Dynamic Quadtree Tokenization and Transformer for Adaptive Mesh PDE Forecasting

Yilin Zhuang **Noah Zambrano** **Karthik Duraisamy***
Department of Aerospace Engineering
Michigan Institute for Computational Discovery and Engineering
University of Michigan, Ann Arbor, MI
{ylzhuang, nzamb, kdur}@umich.edu

Abstract

The quadratic attention cost of Vision Transformers (ViTs) forces a trade-off between spatial resolution and rollout horizon, particularly for fine-scale PDEs where shocks, reaction fronts, and material interfaces occupy small, evolving regions of the domain. Conventional neural surrogates also lack mechanisms to adapt resolution dynamically. We propose WAMRViT, a ViT that tokenizes inputs as balanced quadtrees using a wavelet-inspired refinement criterion, jointly encodes position and refinement level with 3D rotary positional embeddings, and regrids in cell space during inference for stable long-horizon rollouts. A multi-scale variant retains each leaf at its native source resolution and lets the model learn across resolution levels. Unlike fixed-budget adaptive-tokenization methods, WAMRViT imposes no predetermined token count and supports fully adaptive topology throughout autoregressive rollout. To our knowledge, it is the first machine-learning surrogate to natively tokenize multi-level Adaptive Mesh Refinement (AMR) data. On uniform-grid benchmarks, uniform-patch WAMRViT improves finest-level region-of-interest VRMSE over a finest-patch uniform ViT while using substantially fewer tokens. The multi-scale variant achieves the lowest first-step full-field RMSE and VRMSE on both benchmarks and improves rollout-averaged full-field and refined-region accuracy at long horizons. With parallelized regridding, its end-to-end rollout cost lies between finest-patch and approximately token-matched coarser-patch ViTs. On a complex AMR combustion problem whose finest features cannot be represented natively by the evaluated uniform-grid baselines, WAMRViT operates on adaptive cells built from the simulator hierarchy and substantially reduces finest-level error at matched transformer capacity. Code: https://github.com/tonyzyl/wamrvit.

1 Introduction

Deep-learning surrogates have become a practical alternative to classical PDE solvers in computational fluid dynamics, combustion, and shock hydrodynamics [1–4]. Among these, Vision Transformers (ViTs) are attractive because self-attention directly represents long-range correlations that can be difficult for local convolutional or graph operators to capture [5]. The price is quadratic cost in the number of tokens. This cost can impose an accuracy ceiling: on standard vision benchmarks, ViTs improve as patch size shrinks and, for a fixed backbone, can perform best when individual pixels become tokens [6]. For fine-scale PDE forecasts, a finest-grid ViT can exceed practical compute and memory budgets as spatial resolution and rollout horizon grow.

This bottleneck is not inherent. In the complex PDE regimes where transformer surrogates are most valuable, such as detonation fronts and multi-material shocks, the dynamics are spatially sparse, and

*Corresponding author: kdur@umich.edu

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

a uniform tokenization strategy squanders the $\mathcal{O}(N^2)$ self-attention budget on dynamically quiescent regions where the physical fields are largely homogeneous. Classical numerical analysis addresses this mismatch via Adaptive Mesh Refinement (AMR) [7, 8]: the solver commits compute to where it is needed and coarsens elsewhere, producing a hierarchical mesh.

Adaptive-patching ViTs already exploit spatially varying resolution [9–13]. Bringing this idea closer to AMR practice calls for combining three capabilities: (i) a recursive, neighbor-balanced hierarchy with mean-conservative coarsening and explicit inter-scale communication; (ii) direct ingestion of simulator-native AMR data; and (iii) repeated topology updates and state transfer during autoregressive forecasting, where errors compound and active regions move.

Contributions. We introduce **WAMRViT**, a transformer surrogate whose tokens are the leaves of a 2:1-balanced quadtree adapted to each input field. The hierarchy and token count evolve with predicted features, allowing computation to follow local structure throughout rollout. Our contributions address the three capabilities above.

Hierarchical, balanced tokenization. Our tokenizer employs a 2:1-balanced quadtree driven by a wavelet-inspired, non-learned local-range refinement criterion (Section 2.1). The tree supplies explicit face-neighbor relations and mean-conservative coarsening. Spatial positioning is encoded via a 3D rotary embedding [14] extended to the coordinates (x, y, ℓ). Crucially, the refinement level ℓ acts as a soft attention bias rather than a hard mask, allowing tokens at different levels to interact across coarse–fine boundaries where cross-level communication is necessary.

Native AMR ingestion and resolution-preserving uniform-grid forecasting. Uniform-patch WAMRViT reads a simulator's multi-level AMR hierarchy through a shared leaf interface. For uniform-grid inputs, the multi-scale variant retains each leaf's full source region and resolves inter-level resolution differences within the model through per-level patch embedders, resolution-specific decoders, and level-conditioned AdaLN. Both paths produce one token per leaf, supporting adaptive forecasting on native AMR hierarchies and uniform-grid fields.

Long-horizon adaptive rollout. To our knowledge, WAMRViT is the first machine-learning surrogate to perform autoregressive forecasting on a native multi-level AMR hierarchy while dynamically updating its topology and transferring predicted states between successive grids. Refinement and coarsening track moving features rather than freezing the grid, and the token count follows the evolving field rather than a prescribed budget. Parallel CPU-array regridding and a hybrid CPU/GPU path that minimize data exchange make this evolving-grid workflow practical.

Related work. The closest prior methods are adaptive-patching ViTs for PDE surrogates: AMR-Transformer [9], MATEY [12], and IAP [13]. Overtone [15] instead trains one ViT to operate at several uniform-grid patch sizes and cycles those sizes across rollout steps. Earlier adaptive neural-operator work targets chemical kinetics rather than the full flow field [16]. These methods do not evaluate natively multi-resolution AMR data under a moving autoregressive topology. MATEY's reported experiments use two-level partitioning and do not evaluate hierarchy depth. IAP releases only partial code, and we did not include a matched rollout implementation. WAMRViT addresses the resulting systems gap between adaptive tokenization and AMR practice: native multilevel input, balanced hierarchical topology, conservative inter-level transfer, variable token counts without a prescribed budget, and inference-time regridding during autoregressive prediction. Appendix H provides a comprehensive comparison.

2 Method

Let Q_t be the current 2:1-balanced quadtree with N_t leaves, meaning face-neighboring leaves differ by at most one refinement level. In the default uniform-patch path, the T_{in} input frames are represented on Q_t as $x_{t-T_{\text{in}}+1:t}|_{Q_t} \in \mathbb{R}^{N_t \times C \times T_{\text{in}} \times p_h \times p_w}$, where each leaf carries a fixed-size spatial raster. A shared tubelet convolution [17] with temporal patch size $p_t = T_{\text{in}}$ spans the temporal window and leaf raster to produce one d-dimensional token per leaf. The transformer processes this variable-length sequence using each leaf's center and level (x, y, ℓ) through 3D Rotary Positional Embeddings (RoPE) [14], and a shared output projection decodes every token into T_{out} predicted frames on the same leaves.

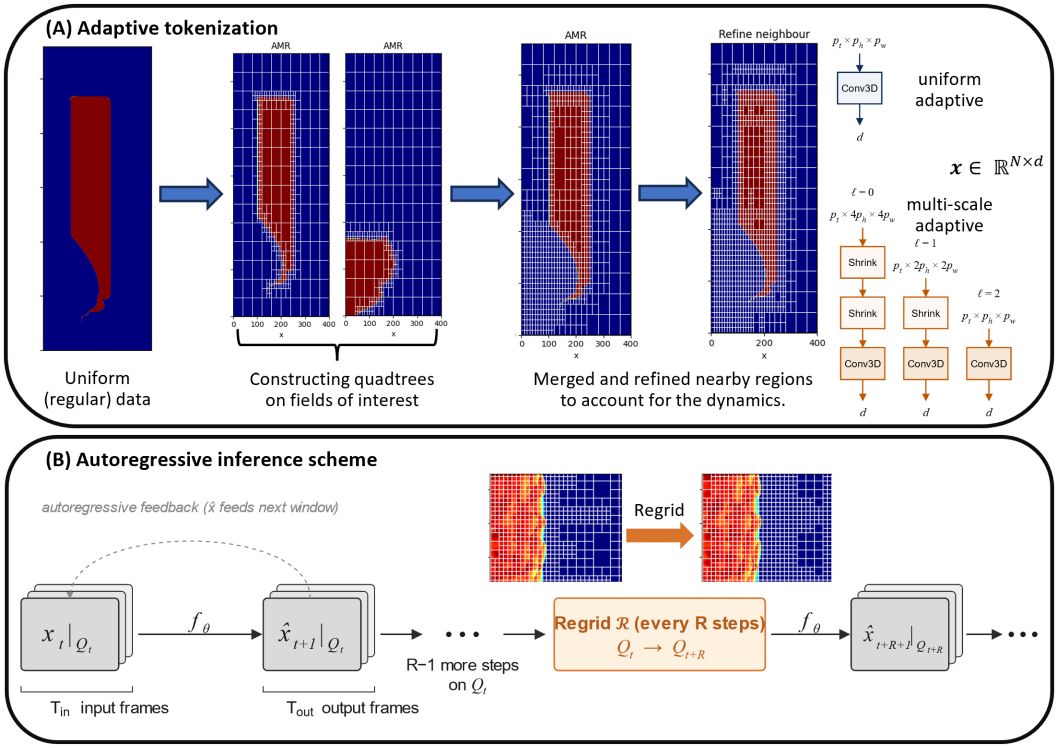


Figure 1: **Overview of WAMRViT.** (a) Adaptive tokenization: the input field is partitioned into a 2:1-balanced quadtree by a wavelet-inspired local-range criterion, with one token per leaf. (b) Autoregressive inference: every R steps a refine-protect-coarsen update revises the quadtree in cell space so the topology tracks moving features.

A model forward pass advances the field while leaving the topology unchanged. Prediction and periodic topology update are distinct operations:

$$\begin{aligned} \hat{x}_{t+1:t+T_{\text{out}}}|_{Q_t} &= f_\theta(x_{t-T_{\text{in}}+1:t}|_{Q_t})\,, \\ \big(Q_{t+R},\, \hat{x}|_{Q_{t+R}}\big) &= \mathcal{R}(Q_t,\, \hat{x}|_{Q_t}) \quad \text{every } R \text{ rollout steps.} \end{aligned} \tag{1}$$

where, in the regridding relation, $\hat{x}$ denotes the state retained after advancing R rollout steps on Q_t, including the history needed for the next input window, and Q_{t+R} is the topology updated at that boundary. Regridding transfers this state without advancing time. Section 2.4 summarizes the refine-protect-coarsen operator $\mathcal{R}$, and Appendix C.10 gives the complete algorithm.

2.1 Adaptive Tokenization and Positional Encoding

The spatial domain is tiled by root cells that may split recursively into four children up to level L. For uniform-grid inputs, a leaf is marked for refinement when the local range of any selected adaptation channel exceeds ε times that channel's global range. Per-channel decisions are unioned. This non-learned local-extremum criterion bounds localized Haar-wavelet detail [18]. Native adaptive inputs instead start Q_t from the simulator's refinement hierarchy, capped at level L, and then adjust it on the selected adaptation channels with the regridding procedure of Section 2.4 (heat release rate for AMReX, Appendix C.6). After either construction path, 2:1 balance is restored by refining any leaf whose face neighbor is more than one level finer. The uniform-patch tokenizer stores a fixed (C, p_h, p_w) raster per leaf and frame, obtained by exact block averaging of uniform-grid data or by sampling the finest native level covering each pixel.

Each training window shares one topology, built from its last input frame and used for all input and target frames, because f_θ predicts on the leaves of its input partition (Eq. (1)). Building it from the newest observed frame matches rollout, where regridding uses the newest predicted frame. For native adaptive inputs, this requirement rules out the simulator's own grids, which change between snapshots. Because the partition stays fixed while features move, refined regions are dilated by δ face-neighbor layers, pre-allocating fine leaves into which nearby features can move. Deterministic

training topologies can be cached per window and reused across epochs, with lazy value loading outside the topology cache (Appendix C.9). Appendix C gives the geometry, balancing procedure, and value-assignment rules.

Each token is associated with its leaf center (x, y) and refinement level ℓ. Our axial 3D RoPE assigns disjoint query/key dimensions to the three coordinates and applies an independent rotary embedding on each axis. The spatial scales convert normalized centers to finest-cell index units, matching the positional convention of the regular-grid ViT controls. We ablate the level-axis scale s_ℓ to evaluate how the strength of refinement-level encoding affects forecasting accuracy. Appendix E.1 gives the dimension partition, frequencies, scale formulas, and geometric interpretation.

2.2 Transformer Backbone

Each parallel-layout block [19] applies a single pre-LayerNorm and feeds the normalized tokens to an attention branch and a SwiGLU MLP branch. The branch outputs are concatenated, linearly projected, and added to the residual stream. Figure 2 illustrates this architecture. Default WAMRViT and ViT-finest/mid use the same block stack, holding the transformer backbone fixed across adaptive and regular-grid representations.

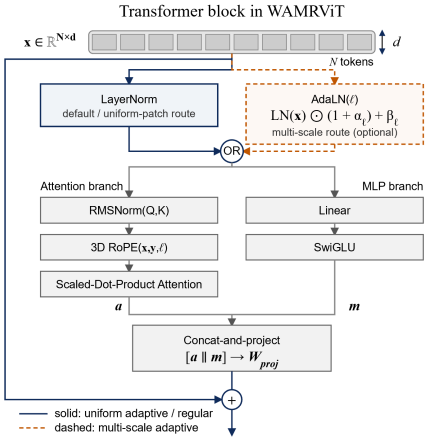


Figure 2: **Parallel-layout transformer block.** Attention and MLP branches are concatenated and projected into the residual stream.

For uniform-grid inputs, multi-scale WAMRViT retains each leaf's full source region rather than block-averaging it before the model. Coarse-region detail is reduced inside a learned encoder rather than discarded during data fill, while attention still processes one token per leaf. A level-ℓ leaf carries a raster of shape $(C, T_{\text{in}}, p_h r_\ell, p_w r_\ell)$, where $r_\ell = 2^{L-\ell}$. Its per-level encoder applies $L - \ell$ learned stride-2 reduction stages followed by a tubelet projection, producing one d-dimensional token regardless of the source raster size. Per-level decoders restore predictions at the retained resolution through corresponding expansion stages. Level-conditioned AdaLN replaces the per-block pre-norm and post-stack output norm, applying learned scale and shift parameters indexed by refinement level. The decoded leaves are reassembled on the finest uniform grid for the full-field objective below. These components define a separate architectural variant. Appendices D.2 and E.2 give the complete encoder, decoder, and normalization specifications.

2.3 Training Objectives

For uniform-patch WAMRViT, we train f_θ with the absolute L_2 norm of the prediction residual, averaged over leaves and channels,

$$\ell(\hat{y}, y) = \frac{1}{N_t C} \sum_{i=1}^{N_t} \sum_{c=1}^{C} \|\hat{y}_{i,c} - y_{i,c}\|_2 \,, \tag{2}$$

where i and c index leaves and channels, respectively, and the norm is taken over the $T_{\text{out}} \times p_h \times p_w$ spatiotemporal raster of each leaf. Leaf losses are weighted equally, regardless of the physical areas they represent.

Let $\hat{y}, y \in \mathbb{R}^{C \times T_{\text{out}} \times H \times W}$ denote full-field predictions and targets. Multi-scale WAMRViT uses a differentiable scatter onto the finest source grid. Regular-grid baselines use their respective output grids. The channel-mean objective takes each norm over the full spatiotemporal grid:

$$\ell_{\text{full}}(\hat{y}, y) = \frac{1}{C} \sum_{c=1}^{C} \|\hat{y}_c - y_c\|_2 \,. \tag{3}$$

Push-forward training unrolls K model applications and averages their single-step losses, $\mathcal{L}(\theta) = K^{-1} \sum_{k=1}^{K} \ell(\hat{y}^{(k)}, y^{(k)})$ [20]. The full-field variants substitute ℓ_{full} for ℓ. Appendix F gives the numerical stabilizer and complete push-forward recursion.

2.4 Autoregressive Rollout and In-Cell-Space Regridding

Every R predicted steps, regridding updates the topology from the newest predicted frame and transfers the entire input window onto it. The model calls in between keep the topology fixed. Because the source grid is unavailable during rollout, the refinement tests operate on leaf rasters. Uniform-patch regridding has three phases. (1) Refine leaves whose within-cell variation or face-boundary jump exceeds the threshold, bilinearly populate their children, and restore 2:1 balance. The jump test detects features between internally smooth leaves, which within-cell variation alone would miss. (2) Protect active leaves and a buffer of δ face-neighbor layers from immediate coarsening. This buffer keeps moving features on refined leaves, so the topology can stay fixed for R steps instead of being regridded after every prediction. (3) Coarsen four unprotected siblings into their parent by exact 2×2 block averaging, but only when their variation is below the hysteresis threshold and a look-ahead confirms that merging preserves 2:1 balance. Multi-scale native storage instead partitions and concatenates the retained source rasters exactly (Appendix D.2). Stopping conditions and complete pseudocode are given in Appendix C.10. The array backend parallelizes topology updates over packed leaf metadata using CPU kernels. For PLI and TRL, the small input windows are transferred to the CPU for regridding. For AMReX, the larger input windows remain on the GPU. Only the adaptation channels of the newest frame are transferred to the CPU for topology decisions, and compact split/merge records are returned to the GPU for state updates.

3 Experiments

3.1 Problems

We evaluate uniform-grid fields with two contrasting spatial distributions and a simulator-native AMR hierarchy. Appendix A provides a visual overview of all three problems.

PLI multimaterial shock propagation. The PLI subset of the HEAT benchmark [21] contains cylindrically symmetric shocks moving through perturbed material interfaces and ambient air. Its 10-channel observations comprise 1000 trajectories of 101 snapshots on a 1120×400 uniform grid. Large quiescent regions surround localized shocks and interfaces, making PLI the clearest test of whether adaptive tokenization can preserve refined-region accuracy without paying for finest patches everywhere.

TRL radiative mixing. TRL is from The Well [22] with 90 trajectories of 101 snapshots, 4 channels, and a 128×384 uniform grid. Its cold–hot interface rolls into dense filaments under shear, leaving substantially less quiescent area than PLI. TRL therefore tests adaptivity where a regular grid is less obviously wasteful and where the available training set is smaller.

AMReX combustion. The native-AMR case follows high-fidelity H_2/air detonation simulations with detailed finite-rate chemistry [23]. Its 17-channel, 396-snapshot trajectory occupies 264 GB and contains six coexisting refinement levels concentrated around the moving shock and reaction zone. To fit training memory, WAMRViT uses the five coarsest ($L = 4$, finest grid 4096×24576). A dense finest-grid representation is not stored. WAMRViT instead builds its quadtree from each reference snapshot's AMReX hierarchy and samples leaf values directly from the AMReX data. Shock discontinuities, stiff chemistry, and solver-sensitive soot-foil morphology make forecasting this hierarchy a demanding test of adaptive surrogates. Full channel lists and provenance are in Appendix B. Representative full-domain PLI and AMReX predictions are in Figures 5 and 7.

3.2 Baselines

ViT-finest uses the default WAMRViT parallel block stack with the finest patch size on a regular grid, providing a backbone-matched regular-grid baseline. ViT-mid uses the same stack with a coarser patch size chosen to approximately match WAMRViT's token count. Multi-scale WAMRViT is evaluated separately as a resolution-preserving architectural variant. On the native-AMR combustion problem, a finest-grid ViT would require approximately 98k tokens, while the validated multi-scale training objective additionally materializes the finest uniform grid to compute its full-field loss. These finest-grid approaches are therefore impractical under the current training protocol. The feasible regular-grid baselines instead project the AMR hierarchy onto a coarser grid, yet still use about 14% more tokens than uniform WAMRViT on average over the accuracy rollouts.

SwinV2 [24] provides an architecture-family control with conventional shifted-window blocks and a refined multiscale decoder. We approximately match its transformer capacity to WAMRViT, rather than its total parameter count or computational cost. This comparison tests an alternative architecture family, not a complete PDE foundation-model system. All models share the data splits and per-dataset optimization protocol. Full architectural specifications are provided in Appendix F.

3.3 Training and Evaluation

Uniform-patch WAMRViT and the two ViT controls share a six-layer backbone with 12 attention heads of dimension $d_h = 192$ ($d = 2304$), held fixed across datasets. Models predict one frame from two input frames on PLI/AMReX. TRL uses four inputs, following The Well's reference protocol [22]. All variants use AdamW with a cosine learning-rate schedule, BF16 mixed precision, gradient clipping, EMA weight averaging, and L_2 prediction-error objectives, evaluated per leaf for uniform-patch WAMRViT and over the full field for the multi-scale variant and regular-grid baselines. We denote the push-forward training depth by K: training unrolls K model applications and averages their per-step losses, with $K = 1$ for PLI and TRL and $K = 4$ for AMReX. To accommodate variable token counts, adaptive runs use micro-batch-one training. Section 2.3 defines the per-leaf and full-field losses. Appendix F gives the full hyperparameters.

PLI uses 800 training and 200 test trajectories; TRL uses The Well's official train/test split [22]; AMReX uses the first 80% of its trajectory for training and the final 20% for chronological testing. We use no separate validation set. Per-dataset adaptation settings, including initial refinement and regrid tolerances and protection radii, are selected qualitatively to align refinement with the physically active regions while keeping the training and inference topology distributions consistent. Their values are reported in Appendix F.

Evaluation uses shared held-out windows: PLI is evaluated on sparsely sampled 20-step rollouts, and AMReX on four 20-step rollouts from the chronological test segment, with WAMRViT regridding every $R = 4$ prediction steps in both cases. TRL instead uses one-step prediction over all test windows because its snapshot interval makes longer autoregressive comparisons uninformative. No inference-time regridding is performed in this setting. We use "rollout step" to denote the evaluation horizon. Main accuracy comparisons use three training seeds. Table 1 reports first-step means and sample standard deviations. Full-field metrics compare predictions on the common finest uniform grid and are available for PLI and TRL. Cell-space metrics use regular predictions mapped onto paired adaptive leaves by block averaging or nearest-neighbor upsampling. "Finest-level" means $\ell = L$. Because regridding changes that leaf set, later finest-level values describe the currently refined population rather than a fixed set of pixels. Within either scope, RMSE and variance-normalized RMSE (VRMSE) are computed per channel and then averaged over channels. VRMSE normalizes with the ground-truth spatial variance of the full frame or individual leaf, respectively. Metric formulas are given in Appendix F.3.

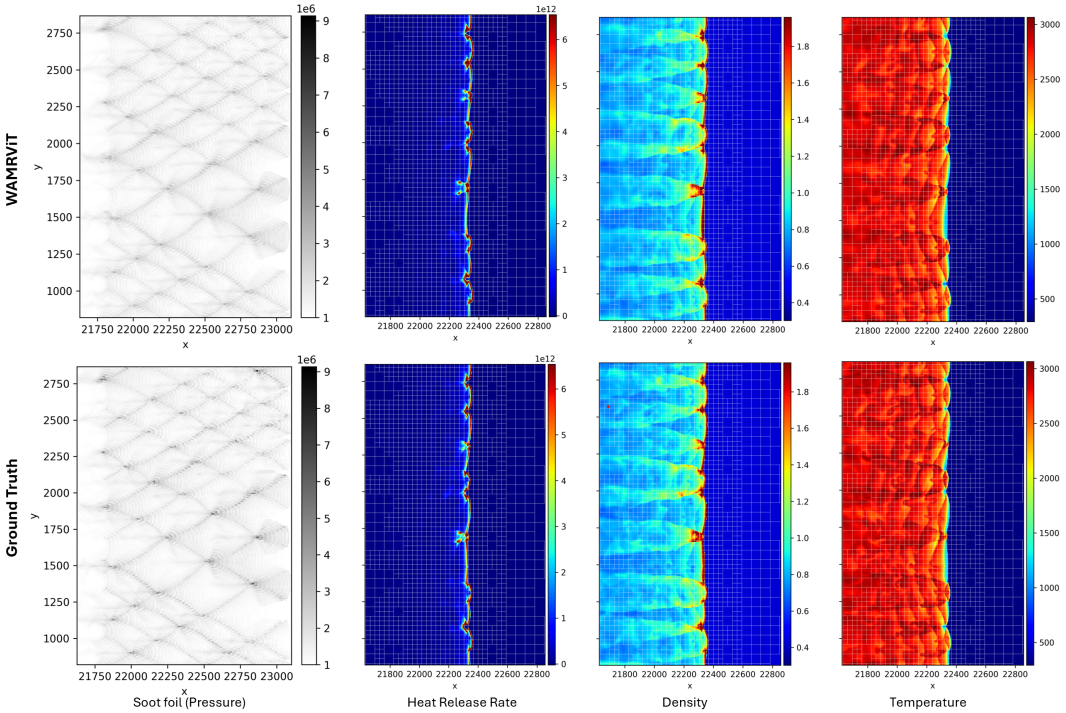
3.4 Results

PLI. At the first rollout step (Table 1), multi-scale WAMRViT has the lowest VRMSE in both scopes, 27.7% below ViT-finest over the full field. Uniform WAMRViT also outperforms ViT-finest in the finest-level region but trails it over the full field. This is consistent with its tokenizer block-averaging coarse leaves before the model, which discards sub-patch detail that full-field metrics still measure. Multi-scale WAMRViT retains that detail, reduces it in a learned encoder, and is trained with the full-field loss, at essentially the same token count as uniform WAMRViT, about 37% of ViT-finest's. These advantages narrow but persist over the rollout. Averaged over all 20 steps and seeds, uniform and multi-scale WAMRViT reduce finest-level VRMSE by 14.2% and 20.1% relative to ViT-finest evaluated on each variant's own finest-level region, and multi-scale WAMRViT reduces full-field VRMSE by 9.6%. RMSE counterparts and complete curves are in Appendix G.

TRL. TRL is a one-step test with matched initial tolerances for the two adaptive variants and no inference-time regrid. Both variants use approximately half as many tokens as ViT-finest. Over the full field, both are within about 1% of ViT-finest in VRMSE, with multi-scale WAMRViT slightly lower and uniform WAMRViT slightly higher. At the finest level, uniform WAMRViT reduces VRMSE by 45.3% relative to ViT-finest, whereas multi-scale WAMRViT is 14.1% higher. RMSE instead slightly favors multi-scale WAMRViT over ViT-finest in both scopes (Table 7), illustrating sensitivity to local variance normalization. The published IAP [13] results in Table 1 remain external



(a) Full-domain temperature prediction.



(b) Zoomed qualitative examples.

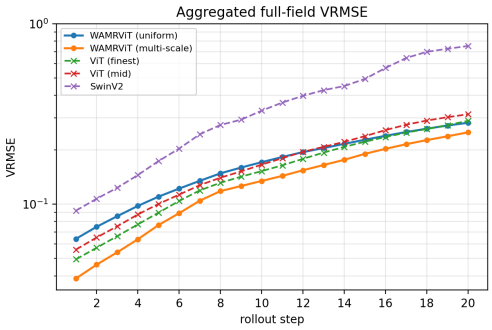
Figure 3: **AMReX combustion prediction on the chronological holdout.** (a) Full-domain WAMRViT prediction at rollout step 25. The white box marks the spatial extent of the zoomed field examples. (b) Predictions (top row) and ground truth (bottom row): peak-pressure soot foils over 50 rollout steps (left), and heat-release rate, density, and temperature at step 25 (right). These longer qualitative diagnostics extend beyond the 20-step quantitative protocol. Ground-truth grids are shown for reference. Additional full-domain comparisons are shown in Appendix A. Animations are provided in the repository.

references with a different training protocol. TRL's spatially widespread mixing leaves less quiescent area than PLI, reducing the opportunity to concentrate tokens on a small active region. Together with its coarse temporal sampling, this may contribute to multi-scale WAMRViT's smaller and metric-dependent gains over ViT-finest.

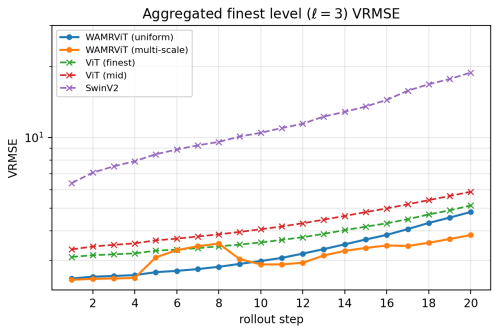
AMReX combustion. Only cell-space comparisons are feasible at the finest tokenized level ($\ell = 4$). WAMRViT reduces first-step finest-level RMSE by 60.8% relative to ViT-mid, with a similar margin over SwinV2. Averaged over seeds and all 20 rollout steps, the reduction remains above 50% against both. The per-leaf VRMSE aggregate is dominated by HRR, so we also report RMSE and the per-field breakdown in Appendix G.9. Topology updates also change the evaluated leaf population, so fluctuations reflect both forecast evolution and changes in the set of refined cells. Figure 3 and Appendix A show the dynamic grid following the reaction front and reproducing the soot-foil morphology. The full-domain views nevertheless reveal field deviations in coarse post-shock regions (Figure 3a and Appendix Figure 7). Uniform-patch WAMRViT weights leaf losses equally rather than by physical area. In the quantitative AMReX rollouts, the coarsest level covers about 84% of the domain but accounts for only 6% of the tokens. This imbalance may place less training emphasis on

Table 1: **First-step VRMSE and rollout-mean token counts.** Accuracy is mean ± sample standard deviation over seeds $\{0, 1, 42\}$. Token counts $\bar{N}$ are averaged over evaluation windows, rollout steps (20 for PLI/AMReX, 1 for TRL), and seeds. SwinV2 reports first-stage tokens. Finest-level metrics use the regions refined to level L in the initial adaptive topology. IAP* [13] reports performance at the two token budgets comparable to ours, 1,020 and 2,040, respectively.

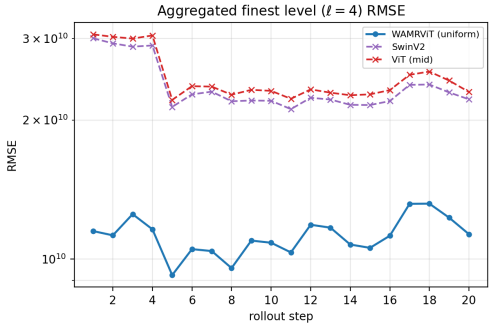
	WAMRViT (uniform)	WAMRViT (multi-scale)	ViT (finest)	ViT (mid)	SwinV2	IAP* [13]
PLI, full-field	0.0661 ±1.67E-3	**0.0389** ±2.52E-4	0.0539 ±6.44E-3	0.0553 ±6.68E-4	0.0920 ±2.32E-4	/
PLI, finest ($\ell = 3$)	2.6839 ±0.166	**2.5000** ±0.046	3.4125 ±0.305	3.3401 ±0.121	6.3348 ±0.054	/
PLI, tokens $\bar{N}$	3,327	3,339	8,960	3,500	8,960	/
TRL, full-field	0.1502 ±2.81E-4	**0.1476** ±5.40E-4	0.1490 ±6.61E-4	0.1629 ±4.87E-4	0.1955 ±6.46E-4	0.1577 / 0.1478
TRL, finest ($\ell = 2$)	**9.0411** ±0.665	18.8458 ±0.699	16.5135 ±0.351	22.1965 ±0.334	69.8422 ±0.503	/
TRL, tokens $\bar{N}$	1,597	1,597	3,072	1,536	3,072	1,020 / 2,040
AMReX, finest ($\ell = 4$)	**1.581E10** ±4.87E8	/	/	6.306E12 ±2.53E11	7.779E12 ±2.76E11	/
AMReX, tokens $\bar{N}$	5,399	/	/	6,144	6,144	/



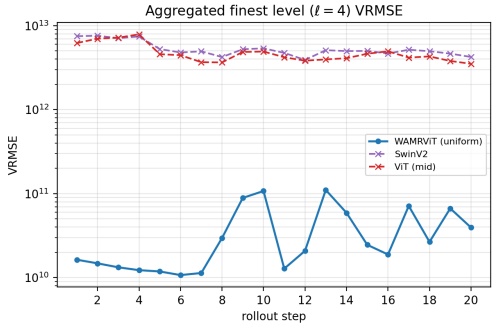
(a) PLI full-field VRMSE.



(b) PLI finest-level VRMSE.



(c) AMReX finest-level RMSE.



(d) AMReX finest-level VRMSE.

Figure 4: **Aggregated rollout error.** PLI full-field and finest-level VRMSE for $K = 1$ models (top), and AMReX finest-level RMSE and VRMSE for $K = 4$ models (bottom). WAMRViT regrids every four rollout steps. At the PLI finest level, the single ViT-finest curve is evaluated on uniform WAMRViT's refined region. Multi-scale WAMRViT uses its own region. AMReX errors are dominated by heat-release rate (HRR). Per-field results are reported in Appendix G.9. Three-seed first-step statistics are given in Table 1.

broad coarse regions than on the finely resolved reaction front. These are forecasts of held-out future states, testing the surrogate on coupled shock and reaction dynamics.

Table 7 reports first-step RMSE from the same three-seed evaluations. Multi-scale WAMRViT has the lowest full-field RMSE on PLI and TRL, and the AMReX finest-level advantage holds under both

metrics. Finest-level rankings can differ between the two metrics because finest-level VRMSE divides each leaf's error by that leaf's ground-truth standard deviation before averaging (Appendix F.3). Nearly constant leaves thus carry large weight in VRMSE, whereas RMSE is dominated by the large absolute errors at high-contrast features. On PLI, for example, uniform WAMRViT's finest-level RMSE is comparable to ViT-finest's, while its VRMSE is lower.

End-to-end efficiency. Table 2 reports training duration, timing-cohort token counts, and inference cost, including complete array-based topology updates and value transfers, with AMReX leaf values kept on the GPU (Section 2.4). Uniform/multi-scale WAMRViT is $3.03\times$ /$2.57\times$ faster than ViT-finest on PLI and approximately $1.21\times$ faster on TRL. Regridding accounts for 5.4%–17.2% of adaptive inference time. On AMReX, it offsets the forward-pass advantage over ViT-mid, making total time 8.6% higher. SwinV2 is fastest and uses the least peak allocated memory across all three datasets. Appendix F.2 gives the timing protocol and regrid settings.

Table 2: **Per-model training and inference cost.** Inference uses ten matched starting windows, batch one, and 20 model calls on one H100, including four complete regrids for adaptive models. Forward and regrid are mean per-window sums. Peak is maximum PyTorch-allocated CUDA memory, and $\bar{N}$ averages tokens per call over the timing cohort. Train reports total minutes for the schedules in Appendix Table 4.

Dataset	Variant	Train (min)	$\bar{N}$	Forward (s)	Regrid (s)	Total (s)	Peak (GiB)
PLI	WAMRViT (uniform)	188	3601	1.454	0.126	1.580 ± 0.328	1.584
	WAMRViT (multi-scale)	333	3606	1.601	0.259	1.860 ± 0.314	1.802
	ViT-finest	236	8960	4.789	–	4.789 ± 0.008	1.979
	ViT-mid	128	3500	1.301	–	1.301 ± 0.003	1.523
	SwinV2	283	8960	0.469	–	0.469 ± 0.001	1.343
TRL	WAMRViT (uniform)	136	2111	0.774	0.159	0.933 ± 0.095	1.354
	WAMRViT (multi-scale)	223	2199	0.888	0.050	0.938 ± 0.102	1.384
	ViT-finest	155	3072	1.132	–	1.132 ± 0.002	1.385
	ViT-mid	129	1536	0.517	–	0.517 ± 0.003	1.252
	SwinV2	443	3072	0.453	–	0.453 ± 0.002	1.180
AMReX	WAMRViT (uniform)	1371	5303	3.162	0.658	3.820 ± 0.233	6.094
	ViT-mid	1805	6144	3.519	–	3.519 ± 0.006	5.281
	SwinV2	2086	6144	0.508	–	0.508 ± 0.002	4.682

3.5 Ablations

Table 3: **Level-axis RoPE ablation: first-step channel-mean VRMSE.** Uniform-patch WAMRViT on all three datasets; multi-scale WAMRViT on PLI. Training uses $K = 1$ for PLI/TRL and $K = 4$ for AMReX. Further results are in Appendix G.5.

	PLI uniform		PLI multi-scale		TRL uniform		AMReX uniform
s_ℓ	full-field	finest ($\ell=3$)	full-field	finest ($\ell=3$)	full-field	finest ($\ell=2$)	finest ($\ell=4$)
1 (default)	0.0642	<u>2.5107</u>	**0.03868**	**2.4804**	0.1515	9.7444	<u>1.631E10</u>
2	**0.0630**	2.5250	0.03889	2.7372	<u>0.1505</u>	**8.9478**	1.805E10
4	<u>0.0632</u>	**2.5042**	<u>0.03874</u>	<u>2.5213</u>	0.1506	9.8886	**1.544E10**
8	0.0666	2.5394	0.03967	2.6803	**0.1502**	<u>9.1519</u>	1.660E10

Push-forward depth. On PLI, WAMRViT and ViT-finest are each trained at $K \in \{1, 2, 4\}$ to test the effect of K on rollout performance. At rollout step 20, WAMRViT's finest-level VRMSE is below ViT-finest's at both $K = 1$ and $K = 4$, with the margin growing to 23.2% at $K = 4$. The intermediate $K = 2$ adaptive run is unstable at this scope. At full field with $K = 4$, WAMRViT is lower than ViT-finest at step 20 but higher when averaged over the rollout. The effect of push-forward depth depends on the evaluation scope and horizon. At step 20, WAMRViT's finest-level advantage holds for $K = 1$ and $K = 4$, but reverses at $K = 2$. Complete curves and values are in Appendix G.3.

Topology resampling. We compare uniform WAMRViT trained with and without random coarsening/refinement on PLI and AMReX. On PLI, resampling degrades both scopes: averaged over the 20-step rollout, VRMSE rises by 18.5% at full field and 32.2% at the finest level. On AMReX, over the same horizon, it lowers finest-level VRMSE by 8.8% but raises coarsest-level VRMSE by 15.7%. Its effect is therefore dataset-dependent, and all Table 1 models train without it (Appendix G.4).

Level-axis scale. Table 3 compares $s_\ell \in \{1, 2, 4, 8\}$ for uniform-patch and multi-scale WAMRViT on PLI and uniform-patch WAMRViT on TRL and AMReX. For uniform-patch models, the preferred scale varies with dataset, scope, metric, and horizon; neither $s_\ell = 2$ (perimeter ratio) nor $s_\ell = 4$ (area ratio) is consistently best. On AMReX, $s_\ell = 4$ leads most VRMSE and $s_\ell = 8$ most RMSE comparisons; the default leads none. Multi-scale WAMRViT on PLI favors $s_\ell = 1$ in first-step full-field and finest-level VRMSE and rollout-mean full-field RMSE and VRMSE. We retain $s_\ell = 1$ in main experiments (details in Appendix G.5).

Mesh-policy sensitivity. With fixed PLI checkpoints, we vary initial and runtime mesh settings (Appendix G.6). Both variants are relatively robust to the tested tolerance rescalings, with rollout-mean full-field errors changing by at most about 3.3%. Neighborhood changes have larger effects, but the multi-scale advantage over uniform-patch WAMRViT persists across all tested policies.

4 Limitations

Variable leaf counts currently require micro-batch-one adaptive training, while inference regridding adds topology updates and value transfers between model calls. AMReX training is limited by data-pipeline materialization rather than accelerator compute, as each leaf carries a 17-channel 32×32 raster per input frame. Regridding accounts for 5.4–17.2% of adaptive inference time. On AMReX, WAMRViT remains 8.6% slower end to end than ViT-mid under the batch-one protocol, and SwinV2 is faster on every dataset. A dense finest-grid AMReX baseline and multi-scale full-field training are infeasible under the current materialization protocol, and training memory limits the AMReX quadtree to five of the six native levels. Padding, packed attention, and GPU-resident topology operations offer potential routes to higher throughput (Appendix I).

Adaptive regridding introduces deployment hyperparameters, including refinement tolerance, protection radius, and regrid interval. Rollout topologies may differ from those seen during training but should remain sufficiently similar. The current refinement criterion is non-learned and has not been compared with learned alternatives. Uniform-patch WAMRViT weights leaf losses equally, potentially underweighting broad coarse regions. The AMReX full-domain views show deviations in coarse post-shock regions. On TRL, multi-scale WAMRViT has lower full-field RMSE and VRMSE than ViT-finest but higher finest-level VRMSE. All experiments are two-dimensional, leaving generalization to three dimensions and broader physics untested. The AMReX study uses a single trajectory with a chronological holdout. Multi-trajectory training and evaluation remain future work.

5 Conclusion

We introduced WAMRViT, an autoregressive PDE surrogate that combines balanced quadtree tokenization, level-aware 3D RoPE, and in-cell-space regridding to adapt its hierarchy as predicted features evolve. Its uniform-patch variant builds its quadtree from simulator-native AMR hierarchies, while its multi-scale variant preserves source-grid resolution through learned level-specific encoding and decoding. On PLI, uniform and multi-scale WAMRViT reduce 20-step rollout-mean finest-level VRMSE by 14.2% and 20.1%, respectively, relative to the same ViT-finest baseline evaluated on the finest-level leaves of each variant's own regridded topology, while using approximately 37% as many tokens. The multi-scale variant also reduces full-field VRMSE by 9.6%. On one-step TRL, the comparison depends on scope and metric: uniform WAMRViT reduces finest-level VRMSE by 45.3% relative to ViT-finest, whereas the multi-scale variant is 14.1% higher there. On native AMReX combustion data, WAMRViT reduces first-step finest-level RMSE by 60.8% relative to the feasible ViT-mid projection, with a similar margin over SwinV2. Both variants achieve faster end-to-end inference than ViT-finest on the uniform-grid benchmarks, whereas on AMReX WAMRViT is 8.6% slower end-to-end than ViT-mid. Together, these results demonstrate the potential of dynamically evolving token hierarchies to concentrate resolution around moving features while supporting autoregressive forecasting on simulator-native AMR data.

Acknowledgments and Disclosure of Funding

This work was supported by Los Alamos National Laboratory through the project "Algorithm/Software/Hardware Co-design for High Energy Density Applications" at the University of Michigan, and by the Center for Prediction, Reasoning, and Intelligence for Multiphysics Exploration (C-PRIME), a PSAAP-IV project funded by the U.S. Department of Energy under Grant No. DE-NA0004264.

References

[1] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[2] 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.

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

[4] Benedikt Alkin, Andreas Fürst, Simon Schmid, Lukas Gruber, Markus Holzleitner, and Johannes Brandstetter. Universal physics transformers: A framework for efficiently scaling neural operators. In *Advances in Neural Information Processing Systems*, volume 37, pages 25152–25194, 2024.

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

[6] Duy-Kien Nguyen, Mahmoud Assran, Unnat Jain, Martin R. Oswald, Cees G. M. Snoek, and Xinlei Chen. An image is worth more than 16x16 patches: Exploring transformers on individual pixels. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=tjNf0L8QjR.

[7] Marsha J. Berger and Joseph Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of Computational Physics*, 53(3):484–512, 1984.

[8] Marsha J. Berger and Phillip Colella. Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics*, 82(1):64–84, 1989.

[9] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5804–5813, 2025.

[10] Enzhi Zhang, Isaac Lyngaas, Peng Chen, Xiao Wang, Jun Igarashi, Yuankai Huo, Masaharu Munetomo, and Mohamed Wahib. Adaptive patching for high-resolution image segmentation with transformers. In *SC24: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16, 2024. doi: 10.1109/SC41406.2024.00082.

[11] Rohan Choudhury, JungEun Kim, Jinhyung Park, Eunho Yang, László A. Jeni, and Kris M. Kitani. Faster vision transformers with adaptive patches. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=SzoowJtd14.

[12] Pei Zhang, M. Paul Laiu, Matthew Norman, Doug Stefanski, and John Gounley. MATEY: Multiscale adaptive transformer models for spatiotemporal physical systems. *Machine Learning: Science and Technology*, 7(3):035016, 2026.

[13] Hanwen Wang and Paris Perdikaris. Adaptive tokenization for vision transformer PDE simulation. In *AI&PDE: ICLR 2026 Workshop on AI and Partial Differential Equations*, 2026. URL https://openreview.net/forum?id=rjsBK0bDx2.

[14] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[15] Payel Mukhopadhyay, Michael McCabe, Ruben Ohana, and Miles Cranmer. Overtone: Cyclic patch modulation for clean, efficient, and flexible physics emulators. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=itUo64aUeK.

[16] Ivan Zanardi, Simone Venturi, and Marco Panesi. Adaptive physics-informed neural operator for coarse-grained non-equilibrium flows. *Scientific Reports*, 13(1):15497, 2023. doi: 10.1038/s41598-023-41039-y.

[17] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. ViViT: A video vision transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6836–6846, 2021.

[18] Alfred Haar. *Zur Theorie der orthogonalen Funktionensysteme*. PhD thesis, Georg-August-Universität Göttingen, 1909.

[19] Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, et al. Scaling vision transformers to 22 billion parameters. In *International Conference on Machine Learning*, pages 7480–7512, 2023.

[20] Johannes Brandstetter, Daniel E. Worrall, and Max Welling. Message passing neural PDE solvers. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=vSix3HPYKSU.

[21] Divya Banesh, Sharmistha Chakrabarti, Soumi De, Gal Egozi, Kyle Hickmann, William S. Hlavacek, Bryan Kaiser, Sourabh Pandit, Jesus Pulido, David Schodt, et al. The High Explosives & Affected Targets (HEAT) dataset, July 2025. URL https://www.osti.gov/biblio/2571471.

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

[23] Michael Ullman, Ral Bielawski, and Venkat Raman. Thermodynamic trajectories in detonations with stratified reactant mixtures. *Combustion and Flame*, 277:114173, 2025.

[24] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, et al. Swin Transformer V2: Scaling up capacity and resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12009–12019, 2022.

[25] Drummond B. Fielding, Eve C. Ostriker, Greg L. Bryan, and Adam S. Jermyn. Multiphase gas and the fractal nature of radiative turbulent mixing layers. *The Astrophysical Journal Letters*, 894(2):L24, 2020.

[26] Mark A. Mueller, Tae J. Kim, Richard A. Yetter, and Frederick L. Dryer. Flow reactor studies and kinetic modeling of the H_2/O_2 reaction. *International Journal of Chemical Kinetics*, 31(2): 113–125, 1999.

[27] Weiqun Zhang, Ann Almgren, Vince Beckner, John Bell, Johannes Blaschke, Cy Chan, Marcus Day, Brian Friesen, Kevin Gott, Daniel Graves, Max P. Katz, Andrew Myers, Tan Nguyen, Andrew Nonaka, Michele Rosso, Samuel Williams, and Michael Zingale. AMReX: A framework for block-structured adaptive mesh refinement. *Journal of Open Source Software*, 4(37):1370, 2019. doi: 10.21105/joss.01370.

[28] Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. Numba: A LLVM-based Python JIT compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, pages 1–6, 2015.

[29] Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I. Jordan, et al. Ray: A distributed framework for emerging AI applications. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 561–577, 2018.

[30] Sibo Cheng, Yilin Zhuang, Lyes Kahouadji, Che Liu, Jianhua Chen, Omar K. Matar, and Rossella Arcucci. Multi-domain encoder–decoder neural networks for latent data assimilation in dynamical systems. *Computer Methods in Applied Mechanics and Engineering*, 430:117201, 2024.

[31] Byeongho Heo, Song Park, Dongyoon Han, and Sangdoo Yun. Rotary position embedding for vision transformer. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 289–305, 2024.

[32] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4195–4205, 2023.

[33] 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, pages 72525–72624, 2024.

[34] Jean Kossaifi, Nikola Kovachki, Zongyi Li, David Pitt, Miguel Liu-Schiaffini, Robert Joseph George, Boris Bonev, Kamyar Azizzadenesheli, Julius Berner, Valentin Duruisseaux, and Anima Anandkumar. A library for learning neural operators. *arXiv preprint arXiv:2412.10354*, 2025.

[35] Justin Sirignano and Konstantinos Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, 2018.

[36] Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. DeepXDE: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, 2021.

[37] Sifan Wang, Jacob H. Seidman, Shyam Sankaran, Hanwen Wang, George J. Pappas, and Paris Perdikaris. CViT: Continuous vision transformer for operator learning. In *International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=cRnCcuLvyr`.

[38] Weihang Ouyang, Yeonjong Shin, Si-Wei Liu, and Lu Lu. NOEM: Efficient and scalable finite element method enabled by reusable neural operators. *Nature Computational Science*, 6(4): 417–429, 2026.

[39] Minglang Yin, Enrui Zhang, Yue Yu, and George Em Karniadakis. Interfacing finite elements with deep neural operators for fast multiscale modeling of mechanics problems. *Computer Methods in Applied Mechanics and Engineering*, 402:115027, 2022.

[40] John Guibas, Morteza Mardani, Zongyi Li, Andrew Tao, Anima Anandkumar, and Bryan Catanzaro. Efficient token mixing for transformers via adaptive Fourier neural operators. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=EXHG-A3jlM`.

A Qualitative overview of the forecasting problems

The following full-domain snapshots introduce the spatial structures that motivate adaptive forecasting before the technical appendices. PLI combines moving shocks and material interfaces with large quiescent regions; TRL develops filamentary mixing under shear and radiative cooling; AMReX resolves coupled shocks and reaction zones on a native refinement hierarchy. Each figure shows rollout step 20 on held-out data. Quantitative comparisons are reported in Tables 1 and 7.

A.1 PLI multimaterial shock propagation

Figure 5 compares density at rollout step 20 on trajectory index 950 (starting frame 20). All five models retain the large-scale material interface, while differences remain around the curved front and lower casing.

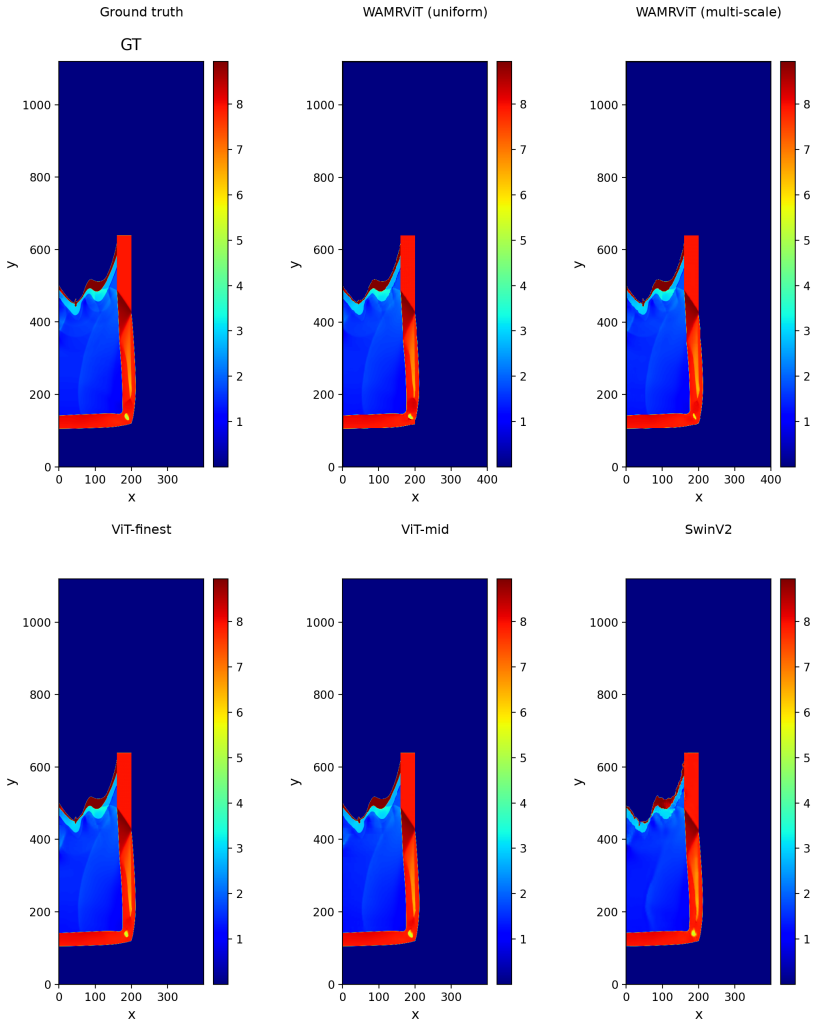


Figure 5: **Qualitative rollout snapshot on PLI.** Density at rollout step 20, (trajectory index 950, in test set), starting frame 20.

A.2 TRL radiative mixing

Figure 6 compares density predictions at rollout step 20 on test trajectory 0 (starting frame 20). This is a qualitative stress test beyond the one-step quantitative TRL protocol. The five learned models show different interface positions and small-scale structures at this horizon. ViT-mid and ViT-finest produce visibly different states despite sharing the same backbone, illustrating sensitivity to patch granularity.

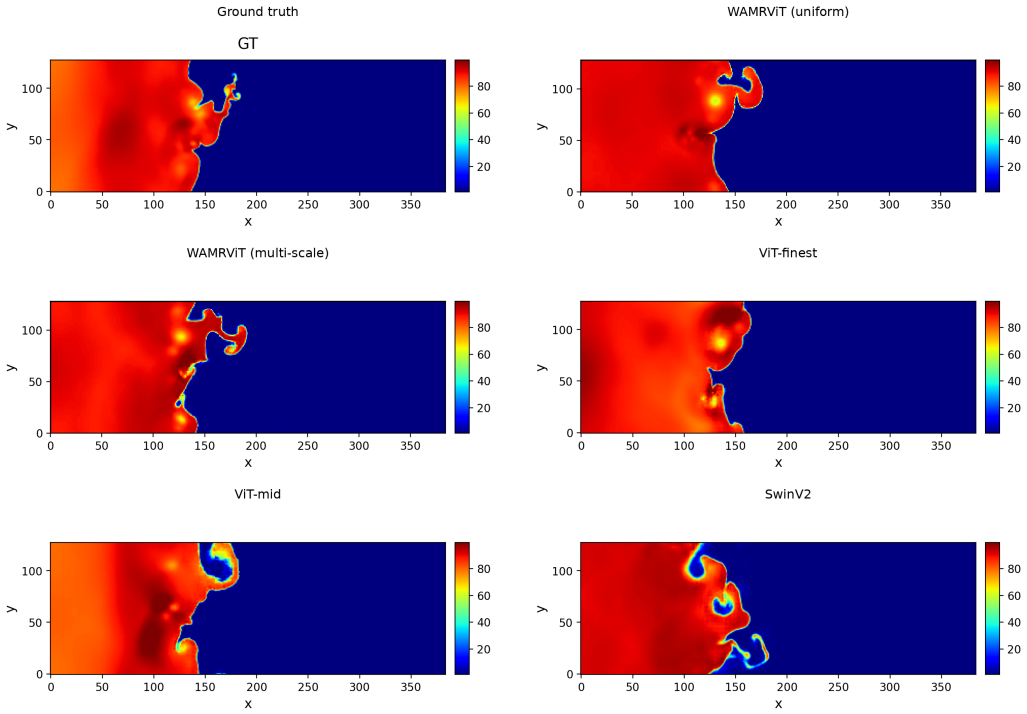


Figure 6: **Qualitative rollout snapshot on TRL.** Density at rollout step 20 on trajectory 0 (starting frame 20). Both adaptive variants use initial and regrid tolerance 0.1 per channel.

A.3 AMReX combustion

Figure 7 compares predictions at rollout step 20 starting from frame 330, within the chronological test region. Unlike the zoomed view in Figure 3, these panels render the full domain at capped resolution because each AMReX snapshot averages about 0.67 GB. All three models track the leading combustion front at this scale. Uniform WAMRViT shows residual post-shock artifacts in coarse regions. Its quantitative benefit is concentrated on the dynamically refined leaves and is therefore clearer in the finest-level metrics and zoomed view.

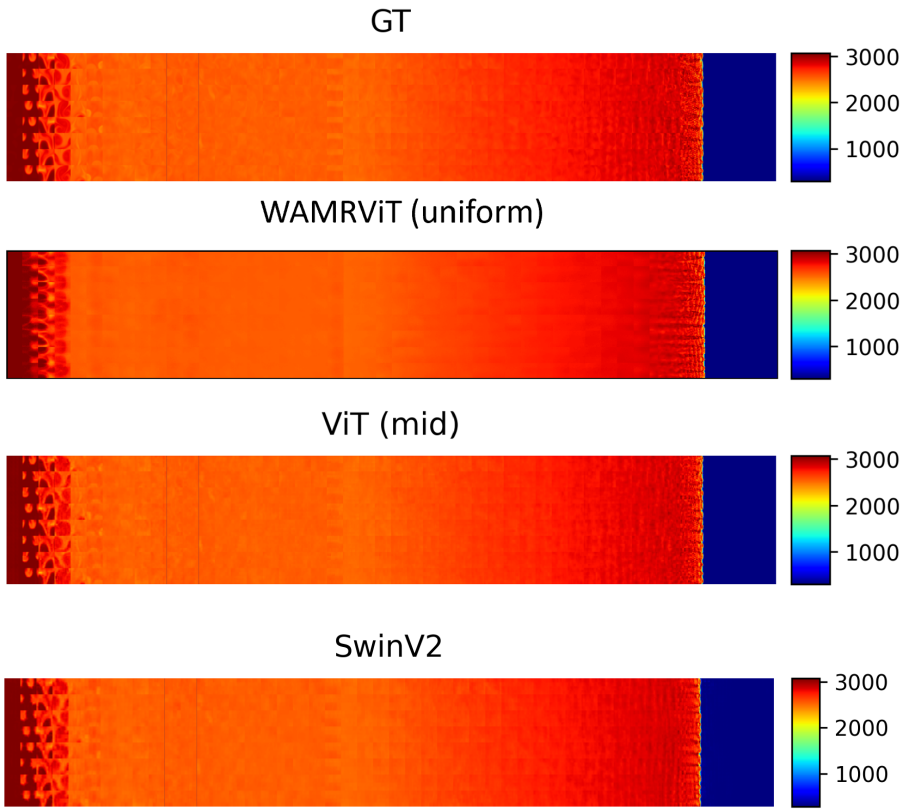


Figure 7: **Qualitative rollout snapshot on AMReX combustion, full domain.** Temperature at rollout step 20 in the chronological test region, starting from frame 330. Top to bottom: ground truth, WAMRViT (uniform), ViT-mid, and SwinV2. Figure 3 shows a zoomed view of the leading shock region.

B Datasets

This appendix supplements Section 3.1 with full channel lists and simulation provenance.

Perturbed-Layer-Interface (PLI) multimaterial shock propagation. The PLI subset of the HEAT (High Explosives & Affected Targets) benchmark [21] consists of cylindrically symmetric (R, Z) simulations in which a high-explosive main charge detonates at $t = 0$ and drives a shock through six fixed material slots: the high explosive, a stainless-steel case, a polymer cushion, an aluminum striker, a copper throw plate, and surrounding air. The per-trajectory degrees of freedom are six spline coefficients defining four perturbed interface-shape splines (cushion-striker anchor sa, striker thickness st, cushion thickness ct, and throw thickness tt in the dataset notation). We use the first 1000 released trajectories, each stored at fp16 with 101 snapshots on a uniform 1120×400 Eulerian grid. From the solver fields, our 10-channel observation retains velocity components (U, W), cell-averaged density ρ, the sum of the six material-energy channels, $E_{\text{sum}} = \sum_m E_m$, and six volume-of-fluid indicators ϕ_{case}, ϕ_{cushion}, $\phi_{\text{maincharge}}$, $\phi_{\text{outside_air}}$, ϕ_{striker}, and ϕ_{throw}. The first 800 trajectories are used for training and the remaining 200 for evaluation.

Turbulent radiative layer 2D (TRL). TRL is drawn from The Well benchmark suite [22], which packages the simulations of Fielding et al. [25]. It contains two-dimensional simulations of a radiatively cooling shear layer in which a cool, dense slab is sheared against a hot, diffuse medium while radiative cooling drives rapid interface transitions. Trajectories vary in the ratio of cooling time to shear time, which controls the filamentary mixing structure. Each snapshot occupies a uniform 128×384 grid. The four observation channels are density ρ, pressure p, and velocity components (u_x, u_y). We adapt on all four channels with a common tolerance because each resolves the same interface structure. We use The Well's official train/test split (72 and 9 trajectories; its 9 validation

trajectories are unused) and follow its reference forecasting protocol, predicting one future frame from four input frames ($4 \to 1$). One-step accuracy is evaluated over all 873 windows from nine held-out trajectories.

AMReX combustion: stratified-detonation turbulent reacting layer. The trajectory is generated from a direct numerical simulation of an H_2/air detonation [23], solving the compressible reactive Navier–Stokes equations with detailed chemistry (9 species and 21 reactions) following Mueller et al. [26]. Block-structured AMR resolves shocks and reaction zones at micrometer scales while maintaining tractable simulation cost. The native hierarchy contains six coexisting levels, indexed 0–5, with refinement dynamically tracking the detonation front. To fit training memory, WAMRViT uses levels 0–4, whose finest grid is 4096×24576 (Appendix C.6).

The 17-channel state comprises heat release rate (HRR), temperature T, Mach number, pressure, density ρ, total energy density ρE, momentum components $(\rho u_x, \rho u_y)$, and nine density-weighted species mass fractions ρY_X for $X \in \{\mathrm{H}, \mathrm{H_2}, \mathrm{H_2O}, \mathrm{H_2O_2}, \mathrm{HO_2}, \mathrm{N_2}, \mathrm{O}, \mathrm{O_2}, \mathrm{OH}\}$. A dense finest-grid representation is neither stored nor practical because the finest level occupies only a localized fraction of the domain. Uniform controls therefore use a projection onto the AMR level-2 grid (1024×6144), whereas WAMRViT samples each adaptive leaf from the finest AMReX level covering it (Appendix C.6).

C Quadtree Construction and Value Assignment

This appendix describes the quadtree data structure, the refinement and coarsening primitives, and the procedures used to populate leaf cells with field values from both uniform-grid arrays and natively adaptive AMR data. Unless stated otherwise, the spatial value formulas describe one frame in the default uniform-patch path. All frames in a retained input window share one topology, and spatial value transfers are applied independently to each frame. The native-storage split/stitch operations are given in Appendix D.2.

C.1 Domain tiling and cell geometry

The physical domain $\Omega = [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$ is partitioned into a regular grid of *root tiles*, each of physical size (τ_w, τ_h). The numbers of tiles in each direction are

$$n_x = \frac{W}{\tau_w}, \qquad n_y = \frac{H}{\tau_h},$$

where $W = x_{\max} - x_{\min}$ and $H = y_{\max} - y_{\min}$, and both ratios must be integers. Each root tile is a leaf cell at level $\ell = 0$ with center (c_x, c_y) and half-extents $(h_x, h_y) = (\tau_w/2, \tau_h/2)$.

A cell at level ℓ can be subdivided into four children at level $\ell + 1$, up to maximum depth L. The four children are indexed by quadrant $q \in \{\mathrm{SW}, \mathrm{SE}, \mathrm{NW}, \mathrm{NE}\}$ with centers and half-extents

$$c_x^{(q)} = c_x \pm \tfrac{h_x}{2}, \qquad c_y^{(q)} = c_y \pm \tfrac{h_y}{2}, \tag{4}$$

$$h_x^{(q)} = \tfrac{h_x}{2}, \qquad h_y^{(q)} = \tfrac{h_y}{2}, \tag{5}$$

where the sign in (4) is determined by the quadrant (e.g. SW $\to (-,-)$, NE $\to (+,+)$). Throughout this appendix we use a y-axis-up convention, so smaller c_y corresponds to the southern children and array storage is indexed accordingly. Each leaf stores a raster $v \in \mathbb{R}^{C \times p_h \times p_w}$ at a fixed patch shape (p_h, p_w) shared across all levels.

C.2 Refinement (bilinear upsampling)

When a leaf at level ℓ with value $v \in \mathbb{R}^{C \times p_h \times p_w}$ is refined under uniform-patch storage, its patch is split at the integer midpoints $h_m = \lfloor p_h/2 \rfloor$ and $w_m = \lfloor p_w/2 \rfloor$:

$$v_{\mathrm{SW}} = v[:,\, 0{:}h_m,\; 0{:}w_m], \tag{6}$$

$$v_{\mathrm{SE}} = v[:,\, 0{:}h_m,\; w_m{:}p_w], \tag{7}$$

$$v_{\mathrm{NW}} = v[:,\, h_m{:}p_h,\; 0{:}w_m], \tag{8}$$

$$v_{\mathrm{NE}} = v[:,\, h_m{:}p_h,\; w_m{:}p_w]. \tag{9}$$

The source quadrants may differ by one pixel when a patch dimension is odd. Each quadrant is independently bilinearly resized to the common child storage shape (C, p_h, p_w), while the four children retain equal physical extents. Align-corners interpolation aligns the corner pixels of the output with those of the source quadrant. For each channel c, given a quadrant $v_q \in \mathbb{R}^{h_q \times w_q}$ with its actual height h_q and width w_q and target size (p_h, p_w), the upsampled value at output position (i, j) is

$$\hat{v}_q[i,j] = (1-w_y)(1-w_x)\, v_q[i_0, j_0] + (1-w_y) w_x\, v_q[i_0, j_1] + w_y (1-w_x)\, v_q[i_1, j_0] + w_y w_x\, v_q[i_1, j_1], \tag{10}$$

with (i_0, j_0), (i_1, j_1) the floor and ceil source indices and $w_y, w_x \in [0, 1]$ the fractional offsets. The four children are then assigned as leaves at level $\ell + 1$ with these upsampled patches.

Training-vs.-inference role. At training time, the bilinear values produced during refinement are immediately overwritten by the data-fill step (Appendix C.5), so the interpolation choice has no effect on the training objective. At inference, regridding (Section 2.4) carries the model's predictions through topology changes via this bilinear operation.

C.3 Coarsening (2×2 block-mean)

Coarsening merges four sibling leaves into their parent: given an internal node whose four children are all leaves with values $v_{\mathrm{SW}}, v_{\mathrm{SE}}, v_{\mathrm{NW}}, v_{\mathrm{NE}} \in \mathbb{R}^{C \times p_h \times p_w}$, we stitch them into a $(C, 2p_h, 2p_w)$ super-patch in spatial layout:

$$V_{\mathrm{stitched}} = \begin{bmatrix} v_{\mathrm{NW}} & v_{\mathrm{NE}} \\ v_{\mathrm{SW}} & v_{\mathrm{SE}} \end{bmatrix} \in \mathbb{R}^{C \times 2p_h \times 2p_w}. \tag{11}$$

The matrix is the spatial layout under our y-up convention. In array storage $V_{\mathrm{stitched}}[:,\, 0{:}p_h,\, :]$ holds the southern children $\{v_{\mathrm{SW}}, v_{\mathrm{SE}}\}$ and $V_{\mathrm{stitched}}[:,\, p_h{:}2p_h,\, :]$ the northern. The super-patch is then downsampled by exact 2×2 block averaging:

$$v_{\mathrm{parent}}[c, i, j] = \frac{1}{4} \sum_{di \in \{0,1\}} \sum_{dj \in \{0,1\}} V_{\mathrm{stitched}}[c,\, 2i + di,\, 2j + dj]. \tag{12}$$

The operation is mean-conservative: $\mathrm{mean}(v_{\mathrm{parent}}) = \mathrm{mean}(V_{\mathrm{stitched}})$ for each stored channel. The four children are removed and the parent becomes a leaf with value v_{parent}.

C.4 2:1 balance enforcement

After any topology modification (refinement or coarsening), we enforce the 2:1 balance constraint: for every leaf C_i at level ℓ_i and every face neighbor C_j, we require $|\ell_i - \ell_j| \leq 1$. The enforcement loop iterates until convergence:

1. For each leaf C_i, check all four face directions. If any neighbor C_j satisfies $\ell_j - \ell_i > 1$ and $\ell_i < L$, add C_i to a pending set.
2. Refine every cell in the pending set.
3. Repeat until the pending set is empty.

Because each iteration only refines cells (never coarsens), and the maximum depth L is finite, the loop terminates in at most L iterations.

C.5 Value assignment from uniform-grid arrays

Given a field $x \in \mathbb{R}^{C \times N_y \times N_x}$ defined on a regular pixel grid of N_y rows and N_x columns covering Ω, and a quadtree Q with a fixed topology, each leaf C_i at level ℓ_i is populated as follows.

The pixel spacing is $\Delta x = W/N_x$, $\Delta y = H/N_y$, where W, H are the physical extents introduced above. The leaf's physical extent maps to a rectangular region of the source array:

$$j_{\mathrm{start}} = \lfloor (c_x - h_x - x_{\min})/\Delta x \rfloor, \qquad j_{\mathrm{end}} = \lceil (c_x + h_x - x_{\min})/\Delta x \rceil, \tag{13}$$

$$i_{\mathrm{start}} = \lfloor (c_y - h_y - y_{\min})/\Delta y \rfloor, \qquad i_{\mathrm{end}} = \lceil (c_y + h_y - y_{\min})/\Delta y \rceil. \tag{14}$$

For this block-average assignment, the source grid satisfies $N_x = n_x p_w 2^L$ and $N_y = n_y p_h 2^L$, so leaf boundaries align with source-pixel boundaries. The extracted region $R_i = x[:, i_{\text{start}} : i_{\text{end}}, j_{\text{start}} : j_{\text{end}}]$ therefore has shape $(C, p_h r_{\ell_i}, p_w r_{\ell_i})$, where $r_{\ell_i} = 2^{L-\ell_i}$. Each stored raster pixel is obtained by averaging an $r_{\ell_i} \times r_{\ell_i}$ source block:

$$v[c, a, b] = \frac{1}{r_{\ell_i}^2} \sum_{da=0}^{r_{\ell_i}-1} \sum_{db=0}^{r_{\ell_i}-1} R_i[c,\, r_{\ell_i} a + da,\, r_{\ell_i} b + db]. \tag{15}$$

For the finest level ($\ell_i = L$), $r_L = 1$ and the assignment is a direct copy with no averaging. This operation involves no interpolation (every pixel in the source array contributes to exactly one leaf's block average), so the assignment is exact and conservative.

C.6 Value assignment from AMReX plotfiles

AMReX [27] simulations store data on a hierarchy of overlapping grids: level-0 grids cover the entire domain at coarse resolution, and successively finer levels cover subregions with refinement ratio r (typically $r = 2$). Where grids from different levels overlap, the finest available level takes precedence. The challenge for our quadtree tokenizer is to resolve this ownership and extract values at each leaf's resolution from the correct AMR level.

Topology construction. AMReX regrids between snapshots, so its grids cannot serve directly as the shared input/target partition of a training window. Instead, each window's quadtree is built from the AMR hierarchy of its reference frame (its last input frame):

1. **Coordinate mapping.** AMReX grids are stored in physical coordinates. We map all grid extents to an index-space $[0, n_x 2^L] \times [0, n_y 2^L]$ via a linear transformation, so that the finest-level cell has unit side length. The native hierarchy has six levels. We cap the quadtree at $L = 4$ to fit training memory, so native level-5 regions map to level-4 leaves whose pixels sample the level-5 data at level-4 spacing (Eq. (18)).
2. **Refinement predicate.** A leaf C_i at level ℓ_i is marked for refinement if any AMR grid at a level finer than ℓ_i has a non-trivial intersection with C_i's spatial extent:

$$C_i \text{ is refined} \iff \exists\, g : \ell_g > \ell_i \ \wedge\ (C_i \cap g) \neq \varnothing. \tag{16}$$

   This predicate is evaluated via a Numba-accelerated [28] scan over all AMR grids.
3. **Iterative refinement.** The predicate is applied across all leaves in a loop until no further refinements occur. The result is a quadtree whose leaf-level distribution mirrors the AMR refinement pattern.
4. **2:1 balance.** After refinement converges, the 2:1 balance restoration of Appendix C.4 is run until stable.
5. **Detector adjustment and dilation.** Using HRR values sampled from the reference frame by the pixel-level extraction rule below, we adjust the hierarchy-derived tree with Algorithm 2 at tolerance $\varepsilon = 0.01$, omitting the protection phase. We then dilate its refined regions by $\delta = 2$ face-neighbor layers, as for uniform-grid sources.

Pixel-level value extraction. Once the topology is fixed, each leaf's (C, p_h, p_w) raster is populated by a Numba-accelerated kernel. For each pixel (i_y, i_x) within a leaf's patch, the kernel computes the pixel's physical center

$$(p_x, p_y) = \big(x_{\min}^{(C_i)} + (i_x + \tfrac{1}{2})\,\Delta x_{\text{leaf}},\ \ y_{\min}^{(C_i)} + (i_y + \tfrac{1}{2})\,\Delta y_{\text{leaf}}\big), \tag{17}$$

where $\Delta x_{\text{leaf}} = 2h_x/p_w$ and $\Delta y_{\text{leaf}} = 2h_y/p_h$. Among all AMR grids that contain (p_x, p_y), the grid at the highest AMR level is selected, and the field value at the nearest grid cell center is copied:

$$v[c, i_y, i_x] = g^*_{\text{data}}\big[c,\ \lfloor (p_y - y_{\min}^{g^*})/\Delta y_{g^*} \rfloor,\ \lfloor (p_x - x_{\min}^{g^*})/\Delta x_{g^*} \rfloor\big], \tag{18}$$

where $g^* = \arg\max_{g \ni (p_x, p_y)} \ell_g$. This "highest-level wins" rule correctly resolves the AMReX overlap convention: where a fine grid exists, its data takes precedence over the coarser grid underneath.

Contrast with uniform-grid projection. A uniform-grid alternative would materialize the entire AMR hierarchy as a dense array. At the finest resolution represented by the capped quadtree, this array has shape $(C, H_{\max}, W_{\max})$, where $H_{\max} = n_y\, p_h\, 2^L$ and $W_{\max} = n_x\, p_w\, 2^L$. Here, L is the quadtree cap, not the simulator's highest AMR level. For problems with localized refinement (such as our AMReX combustion dataset, where refinement follows the flame front), the projected array is dominated by coarse regions that have been trivially interpolated to fine resolution, wasting memory and compute. Our quadtree tokenizer avoids this materialization entirely: it reads values from the finest AMR level available at each pixel location, and stores them in a variable-N token set whose size reflects the actual refinement structure.

C.7 Tensor conversion

For one frame, topology construction and value assignment yield the serialized tensor pair $(X, \mathbf{c})$. Leaves are enumerated in Morton (Z-curve) order and their rasters are stacked:

$$X \in \mathbb{R}^{N \times C \times p_h \times p_w}, \tag{19}$$

$$\mathbf{c} \in \mathbb{R}^{N \times 3}, \tag{20}$$

where N is the number of leaves, and $\mathbf{c}[i] = (\tilde{c}_x^{(i)}, \tilde{c}_y^{(i)}, \ell_i)$ are the normalized centers and level index. Center normalization maps (c_x, c_y) from $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$ to $[0, 1]^2$. The level index ℓ_i is stored as an integer and used directly as the third RoPE coordinate (see Section 2.1). Stacking the T_{in} per-frame tensors in this shared leaf order gives the $(N, C, T_{\text{in}}, p_h, p_w)$ input consumed by the transformer. At time t, $N = N_t$.

C.8 Random resampling augmentation

The wavelet refinement criterion creates a deterministic correlation between refinement level and spatial complexity: coarse tokens see smooth patches while fine tokens carry sharp features. We hypothesized that, if smooth patches also change slowly, this distribution could bias coarse-level predictions toward the identity map, which would be harmful after the inference-time regridder (Section 2.4) coarsens a previously refined region whose underlying physics is still active.

To weaken this correlation, we perturb the quadtree topology at training time between topology construction and data fill.

Procedure. Given ratios $(\rho_c, \rho_r) \in [0, 1]^2$, three phases are applied to the quadtree Q:

1. **Coarsen.** Collect all internal nodes whose four children are leaves with $\ell > 0$ (root tiles are preserved), yielding n_c candidates after deduplication. Sample $\max(1, \lfloor \rho_c n_c \rfloor)$ of them uniformly at random and coarsen them by the block-mean rule of Appendix C.3.
2. **Refine.** Collect all current leaves with $\ell_i < L$, yielding n_r candidates. Sample $\max(1, \lfloor \rho_r n_r \rfloor)$ uniformly at random and refine them by the bilinear-upsample rule of Appendix C.2. Candidates are re-collected after the coarsen phase, so freshly coarsened cells remain eligible for refinement.
3. **Restore balance.** Iterate the 2:1 balance restoration of Appendix C.4 to a fixed point. Termination in at most L passes follows from the monotone-refining argument of Appendix C.4.

Settings and scope. The main Table 1 runs disable this augmentation. When enabled in the PLI and AMReX ablations of Appendix G.4, it uses $\rho_c = 0.1$ and $\rho_r = 0.15$. Resampling is training-only. Inference-time topology follows the regridding procedure of Section 2.4.

C.9 Training-time tokenization pipeline

Algorithm 1 composes the primitives of Appendices C.2–C.7 to convert a single uniform-grid reference frame into the per-frame $(X, \mathbf{c})$ pair. For a training window, the topology is constructed once from its reference frame. Data fill and serialization then reuse that topology for each input and target frame, with the input tensors stacked as in Appendix C.7. The per-channel refinement loop (lines 3–14) is the wavelet-inspired local-range criterion of Section 2.1. Independent per-channel trees are unioned, and thresholding each channel relative to its own range R_c keeps ε a single global hyperparameter. The balance restoration after the union follows the monotone-refining procedure

of Appendix C.4. We then expand the refined regions by δ face-neighbor layers, refining coarser neighboring leaves and restoring 2:1 balance after each expansion. Random resample augmentation (Appendix C.8) is applied before data fill so that augmented topology inherits the block-average assignment rule, which guarantees that augmentation does not leak ground-truth fine detail into coarsened cells.

Algorithm 1 Training-time quadtree tokenization of a single reference frame

Require: uniform-grid field $x \in \mathbb{R}^{C\times H\times W}$, tolerance ε, adaptation channels $\mathcal{A} \subseteq \{1,\dots,C\}$, max depth L, refinement-dilation radius δ

Initialize Q as a uniform tiling of Ω by root cells of size (τ_w, τ_h), maximum depth L
Compute the global per-channel range $R_c \leftarrow \max_\Omega x_c - \min_\Omega x_c$ for each $c \in \mathcal{A}$
for each channel $c \in \mathcal{A}$ **do**
    Initialize Q_c as a copy of Q with all leaves at root level
    for $p = 1,\dots,L$ **do** ▷ iterative refinement passes
        for each leaf C_i of Q_c **do**
            Compute the local range $r_c(C_i) \leftarrow \max_{C_i} x_c - \min_{C_i} x_c$
            if $r_c(C_i) > \varepsilon \cdot R_c$ **and** $\ell_i < L$ **then**
                Split C_i into four equal children
            end if
        end for
        if no cells were split in this pass **then break**
        end if
    end for
end for
Merge the per-channel trees by union: a cell is refined in Q_{merged} iff it is refined in any Q_c
Set $Q \leftarrow Q_{\text{merged}}$ and restore 2:1 balance by refining any leaf whose face neighbor is more than one level finer
for $d = 1,\dots,\delta$ **do**
    Collect coarser face neighbors of the current refined leaves
    Refine the collected leaves and restore 2:1 balance
end for
if augmentation is enabled **then**
    Apply random resample augmentation to Q (Appendix C.8) with coarsen/refine ratios (ρ_c, ρ_r)
    Restore 2:1 balance
end if
Fill each leaf of Q by block-averaging the corresponding region of the source array x
Collect leaves in Morton order: X stacks the per-cell rasters, $\mathbf{c}$ stacks the per-cell centers
return $(X, \mathbf{c})$

The pipeline is staged so its expensive parts amortize across training. An offline *topology pass* runs Algorithm 1 up to and including fine-region expansion and balance restoration once per training window, building each window's tree from its reference frame. The result is serialized as a difference against the trivially uniform coarse tree and written to a Parquet table. At each training step the cached tree is reconstructed in memory and the optional augmentation and data-fill steps execute per visit, with the uniform-grid source frames streamed from disk afresh on every step rather than bundled into the cache. This lazy fill keeps augmented topology inside the exact-conservative regime of Eq. (15) and reduces cache size to a small fraction of the raw dataset. Both passes are scheduled by the distributed dataset abstraction of Ray [29]: the topology pass parallelizes across an autoscaling task pool with per-task memory budgets sized for heavily-refined windows, and the fill pass runs with asynchronous prefetch that overlaps the CPU-bound block-average with the GPU forward pass.

C.10 Three-phase regrid algorithm

Algorithm 2 gives inference-time regridding for default uniform-patch storage. Splitting a cell bilinearly upsamples each retained frame onto four children, and merging four siblings replaces each frame by its 2×2 block mean, as detailed in Appendices C.2–C.3. Multi-scale native storage uses the same three-phase structure with exact split/stitch value transfers (Appendix D.2). All refinement indicators, channel ranges, and protection/coarsening tests use the newest retained frame. Topology

changes transfer every frame of the retained input window. The balance-aware coarsening lookahead predicate used in Phase 3 is defined below.

Refinement criterion at inference. At inference the uniform-grid source field is unavailable and only the cellular state is accessible: a feature whose discontinuity aligns with a cell boundary leaves both adjacent cells looking smooth internally and would be missed by within-cell range alone. For channel c, let $r_c(C_i) = \max_{C_i} x_c - \min_{C_i} x_c$ denote the range within the leaf's raster. To detect variation between leaves, sample the cellular state one leaf-width from C_i's center in each direction $d \in \{W, E, S, N\}$, using the width along that direction. Each sample $v_c^{(d)}(C_i)$ is obtained by bilinear interpolation in the highest-level cell containing the sample point. With $v_c(C_i)$ denoting the center value, define the jump indicator as $b_c(C_i) = \max_d |v_c(C_i) - v_c^{(d)}(C_i)|$. Phase 1 uses the combined indicator $e_c(C_i) = \max\big(r_c(C_i),\, b_c(C_i)\big)$.

Phase 1 refines a leaf when $e_c(C_i) > \varepsilon_c R_c$ for some $c \in \mathcal{A}$, where ε_c is the inference tolerance for channel c. This retains the range-scaled threshold form used during training, but the tolerances may differ (Table 6). A scalar tolerance sets $\varepsilon_c = \varepsilon$ for every selected channel. Phases 2 and 3 instead use the within-cell range r_c alone, with the coarsen-side hysteresis threshold $\gamma\varepsilon_c R_c$ ($\gamma < 1$). For an internal node P whose four children are leaves, $r_c(P)$ denotes the per-channel range over the union of the four children's rasters.

Algorithm 2 Inference-time regridding with uniform-patch storage

Require: quadtree Q carrying the retained T_{in}-frame window, per-channel tolerances $\{\varepsilon_c\}_{c\in\mathcal{A}}$, coarsen ratio $\gamma < 1$, protection radius δ, adaptation channels $\mathcal{A}$, max level L, max passes per phase $p_{\max}$; per-channel global range R_c is computed from the newest retained frame

Phase 1: Refine

for $p = 1, \ldots, p_{\max}$ **do**
  for each leaf C_i of Q **do**
    if $\exists\, c \in \mathcal{A} : e_c(C_i) > \varepsilon_c \cdot R_c$ **and** $\ell_i < L$ **then**
      Split C_i into four children and bilinearly upsample each retained frame onto them
    end if
  end for
  Restore 2:1 balance by recursively splitting any leaf whose neighbor is more than one level finer
  if no cells were split in this pass **then break**
  end if
end for

Phase 2: Protect

Initialize all leaves as unprotected
if $\delta > 0$ **then**
  for each leaf C_i such that $r_c(C_i) \geq \gamma\varepsilon_c R_c$ for some $c \in \mathcal{A}$ **do**
    Mark C_i as protected
  end for
  Dilate the protected set by δ face-neighbor layers
  for $d = 1, \ldots, \delta$ **do**
    Collect protected leaves that are coarser than an adjacent protected leaf
    if no leaves are collected **then break**
    end if
    Refine the collected leaves, propagating protection to their children
    Restore 2:1 balance
  end for
end if

Phase 3: Coarsen (skipped if coarsening is disabled for this step)

for $p = 1, \ldots, p_{\max}$ **do**
  Collect internal nodes whose 4 children are all leaves
  for each collected node P **do**
    if no child is protected **and** lookahead of Eq. (21) holds **and** $\forall\, c \in \mathcal{A} : r_c(P) < \gamma \cdot \varepsilon_c \cdot R_c$ **then**
      Merge the four children of P into P by 2×2 block averaging in each retained frame
    end if
  end for
  if no nodes were merged in this pass **then break**
  end if
end for
return Q ▷ topology and retained window updated in place

When considering whether to coarsen a node P at level ℓ (i.e. remove its four leaf children and make P a leaf at level ℓ), the predicate must verify that no face neighbor of any child C_j of P (excluding siblings) is at level $\geq \ell + 2$. If such a neighbor exists, coarsening P would violate the 2:1 balance constraint and would trigger a cascade of re-refinements. The predicate therefore performs a lookahead:

$$\begin{aligned} &\forall\, C_j \in \text{children}(P), \quad \forall\, d \in \{W, E, S, N\}, \\ &\forall\, C_k \in \text{FaceNeighbors}(C_j, d) \setminus \text{children}(P) : \quad \ell_k < \ell + 2. \end{aligned} \tag{21}$$

The lookahead preserves 2:1 balance during coarsening, while the protection buffer limits immediate coarsening around regions with appreciable within-cell variation.

Implementation. Reported rollouts use a packed-array implementation of this algorithm with compiled CPU kernels (Section 2.4). If its output fails the 2:1-balance or leaf-overlap check, that regrid call is recomputed with a reference tree-based implementation of the same algorithm. Because protection and dilation admit more than one valid balanced result, the two implementations can select different topologies.

D Method (extended)

This appendix collects architectural and algorithmic detail deferred from Section 2. The transformer backbone block is described in Appendix D.1 and the multi-scale (native-resolution) variant of WAMRViT in Appendix D.2. Quadtree primitives (refinement, coarsening, 2:1-balance restoration, AMR mapping, and the random-resample augmentation) are in Appendix C; the 3D RoPE construction is in Appendix E.1 and the multi-scale tokenizer architecture in Appendix E.2; the inference-time regridding algorithm is in Appendix C.10.

D.1 Backbone block

Given the residual stream $h \in \mathbb{R}^{N\times d}$, first compute a shared normalized stream $\tilde{h} = \mathrm{LN}(h)$. The attention branch processes $\tilde{h}$ using RMSNorm on queries and keys (QK normalization [19]), 3D RoPE, and scaled dot-product attention. In parallel, the MLP branch applies a linear map followed by SwiGLU to $\tilde{h}$, with intermediate width $d_{\mathrm{mlp}} = \eta\, d$. The two branch outputs are concatenated and projected back to the residual width using $W_{\mathrm{proj}} \in \mathbb{R}^{d\times(d+d_{\mathrm{mlp}})}$, then added to h. The parallel arrangement uses a single pre-norm rather than the two of the sequential layout. We use unit MLP expansion $\eta = 1$ throughout: at $d_{\mathrm{mlp}} = d$ the joint projection has the same $2d^2$ parameter count as two independent branch projections summed in the residual stream.

After the block stack a final LayerNorm is followed by a linear projection $W_{\mathrm{out}} \in \mathbb{R}^{d\times(C_{\mathrm{out}}T_{\mathrm{out}}p_hp_w)}$ that maps each token back to a per-leaf raster, reshaped to $(N, C_{\mathrm{out}}, T_{\mathrm{out}}, p_h, p_w)$ for adaptive mode or gathered into a dense $(C_{\mathrm{out}}, T_{\mathrm{out}}, H, W)$ grid for the uniform-grid baseline. The multi-scale variant of Appendix D.2 replaces both the per-block pre-norm and this post-stack norm with the level-conditioned AdaLN of Eq. (24) and uses a per-level decoder. The block stack is held identical across default uniform-patch WAMRViT and ViT-finest/mid, providing a shared-backbone comparison. The multi-scale variant differs as stated above.

D.2 Multi-scale (native-resolution) variant

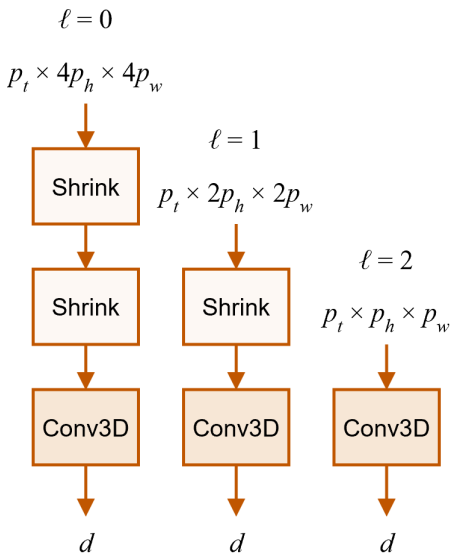


Figure 8: Per-level patchify cascade of the multi-scale variant.

The default tokenizer downsamples coarse leaves by exact block averaging (Eq. (15)) to a shared (C, p_h, p_w) raster, so each output pixel of a level-ℓ leaf summarizes $4^{L-\ell}$ source pixels, and sub-raster

detail of coarse leaves cannot be recovered. With the motivation of a shared latent space [30] across refinement levels, the multi-scale variant retains each leaf's full source region and handles the change of resolution inside the model rather than at the data-fill step. The validated design requires a uniform-grid source and scatters predictions back to its finest grid for the full-field loss. "Native-resolution" here describes the retained per-leaf source region, not direct training on simulator-native AMR data.

Native value storage. In native value storage, the block averaging of Eq. (15) is replaced by a direct copy: each level-ℓ leaf retains its full $(C, p_h r_\ell, p_w r_\ell)$ source region, with native-resolution factor $r_\ell = 2^{L-\ell}$, so finest-level rasters are unchanged and only coarser leaves grow. The rest of Algorithm 1 (per-channel refinement and union, 2:1 balance, dilation, and optional resampling) acts on topology alone and is unchanged. For an input window, tensor conversion (Appendix C.7) stacks the T_{in} frames into a per-level dictionary $\{\ell : (N_\ell, C, T_{\text{in}}, p_h r_\ell, p_w r_\ell)\}_{\ell=0}^{L}$ together with a leaf-to-bucket index recording each token's position within its level. This is the input of the per-level patch embedder (Appendix E.2). Rasters and partition formulas below are written per frame and applied independently across the window.

Refine and coarsen under native storage. Native storage reorganizes but does not destroy information at topology changes, so the bilinear refinement of Appendix C.2 and the block-mean coarsening of Appendix C.3 simplify to exact partition operations. Refinement of a level-ℓ leaf with raster $v \in \mathbb{R}^{C \times p_h r_\ell \times p_w r_\ell}$ splits v along its spatial axes into four contiguous $(C, p_h r_\ell/2, p_w r_\ell/2)$ blocks and assigns them to the four children at level $\ell + 1$, each of which is exactly the native-storage shape $(C, p_h r_{\ell+1}, p_w r_{\ell+1})$ since $r_{\ell+1} = r_\ell/2$. Coarsening is its inverse: the four children are concatenated in their NW/NE/SW/SE layout (Eq. (11)) into a $(C, p_h r_\ell, p_w r_\ell)$ super-patch, which is stored verbatim at the parent rather than reduced by the 2×2 block-mean of Eq. (12). Both operations are bit-exact and conservative. The bilinear smoothing and the mean reduction of the uniform mode are not needed because the native shape already accommodates each child's pixel count.

Architecture and training objective. Each level's bucket is embedded by a per-level patchify cascade of $L - \ell$ learned shrink stages (Figure 8) and decoded back to its native raster by a per-level expansion head, and level-conditioned AdaLN replaces the backbone's pre-norm and output norm. Appendix E.2 specifies these components. Predictions are scattered onto the finest uniform grid and trained with the full-field loss of Appendix F.1.

E Scale-aware design choices

Two design choices govern how WAMRViT handles cells that differ in physical extent and refinement level: (i) the per-axis scales of the 3D rotary positional embedding, which place the level axis on a soft-bias footing relative to the spatial axes; and (ii) the optional multi-scale tokenizer for uniform-grid inputs, which retains each leaf at its source resolution rather than projecting all leaves to a shared finest patch shape. Both target the same mismatch between a fixed-shape transformer block stack and a token set whose elements span $L + 1$ refinement levels and so differ in physical area by factors of up to 4^L within a single sample.

E.1 3D RoPE configuration

Per-axis frequency partition. The head dimension d_h is partitioned across the three axes, $d_h = d_x + d_y + d_\ell$, and each axis $a \in \{x, y, \ell\}$ owns an independent block of $d_a/2$ frequency planes in the axial 2D-RoPE style of Heo et al. [31]. On axis a, plane $j \in \{0, \ldots, d_a/2 - 1\}$ has angular frequency

$$\omega_j^{(a)} \;=\; \theta_{\text{base}}^{-2j/d_a}, \qquad \theta_{\text{base}} = 10{,}000, \tag{22}$$

so each axis is anchored at a common fastest plane $\omega = 1$ at $j = 0$, with the slowest plane decaying as $\theta_{\text{base}}^{-(d_a-2)/d_a}$. Larger d_a extends the spectrum further into low frequency, while smaller d_a truncates it. For the spatial axes, $p_a(C_i) = s_a \tilde{c}_a(C_i)$, $a \in \{x, y\}$, where $\tilde{c}_a \in [0, 1]$ is the normalized center coordinate. For the level axis, $p_\ell(C_i) = s_\ell \ell(C_i)$, where $\ell(C_i) \in \{0, \ldots, L\}$. RoPE applies a block-diagonal 2×2 rotation by phase $\omega_j^{(a)} \cdot p_a$ to each query/key on plane j [14]. The per-axis rotations are concatenated along the head dimension before the attention dot product.

Spatial scales. The spatial scales use the shorter side of a finest-level leaf as a common length unit for both axes. Let $\lambda = \min(\tau_w, \tau_h)/2^L$ be the shorter side of a finest-level leaf in domain units. Two finest-level neighbors along axis a have normalized-coordinate separation $\Delta\tilde{c}_a = (\tau_a/2^L) \,/\, W_a$ with $W_a \in \{W, H\}$ the domain extent on that axis. Setting $s_x = W/\lambda$ and $s_y = H/\lambda$ then yields a unit position step at the finest level along the smaller-tile direction, with the other axis inheriting a proportionally larger step when tiles are not square. Coarser cells inherit proportionally larger steps because their physical separation is larger. For square root tiles ($\tau_w = \tau_h$) this reduces to $s_x = W/\lambda = n_x \cdot 2^L$ and $s_y = H/\lambda = n_y \cdot 2^L$ with n_x, n_y the root-tile counts, recovering the uniform ViT convention exactly.

Level axis as soft bias. The level axis carries the integer refinement level $\ell \in \{0, \ldots, L\}$ rather than a continuous coordinate, and its scale is set to $s_\ell = 1$. The 2:1 balance constraint guarantees that adjacent leaves differ by at most one level, so cross-level attention is the regime in which refined regions communicate with their coarser neighborhoods rather than an edge case. The level coordinate distinguishes refinement levels explicitly within the positional encoding. Setting $s_\ell = 0$ removes this contribution from RoPE, while increasing s_ℓ increases the relative rotary phases associated with a given level difference. The resulting attention scores depend on the learned queries and keys.

To examine the level contribution alone, consider an idealized comparison with equal query and key vectors, $q = k$, and zero spatial phase difference, varying only the level difference $\Delta\ell$. Because RoPE is a per-plane rotation and the dot product across planes is additive, the post-RoPE dot product (with uniform per-plane magnitudes $\|q_j\|^2 = 2/d_h$) is

$$q^\top k(\Delta x{=}\Delta y{=}0,\, \Delta\ell) \;=\; \frac{d_x + d_y}{d_h} \;+\; \frac{2}{d_h} \sum_{j=0}^{d_\ell/2-1} \cos\!\big(\omega_j^{(\ell)} \cdot \Delta\ell\big), \tag{23}$$

normalized so that $\Delta\ell = 0$ yields 1. With the PLI configuration of Table 4 ($d_h = 192$, $d_\ell = 20$, $\theta_{\text{base}} = 10{,}000$), Eq. (23) evaluates to ≈ 0.994 at $\Delta\ell = 1$: the post-RoPE dot product retains roughly 99.4% of its same-level value. In this idealized comparison, the default $s_\ell = 1$ produces only a small change in the dot product for adjacent levels. Larger scales change the cosine terms non-monotonically. Between adjacent quadtree levels, cell side lengths and perimeter differ by a factor of 2, and area by a factor of 4, irrespective of aspect ratio. These ratios motivate the geometric interpretations of $s_\ell = 2$ and $s_\ell = 4$. PLI uses a 2:1 rectangular leaf $(p_h, p_w) = (10, 5)$ and non-square spatial scales $(s_x, s_y) = (80, 224)$, so its level increment occupies a different relative scale than on square-leaf AMReX. The forecasting effects of s_ℓ across datasets, metrics, evaluation scopes, and horizons are evaluated in Appendix G.5.

Frequency-plane sizing. We use $d_\ell \geq 4(L+1)$, allocating at least two 2-dimensional frequency planes per level. The configurations in Table 4 use $d_\ell = 20$ at $L = 3$ on PLI, $d_\ell = 16$ at $L = 2$ on TRL, and $d_\ell = 24$ at $L = 4$ on AMReX.

E.2 Multi-scale tokenization architecture

The multi-scale variant of Appendix D.2 produces a per-level token bucket $\{\ell : V_\ell \in \mathbb{R}^{N_\ell \times C \times T_{\text{in}} \times p_h r_\ell \times p_w r_\ell}\}_{\ell=0}^{L}$, where the native-resolution factor $r_\ell = 2^{L-\ell}$ is distinct from the RoPE level-axis scale s_ℓ, rather than a single $(N, C, T_{\text{in}}, p_h, p_w)$ tensor. A per-level patch embedder converts each bucket to tokens of the shared embedding dimension d, and a per-level output head decodes back to native shape. We describe both below.

Per-level patchify cascade. Let $\Delta = L - \ell$ denote the number of shrink stages at level ℓ. Each stage is a 3D convolution with kernel $(1, 2, 2)$ and stride $(1, 2, 2)$ followed by a learned SwiGLU block, halving both spatial extents and multiplying the channel count by a fixed factor m (the channel multiplier). Starting from $C_0 = C$, the cascade produces a tensor of shape $(Cm^\Delta, T_{\text{in}}, p_h, p_w)$. Its spatial dimensions match those of uniform-mode leaves, with learned reduction replacing block-mean data-fill (Appendix C.5). A final per-level $\mathrm{Conv3d}(Cm^\Delta, d, (p_t, p_h, p_w), \mathrm{stride}{=}(p_t, p_h, p_w))$ projects to the shared embedding dimension d. The finest level ($\Delta = 0$) has no shrink stages and reduces to the uniform-mode patch embedder. With $m = 2$, each shrink stage halves the activation size. Coarser levels require more stages and larger per-level projections.

Per-level output head. The decoder begins with a per-level linear map $W_\ell \in \mathbb{R}^{d \times (C_{\text{out}} m^\Delta T_{\text{out}} p_h p_w)}$, where $\Delta = L - \ell$, followed by a reshape to $(N_\ell, C_{\text{out}} m^\Delta, T_{\text{out}}, p_h, p_w)$. A cascade of Δ expansion

stages then doubles both spatial dimensions and reduces the channel count by a factor m at each stage, using a 3×3 convolution followed by factor-two pixel shuffle. Intermediate stages include a SwiGLU block. The final stage omits the activation. Expansion operates independently on each output frame and produces the native raster $(N_\ell, C_{\text{out}}, T_{\text{out}}, p_h r_\ell, p_w r_\ell)$. At the finest level, the cascade is empty and decoding reduces to the uniform-mode linear projection.

Backbone with level-conditioned norm. The transformer block stack is unchanged in shape (tokens of dimension d, sequence length $N = \sum_\ell N_\ell$ regardless of tokenization mode), but each block's pre-norm and the post-stack output norm are replaced by independent level-conditioned affine modulations in the AdaLN form of Peebles and Xie [32]. For block index b, hidden state $h_i \in \mathbb{R}^d$ at token i, and per-token level index $\ell(i) \in \{0, \dots, L\}$,

$$\mathrm{AdaLN}^{(b)}\big(h_i;\, \ell(i)\big) \;=\; \big(1 + \alpha^{(b)}_{\ell(i)}\big) \odot \mathrm{LN}(h_i) \;+\; \beta^{(b)}_{\ell(i)}, \tag{24}$$

where each block owns a learned embedding $E^{(b)} \in \mathbb{R}^{(L+1)\times 2d}$ whose level-ℓ row splits into $\alpha^{(b)}_\ell$, $\beta^{(b)}_\ell \in \mathbb{R}^d$. All $E^{(b)}$ are initialized to zero, so every $\mathrm{AdaLN}^{(b)}$ reduces to plain LN at initialization. Its level-specific scale and shift parameters are then learned jointly with the rest of the model. We do not adopt the residual-branch gating that completes the AdaLN-Zero variant of Peebles and Xie [32]. The level-conditioned norm is enabled only in the multi-scale tokenizer. On the uniform-patch tokenizer the same modulation regressed in preliminary experiments (not reported), suggesting that its gains are tied to the multi-scale token distribution.

Three roles for the level index. In the multi-scale variant, the refinement level enters the model in three ways: as the third coordinate of the 3D RoPE (Appendix E.1); as the resolution index that fixes each token's native raster size and its patch-embedder depth; and as the conditioning index of the AdaLN modulation in Eq. (24). RoPE encodes relative spatial position and refinement level in the attention mechanism. The multi-scale tokenizer keeps each leaf's raster at its source resolution, so the model sees the underlying field at native fidelity per level. The level-conditioned norm gives each level a learned scale-and-shift that absorbs systematic distributional differences (e.g. coarse leaves dominated by smooth content vs. fine leaves carrying sharp features), without requiring the backbone to recover that distinction from RoPE alone. The variant reported as "WAMRViT (multi-scale)" in Section 3.4 adopts all three.

F Training and evaluation specifications

F.1 Training specifications

Table 4 lists the training hyperparameters used for the adaptive-WAMRViT runs reported in Section 3.4. The SwinV2 counterpart is listed in Table 5. Rows with a single value apply to all three datasets. All runs use Intel Xeon Platinum 8468 CPUs alongside the GPUs listed in the table.

Table 4: Training specifications for the WAMRViT runs reported in Section 3.4. Topology resampling is disabled. AMReX builds its quadtree from the simulator hierarchy (Appendix C.6). Root tiles are given in finest-level cells. The AMReX tile equals 32×32 level-0 cells.

	PLI	**TRL**	**AMReX**
Backbone (shared)			
Layers / heads / per-head d_h		6 / 12 / 192	
Transformer parameters		286.7M	
Complete parameters (uniform)	290.2M	287.5M	407.1M
Complete parameters (multi-scale)	339.0M	292.0M	–
Tokenizer			
Input channels	10	4	17
Window ($T_{\text{in}} \to T_{\text{out}}$)	$2 \to 1$ ($p_t = 2$)	$4 \to 1$ ($p_t = 4$)	$2 \to 1$ ($p_t = 2$)
Patch size (p_h, p_w)	$(10, 5)$	$(4, 4)$	$(32, 32)$
Root tile (τ_h, τ_w)	$(80, 40)$	$(16, 16)$	$(512, 512)$
Levels $(L + 1)$	4	3	5
Adaptation channels	9 of 10	4 of 4	HRR
Tolerance ε	0.01	0.1	0.01
Refinement dilation (hops)		2	
3D RoPE			
Axis head dims (d_x, d_y, d_ℓ)	$(86, 86, 20)$	$(88, 88, 16)$	$(84, 84, 24)$
Axis scales (s_x, s_y, s_ℓ)	$(80, 224, 1)$	$(96, 32, 1)$	$(768, 128, 1)$
Optimization			
Optimizer		AdamW, clip 1.0	
Learning rate		10^{-4}	
LR schedule		cosine, 1000-step warmup, min 10^{-6}	
Training loop			
Micro-batch per GPU; precision		1 (variable N); BF16 autocast	
Effective batch		8	
Push-forward unroll K	1	1	4
Epochs	10	200	600
EMA		0.9999 decay, 1000-step warmup, $\gamma_{\text{inv}} = 1.0$, power $2/3$	
Hardware			
GPUs		8 H100	

Table 5: SwinV2 specifications for the regular-grid architecture-family control. The four conventional shifted-window stages approximately match WAMRViT's transformer capacity. The refined FPN decoder uses two ConvNeXt blocks per non-bottleneck skip and a learned 3×3 correction after bilinear upsampling, inspired by Poseidon [33]. On AMReX, the hierarchy is projected onto the uniform grid of AMR level 2 (1024×6144), four times coarser per direction than the finest level.

	PLI	**TRL**	**AMReX**
Backbone and decoder			
Stage depths		[8, 12, 12, 6]	
Heads per stage		[6, 12, 24, 48]	
Transformer-stage parameters		286.5M	
Embedding dim / decoder channels		192 / 256	
Decoder refinement blocks		2	
Complete model parameters	294.2M	292.9M	334.5M
Tokenizer and attention			
Input channels	10	4	17
Window ($T_{\text{in}} \to T_{\text{out}}$)	$2 \to 1$ ($p_t = 2$)	$4 \to 1$ ($p_t = 4$)	$2 \to 1$ ($p_t = 2$)
Patch size (p_h, p_w)	$(10, 5)$	$(4, 4)$	$(32, 32)$
Image size (H, W)	$(1120, 400)$	$(128, 384)$	$(1024, 6144)$
Attention window w	2	2	8

Micro-batching for adaptive runs. Adaptive tokenization yields a sample-dependent token count, so two adaptive samples cannot be stacked into a single tensor. Two work-arounds exist: padding within a micro-batch to a common $N_{\max}$ with attention masked on the padding slots, or block-diagonal packing with a custom attention mask. We adopt neither and instead train at micro-batch one with data-parallel replication over eight GPUs, for an effective batch size of eight. The ViT-finest, ViT-mid, and SwinV2 baselines use the same configuration. Padded and packed alternatives are left to future work (Appendix I).

Objective function. All reported model variants use an absolute-L_2 objective, with per-leaf reduction for uniform-patch WAMRViT and full-field reduction for the uniform-grid baselines and multi-scale WAMRViT. Section 2.3 defines both objectives. Their tensor layouts, numerical stabilizer, and push-forward recursion are detailed below.

Let $\hat{y}^{(k)}$, $y^{(k)}$ denote the prediction and ground truth at push-forward step k. For uniform-patch WAMRViT, the loss tensor shape is $(N_t, C, T_{\text{out}}, p_h, p_w)$, with N_t the leaf count of the current topology and per-leaf flattened volume $M_p = T_{\text{out}}\, p_h\, p_w$ entering the inner L_2 of Eq. (2). For the uniform-grid baselines (ViT-mid, ViT-finest, and SwinV2), the output head reassembles patches into the full field before the loss, so the loss tensor shape is $(1, C, T_{\text{out}}, H, W)$. Equation (3) suppresses the singleton leading axis and takes the norm over the full $T_{\text{out}} \times H \times W$ spatiotemporal volume of one channel. Both objectives adapt the L_p loss of the NeuralOperator library [3, 34] with $d=3$, $p=2$, adding a numerical stabilizer $\epsilon = 10^{-4}$ inside the square root of the inner L_2. We use the *absolute* variant of L_2 rather than the *relative* default of the same library. The relative variant normalizes by $\|y\|_2$ and degrades on regions where the ground truth is near zero.

For uniform-patch WAMRViT, push-forward training averages the per-leaf losses over the K unrolled steps,

$$\mathcal{L}(\theta) \;=\; \frac{1}{K}\sum_{k=1}^{K} \ell\big(\hat{y}^{(k)},\, y^{(k)}\big), \qquad \hat{y}^{(k)} = f_\theta\big(\hat{y}^{(k-T_{\text{in}}):(k-1)}\big), \tag{25}$$

with $\hat{y}^{(j)} = x_{t+j}$ for $j \in \{-T_{\text{in}}+1, \ldots, 0\}$, so each gradient step sees K autoregressive applications of f_θ. The window passed to the next unrolled step is detached, so each step's loss backpropagates only through its own model application. For the uniform-grid baselines and multi-scale variant, the same K^{-1} average instead uses ℓ_{full}, evaluated after full-field reassembly or the full-field scatter described below, respectively. K equals 1 on the PLI and TRL runs and 4 on the AMReX run (Table 4).

Multi-scale variant: full-field loss. The multi-scale (native-resolution) variant of Appendix D.2 emits per-level predictions $\{\hat{y}_\ell^{(k)} \in \mathbb{R}^{N_\ell \times C \times T_{\text{out}} \times p_h r_\ell \times p_w r_\ell}\}_{\ell=0}^{L}$ with native-resolution factor $r_\ell = 2^{L-\ell}$, rather than a single tensor at a shared raster size. We do not invoke Eq. (2) on each per-level bucket separately. Instead, the per-level predictions and targets are first scattered onto the finest uniform grid via a differentiable advanced-index assignment, producing $\hat{y}_u^{(k)}, y_u^{(k)} \in \mathbb{R}^{1 \times C \times T_{\text{out}} \times H \times W}$, and the full-field loss ℓ_{full} of Eq. (3) is then applied to this pair. Multi-scale WAMRViT therefore uses the same full-field objective as the uniform-grid baselines. The scatter introduces no learnable parameters and admits gradients back into the per-level buckets through the assignment, so the multi-scale model trains end-to-end against the full-field loss.

Training and rollout orchestration. Training and rollout both run on the Ray distributed runtime [29] and share the data pipeline of Appendix C.9. Rollout disables topology resampling. A PyTorch DDP trainer launches one data-parallel worker per GPU, with checkpointing, EMA tracking, and logging owned by rank zero.

F.2 Training and inference wall-clock time

The per-call token count $\bar{N}$ that drives WAMRViT's forward time is set at inference by the regrid hyperparameters in Table 6: the tolerance ε controls refinement and coarsening, the adaptation channels restrict which fields the detector watches, and the protection radius prevents flagged neighborhoods from immediate coarsening (Appendix C.10). Inference timings cover 20 model calls per starting window. Adaptive models additionally perform four complete regrid events at an interval of four calls. Total inference time in Table 2 is reported as mean $\pm$ population standard deviation

across the ten matched starting windows. Peak memory reports the maximum PyTorch-allocated CUDA memory and excludes non-PyTorch allocations. For TRL, the timing run is an engineering throughput benchmark, distinct from its one-step accuracy protocol. Both TRL adaptive variants use tolerance 0.1 in all four channels for initial mesh construction and regridding, as in the full-test accuracy and qualitative evaluations. All WAMRViT models start from identical topologies, but their meshes can diverge after predictions.

Table 6: Inference regrid settings for Table 2. PLI and TRL tolerances are per selected channel. AMReX uses a scalar tolerance. "CPU array" denotes compiled, parallel CPU kernels that update both the topology and the leaf values. For AMReX, "GPU values" means that only the topology is computed on the CPU. The leaf values stay on the GPU, where the resulting split/merge records are applied (Section 2.4). All regrids use coarsen ratio $\gamma = 0.25$ and at most $p_{\max} = 10$ passes per phase (Algorithm 2).

Dataset	Variant	Adaptation channels	Tolerance ε	Radius δ	Execution
PLI	WAMRViT (uniform)	3 of 10	$[0.1,\ 0.1,\ 0.3]$	2	CPU array
	WAMRViT (multi-scale)	3 of 10	$[0.1,\ 0.1,\ 0.3]$	2	CPU array
TRL	WAMRViT (uniform)	4 of 4	$[0.1,\ 0.1,\ 0.1,\ 0.1]$	2	CPU array
	WAMRViT (multi-scale)	4 of 4	$[0.1,\ 0.1,\ 0.1,\ 0.1]$	2	CPU array
AMReX	WAMRViT (uniform)	HRR	0.01	2	CPU array + GPU values

Table 2 separates predictor, complete-regrid, and end-to-end scopes. Regridding accounts for $8.0\%/13.9\%$ of uniform/multi-scale PLI inference, $17.0\%/5.4\%$ on TRL, and 17.2% on AMReX. Including those events, uniform/multi-scale WAMRViT is $3.0\times$ /$2.6\times$ faster than ViT-finest on PLI and approximately $1.2\times$ faster for both TRL variants. Multi-scale TRL takes 0.6% longer than uniform WAMRViT, so their mean totals are nearly tied under matched tolerances. On AMReX, the WAMRViT forward pass is faster than ViT-mid, but complete regridding makes total time 8.6% higher. SwinV2 is fastest and uses the least peak allocated VRAM on all three datasets. Parameter matching therefore does not imply runtime or memory matching.

AMReX adaptive training remains dominated by data-pipeline materialization rather than accelerator compute. The reported uniform-patch path stores every leaf as a $17 \times 32 \times 32$ raster, producing much larger streaming blocks than PLI or TRL. For uniform-grid sources, the validated multi-scale variant computes its full-field loss after scattering predictions onto the finest source grid. Applying this objective to AMReX would require materializing the 4096×24576 grid, so we do not use that variant for the native-AMReX experiment. The reported AMReX training time includes materialization of the uniform-patch data blocks. Length-bucketed batching, block-diagonal packing, and a non-materializing native objective remain future work (Appendix I).

F.3 Evaluation metrics

Full-field evaluation is performed on PLI and TRL; AMReX uses the per-level cell-space metrics defined below. For full-field evaluation, let $\hat{y}, y \in \mathbb{R}^{C\times H\times W}$ denote the prediction and ground truth at a given rollout step on the common finest uniform grid, with $N_p = H \cdot W$ cells and C physical channels. Uniform-patch WAMRViT leaves are expanded to this grid by nearest-neighbor upsampling, whereas multi-scale leaves already store their region at this resolution and are placed directly. All metrics are computed per channel c and then averaged over channels.

RMSE. The root-mean-squared error for channel c is

$$\mathrm{RMSE}_c(\hat{y}, y) \;=\; \sqrt{\frac{1}{N_p}\sum_{j=1}^{N_p}\big(\hat{y}_{c,j} - y_{c,j}\big)^2}. \tag{26}$$

VRMSE. The variance-scaled root-mean-squared error normalizes RMSE_c by the spatial standard deviation of the ground truth on the same frame, as in Wang and Perdikaris [13] and Ohana et al. [22]. We follow The Well's benchmark implementation, using Bessel-corrected spatial variance:

$$\mathrm{VRMSE}_c(\hat{y}, y) \;=\; \sqrt{\frac{\frac{1}{N_p}\sum_{j=1}^{N_p}\big(\hat{y}_{c,j} - y_{c,j}\big)^2}{\sigma_c^2(y) + \varepsilon}}, \qquad \sigma_c^2(y) = \frac{1}{N_p - 1}\sum_{j=1}^{N_p}\big(y_{c,j} - \bar{y}_c\big)^2, \tag{27}$$

where $\bar{y}_c$ is the spatial mean of channel c and $\varepsilon = 10^{-7}$ guards against division by zero on constant frames. Variance scaling reduces sensitivity to channel magnitude, but near-zero target variance can still produce large relative errors. The numerical stabilizer does not prevent such channels from dominating the aggregate.

Per-level cell-space variant. The per-level RMSE and VRMSE reported in Table 1 and Figure 4 are computed directly on the quadtree, without projection to a uniform grid, as an equal per-leaf average. On AMReX's finest leaves, regular-grid predictions are mapped to the finer evaluation raster by piecewise-constant prolongation (nearest-neighbor repetition). Let $\mathcal{N}_\ell$ index the leaves at level ℓ, and let each leaf contain $M = p_h p_w$ pixels under uniform-patch storage or $M = p_h p_w r_\ell^2$ pixels under native multi-scale storage, where $r_\ell = 2^{L-\ell}$. Let $\hat{y}_{i,c}, y_{i,c} \in \mathbb{R}^M$ denote the predicted and ground-truth values of channel c on leaf i. Both storage modes use the same metric definitions, evaluated over their retained raster. At the finest level, $r_L = 1$, and the raster sizes coincide. The per-leaf squared error and per-leaf ground-truth variance are

$$\mathrm{MSE}_{i,c} \;=\; \frac{1}{M}\sum_{j=1}^{M}\big(\hat{y}_{i,c,j} - y_{i,c,j}\big)^2, \qquad \sigma^2_{i,c}(y) \;=\; \frac{1}{M-1}\sum_{j=1}^{M}\big(y_{i,c,j} - \bar{y}_{i,c}\big)^2,$$

and the reported per-level metrics are the mean of per-leaf metrics across the $|\mathcal{N}_\ell|$ leaves at level ℓ,

$$\mathrm{RMSE}_{c,\ell} \;=\; \frac{1}{|\mathcal{N}_\ell|}\sum_{i\in\mathcal{N}_\ell}\sqrt{\mathrm{MSE}_{i,c}}, \qquad \mathrm{VRMSE}_{c,\ell} \;=\; \frac{1}{|\mathcal{N}_\ell|}\sum_{i\in\mathcal{N}_\ell}\sqrt{\frac{\mathrm{MSE}_{i,c}}{\sigma^2_{i,c}(y)+\varepsilon}}. \tag{28}$$

The VRMSE denominator is the *per-leaf* ground-truth variance $\sigma^2_{i,c}(y)$, computed over the M pixels of leaf i with the same Bessel correction, not the frame-level $\sigma^2_c(y)$. This normalization measures each leaf's error relative to its local spatial variation, and the level average gives every leaf equal weight. It is therefore a local-contrast diagnostic rather than a guarantee of balanced contributions across fields: nearly constant leaves can produce very large ratios, particularly for sparse HRR on AMReX. We retain per-field RMSE and VRMSE breakdowns to expose this limitation.

G Additional results

This appendix provides first-step RMSE, aggregated and per-field rollout curves on PLI and AMReX combustion, the push-forward, random-resample, and level-scale ablations, and the PLI mesh-policy sensitivity study. Rollout curves and PLI/AMReX ablations use a fixed seed. Qualitative snapshots are in Appendix A.

G.1 First-step RMSE summary

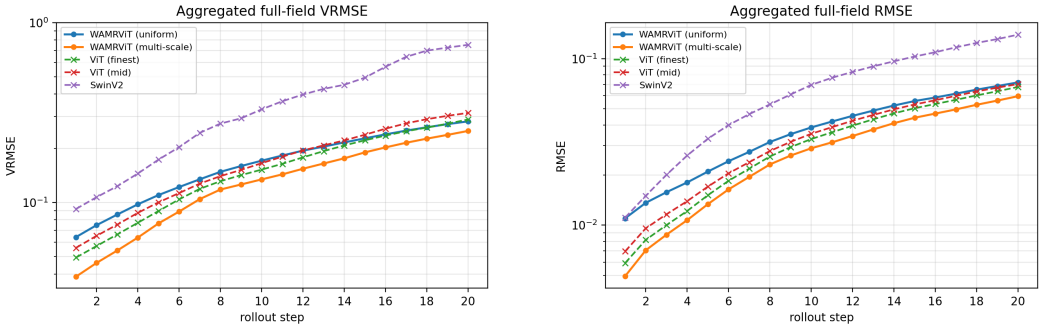
Table 7 reports first-step RMSE for the same evaluations as Table 1. The IAP* column is omitted because its paper does not report RMSE. Multi-scale WAMRViT has the lowest full-field RMSE on PLI and TRL. Evaluated over the regions refined to level L by its own topology, its RMSE is 22.8% below the ViT-finest baseline evaluated on the same regions on PLI and 1.4% below on TRL. These evaluations agree with the ViT-finest values listed in Table 1 at the reported precision at the first step. Refined regions can diverge during subsequent regridding. At the PLI finest level, the ViT-finest and uniform-WAMRViT RMSE means differ by less than 0.1% and their seed spreads overlap. Uniform WAMRViT is $60.8\%/60.2\%$ below ViT-mid/SwinV2 on AMReX.

Table 7: **First-rollout-step RMSE** (channel mean), counterpart of Table 1. Entries are mean ± sample standard deviation over three training seeds. Training depth is $K = 1$ for PLI/TRL and $K = 4$ for AMReX. Finest-level metrics are evaluated on the regions refined to level L in the initial adaptive topology. AMReX omits ViT-finest because its level-4 grid (4096×24576) cannot be materialized. ViT-mid and SwinV2 use $p = 32$.

	WAMRViT (uniform)	WAMRViT (multi-scale)	ViT (finest)	ViT (mid)	SwinV2
PLI, full-field	0.0110 ±6.232E-5	**4.963E-3** ±2.198E-5	6.092E-3 ±2.338E-4	6.932E-3 ±4.526E-5	0.0111 ±6.474E-5
PLI, finest ($\ell = 3$)	2.475E-3 ±8.896E-5	**1.909E-3** ±1.167E-5	2.473E-3 ±1.338E-4	2.858E-3 ±3.869E-5	5.702E-3 ±4.940E-5
TRL, full-field	0.5185 ±4.802E-3	**0.4979** ±8.661E-4	0.5069 ±1.483E-4	0.5972 ±3.193E-3	0.7047 ±3.396E-3
TRL, finest ($\ell = 2$)	**0.3283** ±3.064E-3	0.3333 ±1.103E-3	0.3382 ±2.818E-4	0.4007 ±2.046E-3	0.5193 ±7.857E-4
AMReX, finest ($\ell = 4$)	**1.202E10** ±8.293E8	/	/	3.064E10 ±1.818E8	3.022E10 ±1.498E8

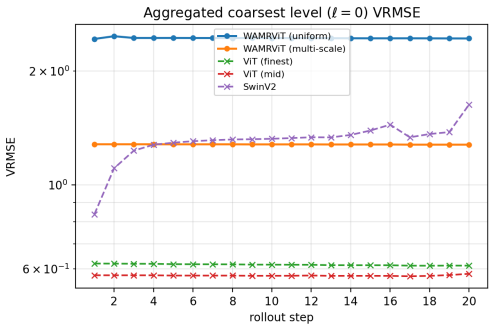
G.2 Aggregated rollout curves on PLI

All panels are at push-forward $K = 1$, averaged over all evaluation windows, sharing the legend of the PLI panels of Figure 4. Only the ViT-finest series paired with uniform-patch WAMRViT is plotted. Its cell-space errors use uniform WAMRViT's evolving topology, while multi-scale WAMRViT retains its own topology. Paired numerical comparisons with multi-scale WAMRViT use its separately evaluated ViT-finest reference, which is not plotted. Intermediate-level ($\ell \in \{1, 2\}$) breakdowns are omitted for readability.

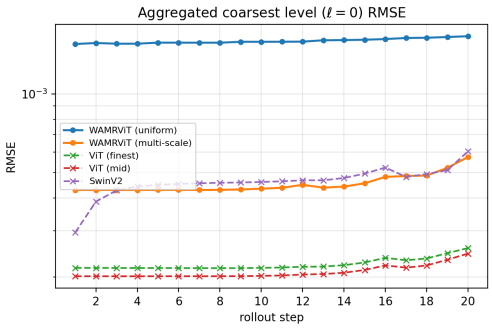


(a) Full-field VRMSE. (b) Full-field RMSE.

Figure 9: **Aggregated full-field error on PLI**, $K = 1$.

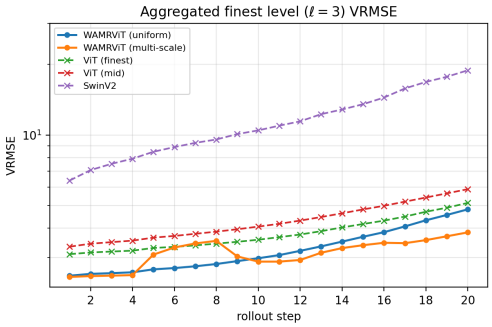


(a) $\ell = 0$ VRMSE.

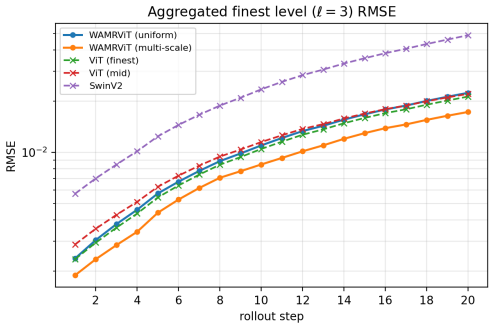


(b) $\ell = 0$ RMSE.

Figure 10: **Aggregated coarsest-level error on PLI**, $K = 1$. Each coarsest leaf spans 8^2 times the finest-leaf area. Uniform-patch WAMRViT stores 50 pixels per leaf, whereas multi-scale WAMRViT retains an 80×40 raster. The displayed regular-grid controls are block-mean projected onto uniform WAMRViT's leaf rasters.



(a) Finest ($\ell = 3$) VRMSE.



(b) Finest ($\ell = 3$) RMSE.

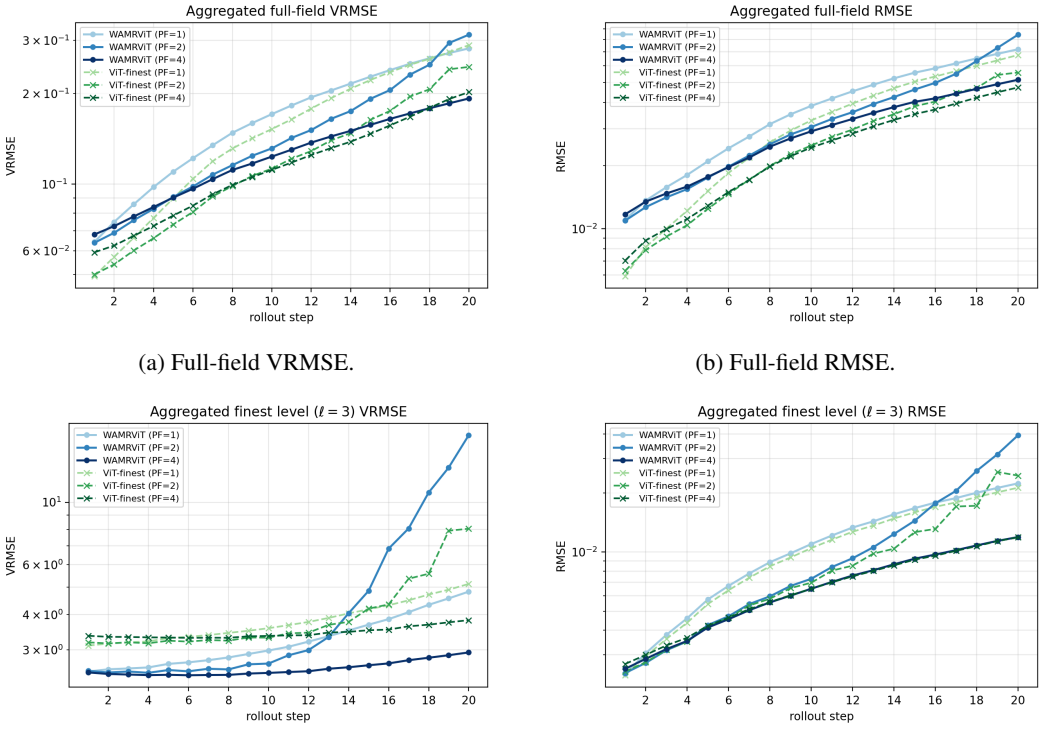
Figure 11: **Aggregated finest-level error on PLI**, $K = 1$.

G.3 Push-forward ablation on PLI

WAMRViT (uniform tokenization) and the ViT-finest baseline are each trained at three push-forward depths $K \in \{1, 2, 4\}$ on PLI, with all other hyperparameters held fixed to the values in Table 4 (which uses $K = 1$ as the PLI main-text setting). First-step VRMSE numbers are summarized in Table 8. Aggregated full-field and finest-level rollout curves averaged over all evaluation windows are shown in Figure 12.

Table 8: First-step VRMSE on PLI under the push-forward sweep, channel-mean over the ten PLI fields.

	full-field		finest ($\ell = 3$)	
K	WAMRViT	ViT-finest	WAMRViT	ViT-finest
1	<u>0.0642</u>	**0.0494**	<u>2.5107</u>	**3.1023**
2	**0.0638**	<u>0.0501</u>	2.5252	<u>3.1863</u>
4	0.0680	0.0593	**2.4921**	3.3610



(a) Full-field VRMSE.

(b) Full-field RMSE.

(c) Finest ($\ell = 3$) VRMSE.

(d) Finest ($\ell = 3$) RMSE.

Figure 12: **Push-forward ablation on PLI.** WAMRViT (solid, blue family) and the ViT-finest baseline (dashed, green family) at $K \in \{1, 2, 4\}$, with light-to-dark shading for $K = 1 \to 4$ within each family. Top row: full-field aggregate. Bottom row: finest refinement level ($\ell = 3$).

At full field, $K = 4$ improves both models at long horizon: at rollout step 20, WAMRViT and ViT-finest reach 0.1925 and 0.2021 VRMSE, respectively. Averaged over all 20 steps at $K = 4$, however, WAMRViT's full-field VRMSE is 7.0% higher than ViT-finest's. At the finest level, WAMRViT is 6.1% below ViT-finest at rollout step 20 for $K = 1$ and 23.2% below it for $K = 4$. The intermediate $K = 2$ adaptive run is unstable at this scope. The effect of push-forward depth depends on the evaluation scope and horizon. At step 20, WAMRViT's finest-level advantage holds for $K = 1$ and $K = 4$, but reverses at $K = 2$.

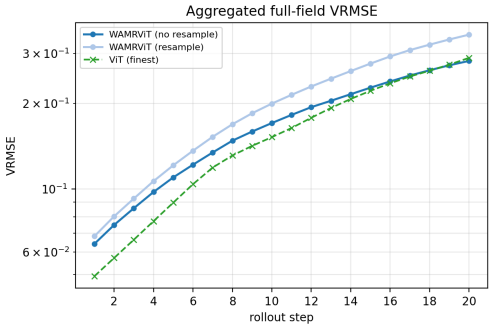
G.4 Random-resample ablation

WAMRViT (uniform tokenization) is trained with and without the random-resample augmentation of Appendix C.8 on both PLI and AMReX combustion, all other hyperparameters held fixed to the values in Table 4.

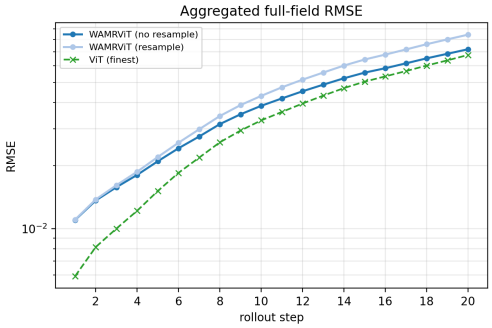
PLI. First-step VRMSE numbers are summarized in Table 9. Aggregated full-field and finest-level rollout curves averaged over all evaluation windows are shown in Figure 13.

Table 9: First-step VRMSE on PLI under the random-resample augmentation, channel-mean over the ten PLI fields.

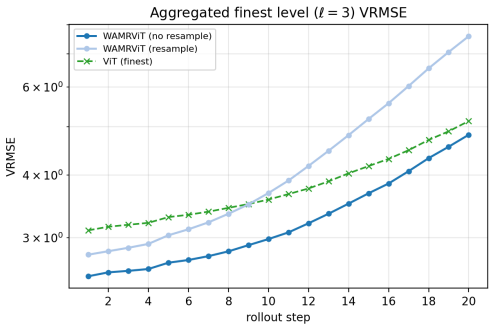
Variant	full-field	finest ($\ell = 3$)
WAMRViT (no resample, default)	**0.0642**	**2.5107**
WAMRViT (with resample)	0.0683	2.7734



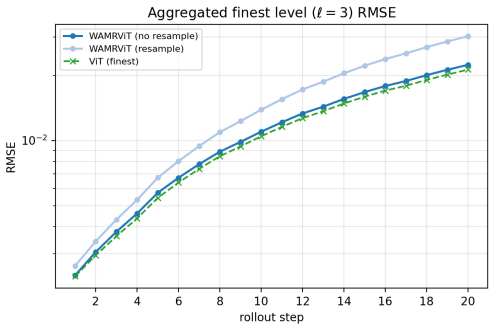
(a) Full-field VRMSE.



(b) Full-field RMSE.



(c) Finest ($\ell = 3$) VRMSE.



(d) Finest ($\ell = 3$) RMSE.

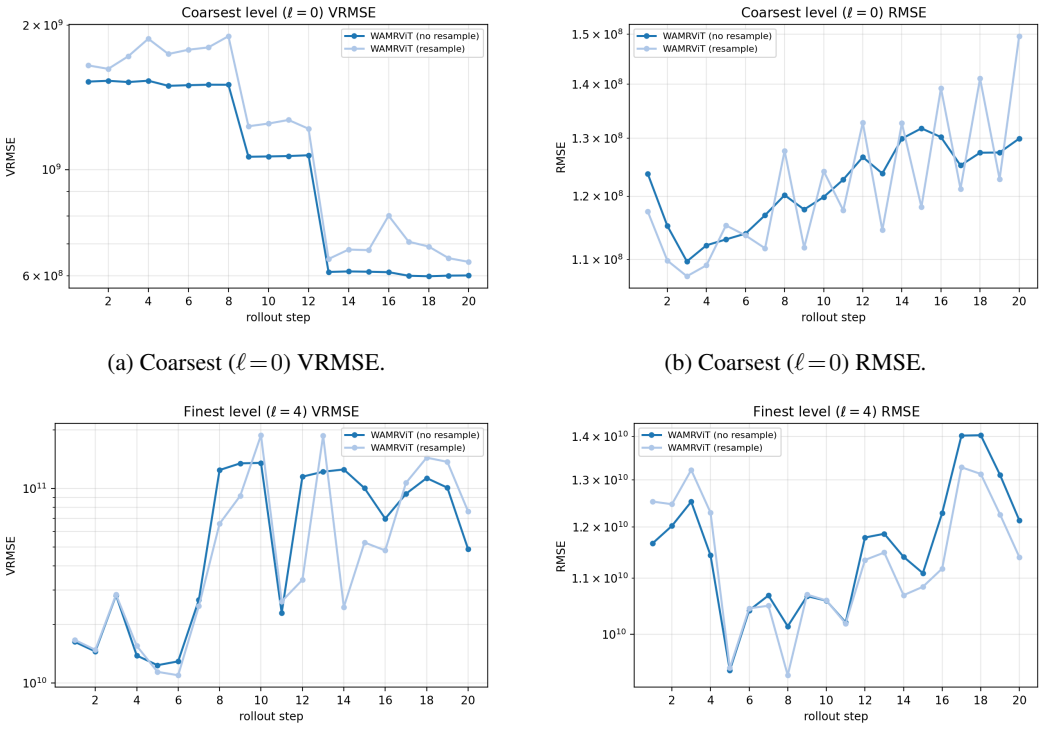
Figure 13: **Random-resample ablation on PLI.** WAMRViT (uniform tokenization) trained without the augmentation (solid, dark blue, the main-text default) vs. with the augmentation of Appendix C.8 (solid, light blue), at $K = 1$, against the ViT-finest baseline (dashed, green). Top row: full-field aggregate. Bottom row: finest refinement level ($\ell = 3$).

On PLI, resampling degrades both scopes. Relative to the default, its first-step VRMSE is 6.4% higher at full field and 10.5% higher at the finest level. Its rollout-mean values are 18.5% and 32.2% higher, respectively. The corresponding RMSE curves give the same ordering. Thus this augmentation does not provide an accuracy gain on PLI at $K = 1$.

AMReX combustion. First-step VRMSE numbers are summarized in Table 10. Coarsest- and finest-level aggregated rollout curves averaged over all evaluation windows are shown in Figure 14.

Table 10: First-step VRMSE on AMReX combustion under the random-resample augmentation, channel-mean over the seventeen combustion fields. AMReX has no full-field row (Section 3.1). Resampling increases coarsest- and finest-level VRMSE by 8.1% and 2.3%, respectively.

Variant	coarsest ($\ell = 0$)	finest ($\ell = 4$)
WAMRViT (no resample, default)	**1.525E9**	**1.631E10**
WAMRViT (with resample)	1.648E9	1.668E10



(a) Coarsest ($\ell = 0$) VRMSE.

(b) Coarsest ($\ell = 0$) RMSE.

(c) Finest ($\ell = 4$) VRMSE.

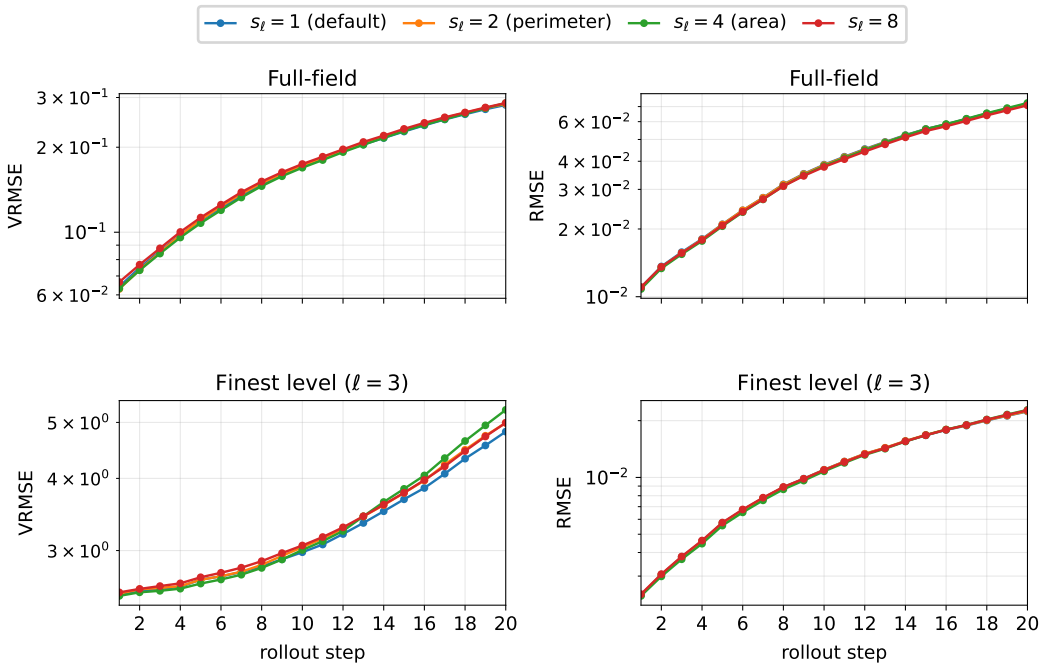
(d) Finest ($\ell = 4$) RMSE.

Figure 14: **Random-resample ablation on AMReX combustion.** WAMRViT (uniform tokenization) trained without the augmentation (solid, dark blue, the no-resampling control) vs. with the augmentation (solid, light blue), at $K = 4$. Top row: coarsest level ($\ell = 0$). Bottom row: finest refinement level ($\ell = 4$).

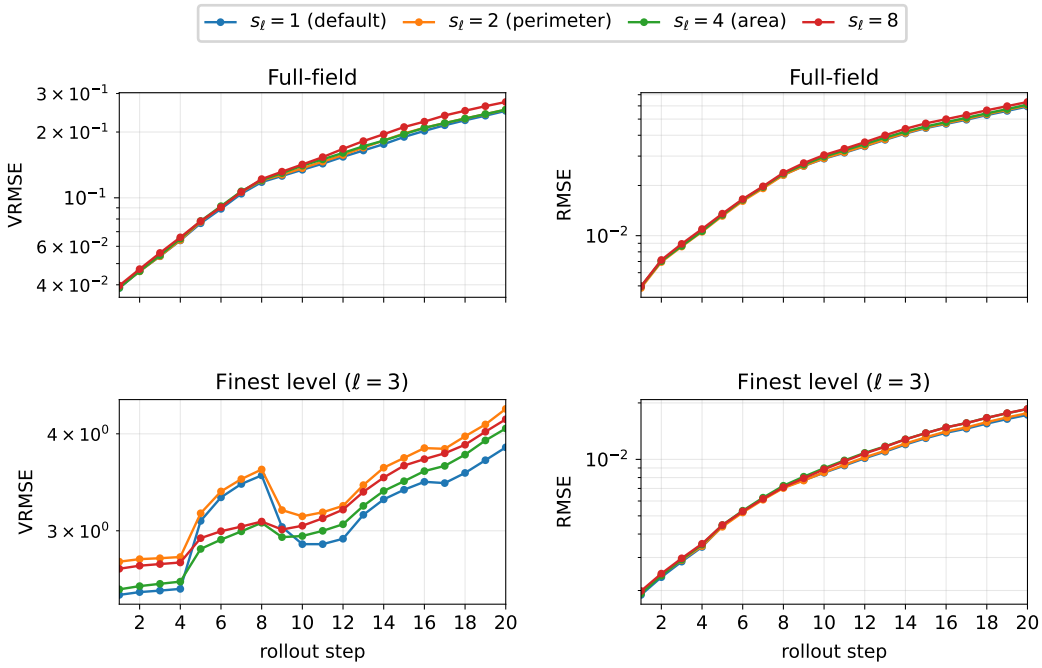
On AMReX, resampling has resolution-, metric-, and horizon-dependent effects. Averaged over all 20 rollout steps, it lowers finest-level VRMSE by 8.8% and RMSE by 1.8%, but raises coarsest-level VRMSE by 15.7%. At the finest level, first-step VRMSE and RMSE increase by 2.3% and 7.4%, respectively. Final-step VRMSE increases by 56.1% even though RMSE decreases by 6.1%. The augmentation trades modest average finest-level gains against worse early predictions and coarsest-level VRMSE.

G.5 RoPE level-axis scale ablation

We compare separately trained models at level-axis RoPE scales $s_\ell \in \{1, 2, 4, 8\}$: uniform-patch WAMRViT on PLI, TRL, and AMReX combustion, and multi-scale WAMRViT on PLI. Topology resampling is disabled, and all other hyperparameters are held fixed within each variant (Table 4). Appendix E.1 gives the geometric interpretation of these scales. First-step VRMSE is reported in Table 3. Figure 15 compares the two PLI variants, and Figure 16 shows the uniform-patch AMReX sweep.



(a) Uniform-patch WAMRViT.



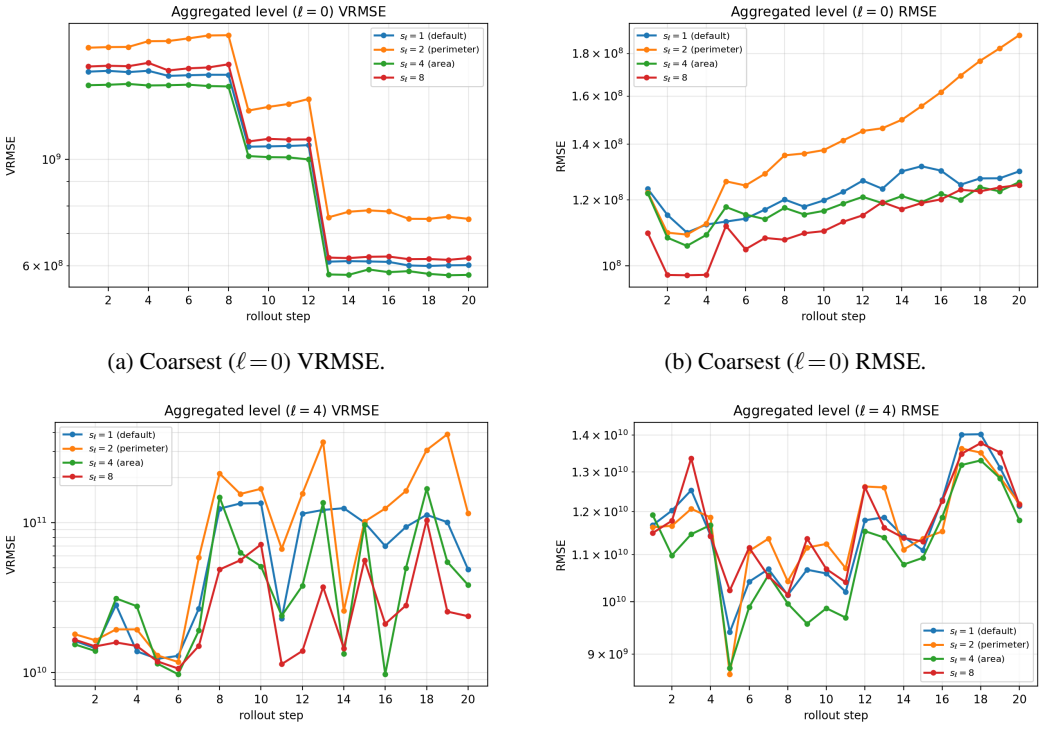
(b) Multi-scale WAMRViT.

Figure 15: **RoPE level-axis scale ablation on PLI.** Single-seed rollouts for (a) uniform-patch and (b) multi-scale WAMRViT at $s_\ell \in \{1, 2, 4, 8\}$, trained without topology resampling. Within each group, rows show full-field (top) and finest-level ($\ell = 3$, bottom) errors; columns show VRMSE (left) and RMSE (right). Colors denote the same scales in both groups. Finest-level metrics use each rollout's evolving topology.

For uniform-patch WAMRViT on PLI (Figure 15a), $s_\ell = 2$ has the lowest first-step full-field VRMSE and $s_\ell = 4$ has the lowest first-step finest-level VRMSE, but the differences from the default are small. Across the rollout, $s_\ell = 4$ has the lowest mean full-field VRMSE, while the default $s_\ell = 1$ has the lowest mean finest-level VRMSE. RMSE produces another ordering. On TRL, $s_\ell = 8$ leads at full field and $s_\ell = 2$ at the finest level in the one-step test (Table 3). These uniform-patch results do not support a monotone effect or a single best scale on PLI. The geometric labels remain useful interpretations of the scale, but the preferred value depends on the evaluation scope and horizon.

Multi-scale WAMRViT on PLI. We additionally train multi-scale WAMRViT at $s_\ell \in \{2, 4, 8\}$, retaining native-resolution patchification, level-conditioned AdaLN, and full-field loss, and compare with the existing $s_\ell = 1$ model under the same fixed-seed protocol. Evaluation uses the same 761 held-out windows over 20 rollout steps. Table 3 reports first-step VRMSE, and Figure 15b shows the rollout curves. Relative to $s_\ell = 1$, scales 2, 4, and 8 increase rollout-mean full-field VRMSE by 2.1%, 2.7%, and 8.1%, and RMSE by 0.6%, 2.2%, and 5.6%, respectively. Scale 2 nevertheless lowers first-step full-field RMSE by 1.7%, illustrating that a short-horizon improvement need not persist over rollout. The default therefore has the lowest rollout-mean full-field errors among the tested scales in this single-seed multi-scale sweep.

The different scale sensitivity of the two variants may partly reflect the additional level conditioning in multi-scale WAMRViT. Its AdaLN layers apply learned level-specific affine transformations throughout the backbone, providing an explicit level signal alongside RoPE and potentially changing sensitivity to the rotary level scale.



(a) Coarsest ($\ell = 0$) VRMSE.

(b) Coarsest ($\ell = 0$) RMSE.

(c) Finest ($\ell = 4$) VRMSE.

(d) Finest ($\ell = 4$) RMSE.

Figure 16: **RoPE level-axis scale ablation on AMReX combustion.** WAMRViT rollouts at $s_\ell \in \{1, 2, 4, 8\}$. Top row: coarsest level ($\ell = 0$). Bottom row: finest level ($\ell = 4$). Full-field materialization is omitted.

On AMReX, $s_\ell = 4$ has the lowest first-step VRMSE at both coarsest and finest levels and the lowest rollout-mean coarsest-level VRMSE. Under RMSE, $s_\ell = 8$ is lowest at the first step on both levels and in the coarsest-level rollout mean. At the finest level, $s_\ell = 4$ also has the lowest rollout-mean RMSE, reducing mean VRMSE and RMSE by 28.6% and 4.1% relative to $s_\ell = 1$. Both metrics improve in all seventeen fields when averaged over the rollout. Scale $s_\ell = 8$ has the lowest

rollout-mean finest-level VRMSE, a 57.2% reduction, but its RMSE is 1.4% higher than the default. Scale $s_\ell = 2$ raises the corresponding means by 74.1% and 0.7%, respectively. HRR dominates the channel-mean finest-level VRMSE and causes sharp step-to-step changes, so the per-field curves remain necessary for interpretation. Together with PLI and TRL, this sweep shows that level scale is configuration-, scope-, metric-, and horizon-dependent. No single geometric choice is uniformly preferred.

G.6 Mesh-policy sensitivity on PLI

We evaluate uniform-patch and multi-scale WAMRViT with fixed checkpoints on the same held-out PLI trajectories, forecasting 20 steps with regridding every $R = 4$ steps. The default uses a neighborhood radius $\delta = 2$, initial tolerance 0.01, and runtime tolerances $[0.1, 0.1, 0.3]$ for the two velocity components and density. We vary $\delta \in \{1, 3\}$ at fixed tolerances, or scale both initial and runtime tolerances by 0.5 or 1.5 at $\delta = 2$. Each change applies to initial mesh construction and subsequent regridding. Model weights are unchanged.

Table 11 compares full-field predictions reconstructed onto the same finest physical grid against the original uniform-grid ground truth, rather than evaluating each model's own refined regions. Errors are averaged over the ten PLI fields and all 20 rollout steps.

Table 11: **PLI mesh-policy sensitivity.** Full-field rollout-mean errors and token counts for the fixed checkpoints. Token counts $\bar{N}$ are averaged over the same evaluation windows and rollout steps.

	WAMRViT (uniform)			WAMRViT (multi-scale)		
Mesh policy	$\bar{N}$	RMSE	VRMSE	$\bar{N}$	RMSE	VRMSE
Default	3,324	**0.0403**	**0.1749**	3,336	**0.0306**	**0.1405**
Radius $\delta = 1$	2,717	0.0486	0.2238	2,739	0.0414	0.1950
Radius $\delta = 3$	3,981	0.0427	0.1836	3,998	0.0329	0.1476
Tolerances $\times 0.5$	3,745	0.0408	0.1780	3,717	0.0316	0.1435
Tolerances $\times 1.5$	3,269	<u>0.0404</u>	<u>0.1753</u>	3,299	<u>0.0308</u>	<u>0.1428</u>

The default has the lowest rollout-mean RMSE and VRMSE among the tested settings for both models. Both models were trained only with the default initial tolerance and neighborhood radius, so the degradation under these perturbations is consistent with a training-to-test mesh-distribution shift. Reducing δ to 1 removes about 18% of tokens but increases mean VRMSE by 28.0% for uniform-patch WAMRViT and 38.8% for the multi-scale variant. RMSE increases by 20.6% and 35.2%, respectively. Increasing δ to 3 adds about 20% more tokens but also raises both errors. Thus, denser refinement alone does not guarantee improved forecasting with fixed weights.

Tolerance scaling changes either rollout-mean metric by at most 3.3% across the two models, although individual fields can be more sensitive: halving the tolerances raises multi-scale case-material volume-fraction RMSE by 16.3%. Multi-scale WAMRViT retains lower channel-mean full-field RMSE and VRMSE than uniform-patch WAMRViT at every rollout step under each matched policy. The aggregate sensitivity is therefore stronger for the tested neighborhood changes than for tolerance scaling, without reversing the ordering of the two variants.

G.7 Per-field rollout curves on PLI

The PLI panels of Figure 4 aggregate across physical channels (mean over the ten PLI fields). The figures below show the same rollout broken out by field. All are at push-forward $K = 1$, averaged over all evaluation windows. Adaptive tracks are solid and uniform-grid baselines dashed (uniform-grid predictions are block-mean-projected onto the adaptive quadtree for the per-level figures). For readability, we show four field groups: a velocity component (U-velocity), bulk density, summed material energy, and the arithmetic mean of the six per-channel volume-of-fluid error curves. The three scopes match the aggregated figures (full-field, coarsest, finest). Intermediate levels are omitted.

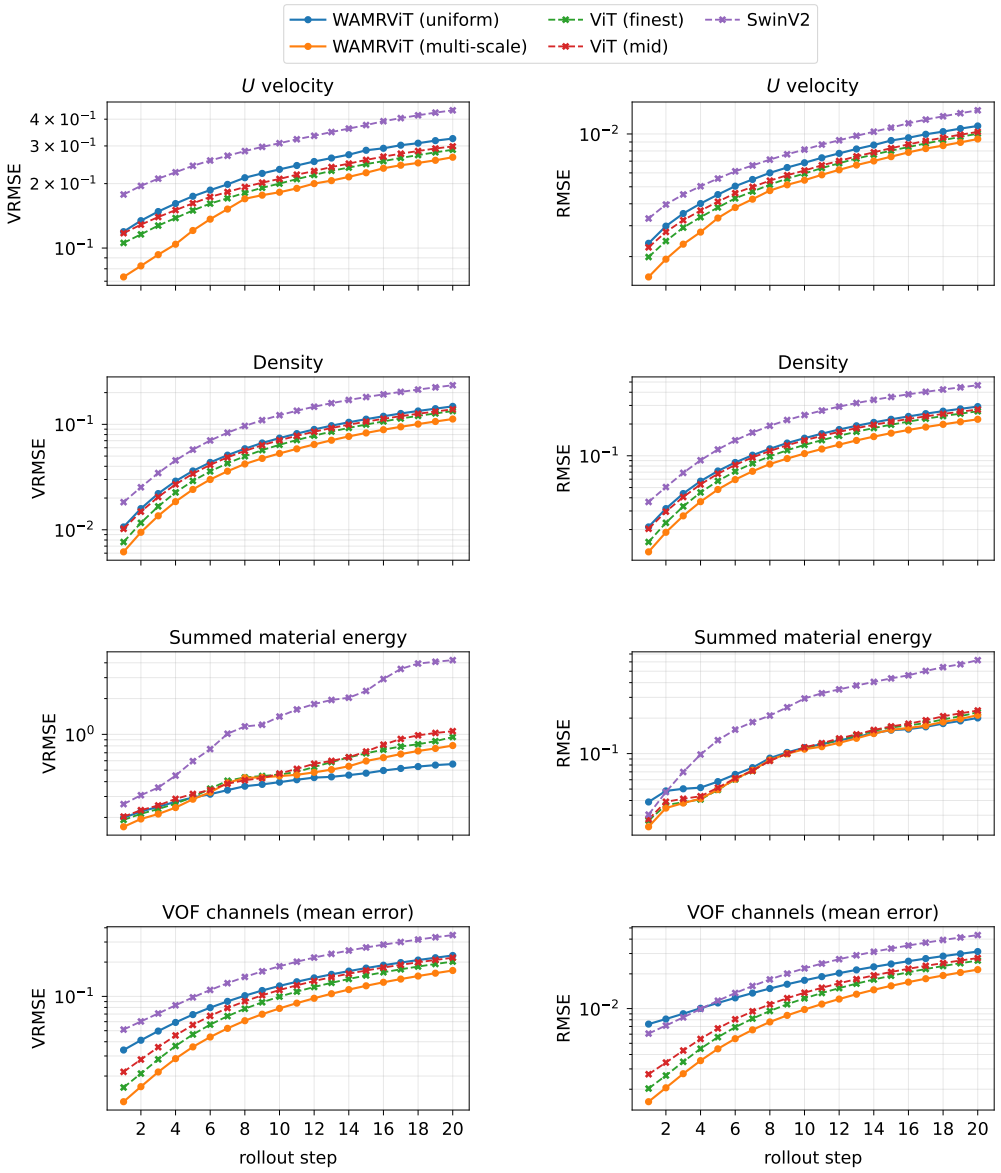


Figure 17: **PLI full-field errors**, $K = 1$. Rows show the four field groups described above. Columns show VRMSE (left) and RMSE (right).

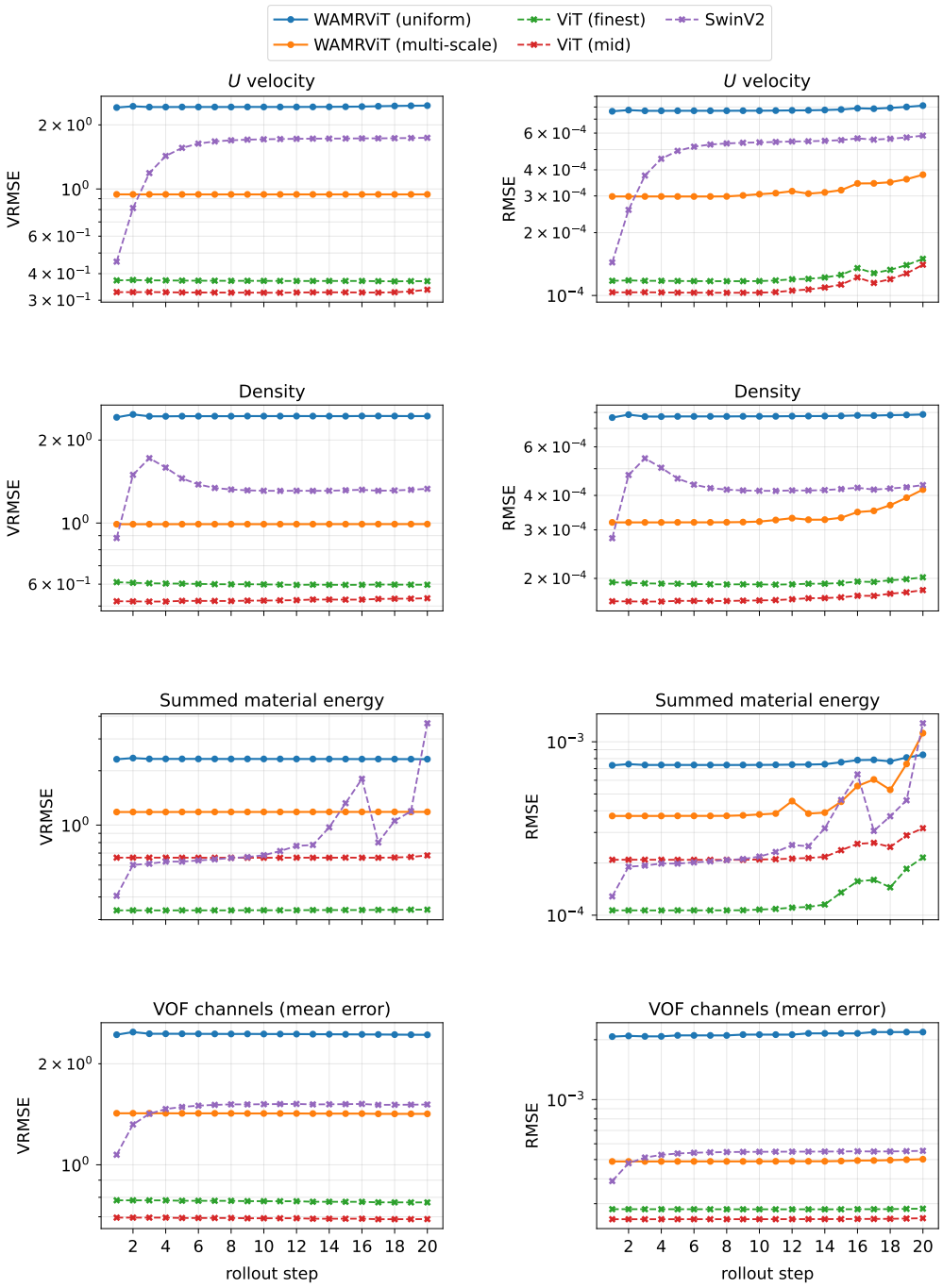


Figure 18: **PLI coarsest-level errors** ($\ell = 0$), $K = 1$. Rows show the same four field groups. Columns show VRMSE (left) and RMSE (right).

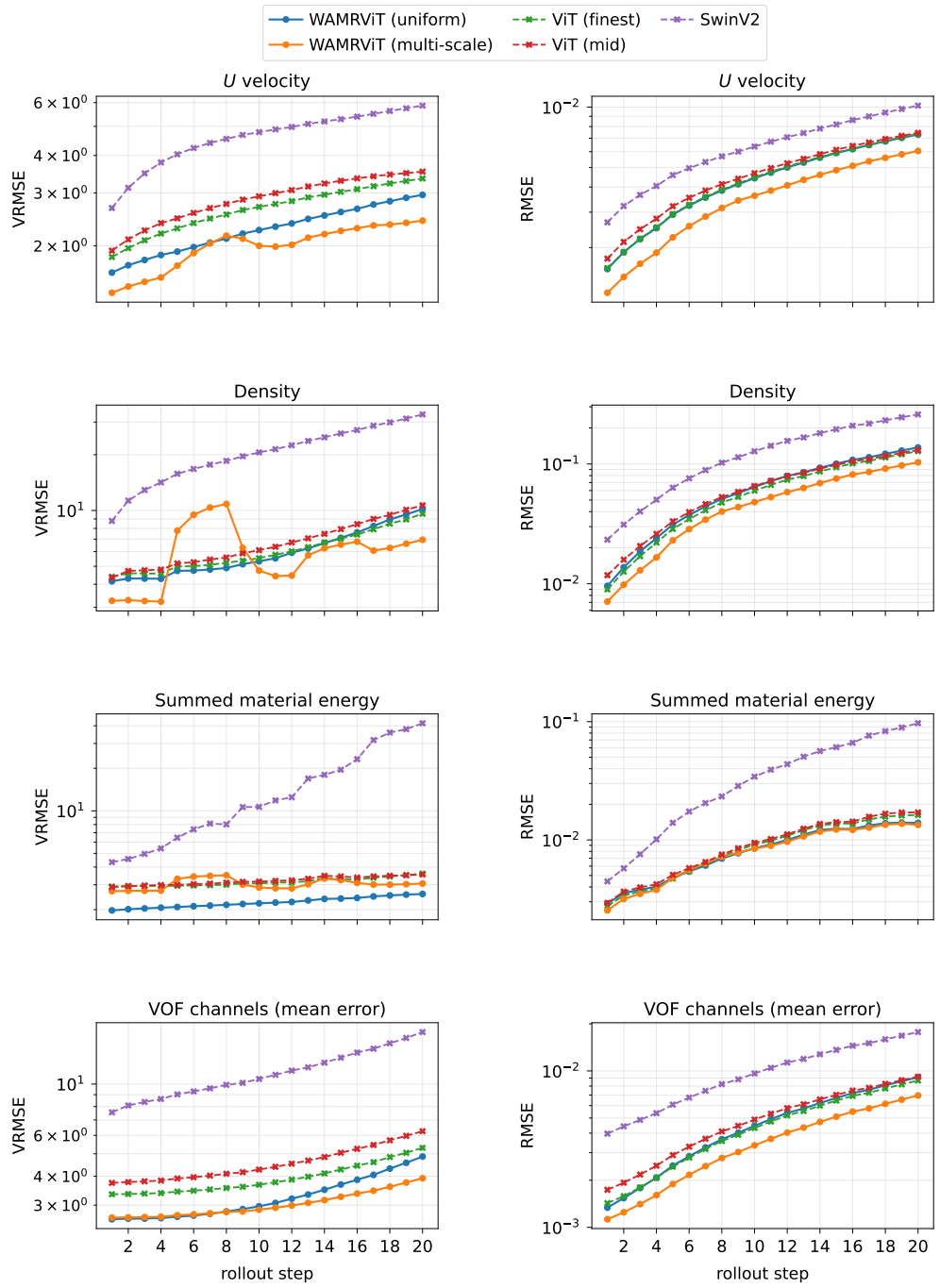


Figure 19: **PLI finest-level errors** ($\ell = 3$), $K = 1$. Rows show the same four field groups. Columns show VRMSE (left) and RMSE (right).

G.8 Aggregated rollout curves on AMReX combustion

All panels use checkpoints trained at $K = 4$, with the three-way legend of the AMReX panels in Figure 4: WAMRViT (uniform), ViT (mid), and SwinV2. The $p = 32$ regular controls use the feasible intermediate projection rather than a full-domain uniform-grid representation at the native finest resolution. Intermediate levels $\ell \in \{1, 2, 3\}$ are omitted for readability.

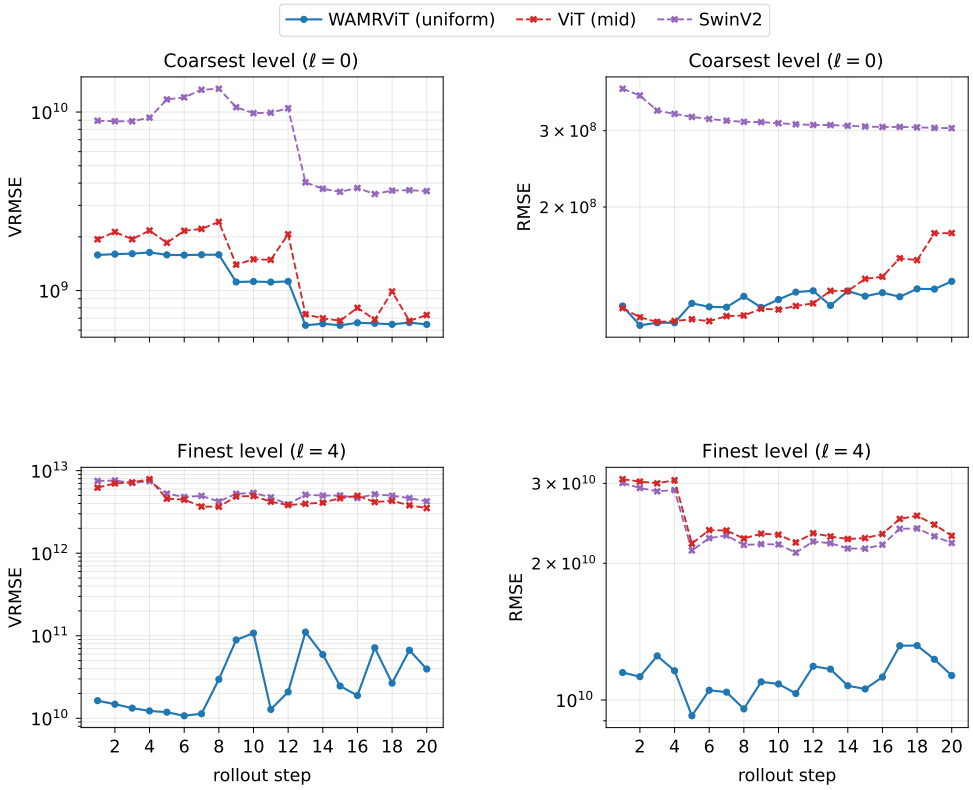


Figure 20: **Aggregated AMReX combustion errors**, $K = 4$. Rows show the coarsest ($\ell = 0$) and finest ($\ell = 4$) levels. Columns show VRMSE (left) and RMSE (right), averaged over the 17 channels. As on PLI, the regular-grid predictions are block-mean-downsampled onto the adaptive coarsest-level leaves and can benefit from error-cancelling averages at this scope. The finest row corresponds to the AMReX panels of Figure 4.

G.9 Per-field rollout curves on AMReX combustion

The AMReX panels of Figure 4 aggregate across seventeen combustion channels. The figures below separate six representative fields: heat-release rate (HRR), temperature T, Mach number, pressure, density ρ, and total energy density ρE. They use the same WAMRViT, ViT-mid, and SwinV2 legend and the coarsest ($\ell = 0$) and finest ($\ell = 4$) scopes. Density-weighted species mass fractions are omitted for readability. All panels use checkpoints trained at $K = 4$.

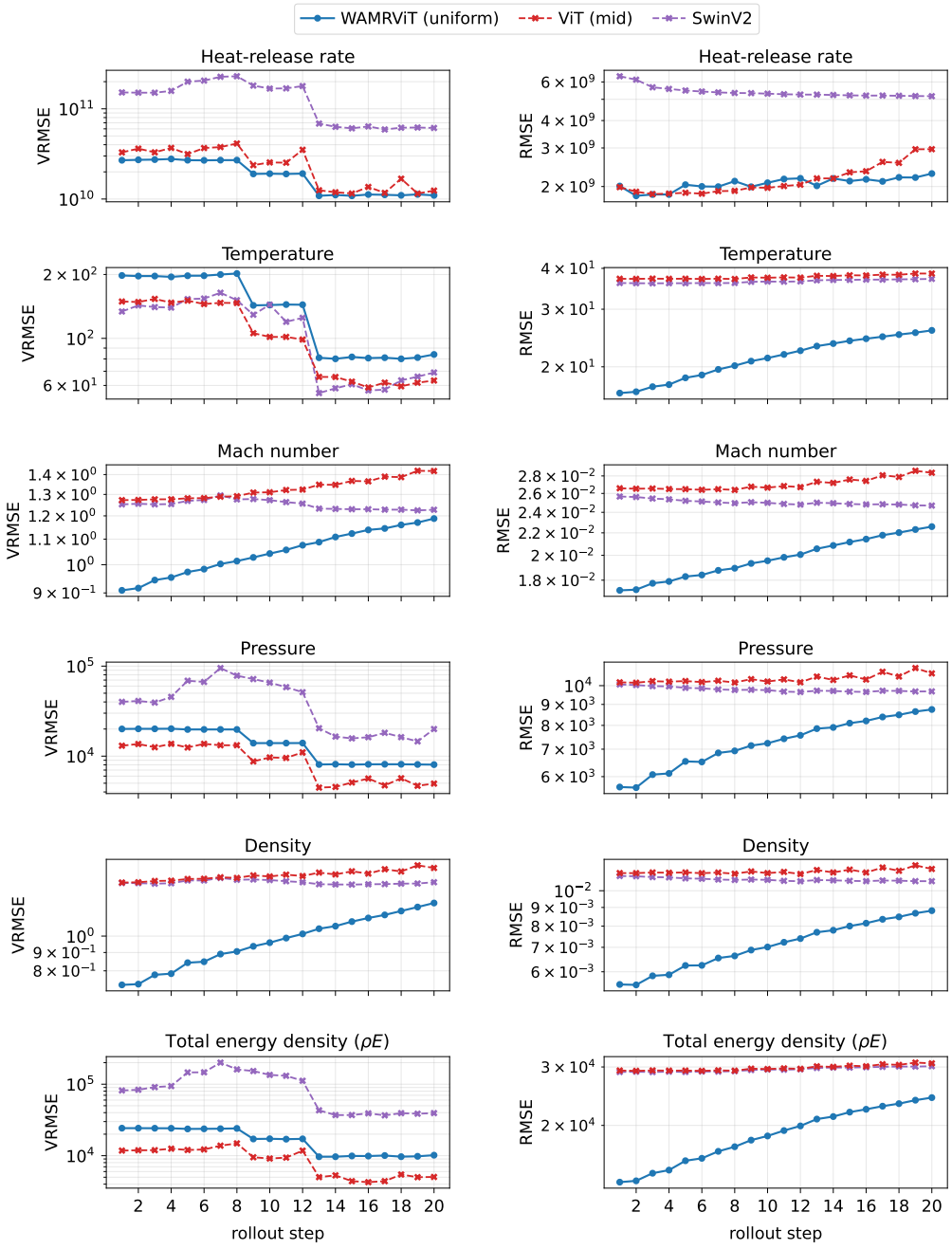


Figure 21: **AMReX coarsest-level errors** ($\ell = 0$), $K = 4$. Each row corresponds to one of the six representative fields. Columns show VRMSE (left) and RMSE (right).

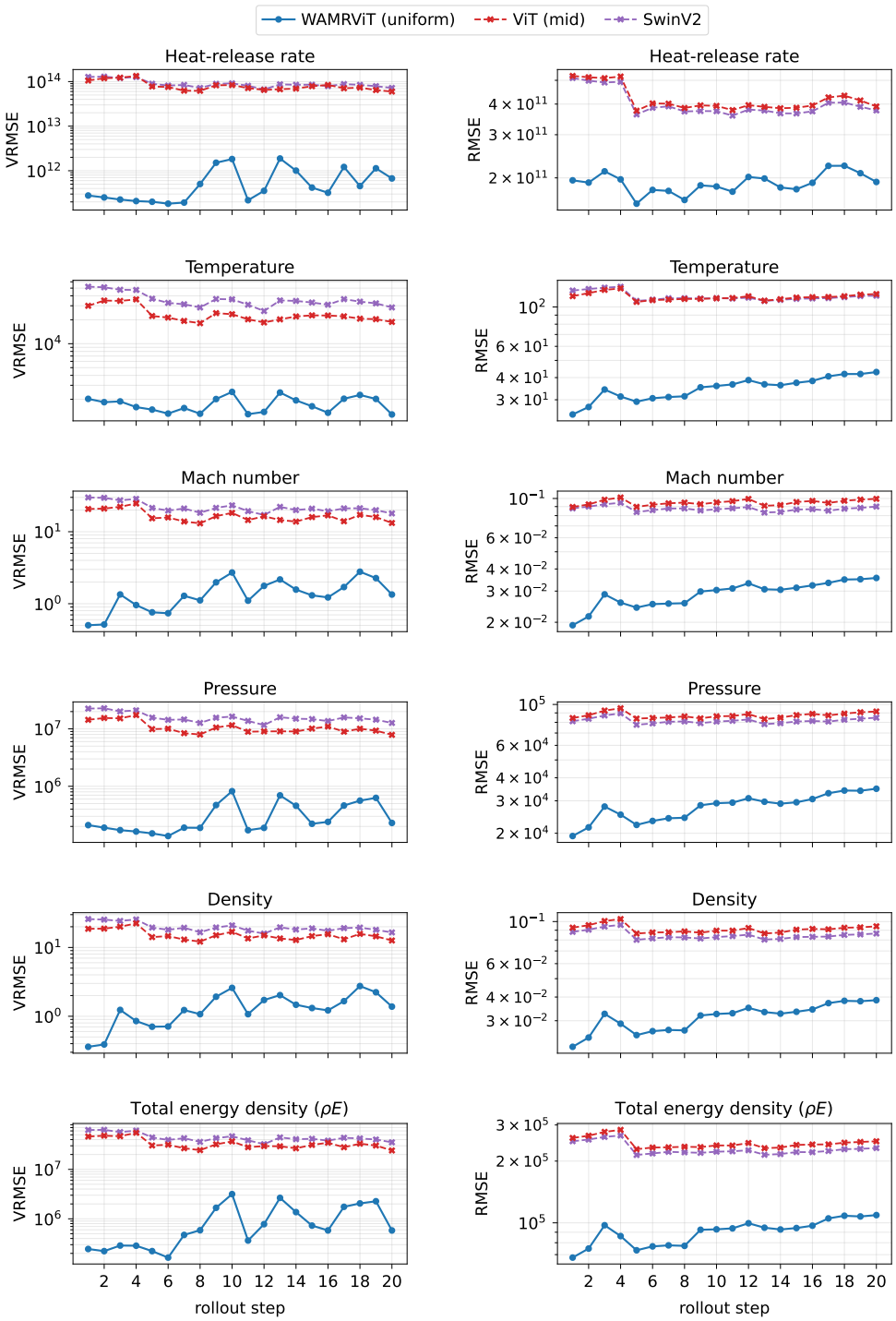


Figure 22: **AMReX finest-level errors** ($\ell = 4$), $K = 4$. Rows show the same six fields. Columns show VRMSE (left) and RMSE (right).

H Extended Related Work

Adaptive patching for ViT-based PDE surrogates. AMR-Transformer [9] decides splits via four velocity-field criteria. Two image-domain methods use hand-crafted scores: the Adaptive Patch Framework (APF) [10], for high-resolution segmentation, uses a Canny edge detector, and APT [11] scores patches by entropy. MATEY [12] is the closest in spirit: it is explicitly AMR-inspired, computes a per-patch variance, refines patches above a global threshold to a single finer size, uses two resolution levels in its reported experiments, positions itself as a multiscale adaptive transformer for spatiotemporal PDE forecasting, and handles variable token counts via padding masks. Iterative Adaptive Patchification [IAP; 13] fixes a target token count and iteratively splits the top-scoring patches under a Sobel-based gradient score until the budget is reached. Overtone [15] does not choose where to refine. It trains a single ViT to run at any of a discrete patch-size set $\{4, 8, 16\}$ on uniform-grid inputs via convolutional kernel/stride modulation, cycling sizes across rollout steps to disrupt grid-aligned artifacts of fixed-patch tokenization. The partition itself remains spatially uniform within any forward pass and the per-step size is set externally. These approaches motivate adaptive patching on uniform-grid data. WAMRViT combines four capabilities: (i) tokens carry an explicit 2:1-balanced quadtree hierarchy with level-aware 3D RoPE and mean-conservative coarsening, in contrast to recursive trees without a balance constraint (AMR-Transformer, APF, IAP) or MATEY's evaluated two-level partition; (ii) variable token counts are handled by micro-batch-one training with data-parallel replication (Section 3.3) rather than by padding masks or a fixed token budget; (iii) we evaluate changing-topology 20-step rollouts on PLI and AMReX; and (iv) we forecast directly on native multi-level AMReX data. To our knowledge, the novelty is the integration of variable-token prediction, repeated balanced-topology updates, and state transfer across those updates in one ML-based AMR surrogate, rather than any one component in isolation.

Mesh-free neural surrogates. Mesh-free methods represent the solution as a coordinate MLP $u_\theta(x, t)$ trained against PDE residuals or a variational principle. Physics-Informed Neural Networks [1] and the Deep Galerkin Method [35] are canonical, with the DeepXDE library [36] consolidating these formulations and introducing residual-based adaptive refinement that densifies collocation points where the PDE residual is large. This concentrates compute in spirit similar to our wavelet-driven token refinement, but applied to where the optimizer evaluates the loss rather than to where the model places tokens. The methodological gap with our setting is consequential. Mesh-free methods typically solve a separate optimization per initial condition, amortizing only over conditions supplied as network inputs, and the long autoregressive rollouts we report are uncommon in this family because temporal causality must be enforced through the loss rather than through model architecture, so per-instance training cost compounds with rollout horizon. WAMRViT instead learns a single autoregressive operator amortized across trajectories, concentrates compute via adaptive tokenization rather than collocation density, and maintains a moving topology over the rollout via inference-time regridding.

Neural operators and PDE ViTs. FNO [3], DeepONet [2], UPT [4], and CViT [37] provide the broader context in which our tokenization sits. FNO and CViT consume a single uniform-resolution input field at inference time, with any hierarchical or multi-resolution structure internal to the operator (e.g. in the kernel representation of FNO) rather than expressed at the discretization. CViT encodes gridded inputs using fixed spatial patches, while its cross-attention decoder predicts at queried coordinates. Both WAMRViT variants use a spatially adaptive partition whose leaf areas can differ by factors of up to 4^L within one sample ($L = 3$ on PLI and $L = 4$ for uniform-patch AMReX; Table 4). The transformer processes these leaves as one sequence under level-aware 3D RoPE rather than a multi-resolution cascade. The variant named "multi-scale" additionally retains each leaf's full source-resolution raster from a uniform-grid source. Graph-based and point-cloud operators such as UPT reach discretization-level flexibility on unstructured meshes. We instead use a 2:1-balanced quadtree, constructed from uniform-grid fields or initialized from a simulator hierarchy and subsequently adjusted as described in Appendix C.6. This structure provides explicit face-neighbor relations and mean-conservative merging of sibling rasters. A separate line of work embeds learned operators inside classical FE solvers rather than replacing them. NOEM [38], building on earlier FE-NO coupling [39], pre-trains neural operators to approximate boundary-to-solution maps on user-chosen subdomains and reconciles FE and NO degrees of freedom by energy minimization at inference. These methods use static partitions, and NOEM targets steady variational problems, in contrast to our dynamic wavelet refinement over autoregressive rollouts.

I Future work

Throughput in the variable-N regime admits several concrete follow-ups: padded micro-batching with zero-attention masks (in the spirit of MATEY [12]), length-sorted bucketing, block-diagonal flex-attention packing, and dynamic-shape GPU kernels each address a specific bottleneck. The current backend computes topology updates with packed, parallel CPU kernels and, for AMReX, applies the resulting split/merge records to leaf values kept on the GPU (Section 2.4). Moving neighbor lookup and balancing onto the GPU would remove the remaining synchronization and topology-transfer cost. The topology-cache machinery of Appendix C.9 composes with these changes. A native multi-scale objective that avoids materializing the finest uniform grid would also make the multi-scale variant applicable to simulator-native AMR data (Appendix F.2). A learned, multi-field generalization of the current non-learned wavelet score could also replace the global ε and per-channel union with a content-aware detector. Its accuracy or token reduction would need to outweigh added inference and CPU–GPU synchronization costs.

WAMRViT factors cleanly into an adaptive quadtree tokenizer and a transformer backbone joined by a standard token interface, so the backbone is interchangeable without altering the tokenizer or the inference-time regrid. In large-patch regimes such as AMReX combustion ($p_h{=}p_w{=}32$), where the convolutional patch embedder carries a disproportionate share of the model's representational capacity, two spectral substitutions warrant investigation: an FNO-style spectral patch embedder [3] in place of the convolutional one, and AFNO-style Fourier token mixing [40] in place of self-attention. Either substitution leaves the tokenizer and rollout machinery untouched, isolating the architectural change. A further extension is a controllable variant of WAMRViT in the spirit of Mukhopadhyay et al. [15], exposing the wavelet refinement tolerance as an explicit compute-budget control distinct from the patch-size selector they study at inference.

The level-scale ablation (Appendix G.5) shows neither a monotone trend nor a dataset-independent optimum, suggesting that a single hand-set s_ℓ is a restrictive parametrization of refinement level. Learning the level scale jointly with the model, or encoding level differences through a learned scale-aware attention bias, could remove the need to tune it per dataset.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the contributions (wavelet-driven adaptive tokenization, the level-aware 3D RoPE, and the multi-scale native variant), which are substantiated by the experiments in Section 3 and the qualitative rollouts in Appendix A; caveats are stated in Section 4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

Justification: Section 4 discusses micro-batch-one adaptive training, measured regridding overhead and the AMReX accuracy–cost trade-off, configuration sensitivity, and the two-dimensional single-trajectory evidence scope; concrete follow-ups appear in Appendix I.

Guidelines:

- The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not include theoretical results; the contributions are architectural and empirical.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

Justification: The architecture is described in Section 2 and Appendix D; training hyperparameters, optimizer settings, and per-dataset configurations are given in Appendix F; datasets and preprocessing in Appendix B; and evaluation metrics in Appendix F.3. The code is publicly available at `https://github.com/tonyzyl/wamrvit`.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The code is available at `https://github.com/tonyzyl/wamrvit`, and the PLI and TRL datasets are publicly available. The AMReX combustion data are available only by request to the original simulation authors, so the complete data collection is not openly downloadable.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.

- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Appendix F specifies the optimizer, learning-rate schedule, gradient clipping, EMA, batch size, push-forward depth K, and per-dataset hyperparameters used for every reported run; data splits and the rationale for the adaptive hyperparameter choices are stated in Section 3 and Appendix B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Table 1 reports the mean and sample standard deviation across three independently trained seeds for PLI, TRL, and AMReX, covering the main accuracy claims. Table 2 reports variability across ten matched starting windows for the efficiency claims. Ablations use single runs per setting (a fixed seed on PLI/AMReX; the TRL archive does not record the seed) and are not used as multi-seed significance claims.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix F reports the training hardware and configurations; Table 2 reports training durations and inference wall-clock breakdowns, with the timing protocol in Appendix F.2.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work studies neural surrogates for fluid-dynamics simulations using publicly available physics datasets; it does not involve human subjects, personal data, or other ethically sensitive material, and the authors have reviewed the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: The work is foundational research on neural surrogates for compressible-flow and radiation-hydrodynamics simulations; the artifact is not generative and the offline scientific-simulation deployment domain has no direct path to negative societal impact beyond generic risks of any predictive ML model, so we do not include a separate societal-impact discussion.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to

generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The released artifact is a deterministic neural surrogate for physical simulations; it is not a generative or dual-use model and uses publicly released physics benchmarks rather than scraped data, so no high-risk-misuse safeguards apply.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The three datasets used are cited with their original sources: PLI from the HEAT benchmark [21], TRL from The Well [22], and AMReX combustion following Ullman et al. [23] stored in AMReX [27] format. The overall pipeline is built upon Ray [29].

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The implementation accompanying the paper is publicly available at `https://github.com/tonyzyl/wamrvit`.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve research with human subjects, so IRB review does not apply.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLM was used for non-core software-engineering tasks: visualization scripting, implementing standard methods, and for writing, editing and formatting assistance.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.