LITE: LAYER-WISE INTERMEDIATE TOKEN-DROPPING FOR EFFICIENT STRUCTURED-GRID PDE EMULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Transformer-based neural surrogates for partial differential equations achieve strong predictive accuracy on structured grids, but their training cost grows rapidly with resolution, and in three dimensions it becomes the binding constraint on what can be trained at all. We introduce **LITE** (Layer-wise Intermediate Token-dropping), which makes sparse–dense token processing work in hierarchical PDE transformers. Rather than routing tokens around the entire network, LITE applies a drop–process–reconstruct cycle independently at each encoder and decoder stage while keeping the latent stage dense. Before each stage an ℓ_2-norm importance score retains one token per spatial block, concentrating computation on physically active regions. The stage's transformer blocks operate on that subset alone. A depthwise spread convolution then propagates the result to the dropped positions, and a learned linear fusion recombines it with the dense features carried past the stage untouched. Because full resolution is restored between stages, multi-scale feature flow and skip connections remain intact and no full-token fine-tuning phase is needed, so peak memory never returns to that of the dense model. In 2D, LITE halves wall-clock training time while reducing normalized RMSE by 23% on a 16-equation benchmark. Whole-network dropping at the same keep rate degrades sharply over a ten-step rollout, ending up worse than the dense model it aims to accelerate. In 3D at 256^3, dropping cuts per-sample latency and peak memory by 2.0–$2.8\times$ without increasing terminal rollout error, so a 167M-parameter model trains with lower latency and less memory than a 42M-parameter dense backbone. On full 384^3 volumes, three times the linear extent used in prior work on this data, the dense baselines exhaust memory, while LITE trains and improves rollout accuracy. At these resolutions token dropping is thus a precondition for training rather than an optimization.

1 INTRODUCTION

Neural surrogate models can approximate the solution operators of partial differential equations orders of magnitude faster than classical numerical solvers, opening the door to real-time design optimization, uncertainty quantification, and interactive simulation (Li et al., 2021; Lu et al., 2021; Kovachki et al., 2023). Transformers have become a common backbone for these models: shifted-window (Swin) attention (Liu et al., 2021) provides efficient local-global mixing on regular grids, axial attention (McCabe et al., 2024) factorizes computation along individual spatial axes, and hierarchical encoder-decoder topologies (Herde et al., 2024; Li et al., 2023b) further reduce cost while preserving multi-scale structure. Building on these ideas, the PDE-Transformer (Holzschuh et al., 2025) combines shifted-window attention with a hierarchical U-Net topology and per-channel token embedding to achieve state-of-the-art accuracy across 16 PDE families. Training cost, however, scales poorly. A single run on 256^2 grids exceeds twelve hours on eight GPUs, and moving to 512^2 nearly quadruples that cost. In three dimensions the constraint is sharper still, since memory rather than time determines which models can be trained at all.

In computer vision, aggressive token sparsification during training has been shown to reduce cost by almost an order of magnitude with little loss in quality. SPRINT (Park et al., 2026) does this for diffusion transformers by routing a randomly selected 25% of tokens through the bulk of the network

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

while preserving a dense copy for reconstruction afterward. The central question we address is: *How much improvement can be gained from token sparsification in the context of PDE solving?*

Applying methods like SPRINT directly to a modern PDE transformer architectures raises four difficulties. Whole-network routing drops tokens once and processes all layers at reduced count, conflicting with the encoder–latent–decoder structure and with skip connections that expect spatially complete feature maps. Random within-block selection ignores the spatial heterogeneity of physics fields, where shocks, vortices, and reaction fronts carry far more information than quiescent zones. Merging the stored dense copy with the processed tokens at each position independently cannot move information across space, so a dropped location is never updated by its retained neighbors, yet a PDE surrogate must be accurate everywhere. Finally, SPRINT closes the resulting train–inference gap with a full-token fine-tuning stage. That stage has the peak memory of dense training, so it cannot run where the dense model does not fit, which in 3D is exactly where sparsity is needed most.

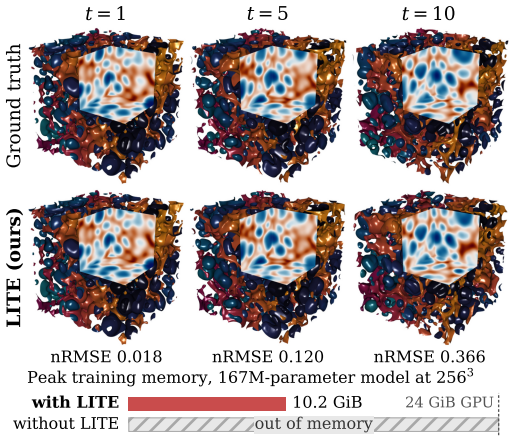

Figure 1: LITE rolling out a 3D Kuramoto–Sivashinsky trajectory at 256^3. Without LITE, training the same model exceeds a 24 GiB GPU.

We address these challenges by introducing **LITE** (Layer-wise Intermediate Token-dropping), a method that makes sparse–dense token fusion practical for hierarchical PDE transformers. Our contributions are as follows.

1. **Per-stage token dropping for hierarchical transformers.** We apply a drop–process–reconstruct cycle independently at each encoder and decoder stage, leaving the latent stage dense. Because every spatial position is restored between stages, the method interacts correctly with multi-scale downsampling and skip connections.
2. **Content-adaptive importance scoring.** The ℓ_2 norm of each token embedding serves as a zero-cost importance signal, concentrating computation on spatially active regions such as vortex cores, shock fronts, and reaction zones.
3. **Reconstruction with spread convolutions.** A depthwise convolution propagates information from retained positions to their masked neighbors, and a learned linear fusion combines the result with the stored dense features. This linear fusion is the default throughout, including all 3D experiments; an optional gated variant is described in Appendix D.1.
4. **Scalable sparse training without a full-token phase.** LITE is designed to not require any full-token processing. Training and inference use the same sparse configuration, so peak memory never returns to that of the dense model, and LITE trains where the dense backbone itself exhausts memory (Section 4.3; Appendix E).

We evaluate LITE on a multi-PDE benchmark covering 16 equation types at resolutions up to 512^2, on five standard single-PDE benchmarks, and in three dimensions on 14 PDE families at 256^3 (Kuramoto–Sivashinsky in Figure 1) and on full 384^3 volumes. On the multi-PDE task, LITE trains in half the wall-clock time of a dense PDE-Transformer while reducing nRMSE_1 from 0.043 to 0.033, and outperforms published baselines trained on four H100 GPUs. In 3D it reduces per-sample latency and peak memory by 2.0–$2.8\times$ without increasing terminal rollout error, and at 384^3 it trains in settings where the dense backbones exhaust memory. On the standard benchmarks it remains competitive in accuracy while training faster than MSPT (Curvo et al., 2026) and Transolver (Wu et al., 2024), and using less memory than dense backbones throughout (Appendix C).

2 Related Work

Neural PDE surrogates. Neural operators learn maps between function spaces and are the dominant paradigm for data-driven PDE solving. The Fourier Neural Operator (Li et al., 2021) parameterizes integral kernels in spectral space, and graph-based formulations (Li et al., 2020) extend the idea to irregular domains. Transolver (Wu et al., 2024) attains linear complexity by grouping mesh

points into learnable slices, its successor scales to million-point meshes (Luo et al., 2025), and MSPT (Curvo et al., 2026) combines local point attention within patches with global patch-level attention. Foundation models such as Poseidon (Herde et al., 2024) and MPP (McCabe et al., 2024) pretrain across diverse PDE families. These methods have improved accuracy, but their training costs remain substantial, and most target irregular meshes and do not exploit the regular structure of uniform grids.

Transformers for physics on structured grids. Shifted-window (Swin) attention (Liu et al., 2021) is the de facto building block for transformers on regular grids, giving linear complexity through local windowed self-attention with cross-window connectivity. Several PDE architectures adopt it or a related factorization: MPP (McCabe et al., 2024) attends along one spatial axis at a time; FactFormer (Li et al., 2023b) decomposes multi-dimensional attention into learnable one-dimensional projections; and scOT (Herde et al., 2024) uses a hierarchical Swin-style encoder–decoder with skip connections as the backbone of Poseidon. PDE-Transformer (Holzschuh et al., 2025) combines shifted-window attention with token down/upsampling in a U-Net topology and per-channel embedding for multi-physics coupling, and P3D (Holzschuh et al., 2026) extends that backbone to three dimensions with patch-based training. These works establish that structured-grid physics benefits from vision-transformer inductive biases, but none address training efficiency through token sparsification.

Efficient transformer training via token reduction. Token reduction during training is well explored in vision and generation. Conditional computation applied per layer is the closest line of work to ours: DynamicViT (Rao et al., 2021) prunes tokens progressively across layers with a learned importance predictor, EViT (Liang et al., 2022) keeps the top-scoring tokens at each layer by class-token attention and fuses the remainder, and Mixture-of-Depths (Raposo et al., 2024) routes a fixed top-k through each layer of a language model while the rest bypass it on the residual stream. All three act on flat, single-scale token sequences, select tokens individually rather than over spatial blocks, and have no downsampling hierarchy or skip connections to keep consistent. Masked autoencoders (He et al., 2022) randomly mask 75% of patches during pretraining, and diffusion transformers use related masking schemes (Gao et al., 2023; Zheng et al., 2024; Sehwag et al., 2025). SPRINT (Park et al., 2026) is the most aggressive: it partitions a flat DiT into a shallow encoder that processes all tokens, a deep middle that sees one randomly selected token per 2×2 patch, and a shallow decoder reached through concat-linear fusion.

Token reduction has seen limited use in PDE modeling. Masked autoencoders have been applied as a pretraining pretext for PDE surrogates (Zhou & Barati Farimani, 2024), but the masking is confined to pretraining and the fine-tuned model processes all tokens, so no inference-time saving is realized. AMR-Transformer (Xu et al., 2025) prunes cells through adaptive mesh refinement driven by Navier–Stokes criteria on the input field, but pruning occurs once at tokenization rather than repeatedly inside the network, and only in two dimensions. To the best of our knowledge, a generalizing and holistic approach for token dropping has not been proposed in the field. Additionally, our architecture departs from regular SPRINT in three structural ways: (i) per-stage rather than whole-network dropping, so that the hierarchical multi-scale structure and skip connections remain intact; (ii) content-adaptive importance scoring via the ℓ_2 norm of learned token embeddings rather than random selection; and (iii) a depthwise spread convolution that propagates evolved information from retained tokens to masked positions before fusion, rather than fusing the zero-padded sparse map directly. Relative to those per-layer methods, selection here is made over spatial blocks so the retained set remains a valid grid after downsampling, and scores tokens by embedding norm rather than class-token attention, which a field-regression model does not have.

3 METHOD

3.1 BACKGROUND: PDE-TRANSFORMER

Our approach starts from canonical, hierarchical transformer architectures for PDEs to predict PDE dynamics on regular grids (Holzschuh et al., 2025). Given an input field $\mathbf{u}_t \in \mathbb{R}^{C\times H\times W}$ at time t, it predicts the next state $\hat{\mathbf{u}}_{t+\Delta t}$. These architectures follow a U-Net topology with L encoder stages, a

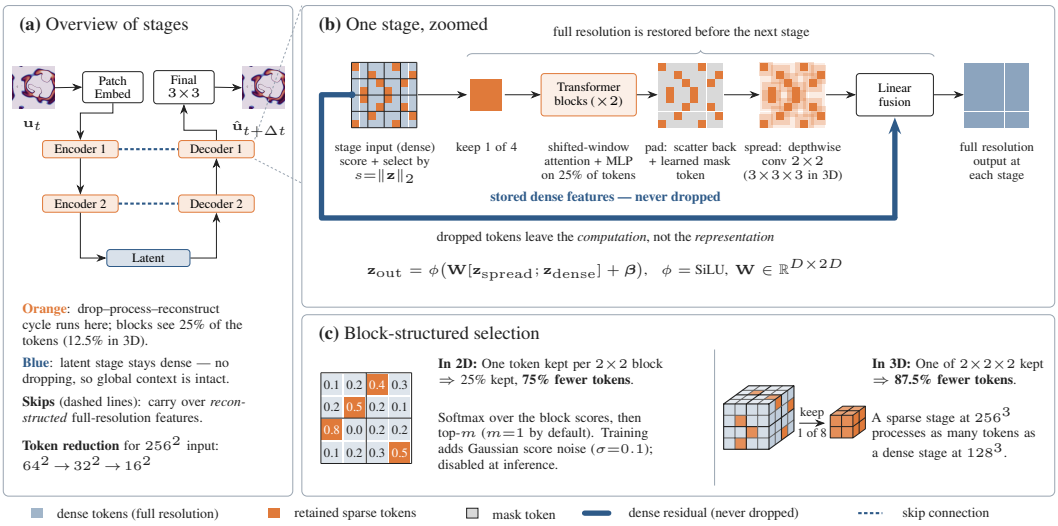

Figure 2: Overview of LITE. **(a)** The drop–process–reconstruct cycle runs at every encoder and decoder stage; the latent stage stays dense. **(b)** Dropped tokens leave the computation but not the representation. **(c)** The rule keeps one token in four in two dimensions and one in eight in three.

latent stage, and L decoder stages with skip connections:

$$\hat{\mathbf{u}}_{t+\Delta t} = \mathcal{D}\big(\mathcal{L}\big(\mathcal{E}(\mathbf{u}_t; \theta_E); \theta_L\big); \theta_D\big), \tag{1}$$

where $\mathcal{E}$, $\mathcal{L}$, $\mathcal{D}$ denote encoder, latent, and decoder respectively. Each stage consists of transformer blocks with shifted-window self-attention (Liu et al., 2021), adaptive layer normalization conditioned on a timestep embedding, and optional carrier tokens for global context. Between stages, spatial resolution halves (or doubles) via strided (transposed) convolutions while the channel dimension doubles (halves) up to a maximum $d_{\max}$. On a 256^2 grid with patch size 4 this gives 64^2 tokens at the first encoder stage, 32^2 at the second and 16^2 at the latent, with attention computed within 8×8 windows throughout. Training minimizes MSE against the ground-truth next state.

3.2 Layer-wise Intermediate Token-Dropping

The hierarchical design of PDE-Transformer already separates spatial scales: encoder stages capture fine-grained local features while deeper stages model coarse global dynamics. We amplify this separation by processing only a sparse token subset at each encoder and decoder stage, while maintaining a dense residual that preserves full-resolution information (Figure 2). The latent stage already operates at a heavily reduced spatial resolution (e.g., 16^2 tokens from an original 64^2), so further sparsification there would yield minimal computational savings while risking loss of the global context that skip connections rely on. Dropping is therefore confined to the *intermediate* stages of the hierarchy, namely the encoder and decoder stages surrounding the dense latent bottleneck, which is what the name LITE denotes. We apply this sparsification independently at every stage rather than routing around the entire network, so that full spatial information is restored between consecutive stages and the drop ratio can be tuned per stage.

Block-structured spatial token dropping. We partition the feature map $\mathbf{z} \in \mathbb{R}^{B\times D\times H'\times W'}$ at any stage into non-overlapping $k\times k$ spatial blocks (k=2 by default). Within each block, exactly one token is selected based on a content-adaptive importance score, yielding a downsampled representation $\mathbf{z}_{\text{sparse}} \in \mathbb{R}^{B\times D\times H'/k\times W'/k}$. This achieves 75% token reduction while preserving regular grid structure compatible with windowed attention. In three dimensions the same construction uses $k\times k\times k$ blocks, so retaining one token per block gives an 87.5% reduction. The importance score is the embedding ℓ_2 norm:

$$s_{b,i,j} = \|\mathbf{z}_{b,:,i,j}\|_2. \tag{2}$$

We then apply softmax on these scores and select top-m patches from the $k \times k$ block depending on the drop ratio we are using. During training, a random Gaussian noise on the scale of 0.1 is added as regularization, which is turned off during inference.

Per-stage application within the hierarchy. Token dropping is applied independently at each encoder and decoder stage of the hierarchy. Before each stage, tokens are scored, the sparse subset is selected, and the original dense features are stored. All transformer blocks within the stage then process only the retained tokens. After the final block, the reconstruction module (described below) restores the full spatial map. The latent stage processes all tokens densely to preserve the global context that higher-resolution encoder and decoder stages rely on through skip connections. This design has two advantages. First, tokens dropped within one stage are fully restored before the next stage processes them, so no spatial region is permanently starved of attention across the network depth. Second, because reconstruction occurs at every stage, skip connections between encoder and decoder carry fully reconstructed features, preserving gradient flow and spatial coherence throughout the hierarchy.

Reconstruction with spread convolutions. After the sparse path processes its reduced token set through attention and feed-forward layers, we reconstruct the full-resolution representation in three steps:

1. **Padding.** The sparse output is scattered back to the original spatial positions using stored indices; a learned mask token $\mathbf{m} \in \mathbb{R}^D$ fills the unoccupied positions.
2. **Spread convolution.** A depthwise convolution with weights initialized to a uniform box filter propagates information from retained positions to their masked neighbors:
$$\mathbf{z}_{\text{spread}} = \text{DWConv}(\mathbf{z}_{\text{padded}}). \tag{3}$$
The convolution is depthwise, performing spatial diffusion with no channel mixing. In 2D we use a 2×2 kernel. In 3D we use a $3 \times 3 \times 3$ kernel; centered on each voxel, it always covers the voxel's whole $2 \times 2 \times 2$ block, so every masked voxel receives that block's retained token.
3. **Linear fusion.** The spread features are concatenated channel-wise with the stage input $\mathbf{z}_{\text{dense}}$, which was retained at full resolution and left untouched by the sparse path, and passed through a single affine map and pointwise nonlinearity:
$$\mathbf{z}_{\text{out}} = \phi\big(\mathbf{W}[\mathbf{z}_{\text{spread}}; \mathbf{z}_{\text{dense}}] + \boldsymbol{\beta}\big), \qquad \mathbf{W} \in \mathbb{R}^{D \times 2D}, \tag{4}$$
where ϕ is SiLU and $[\cdot;\cdot]$ denotes channel-wise concatenation.

The mixing weights $\mathbf{W}$ are learned constants, so the same blend is applied at every position and for every input. The operator is not a gating mechanism. The dense branch $\mathbf{z}_{\text{dense}}$ carries full-resolution information past the stage. In this way tokens removed by selection are absent from the computation performed at that stage but remain present in the output. We use this linear fusion with $\mathbf{z}_{\text{dense}}$ by default, including in every 3D experiment. An optional input-dependent gated variant (Appendix D.1) improves on it in two dimensions and is used for the multi-PDE runs in Table 1 and the Plasticity benchmark, but gives no improvement in three dimensions.

End-to-end architecture. The complete forward pass proceeds as follows. The input field $\mathbf{u}_t$ is embedded into patch tokens. At each encoder and decoder stage, the drop–process–reconstruct cycle runs: tokens are scored, the top-scoring subset is retained, all transformer blocks within the stage operate on the reduced set, and the reconstruction module restores full resolution afterward. The latent stage operates on all tokens densely. Skip connections carry the reconstructed (full-resolution) features from encoder to decoder stages. The final layer maps to the output $\hat{\mathbf{u}}_{t+\Delta t}$. The core encoder–decoder architecture is unchanged. LITE is a modular wrapper around each stage, adding less than 2% additional parameters.

4 Experiments

4.1 Experimental Setup

Multi-PDE pretraining (APEBench). Following Holzschuh et al. (2025), we train on the 16 PDE families of APEBench (Koehler et al., 2024) at 256^2 and 512^2 with model size S ($\sim$33M parameters),

Table 1: Multi-PDE pretraining on APEBench (16 PDE types, model S). We reproduce PDE-Transformer S on our hardware for a direct wall-clock comparison. The SPRINT-style baseline applies whole-network token dropping (Park et al., 2026) to the same backbone at the same 25% keep rate, without the full-token fine-tuning stage that SPRINT uses to close the train–inference gap.

Model	Train Time	Params	nRMSE_1	nRMSE_{10}
Published baselines (Holzschuh et al., 2025) (4×H100)				
DiT-S	—	39.8M	0.066	0.78
scOT-S	—	39.8M	0.051	0.59
FactFormer	—	3.8M	0.069	0.65
UNet	—	35.7M	0.075	0.68
UDiT-S	—	58.9M	0.042	0.39
PDE-Transformer S	—	33.2M	0.044	0.36
Our results (8×L4 GPUs, 256^2)				
PDE-Transformer S (reproduced)	12.5 h	33.2M	0.0430	0.3597
SPRINT-style whole-network dropping	6.0 h	31.8M	0.0372	0.5333
LITE S (ours)	6.0 h	33.6M	**0.0331**	**0.3211**
Our results (8×L4 GPUs, 512^2)				
PDE-Transformer S (reproduced)	43.5 h	33.2M	0.0389	0.3296
LITE S (ours)	20.5 h	33.6M	**0.0289**	0.3383

for 100 epochs on 8×L4 GPUs; the families are listed in Appendix B.1 and the full configuration in Appendix B.2. We report normalized RMSE at one step (nRMSE_1) and over a ten-step autoregressive rollout (nRMSE_{10}).

Standard PDE benchmarks. We additionally evaluate on five single-PDE benchmarks from the Neural-Solver-Library (Wu et al., 2024) using a single NVIDIA L4 GPU (24 GB). That setup and its results, including wall-clock and memory comparisons against MSPT (Curvo et al., 2026) and Transolver (Wu et al., 2024), are reported in Appendix C.

Metrics. We report relative ℓ_2 error for standard benchmarks and nRMSE for the multi-PDE task, together with wall-clock training time, peak GPU memory, parameter count, and GFLOPs per sample (measured via PyTorch's built-in FlopCounterMode, or estimated from architecture specifications for published baselines). All reported results are single runs under one fixed seed, held constant across methods, resolutions, and evaluation. We therefore do not attach variance estimates to the tables and do not read small differences as decisive.

4.2 Multi-PDE Pretraining Results

Table 1 reports results on the multi-PDE benchmark. We include the architecture baselines of Holzschuh et al. (2025), which were trained on 4×H100 GPUs. LITE trains in half the wall-clock time of our reproduced dense baseline and is more accurate than all published baselines, including those trained on the faster hardware.

At 256^2, LITE achieves a 2.1× wall-clock speedup and 1.76× FLOPs reduction while reducing nRMSE_1 by 23%. The SPRINT-style comparison quantifies the cost of ignoring the hierarchy: at the same 25% keep rate on the same backbone, dropping tokens once for the whole network raises nRMSE_{10} to 0.5333, 66% worse than our per-stage design and 48% worse than the dense baseline it is meant to accelerate. To-

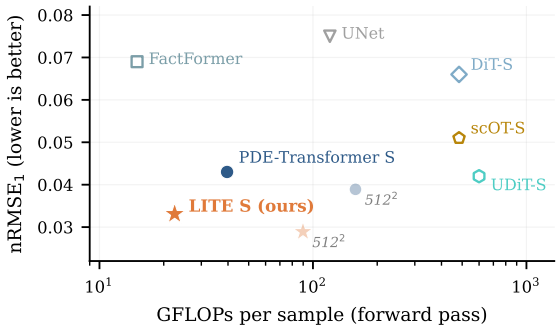

Figure 3: Efficiency–accuracy trade-off on the multi-PDE benchmark (256^2). Filled markers are our own measurements; open markers are published baselines.

Table 2: 3D results on 14 APEBench PDE families at 256^3 (100 epochs, nRMSE, raw weights, 15-step horizon). Latency is per-sample forward+backward at micro-batch 1 on one RTX 6000 Ada; memory is peak allocation. All models use the same training recipe. †The largest model uses a reduced learning rate of 1×10^{-4}, as the setting used for S and B diverged at this scale. ‡Not trained.

Model	Params	ms/sample	Peak Mem.	nRMSE (mean)	nRMSE$_{t=15}$
P3D S (Holzschuh et al., 2026)	11.2M	395	18.5 GiB	1.9919	8.9058
P3D B (Holzschuh et al., 2026)	46.1M	OOM	OOM	—	—
P3D-Patch S	11.1M	126	7.9 GiB	0.3659	0.7362
LITE-3D S (ours)	11.2M	**62**	**3.2** GiB	0.3910	0.7248
P3D-Patch B	42.3M	258	14.7 GiB	0.3692	0.7422
LITE-3D B (ours)	42.7M	106	5.3 GiB	0.3627	0.7077
P3D-Patch L‡	164.9M	569	28.9 GiB	—	—
LITE-3D L (ours)†	166.6M	239	10.2 GiB	**0.3517**	**0.6813**

ken dropping alone is therefore not sufficient; where in the hierarchy it is applied determines the accuracy. At 512^2 the speedup holds at $2.1\times$ and nRMSE$_1$ improves by 26%, while nRMSE$_{10}$ moves the other way; we do not claim an accuracy gain here, as a margin this size is not resolved by single-seed runs and our recipe was tuned at 256^2 and reused unchanged (Appendix B.2). The speedup exceeds the FLOPs reduction, which we attribute to reduced memory-bandwidth pressure on the L4s; Appendix B.3 shows this margin is hardware-dependent. Figure 3 shows the efficiency–accuracy trade-off.

4.3 Extension to 3D

We next apply the same stage-wise token dropping to P3D (Holzschuh et al., 2026), a 3D architecture pairing a convolutional encoder/decoder with a windowed-attention backbone. We compare three backbones: *P3D S*, the published form; *P3D-Patch*, which replaces its convolutional encoder/decoder with patch embedding to give a pure-transformer counterpart for which the effect of dropping can be isolated; and *LITE-3D*, which adds the per-stage cycle to that backbone, keeping one token per $2\times2\times2$ block (87.5% reduction) with linear fusion throughout. All models train with the same recipe on 14 APEBench families at 256^3 for 100 epochs and are evaluated with raw (non-EMA) weights over a 15-step horizon; we train directly on 256^3 crops, twice the largest (128^3) reported for this benchmark by Holzschuh et al. (2026). Architectures, training setup, and the latency and memory measurement protocol are in Appendix A.

Table 2 and Figure 4 show the same pattern as in 2D. At matched model size, dropping reduces per-sample latency by $2.0\times$ and peak memory by $2.5\times$ at S, and by $2.4\times$ and $2.8\times$ at B. Accuracy is largely unchanged: at S the horizon-averaged nRMSE rises by 6.9% while the error at the final rollout step $t{=}15$ falls by 1.6%, and at B dropping is better on both metrics, by 1.8% on the horizon average and 4.6% at the final step. Terminal error is lower with dropping in both cases, the regime that matters for long rollouts.

P3D B exhausts memory at 256^3. We profile but do not train P3D-Patch L, since P3D-Patch B already has higher error than P3D-Patch S. LITE-3D L (166.6M) is marginally *larger* than P3D-Patch L (164.9M), yet is $2.4\times$ faster (239 vs. 569 ms) and needs $2.8\times$ less peak memory (10.2 vs. 28.9 GiB), so it fits even on 24 GiB GPUs such as the RTX 4090 or L4, which P3D-Patch L exceeds (Figure 5). The saved compute is what makes the larger scales reachable. LITE-3D L has $3.9\times$ the parameters of P3D-Patch B, yet trains with lower latency (239 vs. 258 ms) and lower peak memory (10.2 vs. 14.7 GiB), and attains the best accuracy of any configuration we trained. The same holds for the smaller models: LITE-3D B has $3.8\times$ the parameters of P3D-Patch S while being faster (106 vs. 126 ms), roughly $1.5\times$ lighter, and is more accurate. Relative to P3D S, LITE-3D L is $1.7\times$ faster and uses $1.8\times$ less memory with $14.9\times$ the parameters. At a fixed budget, token dropping therefore admits a substantially larger 3D model, which in turn yields gains in accuracy.

Scaling to 384^3. The experiments above train on crops, which bounds the spatial context each model sees. We therefore test whether stage-wise dropping permits training on full volumes, using the

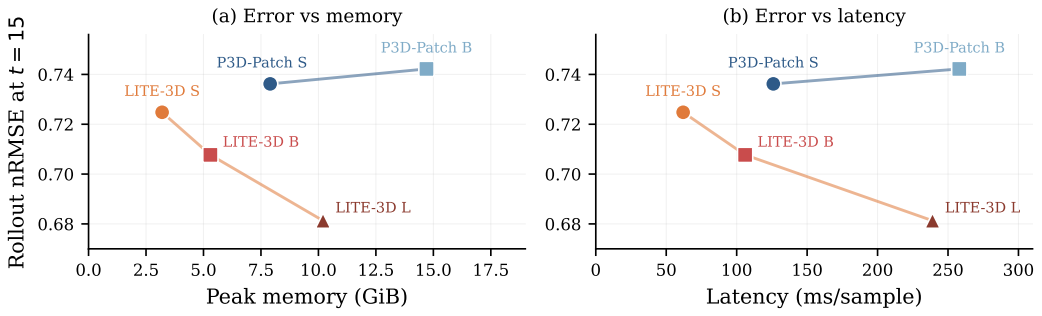

Figure 4: Rollout error against cost at 256^3 (values from Table 2). Every LITE-3D configuration is Pareto-optimal on both axes and both P3D-Patch configurations are dominated: LITE-3D S alone reaches lower terminal error than P3D-Patch B on $4.6\times$ less memory. The families scale in opposite directions: P3D-Patch S $\rightarrow$ B costs $1.9\times$ the memory and $2.1\times$ the latency yet *raises* terminal error, whereas LITE-3D improves monotonically with size. P3D S is off-scale (terminal error 8.91).

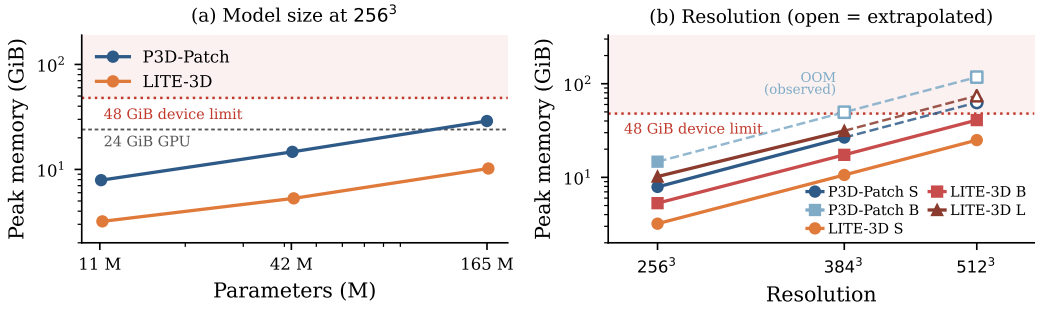

Figure 5: Peak memory of the 3D backbones, with the 48 GiB capacity of the profiling device shaded. **(a)** At 256^3, LITE-3D L holds $3.9\times$ the parameters of P3D-Patch B in 31% less memory and needs $2.8\times$ less than P3D-Patch L. **(b)** Memory tracks grid volume, so filled markers are measured and open ones extrapolated (see Appendix A). At 512^3, LITE-3D S and B still fit (25.0 and 40.9 GiB).

Korteweg–de Vries (KdV) dataset of the 3D APEBench suite, which is released at a native periodic resolution of 384^3 with three fields. KdV is dispersive and non-dissipative, developing sharp solitons, so accuracy depends on resolving fine structure rather than diffusive smoothing. We train on the full 384^3 domain, whereas Holzschuh et al. (2026) had to resort to training on smaller regions of at most 128^3. This means we are able to process $27\times$ higher voxel counts with our method. To the best of our knowledge, no prior neural PDE surrogate has been trained on full, dense volumes at this resolution. Training and evaluation follow Appendix A.

At this resolution dropping is what makes training possible, not what makes it faster: neither P3D S nor P3D-Patch B runs on full 384^3 volumes at all. Among the configurations that do run, dropping again costs no accuracy. At matched model size LITE-3D S is $2.3\times$ faster than the P3D-Patch backbone and more accurate on both metrics (-7.7% horizon-averaged nRMSE, -10.6% terminal error), and LITE-3D B carries $3.8\times$ the parameters of P3D-Patch S while still training faster per sample (421 vs. 454 ms) and improving those metrics by 14.3% and 19.0%. The margin is larger than in the crop-based setting, which is consistent with the proposed mechanism: longer token sequences leave more redundancy for the importance score to exploit.

5 Discussion

Why dropping helps more at scale. The cleanest test of what dropping does to accuracy is in 3D, where P3D-Patch and LITE-3D differ only in the per-stage cycle. There, dropping lowers terminal rollout error in every matched pair, and the margin widens with size: by 1.6% at S and 4.6% at B on

Table 3: KdV at full 384^3 resolution (100 epochs, same recipe, nRMSE, raw weights, 29-step autoregressive rollout). Latency and peak memory are as in Table 2, with four-GPU training.

Model	Params	ms/sample	Peak Mem.	nRMSE (mean)	nRMSE$_{t=29}$
P3D S (Holzschuh et al., 2026)	11.2M	OOM	OOM	—	—
P3D-Patch B	42.3M	OOM	OOM	—	—
P3D-Patch S	11.1M	454	26.6 GiB	0.2995	0.5396
LITE-3D S (ours)	11.2M	**196**	**10.6** GiB	0.2764	0.4823
LITE-3D B (ours)	42.7M	421	17.4 GiB	**0.2568**	**0.4369**

256^3 crops, and by 10.6% at S on full 384^3 volumes (Tables 2 and 3). The horizon average agrees except at S on crops, where it rises by 6.9%. In every pair the advantage is larger at the final step than over the horizon, suggesting that dropping mainly slows the accumulation of error over the rollout. This ordering is consistent with the mechanism, as a larger volume holds more tokens that the score can discard without losing structure, although the step to 384^3 also changes the dataset and the rollout horizon. In 2D the grids are far smaller, a 256^2 field holding $256\times$ fewer points than a 256^3 crop, so less is redundant and what matters is how tokens are dropped. LITE improves on the dense baseline there, lowering nRMSE$_{10}$ at 256^2 from 0.3597 to 0.3211, whereas dropping once for the whole network, on the same backbone at the same keep rate, raises it to 0.5333 (Table 1).

Why not apply SPRINT directly? SPRINT targets flat DiT architectures in which all blocks are identical, and it spends its sparse budget at random rather than on informative regions. It also commits to a single subset: tokens are dropped once and the same retained set passes through every sparse block. LITE instead restores the full map after each stage and scores it again, so each stage can process a different set of tokens, while the skip connections and the dense latent stage always see every position.

Why having no full-token phase matters. Sparse-training methods that recover accuracy with a full-token phase, such as SPRINT's fine-tuning stage or the dense fine-tuning of MAE-PDE (Zhou & Barati Farimani, 2024), face a dilemma. Keeping the full-token phase returns training to dense peak memory, while omitting it costs accuracy. This leaves the SPRINT-style baseline above worse than the dense model it is meant to accelerate. LITE avoids this dilemma because it works without a full-token phase. Its encoder and decoder blocks only ever process the retained tokens, and returning to full resolution costs a depthwise convolution and a linear fusion, so peak memory stays 2.5–$2.8\times$ below that of the matched dense backbone. In 3D this determines what can be trained. At 256^3, LITE-3D L peaks at 10.2 GiB and P3D-Patch L at 28.9 GiB, more than a 24 GiB GPU holds. At 384^3, LITE-3D B peaks at 17.4 GiB while P3D-Patch B exhausts a 48 GiB card (Tables 2 and 3).

Limitations. LITE is designed for regular grids and cannot be applied directly to irregular meshes, and it assumes the spatial dimensions are divisible by the block size at every scale of the hierarchy. Retaining m of the k^d tokens per block quantizes the keep rate to multiples of $1/k^d$. With $k{=}2$ and our default $m{=}1$ this is a 75% reduction in two dimensions and 87.5% in three, and intermediate rates require non-cubic blocks. On spatially compact benchmarks the wall-clock saving can vanish, and LITE is sometimes slower than the dense backbone, though the memory reduction persists (Appendix C).

6 Conclusion

We have presented LITE, a per-stage token dropping strategy for PDE transformers on structured grids. In two dimensions it halves training time and memory while matching or exceeding dense-baseline accuracy across 16 PDE families. In three dimensions this turns into a substantial advantage: at 256^3 a $3.9\times$ larger model trains more cheaply than the dense backbone, and on full 384^3 volumes the dense backbones do not fit at all. To conclude, by enabling fundamental scaling benefits, token dropping provides an important building block for scientific AI methods and future real-world applications.

ETHICS STATEMENT

This work develops efficiency techniques for training neural surrogates of partial differential equations. Reducing the cost of training these models lowers the hardware barrier for groups with limited GPU budgets, and the memory reduction makes higher-resolution training feasible on a single workstation-class device; we see no application-specific risks beyond those already associated with neural surrogate models. It involves no human subjects, no personally identifiable information, and no crowdsourced data collection. All datasets used (the APEBench suite (Koehler et al., 2024) and the benchmarks distributed with the Neural-Solver-Library (Wu et al., 2024)) are publicly available, synthetically generated from numerical solvers, and used in accordance with their licenses; the original authors are credited throughout. The baselines we compare against are run from their official implementations with published hyperparameters. We see no foreseeable harmful applications specific to this work beyond those already inherent in scientific computing and simulation, and we declare no conflicts of interest or sponsorship that could bias the reported results. By reducing the compute and memory required to train PDE transformers by roughly a factor of two, the method lowers the hardware barrier for researchers without access to large GPU clusters, as discussed in Section 5.

REPRODUCIBILITY STATEMENT

We have made the following efforts to ensure that our results can be reproduced. The method is specified in full in Section 3, with the exact token-scoring rule (Eq. 2) and the reconstruction operator (Eq. 4) given in closed form, and the parameter shapes and initializations of every added module listed in Appendix D.1. Datasets, splits, resolutions, and preprocessing are described in Section 4 and Appendices A.2, B.1 and C.1; all datasets are publicly available. Complete per-benchmark hyperparameter configurations are tabulated in Appendix C.2, and the multi-PDE training recipe (learning rate, schedule, batch size, gradient accumulation, block variants) is given in Appendix B.2. The ablation protocol, including the exact PDE subset and epoch budget, is detailed in Appendix D.2, and the 3D setup in Appendix A. Hardware, wall-clock training times, and peak memory for every run are reported in Section 4, Figure 8, and Appendix B.3, with FLOP measurement methodology described in Appendix C. Every experiment in this paper is a single run under one fixed random seed, held constant across methods, resolutions, and evaluation, so reported differences reflect that seed rather than an average over repetitions; the seed is set in the released configuration files. Source code for LITE, together with the configuration files needed to reproduce every table and figure, is provided as anonymous supplementary material and will be released publicly upon acceptance.

AI USE STATEMENT

In this work, we used generative AI tools in four ways. First, we used them to aid and polish the writing of the manuscript, including grammar, phrasing, and LaTeX formatting. Second, we used them for retrieval and discovery, i.e. to help locate and surface related work, whose relevance and claims we then verified by reading the original papers ourselves. Third, we used them to assist with the code that generates and formats the figures and tables reported in the paper. Fourth, we used them to prepare the supplementary code release: cleaning up our code base, writing configuration files, launch scripts and documentation, and anonymizing it. We did not use generative AI tools to derive the method, to run experiments, or to produce, alter, or interpret any experimental results: all numbers reported in this paper come from our own training and profiling runs.

We have reviewed all AI-assisted work. AI-polished text was checked by the authors for factual accuracy and for unintentional overlap with existing literature, all AI-suggested references were verified against the original sources before citation, and all AI-assisted plotting code was inspected and its outputs cross-checked against the underlying logged results. The cleaned release code was checked against our original code: it reproduces a reported KdV rollout evaluation exactly and matches the reported compute measurements. We take responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Shuhao Cao. Choose a transformer: Fourier or Galerkin. *Advances in Neural Information Processing Systems*, 34, 2021.

Pedro M. P. Curvo, Jan-Willem van de Meent, and Maksim Zhdanov. MSPT: Efficient large-scale physical modeling via parallelized multi-scale attention. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Shanghua Gao, Pan Zhou, Ming-Ming Cheng, and Shuicheng Yan. Masked diffusion transformer is a strong image synthesizer. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Maximilian Herde, Bogdan Raonić, Tobias Rohner, Roger Käppeli, Roberto Molinaro, Emmanuel de Bézenac, and Siddhartha Mishra. Poseidon: Efficient foundation models for PDEs. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Benjamin Holzschuh, Qiang Liu, Georg Kohl, and Nils Thuerey. PDE-Transformer: Efficient and versatile transformers for physics simulations. In *Forty-second International Conference on Machine Learning (ICML)*, 2025.

Benjamin Holzschuh, Georg Kohl, Florian Redinger, and Nils Thuerey. P3D: Highly scalable 3D neural surrogates for physics simulations with global context. In *International Conference on Learning Representations (ICLR)*, 2026.

Felix Koehler, Simon Niedermayr, Rüdiger Westermann, and Nils Thuerey. APEBench: A benchmark for autoregressive neural emulators of PDEs. In *Advances in Neural Information Processing Systems*, volume 37, 2024. Datasets and Benchmarks Track.

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.

Zijie Li, Kazem Meidani, and Amir Barati Farimani. Transformer for partial differential equations' operator learning. *Transactions on Machine Learning Research (TMLR)*, 2023a.

Zijie Li, Dule Shu, and Amir Barati Farimani. Scalable transformer for PDE surrogate modeling. In *Advances in Neural Information Processing Systems*, volume 36, 2023b.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations (ICLR)*, 2021.

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, 2023c.

Youwei Liang, Chongjian Ge, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. Not all patches are what you need: Expediting vision transformers via token reorganizations. In *International Conference on Learning Representations (ICLR)*, 2022.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

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 (ICML)*, 2025.

Michael McCabe, Bruno Régaldo-Saint Blancard, Liam Holden Parker, Ruben Ohana, Miles Cranmer, Alberto Bietti, Michael Eickenberg, Siavash Golkar, Géraud Krawezik, François Lanusse, Mariel Pettee, Tiberiu Tesileanu, Kyunghyun Cho, and Shirley Ho. Multiple physics pretraining for spatiotemporal surrogate models. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Dogyun Park, Moayed Haji-Ali, Yanyu Li, Willi Menapace, Sergey Tulyakov, Hyunwoo J. Kim, Aliaksandr Siarohin, and Anil Kag. SPRINT: Sparse-dense residual fusion for efficient diffusion transformers. In *International Conference on Learning Representations (ICLR)*, 2026.

Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. DynamicViT: Efficient vision transformers with dynamic token sparsification. In *Advances in Neural Information Processing Systems*, 2021.

David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-Depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint arXiv:2404.02258*, 2024.

Vikash Sehwag, Xianghao Kong, Jingtao Li, Michael Spranger, and Lingjuan Lyu. Stretching each dollar: Diffusion training from scratch on a micro-budget. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Haixu Wu, Tengge Hu, Huakun Luo, Jianmin Wang, and Mingsheng Long. Solving high-dimensional PDEs with latent spectral models. In *International Conference on Machine Learning (ICML)*, 2023.

Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. Transolver: A fast transformer solver for PDEs on general geometries. In *Forty-first International Conference on Machine Learning (ICML)*, 2024.

Zipeng Xiao, Zhongkai Hao, Bokai Lin, Zhijie Deng, and Hang Su. Improved operator learning by orthogonal attention. In *International Conference on Machine Learning (ICML)*, 2024.

Zeyi Xu, Jinfan Liu, Kuangxu Chen, Ye Chen, Zhangli Hu, and Bingbing Ni. AMR-Transformer: Enabling efficient long-range interaction for complex neural fluid simulation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Hongkai Zheng, Weili Nie, Arash Vahdat, and Anima Anandkumar. Fast training of diffusion models with masked transformers. *Transactions on Machine Learning Research (TMLR)*, 2024.

Anthony Zhou and Amir Barati Farimani. Masked autoencoders are PDE learners. *Transactions on Machine Learning Research*, 2024.

APPENDIX

TABLE OF CONTENTS

A LITE-3D: Architecture, Data, and Training **13**
A.1 Architectures . . . 13
A.2 Datasets . . . 13
A.3 Training Setup . . . 14
A.4 Compute Measurement Protocol . . . 14
A.5 Korteweg–de Vries at Full 384^3 Resolution . . . 15

B 2D Multi-PDE Benchmark **16**
B.1 Dataset . . . 16
B.2 Training Recipe . . . 16
B.3 Resolution Scaling . . . 16

C Standard 2D Benchmarks **17**
C.1 Datasets . . . 18
C.2 Hyperparameters . . . 18
C.3 Training Efficiency . . . 18
C.4 Per-Sample FLOPs . . . 19
C.5 Accuracy . . . 19
C.6 MSPT on Darcy Flow . . . 19

D LITE Components and Ablations **20**
D.1 Components . . . 20
D.2 Mechanism Ablation . . . 21
D.3 Selection Rule: ℓ_2 Scoring against Random . . . 21

E Why Fine-Tuning-Based Sparse Training Does Not Scale **22**

F Qualitative Results **24**
F.1 3D Rollouts . . . 24
F.2 2D Predictions . . . 27
F.3 2D Rollout Stability . . . 27

A LITE-3D: ARCHITECTURE, DATA, AND TRAINING

This appendix details the 3D experiments of Section 4.3. Appendices B and C detail the 2D experiments, Appendices D and E the components, ablations and analysis shared by both settings, and Appendix F the qualitative results.

A.1 ARCHITECTURES

The 3D experiments build on P3D (Holzschuh et al., 2026), a hierarchical 3D transformer whose published form uses a convolutional encoder/decoder: two stride-2 3D convolutions reduce each spatial axis by $4\times$ before the transformer backbone, so a 256^3 input yields 64^3 tokens. Our *P3D-Patch* baseline uses only a single linear conv-layer for embedding, giving a pure, patch-based transformer backbone to isolate the effect of token dropping. *LITE-3D* applies the same per-stage drop–process–reconstruct cycle described in Section 3.2 to that backbone, with the latent stage left dense. The S, B and L variants scale the width, and at L the number of attention heads, at a fixed depth (Table 4), giving roughly 11M, 42M and 167M parameters. In each of the four encoder and decoder stages, LITE-3D keeps one token per $2\times2\times2$ block, selected by the ℓ_2 score of Eq. 2, so the stage's transformer blocks process one token in eight; the full grid is then restored with the mask token, a $3\times3\times3$ depthwise spread convolution and linear fusion (Eq. 4).

A.2 DATASETS

The 3D experiments use 14 PDE families of the 3D APEBench suite (Koehler et al., 2024), listed in Table 5: six families at 384^3 whose trajectories vary the PDE coefficients and the initial condition,

Table 4: 3D model configurations. Every model has five stages (two encoder stages, a dense latent stage and two decoder stages) of two transformer blocks with MLP ratio 4 and 4^3 attention windows without shifting, and resamples by $2\times$ per axis between stages. At 256^3 the first stage holds 64^3 tokens and the latent stage 16^3.

	S	**B**	**L**
Stage widths, encoder to latent	64, 128, 256	128, 256, 512	256, 512, 1024
Attention heads per stage	4	4	8
Parameters, P3D	11.2M	46.1M	—
Parameters, P3D-Patch	11.1M	42.3M	164.9M
Parameters, LITE-3D	11.2M	42.7M	166.6M

and eight Gray–Scott regimes at 320^3 that differ in their feed and kill rates. All grids are periodic, and every training trajectory has 30 frames, giving 29 input–target pairs. Families with fewer than three fields are zero-padded to three input channels, and training draws random 256^3 crops. The Korteweg–de Vries family is also the full-resolution benchmark of Appendix A.5.

Table 5: The 14 PDE families of the 3D experiments. For the first six, the coefficients are drawn per trajectory from the listed ranges and the initial condition from truncated Fourier series, Gaussian random fields or diffused noise. The Gray–Scott regimes fix the feed and kill rates (F, k) and vary Gaussian-blob initial conditions.

Family	**Fields**	**Grid**	**Coefficients**	**Train / held-out**
Burgers	3	384^3	viscosity $[10^{-3}, 5\times10^{-3}]$	50 / 10
Fisher–KPP	1	384^3	diffusivity $[10^{-4}, 0.02]$, reactivity $[5, 15]$	50 / 10
Hyper-diffusion	1	384^3	hyper-diffusivity $[5\times10^{-5}, 5\times10^{-4}]$	50 / 10
Korteweg–de Vries	3	384^3	domain extent $[30, 120]$, viscosity $[0.1, 0.25]$	50 / 10
Kuramoto–Sivashinsky	1	384^3	domain extent $[10, 130]$	60 / 5
Swift–Hohenberg	1	384^3	reactivity $[0.4, 1]$, critical number $[0.8, 1.2]$	50 / 10
Gray–Scott α	2	320^3	$(F, k) = (0.008, 0.046)$	10 / 3
Gray–Scott β	2	320^3	$(F, k) = (0.020, 0.046)$	10 / 3
Gray–Scott γ	2	320^3	$(F, k) = (0.024, 0.056)$	10 / 3
Gray–Scott δ	2	320^3	$(F, k) = (0.028, 0.056)$	8 / 2
Gray–Scott ϵ	2	320^3	$(F, k) = (0.020, 0.056)$	10 / 5
Gray–Scott θ	2	320^3	$(F, k) = (0.040, 0.060)$	8 / 2
Gray–Scott ι	2	320^3	$(F, k) = (0.050, 0.0605)$	8 / 2
Gray–Scott κ	2	320^3	$(F, k) = (0.052, 0.063)$	8 / 2

A.3 Training Setup

All models are trained on 14 PDE families from the 3D APEBench suite at 256^3 resolution for 100 epochs under a cosine learning-rate schedule with bf16 mixed precision, on four NVIDIA RTX 6000 Ada GPUs (48 GiB). P3D, P3D-Patch and LITE-3D share this recipe in full: a base learning rate of 2×10^{-4} at S and B, the same effective batch size, and identical transformer blocks. The models therefore differ only in the architectural changes described above. For reference, Holzschuh et al. (2026) train for 1000 epochs on four H100 GPUs on the corresponding benchmark, with crops of at most 128^3; our budget is therefore an order of magnitude shorter at twice the spatial resolution, which should be kept in mind when reading absolute error values in Table 2. Evaluation uses raw (non-EMA) weights and reports normalized RMSE over a 15-step autoregressive rollout, both averaged over the full horizon (t_1–t_{15}) and at the final step. The L variant is trained with a reduced learning rate of 1×10^{-4}; at the learning rate used for the S and B variants, training at this scale diverged. Table 6 lists the full configuration.

A.4 Compute Measurement Protocol

We report per-sample forward+backward latency, measured in isolation from the input pipeline so that the reported cost reflects model computation alone. For each variant we instantiate the model

Table 6: 3D training configuration, shared by P3D, P3D-Patch and LITE-3D.

Objective	single-step prediction ($t \to t{+}1$), MSE loss
Optimizer	AdamW, no weight decay
Learning rate	$2{\times}10^{-4}$ (S, B); $1{\times}10^{-4}$ (LITE-3D L)
Schedule	cosine annealing per step to $2{\times}10^{-6}$, no warm-up
Epochs	100
Effective batch	16, data-parallel over four GPUs
Gradient clipping	global norm 1.0
Precision	bf16 mixed, `torch.compile`
Training crops	random 256^3 (14 families); full 384^3 domain (KdV)
Evaluated weights	raw; an EMA (decay 0.999) is tracked but not used

from the same configuration used for training, place it on a single NVIDIA RTX 6000 Ada (48 GiB; driver 580.173.02, PyTorch 2.10.0, CUDA 12.8) in training mode, and pre-allocate a synthetic input of shape $1{\times}3{\times}256^3$ in bfloat16 directly on the device. Because the input is allocated once outside the measurement loop, the timed region contains no data loading, decoding, augmentation, or host-to-device transfer. We then time, using CUDA events with an explicit device synchronization before reading elapsed time, a region comprising the forward pass under bfloat16 autocast, a scalar loss, gradient zeroing, and the full backward pass; the optimizer step, distributed gradient all-reduce, and exponential-moving-average update are excluded, so the measurement isolates the cost attributable to the architecture rather than to the optimization or communication schedule. We report the minimum over ten iterations, which discards first-iteration autotuning and allocator growth and is robust to clock jitter, and we additionally report peak allocated memory, which covers parameters, gradients, and activations but excludes optimizer state. We use a micro-batch of one so that latency is directly comparable across model scales; consequently these numbers characterize per-sample latency at low occupancy rather than saturated throughput. Measurements are taken in eager mode, whereas our training runs use `torch.compile`; absolute latencies are therefore upper bounds on the compiled cost, and we use them for relative comparison between architectures rather than as absolute throughput figures.

Memory extrapolation. Peak memory tracks grid volume, so the open markers in Figure 5(b) scale the peak at the largest measured resolution r_{meas} by $(r/r_{\text{meas}})^3$. Checked against measured points, the law extrapolated from 256^3 matches the 384^3 measurements to within 2.8% at S and B and 9.6% at L, and extrapolated from 384^3 it predicts the measured 512^3 peaks of LITE-3D S and B (25.0 and 40.9 GiB) to within 0.9%. At 384^3 it places P3D-Patch B at 49.6 GiB and P3D S at 62.4 GiB, above the 48 GiB capacity of the card, and both were seen to fail there (Table 3).

A.5 Korteweg–de Vries at Full 384^3 Resolution

KdV benchmark. The full-resolution experiment of Table 3 uses the three-dimensional Korteweg–de Vries equation from APEBench, simulated on a periodic 384^3 grid with three fields. The dataset comprises 60 independent trajectories, split into 50 for training and 10 held out for evaluation; no trajectory is shared across the split. Each model is trained for 100 epochs on single-step prediction, mapping the state at time t to the state at $t{+}1$ under an L_2 objective, using AdamW with a base learning rate of $2{\times}10^{-4}$ annealed by a cosine schedule to $\eta_{\min} = 2{\times}10^{-6}$, gradient-norm clipping at 1.0, and bfloat16 mixed precision. Training is distributed over four GPUs with an effective batch of 16, and crops are taken at the full 384^3 domain size so that training and evaluation resolutions match. The random seed is the single fixed value used throughout the paper (Section 4). We evaluate the most recent of the three checkpoints with the lowest validation loss, which for every model is from one of the last two epochs. Although an exponential moving average of the weights is maintained during training, we report raw (non-EMA) weights throughout, as the EMA weights produced similar results.

KdV rollout evaluation. We assess long-horizon stability by autoregressive rollout: starting from the initial state of each held-out trajectory, the model is applied recursively to its own output for 29 successive steps, with no teacher forcing and no access to ground truth after the first frame. At every step we compute a normalized root-mean-square error against the reference trajectory, defined per

sample as $\sqrt{\langle(\hat{u}-u)^2\rangle/\langle u^2\rangle}$ for prediction $\hat{u}$ and reference u, where $\langle\cdot\rangle$ averages over all three fields and all grid points; we then average over the held-out trajectories. This yields an error-versus-horizon curve, from which we report the mean over all 29 steps as a summary of overall rollout quality and the error at the final step as a measure of terminal stability. Because the sparse token-selection mechanism is stochastic during training, each model is evaluated with the same selection rule it was trained with; evaluating under a mismatched rule degraded accuracy by 17–38% in preliminary experiments.

B 2D Multi-PDE Benchmark

B.1 Dataset

The multi-PDE benchmark uses the APEBench dataset (Koehler et al., 2024) following the setup of Holzschuh et al. (2025). It comprises 16 PDE families spanning advection, diffusion, Fisher–KPP, Gray–Scott (α, β, γ variants), Kolmogorov flow, Burgers, KdV, Kuramoto–Sivashinsky, Swift–Hohenberg, and decaying turbulence. Each PDE is simulated on periodic 256^2 or 512^2 grids with 10 time steps per trajectory. The training set contains 128 trajectories per PDE type (2048 total), and the test set contains 32 per type (512 total).

B.2 Training Recipe

LITE is trained on the multi-PDE benchmark with the following setup:

- **Learning rate.** A base learning rate of 5.5×10^{-4}.
- **Schedule.** A cosine learning rate schedule with linear warmup.
- **Batching.** A per-GPU batch size of 16 with gradient accumulation 2, giving an effective batch size of 32.
- **Blocks and activations.** Gated MLP blocks, and GELU activation on the output and upsample layers.

These choices improved both sparse and dense models, with a larger effect for the sparse variant. We attribute this to the increased gradient signal that sparse processing provides to early stages, as also observed by Park et al. (2026). The standard benchmark experiments use dataset-specific hyperparameters (Appendix C.2) with varying drop modes and fusion strategies tuned per task.

Tuning is not matched across resolution. This recipe was tuned at 256^2 and transferred unchanged to 512^2; we did not retune per resolution. The reproduced PDE-Transformer baselines instead use the reference configuration of Holzschuh et al. (2025) at both resolutions, since it is their architecture and their tuning is the appropriate reference for it. The compared models therefore share the backbone, data, and hardware, but not the block variants or the tuning. LITE and the SPRINT-style baseline use the gated MLP blocks and GELU activations listed above, which the reference configuration does not. The tuning mismatch points in opposite directions at the two resolutions: at 256^2 our recipe is tuned for the setting in which it is evaluated, whereas at 512^2 it is not, while the baseline is author-tuned throughout. Learning rates also fail to transfer across model scale: in three dimensions, the value used for S and B diverged for LITE-3D L, which required a reduced one (Section 4.3). We therefore report the 512^2 rollout number as measured and do not read it as a property of the dropping mechanism.

B.3 Resolution Scaling

Table 7 reports per-sample FLOPs, per-step training time, and peak GPU memory across three resolutions for the multi-PDE benchmark (model S, batch size 4). All profiling is performed on a single NVIDIA L40S (48 GB) GPU. LITE maintains a consistent $1.76\times$ FLOPs reduction across resolutions, while wall-clock and memory savings grow substantially: at 1024^2, LITE is $5.2\times$ faster per step and uses $4.8\times$ less memory than PDE-Transformer. The 1024^2 row reports profiling results only; we did not train at this resolution but include it to demonstrate favorable scaling. Figure 6 plots these trends, and Figure 7 shows the end-to-end training times of Table 1.

These timings are measured on one L40S and are not directly comparable to the end-to-end training times of Table 1, which use $8\times$L4 GPUs. The two differ most at 256^2, where LITE is $2.08\times$ faster

end-to-end on the L4s but only $1.10\times$ faster per step here. We read this as a hardware effect rather than an inconsistency. The L40S has roughly double the memory and substantially greater bandwidth, so at 256^2 this model does not saturate it and the saving falls below the FLOPs ratio; on the L4, where bandwidth and capacity are the binding constraint, the same reduction in tokens returns more than the FLOPs ratio. As resolution grows the L40S becomes constrained in turn, and its per-step speedup rises to $1.76\times$ at 512^2 and $5.2\times$ at 1024^2. The efficiency of token dropping therefore depends on how tightly the accelerator is bound by memory traffic, and is largest on the smaller GPUs that motivate this work.

Table 7: Resolution scaling (model S, multi-PDE benchmark, patch size 4, batch size 4, single L40S GPU). FLOPs measured per sample; memory is peak GPU allocation during a training step.

Resolution	Model	GFLOPs/sample	Time/step	Peak Memory
256^2	PDE-Transformer S	39.6	89 ms	2.88 GB
256^2	LITE S	22.5	81 ms	1.64 GB
512^2	PDE-Transformer S	158.2	150 ms	10.18 GB
512^2	LITE S	89.8	85 ms	5.18 GB
1024^2	PDE-Transformer S	632.6	606 ms	36.7 GB
1024^2	LITE S	358.9	116 ms	7.6 GB

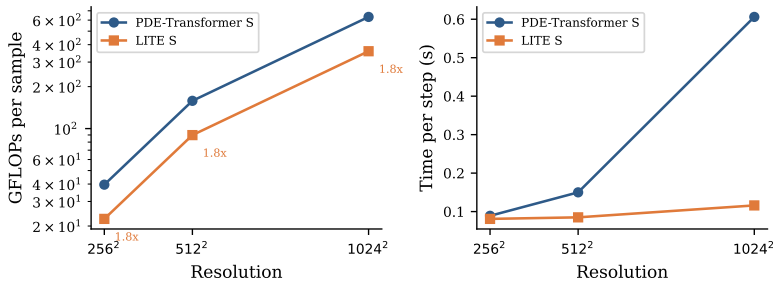

Figure 6: Resolution scaling behavior (model S). LITE maintains a consistent $1.76\times$ FLOPs reduction and improved time scaling across resolutions.

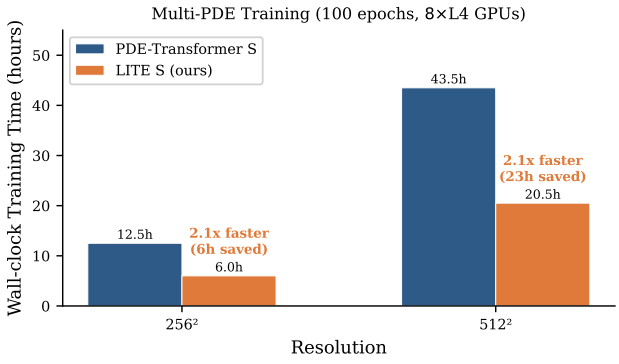

Figure 7: Wall-clock training time on the multi-PDE benchmark (model S, 100 epochs, $8\times$L4 GPUs). LITE saves over 6 GPU-hours at 256^2 and over 20 GPU-hours at 512^2.

C Standard 2D Benchmarks

We evaluate on five benchmarks from the Neural-Solver-Library (Wu et al., 2024): Navier–Stokes turbulence (64×64, 20 time steps, $\nu=10^{-5}$), plasticity (101×31, 20 steps), pipe flow (129×129),

airfoil pressure (221×51), and Darcy flow (85×85, downsampled $5\times$ from 421×421). All experiments use a single NVIDIA L4 GPU (24 GB). We compare against the dense PDE-Transformer baseline, MSPT (Curvo et al., 2026), and Transolver (Wu et al., 2024), all trained from their official implementations with published hyperparameters.

These benchmarks are considerably smaller than the settings LITE is designed for. Their grids contain between 3.1×10^3 and 1.7×10^4 points, one to two orders of magnitude below the 512^2 setting of Table 1 and three to four below the 384^3 volumes of Section 4.3. At this scale the sparse path has little spatial redundancy to exploit and the fixed cost of reconstruction is a larger fraction of the total, so these tasks are not where we expect the method to be most useful. We report them for two reasons: to measure efficiency and memory against established solvers under identical conditions, and to verify across a broad set of equations that per-stage dropping does not degrade accuracy outside the regimes of Section 4. We therefore present the efficiency results first, and then use the accuracy table to establish that these savings do not come at the cost of solution quality.

C.1 Datasets

Table 8 summarizes the five benchmarks.

Table 8: Dataset details for standard PDE benchmarks. All datasets are from the Neural-Solver-Library (Wu et al., 2024). Resolution refers to the grid size used for training after any downsampling.

Dataset	**Resolution**	**Train/Test**	**Input → Output**	**Out dim**	**Type**
Navier–Stokes	64×64	1000 / 200	$10 \to 10$ steps	1	Time-dependent
Plasticity	101×31	900 / 80	$1 \to 20$ steps	4	Time-dependent
Pipe	129×129	1000 / 200	$1 \to 1$	1	Steady-state
Airfoil	221×51	1000 / 200	$1 \to 1$	1	Steady-state
Darcy	85×85	1000 / 200	$1 \to 1$	1	Steady-state

C.2 Hyperparameters

Table 9: Hyperparameters for LITE on standard benchmarks. All experiments share: window size 8, max hidden 512, LR 5.5×10^{-4}, 500 epochs.

	Plasticity	Nav.-Stokes	Pipe	Airfoil	Darcy
Hidden dim	96	128	96	96	96
Heads	8	8	8	8	4
MLP ratio	4	4	3	3	4
Patch size	4	1	2	2	1
Depth	2,4,2	2,4,2	1,3,1	1,3,1	2,4,2
Batch size	8	2	4	8	4
Drop mode	L2	Random	Random	Random	Random
Fusion	Gated	Linear	Linear	Linear	Linear
Output act.	GELU	GELU	—	—	—
Gated MLP	✓	—	—	—	—
Carrier tokens	—	—	—	✓	✓
Periodic	—	✓	—	—	—

C.3 Training Efficiency

Figure 8 compares wall-clock training time and peak GPU memory across methods. All experiments run on a single L4 GPU (24 GB) for 500 epochs under each method's reference hyperparameters. On memory the result is uniform: LITE uses 19–51% less than the dense PDE-Transformer on all five benchmarks (largest on Navier–Stokes, from 10.2 GB to 5.0 GB), and less than Transolver on all five. The one exception across every method and benchmark is MSPT on Navier–Stokes, which peaks at 3.9 GB against LITE's 5.0 GB. MSPT reaches that figure with gradient checkpointing, recomputing activations in the backward pass, and pays for it in time: 99 h against 30 h for LITE on the same task.

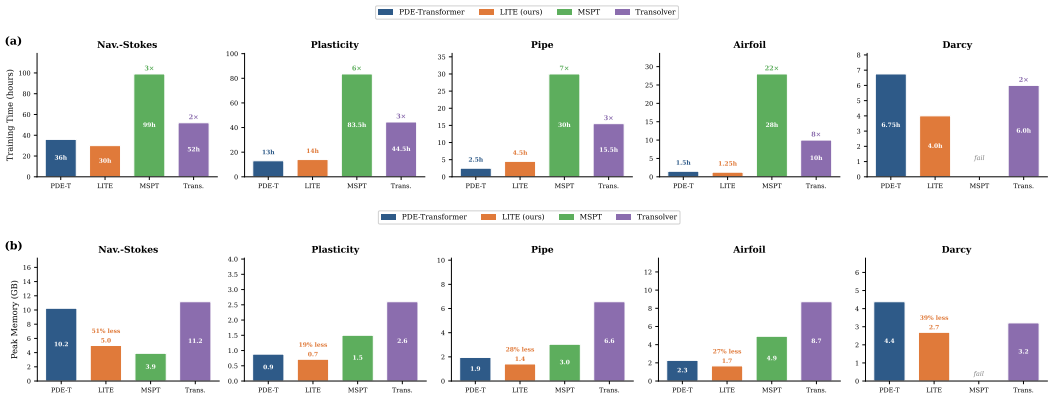

Figure 8: **(a)** Training time and **(b)** peak GPU memory on standard benchmarks. All methods trained for 500 epochs on a single L4 GPU. Slowdown factors are relative to LITE. MSPT employs gradient checkpointing, which reduces peak memory at the cost of increased training time. LITE consistently uses 19–51% less memory than PDE-Transformer without gradient checkpointing.

We therefore do not treat checkpointing as a substitute for dropping, as it buys memory with compute, whereas per-stage dropping reduces both. On time, LITE is faster than MSPT and Transolver by 1.5–22× across the benchmarks they completed, from Transolver on Darcy at the low end to MSPT on Airfoil at the high end (28 h against 1.25 h).

Against the dense PDE-Transformer the training-time picture is mixed. LITE is faster on Navier–Stokes (1.2×), Airfoil (1.2×), and Darcy (1.7×), and slower on Plasticity (14 h against 13 h) and Pipe (4.5 h against 2.5 h). This follows from the scale argument above. Per-stage dropping removes only 1.2–1.5× of the forward FLOPs at these grid sizes (Appendix C.4), against 1.76× at the resolutions of Table 1, while the cost of scoring, spreading, and fusion is incurred at every stage regardless of how few tokens that stage holds. Where the margin is thin the reconstruction overhead can consume the attention it saves, and the memory reduction is obtained at no wall-clock benefit. We report these benchmarks to establish that per-stage dropping transfers across equation families and geometries without loss of accuracy, not to claim a speedup at this scale.

C.4 Per-Sample FLOPs

Figure 9 compares per-sample FLOPs across all methods on the five standard benchmarks, measured in eval mode with batch size 1 using PyTorch's `FlopCounterMode`. LITE consistently requires fewer FLOPs than PDE-Transformer (by 1.2–1.5×), while MSPT and Transolver require 1.9–123× more, from Transolver on Darcy at the low end to MSPT on Darcy at the high end, due to their full-attention or slice-attention mechanisms. MSPT's training cost is further amplified by gradient checkpointing, which is not reflected in these forward-pass measurements.

C.5 Accuracy

Table 10 reports relative L_2 error against published baselines from Curvo et al. (2026), spanning spectral methods (FNO, geo-FNO, LSM) and transformer methods (Galerkin, OFormer, ONO, Transolver, MSPT). The efficiency gains above do not come at the expense of accuracy: LITE stays within 0.6×10^{-2} relative error of the dense PDE-Transformer on all five tasks, and is the best or second-best method on three of them. The only notable gap is on Darcy flow, whose static elliptic character offers less spatial redundancy for the sparse path to exploit.

C.6 MSPT on Darcy Flow

We attempted to reproduce MSPT on Darcy flow using the official implementation. The run completed 267 of 500 epochs in 175 hours (∼7.3 days) before crashing, at a rate of ∼0.66 hours per epoch. At this rate, the full training would require an estimated 328 hours (∼13.7 days) on a single L4 GPU.

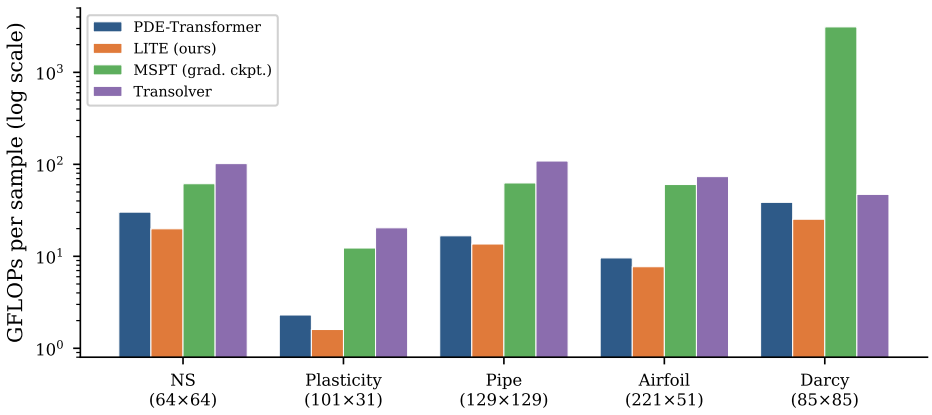

Figure 9: GFLOPs per forward pass on standard benchmarks (log scale, eval mode, batch size 1). LITE consistently uses fewer FLOPs than PDE-Transformer. MSPT and Transolver require substantially more FLOPs. MSPT additionally uses gradient checkpointing during training, incurring further recomputation overhead not shown here.

Table 10: Standard PDE benchmark results (relative L_2 error $\times 10^{-2}$ ↓). Best in **bold**; second best and within 2×10^{-4} of it underlined. Baselines from Curvo et al. (2026) except where noted. †Reproduced on our hardware. ‡Published value; MSPT on Darcy did not complete within our compute budget (see Appendix C.6). Column order matches Figure 8.

Model	Nav.-Stokes	Plasticity	Pipe	Airfoil	Darcy
Spectral / FNO family					
FNO (Li et al., 2021)	15.56	—	—	—	1.08
geo-FNO (Li et al., 2023c)	15.56	0.74	0.67	1.38	1.08
LSM (Wu et al., 2023)	15.35	0.25	0.50	0.59	0.65
Transformer-based					
Galerkin (Cao, 2021)	14.01	1.20	0.98	1.18	0.84
OFormer (Li et al., 2023a)	17.05	0.17	1.68	1.83	1.24
ONO (Xiao et al., 2024)	11.95	0.48	0.52	0.61	0.76
Transolver (Wu et al., 2024)†	9.00	0.12	0.33	0.53	**0.57**
MSPT (Curvo et al., 2026)†	6.32	0.10	**0.31**	**0.51**	0.63‡
PDE-Trans. (Holzschuh et al., 2025)†	**5.70**	**0.03**	0.32	0.68	0.64
Ours (single L4 GPU)					
LITE (ours)	6.30	**0.03**	0.32	0.74	0.72

For comparison, LITE trains on Darcy in 0.69 hours and PDE-Transformer in 0.83 hours for 500 epochs. The MSPT run achieved a validation relative L_2 error of 0.0063 at epoch 267, consistent with the published value of 0.63 ($\times 10^{-2}$), suggesting the model had largely converged before the crash. We therefore report the published value in Table 10 rather than our partial reproduction.

D LITE Components and Ablations

D.1 Components

The LITE architecture augments PDE-Transformer with the following additional components (parameter counts for model S with d=192, d_{out}=384):

- **Mask token:** $\mathbf{m} \in \mathbb{R}^{d_{\text{out}}}$, initialized from $\mathcal{N}(0, 0.02^2)$ (truncated normal).
- **Spread convolution:** Depthwise conv with d_{out} channels, initialized to a uniform box filter. In 2D the kernel is 2×2 (weights $1/4$), costing $4 \cdot d_{\text{out}} = 1536$ parameters; in 3D it is $3\times3\times3$ (weights $1/27$), costing $27 \cdot d_{\text{out}}$ parameters.

- **Linear fusion (default):** A single affine map $\mathbf{W} \in \mathbb{R}^{d_{\text{out}} \times 2d_{\text{out}}}$ with bias $\boldsymbol{\beta} \in \mathbb{R}^{d_{\text{out}}}$ applied to the channel-wise concatenation of the spread features and the stage input, followed by SiLU (Eq. 4). Parameters: $2d_{\text{out}}^2 + d_{\text{out}}$.
- **Total overhead:** $< 2\%$ additional parameters relative to the base model.

Optional gated fusion. We also implement an input-dependent alternative to Eq. 4, in which a learned gate blends the spread features with the stored dense features through a residual highway:

$$\mathbf{g} = \sigma\big(\text{MLP}([\mathbf{z}_{\text{spread}}; \mathbf{z}_{\text{dense}}])\big), \quad \mathbf{z}_{\text{out}} = \mathbf{z}_{\text{dense}} + \mathbf{g} \odot \phi(\mathbf{W}_f \mathbf{z}_{\text{spread}}), \tag{5}$$

where σ is the sigmoid, ϕ is SiLU, and the gate is a two-layer MLP ($2d_{\text{out}} \to d_{\text{out}} \to d_{\text{out}}$, SiLU activation, sigmoid output) accompanied by a linear projection $\mathbf{W}_f \in \mathbb{R}^{d_{\text{out}} \times d_{\text{out}}}$, costing $\approx 4d_{\text{out}}^2 = 589{,}824$ parameters. Unlike Eq. 4 this form is residual, preserving the dense features as an explicit highway. This variant helps only in two dimensions, where we use it for the multi-PDE results (Table 1) and the Plasticity benchmark. In three dimensions it gave no improvement over Eq. 4, so every 3D experiment uses linear fusion.

D.2 Mechanism Ablation

The 2D ablations in this appendix use model XS ($\sim$18.8M parameters), 6 PDE types from APEBench (hyp, fisher, gs_beta, gs_theta, gs_kappa, kolm_flow), 25 epochs on a single L4 GPU at 256^2. Table 11 starts from naive token dropping (random selection, linear fusion) and adds components one at a time in a single sequence. The sequence also added two recipe settings, a GELU output activation and gated MLP blocks (Appendix B.2); they are not part of the dropping mechanism and are not shown in the table.

Table 11: Mechanism ablation (model XS, 6 PDE types, 25 epochs, 256^2). Relative reduction in validation loss when each component is added.

Component	Val. loss reduction
Spread convolution	6.4%
ℓ_2 importance scoring	2.6%†
Gated fusion *(2D only)*	4.6%

†Within the resolution of this measurement (Appendix D.3).

Because the components are added in sequence, the reductions depend on the order and should not be read as independent effect sizes. The sequence ends at the configuration used for the 2D results of Table 1, gated fusion included. That fusion is not the default elsewhere: in three dimensions it gave no improvement over the linear form, so all 3D experiments use the linear form and the results of Section 4.3 do not depend on it. The selection rule is examined on its own in Appendix D.3.

D.3 Selection Rule: ℓ_2 Scoring against Random

Contribution (ii) replaces random within-block selection with ℓ_2-norm scoring. Table 11 varies it as one step of a cumulative sequence; here we vary the selection rule alone, holding backbone, recipe, data and budget fixed.

Keep-ratio sweep in two dimensions. Table 12 sweeps the keep ratio on model XS over the six PDE types listed above. So that ratios as low as $1/64$ are reachable, selection here operates within 8×8 windows rather than the 2×2 blocks used everywhere else in the paper. The two are not the same configuration at a matched ratio: keeping 16 of 64 tokens permits the spatial clustering that keeping 1 of 4 excludes by construction.

We draw no trend from this sweep. The difference between the rules is not ordered by keep ratio: across the seven settings its sign changes four times, and the correlation between tokens kept and the ℓ_2 advantage is $+0.21$, the opposite of the sign a "harsher dropping favours scoring" reading would require. The measurement is also close to its own resolution. Validation loss moves by under 3% between keeping 1 token per window and keeping 49, so at this scale and budget the objective barely separates configurations that differ by a factor of 49 in token count, let alone by a choice of which token to keep.

Table 12: Selection rule against keep ratio (model XS, 8×8 selection windows, 6 PDE types, short training). Neither rule is consistently ahead: ℓ_2 wins three of seven ratios on each metric, and the sign of the difference changes four times across the sweep.

		Validation loss		nRMSE_{10}	
Keep ratio	Kept	ℓ_2	Random	ℓ_2	Random
$1/64$	1	0.0224	0.0225	1.237	1.284
$1/16$	4	0.0221	0.0226	1.237	1.192
$9/64$	9	0.0223	0.0217	1.172	1.213
$1/4$	16	0.0222	0.0218	1.238	1.224
$25/64$	25	0.0221	0.0219	1.197	1.156
$9/16$	36	0.0218	0.0215	1.176	1.132
$49/64$	49	0.0218	0.0219	1.168	1.187

Where the two rules do differ. Table 13 breaks the harshest setting down by equation. The difference tracks the spatial structure of the solution rather than the keep ratio. On the three Gray–Scott variants, whose dynamics concentrate in localized reaction fronts, ℓ_2 selection is ahead by 6.2% in aggregate. On the three fields whose activity fills the domain it is behind by 1.4%.

Table 13: Per-equation rollout error at the harshest ratio ($1/64$, rollout step t=7). The advantage of ℓ_2 selection is confined to the equations with localized structure.

Equation	Structure	ℓ_2	Random
gs_beta	localized fronts	1.118	1.134
gs_theta	localized fronts	1.957	2.161
gs_kappa	localized fronts	1.504	1.585
hyp	advective, smooth	0.749	0.717
fisher	fills the domain	0.607	0.607
kolm_flow	forced turbulence	0.990	0.990

Three dimensions. The 2D sweep above uses model XS on six equations at reduced budget. Repeating the comparison on the full-resolution KdV experiment of Table 3, which uses the 3D models and the keep ratio of $1/8$ actually reported, replacing ℓ_2 scoring with random selection and changing nothing else raises terminal nRMSE from 0.4823 to 0.4877 for LITE-3D S (1.1%) and from 0.4369 to 0.4457 for LITE-3D B (2.0%). Both model sizes move in the same direction, and KdV develops solitons, the localized structure that Table 13 associates with the difference.

Reading these results. The effect of the selection rule is modest and structural. It appears where the solution concentrates in a small part of the domain and is absent where activity is spread across it, which is the behaviour the ℓ_2 score is meant to produce; it is not a function of how aggressive the dropping is. At the scale of the 2D sweep the difference is within the resolution of the measurement, and we report it as such. These are single runs under the fixed seed of Section 4.

E Why Fine-Tuning-Based Sparse Training Does Not Scale

Several sparse-training methods recover accuracy by pairing a sparse phase with a full-token phase. SPRINT (Park et al., 2026) trains with whole-network dropping and then fine-tunes on all tokens to close the gap between the sparse training distribution and dense inference. Masked-pretraining approaches for PDE surrogates follow the same pattern: MAE-PDE (Zhou & Barati Farimani, 2024) masks patches during pretraining, but the downstream model is fine-tuned and deployed densely, so the masking reduces pretraining cost without reducing the cost of the model that is ultimately used.

The difficulty is that the full-token phase runs transformer blocks over the complete token grid and must therefore materialize the corresponding activations. Its peak memory is that of training the dense model, and no amount of sparsity in the other phase reduces it. In two dimensions this bounds

the achievable saving but leaves the method usable. In three dimensions it decides feasibility, because memory rather than wall-clock time determines which configurations can be trained at all.

Our measurements make the consequence concrete. On full 384^3 volumes P3D-Patch B exhausts memory on a 48 GiB card even at micro-batch 1 (Table 3), so a method requiring a full-token phase cannot be applied there, since the phase it depends on is exactly the one that does not fit. Sparse training alone would produce a model that could never be adapted back to full tokens, and the memory ceiling of the dense baseline transfers to any method layered on top of it. At 256^3 the full-token phase of the L backbone fits on that card but sets the peak at 28.9 GiB, $2.8\times$ the 10.2 GiB of LITE-3D L and more than a 24 GiB GPU holds (Table 2).

Omitting the fine-tuning phase does not resolve this. Our SPRINT-style baseline applies whole-network dropping to the PDE-Transformer backbone at the same 25% keep rate and is trained and evaluated without a full-token phase, matching the regime in which such a method would have to operate at 384^3. It reaches nRMSE_{10} of 0.5333 (Table 1), 48% worse than the dense baseline it is intended to accelerate. This is the sense in which the fine-tuning stage is load-bearing rather than incidental: it supplies accuracy that the sparse phase does not reach on its own.

Methods of this family therefore face a dilemma that sharpens with resolution. Retaining the full-token phase inherits the dense memory ceiling and forfeits the settings where sparsity matters most; removing it forfeits the accuracy that motivated the design. LITE is not subject to the trade-off because it has no full-token phase to begin with. Transformer blocks operate on the retained subset at every stage, and each stage returns to full resolution through a depthwise spread convolution and a linear fusion, whose cost is small beside attention. Training and inference use the same sparse configuration, so there is no train–inference gap for an adaptation phase to close.

F Qualitative Results

F.1 3D Rollouts

Figures 10–13 show sample rollouts of LITE-3D L (Table 2) at 256^3, rendered as volumes against the reference solver. Boxed numbers give the nRMSE of the prediction they label.

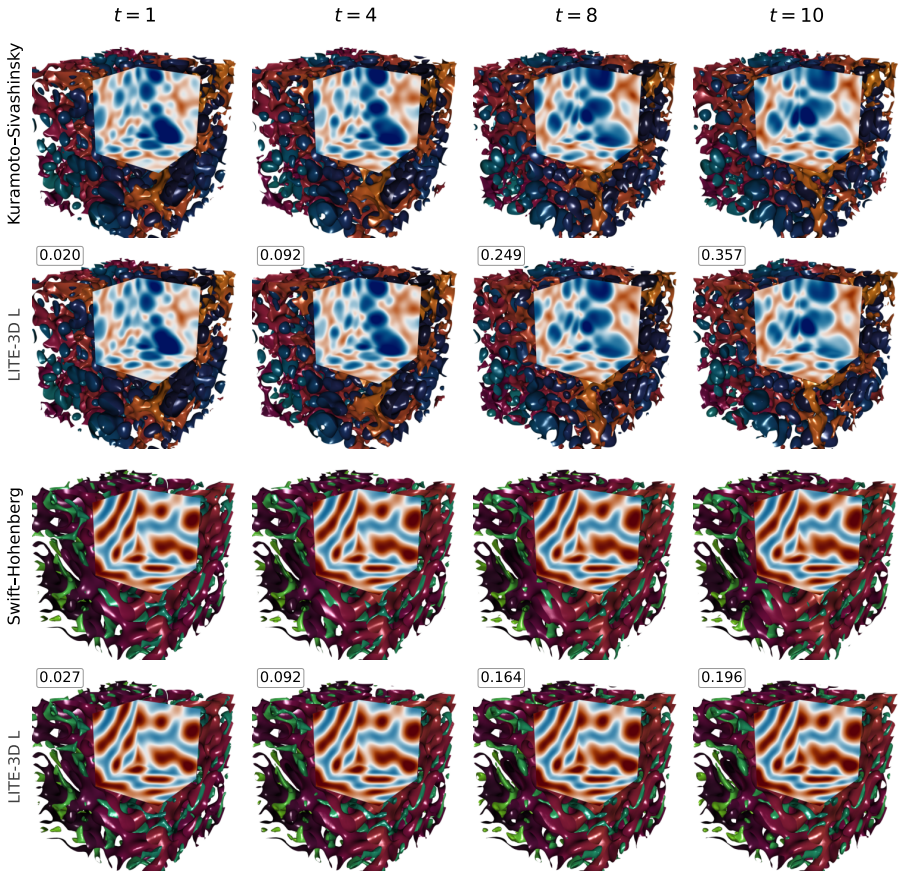

Figure 10: Sample rollout on Kuramoto–Sivashinsky and Swift–Hohenberg (256^3), shown as octant cutaways at steps t=1, 4, 8, 10. Top row of each pair is the reference, bottom row LITE-3D L.

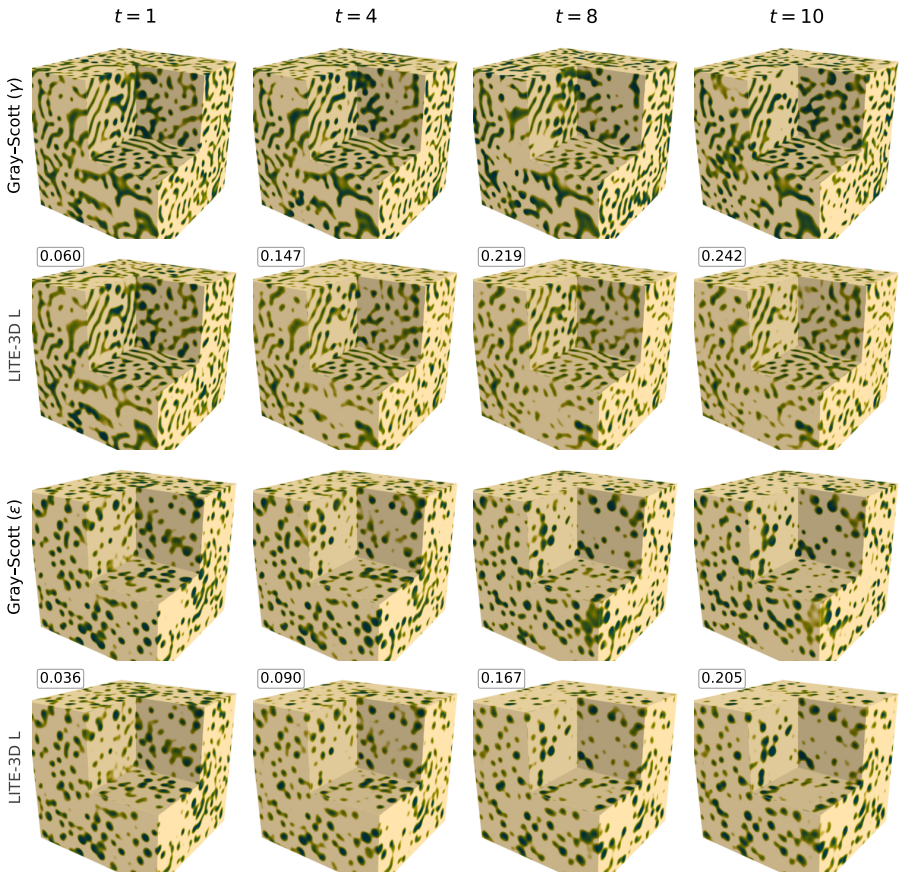

Figure 11: Sample rollout on two Gray–Scott pattern regimes (256^3) at steps t=1, 4, 8, 10. LITE-3D L tracks both the labyrinthine (γ) and spot (ϵ) morphologies over the full horizon.

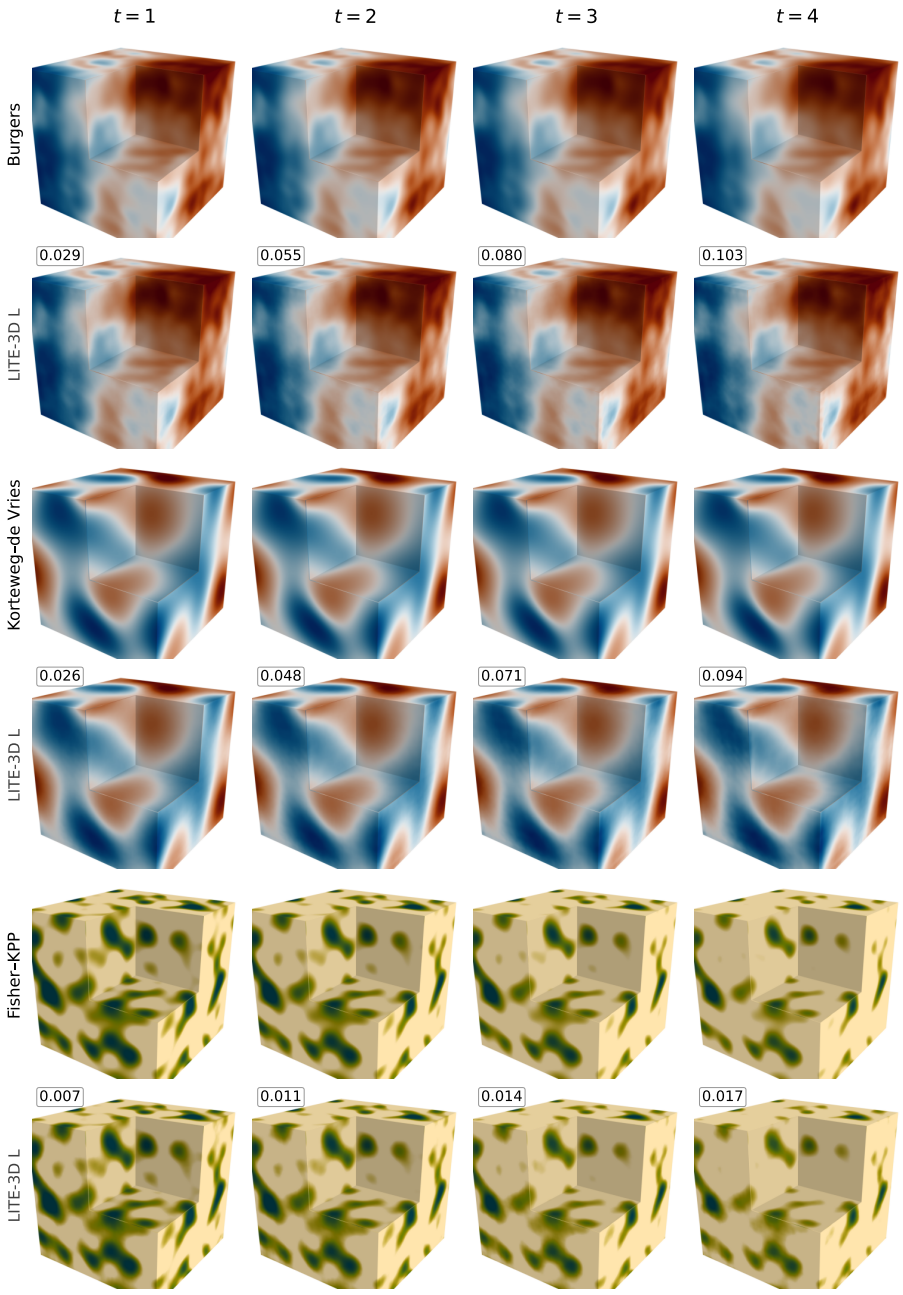

Figure 12: Sample rollout on Burgers, Korteweg–de Vries and Fisher–KPP (256^3) at steps t=1, 2, 3, 4.

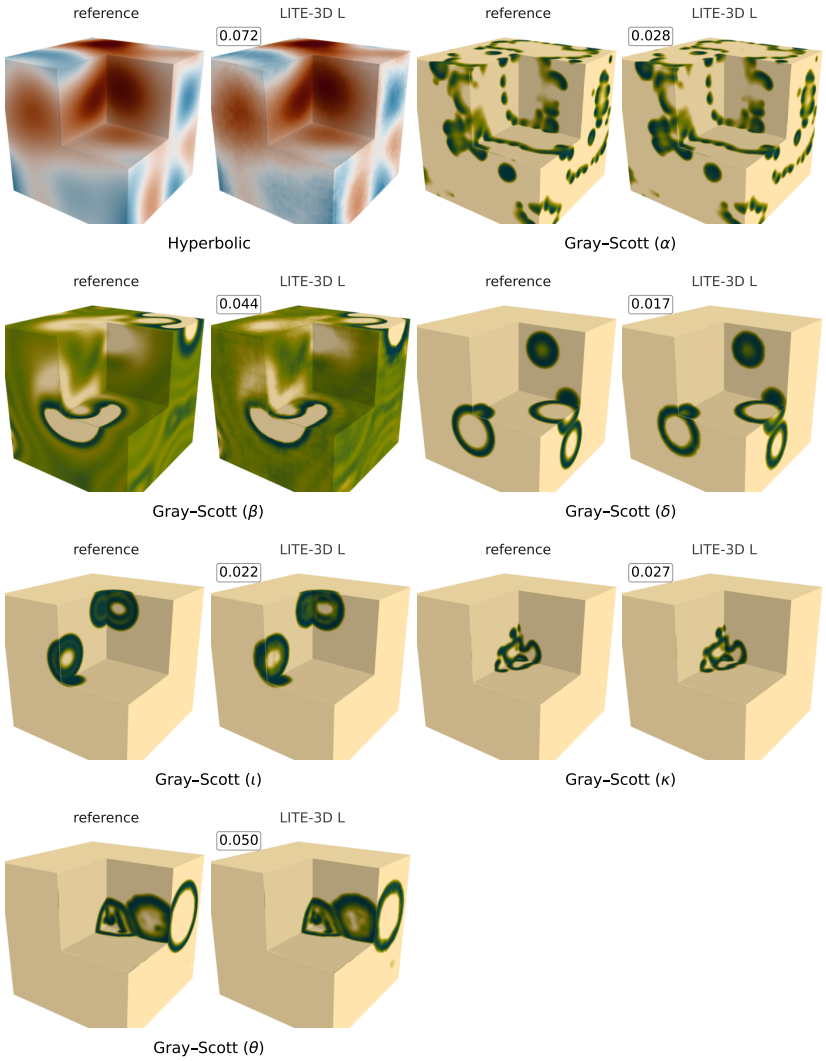

Figure 13: Sample single-step predictions (t=1, 256^3) on the remaining seven datasets (nRMSE 0.017–0.072).

F.2 2D Predictions

Figures 14 and 15 show representative predictions from LITE S and the dense PDE-Transformer S baseline on all 16 PDE families in the multi-PDE benchmark at 256^2 resolution. Each row depicts a different PDE; columns show the ground truth, PDE-Transformer prediction, LITE prediction, and absolute error maps for both models at rollout step 5. The predictions are visually indistinguishable from ground truth across all 16 PDE types. Error magnitudes are comparable between LITE and PDE-Transformer, confirming that token sparsification does not introduce systematic spatial artifacts despite processing only 25% of tokens at each stage.

F.3 2D Rollout Stability

To assess autoregressive rollout stability, Figures 16 and 17 compare signed error maps of PDE-Transformer S and LITE S at rollout steps t=1, t=5, and t=9 across all 16 PDE families. At early steps (t=1), LITE error is comparable to or lower than PDE-Transformer across most PDE types. As the rollout progresses, both models accumulate error, yet LITE does not exhibit faster divergence despite processing only 25% of tokens per stage. This is consistent with per-stage reconstruction preserving spatial coherence over extended autoregressive horizons.

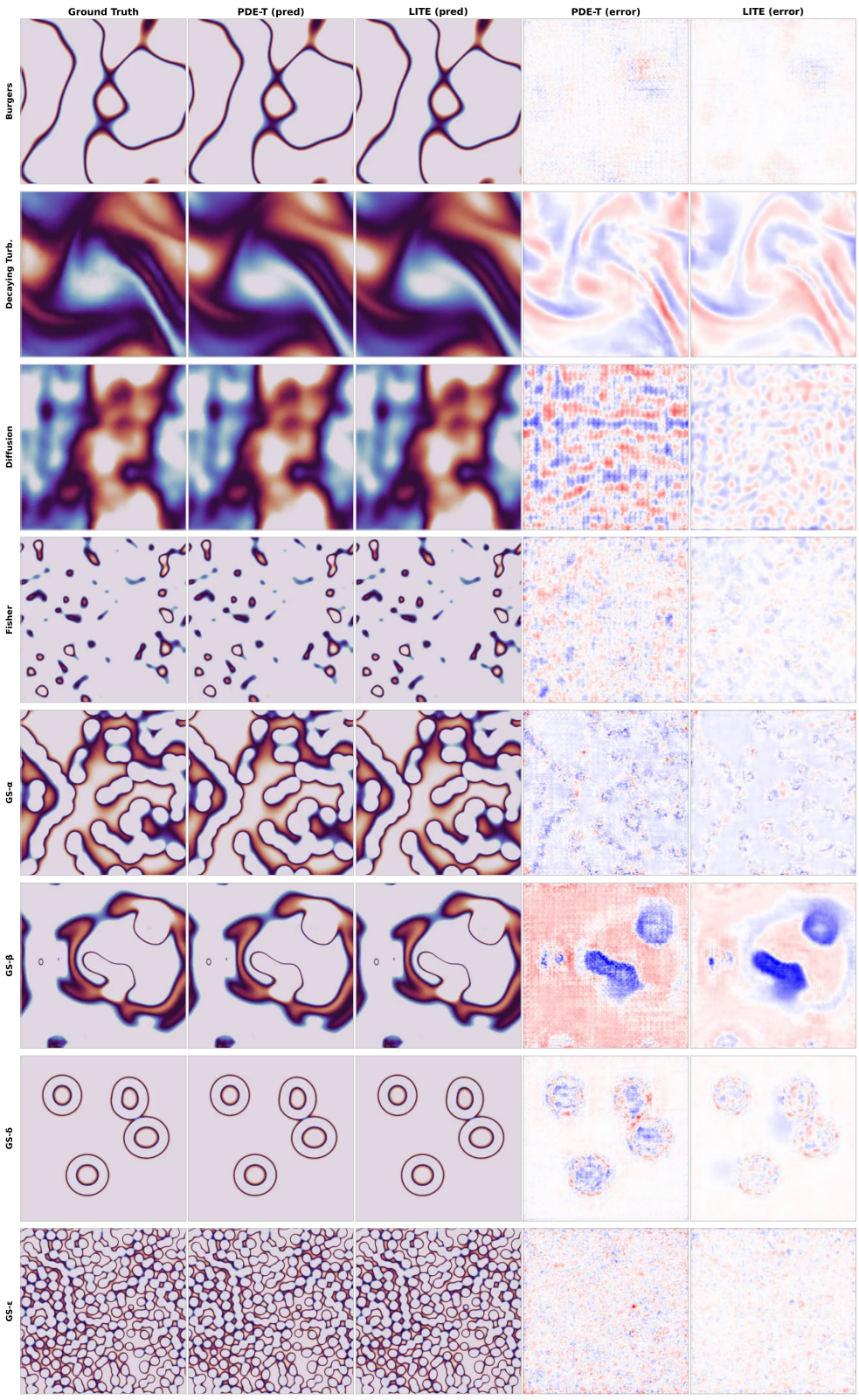

Figure 14: Qualitative predictions on PDE families 1–8 from the multi-PDE benchmark (256^2, rollout step 5). Columns: ground truth, PDE-Transformer prediction, LITE prediction, PDE-Transformer error, LITE error.

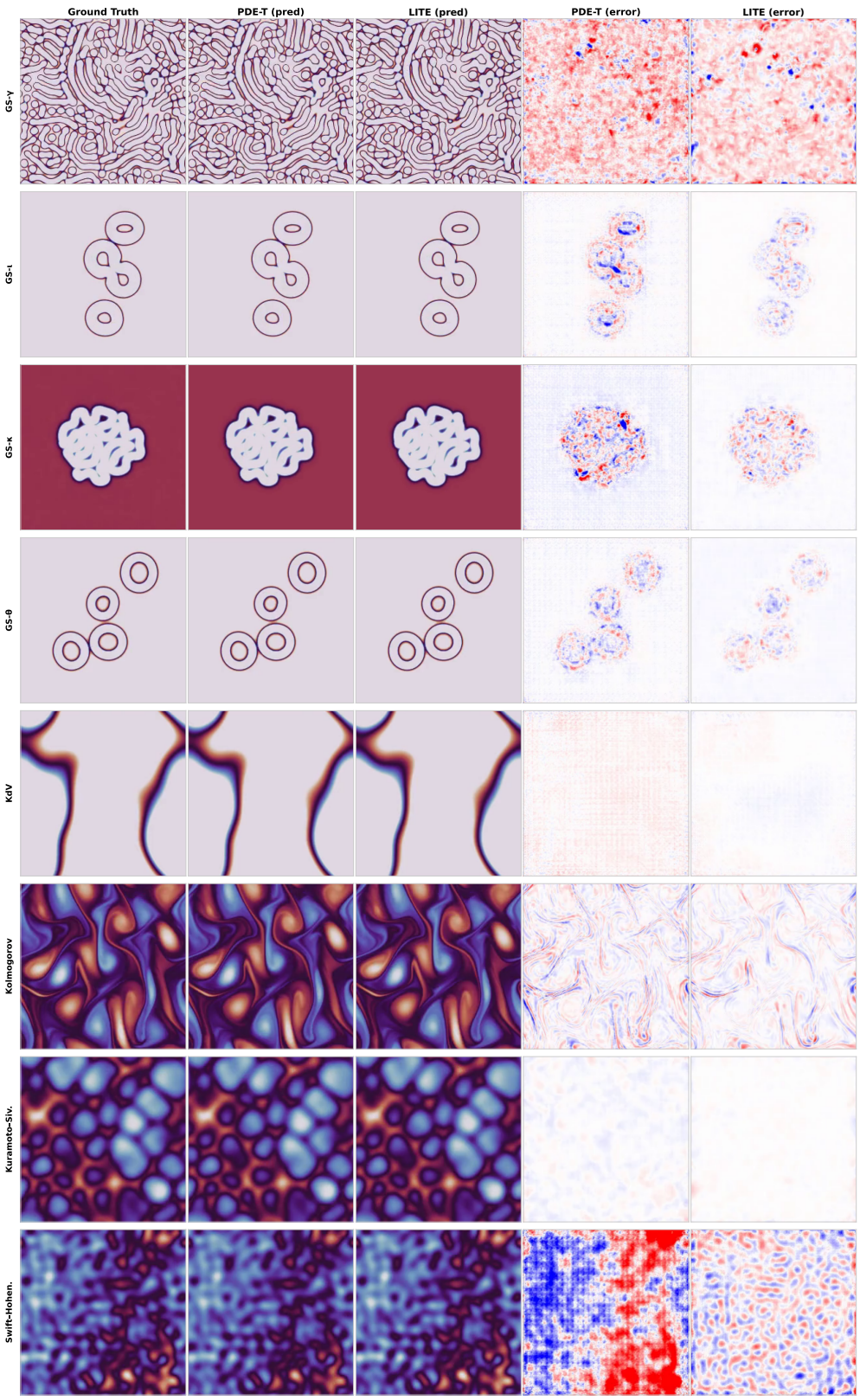

Figure 15: Qualitative predictions on PDE families 9–16 (continued). Both models produce visually indistinguishable predictions; error magnitudes are comparable despite 75% token sparsification.

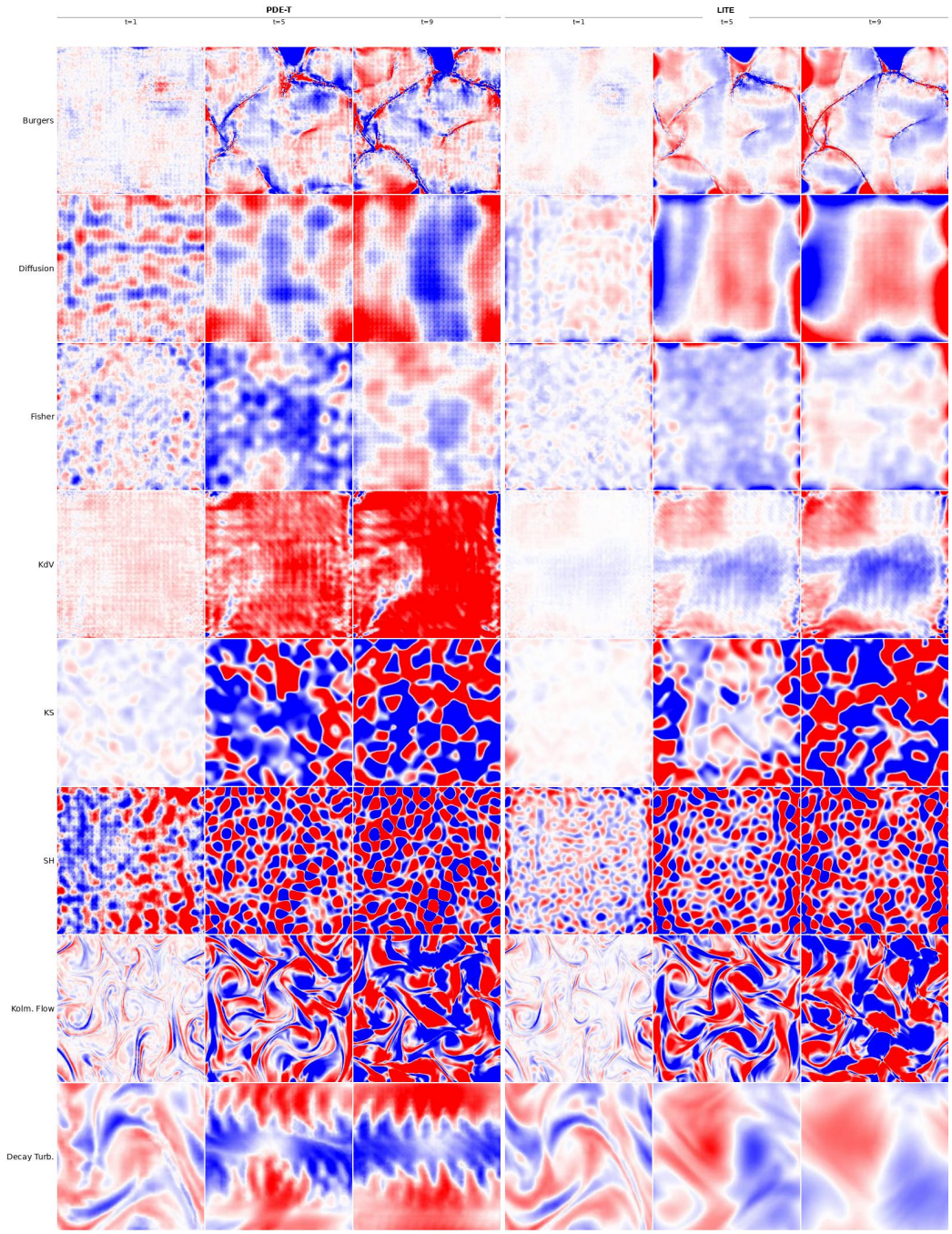

Figure 16: Rollout error comparison (PDE families 1–8, 256^2). Columns: PDE-Transformer error and LITE error at rollout steps t=1, t=5, and t=9. Error magnitude grows with rollout length for both models; LITE does not degrade faster.

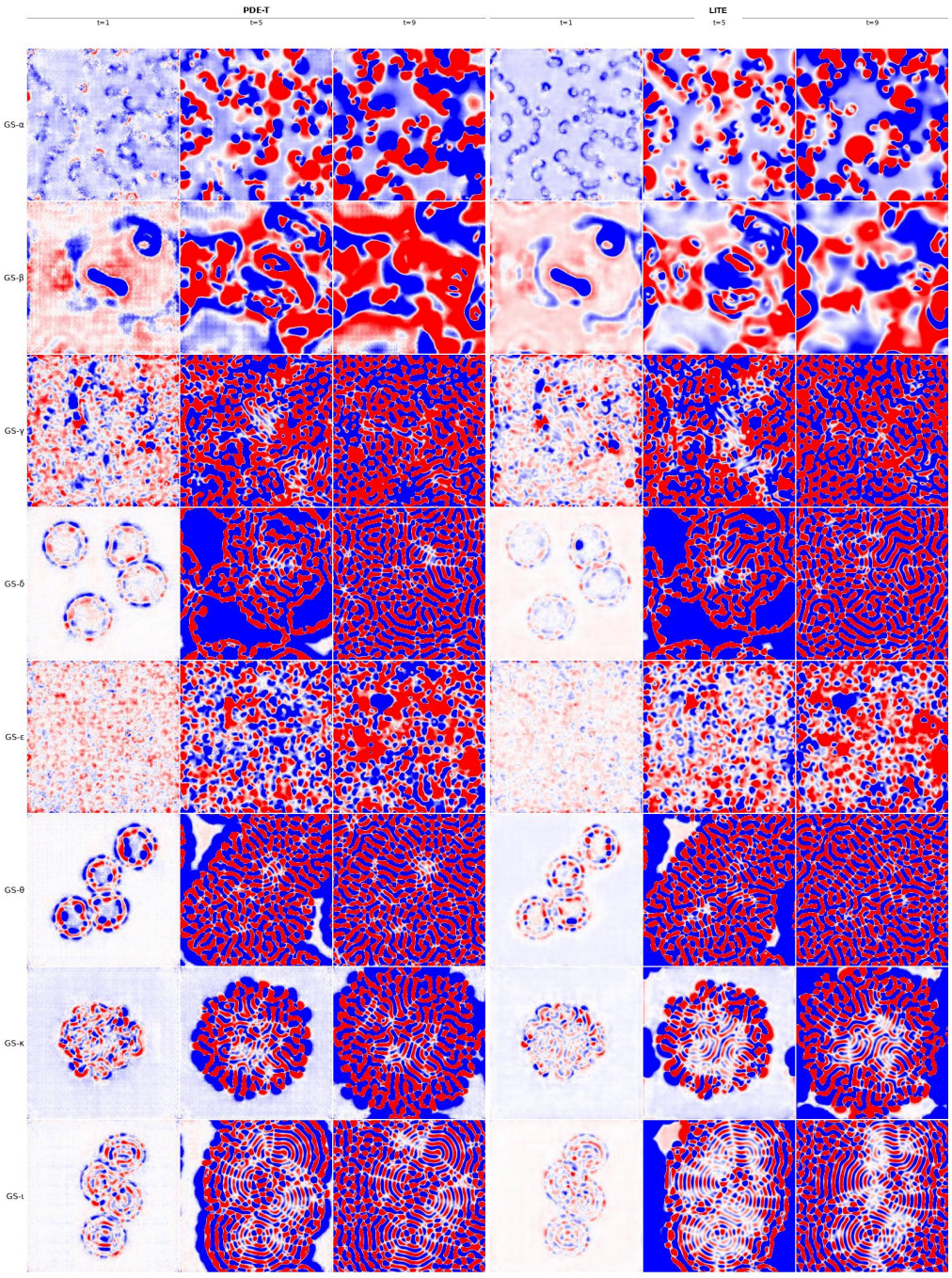

Figure 17: Rollout error comparison (PDE families 9–16, continued). LITE maintains comparable rollout stability to the dense baseline across all PDE types.