$ input-space mixing matrix-operator $W = W_{\text{dec}} \cdot W_{\text{enc}}$. Unlike PerceiverIO (Jaegle et al., 2021a), Transolver (Wu et al., 2024), and LNO (Wang & Wang, 2024a), FLARE uses no latent self-attention and executes both steps with fused SDPA kernels (Dao et al., 2022) (see Figure 6 for a summary).

keeps the dominant $\mathcal{O}(NM)$ computation fully SDPA-compatible, enabling fused attention kernels without materializing $M \times N$ projection weights.

FLARE operationalizes this idea with a minimal architecture: each block performs exactly two cross-attention operations (encode and decode). Relative to Linformer (Wang et al., 2020), FLARE parameterizes low-rank token mixing without learned $M \times N$ projection matrices or a fixed token ordering. Relative to Set Transformer / Induced Set-Attention Block (ISAB) (Lee et al., 2019), which is a close prior work, FLARE isolates the induced attention mechanism down to the low-rank routing primitive itself: two bare SDPA calls with tied encode/decode query-key logits, no multihead-attention-block (MAB) wrapper around the projections, and no latent-space feedforward stack. Relative to other latent-token architectures such as PerceiverIO (Jaegle et al., 2021a), Transolver (Wu et al., 2024), and LNO (Wang & Wang, 2024a), which use latents as a computational workspace and introduce additional latent token mixing, FLARE uses latents only for routing: all token mixing is realized by a single encode-decode factorization on the input tokens, eliminating latent-space self-attention and yielding a mathematically explicit attention matrix-operator. Counterintuitively, we find through systematic ablation that this architectural simplification (removing latent self-attention entirely) consistently improves accuracy while reducing parameters compared to prior latent-attention methods.

FLARE further assigns disjoint latent slices to each attention head, yielding a mixture of head-specific low-rank matrix-operators rather than a single shared bottleneck. As a result, each head implements an independent low-rank projection–reconstruction pathway, and global communication arises from aggregating multiple such matrix-operators. We analyze these matrix-operators using a linear-time eigenanalysis method and observe diverse eigenvalue decay patterns across heads. A controlled shared-latent ablation makes these spectra more similar and consistently degrades accuracy, indicating that independent latent routing is an important contributor to performance.

Taken together, the architectural features of FLARE, (i) explicit encode-decode low-rank factorization, (ii) elimination of latent-space self-attention, and (iii) head-wise independent latent routing, define a simple low-rank attention design that is both efficient and expressive in our experiments. This minimal design

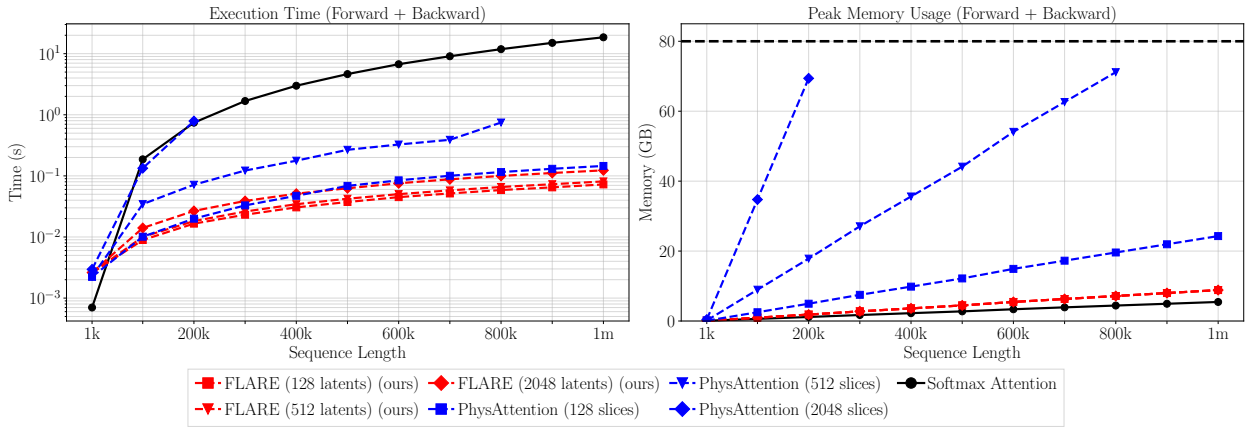


Figure 2: Time and memory requirements of different attention schemes. On an input sequence of one million tokens, FLARE (red) is over 200× faster than vanilla attention, while consuming marginally more memory. All models are implemented with FlashAttention (Dao et al., 2022), and the memory upper bound on a single H100 80GB GPU is depicted with a dashed line. Note that the curves for FLARE are somewhat overlapping. A detailed analysis is presented in Appendix G.

enables FLARE to achieve strong performance across PDE surrogate benchmarks, scale to one-million-point unstructured meshes on a single GPU, and perform competitively on standard efficient-attention benchmarks, all while using fewer parameters than existing approaches.

Figure 1 illustrates a single FLARE block. We summarize our main contributions below.

- **Flexible low-rank attention via latent routing.** We introduce a self-attention formulation in which routing through latent tokens induces an implicit rank-$\leq M$ attention matrix-operator on the input sequence. FLARE eliminates all latent-space self-attention and expresses attention as a single encode-decode factorization, yielding a mathematically transparent matrix-operator.

- **Head-wise independent low-rank pathways.** By assigning each attention head its own latent slice, FLARE learns a mixture of parallel low-rank matrix-operators; targeted ablations show this head-wise independence is critical for accuracy.

- **Extreme scalability with strong empirical performance.** FLARE trains end-to-end on one-million-point unstructured meshes on a single GPU, remains competitive across PDE surrogate benchmarks, and performs competitively on Long Range Arena as a secondary evaluation.

- **Benchmark dataset for additive manufacturing.** We introduce a large-scale thermomechanical additive manufacturing dataset to support research on scalable scientific surrogates.

2 Related work

Neural PDE surrogates. Neural operators (Li et al., 2020; Lu et al., 2021; Kovachki et al., 2023) learn mappings between function spaces and enable mesh-independent generalization. Extensions incorporating geometric priors and graph-based representations improve performance on unstructured meshes (Li et al., 2023b;c). Transformer-based architectures have emerged more recently as powerful PDE surrogates (Alkin et al., 2024; Cao, 2021; Li et al., 2022; 2023a; Hao et al., 2023), allowing global context aggregation. Recent works (Alkin et al., 2024; Wu et al., 2024; Luo et al., 2025; Wang & Wang, 2024a) leverage latent space attentions for PDE modeling, achieving high accuracy with reduced computational cost. Building upon these advances, FLARE presents a linear-complexity attention matrix-operator tailored to extreme-resolution unstructured domains.

Efficient attention mechanisms. Efficient attention methods reduce cost by constraining attention structure (e.g., rank, sparsity, or locality), yielding restricted subsets of full self-attention with bounded expressivity. Low-rank projection methods such as Linformer (Wang et al., 2020) reduce attention complexity with learned sequence projections, while Reformer (Kitaev et al., 2020) and Funnel-Transformer (Dai et al., 2020) improve efficiency through hashing or hierarchical sequence reduction. Kernel and linear-attention methods such as Performer (Choromanski et al., 2020), cosFormer (Qin et al., 2022b), and Hedgehog (Zhang et al., 2024) instead replace softmax attention with feature-map-based or learned linear approximations.

For permutation-invariant set inputs, Set Transformer (Lee et al., 2019) introduced Induced Set-Attention Block (ISAB), which is a close prior mechanism to FLARE. ISAB first lets a small set of M learned *inducing* points attend to the input set and then lets the input tokens attend back to those induced summaries, reducing the cost of a self-attention block from quadratic in N to $\mathcal{O}(NM)$. The important distinction is not the presence of inducing tokens, which FLARE inherits, but how much machinery surrounds them. In Set Transformer, the two cross-attention steps are implemented as MAB modules, so each projection is wrapped with residual connections, layer normalization, and row-wise feedforward layers. FLARE removes this block-level latent-processing stack and keeps only the two dot-product-attention calls needed to form a low-rank attention-routing matrix-operator. This makes the induced input-space mixing matrix explicit as $W_{\text{dec}}W_{\text{enc}}$, ties encode/decode scores through the same learned latent queries and input keys, assigns independent latent slices to each head, and keeps the dominant $\mathcal{O}(NM)$ computation expressible as standard fused SDPA.

PerceiverIO (Jaegle et al., 2021a) and related latent-token models use latent-space self-attention as a computational workspace to aggregate and transform global information. In this design, interactions between input tokens are mediated through a sequence of latent-space transformations rather than defined by a single, explicit operator on the input tokens. Transolver (Wu et al., 2024; Luo et al., 2025), LNO (Wang & Wang, 2024a), and the concurrent LANO (Zhong & Yan, 2025) adapt this latent projection-unprojection paradigm for PDE surrogate modeling. FLARE differs from the projection- and kernel-based efficient-attention methods above by realizing token mixing through latent routing rather than explicit $M \times N$ projections or softmax approximations. Relative to ISAB and later Perceiver-style latent architectures, FLARE takes the more stripped-down route of using latent tokens only to induce an explicit low-rank matrix-operator on the input tokens, with no latent self-attention or additional latent-space processing, yielding a simpler and more analyzable formulation.

3 Method

3.1 Preliminary: Multi-Head Self-Attention

Let $X \in \mathbb{R}^{N\times C}$ denote the input sequence of N tokens with C features each. The query, key, and value matrices $Q, K, V \in \mathbb{R}^{N\times C}$ are obtained by applying learned linear projections to X:

$$Q = X \cdot W^q, \quad K = X \cdot W^k, \quad V = X \cdot W^v, \tag{1}$$

where $W^q, W^k, W^v \in \mathbb{R}^{C\times C}$. The Q, K, V matrices are then split along the feature dimension and passed to H heads, each with dimension $D = C/H$, enabling parallel computation of attention: $[Q_1, \ldots, Q_H] = Q$, $[K_1, \ldots, K_H] = K$, $[V_1, \ldots, V_H] = V$. The scaled dot-product attention (SDPA) operation, introduced by (Vaswani et al., 2017), computes the output as

$$Y_h = \text{SDPA}(Q_h, K_h, V_h, s) = \text{softmax}\left(\frac{Q_h \cdot K_h^T}{s}\right) \cdot V_h, \tag{2}$$

where Q_h, K_h, $V_h \in \mathbb{R}^{N\times D}$ are query, key, and value matrices belonging to head h, and s is typically $\sqrt{D}$. Note that softmax is taken along the row-dimension. The outputs Y_h from all heads are then concatenated along the feature dimension to form the final output

$$Y = [Y_1, \ldots, Y_H]\,. \tag{3}$$

```python
import torch.nn.functional as F
def flare_multihead_mixer(Q, K, V):
    # Args - Q: [H M D], K, V: [B H N D]
    # Ret - Y: [B H N D]
    Z = F.scaled_dot_product_attention(Q, K, V)
    Y = F.scaled_dot_product_attention(K, Q, Z)
    return Y
```

Figure 3: PyTorch code for multi-head token mixing operation in FLARE. See Figure 7 for an implementation without the fused attention kernel.

This concatenation, followed by a linear layer, enables the model to integrate information across attention heads efficiently.

The greatest cost in multi-head self-attention is the call to SDPA which is $\mathcal{O}(N^2)$ in time and memory complexity. This is because $Q_h \cdot K_h^T \in \mathbb{R}^{N\times N}$ requires $\mathcal{O}(N^2)$ storage and softmax, matrix-vector product with V_h takes $\mathcal{O}(N^2)$ operations. Fortunately, GPU-optimized multi-head implementations of SDPA are available in PyTorch (Paszke, 2019).

3.2 FLARE: Fast Low-rank Attention Routing Engine

FLARE is a linear-complexity token mixing layer that learns low-rank global communication structures via attention projections. The FLARE mechanism introduces a set of $M \ll N$ learnable latent tokens that serve as a bottleneck for information exchange. The process consists of two stages:

1. **Encoding.** The input sequence is projected onto the latent tokens via cross-attention, compressing global information.

2. **Decoding.** The latent tokens are then projected back onto the input sequence, distributing the aggregated information.

Formally, we define a learnable query matrix $Q \in \mathbb{R}^{M\times C}$, where each row corresponds to a latent token. The key and value matrices, $K, V \in \mathbb{R}^{N\times C}$, are obtained by applying learned projection modules to the input X.

The matrices Q, K, and V are first split along the feature dimension into H heads, each of dimension $D = C/H$. Then, for encoding, each head performs SDPA with a scaling factor $s = 1$:

$$Z_h = \mathrm{SDPA}(Q_h, K_h, V_h, s = 1). \tag{4}$$

Here, $Q_h \in \mathbb{R}^{M\times D}$, $K_h, V_h \in \mathbb{R}^{N\times D}$ are query, key, and value matrices belonging to head h and $Z_h \in \mathbb{R}^{M\times D}$ is the latent sequence for head h. For decoding and propagating information back to the input tokens, we perform a second SDPA operation, swapping the roles of queries and keys and using the latent sequence as values:

$$Y_h = \mathrm{SDPA}(K_h, Q_h, Z_h, s = 1) \tag{5}$$

where $Y_h \in \mathbb{R}^{N\times D}$ is the output for each head. Similar to multi-head self-attention, the outputs from all heads are concatenated along the feature dimension and passed through a final linear projection to mix information across heads. As the query matrix has only M tokens, the cost of SDPA calls in Eq. 4 and Eq. 5 is $\mathcal{O}(NM)$. PyTorch code for the multi-head implementation is presented in Figure 3. Note that we use a scaling factor $s = 1$ instead of $\sqrt{D}$ in typical transformers (Vaswani et al., 2017) in SDPA. This modification is explained in Appendix G.

Low-rank communication. The two-step attention process can be written as

$$Y_h = (W_{\mathrm{decode},h} \cdot W_{\mathrm{encode},h}) \cdot V_h \tag{6}$$

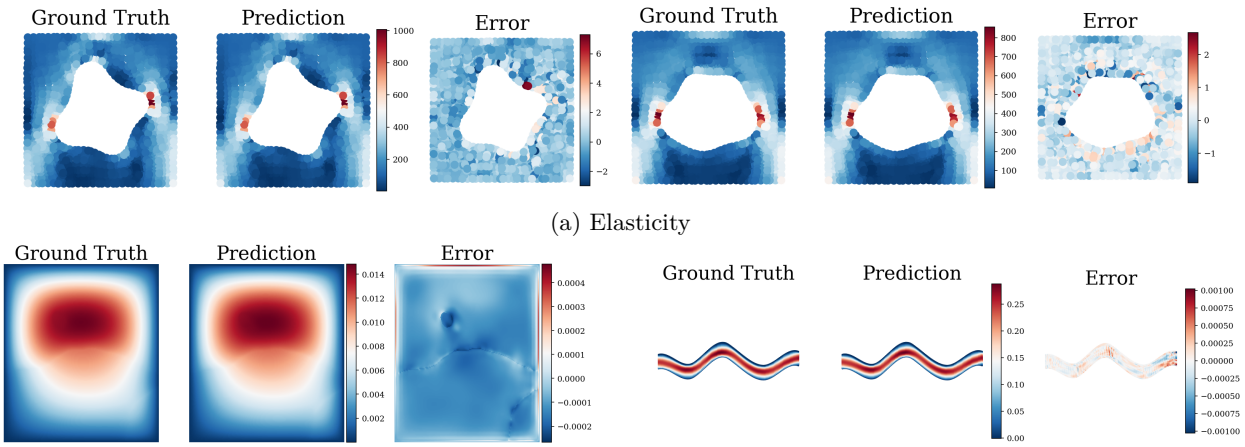


Figure 4: Qualitative results for FLARE on the (a) Elasticity, (b) Darcy, and (c) Pipe datasets. We show the ground truth, the model prediction, and the corresponding error (Ground Truth − Prediction).

where

$$\begin{aligned} W_{\text{encode},h} &= \text{softmax}(Q_h \cdot K_h^T) \in \mathbb{R}^{M\times N}, \quad \text{and} \\ W_{\text{decode},h} &= \text{softmax}(K_h \cdot Q_h^T) \in \mathbb{R}^{N\times M}. \end{aligned} \tag{7}$$

Note that softmax is taken along the row-dimension. We define

$$W_h = W_{\text{decode},h} \cdot W_{\text{encode},h} \in \mathbb{R}^{N\times N} \tag{8}$$

as the dense global communication matrix with rank at most M. This low-rank structure, illustrated in Figure 1, enables efficient all-to-all communication without explicitly forming W_h; instead, $W_{\text{encode},h}$ and $W_{\text{decode},h}$ are applied sequentially, at an overall cost of $\mathcal{O}(MN)$ per head.

Discussion on the design principles of FLARE. FLARE's latent tokens implement a structured gather-scatter communication pattern: encoding pools input values into M compact global descriptors, then decoding selectively broadcasts these back along the same pathways, yielding a low-rank attention-routing matrix-operator that avoids explicit all-to-all interactions. This mechanism relies on two design choices discussed in Appendix F: (i) symmetric encode-decode operators promote stable information flow without redundant parameters, and (ii) fixed input-independent latent queries simplify the low-rank structure and improve efficiency, shifting expressive capacity to deep residual key/value projections that recover representational power.

FLARE block. Figure 1 (left) illustrates a single FLARE block. Given input tokens $X \in \mathbb{R}^{N\times C}$, the output of an FLARE block is computed as

$$\begin{aligned} X &= X + \text{FLARE}\left(\text{Norm}\left(X\right)\right) \\ X &= X + \text{FFN}\left(\text{Norm}\left(X\right)\right). \end{aligned} \tag{9}$$

Here, FFN denotes a pointwise feedforward module and Norm denotes a normalization layer; both are instantiated differently for the LRA and PDE variants detailed in Appendix B. To summarize, a FLARE block consists of a token mixing operation via FLARE, a pointwise feedforward module, and normalization in pre-norm format (Xiong et al., 2020). This benchmark-dependent instantiation is important in practice. The primary PDE surrogate setting places substantially more burden on pointwise function approximation, so in that setting we instantiate both the FFN and the key/value projection modules as deeper residual MLPs. By contrast, the secondary LRA experiments use a shallower standard efficient-attention backend with a conventional two-layer FFN, linear key/value projections, and query/key normalization. Appendix B discusses these two block instantiations in full detail.

Table 1: Relative L_2 error ($\times 10^{-3}$) for different models across PDE benchmark problems. Parameter count for each model is shown in the leftmost column. Darcy, Airfoil, Pipe, and Plasticity are structured quad-mesh benchmarks that permit neighborhood augmentation via convolutional or analogous local aggregation; Plasticity is additionally *time-dependent*, whereas the remaining tasks are static or final-state surrogates. We report mesh-dependent / neighborhood-augmented models separately from mesh-connectivity-free token mixers because the former augment global communication with explicit local grid/mesh aggregation. Bold and underline indicate the best and second-best results *within each family*, respectively. Baseline citations are given in the accompanying text. A backslash (\) indicates that the model cannot be applied to the benchmark, and tilde (∼) indicates that the model is prohibitively slow on the benchmark. For Elasticity, the FLARE and Transolver w/o conv entries are 10-seed means; full per-seed results and standard deviations are reported in Section D.4.

Model (params)	**Elasticity**	**Darcy**	**Airfoil**	**Plasticity**	**Pipe**	**DrivAerML-40k**	**LPBF**
Mesh-Dependent / Neighborhood-Augmented Models							
Transolver with conv (2.81m)	\	<u>5.94</u>	<u>5.50</u>	1.30	**3.90**	\	\
LaMO with conv (3.76m)	\	**5.44**	**4.65**	<u>1.28</u>	<u>4.26</u>	\	\
MambaNO (1.83m)	\	7.77	8.30	**0.65**	4.71	\	\
Mesh-Connectivity-Free / Pure Point-Cloud Token Mixers							
Vanilla Transformer (660k)	5.37	**4.38**	<u>6.28</u>	3.22	∼	∼	∼
PerceiverIO (1.87m)	28.0	20.6	7.65	1.68	6.90	248	23.1
Set Transformer (1.76m)	5.17	12.5	7.99	<u>1.57</u>	<u>3.85</u>	62.0	22.3
GNOT (4.88m)	13.3	16.9	103	44.1	5.89	115	24.3
LNO (915k)	9.25	7.64	17.8	1.92	8.10	146	24.7
Transolver w/o conv (715k)	8.15	18.6	8.24	1.96	4.87	70.5	<u>20.4</u>
LaMO w/o conv (1.72m)	<u>4.25</u>	8.72	6.59	1.71	4.17	<u>61.9</u>	22.7
FLARE (ours) (640k)	**3.95**	<u>5.10</u>	**4.28**	**1.39**	**2.85**	**60.8**	**18.5**

Overall design. The overall architecture is given by B sequential FLARE blocks sandwiched between an input projection and an output projection, which are detailed in Appendix B. Such a design enables the model to efficiently integrate local and global information across multiple layers, making it well suited for large-scale, high-dimensional data such as point clouds.

3.3 Spectral analysis

The matrix W_h in Eq. 8 represents the attention weights between N tokens, where $[W_h]_{ij}$ quantifies how much token j communicates to token i within head h. Since W_h has rank at most M, the model can capture at most M independent global communication patterns per head. The eigenvalues of W characterize the distribution of spectral mass and the effective rank of the attention matrix W. We apply an algorithm to obtain the eigendecomposition of W in $\mathcal{O}(M^3 + M^2N)$ time compared to $\mathcal{O}(N^3)$ for a dense communication matrix. The algorithm is predicated on computing the eigenspectra of an $M \times M$ matrix JJ^T where $J \in \mathbb{R}^{M\times N}$ is chosen such that J^TJ is similar to W. The algorithm is detailed in Section C.1, and summarized in Algorithm 1. Figure 13 presents the M nonzero eigenvalues of W_h for an FLARE model trained on the Elasticity benchmark problem with 972 points per input. The distinct spectra show that the heads use the available rank differently. Further experiments, such as eigenspace-overlap or head-intervention analyses, are needed to establish whether the heads span different eigenspaces or learn different functional roles.

The eigenvalue analysis detailed in Section C.2 shows that while FLARE provides capacity for rank-M attention, the model learns to use only a small fraction of this in early blocks indicating effective compression. In deeper blocks, more of the latent capacity is utilized, with diverse spectral profiles across heads. We therefore treat this spectral study as a characterization of rank utilization rather than as evidence of head specialization. Appendix G's shared-vs-independent-latent ablation separately shows that head-wise independence improves downstream accuracy.

4 Benchmark dataset for additive manufacturing

We introduce a benchmark dataset of *laser powder bed fusion* (LPBF) simulations designed to evaluate surrogate models on large, irregular 3D geometries. LPBF is a widely used metal additive manufacturing process in which a laser fuses thin layers of powder, producing complex parts but often inducing *residual stresses* and *distortions* that can lead to build failures (Zhang et al., 2018). To capture these effects, we simulate the thermo–mechanical LPBF process on thousands of geometries drawn from the Fusion 360 segmentation dataset (Lambourne et al., 2021).

Our benchmark task is *predicting vertical (Z) displacement field* for each geometry, a quantity directly associated with recoater-blade collisions and build-failure risk in LPBF. Each sample consists of the 3D mesh coordinates as input and the final Z-displacement at all nodes as output. All parts are scaled into a standardized build volume and simulated in Autodesk NetFabb for a Renishaw AM250 machine with Ti-6Al-4V material, using 40 μm layer thickness and layer lumping with 2.5 mm lumped layers. The dataset spans a wide variety of part shapes, mesh resolutions, and deformation magnitudes, making it a challenging and practically meaningful testbed for large-scale surrogate modeling. Out of 19,732 successful simulations, we construct a filtered benchmark subset with 1,100 training cases and 290 test cases. The meshes in this subset contain a mean of 20,972 points (median 19,743), with sizes ranging from 736 to 47,542 points and a mean of 114,140 edges. We additionally filter for meshes with at most 50,000 points, at most 300,000 edges, and acceptable element aspect ratios to avoid unstable FEM solutions. This broad distribution of mesh sizes and geometric complexity makes the benchmark substantially more demanding than prior LPBF surrogate datasets based on much coarser meshes. Summary statistics are provided in Table 10 and Figure 15, with further examples of successful simulations shown in Figure 14; full details are given in Appendix H.

5 Experiments

5.1 PDE surrogate benchmarks

Benchmark problems. We consider a diverse set of benchmark datasets (Table 4) for regressing PDE solutions on point sets spanning structured and unstructured grids with up to 50,000 points. FLARE is mesh-connectivity-free: it operates solely on point coordinates and features, without using mesh connectivity, neighborhood graphs, or grid-specific local aggregation. This term describes the information consumed by the architecture and does not by itself imply discretization invariance or zero-shot generalization across resolutions and sampling distributions. The 2D Elasticity, Darcy, Airfoil, and Pipe benchmarks (Li et al., 2020; 2023b) cover a wide range of physical phenomena; the Plasticity benchmark adds a structured *time-dependent* deformation task; and the 3D DrivAerML benchmark (Ashton et al., 2024) provides automotive aerodynamic simulations. Visualizations of the Elasticity, Darcy, and Pipe benchmark problems are presented in Figure 4. Additional details of the dataset are presented in Section D.2. We also introduce a 3D field-prediction benchmark derived from laser powder bed fusion (LPBF) simulations, with diverse 3D-printed parts containing up to 50,000 grid points (see Appendix H).

Baselines. We compare FLARE with state-of-the-art PDE surrogates: generic attention models (vanilla Transformer (Vaswani et al., 2017), PerceiverIO (Jaegle et al., 2021a), and Set Transformer / ISAB (Lee et al., 2019)); attention-based PDE surrogates (Transolver (Wu et al., 2024), LNO (Wang & Wang, 2024a)); state-space neural operators (LaMO (Tiwari et al., 2025) and MambaNO (Zheng et al., 2024)); and the neural operator GNOT (Hao et al., 2023). We exclude graph-based models because graph connectivity is unavailable for most problems. Transolver++ (Luo et al., 2025) is a relevant recent baseline. We reran the latest official implementation with the stated hyperparameters, but were unable to match the reported numbers, so we do not include it in our comparisons. This concern is not unique to our setup: related issues are also noted in Appendix B.7 of AB-UPT (Alkin et al., 2025), and NVIDIA PhysicsNeMo reports that in their external-aerodynamics experiments they did not observe gains from Transolver++ (NVIDIA PhysicsNeMo Team, 2026). Full details are provided in Section D.3.

We follow the experimental setup of Transolver as it is a strong recent surrogate model and attempt to match its parameter count. Note that Transolver can be instantiated in two configurations: *without convolution*,

where point-to-point communication relies solely on physics attention, and *with convolution*, where convolution layers are added to inject information from neighboring points when the input grid is structured. We evaluate these two configurations separately to isolate the impact of convolution versus physics attention. We likewise report LaMO with and without convolution separately, and list MambaNO with the same mesh-dependent family, because these models augment global communication with explicit grid/mesh-local aggregation and therefore should not be interpreted as purely point-cloud token mixers. In our model, we choose not to employ any convolution layers and rely entirely on FLARE for token mixing. For the PDE benchmarks, we use the PDE-oriented FLARE variant with 3-layer residual MLPs (5 layers total) for both the key/value projections and the FFN. Because these deeper pointwise modules increase parameter count, we reduce FLARE's channel dimension from the $C = 128$ commonly used in Transolver to $C = 64$ so that the parameter budget remains comparable to Transolver without convolution. Because these PDE problems are relatively small (up to 50,000 points), we train all models in FP32. As shown in Figure 8, the vanilla Transformer is drastically slower than FLARE and Transolver on large point clouds; accordingly, we evaluate it only on problems up to $\sim$ 10,000 points.

Discussion. The results in Table 1 show that FLARE performs well across the PDE suite and is the strongest mesh-connectivity-free model on most of the benchmarks in our setup, while using fewer parameters than most latent-attention alternatives. Darcy, Airfoil, Pipe, and Plasticity are all structured quad-mesh benchmarks for which local neighborhood augmentation is available in principle; Plasticity is the clearest case because it is also *time-dependent*. On that benchmark in particular, the best results come from models that augment global communication with explicit local grid/mesh aggregation. We therefore view Plasticity less as a pure comparison of global attention mechanisms, and more as evidence that mesh-dependent local inductive bias can be especially valuable on structured spatiotemporal problems. This separation is also important for interpreting the Transolver and LaMO rows. The poor performance of Transolver without convolutions (Wu et al., 2024) indicates that its built-in physics-attention mechanism alone is not sufficient on these tasks, whereas the convolutional variants first amass neighborhood information before global mixing. For that reason, we report those mesh-dependent models separately from the mesh-connectivity-free point-cloud token mixers.

Although the vanilla transformer is extremely effective on small-scale PDE problems, it becomes prohibitively slow on large point clouds due to its quadratic cost as illustrated in Figure 2. By contrast, PerceiverIO (Jaegle et al., 2021a) (with only a single encoding and decoding step) performs poorly even with $M = 1,024$ latent tokens and $B = 8$ latent self-attention blocks. Set Transformer / ISAB (Lee et al., 2019), a close prior work, is more competitive than PerceiverIO and remains close on the 3D tasks, but is still consistently worse than FLARE while using roughly three times as many parameters. Together with the controlled allocation study in Figure 12, these results support a more specific conclusion: latent-space self-attention can be useful, but repeated input-to-latent-to-input routing provides a better accuracy-efficiency tradeoff for the tasks and budgets studied here. Once information is lost during the initial projection, latent self-attention alone cannot recover it; repeated routing lets the model revisit the input domain at every block. Instead, performing multiple (head-wise) parallel projections and reconstructions directly between the input and latent sequences preserves expressivity while simplifying the architecture.

5.2 Field-prediction on million-point geometries

The benchmark problems in Section 5.1 contain a variety of PDE problems, but are relatively small compared to industrial use cases that demand PDE solutions on complex geometries with millions of grid points (Ashton et al., 2024). Million-scale regression remains difficult for many attention-based surrogate models because quadratic attention and non-fused projection/unprojection steps create severe time and memory bottlenecks, as illustrated in Figure 2. The FlashAttention (Dao et al., 2022) algorithm has alleviated the memory bottleneck thanks to online softmax computation; however, these methods remain impractical due to their long training times. Furthermore, SOTA models such as Transolver and LNO are not readily expressible purely in terms of SDPA calls for their dominant $\mathcal{O}(NM)$ projection/unprojection steps; in typical implementations these steps explicitly materialize $M \times N$ projection weights, which prevents using PyTorch's fused SDPA backends (e.g., FlashAttention-style kernels) for the $\mathcal{O}(NM)$ bottleneck.

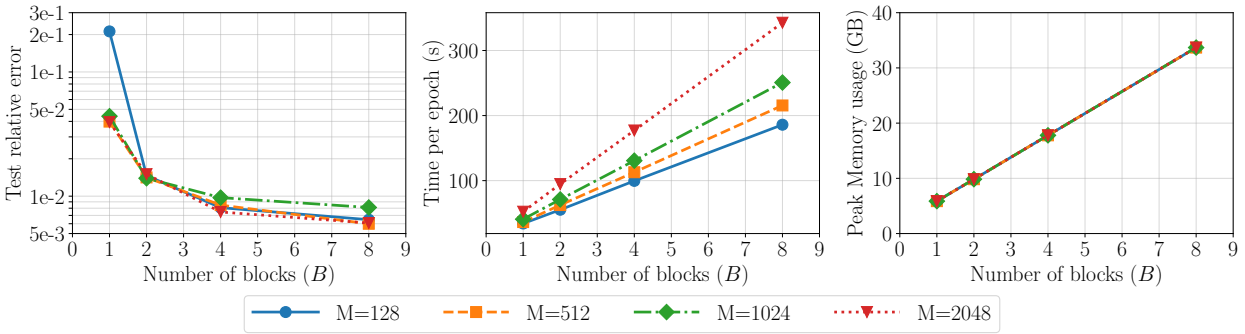


Figure 5: We train FLARE on the DrivAerML dataset (Ashton et al., 2024) with one million points per geometry on a single Nvidia H100 80GB GPU as a scaling demonstration. We present (left) the test relative error, (middle) time per epoch (s), and (right) peak memory utilization (GB) as a function of the number of FLARE blocks (B) for different number of latent tokens (M).

Table 2: Accuracy (%) on Long Range Arena (LRA) tasks (Tay et al., 2021b). Rows marked with † use the reported result from Zhang et al. (2024); unmarked rows are our reruns. The best result in each column is **bold** and the second best is underlined.

Model	ListOps	Text	Retrieval	Image	Pathfinder-32	Avg
Vanilla attention	36.70	64.93	77.18	38.22	70.52	57.51
Local attention†	15.82	52.98	53.39	41.46	66.63	46.06
Reformer†	37.27	56.10	53.40	38.07	68.50	50.67
Sparse Transformer†	17.07	63.58	59.59	44.24	71.71	51.24
Sinkhorn Transformer†	33.67	61.20	53.83	41.23	67.45	51.29
Linformer (KV sharing)	36.95	51.74	77.86	42.82	50.02	51.88
Performer	35.90	64.21	68.42	37.60	53.83	51.99
Funnel-Transformer	38.50	61.17	61.55	**53.10**	49.98	52.86
Synthesizer†	36.99	61.68	54.67	41.61	69.45	52.88
Linear attention	17.95	**66.00**	71.84	34.66	75.00	53.09
Longformer†	35.63	62.85	56.89	42.22	69.71	53.46
Linformer (headwise sharing)	36.70	53.00	64.72	43.42	70.09	53.59
BigBird†	36.05	64.02	59.29	40.83	74.87	55.01
Norm attention	18.30	63.08	76.07	48.22	70.15	55.16
Performer (FAVOR++)	36.00	64.26	76.74	35.60	69.67	56.45
cosFormer	36.20	64.59	76.78	41.52	75.38	58.89
Nyströmformer†	37.15	65.52	79.56	41.58	70.94	58.95
Skyformer†	**39.25**	64.70	82.06	40.77	70.73	59.50
Hedgehog†	37.15	64.60	**82.24**	40.15	74.16	59.66
Transolver	17.65	60.95	76.56	34.50	66.94	51.32
FLARE (ours)	38.20	65.48	79.21	41.56	**76.64**	**60.22**

We use Figure 5 as a scaling demonstration for FLARE by training on the DrivAerML dataset (Ashton et al., 2024) where each mesh is subsampled to contain one million points. The goal of this experiment is to test a model's ability to ingest and process a full 1M-point context at once, rather than to aggregate over chunks or windows. These calculations are performed in mixed precision on a single Nvidia H100 80GB GPU provisioned through Google Cloud Platform. We note the clear trend in Figure 5 (left) that the error consistently decreases as we increase the number of FLARE blocks.

We investigated the strongest plausible comparison models at the million-point scale, but were unable to obtain a fair end-to-end baseline due to a combination of reproducibility and feasibility issues, most notably for Transolver++. In particular, we directly attempted to reproduce Transolver++ using the official release and reported hyperparameters, including our own large-scale reruns, but were unable to match the reported accuracy. In our large-scale experiments, both Transolver and Transolver++ also exhibited numerical instability under the mixed-precision regime (BF16 in the forward pass, FP32 in the backward pass) that is effectively required to make 1M-point training viable on a single GPU. We also do not treat chunked approaches such as AB-UPT (Alkin et al., 2025) as direct baselines for this experiment, since our objective here is specifically full-context 1M-point ingestion rather than chunked processing. These limitations are also consistent with NVIDIA PhysicsNeMo's report that they did not observe gains from Transolver++ in their external-aerodynamics experiments (NVIDIA PhysicsNeMo Team, 2026). Appendix D.3 discusses these limitations in detail. To our knowledge, this is the first attention-based neural surrogate model trained on one million points on a single GPU without memory offloading or distributed computing. Additional details are presented in Appendix E.

5.3 Long Range Arena benchmark problems

We also evaluate FLARE on the Long Range Arena (LRA) benchmark suite (Tay et al., 2021b), which covers a diverse set of long-context tasks ranging from logical reasoning and text classification to retrieval, image classification, and visual pathfinding. PDE surrogate modeling on large unstructured meshes and point clouds remains the main empirical focus of the paper, but LRA provides a standard reference point for efficient-attention methods and lets us evaluate FLARE on a familiar long-context benchmark outside the PDE setting.

LRA is not a perfect benchmark. In particular, recent work has pointed out a strong locality and positional bias in several LRA tasks (Miralles-González et al., 2025). Nevertheless, it remains the most widely used controlled benchmark for efficient-attention comparisons, and we use it here only as a standard reference point for situating FLARE against prior sub-quadratic attention methods.

Crucially, FLARE is designed so that token order does not matter unless positional information is injected through position embeddings. In that sense, like a vanilla Transformer without positional encodings, the FLARE attention matrix-operator itself does not assume any canonical sequence order. For this reason, our LRA comparison focuses on efficient-attention architectures rather than fixed-order sequence models such as MEGA (Ma et al., 2022), S4 (Gu et al., 2021), or Mamba (Gu & Dao, 2024). Those state-space and scan-based models often do better on LRA precisely because their fixed-order inductive bias is useful on sequence benchmarks, whereas our target PDE domains consist of unordered point clouds and unstructured meshes where no canonical scan order is available.

For all models, we use the same transformer-block backbone model, including the same linear input and output projections, query/key/value projections, and FFN; only the token mixer is varied. Our comparison set includes vanilla attention (Vaswani et al., 2017); local attention and the original LRA baselines (Tay et al., 2021b); Reformer (Kitaev et al., 2020); Sparse Transformer (Child et al., 2019); Sinkhorn Transformer (Tay et al., 2020); Linformer (Wang et al., 2020); Performer and FAVOR++ (Choromanski et al., 2020); Funnel-Transformer (Dai et al., 2020); Synthesizer (Tay et al., 2021a); linear attention (Katharopoulos et al., 2020); Longformer (Beltagy et al., 2020); BigBird (Zaheer et al., 2020); Norm attention (Qin et al., 2022a); cosFormer (Qin et al., 2022b); Nyströmformer (Xiong et al., 2021); Skyformer (Chen et al., 2021); and Hedgehog (Zhang et al., 2024). For LRA specifically, we use the lightweight FLARE variant with linear key/value projections, a standard FFN with GELU activation, and query/key normalization. As noted in Table 2, rows marked with † use the reported result from Zhang et al. (2024), while the remaining rows are our reruns. We also include Transolver (Wu et al., 2024) as a token-mixer reference.

As shown in Table 2, FLARE does not dominate every task individually, but it achieves the strongest average among the efficient-attention baselines in this comparison while retaining $\mathcal{O}(NM)$ complexity. FLARE outperforms Transolver on all five tasks and improves average accuracy from 51.32 to 60.22; Appendix G.1 discusses this controlled comparison alongside the Elasticity ablation.

5.4 Model analysis and ablation studies

We conduct focused ablations in Appendix G that causally isolate the contribution of each component and clarify trade-offs between expressivity, time, and memory complexity. These architectural ablations are carried out in the context of PDE surrogate modeling, which is the primary application focus of this paper. An anonymized supplementary reproducibility package provides the training pipeline, model implementations, benchmark launch scripts, environment specification, and dataset access utilities used for these experiments. To maintain the double-blind review process, the URLs to our fully reproducible codebase and the LPBF dataset are omitted during review. These links will be made publicly available in the camera-ready version.

We first examine **time and memory scaling** as a function of sequence length. FLARE exhibits true linear $\mathcal{O}(NM)$ scaling in both runtime and peak memory usage, even up to one million tokens. This behavior follows directly from FLARE's SDPA-compatible formulation, which avoids materializing attention matrices or $M \times N$ projection weights and allocates sequence-length–dependent memory only for input activations.

The six architectural ablations in Appendix G then proceed in sequence. First, we study the trade-off between **model depth and latent dimensionality**. Across all PDE benchmarks, including the 1M-point DrivAerML dataset, accuracy improves consistently with more encode–decode blocks, while increasing the number of latent tokens M yields diminishing returns except on inherently high-rank tasks. This demonstrates that repeated low-rank mixing through depth is a more effective driver of global communication than increasing the rank of a single attention matrix-operator.

Second, we vary the **depth of the key/value residual MLP projections**. Deeper key/value encoders consistently improve accuracy, supporting the view that fixed latent queries work best when the key and value pathways are sufficiently expressive.

Third, we vary the **depth of the pointwise feedforward residual MLP**. Here too, deeper residual pointwise blocks improve stability and accuracy, justifying the PDE-oriented residual MLP instantiation used in the main benchmark models.

Fourth, we evaluate **head dimension and the number of parallel low-rank pathways**. Varying the number of attention heads while keeping total width fixed shows that FLARE performs best with many small heads rather than a few large ones, indicating that aggregating multiple independent low-rank matrix-operators approximates richer attention patterns than concentrating capacity into fewer projections.

Fifth, we compare **latent-space self-attention versus encode–decode blocks**. Latent self-attention consistently worsens accuracy and increases computation, whereas allocating the same budget to additional encode–decode blocks improves performance; the best results occur when latent self-attention is entirely removed.

Sixth, we study **shared versus independent latent tokens**. Sharing latents across heads collapses the induced spectra and degrades accuracy, while independent latent slices yield diverse eigenvalue decay profiles and lower error.

Together, these ablations validate FLARE's architectural choices: explicit low-rank matrix-operator factorization, repeated encode-decode mixing, and head-wise latent independence. Its gains, therefore, do not stem from increased parameter count or implementation-level optimizations.

6 Conclusion

FLARE is a token mixing layer that bypasses the quadratic cost of self-attention by leveraging low-rankness. Mechanically, FLARE routes attention through a fixed-size latent sequence via cross-attention projection and unprojection. FLARE achieves strong overall accuracy across a diverse set of PDE benchmarks, and demonstrates feasible single-GPU training on PDE problems with million-scale geometries.

There are also several clear directions for improving the current PDE-oriented instantiation of FLARE. In particular, the PDE variant's reliance on deep residual MLPs can introduce sequential bottlenecks and increase latency, leaving room for further speedups and architectural refinements. Natural next steps include (1)

incrementally increasing the number of latent tokens during training; (2) conditioning the latent queries/tokens on the input to relax the fixed-Q constraint; (3) conditioning latent tokens on time for diffusion modeling; (4) designing decoder-only variants for autoregressive modeling; (5) studying FLARE as an operator-learning model, including generalization across discretizations and sampling distributions, building on theoretical results for transformers as neural operators (Shih et al., 2024); and (6) studying FLARE as a scalable backbone for pretrained operator models and combining its sequence-efficient routing with frequency-aware parameter-efficient adaptation methods such as F-Adapter (Zhang et al., 2025). Overall, FLARE offers a simple explicit low-rank attention design that scales unusually well for PDE surrogate modeling.

Impact Statement

Self-attention underpins many modern AI systems, but its quadratic computational and memory cost makes both training and inference expensive and difficult to scale. By making attention-based models practical on very large inputs within single-GPU workflows, FLARE lowers the barrier to deploying powerful sequence models across a wide range of applications. In engineering surrogate-modeling settings, faster inference can accelerate design iteration and simulation triage, but such surrogates should supplement rather than replace validated numerical simulation, physical testing, or domain-expert review in safety-critical decisions. This is particularly important for additive manufacturing: an inaccurate LPBF deformation or build-risk prediction could lead to poor manufacturing choices if used without downstream validation.

References

Benedikt Alkin, Andreas Fürst, Simon Schmid, Lukas Gruber, Markus Holzleitner, and Johannes Brandstetter. Universal physics transformers. *arXiv e-prints*, pp. arXiv–2402, 2024.

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, 2025. URL `https://arxiv.org/abs/2502.09692`.

Neil Ashton, Charles Mockett, Marian Fuchs, Louis Fliessbach, Hendrik Hetmann, Thilo Knacke, Norbert Schonwald, Vangelis Skaperdas, Grigoris Fotiadis, Astrid Walle, et al. Drivaerml: High-fidelity computational fluid dynamics dataset for road-car external aerodynamics. *arXiv preprint arXiv:2408.11969*, 2024.

Autodesk. *NetFabb Simulation Utility and Local Simulation*. Autodesk Inc., San Francisco, California, United States, 2025.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Shuhao Cao. Choose a transformer: Fourier or galerkin. *Advances in Neural Information Processing Systems*, 34:24924–24940, 2021.

Yifan Chen, Qi Zeng, Heng Ji, and Yun Yang. Skyformer: Remodel self-attention with gaussian kernel and nyström method. *Advances in Neural Information Processing Systems Workshop on Efficient Natural Language and Speech Processing*, 2021.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

Zihang Dai, Guokun Lai, Yiming Yang, and Quoc V. Le. Funnel-transformer: Filtering out sequential redundancy for efficient language processing. *arXiv preprint arXiv:2006.03236*, 2020.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

Erik R. Denlinger, Jeff Irwin, and Pan Michaleris. Thermomechanical modeling of additive manufacturing large parts. *Journal of Manufacturing Science and Engineering*, 136(6):061007, 10 2014. ISSN 1087-1357. doi: 10.1115/1.4028669. URL `https://doi.org/10.1115/1.4028669`.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL `https://aclanthology.org/N19-1423/`.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, G Heigold, S Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.

Kevin Ferguson, Yu-hsuan Chen, Yiming Chen, Andrew Gillman, James Hardin, and Levent Burak Kara. Topology-agnostic graph U-Nets for scalar field prediction on unstructured meshes. *Journal of Mechanical Design*, 147(4):041701, 2025.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

Zhongkai Hao, Zhengyi Wang, Hang Su, Chengyang Ying, Yinpeng Dong, Songming Liu, Ze Cheng, Jian Song, and Jun Zhu. GNOT: A general neural operator transformer for operator learning. In *International Conference on Machine Learning*, pp. 12556–12569. PMLR, 2023.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016.

Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, et al. Perceiver IO: A general architecture for structured inputs & outputs. *arXiv preprint arXiv:2107.14795*, 2021a.

Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In *International Conference on Machine Learning*, pp. 4651–4664. PMLR, 2021b.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pp. 5156–5165. PMLR, 2020.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

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

Joseph G. Lambourne, Karl D.D. Willis, Pradeep Kumar Jayaraman, Aditya Sanghi, Peter Meltzer, and Hooman Shayani. Brepnet: A topological message passing system for solid models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12773–12782, June 2021.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 3744–3753, 2019.

Zijie Li, Kazem Meidani, and Amir Barati Farimani. Transformer for partial differential equations' operator learning. *arXiv preprint arXiv:2205.13671*, 2022.

Zijie Li, Dule Shu, and Amir Barati Farimani. Scalable transformer for PDE surrogate modeling. *Advances in Neural Information Processing Systems*, 36:28010–28039, 2023a.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations, 2021. URL `http://arxiv.org/abs/2010.08895`. arXiv:2010.08895.

Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. Fourier neural operator with learned deformations for PDEs on general geometries. *Journal of Machine Learning Research*, 24(388): 1–26, 2023b.

Zongyi Li, Nikola Kovachki, Chris Choy, Boyi Li, Jean Kossaifi, Shourya Otta, Mohammad Amin Nabian, Maximilian Stadler, Christian Hundt, Kamyar Azizzadenesheli, et al. Geometry-informed neural operator for large-scale 3D PDEs. *Advances in Neural Information Processing Systems*, 36:35836–35854, 2023c.

Xuan Liang, Qian Chen, Lin Cheng, Devlin Hayduke, and Albert C To. Modified inherent strain method for efficient prediction of residual deformation in direct metal laser sintered components. *Computational Mechanics*, 64(6):1719–1733, 2019.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL `http://arxiv.org/abs/1711.05101`. arXiv:1711.05101.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021.

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 *Forty-second International Conference on Machine Learning*, 2025.

Xuezhe Ma, Chunting Zhou, Xiang Kong, Junxian He, Liangke Gui, Graham Neubig, Jonathan May, and Luke Zettlemoyer. Mega: Moving average equipped gated attention. *arXiv preprint arXiv:2209.10655*, 2022.

Pablo Miralles-González, Javier Huertas-Tato, Alejandro Martín, and David Camacho. On the locality bias and results in the long range arena. *arXiv preprint arXiv:2501.14850*, 2025.

NVIDIA PhysicsNeMo Team. Transformer models for external aerodynamics on irregular meshes, 2026. URL `https://docs.nvidia.com/physicsnemo/26.03/physicsnemo/examples/cfd/external_aerodynamics/transformer_models/README.html`. NVIDIA PhysicsNeMo Framework documentation. Accessed April 30, 2026.

A Paszke. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019.

Zhen Qin, Xiaodong Han, Weixuan Sun, Dongxu Li, Lingpeng Kong, Nick Barnes, and Yiran Zhong. The devil in linear transformer. *arXiv preprint arXiv:2210.10340*, 2022a.

Zhen Qin, Weixuan Sun, Hui Deng, Dongxu Li, Yunshen Wei, Baohong Lv, Junjie Yan, Lingpeng Kong, and Yiran Zhong. cosformer: Rethinking softmax in attention. *arXiv preprint arXiv:2202.08791*, 2022b.

Joni Reijonen, Alejandro Revuelta, Sini Metsä-Kortelainen, and Antti Salminen. Effect of hard and soft re-coater blade on porosity and processability of thin walls and overhangs in laser powder bed fusion additive manufacturing. *The International Journal of Advanced Manufacturing Technology*, 130(5):2283–2296, 2024.

Benjamin Shih, Ahmad Peyvan, Zhongqiang Zhang, and George Em Karniadakis. Transformers as neural operators for solutions of differential equations with finite regularity, 2024. URL `https://arxiv.org/abs/2405.19166`.

Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, volume 11006, pp. 369–386. SPIE, 2019.

Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. Sparse sinkhorn attention. *arXiv preprint arXiv:2002.11296*, 2020.

Yi Tay, Dara Bahri, Donald Metzler, Da-Cheng Juan, Zhe Zhao, and Che Zheng. Synthesizer: Rethinking self-attention in transformer models. In *International Conference on Machine Learning*, pp. 10183–10192. PMLR, 2021a.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long range arena: A benchmark for efficient transformers. In *International Conference on Learning Representations*, 2021b.

Karn Tiwari, Niladri Dutta, NM Anoop Krishnan, and Prathosh AP. Latent mamba operator for partial differential equations. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 59639–59667, 2025.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Tian Wang and Chuang Wang. Latent neural operator for solving forward and inverse pde problems. *arXiv preprint arXiv:2406.03923*, 2024a.

Tian Wang and Chuang Wang. Latent neural operator for solving forward and inverse PDE problems, June 2024b. URL `http://arxiv.org/abs/2406.03923`. arXiv:2406.03923 [cs, math].

Tianshu Wen, Kookjin Lee, and Youngsoo Choi. Geometry aware operator transformer as an efficient and accurate neural surrogate for pdes on arbitrary domains. In *Advances in Neural Information Processing Systems*, 2025.

Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. Transolver: A fast transformer solver for pdes on general geometries. *arXiv preprint arXiv:2402.02366*, 2024.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On layer normalization in the transformer architecture. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 10524–10533, 2020.

Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. Nyströmformer: A nyström-based algorithm for approximating self-attention. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 14138–14148, 2021.

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33:17283–17297, 2020.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL `https://arxiv.org/abs/1910.07467`.

Hangwei Zhang, Chun Kang, Yan Wang, and Difan Zou. F-Adapter: Frequency-adaptive parameter-efficient fine-tuning in scientific machine learning. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL `https://proceedings.neurips.cc/paper_files/paper/2025/hash/a12e362d89d4e0b40760f839f91550ee-Abstract-Conference.html`.

Michael Zhang, Kush Bhatia, Hermann Kumbong, and Christopher Ré. The hedgehog & the porcupine: Expressive linear attentions with softmax mimicry. *arXiv preprint arXiv:2402.04347*, 2024.

Wentai Zhang, Brandon Abranovic, and Jacob Hanson-Regalado. Flaw detection in metal additive manufacturing using deep learned acoustic features. *Smart and Sustainable Manufacturing Systems*, 2018.

Jianwei Zheng, Wei Li, Ni Xu, Junwei Zhu, Xiaoxu Lin, and Xiaoqin Zhang. Alias-free mamba neural operator. In *Advances in Neural Information Processing Systems*, 2024.

Ming Zhong and Zhenya Yan. Efficient high-accuracy pdes solver with the linear attention neural operator. *arXiv preprint arXiv:2510.16816*, 2025.

A Extended related work

A.1 Set Transformer / ISAB

Set Transformer (Lee et al., 2019) introduced the Induced Set-Attention Block (ISAB), a latent-bottleneck attention mechanism for reducing the cost of global communication over sets. The central distinction between ISAB and FLARE is how much processing is placed around the latent bottleneck: ISAB uses full attention blocks for both directions of the bottleneck, while FLARE keeps only the routing attentions needed to form an explicit low-rank matrix-operator on the input tokens.

ISAB maps the input tokens to a learned latent array and then maps back to the input set as follows: with $I \in \mathbb{R}^{M\times C}$ and $X \in \mathbb{R}^{B\times N\times C}$,

$$H = \mathrm{AttentionBlock}(I, X) \in \mathbb{R}^{B\times M\times C}, \qquad Y = \mathrm{AttentionBlock}(X, H) \in \mathbb{R}^{B\times N\times C}. \tag{10}$$

Both the encode and decode directions in ISAB are full cross-attention blocks. Consequently, ISAB is a complete set-to-set block rather than a token-mixer module with a separate standard feedforward boundary. There is no canonical ISAB mixer that can be substituted into the token-mixing position of a pre-norm transformer while holding the surrounding FFN fixed; extracting only its attention operations would define a new hybrid architecture rather than the original ISAB. Schematically, for query input A and context input B,

$$U = \mathrm{LN}\left(A + \mathrm{CrossAttention}(A, B)\right), \qquad \mathrm{AttentionBlock}(A, B) = \mathrm{LN}\left(U + \mathrm{FFN}(U)\right). \tag{11}$$

Thus the global-communication operator induced by ISAB is not simply the product of two attention matrices; it also includes learned projections, head recombination, residual streams, layer normalization, and feed-forward transformations.

FLARE isolates the routing operation from the surrounding transformer-block machinery. For each head, the routing path consists of two attention calls,

$$Z_h = \mathrm{SDPA}(Q_h, K_h, V_h), \qquad Y_h = \mathrm{SDPA}(K_h, Q_h, Z_h), \tag{12}$$

which gives the induced input-space communication matrix-operator

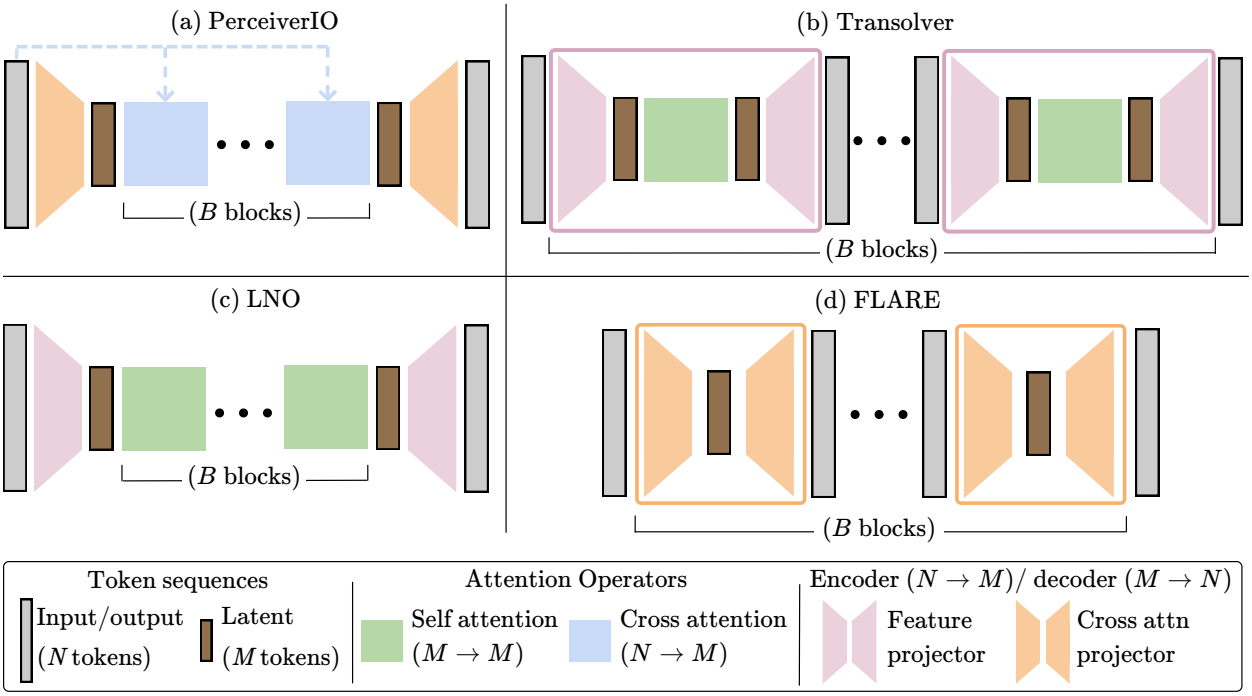
$$Y_h = W_{\mathrm{dec},h} W_{\mathrm{enc},h} V_h, \qquad \mathrm{rank}\left(W_{\mathrm{dec},h} W_{\mathrm{enc},h}\right) \le M. \tag{13}$$

The resulting layer is a low-rank analog of self-attention: the communication matrix-operator is explicit, its rank is controlled by the number of latents, and its spectrum can be analyzed directly. Empirically, Set Transformer / ISAB is stronger than PerceiverIO in Table 1, but FLARE performs better while using roughly one third as many parameters. Together with the ablations in Appendix G, these results indicate that repeated latent routing is the dominant useful mechanism in this PDE-surrogate setting, while additional latent-space processing is not necessary for strong performance.

A.2 Comparison with PerceiverIO, Transolver, and LNO

Figure 6 provides a structural and computational comparison between PerceiverIO Jaegle et al. (2021a), Transolver Wu et al. (2024), Latent Neural Operator (LNO) Wang & Wang (2024a), and FLARE. All four architectures introduce a latent bottleneck to reduce the quadratic cost of self-attention, but they differ fundamentally in how latent tokens are used, how information is mixed across tokens, and how the resulting attention matrix-operator is realized and implemented.

Perceiver and PerceiverIO. Perceiver and PerceiverIO (Jaegle et al., 2021b;a) introduce a general latent-attention framework in which a fixed-size latent array of length M attends to an input sequence of length N via cross-attention. This step maps the input into latent space, after which multiple layers of latent self-attention are applied. Optionally, PerceiverIO performs additional cross-attention from the latent array back to the input or output tokens. This design reduces the cost of attention on the input sequence and enables flexible input and output modalities.



Criteria	PerceiverIO	Transolver	LNO	FLARE
Asymptotic evaluation cost*	$2MN + BMN$	$2BMN + BM^2$	$2MN + BM^2$	$2BMN$
Sequential enc/dec blocks	✗	✓	✗	✓
Headwise parallel enc/dec	✓	✓†	✗	✓
Encoder/decoder coupling	✗	✓	✓	✓
Latent space attention	✓	✓	✓	✗
Fused attention compat.	✓	✗	✗	✓

*Evaluating cost of processing $N \gg M$ tokens ignoring feedforward layers (FFN are $\mathcal{O}(N)$).
†Physics Attention in Transolver uses the same projection weights for all heads, whereas PerceiverIO and FLARE use cross-attention projection where each head learns a distinct slice of the latent tokens.

Figure 6: Comparison of (a) PerceiverIO (Jaegle et al., 2021a), (b) Transolver (Wu et al., 2024), (c) LNO (Wang & Wang, 2024b), and (d) FLARE (ours). Each model contains B blocks and M latents. The criteria in the table are described in detail in Appendix A.

However, Perceiver-style models use latent tokens as a *computational workspace*: information is repeatedly transformed within the latent space through self-attention. As a result, the effective input-input communication pattern is a deep composition of latent transformations rather than a single, explicit attention matrix-operator. While expressive, this structure makes it difficult to characterize the induced token-to-token interactions or to interpret the model as implementing a specific low-rank attention matrix-operator on the input sequence.

Transolver and Latent Neural Operator. Transolver (Wu et al., 2024) and Latent Neural Operator (LNO) (Wang & Wang, 2024a) adapt latent-attention ideas specifically for PDE surrogate modeling on point clouds and meshes. Both architectures project variable-length inputs into a fixed-length latent representation and use latent self-attention to aggregate global information. A key difference from PerceiverIO is that Transolver performs projection and unprojection in *every* transformer block, allowing global context to accumulate progressively with depth. LNO, by contrast, performs a single projection-unprojection step followed by latent self-attention.

In practice, both Transolver and LNO rely on feature-projection mechanisms that explicitly construct or manipulate $M \times N$ projection weights, typically via feature expansion followed by attention or linear layers. These projection layers introduce additional memory and compute overhead that becomes significant for large

N and M. Moreover, both models rely on latent-space self-attention as the primary mechanism for global mixing, which introduces an $\mathcal{O}(M^2)$ cost per block.

We note that Transolver can be instantiated with or without convolutional layers. Without convolution, point-to-point communication relies solely on physics attention; with convolution, local neighborhood aggregation is explicitly injected before attention. We evaluate both configurations separately to disentangle the effects of convolutional inductive bias and latent attention. Our results in Table 1 show that Transolver without convolution performs poorly, indicating that its physics-attention mechanism alone is insufficient for effective global communication. When convolutions are included, local aggregation significantly improves performance, suggesting that much of the gain arises from convolutional preprocessing rather than latent attention itself. In contrast, FLARE does not rely on any convolutional layers and achieves strong performance using attention-based token mixing alone.

FLARE: latent routing as a low-rank matrix-operator. FLARE differs from prior latent-attention architectures in how latent tokens are used. Rather than serving as a computational workspace, latent tokens in FLARE act purely as a *routing mechanism*. Each block consists of exactly two cross-attention operations (encode and decode) that together induce an explicit rank-$\leq M$ matrix-operator on the input tokens. FLARE eliminates latent-space self-attention entirely, ensuring that all token mixing occurs in the input space through a single encode-decode factorization.

A further distinction is FLARE's use of *head-wise independent latent slices*. Each attention head projects into its own latent subspace, forming multiple parallel low-rank projection-reconstruction pathways. This contrasts with Transolver, which shares projection weights across heads, and LNO, which uses a single global projection. As shown in our spectral analysis and ablations, head-wise independence produces different effective-rank profiles across heads and improves accuracy relative to a shared-latent design.

Summary of architectural trade-offs. As summarized in Figure 6, FLARE combines several desirable properties of prior approaches while avoiding their key limitations. Like Transolver, it performs encode-decode operations in every block, enabling progressive global mixing. Like PerceiverIO, it relies on cross-attention and is fully compatible with fused SDPA kernels. Unlike all three prior methods, FLARE eliminates latent self-attention and uses head-wise independent latent routing, yielding a simple, interpretable, and strictly linear-time attention matrix-operator. These differences explain both FLARE's scalability to extreme sequence lengths and its strong empirical performance.

Below we briefly clarify the criteria used in Figure 6. Asymptotic evaluation cost refers to the dominant attention-related complexity for processing $N \gg M$ tokens with B blocks, ignoring feed-forward layers which are linear in N. The remaining criteria describe architectural structure rather than cost. Sequential encode-decode blocks indicate whether global context is progressively refined across depth (as in Transolver and FLARE). Head-wise parallel encode-decode denotes whether each attention head learns an independent latent projection, which is true for PerceiverIO and FLARE but not for Transolver or LNO. Encoder/decoder coupling refers to whether encoding and decoding projections are tied, and latent space attention indicates the presence of explicit self-attention over latent tokens.

Finally, fused-attention compatibility captures whether all attention operations can be expressed using standard scaled dot-product attention (SDPA) primitives. PerceiverIO and FLARE satisfy this property, enabling efficient fused implementations, whereas Transolver and LNO rely on additional projection operations that limit hardware efficiency at large N. Note that Transolver can employ fused SDPA kernels for latent-space self-attention ($\mathcal{O}(M^2)$). However, encoding and decoding ($\mathcal{O}(NM)$) operations, which form the bottleneck in these computations, are implemented with linear projections and cannot employ SDPA kernels.

B Architecture details

B.1 Input and output projection

ResMLP. We implement a residual MLP block to serve as a flexible non-linear function approximator. Given input/output dimensions C_{i} and C_{o}, the layer first applies a linear transformation to a hidden space of

size C_h, followed by L residual layers, each consisting of a linear layer with GELU activation (Hendrycks & Gimpel, 2016). These are the only instances of pointwise nonlinear activations in the model. An optional input residual connection is applied after the first layer when $C_\mathrm{i} = C_\mathrm{h}$, and an optional output residual connection is applied at the end when $C_\mathrm{h} = C_\mathrm{o}$. The final output is projected to dimension C_o via a linear layer. This design allows control over depth and expressivity while preserving stability through residual connections.

Input projection. The input projection consists of a ResMLP with $L = 2$, where C_i is the input feature dimension and $C_\mathrm{h} = C_\mathrm{o} = C$, the feature dimension of the model.

Output projection. For the PDE surrogate models, the output projection consists of a normalization layer followed by a ResMLP with $C_\mathrm{i} = C$, $L = 2$, and C_o equal to the output label dimension. As elsewhere in the PDE model, this normalization layer is instantiated as LayerNorm (Ba et al., 2016) in full precision and RMSNorm (Zhang & Sennrich, 2019) in mixed precision.

B.2 FLARE block

The FLARE block illustrated in Figure 1 (left) and described in Section 3 consists of the FLARE token mixer together with a pointwise feedforward module and a normalization layer in a pre-norm layout.

Because we evaluate FLARE on both PDE surrogate regression and the secondary Long Range Arena transfer tasks, we support two benchmark-specific instantiations that share the same overall block structure:

- The **standard LRA variant** uses the shallow setting from our LRA backend. Its FFN is a two-layer pointwise MLP with width expansion $4C$ (equivalently $C \to 4C \to \mathrm{GELU} \to C$). Its key/value Project modules use the linear $C \to C$ setting, and q/k normalization is enabled. The Norm layers are instantiated as LayerNorm (Ba et al., 2016) for full-precision training and RMSNorm (Zhang & Sennrich, 2019) for mixed-precision training.
- The **PDE variant** instantiates both the FFN and the key/value Project modules as residual MLPs of depth 3 (five linear layers total counting input and output layers) to better capture smooth, function-approximation-style mappings. These deeper modules are a major accuracy driver on the PDE benchmarks (see Appendix G for ablation on residual mlp depth), but they also increase both parameter count and runtime relative to the linear key/value projections used in the LRA variant; see Table 3. Accordingly, in the PDE experiments we reduce FLARE's channel dimension to keep its parameter budget comparable to Transolver without convolution; the exact settings are listed in Table 6. For normalization, we use LayerNorm (Ba et al., 2016) in full-precision (FP32) training and RMSNorm (Zhang & Sennrich, 2019) in mixed-precision training (FP16/BF16 in the forward pass, FP32 in the backward pass).

FLARE. FLARE consists of an encode-decode token operation described in Figure 3 and an output projection; the main differences between the two variants above are the instantiations of the FFN, key/value projection, and normalization modules. Figure 7 presents a mathematically equivalent PyTorch implementation for multi-head token-mixing operation without the fused SDPA kernels. The primary memory bottleneck in this implementation is materializing the $M \times N$ encoding weights and the $N \times M$ decoding weights. Its storage requirement is, thus, $\mathcal{O}(MN)$. Finally, the output projection is set to a single linear layer.

C Spectral analysis

C.1 Eigenanalysis procedure

We exploit the low-rank structure of the global communication matrix $W = W_\mathrm{decode} \cdot W_\mathrm{encode} \in \mathbb{R}^{N \times N}$ to obtain its eigendecomposition in $\mathcal{O}(M^3 + M^2 N)$ time, compared to the $\mathcal{O}(N^3)$ cost for a dense communication matrix. We find the eigendecomposition of the FLARE attention matrix without actually forming the $N \times N$

Table 3: Forward+backward speedups (H100 80GB) at different sequence lengths comparing deep residual key/value projection modules against linear $C \to C$ projection modules at fixed latent sizes. The deep variants are used in the PDE benchmarks, while the linear alternatives are the ones used on Long Range Arena.

Model	100k	500k	1m
Softmax Attention	1.0x	1.0x	1.0x
FLARE (M=128, deep KV)	21.1x	123.8x	255.5x
FLARE (M=128, linear KV proj)	30.9x	169.0x	345.1x
FLARE (M=2048, deep KV)	13.3x	73.6x	150.4x
FLARE (M=2048, linear KV proj)	16.5x	87.4x	174.5x

```python
import torch.nn.functional as F
def flare_multihead_mixer_inefficient(Q, K, V):

    # Args - Q: [H M D], K, V: [B H N D]
    # Ret - Y: [B H N D]

    # Compute projection weights
    scores = Q @ K.mT                        # [B H M N]
    W_encode = F.softmax(scores, dim=-1)     # [B H M N]
    W_decode = F.softmax(scores.mT, dim=-1) # [B H N M]

    # Encode: Project to latent sequence (M tokens)
    Z = W_encode @ V

    # Decode: Project back to input space (N tokens)
    Y = W_decode @ Z

    return Y
```

Figure 7: Pseudocode of FLARE if attention kernel is not available. See Figure 3 for efficient implementation.

matrix. We first note that W_{encode} and W_{decode} can be written in terms of the exponentiated score matrix $A = \exp\big(Q \cdot K^T\big) \in \mathbb{R}^{M \times N}$ as

$$W_{\text{encode}} = \Lambda_M \cdot A, \text{ and } W_{\text{decode}} = \Lambda_N \cdot A^T, \tag{14}$$

where $\Lambda^M \in \mathbb{R}^{M \times M}$, and $\Lambda^N \in \mathbb{R}^{N \times N}$ are diagonal matrices whose entries are

$$[\Lambda_M]_m = \frac{1}{\sum_{n=1}^{N} [A]_{m,n}}, \text{ and } [\Lambda_N]_n = \frac{1}{\sum_{m=1}^{M} [A]_{m,n}}. \tag{15}$$

Thus we have

$$W = \Lambda_N A^T \Lambda_M A \tag{16}$$

as the low-rank attention matrix. We observe that W is similar to $J^T J \in \mathbb{R}^{N \times N}$ where $J = \Lambda_M^{1/2} A \Lambda_N^{1/2} \in \mathbb{R}^{M \times N}$. This is because

$$\begin{aligned} W = \Lambda_N A^T \Lambda_M A &= \underbrace{(\Lambda_N^{1/2} \Lambda_N^{1/2})}_{\Lambda_N} A^T \underbrace{(\Lambda_M^{1/2} \Lambda_M^{1/2})}_{\Lambda_M} A \underbrace{(\Lambda_N^{-1/2} \Lambda_N^{1/2})}_{I_N} \\ &= \Lambda_N^{1/2} \underbrace{(\Lambda_N^{1/2} A^T \Lambda_M^{1/2})}_{J^T} \underbrace{(\Lambda_M^{1/2} A \Lambda_N^{1/2})}_{J} \Lambda_N^{-1/2}. \end{aligned} \tag{17}$$

Thus $J^T J$ is symmetric, positive semi-definite, with rank at most M. Now suppose a singular value decomposition of J as $J = U \Sigma V^T$ where $U \in \mathbb{R}^{M \times M}$ and $V \in \mathbb{R}^{N \times M}$ are the matrices whose columns are

Algorithm 1 Eigenvalues and Eigenvectors from Q, K

Require: $Q \in \mathbb{R}^{M\times D}$, $K \in \mathbb{R}^{N\times D}$
1: $A \leftarrow \exp\big(Q \cdot K^T\big)$
2: $L_N \leftarrow \mathrm{diag}(1/\sum_{m=1}^{M}[A]_{m,n})$
3: $L_M \leftarrow \mathrm{diag}(1/\sum_{n=1}^{N}[A]_{m,n})$
4: $J \leftarrow L_M^{1/2} \cdot A \cdot L_N^{1/2}$
5: Compute SVD: $U\Sigma^2 U^T \leftarrow JJ^T \in \mathbb{R}^{M\times M}$
6: Eigenvalues $\leftarrow \Sigma^2$
7: Eigenvectors $\leftarrow L_N^{1/2} J^T U \Sigma^{-1}$

the left and right singular vectors of J respectively, and $\Sigma \in \mathbb{R}^{M\times M}$ is the diagonal matrix of singular values. Then, we obtain

$$J^T J = V\Sigma \underbrace{U^T U}_{I_M} \Sigma V^T = V\Sigma^2 V^T, \tag{18}$$

and

$$W = \Lambda_N^{1/2} V \Sigma^2 V^T \Lambda_N^{-1/2}. \tag{19}$$

Post-multiplying both sides by $\Lambda_N^{1/2} V$, we have

$$W(\Lambda_N^{1/2} V) = (\Lambda_N^{1/2} V)\Sigma^2. \tag{20}$$

Therefore, the M nonzero eigenvalues of W are the squares of the singular values of J, and the corresponding eigenvectors are the columns of $\Lambda_N^{1/2} V$. Obtaining the eigenvalues and eigenvectors of W this way requires the singular value decomposition of $J \in \mathbb{R}^{M\times N}$. We can do better by relating V and Σ to the eigen decomposition of JJ^T. Consider the matrix $JJ^T \in \mathbb{R}^{M\times M}$ with singular value decomposition, we have

$$JJ^T = U\Sigma \underbrace{V^T V}_{I_M} \Sigma U^T = U\Sigma^2 U^T. \tag{21}$$

We note that the nonzero eigenvalues of W are the same as the singular values of JJ^T. To obtain the eigenvectors of W, we need an expression for V in terms of U, J, and Σ. We do so by noting that

$$J^T U = (V\Sigma U^T)U = V\Sigma \implies V = J^T U \Sigma^{-1}. \tag{22}$$

Therefore, the eigenvectors of W are

$$\Lambda_N^{1/2} V = \Lambda_N^{1/2} J^T U \Sigma^{-1}. \tag{23}$$

To find the eigenvalues and eigenvectors of W, one only needs to compute the eigen-decomposition of the $M \times M$ matrix JJ^T. The overall algorithm, summarized in Algorithm 1, takes $\mathcal{O}(M^3 + NM^2)$ time where the $\mathcal{O}(M^3)$ is for computing the SVD of JJ^T.

C.2 Qualitative Analysis

The matrix JJ^T, being $M \times M$, captures the structure of this latent space and provides insights into how these M dimensions contribute to the attention mechanism. Large eigenvalues correspond to dominant latent dimensions that contribute significantly to attention patterns. If some eigenvalues are small or zero, those latent dimensions contribute little, suggesting redundancy in the latent space. The number of nonzero eigenvalues gives the effective rank of W, which reflects how many independent patterns the attention mechanism captures.

Figure 13 (middle) presents the $M = 64$ nonzero eigenvalue spectra of an FLARE model trained on the elasticity dataset with $M = 64$ latents. Some observations are as follows. In all blocks, especially block 1, the eigenvalues drop sharply within the first $10 - 20$ indices. This indicates that even though the communication

Table 4: Summary of PDE benchmarks.

Benchmark	#Dim	Grid Type	#Points	#Input/ Output Features	#Train/ Test Cases
Elasticity	2D	Unstructured	972	2 / 1	1000 / 200
Plasticity	2D+Time	Structured	3,131	3 / 4	900 / 80
Darcy	2D	Structured	7,225	1 / 1	1000 / 200
Airfoil	2D	Structured	11,271	2 / 1	1000 / 200
Pipe	2D	Structured	16,641	2 / 1	1000 / 200
DrivAerML-40k	3D	Unstructured	40,000	3 / 1	387 / 97
LPBF	3D	Unstructured	1,000–50,000	3 / 1	1100 / 290

matrices W_h could have rank up to $M = 64$, most of the energy (information) is captured by a much smaller subset of modes. This result validates the hypothesis that the global communication pattern is inherently low-rank.

We also observe that the eigenvalue curves in block 5 and block 8 decay more slowly, retaining more moderate-magnitude eigenvalues beyond index 20–30. This indicates that as depth increases, the effective rank increases and the model uses more of the available projection capacity in later blocks.

Finally, we note that the curves for different heads (colors) have distinct decay patterns, especially in later blocks. This shows that the heads have different spectral energy distributions and use the available rank differently. Eigenvalue decay alone does not establish that the heads span different eigenspaces or perform specialized functional roles.

D Benchmarking and Comparison

D.1 Benchmark metrics

The primary evaluation metric for all benchmarks is the relative $\mathcal{L}_2$ error, which quantifies the normalized discrepancy between the predicted solution $\hat{u}$ and the ground truth solution u over all points in the domain. For a given test sample, the relative L_2 error is defined as:

$$\text{Relative } L_2 = \frac{\|\hat{u} - u\|_2}{\|u\|_2} \tag{24}$$

where $\|\cdot\|_2$ denotes the standard Euclidean norm. For datasets where each sample consists of N points (or grid locations), this expands to:

$$\text{Relative } L_2 = \frac{\left(\sum_{i=1}^{N} (\hat{u}_i - u_i)^2\right)^{1/2}}{\left(\sum_{i=1}^{N} u_i^2\right)^{1/2}}. \tag{25}$$

The reported metric is averaged over all test samples.

D.2 Benchmark datasets

We evaluate all models on a collection of benchmark datasets and our proposed AM dataset, each designed to assess generalization, scalability, and robustness to domain irregularity in PDE surrogate modeling. A summary is provided in Table 4.

Elasticity. This benchmark estimates the inner stress distribution of elastic materials based on their structure. Each sample consists of a tensor of shape 972×2 representing the 2D coordinates of discretized

points, and the output is the stress at each point (972×1). The dataset contains 1,000 training and 200 test samples with different material structures (Li et al., 2023b).

Plasticity. This benchmark predicts the future deformation of a plastic material under impact from an arbitrary-shaped die. It is the only explicitly *time-dependent* PDE benchmark in our main table, and it is defined on a structured mesh rather than an unordered point cloud. We include it to broaden the PDE coverage of the paper, while also using it to highlight the distinction between mesh-connectivity-free token mixers and models that additionally exploit mesh-local neighborhood information.

Darcy. This benchmark models fluid flow through a porous medium. The porous structure is discretized on a 421×421 regular grid, downsampled to 85×85 for main experiments. The output is the fluid pressure at each grid point. There are 1,000 training and 200 test samples with varying medium structures (Li et al., 2021).

Airfoil. This task estimates the Mach number distribution around airfoil shapes. The input geometry is discretized into a structured mesh of shape 221×51, representing deformations of the NACA-0012 profile, and the output is the Mach number at each mesh point. The dataset includes 1,000 training and 200 test samples with unique airfoil designs (Li et al., 2023b).

Pipe. This benchmark predicts horizontal fluid velocity in pipes with varying internal geometries. Each sample is represented as a $129 \times 129 \times 2$ tensor encoding mesh point positions, with the output being the velocity at each mesh point ($129 \times 129 \times 1$). The dataset consists of 1,000 training and 200 test samples, with pipe shapes generated by controlling the centerline (Li et al., 2023b).

DrivAerML-40k. The DrivAerML dataset (Ashton et al., 2024) has high-fidelity automotive aerodynamic simulations, featuring 500 parametrically morphed DrivAer vehicles. CFD simulations are performed on 160 million volumetric mesh grids using hybrid RANS-LES, the highest-fidelity scale-resolving CFD approach routinely deployed by the automotive industry (Ashton et al., 2024). Each sample in the dataset includes a surface mesh with approximately 8.8 million points and corresponding pressure values. Since no official dataset split is provided, an 80/20 random split is used, with 40,000 points sampled per case for training and evaluation.

LPBF. We introduce the laser powder bed fusion (LPBF) additive manufacturing dataset which involves field prediction on complex geometries. In metal additive manufacturing, variations in design geometry can affect the dimensional accuracy of the part and lead to shape distortions. We perform several LPBF process simulations of a set of geometries to obtain the deformation field over the geometry. Out of 19,732 successful simulations, we select a filtered subset with up to 50,000 points per geometry (mean 20,972 points) and divide it into 1,100 training cases and 290 test cases. We train models to learn the Z (vertical) component of the deformation field at each grid point. Additional details are provided in Appendix H.

D.3 Benchmark models and training details

We follow the recommended hyperparameter configuration for LNO, Transolver, and GNOT wherever possible. For consistency, we set the number of blocks to $B = 8$, and strive to match the parameter counts of Transolver without (w/o) convolution for all models. As such, the hidden feature dimension for the vanilla transformer is set to $C = 80$. Its head dimension is set to $D = 16$, and the MLP ratio is 4, which is typical for transformers (Vaswani et al., 2017).

For FLARE, we set the hidden feature dimension to $C = 64$ and use $H = 8$ or 16 heads, which result in a head dimension of $D = 8$ or 4 respectively. We keep FLARE at this width because the deep residual MLPs used for the PDE variant's FFN and key/value projections materially increase parameter count compared with the linear-projection alternative (see Table 3), so reducing the channel dimension is necessary to keep FLARE comparable in size to Transolver without convolution and the other baselines, even though Transolver uses $C = 128$ in the standard benchmarks; in our settings, Transolver with convolution has roughly $4\times$ larger

Table 5: Standard training configuration on PDE datasets. Identical values are grouped for clarity.

Dataset	**Batch Size**	**Weight Decay**	**Learning Rate**	**Epochs**	Loss
Elasticity	2	10^{-5}	10^{-3}	500	Rel. L_2
Darcy	2	10^{-5}	10^{-3}	500	Rel. $L_2 + 0.1\ L_{\mathrm{g}}$
Airfoil	2	10^{-5}	10^{-3}	500	Rel. L_2
Pipe	2	10^{-5}	10^{-3}	500	Rel. L_2
DrivAerML	1	10^{-2}	10^{-3}	500	Rel. L_2
LPBF	1	10^{-4}	10^{-3}	250	Rel. L_2

Note: For Darcy test case, we include an additional gradient regularization term L_g following Transolver (Wu et al., 2024).

Table 6: Model configurations for FLARE for PDE datasets. Identical values are grouped for clarity.

Dataset	**#Heads** (H)	**#Latents** (M)	**#Blocks** (B)	**#Features** (C)
Elasticity	8	64	8	64
Darcy	16	256	8	64
Airfoil	8	256	8	64
Pipe	8	128	8	64
DrivAerML-40k	8	256	8	64
LPBF	16	256	8	64

parameter counts than the no-convolution variant. The exact FLARE hyperparameters for each benchmark are given in Table 6. The number of latents is chosen from $M \in \{64, 128, 256\}$ depending on the problem.

As PerceiverIO was not designed to be a surrogate model, we generously set its channel dimension to $C = 128$, and number of latents to $M = 1,024$. Furthermore, the input and output projections for vanilla transformer, perceiver, and FLARE are held consistent to facilitate an equitable comparison of their point-to-point communication schemes.

We evaluate Transolver, GNOT with the hyperparameter configurations provided in Wu et al. (2024) and LNO on the ones provided in Wang & Wang (2024a) on the 2D test cases. For the remaining problems, we choose the best performing parameter set from the 2D cases.

Unless otherwise stated, all models are trained with the AdamW (Loshchilov & Hutter, 2019) optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.999$) to minimize the relative L_2 error. We use the OneCycleLR (Smith & Topin, 2019) scheduler with 10% of epochs dedicated to warming-up to a learning rate of 10^{-3}, followed by cosine decay. We train on LPBF for 250 epochs, and 500 epochs for all other models. Note that we use gradient clipping with max_norm= 1.0 unless otherwise stated. Unless otherwise stated, the weight decay regularization parameter is set to 10^{-5} for the 2D test cases, 10^{-4} for DrivAerML-40k, and 10^{-4} for LPBF. The batch size is set to 2 for the 2D problems and 1 for the 3D problems unless otherwise stated. Unless otherwise stated, all models are trained in full precision (FP32). Unless otherwise stated, benchmark numbers are reported from a single run under a fixed train/test split and shared training protocol. We use single-run reporting for the largest 3D benchmarks because repeated full training runs are computationally expensive; we therefore interpret small numerical gaps on those tasks cautiously and rely on the broader pattern across datasets, ablations, and scaling measurements.

Vanilla transformer. For the vanilla transformer, we set the hidden feature dimension to $C = 80$, and choose $H = 5$ heads so that the head dimension $D = 16$. The number of blocks is set to $B = 8$, and the MLP ratio for the feedforward block is set to 4. The vanilla transformer can be prohibitively slow for test cases with over 10,000 test points.

PerceiverIO. For PerceiverIO, we use $B = 8$, $C = 128$, $H = 8$, and set the latent sequence length to $M = 512$ for all test cases.

Transolver. Transolver (Wu et al. (2024)) introduces Physics-Attention: each mesh point is softly assigned to a few learnable *slices*, shrinking thousands of points to only tens of tokens. Self-attention runs on this compact set and the tokens are then desliced back to points. Following the hyperparameter recommendations in the code of Wu et al. (2024), Transolver is trained with 30% of the steps dedicated to warm-up, and gradient clipping with max_norm = 0.1. The recommended batch size for Transolver is 1 for elasticity, and 4 for the remaining 2D problems. For the 3D problems, we set the batch size to 1. In the standard benchmark setting we compare against the no-convolution version to match parameter budget as closely as possible; the convolutional variant has substantially more parameters in our setup, so it is not the fairest baseline for the main FLARE comparison.

Transolver++. We evaluated Transolver++ using the latest version of the official code release (`https://github.com/thuml/Transolver_plus`) together with the hyperparameters reported in the paper. All experiments were run using PyTorch 2.7 and CUDA 12.8; the supplementary material provides a script to instantiate the standard environment used for all benchmark runs. In addition to testing the official release directly, we implemented a multi-GPU version for DrivAerML so that Transolver++ would remain a viable large-scale comparison point under our training setup. Nevertheless, we were unable to reproduce the accuracy numbers reported in the paper using the stated hyperparameters. This reproducibility issue has also been noted independently in the authors' GitHub issue tracker, in Appendix B.7 of AB-UPT (Alkin et al., 2025), and in NVIDIA PhysicsNeMo's external-aerodynamics documentation (`https://docs.nvidia.com/physicsnemo/26.03/physicsnemo/examples/cfd/external_aerodynamics/transformer_models/README.html`), which states that they did not observe gains from Transolver++ in their experiments (NVIDIA PhysicsNeMo Team, 2026).

Only a small number of baseline families are even plausible comparison points at the million-point scale. In our view, the main candidates are Transolver/Transolver++ and GAOT (Wen et al., 2025). In practice, however, these come with caveats that make a direct end-to-end 1M-point comparison difficult to interpret fairly: Transolver++ has substantial reproducibility issues; Transolver and Transolver++ were numerically unstable for us under the mixed-precision regime (BF16 forward, FP32 backward) needed at this scale, while standard Transolver also runs out of memory; and GAOT depends on several problem-specific dynamic-graph hyperparameters that are not specified in enough detail for us to reproduce reliably. We do not consider chunked methods such as AB-UPT (Alkin et al., 2025) to be direct baselines for this section, because they test a different capability from direct ingestion of a full 10^6-point context. For this reason, we use DrivAerML-40k for matched baseline comparisons and present the 1M-point study primarily as a characterization of FLARE's scaling behavior.

Latent Neural Operator (LNO). LNO (Wang & Wang (2024b)) moves computation into a small latent space. An embedder lifts the input field, cross-attention compresses N points into M latent tokens, transformers act solely on these tokens, and a decoder maps them to any query location. In line with the recommended hyperparameter configuration in Wang & Wang (2024b), we set $\beta_2 = 0.99$ in the AdamW optimizer, do warmup for the first 20% of epochs, and clip gradient norms greater than 1,000. The number of hidden features is set to $C = 192$ for elasticity, and $C = 128$ for all other test cases. The number of residual layers is set to 3 for elasticity and 4 for other test cases. The number of latent self-attention blocks is 8 for pipe and airfoil test cases and 4 for all other cases. The number of latent modes is set to 256 for all test cases. The batch size during training is set to 4 for the 2D test cases and 1 for the 3D test cases. The LNO code also recommends weight decay regularization of $5 \cdot 10^{-5}$ for the 2D test cases.

In running our experiments, we noticed discrepancies between our LNO results and those presented in their article (Wang & Wang, 2024a). Upon further investigation, we found that the datasets used in the LNO paper and in the original Transolver paper are not the same. For example, in the Elasticity dataset, LNO was trained and tested on a 1000/1000 split, whereas Transolver used a 1000/200 split. In the Darcy dataset, LNO employed a higher resolution of 241×241, compared to 85×85 in Transolver. Because these differences

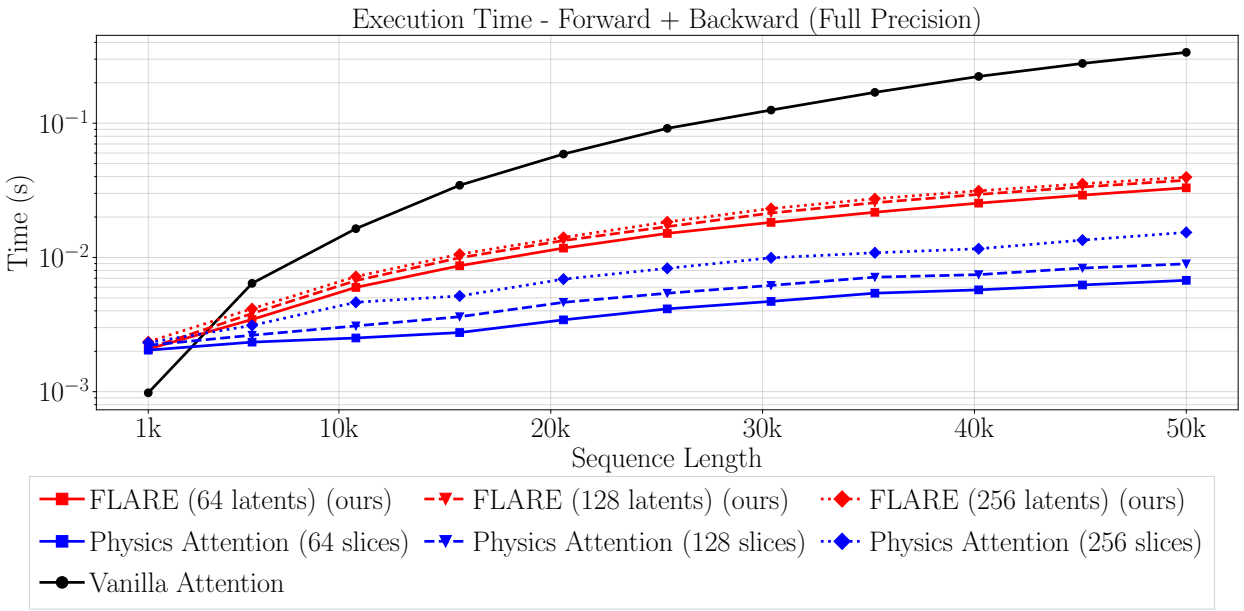


Figure 8: Execution times in FP32 for a single vanilla self-attention layer, a physics attention layer, and FLARE. The models are set to have approximately the same number of parameters as in Section 5.1. This calculation is performed on a single H100 80GB GPU. Note that the curves for FLARE are somewhat overlapping.

make direct comparison unreliable, all models (including LNO and Transolver) were re-trained and evaluated on the standardized Transolver splits and resolutions to ensure fairness.

General Neural Operator Transformer (GNOT). GNOT (Hao et al. (2023)) employs heterogeneous normalized attention, separately normalizing keys and values, to fuse multiple input fields on irregular meshes. A learnable geometric gate decomposes the domain and routes tokens to scale-specific expert MLPs. Linear cost attention plus this gating scales to large problems and surpasses earlier operator learners.

Following the hyperparameter recommendations outlined in the code of Wu et al. (2024), GNOT is trained with 30% of the steps dedicated to warm-up, and gradient clipping with max_norm = 0.1. The recommended batch size for Transolver is 1 for elasticity, and 4 for the remaining 2D problems. The batch size is set to 2 for elasticity, 4 for the remaining 2D benchmarks and 1 for the 3D benchmarks.

FLARE. All attention operations in FLARE are implemented using `scaled_dot_product_attention` (SDPA) in `torch.nn.functional`, as shown in Figure 3; this allows PyTorch to dispatch to fused SDPA backends (e.g., FlashAttention-style kernels) when available. For all problems, we employ $B = 8$ blocks with a feature dimension of $C = 64$. We set the number of residual layers and the number of key/value projection layers to 3, and vary the head dimension as $D \in \{4, 8\}$ and the number of latent tokens as $M \in \{64, 128, 256\}$. The hyperparameters for each test case are presented in Table 6.

D.4 Random-seed variance

To measure sensitivity to random initialization, we repeat the Elasticity benchmark for FLARE and Transolver over ten seeds (0–9). Table 7 and Table 8 report the full per-seed train and test losses.

Table 7: Elasticity losses for FLARE across random seeds ($\times 10^{-3}$). Mean and standard deviation are over ten seed-only runs.

Metric	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Mean**	**Std**
Train loss	2.36	2.29	2.39	2.60	2.35	2.87	2.40	2.43	2.36	2.56	2.46	0.17
Test loss	3.81	3.64	3.68	4.25	3.88	4.56	3.82	4.10	3.71	4.08	3.95	0.29

Table 8: Elasticity losses for Transolver across random seeds ($\times 10^{-3}$). Mean and standard deviation are over ten seed-only runs.

Metric	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Mean**	**Std**
Train loss	4.77	4.98	6.49	9.88	8.33	5.05	5.47	4.44	5.38	6.83	6.16	1.76
Test loss	6.07	6.70	8.81	12.71	10.82	6.85	7.30	6.16	7.15	8.89	8.15	2.18

The multiseed results support the Elasticity comparison in Table 1: FLARE obtains lower test error than Transolver for every seed, with a 10-seed mean of 3.95 compared with 8.15. We therefore report these 10-seed means, rather than a single run, for the FLARE and Transolver w/o conv Elasticity entries in Table 1. The remaining benchmark entries are single-run results and should be read as evidence of broad trends across tasks and model families, not as statistical claims about isolated small gaps.

E Field-prediction on million-point geometries

Experimental setup. We train FLARE on the DrivAerML dataset (Ashton et al., 2024) using meshes subsampled to contain $N = 10^6$ points per geometry. All experiments are run on a single Nvidia H100 80GB GPU with batch size 1. Unless otherwise stated, models use feature dimension $C = 64$ and $H = 8$ attention heads. We vary the number of latent tokens $M \in \{128, 512, 1024, 2048\}$ and the number of FLARE blocks B (as reported in Figure 5).

Training details. All FLARE models in this study are trained in mixed precision (FP16/BF16 in the forward pass, FP32 in the backward pass) to take advantage of fused SDPA backends (e.g., FlashAttention-style kernels). We train for 500 epochs with the OneCycleLR scheduler (Smith & Topin, 2019) where the first 5% of epochs are spent warming up to a learning rate of $5 \cdot 10^{-4}$ followed by cosine decay. No hidden sparsity, memory offloading, or distributed training is used.

F Discussion on the design principles of FLARE.

Latent tokens enable gather-scatter communication. In FLARE, information flows through latent tokens by first *gathering* from the input sequence and then *scattering* back. The encoding step can be understood as a gather (all-reduce) operation, where each latent token pools information from the input according to its learned query pattern. Formally, for latent query q_m,

$$z_m = \sum_{n=1}^{N} \frac{\exp(q_m \cdot k_n)}{\sum_{n'=1}^{N} \exp(q_m \cdot k_{n'})} v_n, \quad m = 1, \ldots, M, \tag{26}$$

z_m aggregates input values v_n with convex weights. When the similarity scores are sharp, z_m emphasizes a few dominant inputs; when they are flatter, z_m acts like a specialized *pooling token* that averages a select set of tokens. Across M latents, this yields a compact set of global descriptors, each specializing in pooling different aspects of the input.

The decoding step is the dual scatter (broadcast) operation, but importantly, the broadcast is *selective*: each latent z_m contributes only to the input tokens that assigned it high weight in the encoding step. Concretely,

$$y_n = \sum_{m=1}^{M} \frac{\exp(k_n \cdot q_m)}{\sum_{m'=1}^{M} \exp(k_n \cdot q_{m'})} z_m, \quad n = 1, \ldots, N, \tag{27}$$

the output token y_n only receives substantial information from the latents whose query pattern matches its key k_n strongly. In this sense, each latent acts as both a selective *pooling hub* and *broadcaster*, routing information back along the pathways that originally recruited it. Together, these gather-scatter stages form a low-rank, butterfly-like transformation: first contracting N tokens into $M \ll N$ global features, and then expanding back to N outputs, enabling efficient, yet expressive, global communication.

Symmetry in latent token communication. The compression–expansion attention structure in FLARE is most effective when the encoding and decoding operators are structurally aligned. Specifically, $W_{\text{encode},h} = \text{softmax}(Q_h \cdot K_h^T)$ and $W_{\text{decode},h} = \text{softmax}(K_h \cdot Q_h^T)$ are transposes of each other up to diagonal scaling. We experimented with breaking this symmetry by using distinct query–key pairs for encoding and decoding, but observed no accuracy gains. This suggests that the near-adjoint relationship between W_{encode} and W_{decode}, both derived from the same parameters, may provide a form of mathematical optimality, ensuring stable information flow through latent tokens while reducing representational redundancy.

Tradeoff between query dynamics and key/value expressivity. An important design choice in FLARE is to use fixed, input-independent queries Q, which constrain the flexibility of the attention pattern. Detaching Q from X allows for the low-rank communication structure, but requires compensating expressivity in the key and value projections. In practice, we find that deeper residual MLPs for K and V are crucial for capturing rich, feature interactions under this constraint. Conversely, one could imagine making Q dynamic by conditioning it on X, which would shift more of the modeling burden to the query side and potentially allow shallower K, V projections. Thus, FLARE embodies a clear tradeoff: fixing Q encourages stability and efficiency, but places greater importance on the depth and expressivity of the key/value encoders.

G Model analysis and ablations

G.1 Controlled token-mixer comparison

The PDE results in Table 1 compare complete surrogate architectures and therefore do not isolate the token mixer. For the approximately parameter-matched comparison between FLARE and Transolver without convolution, Transolver retains its recommended channel dimension $C = 128$, while FLARE uses $C = 64$ to offset the parameter cost of its residual MLPs. The complete configurations are documented in Section 5.1 and Section D.3.

To isolate the effect of the token mixer, we conduct an identical-backbone comparison without the deep residual MLPs used by the PDE-oriented models on the Elasticity benchmark. Only the mixer changes among vanilla self-attention, FLARE, and Transolver. We evaluate both $C = 64$ and $C = 128$; all models use eight heads, linear input/output and query/key/value projections, and the standard FFN with ratio 2 recommended for Transolver. B denotes the number of transformer blocks. Table 9a and Table 9b report the best-checkpoint test relative error in units of 10^{-3}.

(a) $C = 64$

Mixer	$B = 2$	$B = 4$	$B = 8$
Vanilla attention	31.584	**12.356**	**7.454**
FLARE ($M = 32$)	**29.025**	15.014	8.531
FLARE ($M = 64$)	31.088	14.001	7.715
FLARE ($M = 128$)	30.645	12.735	8.330
Transolver (32 slices)	39.890	19.195	12.653
Transolver (64 slices)	40.000	16.591	11.894
Transolver (128 slices)	35.786	16.643	11.258

(b) $C = 128$

Mixer	$B = 2$	$B = 4$	$B = 8$
Vanilla attention	23.925	**7.478**	**4.600**
FLARE ($M = 32$)	21.204	10.915	4.835
FLARE ($M = 64$)	**20.311**	10.746	7.887
FLARE ($M = 128$)	21.166	9.305	4.618
Transolver (32 slices)	28.783	10.576	6.737
Transolver (64 slices)	25.209	11.335	5.870
Transolver (128 slices)	23.794	8.874	5.530

Table 9: Test relative error ($\times 10^{-3}$) in controlled Elasticity comparison with an identical transformer-block backbone at two channel widths. Lower is better.

Across both channel widths, FLARE outperforms the best Transolver configuration in five of the six depth and width settings, demonstrating a clear advantage over Transolver in this controlled Elasticity comparison.

The existing LRA results in Table 2 provide a complementary identical-backbone token-mixer comparison among several models including vanilla attention, Transolver, and FLARE. In this setting, FLARE outperforms Transolver on every task and improves average accuracy from 51.32 to 60.22. We point to that comparison here as additional evidence rather than introduce a new LRA experiment. Together with the Elasticity results above, this shows that FLARE is the strongest efficient token mixer in the controlled comparisons while remaining competitive with vanilla self-attention.

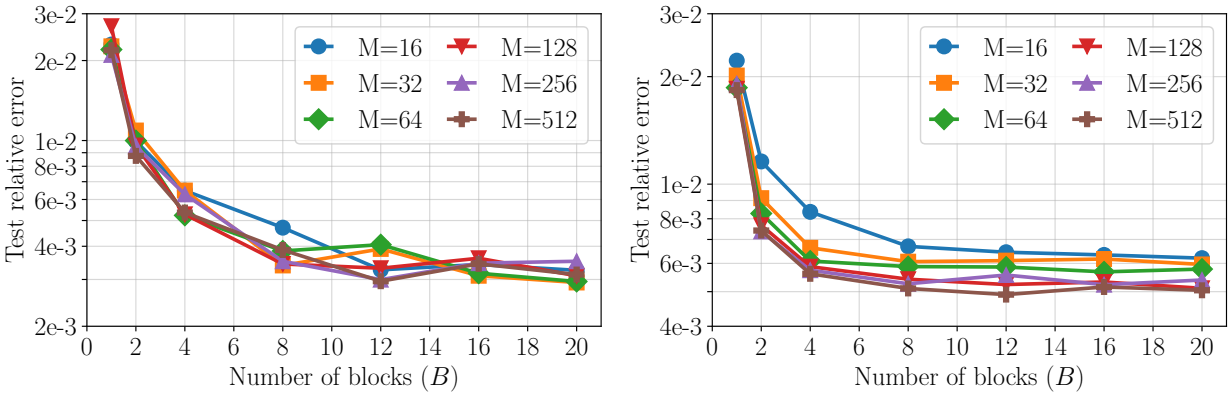


Figure 9: Effect of number of blocks (B) and number of latent tokens (M) on test accuracy on the Elasticity (left) and Darcy (right) test cases. Experiment details are presented in Appendix G.

Time and memory complexity. Figure 2 illustrates the time and memory complexity of a single forward and backward pass for different attention schemes on long sequences. The experiment is done in mixed precision (FP16/BF16 in the forward pass, FP32 in the backward pass) using PyTorch's autocast functionality with $C = 128$ features and $H = 8$ heads for all models. The FlashAttention backend (Dao et al., 2022) is employed for SDPA wherever possible.

Although vanilla self-attention has the lowest memory cost thanks to the FlashAttention algorithm, which eliminates the need to materialize the score matrices ($Q_h \cdot K_h^T$), its compute time still scales poorly with the sequence length. In contrast, the compute time for FLARE exhibits strong scaling with sequence length. Its memory requirement is marginally greater than vanilla attention due to the presence of deep residual networks for key/value projections, and due to the need to materialize Z_h, the latent sequence of M tokens. As these costs are marginal compared to the SDPA operation, the curves for different M values of FLARE are somewhat overlapping. Finally, the compute time for Physics Attention of Transolver (Wu et al., 2024)

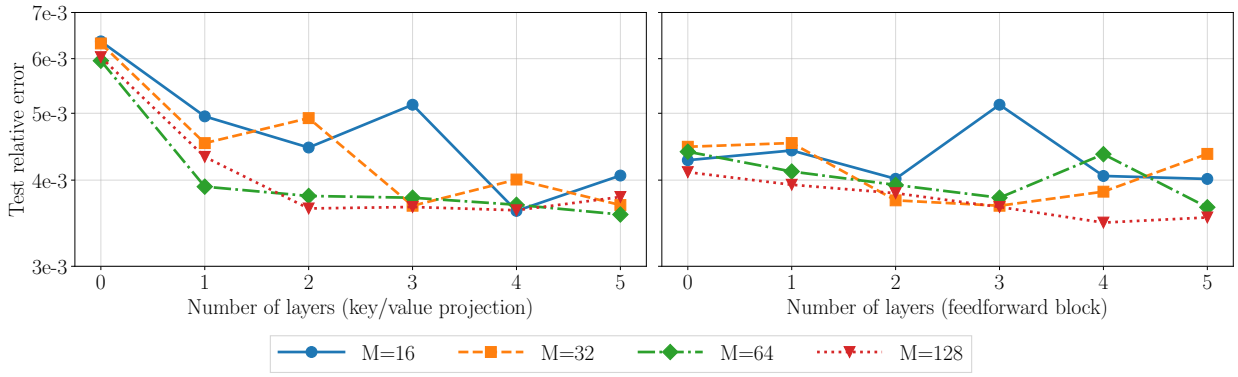


Figure 10: (Left) effect of the number of residual layers in key/ value projection, and (right) of residual layers in residual block on test accuracy. In both cases, deeper networks lead to greater accuracy.

exhibits somewhat good scaling. However, its memory cost and compute time blow up for large slice counts due to the need for materializing the projection matrices.

Number of blocks (B) and latent tokens (M). Figure 9 presents the test relative error of FLARE on the Elasticity (left) and Darcy (right) benchmark datasets as a function of the number of blocks (B) and the number of latent tokens (M). Figure 5 (left) presents the same for the DrivAerML dataset with one million points per geometry. In all cases, we note the favorable trend that relative error consistently decreases as we increase the number of blocks. Similarly, we observe that the relative error generally decreases with M, though the trend is not strictly monotonic. In the Elasticity problem, improvements with rank diminish rapidly, indicating that global communication in that problem is fundamentally low-rank. On the other hand, increasing M monotonically increases performance on the Darcy problem, indicating that the problem is *rank-limited*. This also explains why vanilla transformer with a full-rank attention pattern outperforms rank-deficient FLARE on the Darcy problem. However, the accuracy gain comes at the cost of greater latency as the vanilla transformer is $\sim 5\times$ slower than FLARE on the Darcy problem. Figure 5 indicates that time per epoch (middle) and memory (right) scaling of FLARE with B and M. Here, increasing M leads to increased latency, and that increasing M does not come at the cost of greater memory requirements.

ResMLP depth: key/value projections. A substantial distinction between vanilla self-attention and FLARE is the introduction of deep residual blocks for key and value projections in place of simple linear layers. In standard self-attention, queries (Q_h) and keys (K_h) determine the global communication pattern through $W_h = \mathrm{softmax}(Q_h K_h^T / s)$, while values ($V_h$) carry the information to be communicated. All three are typically computed as shallow linear projections of the input X. In contrast, FLARE computes the attention pattern as $W_h = \mathrm{softmax}(K_h Q_h^T) \cdot \mathrm{softmax}(Q_h K_h^T)$, where the query embeddings Q_h are learned parameters independent of the input. This makes the attention pattern less dynamic, motivating architectural modifications to enhance flexibility.

To address this, we replace the linear key and value projections with deep residual MLPs. Using residual networks for key and value encodings allows each token to learn richer and more structured features rather than shallow embeddings, which is particularly crucial in FLARE since the queries are fixed and cannot adapt to the input. Figure 10 (left) shows the impact of varying the number of residual layers in key/value projections and within the residual block on test accuracy for the elasticity benchmark dataset. We suspect that deeper key/value encodings lead to more meaningful and focused attention, encoding structured inductive priors beneficial to downstream prediction.

ResMLP depth: feedforward block. A second difference between the standard attention block and FLARE block is that we replace the feed-forward block in vanilla self-attention with a deep residual MLP. Preliminary experiments using standard feed-forward blocks led to training instabilities and poor convergence. In contrast, residual MLPs consistently enabled stable training and allowed us to increase model capacity.

Figure 10 (right) indicates that increasing the number of residual layers leads to slight improvements in accuracy. Based on these results, we use three residual layers in both key/value projections and the residual block, as this provides a good trade-off between model capacity and computational cost in all subsequent experiments.

Effect of head dimension and parallel low-rank projections. A core design of FLARE is that each attention head implements an independent rank–$\leq M$ projection–reconstruction pathway. When multiple such pathways operate in parallel, the resulting attention matrix-operator becomes a mixture of low-rank factors. This is in contrast to Transolver, which shares projection weights across all heads, and LNO, which uses a single global projection: both designs restrict the model to learning at most one or a few shared low-rank directions. FLARE, by contrast, gives each head an independently parameterized routing pathway by assigning it a separate slice of the latent tokens.

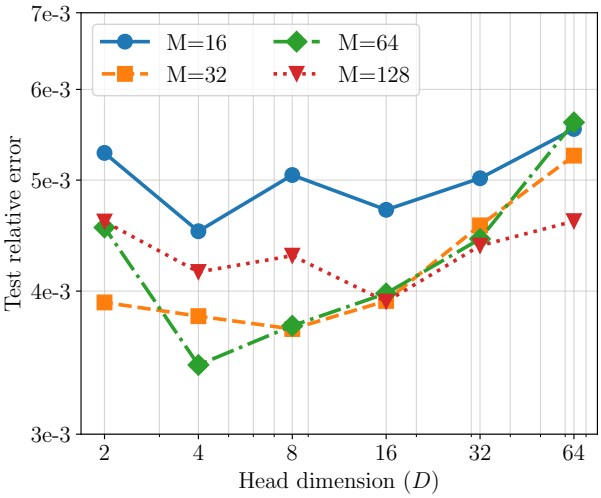


Figure 11: Effect of head dimension (D) on test accuracy. We design FLARE to work optimally for $D = 4$–8.

To examine this hypothesis quantitatively, we vary the number of heads H while keeping the total feature dimension $C = 64$ and number of blocks $B = 8$ fixed, thereby trading off head dimension $D = C/H$ against the number of parallel low-rank projections available to the model. As shown in Figure 11, FLARE consistently achieves the best accuracy for $D = 4$ or $D = 8$, outperforming configurations with larger head dimensions. This behavior is the reverse of what is commonly observed in standard transformers—where typical head dimensions are $D = 16$–32, but is entirely consistent with FLARE's architectural role for each head. Larger D increases the per-head representational capacity but reduces the number of parallel low-rank factors; smaller D yields more independently parameterized projection–reconstruction pathways, which can better approximate a full attention map.

We additionally note a practical implication of this analysis: because the dot-product magnitudes are naturally small for $D \in \{4, 8\}$, we use a scaling factor of 1 rather than the usual $1/\sqrt{D}$ in these cases, simplifying implementation without sacrificing stability. Overall, the head-dimension ablation supports our architectural claim that FLARE benefits from *multiple parallel low-rank projections* and per-head independence.

Ablation on latent-space blocks vs. FLARE encode-decode blocks. To directly investigate whether FLARE's encode-decode mechanism is responsible for the observed performance gains, we conduct a controlled ablation varying the number of latent-space self-attention blocks (L_B) and the number of full FLARE blocks (B). This experiment spans the continuum between a Perceiver/LNO-like architecture (few encode-decode operations and many latent-space blocks; bottom-left of Figure 12) and a FLARE-like architecture (many encode-decode blocks and no latent-space self-attention; top-right).

Across the evaluated configurations, allocating additional parameters and computation to encode-decode blocks provides a better accuracy-efficiency tradeoff than allocating them to latent-space self-attention blocks. Latent-space self-attention may refine the compressed representation, but it adds parameters and $\mathcal{O}(M^2)$ cost without revisiting information discarded by the initial projection. In contrast, additional encode-decode blocks repeatedly mix information between the input and latent domains and steadily improve accuracy. The lowest errors in the entire grid occur in the top-right corner, corresponding to *zero* latent-space blocks and the largest number of FLARE blocks.

This result does not imply that latent-space self-attention is generally useless. It shows that, in this controlled study, the routing operations recover the overwhelming majority of the performance and are a more effective use of the available parameter and compute budget. If one interprets the Perceiver $\leftrightarrow$ FLARE spectrum as

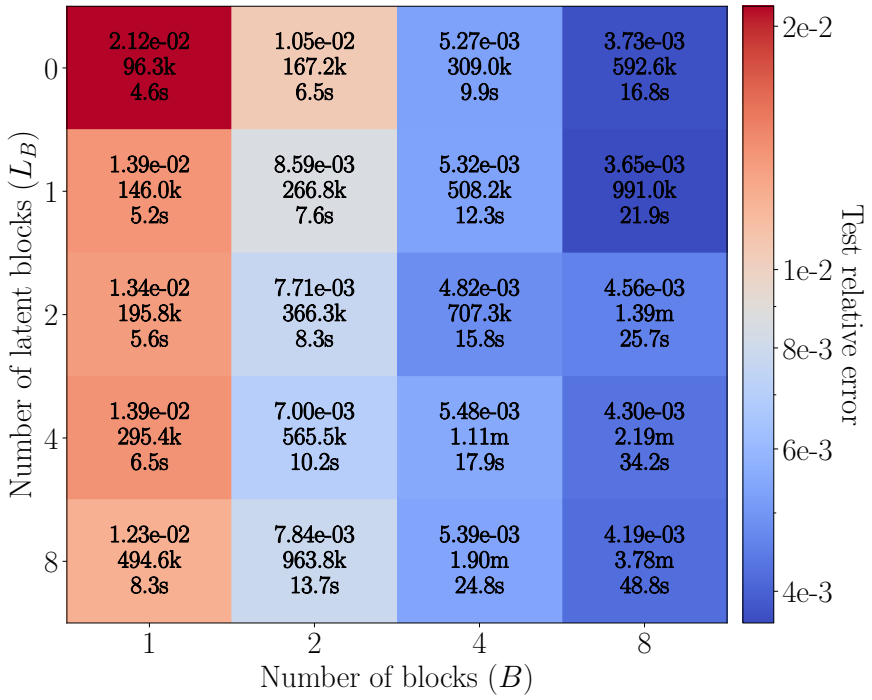


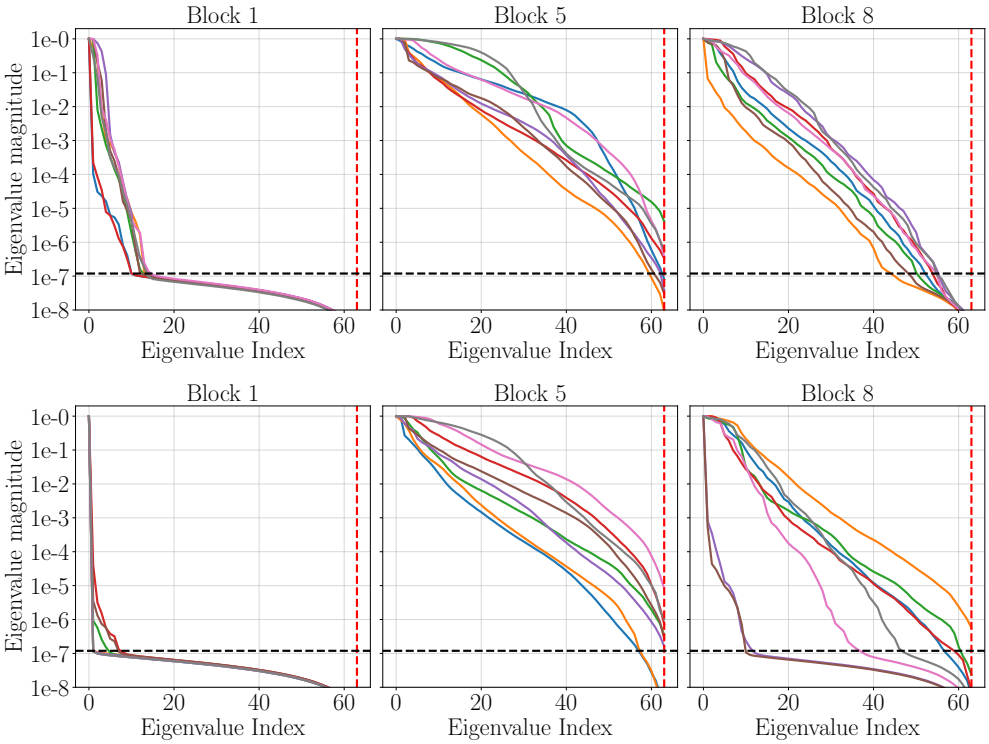
Figure 12: **Ablation on the number of latent-space self-attention blocks (L_B) versus the number of FLARE encode-decode blocks (B).** Each cell reports: (top) test relative error, (middle) parameter count, and (bottom) training time per epoch. The total cost of a model with B FLARE blocks and L_B latent blocks is $\mathcal{O}(B\,NM + BL_B\,M^2)$. For the evaluated budgets, allocating compute to additional FLARE blocks provides a better accuracy-efficiency tradeoff than allocating it to latent-space self-attention blocks. The best-performing regime lies near the *top-right corner*: many encode-decode blocks and *zero* latent-space blocks. This does not imply that latent-space self-attention is generally useless; rather, repeated routing captures most of the benefit more efficiently in this setting.

trading latent refinement for repeated global mixing, the best-performing region for these tasks is FLARE-like: many encode-decode blocks and few or no latent-space blocks.

Ablation on shared vs. independent per-head latent tokens. To assess the effect of head-wise latent diversity, which is central to FLARE's design, we compare two variants: (i) *shared-latent* models, where all heads use the same latent sequence, and (ii) *independent-latent* models, where each head receives its own slice of the latent tokens. This isolates whether FLARE's expressivity arises merely from the low-rank encode-decode structure or whether the ability of different heads to learn distinct low-rank projections provides additional modeling capacity.

The spectral plots in Figure 13 reveal a clear difference between the two settings. With shared latents, all heads exhibit nearly identical eigenvalue decay. In contrast, independent latents produce noticeably different spectra across heads, particularly in deeper blocks, showing different effective-rank and spectral-energy profiles. These differences are modest in early layers but become increasingly pronounced as depth increases. They do not, by themselves, establish differences in eigenspaces or functional roles.

Crucially, the quantitative results mirror this qualitative behavior: across all depths, models with independent per-head latents achieve lower test relative error despite having comparable parameter counts. This provides causal evidence that head-wise independence is beneficial, while leaving the learned function of each head unresolved. Finally, we note that applying the same spectral analysis to Transolver or LNO is not directly meaningful, since their latent-space transformations are nonlinear and not expressible as a single low-rank matrix-operator; FLARE's structure uniquely enables such eigenanalysis.



Blocks	Shared latents	Independent latents
1	**20.8**	21.9
2	10.8	**9.99**
4	5.91	**5.21**
8	4.37	**3.85**

Figure 13: **Ablation of shared vs. independent latent tokens across attention heads.** Each plot shows the M=64 nonzero eigenvalues of the head-specific communication matrices W_h (Eq. 8) for FLARE with B=8 blocks and C=64 features trained on the elasticity dataset. When heads share a single latent sequence (top row), the eigenvalue spectra across heads are nearly identical, indicating similar spectral-energy and effective-rank profiles. When heads use independent latent slices (bottom row), the eigenvalue decay varies noticeably across heads, reflecting more diverse low-rank structures. The accompanying table reports test relative errors for different depths B, showing that models with independent latents consistently achieve lower error.

H Benchmark dataset of additive manufacturing simulations

H.1 Introduction & Background

In metal additive manufacturing (AM), subtle variations in design geometry and process parameter selection may result in undesirable part artifacts or even costly build failures. Numerical simulations of laser powder bed fusion (LPBF), a popular additive manufacturing process, can be used to predict build failures, but this may take several minutes to hours depending on the part size.

Although AM enables the fabrication of a wide variety of new designs, the final product must nonetheless comply with the constraints of the underlying manufacturing process. Metal AM parts often have anisotropic and spatially varying material properties that depend on geometric features (such as overhang, or support structure) and fabrication process parameters (such as laser energy density, hatch spacing, layer thickness, and raster path). As-built parts with thin features often suffer distortions when manufactured with the LPBF process, which uses a high-power laser to selectively melt and fuse metal powder, layer-by-layer,

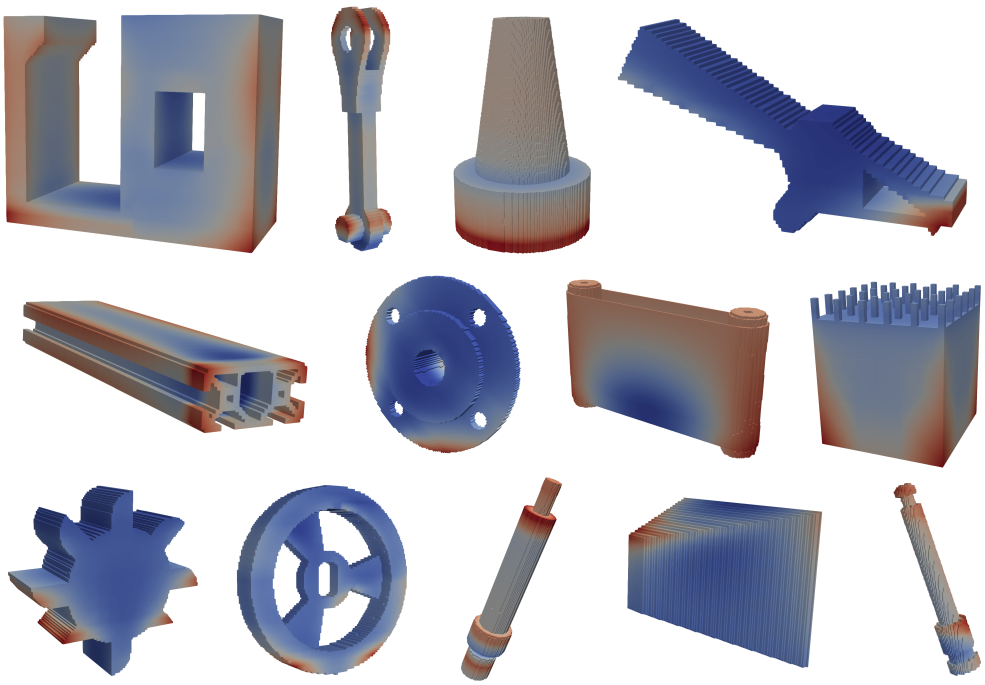


Figure 14: We simulate the LPBF process on selected geometries from the Autodesk segmentation dataset (Lambourne et al., 2021) to generate a benchmark dataset for AM calculations. Several geometries are presented in this gallery. The color indicates Z (vertical) displacement field.

to create complex, high-precision parts. Thermal stresses, termed *residual stresses* (RS), accumulate in LPBF-fabricated parts as a result of rapid thermal cycling due to laser exposure. These stresses can be severe to the point of inducing localized plastic deformations or delamination. As such, due to these *residual deformations*, the final shape of the part may deviate from the designed geometry.

We present a high-fidelity thermomechanical RS calculation data set on the Fusion 360 segmentation dataset, a publicly available dataset of complex 3D geometries (Lambourne et al., 2021). Numerical solvers for simulating the LPBF build process perform expensive quasi-static thermo-mechanical equations,

$$\underbrace{\rho C_p \frac{\mathrm{d}T}{\mathrm{d}t} = \boldsymbol{\nabla} \cdot k \Delta T(\boldsymbol{x}, t) + Q(\boldsymbol{x}, t),}_{\text{thermal transport}} \qquad \underbrace{\boldsymbol{\nabla} \cdot \boldsymbol{\sigma} = 0 \qquad \boldsymbol{\sigma} = \boldsymbol{C}\boldsymbol{\varepsilon}_e,}_{\text{stress equilibrium}} \tag{28}$$

layer-by-layer within a finite element framework (Denlinger et al., 2014; Liang et al., 2019; Autodesk, 2025), and are integrated in commercial software products (Autodesk, 2025). These calculations take several minutes to hours, making them prohibitively expensive for part design scenarios that can involve hundreds of evaluations.

Ferguson et al. (2025) introduced a dataset of LPBF simulations in which multiple finite element calculations were performed on a collection of 3D shapes. That dataset, however, was restricted to relatively coarse meshes with approximately 3,500 points per mesh. The present work extends this line of research by considering larger and more refined meshes, with up to 50,000 grid points.

H.2 Dataset Generation

To generate a dataset of LPBF simulations, we employ Autodesk NetFabb (Autodesk, 2025), a commercially available software tool for numerically simulating RS and associated physics. We begin with the Fusion 360 segmentation dataset (Lambourne et al., 2021), and scale each shape to lie within $[-30, 30] \times [-30, 30] \times [0, 60]$ mm such that it rests atop the build plate at $z \in [-25, 0]$ mm; the parts are not otherwise rotated or

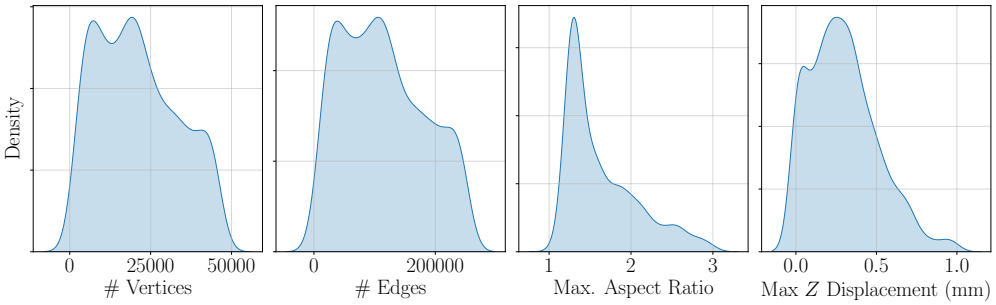


Figure 15: Summary of LPBF dataset statistics.

Table 10: Summary statistics of our proposed LPBF dataset.

	#Points	**#Edges**	**Avg./ max aspect ratio**	**Max height (mm)**	**Max Displacement**
Mean	20,972	114,140	1.6421 / 1.6421	29.429	0.29526
Std.	12,476	68,308	0.43794 / 0.43794	23.246	0.21064
Min	736	2,860	1.0667 / 1.0667	0.60000	0.00048500
25%	10,229	56,208	1.2800 / 1.2800	7.8000	0.13827
50%	19,743	107,680	1.4733 / 1.4733	21.600	0.27075
75%	30,503	166,250	1.9072 / 1.9072	60.000	0.41962
Max	47,542	249,930	2.9932 / 2.9933	60.000	0.99777

transformed. The simulation is then carried out for the Renishaw AM250 machine and *Ti-6Al-4V* material system deposited with 40 μm thickness. Other parameters are left as their default nominal values and no support structures are added (Ferguson et al., 2025).

In AM, the material is deposited layer by layer, and ideally, a high-fidelity simulation would model each layer individually. However, this can be computationally expensive, especially for builds with hundreds or thousands of layers. Layer lumping simplifies this process by combining multiple physical layers into a single *lumped* computational layer. In our calculations, NetFabb applies layer-lumping with a lumped layer thickness of 2.5 mm.

NetFabb re-meshes the geometry to contain axis-aligned hexahedral elements before simulating the build process. Then, NetFabb generates a thermal and a mechanical history for each part corresponding to lumped layer deposition steps during the build. We obtain the displacement, elastic strain, von Mises stress, and temperature fields evaluated at all nodal locations throughout the build. NetFabb also provides field values after the part has cooled down and detached from the build plate.

H.3 Benchmark task

To evaluate neural surrogate models, we target a field prediction task that is both central to our dataset and broadly relevant to the AM community: predicting residual vertical (Z) displacement. In LPBF, each layer is fused by a laser and followed by a recoater blade that spreads powder uniformly across the build area (Reijonen et al., 2024). Overhanging features may cause vertical displacements that interfere with the blade's path, potentially leading to collisions. Predicting the Z-displacement field can therefore help identify risk of blade collision failures. Rapid estimation of displacement prior to a build, thus, is highly desirable for design troubleshooting, as severe distortion can render a part unusable. Moreover, accurate prediction of nodal displacements can help anticipate build failures (Ferguson et al., 2025).

Since full-scale LPBF simulations are computationally expensive, taking minutes to hours, a fast surrogate model offers a valuable alternative for accelerating AM design. Accordingly, we train our models to predict the Z-displacement at every node at the final time step.

More formally, the input to a neural surrogate model is the volumetric axis-aligned hexahedral mesh describing the geometry. This includes the point-coordinates (array of size $N \times 3$ where N is the number of points) and, optionally, mesh connectivity information. The corresponding label is the Z displacement value at each point (array of size $N \times 1$).

While this dataset focuses on a steady-state prediction problem, future iterations of this benchmark could involve learning dynamic surrogate models that track the time-history of stress and deformation fields during the build process.

H.4 Data filtering

The dataset contains a wide-ranging set of shapes, making the dataset general enough to train a strong data-driven field prediction model. Out of ∼27,000 shapes, 19,732 were successfully simulated. We analyze the first 3,500 successful simulations and filter them according to several statistics to design a balanced training and test set. For example, we limit the learning problem to meshes with up to 50,000 points and up to 300,000 edges. This is done to reduce memory usage which becomes a bottleneck when training on small GPUs. We also filter meshes that have high aspect ratio elements as the FEM calculation could be unreliable on highly distorted geometries. The statistics for the filtered dataset are presented in Table 10 along with histograms in Figure 15. A gallery of successful simulations is presented in Figure 14.

H.5 Qualitative Results for LPBF Z-Displacement Prediction

In Figure 16, we present visualizations of ground-truth Z-displacement, predictions by FLARE, and the corresponding error. Across a set of representative geometries from the LPBF test set, FLARE produces displacement fields that are visually indistinguishable from the ground truth and capture both the global deformation patterns and localized high-gradient regions. Importantly, the error fields remain low-magnitude and spatially diffuse, with no systematic accumulation near edges or corners—indicating that the model does not rely on positional shortcuts or overfit to particular geometric artifacts. This suggests that FLARE successfully learns the dominant thermo-mechanical modes of deformation while only missing small higher-order variations that are difficult to resolve without substantially larger latent bottlenecks. These qualitative observations are consistent with the low average relative L^2 error and illustrate that FLARE maintains stable accuracy across a diverse distribution of part geometries.

H.6 Dataset split and benchmark protocol

Splits. From the full set of 19,732 successful LPBF simulations, we construct a filtered subset of cases with up to 50,000 points per geometry. We use 1,100 cases for training and 290 cases for testing, as summarized in Table 4.

Inputs and outputs. For each case, the benchmark input consists of the 3D mesh node coordinates $\in \mathbb{R}^{N\times 3}$. The benchmark target is the final-time-step vertical displacement field (Z-displacement) $\in \mathbb{R}^{N\times 1}$ at the same nodes. We focus on this scalar displacement because it is directly tied to recoater-blade interference risk in LPBF.

Simulation context. All parts are scaled to lie within a standardized build volume $[-30, 30] \times [-30, 30] \times [0, 60]$ mm and simulated in Autodesk NetFabb for the Renishaw AM250 machine and Ti-6Al-4V material system, using 40 μm layer thickness and layer lumping with a lumped layer height of 2.5 mm. These settings follow the procedure described in Appendix H.

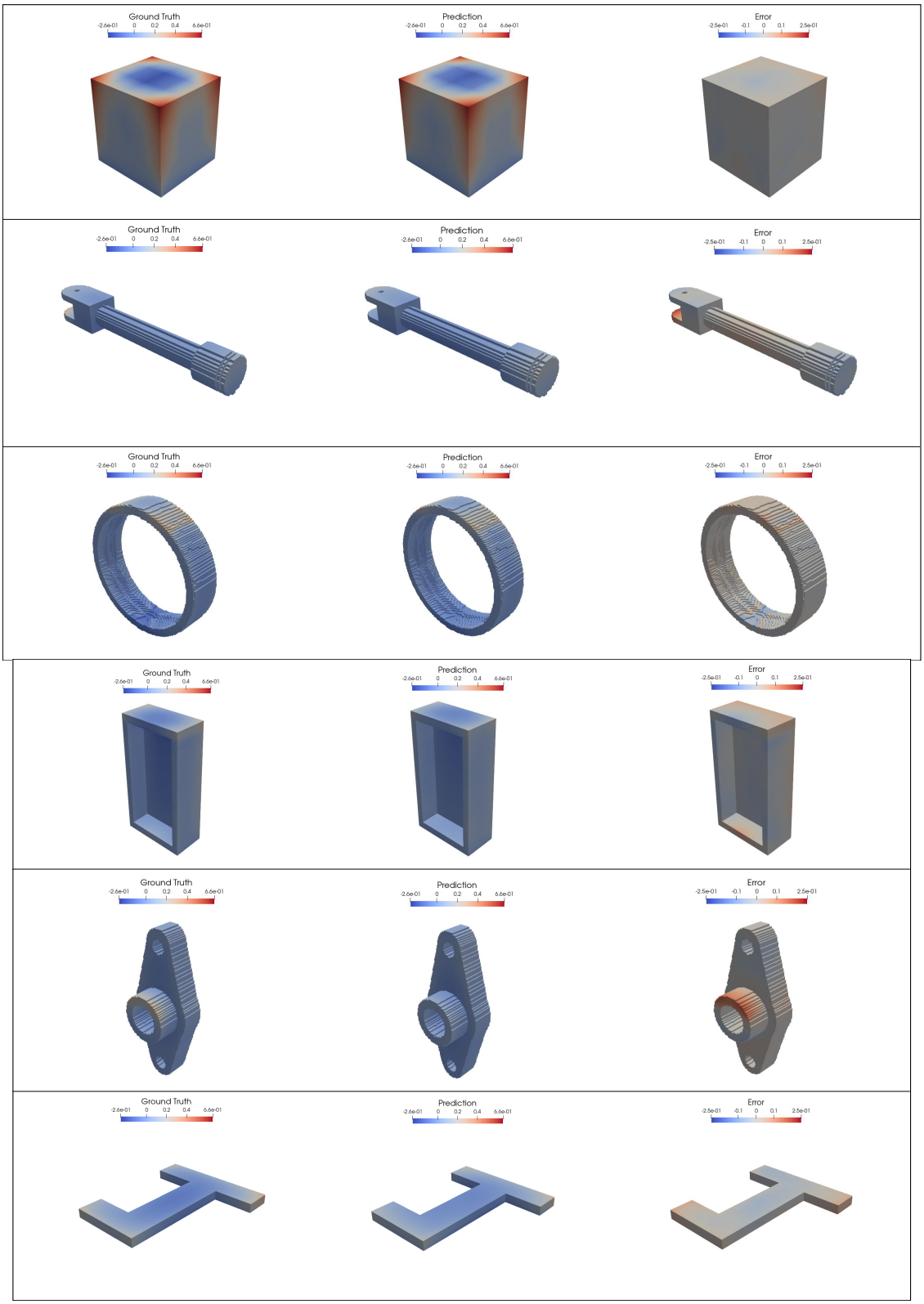


Figure 16: Qualitative results for FLARE on the LPBF Z-displacement prediction task. For each test geometry, we show the ground truth Z-displacement field, the model prediction, and the corresponding error (Ground Truth − Prediction).