PARA: PATCH-BASED ADAPTIVE MULTI-RESOLUTION, TIME-ADVANCING AND SCALE-EQUIVARIANT NEURAL PDE SOLVER

Anonymous authors
Paper under double-blind review

ABSTRACT

We present **PARA**, a simple and yet effective unified framework for advancing *multi-resolution* states of hyperbolic PDEs. By representing the spatial solution as a tree of independent local patches at varying resolutions (TreeGrid), PARA shifts structural complexity out of the neural network. The future state of each patch is predicted concurrently using a lightweight network from its *localized neighborhood* (stencil). Restricting the network to the stencil of a patch gives it an inductive bias towards locality while errors fall and flatten once the stencil captures the physical domain of dependence, in contrast to whole-frame models which capture the entire domain yet yield higher errors. Crucially, we introduce *scale normalization*, which uses the scaling symmetry of the Euler equations to make the solver scale-equivariant by construction, so a single network learns the solution map in dimensionless units. By expanding the TreeGrid into a hierarchy of dense grids, one per resolution level (MultiGrid), PARA cuts every patch with its halo into a stencil of one shape, so all levels advance in a single batched tensor operation. On the Well's challenging compressible Euler benchmark, our symmetry-driven, patch-based approach achieves exceptional accuracy and generalization across spatial resolutions, outperforming whole-frame single-resolution SOTA neural solvers in one-step accuracy and across long-horizon autoregressive rollouts under identical capacity restrictions.

1 INTRODUCTION

Numerical simulations of Partial Differential Equations (PDEs) underpin foundational domains in engineering and science across a wide range of scales, from aerodynamics to climate modeling and astrophysics. While classical numerical solvers provide robust convergence guarantees, their massive computational cost at fine grid resolutions limits their utility in real-time or outer-loop optimization workflows. Consequently, as deep learning spread through the sciences (LeCun et al., 2015; Brunton et al., 2020), data-driven techniques such as Fourier Neural Operators (FNOs) (Li et al., 2021) and vision-transformer variants (Herde et al., 2024; Hao et al., 2024), have emerged to bypass the limitations of traditional solvers.

Despite their rapid advancement, these solvers take the whole frame as input and encounter a structural mismatch when modeling hyperbolic conservation laws, such as the Euler equations of inviscid compressible flow. Information in a hyperbolic system travels along characteristics at finite speed, so the state of a region one step ahead depends only on a bounded neighborhood, its domain of dependence. A whole-frame solver nevertheless takes the entire frame as input, of which the relevant neighborhood of each local output is a small and state-dependent fraction. The network must therefore learn two tasks at once, the state-dependent filtering of the relevant input and the solution map itself, at a cost that scales with the resolution of the whole domain.

A second mismatch concerns resolution. Physical systems carry features across orders of magnitude in scale, sharp shocks besides smooth expansions, and these features are not uniformly distributed. A single tensor representation of the solution must either be as fine everywhere as its finest feature or under-resolve it, so the cost of a solution is set by the size of the domain rather than by the features it contains. Adaptive multi-resolution neural solvers address this by advancing evolving meshes

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

with graph networks (Pfaff et al., 2021; Perera & Agrawal, 2024) or by tokenizing transient tree views of a uniform array (Xu et al., 2025; Zhao et al., 2026), but they couple the multi-resolution representation to the network, so the network must be designed around that representation.

A third mismatch concerns scale itself. The Euler equations contain no dimensional constant, so the same physics occurs at every physical scale, and the scaling symmetry of the equations states this exactly. A neural solver normalized by dataset statistics is nevertheless tied to the dataset that set them, and it must learn from data that a rescaled state behaves as the original, an inductive bias the equations already give for free.

In this paper, we present **PARA (Patch-based Adaptive multi-Resolution time-Advancing scale-equivariant neural PDE solver)**, a framework that addresses the three mismatches with three design decisions. First, each region is predicted from a bounded neighborhood, a patch and its halo, whose width sets the share of patches whose domain of dependence lies inside their stencil, so the network has less state-dependent filtering to learn alongside the solution map. Second, the solution is stored as a TreeGrid, a quadtree of patches that resolves each region as finely as its own features require, so the cost of a step follows the features rather than the size of the domain. Third, the network learns in dimensionless units. Scale normalization takes its reference quantities from each stencil and conditions the network on the one dimensionless number that remains, replacing the many-to-one correspondence between physical scale and physical behavior with a one-to-one correspondence between dimensionless state and behavior, so the solver is scale-equivariant by construction. In summary, the contributions of PARA are the following.

Locality. Every patch is advanced from its stencil, the patch and its halo, and a conditioning input. The halo width study of Section 4.2 shows the error falls as the share of patches whose domain of dependence lies inside the stencil rises, and flattens once most patches are contained. Extending the stencil to the whole frame raises the error again, higher than at any halo width from 8 to 32 cells.

Adaptive multi-resolution. A MultiGrid expands the TreeGrid so that every patch at every level is cut with its halo into a stencil of one shape and advanced by one network in one batch. The adaptive rollout reduces the cost of a step far more than it raises the error, and the user sets this trade-off at inference time through the refinement thresholds, without retraining (Section 3.4).

Scale equivariance. Scale normalization makes the solver equivariant under the three scale symmetries of the Euler equations, in density, time and length, by construction. Table 4 confirms the density symmetry, with the same VRMSE when every conserved field is scaled by 0.1 or 10, where statistical normalization degrades twenty and five times.

PARA is a unified framework because its three design decisions reinforce one another through the stencil. The stencil bounds the input of the network, has one shape at every level, and supplies the reference quantities of scale normalization. On the Well's compressible Euler benchmark, PARA lowers the one-step VRMSE of all four compatible local and convolutional/attention backbones, namely, U-Net, CNextU-Net, Flower and scOT, by factors of 1.2 to 2.9 against the same networks trained whole-frame on the same data.

At first glance it might suggest that PARA merely combines classical adaptive mesh refinement (AMR) with localized neural solver. Crucially, our novelty lies in the structural synergy of these components: by mapping localized patch halos to a MultiGrid, we decouple the multi-resolution representation from the neural network weights. This allows standard deep learning backbones to natively inherit a strict numerical domain-of-dependence bias and exact, data-free physical scale equivariance by construction—properties that prior whole-frame or coupled graph architectures must waste model capacity to approximate.

2 Prior work

PARA introduces a fundamentally decoupled approach that unifies locality, adaptivity, and symmetry through the lens of a single local stencil. While existing adaptive neural operators pack multi-resolution tracking inside modified network layers—requiring specialized graph or tokenizer backbones—PARA isolates this structural complexity entirely within the grid mechanics. This allows standard, unchanged deep learning architectures to operate on uniform localized patch profiles concurrently. Furthermore, unlike prior local solvers that retain global branches, or symmetry meth-

ods that rely on loss penalties or data augmentation, PARA restricts the network's input to a bounded stencil and, with scale normalization, is scale-equivariant by construction. The stencil bounds the input, which reduces the state-dependent filtering the network must learn. It has one shape at every level, so one network advances the patches of a multi-resolution TreeGrid. It supplies the reference quantities of scale normalization, so the network learns in dimensionless units.

Locality. Over one step a hyperbolic solution depends only on its domain of dependence (Ye et al., 2024; Cheng et al., 2025; Ovadia & Turkel, 2026), while a whole-frame solver lets every output cell depend on the entire frame, as (Zhang et al., 2026b) note. Global receptive fields are motivated by elliptic or incompressible coupling (Li et al., 2020; Wu et al., 2026; Kumar et al., 2026; Sharpe, 2025). Prior solvers add local layers beside or after a global FNO or attention backbone (Liu-Schiaffini et al., 2024; Kalimuthu et al., 2025; Wang et al., 2026; Ovadia & Turkel, 2026; Wu et al., 2026), or predict a cell or node, or the stencil weights that form it, from a local neighborhood (Bar-Sinai et al., 2019; Maddu et al., 2021; Brandstetter et al., 2022b; Zhang et al., 2026a), in some cases with the stencil radius set by the transport of one step (Lan et al., 2026; Zhang et al., 2026b). The local neural operator predicts a patch from the patch and a margin (Li et al., 2024), and Ye et al. (2024) find that a receptive range too small to cover the range of dependence may cause oscillation, while one too large loses accuracy. The hybrid solvers keep the global dependence through their global branch, and the only study of the neighborhood size against the domain of dependence (Ye et al., 2024) uses a uniform grid of smooth flows without a global baseline. PARA addresses both, with no global branch and a halo width tested against both the domain of dependence and the whole frame on flows with shocks.

Adaptive multi-resolution. Adaptive mesh refinement concentrates the cost where the features are (Berger & Oliger, 1984; Berger & Colella, 1989), and tree-based AMR keeps the patches 2:1 balanced (Abdi et al., 2024). Neural solvers handle several scales in three ways. U-Nets (Ronneberger et al., 2015; Gupta & Brandstetter, 2023), multigrid and multipole operators (Li et al., 2020; 2026; Kumar et al., 2026) and hierarchical transformers (Herde et al., 2024) coarsen inside the network. Discretization-invariant operators (Li et al., 2021; Kovachki et al., 2023; Raonic et al., 2023; Liu-Schiaffini et al., 2024) and foundation models (Hao et al., 2024; McCabe et al., 2026) serve several resolutions of a whole frame. Graph networks advance adaptive meshes refined by a learned sizing field (Pfaff et al., 2021), reinforcement learning (Wu et al., 2023; Yang et al., 2023) or a classical AMR solver (Perera & Agrawal, 2024; Nägele et al., 2026), and adaptive tokenizers cut a uniform frame into a quadtree of tokens (Xu et al., 2025; Zhao et al., 2026). The first two keep the finest grid as input and output, or one resolution per sample, so the cost of a step scales with the domain. The adaptive methods couple the representation to the network, which must be designed around it, and the tokenizers rebuild their tree every step while the stored state stays uniform. PARA stores the TreeGrid as its state, and the MultiGrid gives every patch a stencil of one shape, so standard backbones advance all levels unchanged.

Scale symmetry. The Well normalizes each field by statistics of its training split (Ohana et al., 2024), and MPP (McCabe et al., 2024) and Walrus (McCabe et al., 2026) take statistics from the input history. Rescaled training samples (Brandstetter et al., 2022a; Mialon et al., 2023) or a symmetry penalty (Akhound-Sadegh et al., 2023) teach the scaling symmetry. Li et al. (2025) feed a rescaled PDE coefficient, Villar et al. (2023) divide the inputs by supplied units, Horie & Mitsume (2024) divide by the mesh ratio inside every layer, and Shumaylov et al. (2025) canonicalize each input by an optimization over the group. Statistical scales carry no units, lose the ratios between fields and tell the network nothing about the time step. Taught symmetry holds only as far as the data or the penalty enforce it, and built-in symmetry needs a coefficient, units or a mesh ratio supplied with the input, or an optimization per input. PARA takes its reference quantities from each stencil, so the symmetry holds by construction with nothing supplied, and the network is conditioned on the one dimensionless number left.

3 METHOD

The domain is periodic and discretized into cells. We describe the 2D case. The extension to 3D replaces squares with cubes, four children with eight, and quadtree with octree. Table 6 in Appendix A.1 lists the symbols of this section.

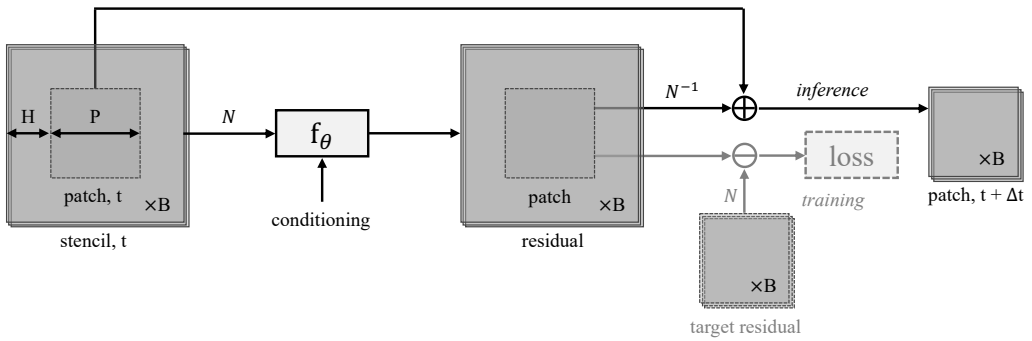


Figure 1: **One step of PARA**. N is the normalization and N^{-1} its inverse, and the conditioning depends on N (Section 3.1). $\times B$ marks a batch of B stencils processed concurrently.

The network f_θ maps a normalized stencil and conditioning (Section 3.1) to its normalized residual, the change of the stencil from t to $t + \Delta t$, from which the residual patch r is sliced for training or inference (Figure 1). A patch is a square block of $P \times P$ cells at one resolution level, holding the physical state of the C fields over the area it covers. The stencil is a patch with a band of width H cells wrapped around it, referred to as its halo.

The network is trained on the mean square error between the residual patch r and the target residual, the difference between the ground truth patches at t and $t + \Delta t$ passed through N. Both the stencils and the target residuals are cut from the MultiGrid (Section 3.2). At inference, the next patch state is $u(t + \Delta t) = u(t) + N^{-1}(r)$, where N^{-1} returns r to physical units.

3.1 Scale Normalization

We compare two normalizations N, statistical normalization, the z-score used by the Well (Ohana et al., 2024), and scale normalization, which we propose. Both bring the fields to comparable magnitudes, since fields of unequal scale make the loss ill-conditioned (LeCun et al., 1998) and let the largest field dominate the mean square error. Statistical normalization maps each input field u_c to $(u_c - \mu_c)/\sigma_c$, with μ_c and σ_c its mean and standard deviation over the training split, and N^{-1} returns the residual as $\sigma_c\, r_c$, the mean canceling in the difference. The corresponding conditioning is the level of the stencil, one-hot encoded with one channel per level.

Scale normalization uses the scaling symmetry of the Euler equations, which contain no dimensional constants and admit independent dilations of length, time and density (Appendix A.2.2). Three dimensionally independent reference quantities of the stencil therefore make the fields dimensionless, the mean density $\bar{\rho}$, the side length L_s and the energy speed $U_E = \sqrt{\bar{E}/\bar{\rho}}$, with $\bar{E}$ the mean total energy density. N maps the fields to $\rho/\bar{\rho}$, $\mathbf{m}/(\bar{\rho}\, U_E)$ and $E/(\bar{\rho}\, U_E^2)$. Each input channel is then multiplied by a constant k_c^{in} and the residual by k_c^{out}. Both are one in the data-free variant, and in the data-derived variant each is the reciprocal of that channel's spread over the training split. N^{-1} divides the residual by k_c^{out} and multiplies it by $\bar{\rho}$, $\bar{\rho}\, U_E$ or $\bar{\rho}\, U_E^2$. The corresponding conditioning is the one dimensionless number left, $\tau = U_E \Delta t / L_s$, the fraction of the stencil a signal crosses in one step, supplied as one input channel constant over the stencil.

3.2 MultiGrid

A MultiGrid is a hierarchy of L dense grids, one per resolution level, each representing the whole solution. Level 1 is the coarsest and level L the finest, and each level doubles the resolution of the previous one along each axis. A MultiGrid is built from a TreeGrid in three steps (Figure 2). Each patch of the TreeGrid is placed in the grid of its level. Restriction then fills the empty cells of the coarser levels, from fine to coarse, with the average of their four child cells. Prolongation fills the remaining empty cells, from coarse to fine, with a minmod-limited linear reconstruction from their parent cells (Appendix A.2.3). Unlike bilinear interpolation, this reconstruction preserves cell averages and introduces no artificial extrema near shocks. Restriction and prolongation never overwrite a placed patch.

The stencils and patches are sliced from the completed levels for training or inference. Each level is cut at stride P into patches, where each patch with its halo of width H into a stencil. Every stencil has the shape $(C, P + 2H, P + 2H)$ regardless of its level, so stencils of different levels can share a batch. For training, a MultiGrid is built from each frame of a pair $(t, t + \Delta t)$. The stencils of every level are cut from the first and their target patches from the second. Each frame pair therefore yields samples at every resolution, data augmentation that conserves mass, momentum and energy. For inference, a MultiGrid is built from the TreeGrid at t, and only the stencils of the TreeGrid's patches are cut and advanced.

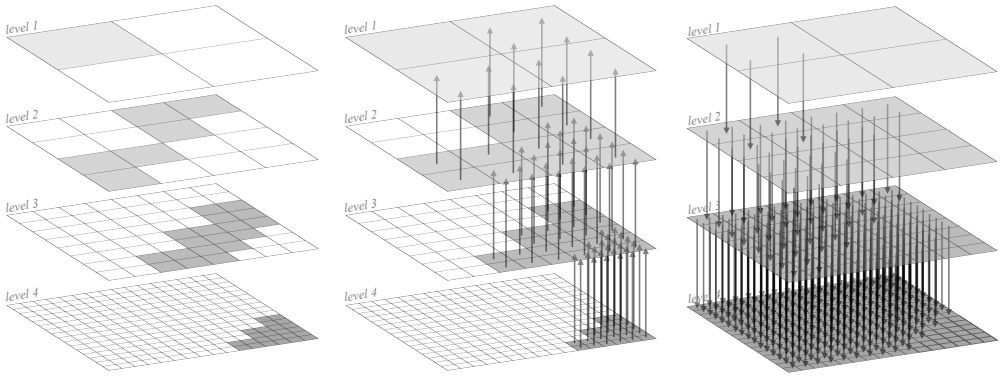


Figure 2: **Building MultiGrid from TreeGrid with $L = 4$ levels.** Left, patches are placed into the grid of their level, shaded. Middle, restriction fills the coarser levels from the finer ones. Right, prolongation fills the finer levels from the coarser ones. Arrows show the direction of data flow.

3.3 TreeGrid

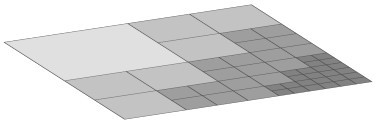


Figure 3: **A TreeGrid with four resolution levels, collapsed onto the domain.** Each non-overlapping patch is drawn at its own resolution, darker shades are finer levels.

A TreeGrid is a multi-resolution solution, represented by a pair of lists per resolution level, one holding the patches and the other their positions in the grid of that level, which together form its layout. All the patches across the levels cover the domain without overlap (Figure 3), and a patch at level l covers the same region as four patches at level $l + 1$. During training, a TreeGrid is built from each uniform fine frame of the training data, with every patch at the finest level L. During inference, the predicted patches form the TreeGrid at $t + \Delta t$, the input of the next autoregressive step.

3.4 Adaptive rollout

Every PARA model is trained on the stencils of every level of the MultiGrid (Section 3.2), so it can advance a TreeGrid of any layout without retraining. During an adaptive rollout, a new layout is chosen at every step. On the finest level of the MultiGrid, a pointwise score is computed as the gradient magnitude of the product of density and pressure, ρp, by central differences, divided by its maximum over the frame. Over each patch, the maximum pointwise score becomes the indicator score. Three thresholds assign each patch a level, from level 1 below the first threshold to level 4 above the third. Four patches merge into one coarser patch only when every finest patch they cover is assigned that coarser level or a coarser one, and neighboring patches differ by at most one level. The two rules are applied until the layout no longer changes. The stencils of the new layout are cut from the MultiGrid and advanced, and their predicted patches form the next TreeGrid.

4 Experiments

We use only the $\gamma = 1.4$ subset of the Well's compressible Euler dataset (Ohana et al., 2024), 512^2 frames of two-dimensional flows with shocks. We focus on this subset instead of all γ values to

Table 1: **One-step VRMSE on compressible Euler.** Top, each network trained whole-frame and inside PARA ($P = 32$, $H = 16$), both with statistical normalization and each under its own training procedure (Appendix A.3.2), on the same $\gamma = 1.4$ data and epochs, with one input frame. Bottom, published results on all ten γ values for reference, with the input frames per row. Lower is better. Parameters are rounded to 0.1 M, at which both forms match, exact counts in Table 9.

Backbone	Parameters	Whole frame	PARA	Ratio
U-Net 4	17.5 M	0.0366	**0.0126**	2.9
CNextU-Net	18.6 M	0.0372	**0.0150**	2.5
Flower (Muser et al., 2026)	17.3 M	0.0292	**0.0248**	1.2
scOT (Herde et al., 2024)	17.7 M	0.0563	**0.0354**	1.6
CNextU-Net, 4 frames (Ohana et al., 2024)	15–20 M	0.1531	–	–
U-Net, 4 frames (Ohana et al., 2024)	15–20 M	0.1834	–	–
Walrus, 6 frames (McCabe et al., 2026)	1.29 B	0.0194	–	–
Poseidon-L, 1 frame (McCabe et al., 2026)	628 M	0.0142	–	–
Flower-Tiny, 4 frames (Muser et al., 2026)	17.3 M	0.0160	–	–
Flower-Small, 4 frames (Muser et al., 2026)	69.3 M	0.0124	–	–
Flower-Medium, 4 frames (Muser et al., 2026)	155.8 M	0.0108	–	–

Table 2: **Rollout VRMSE on compressible Euler.** Top, the networks of Table 1, every prediction fed back as the next input. Frames 1 to 20 are rolled out from the ground truth at frame 0, and frames 21 to 60 from the ground truth at frame 20. Bottom, published results on all ten γ values for reference, reported for frames 1 to 20 only. Lower is better. Table 14 gives the medians.

	Frames 1–20		Frames 21–60	
Model	Whole frame	PARA	Whole frame	PARA
U-Net	0.651	**0.110**	0.761	**0.180**
CNextU-Net	0.700	**0.144**	0.830	**0.222**
Flower	0.207	**0.169**	0.320	**0.297**
scOT	**0.634**	1.46	0.770	**0.300**
Flower-Tiny, 4 frames (Muser et al., 2026)	0.1114	–	–	–
Flower-Small, 4 frames (Muser et al., 2026)	0.0850	–	–	–
Flower-Medium, 4 frames (Muser et al., 2026)	0.0739	–	–	–

decouple learning the solution map from identifying γ (Appendix A.3.1). We use four backbones at about 17 M parameters, the U-Net (Ronneberger et al., 2015), CNextU-Net (Ohana et al., 2024), Flower (Muser et al., 2026) and scOT (Herde et al., 2024). Each network is trained on the same data in its whole-frame and PARA forms. The whole-frame form predicts the next frame, and the PARA form the residual of each patch. Model performance is the mean of the Well's VRMSE over the test set, on the 512^2 grid. The experimental setup is detailed in Appendix A.3.

4.1 PARA AGAINST WHOLE-FRAME BASELINES

One step. PARA lowers the one-step VRMSE of every backbone, by a factor of 2.9 for the U-Net down to 1.2 for Flower (Table 1). The PARA U-Net reaches the lowest error, 0.0126. PARA's gain over the whole frame holds under either training procedure (Table 15). We attribute the gain of every backbone to the stencil, which reduces the filtering the network must learn. By contrast, a whole-frame network, whose receptive field is far wider than the domain of dependence (Appendix A.4), must learn for every output cell which input cells form its domain of dependence. Flower gains least, likely because its warps already do part of this filtering by sampling the frame locally along a learned displacement.

Rollout. Stepping patch by patch accumulates less error than stepping the whole frame (Table 2). For the U-Net the gain grows from a factor of 2.9 in one step to 5.9 over frames 1 to 20, and narrows to 4.2 over the longer rollout of frames 21 to 60. The exception is the PARA scOT over frames 1 to 20, which diverges on 17, 18, 20 trajectories. Its mean of 1.46 lies above the whole frame at 0.634, while its median of 0.178 lies below the whole-frame median of 0.624 (Table 14).

Table 3: **Halo width H against the stencil's coverage of the domain of dependence and the one-step VRMSE.** The PARA U-Net of Table 1, trained for ten epochs instead of forty. S is the stencil size in cells, c the fraction of training patches whose domain of dependence lies inside the stencil, and m the reaching cells outside the stencil as a fraction of the stencil's cells. Rows with $H > 16$ train on levels 2 to 4 only, since their stencil exceeds the 64 cells of level 1, which leaves the level-4 score comparable. The last row is the whole-frame U-Net of Table 1.

H	S	c (%)	m (%)	VRMSE
0	32^2	0	137	0.1080
8	48^2	5.2	16.1	0.0289
16	64^2	81.5	1.33	0.0167
24	80^2	95.0	0.090	0.0151
32	96^2	99.5	0.0037	0.0150
frame	512^2	100	0	0.0366

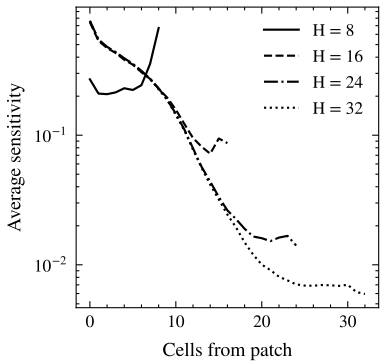


Figure 4: **Sensitivity falls with distance from the patch, the same for every $H \geq 16$.** The sensitivity of the PARA U-Nets of Table 3 is the absolute gradient of the patch residual's L_1 norm with respect to each input cell, averaged over the four input fields (Figure 13). Each curve averages the level-4 sensitivity over the square ring of cells at each distance r from the patch edge, one curve per halo width H, and ends at its halo edge.

4.2 Halo width

Coverage. Prediction performance saturates with the stencil's coverage of the domain of dependence, but degrades at the whole-frame extreme (Table 3). As the halo width increases from 0 to 32, the one-step VRMSE drops and then flattens, and it rises again when the stencil is the whole frame, even with four times the training epochs. To understand this, we compute two measures from the training data. c is the fraction of training patches whose domain of dependence lies inside the stencil, and m counts the cells outside the stencil that reach the patch, as a percentage of the cells in the stencil. The dependence map visualizes m beside the density and the signal speed of the frame (Figure 6). As H grows, c saturates toward 100 percent and m decays toward zero, so beyond $H = 16$ a wider halo adds little information. We therefore use $H = 16$ throughout the paper, which covers the signal reach of most patches, a median of 12.1 cells and a 90th percentile of 17.9 (Table 7).

Sensitivity. The network learns the typical extent of the domain of dependence and weights its input by it (Figure 4). The sensitivity of the patch residual to an input cell falls by an order of magnitude over the first 16 cells from the patch, about the distance a signal travels in one step (Table 7). From $H = 16$ the curves of different halo widths coincide where they overlap, so a wider halo does not change how the network weights a cell at a given distance. At $H = 8$, where the stencil holds the domain of dependence of only 5 percent of patches, the sensitivity instead rises toward the stencil edge, as the network leans on the boundary of what it sees. The stencil therefore reduces the filtering but does not remove it, and the network still learns which cells of the stencil matter.

4.3 Scale Normalization

Scale normalization makes the PARA U-Net scale-equivariant without loss of accuracy, while statistical normalization does not (Table 4). Under the three scale symmetries of the Euler equations, in density, time and length, rescaling the current state rescales the next state in the same way. An equivariant predictor rescales its prediction in the same way, so the error of each field and the standard deviation of its ground truth scale by the same factor, and their ratio, the VRMSE, is unchanged. We compare only the density scaling, which multiplies every conserved field by a, because statistical normalization cannot represent the time step and cell size that the time and length scalings change. Scale normalization is equivariant under those two by construction, since both are absorbed in τ (Appendix A.2.2). Across a, the one-step VRMSE varies by 0.064 percent for the data-free variant and 0.010 percent for the data-derived variant, below the 0.11 percent the data-free variant varies between two training seeds (Table 16). By contrast, the error of statistical normalization rises twenty times at $a = 0.1$ and five times at $a = 10$, since its fixed statistics no longer describe the input. At

Table 4: **Scale normalization is equivariant under field scaling, statistical normalization is not.** One-step VRMSE of the PARA U-Net of Table 1 by the level every patch is advanced at, coarser levels prolonged to the 512^2 grid before scoring. Rows differ only in the normalization and its conditioning (Section 3.1). The first two columns multiply every conserved field of the test input and truth by a.

	Level 4, fields $\times\, a$		Level, $a = 1$			
Normalization	$a = 0.1$	$a = 10$	4	3	2	1
Statistical, level channels	0.2511	0.0585	0.0126	0.0347	0.0811	0.1515
Scale, data-free, τ	0.0120	0.0120	0.0120	0.0337	0.0795	0.1503
Scale, data-derived, τ	**0.0116**	**0.0116**	**0.0116**	**0.0334**	**0.0793**	**0.1500**

Table 5: **Adaptive rollout.** Rollout VRMSE of the statistical and scale-normalized PARA U-Nets on the test split, averaged as in Table 2, on the uniform grid and on adaptive layouts at three threshold settings (Section 3.4). The row stencils per step gives the mean number of stencils the network advances each step, 256 on the uniform grid. The uniform rollout of the scale-normalized U-Net diverges on one trajectory, 17, over frames 1 to 20, with a median of 0.065, while none of its adaptive rollouts does.

		uniform	adaptive, thresholds		
Network	frames		0.01, 0.05, 0.15	0.02, 0.1, 0.3	0.05, 0.2, 0.5
U-Net	1–20	**0.110**	0.116	0.124	0.135
U-Net scale	1–20	3.475	**0.101**	**0.108**	**0.116**
U-Net	21–60	0.180	0.201	0.218	0.232
U-Net scale	21–60	**0.171**	**0.176**	**0.191**	**0.207**
stencils per step	1–20	256	112 (0.44x)	79 (0.31x)	56 (0.22x)
stencils per step	21–60	256	130 (0.51x)	86 (0.34x)	59 (0.23x)

the original scale, $a = 1$, both scale variants have a lower error than statistical normalization at every level, by five and eight percent at the finest.

4.4 Adaptive Refinement

Cost. Adaptive rollouts cut the cost of a step at a small rise in error, without retraining (Table 5). At the finest thresholds the statistical U-Net advances 0.44 of the uniform stencils over frames 1 to 20 for 5.5 percent more error, and coarser thresholds trade error for cost smoothly, down to 0.22 of the stencils for 23 percent more error. Over frames 21 to 60, the flow features spread out from the discontinuous initial conditions and need more stencils to represent, 0.51 of the uniform grid against 0.44 over frames 1 to 20 (Figure 5). The thresholds are free parameters, so a user can set the balance between cost and error for the task at hand.

Stability. The uniform rollout of the scale-normalized U-Net diverges on the fastest trajectory, 17, over frames 1 to 20, which lifts its mean to 3.475 against a median of 0.065, while none of its adaptive rollouts does (Table 5). Trajectory 17 has the fastest signals of the test set, up to 55.5 cells a step against the halo of 16 (Figure 7).

5 Conclusion and Future Work

In this paper, we introduced **PARA**, a patch-based, symmetry-driven framework for hyperbolic PDEs. By shifting multi-resolution tracking to a patch tree, PARA eliminates network complexity and enables single-batch operations. Localized stencils enforce an inductive bias toward the domain of dependence. Furthermore, our scale normalization embeds physical symmetries directly into the solver, guaranteeing scale equivariance. This mathematically grounded approach consistently outperforms whole-frame baselines. While **PARA** demonstrates exceptional accuracy on the challenging compressible Euler benchmark, our current evaluation focuses on a representative family of two-dimensional Riemann problems under periodic boundary conditions. Extending this patch-based framework to broader physical regimes offers fruitful future work. For instance, compressible

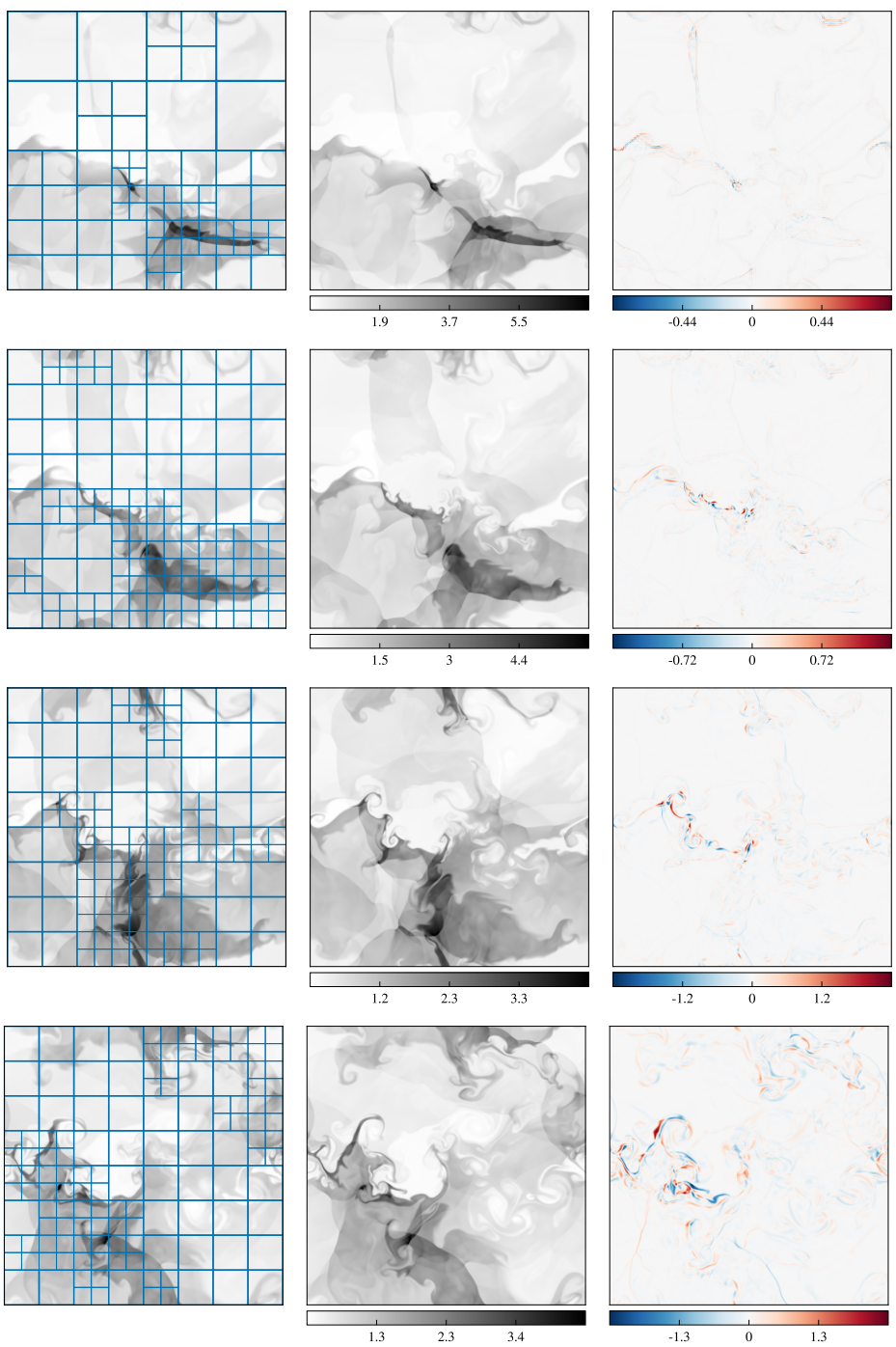


Figure 5: **An adaptive rollout.** Density of the scale-normalized U-Net on test trajectory 17 at frames 21, 25, 30 and 40, top to bottom, rolled out from the ground truth at frame 20 with thresholds 0.01, 0.05 and 0.15. Left, the prediction with its layout. Middle, the ground truth. Right, their difference. Frames 1 to 20 are in Figure 14.

Navier-Stokes equations preserve the finite signal speed central to our localized stencil while introducing viscosity, which can be naturally incorporated into our scale normalization via Reynolds-number conditioning. Conversely, globally coupled systems such as incompressible flows present an exciting avenue for adapting the stencil to manage non-local pressure field interactions. Finally, the localized divergence observed in the uniform rollout of the scale-normalized U-Net (Table 5) highlights an intriguing stability trade-off inherent to dynamic stencil tracking; isolating and resolving this via bounded reference configurations remains a valuable direction for future refinement.

AI USE STATEMENT

In this work, we used generative AI tools for helping refine conceptual frameworks, formulating mathematical claims, refining hypotheses, providing feedback on research methodology and experiments, implementing methods, and interpreting results. We have not used generative AI tools for generating synthetic data sets, providing critical ingredients for proving mathematical claims, cleaning and reformatting data sets, and supporting qualitative and thematic data analysis, and assisting in the writing of proofs and assisting with translation are not applicable to this work. Additionally, we used generative AI tools for summarizing or analyzing existing literature, improving readability, identifying relevant literature, formatting references, and suggesting a structure for a research paper. We have reviewed all AI-assisted work. AI-generated code served only as a draft, and the authors rewrote the architecture and the code themselves. The authors reviewed all AI-drafted text sentence by sentence and revised it where needed, checked every reported number against their own experimental logs, and checked every claim about prior work against the cited paper. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work trains neural PDE solvers on the publicly available simulations of the Well (Ohana et al., 2024). It involves no human subjects or personal data, and we see no ethical concerns beyond those of scientific computing in general.

REPRODUCIBILITY STATEMENT

The architectures, their widths and parameter counts are given in Appendix A.4, the two training procedures in Table 8, the transfer operators between levels in Appendix A.2.3, the scale normalization with its references and constants in Section 3.1, and the adaptive rollout's indicator, thresholds and balance rules in Section 3.4. The dataset is the Well's compressible Euler set with its published normalization statistics, and the whole-frame baselines are trained through the public Flowers repository under the settings of Table 8. Every run's configuration is stated in the appendix tables, with the seed where the trainer takes one, and the seed comparison of Appendix A.5.5 gives the spread a reader should expect. We will release the code upon publication.

REFERENCES

Daniel S. Abdi, Ann Almgren, Francis X. Giraldo, and Isidora Jankov. Comparison of adaptive mesh refinement techniques for numerical weather prediction, 2024. URL https://arxiv.org/abs/2404.16648.

Tara Akhound-Sadegh, Laurence Perreault-Levasseur, Johannes Brandstetter, Max Welling, and Siamak Ravanbakhsh. Lie point symmetry and physics-informed networks. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=ba4boN3W1n.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016. URL https://arxiv.org/abs/1607.06450.

Yohai Bar-Sinai, Stephan Hoyer, Jason Hickey, and Michael P. Brenner. Learning data-driven discretizations for partial differential equations. *Proceedings of the National Academy of Sciences*, 116(31):15344–15349, 2019. ISSN 1091-6490. doi: 10.1073/pnas.1814058116. URL http://dx.doi.org/10.1073/pnas.1814058116.

Grigory Isaakovich Barenblatt. *Scaling, Self-similarity, and Intermediate Asymptotics: Dimensional Analysis and Intermediate Asymptotics*. Cambridge University Press, December 1996. ISBN 9781107050242. doi: 10.1017/cbo9781107050242. URL http://dx.doi.org/10.1017/CBO9781107050242.

Marsha J. Berger and Phillip Colella. Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics*, 82(1):64–84, 1989. doi: 10.1016/0021-9991(89)90035-1.

Marsha J. Berger and Joseph Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of Computational Physics*, 53(3):484–512, 1984. doi: 10.1016/0021-9991(84)90073-1.

Johannes Brandstetter, Max Welling, and Daniel E Worrall. Lie point symmetry data augmentation for neural PDE solvers. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 2241–2256. PMLR, 17–23 Jul 2022a. URL https://proceedings.mlr.press/v162/brandstetter22a.html.

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

Steven L. Brunton, Bernd R. Noack, and Petros Koumoutsakos. Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics*, 52(1):477–508, January 2020. ISSN 1545-4479. doi: 10.1146/annurev-fluid-010719-060214. URL http://dx.doi.org/10.1146/annurev-fluid-010719-060214.

E. Buckingham. On physically similar systems; illustrations of the use of dimensional equations. *Physical Review*, 4(4):345–376, October 1914. ISSN 0031-899X. doi: 10.1103/physrev.4.345. URL http://dx.doi.org/10.1103/PhysRev.4.345.

Chun-Wun Cheng, Bin Dong, Carola-Bibiane Schönlieb, and Angelica I Aviles-Rivero. Pde solvers should be local: Fast, stable rollouts with learned local stencils, 2025. URL https://arxiv.org/abs/2509.26186.

Jayesh K Gupta and Johannes Brandstetter. Towards multi-spatiotemporal-scale generalized PDE modeling. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=dPSTDbGtBY.

Zhongkai Hao, Chang Su, Songming Liu, Julius Berner, Chengyang Ying, Hang Su, Anima Anandkumar, Jian Song, and Jun Zhu. DPOT: Auto-regressive denoising operator transformer for large-scale PDE pre-training. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 17616–17635. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/hao24d.html.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus), 2016. URL https://arxiv.org/abs/1606.08415.

Maximilian Herde, Bogdan Raonic, 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 37*, NeurIPS 2024, pp. 72525–72624. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2024. doi: 10.52202/079017-2311. URL http://dx.doi.org/10.52202/079017-2311.

Masanobu Horie and Naoto Mitsume. Graph neural PDE solvers with conservation and similarity-equivariance. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 18785–18814. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/horie24a.html.

Nail H. Ibragimov. *CRC Handbook of Lie Group Analysis of Differential Equations, Volume I: Symmetries, Exact Solutions, and Conservation Laws*. CRC Press, August 2023. ISBN 9781003419808. doi: 10.1201/9781003419808. URL http://dx.doi.org/10.1201/9781003419808.

Marimuthu Kalimuthu, David Holzmüller, and Mathias Niepert. LOGLO-FNO: Efficient learning of local and global features in fourier neural operators. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=MQ1dRdHTpi. J2C Certification.

Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24(89):1–97, 2023. URL http://jmlr.org/papers/v24/21-1524.html.

Anuj Kumar, Heiko Zimmermann, Josiah Bjorgaard, Jacan Chaplais, Nikolaos Bouklas, Matteo Salvador, and Alexander Lavin. Read, write, relax: Why neural pde surrogates need both global and local processing, 2026. URL https://arxiv.org/abs/2608.21677.

Zishuo Lan, Junjie Li, Lei Wang, and Jincheng Wang. Fluxnet: Learning capacity-constrained local transport operators for conservative and bounded PDE surrogates. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=1KRpajnd6u.

Yann LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Efficient BackProp. In Genevieve B. Orr and Klaus-Robert Müller (eds.), *Neural Networks: Tricks of the Trade*, volume 1524 of *Lecture Notes in Computer Science*, pp. 9–50. Springer, Berlin, Heidelberg, 1998. doi: 10.1007/3-540-49430-8_2.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015. ISSN 1476-4687. doi: 10.1038/nature14539. URL http://dx.doi.org/10.1038/nature14539.

Randall J. LeVeque. *Finite Volume Methods for Hyperbolic Problems*. Cambridge University Press, August 2002. ISBN 9780511791253. doi: 10.1017/cbo9780511791253. URL http://dx.doi.org/10.1017/CBO9780511791253.

Hongyu Li, Ximeng Ye, Peng Jiang, Guoliang Qin, and Tiejun Wang. Local neural operator for solving transient partial differential equations on varied domains. *Computer Methods in Applied Mechanics and Engineering*, 427:117062, 2024.

Zhihao Li, Zhilu Lai, Xiaobo Zhang, and Wei Wang. M2no: An efficient multi-resolution operator framework for dynamic multi-scale pde solvers. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1*, KDD '26, pp. 771–782, New York, NY, USA, 2026. Association for Computing Machinery. ISBN 9798400722585. doi: 10.1145/3770854.3780179. URL https://doi.org/10.1145/3770854.3780179.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Andrew Stuart, Kaushik Bhattacharya, and Anima Anandkumar. Multipole graph neural operator for parametric partial differential equations. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 6755–6766. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/4b21cf96d4cf612f239a6c322b10c8fe-Paper.pdf.

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.

Zongyi Li, Samuel Lanthaler, Catherine Deng, Michael Chen, Yixuan Wang, Kamyar Azizzadenesheli, and Anima Anandkumar. Scale-consistent learning for partial differential equations, 2025. URL https://arxiv.org/abs/2507.18813.

Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, Furu Wei, and Baining Guo. Swin transformer v2: Scaling up capacity and resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12009–12019, June 2022a.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11976–11986, June 2022b.

Miguel Liu-Schiaffini, Julius Berner, Boris Bonev, Thorsten Kurth, Kamyar Azizzadenesheli, and Anima Anandkumar. Neural operators with localized integral and differential kernels. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32576–32594. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/liu-schiaffini24a.html.

Suryanarayana Maddu, Dominik Sturm, Bevan L. Cheeseman, Christian L. Müller, and Ivo F. Sbalzarini. Stencil-net: Data-driven solution-adaptive discretization of partial differential equations, 2021. URL https://arxiv.org/abs/2101.06182.

Michael McCabe, Bruno Régaldo-Saint Blancard, Liam Parker, Ruben Ohana, Miles Cranmer, Alberto Bietti, Michael Eickenberg, Siavash Golkar, Geraud Krawezik, Francois Lanusse, Mariel Pettee, Tiberiu Tesileanu, Kyunghyun Cho, and Shirley Ho. Multiple physics pretraining for spatiotemporal surrogate models. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 119301–119335. Curran Associates, Inc., 2024. doi: 10.52202/079017-3791. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/d7cb9db5ade2db7814fbd01ee59f4c7b-Paper-Conference.pdf.

Michael McCabe, Payel Mukhopadhyay, Tanya Marwah, Bruno Régaldo-Saint Blancard, François Rozet, Cristiana Diaconu, Lucas Thibaut Meyer, Kaze W. K. Wong, Mohammad-Hadi Sotoudeh, Alberto Bietti, Irina Espejo Morales, Rio Alexa Fear, Siavash Golkar, Tom Hehir, Keiya Hirashima, Geraud Krawezik, Francois Lanusse, Rudy Morel, Ruben Ohana, Liam Holden Parker, Mariel Pettee, Jeff Shen, Kyunghyun Cho, Miles Cranmer, and Shirley Ho. Walrus: A cross-domain foundation model for continuum dynamics. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=kcyOjXoUZu.

Grégoire Mialon, Quentin Garrido, Hannah Lawrence, Danyal Rehman, Yann LeCun, and Bobak Kiani. Self-supervised learning with lie symmetries for partial differential equations. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 28973–29004. Curran Associates, Inc., 2023. doi: 10.52202/075280-1262. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/5c46ae130105fa012da0446126c01d1d-Paper-Conference.pdf.

Till Muser, Alexandra Spitzer, Matti Lassas, Maarten V. de Hoop, and Ivan Dokmanić. Flowers: A warp drive for neural PDE solvers. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=TPpUKH8fuQ.

Matthias Nägele, Cedric Bös, Chester Tan, Christian M. Fromm, Ingo Scholtes, and Karl Mannheim. Learning neural operator surrogates for the black hole accretion code, 2026. URL https://arxiv.org/abs/2604.25985.

Ruben Ohana, Michael McCabe, Lucas Meyer, Rudy Morel, Fruzsina J. Agocs, Miguel Beneitez, Marsha Berger, Blakesley Burkhart, Stuart B. Dalziel, Drummond B. Fielding, Daniel Fortunato, Jared A. Goldberg, Keiya Hirashima, Yan-Fei Jiang, Rich R. Kerswell, Suryanarayana Maddu, Jonah Miller, Payel Mukhopadhyay, Stefan S. Nixon, Jeff Shen, Romain Watteaux, Bruno Régaldo-Saint Blancard, François Rozet, Liam Parker, Miles Cranmer, and Shirley Ho. The well: a large-scale collection of diverse physics simulations for machine learning. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 44989–45037. Curran Associates, Inc., 2024. doi: 10.52202/079017-1430. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/4f9a5acd91ac76569f2fe291b1f4772b-Paper-Datasets_and_Benchmarks_Track.pdf.

Oded Ovadia and Eli Turkel. LLT: Local linear transformer for PDE operator learning, 2026. URL https://arxiv.org/abs/2607.07718.

Roberto Perera and Vinamra Agrawal. Multiscale graph neural networks with adaptive mesh refinement for accelerating mesh-based simulations. *Computer Methods in Applied Mechanics and Engineering*, 429:117152, 2024.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=roNqYL0_XP.

Bogdan Raonic, Roberto Molinaro, Tim De Ryck, Tobias Rohner, Francesca Bartolucci, Rima Alaifari, Siddhartha Mishra, and Emmanuel de Bézenac. Convolutional neural operators for robust and accurate learning of pdes. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 77187–77200. Curran Associates, Inc., 2023. doi: 10.52202/075280-3376. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/f3c1951b34f7f55ffaecada7fde6bd5a-Paper-Conference.pdf.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*, pp. 234–241. Springer International Publishing, 2015. ISBN 9783319245744. doi: 10.1007/978-3-319-24574-4_28. URL http://dx.doi.org/10.1007/978-3-319-24574-4_28.

Peter Sharpe. Globe: Accurate and generalizable pde surrogates using domain-inspired architectures and equivariances, 2025. URL https://arxiv.org/abs/2511.15856.

Zakhar Shumaylov, Peter Zaika, James Rowbottom, Ferdia Sherry, Melanie Weber, and Carola-Bibiane Schönlieb. Lie algebra canonicalization: Equivariant neural operators under arbitrary lie groups. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=7PLpiVdnUC.

Soledad Villar, Weichi Yao, David W Hogg, Ben Blum-Smith, and Bianca Dumitrascu. Dimensionless machine learning: Imposing exact units equivariance. *Journal of Machine Learning Research*, 24(109):1–32, 2023.

Hao Wang, Chi-Wang Shu, and Qi Tang. LGNO: A local-global neural operator for hyperbolic conservation laws, 2026. URL https://arxiv.org/abs/2606.18221.

Hao Wu, Yuan Gao, Fan Xu, Fan Zhang, Qingsong Wen, Xiaomeng Huang, and Xian Wu. Differential-integral neural operator for long-term turbulence forecasting. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, pp. 5431–5441. ACM, August 2026. doi: 10.1145/3770855.3818015. URL http://dx.doi.org/10.1145/3770855.3818015.

Tailin Wu, Takashi Maruyama, Qingqing Zhao, Gordon Wetzstein, and Jure Leskovec. Learning controllable adaptive simulation for multi-resolution physics. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PbfgkZ2HdbE.

Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

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)*, pp. 5804–5813, June 2025.

Jiachen Yang, Tarik Dzanic, Brenden Petersen, Jun Kudo, Ketan Mittal, Vladimir Tomov, Jean-Sylvain Camier, Tuo Zhao, Hongyuan Zha, Tzanio Kolev, Robert Anderson, and Daniel Faissol. Reinforcement learning for adaptive mesh refinement. In Francisco Ruiz, Jennifer Dy, and Jan-Willem van de Meent (eds.), *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pp.

5997–6014. PMLR, 25–27 Apr 2023. URL https://proceedings.mlr.press/v206/yang23e.html.

Ximeng Ye, Hongyu Li, Jingjie Huang, and Guoliang Qin. On the locality of local neural operator in learning fluid dynamics. *Computer Methods in Applied Mechanics and Engineering*, 427:117035, 2024. ISSN 0045-7825. doi: 10.1016/j.cma.2024.117035. URL http://dx.doi.org/10.1016/j.cma.2024.117035.

Baiming Zhang, Jinsong Tang, Ying Xu, Lihua Chen, and Shiying Xiong. Local gradient neural operator, 2026a. URL https://arxiv.org/abs/2609.07752.

Xiaowen Zhang, Ziming Zhou, Fengnian Zhao, and David L.S. Hung. Flowforge: A staged local rollout engine for flow field prediction. In *ICML 2026 Workshop on AI for Physics*, 2026b. URL https://openreview.net/forum?id=vuHHaqe0Hy.

Yanshun Zhao, Xiaoyu Peng, Jiamin Jiang, Congcong Zhu, and Jingrun Chen. Meshtok: Efficient multi-scale tokenization for scalable PDE transformers. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=I8QGwNBO4T.

A Appendix

A.1 Notation

Table 6: Notation.

Symbol	Meaning	Shape
D	domain size in cells along each axis	512
C	Number of fields	4
P	patch size in cells	32
H	halo width in cells	16
L	number of resolution levels	4
B	batch size, patches processed concurrently	64
N	normalization, physical to normalized units	
N^{-1}	inverse normalization, normalized to physical units	
f_θ	network, maps a stencil to a residual	
patch	unit of state at one level	(C, P, P)
stencil, S	patch with its halo	$(C, P+2H, P+2H)$

A.2 Method Details

A.2.1 Compressible Euler Equations

The Well's compressible Euler dataset solves the two-dimensional Euler equations for a compressible gas in conservative form,

$$U_t + F(U)_x + G(U)_y = 0, \tag{1}$$

$$U = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ e \end{bmatrix}, \quad F(U) = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(e+p) \end{bmatrix}, \quad G(U) = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(e+p) \end{bmatrix}, \tag{2}$$

closed by the ideal gas law

$$e = \frac{p}{\gamma - 1} + \frac{\rho(u^2 + v^2)}{2}, \tag{3}$$

with U the vector of conserved fields, the mass density ρ, the momentum densities ρu and ρv along x and y, the total energy density e, and $F(U)$ and $G(U)$ the fluxes along x and y. Here u and v are the velocity components, p the pressure and γ the ratio of specific heats. The system is hyperbolic with largest signal speed $|u| + c$ along each axis, where $c = \sqrt{\gamma p/\rho}$ is the sound speed.

A.2.2 SCALE NORMALIZATION

The source-free compressible Euler equations, closed by the ideal-gas equation of state with fixed dimensionless γ, contain no dimensional constants and admit a three-parameter scaling symmetry (Ibragimov, 2023), which is the basis of the scale normalization of Section 3.1. Independent dilations of length, time and density induce dilations of velocity, momentum and energy that preserve the equations. The equations therefore prescribe no intrinsic length, time or density scale, and the scales of a problem come from its geometry and its initial and boundary data. This is what allows the constants of the normalization to be taken from each stencil.

Let $\bar{\rho} > 0$ and $\bar{E} > 0$ be the stencil means of density and total energy density, and L_s the stencil side length, with $L_s/\Delta x = P + 2H$ fixed across levels. The reference density, velocity and time are

$$\rho_{\mathrm{ref}} = \bar{\rho}, \qquad U_E = \sqrt{\frac{\bar{E}}{\bar{\rho}}}, \qquad T_E = \frac{L_s}{U_E}.$$

The input fields in these units are

$$\widehat{\rho} = \frac{\rho}{\bar{\rho}}, \qquad \widehat{\mathbf{m}} = \frac{\mathbf{m}}{\bar{\rho}\, U_E}, \qquad \widehat{E} = \frac{E}{\bar{\rho}\, U_E^2} = \frac{E}{\bar{E}},$$

and the normalized density and energy have stencil mean one. The momentum scale is tied to the density and energy scales so that the ideal-gas pressure relation keeps its form,

$$\widehat{p} = (\gamma - 1)\left(\widehat{E} - \frac{|\widehat{\mathbf{m}}|^2}{2\widehat{\rho}}\right), \qquad \widehat{p} = \frac{p}{\bar{\rho}\, U_E^2}.$$

With lengths in units of L_s and time in units of T_E, the Euler equations keep their original form, and the time step becomes

$$\tau = \frac{\Delta t}{T_E} = \frac{U_E\, \Delta t}{L_s}.$$

For a fixed stencil geometry and a dimensionally consistent numerical scheme, the normalized one-step update is therefore a function of the normalized fields and τ (Buckingham, 1914; Barenblatt, 1996). τ is supplied as a conditioning channel constant over the stencil, five input channels in two dimensions. It gives the update interval relative to the local reference time and carries the dependence on grid spacing and time step.

Taking U_E from the state keeps the field normalization independent of Δx and Δt. A velocity reference $\Delta x/\Delta t$ would change the normalized amplitudes whenever that ratio changes between levels. At fixed Δt, doubling Δx halves the normalized momentum and quarters the normalized energy of an unchanged state. Our choice places this dependence in τ and leaves the normalized fields unchanged across levels.

The representation is invariant under the Euler scaling

$$\mathbf{x}' = \lambda \mathbf{x}, \qquad t' = \mu t, \qquad \rho' = \nu \rho, \qquad \mathbf{m}' = \nu \frac{\lambda}{\mu}\, \mathbf{m}, \qquad E' = \nu \left(\frac{\lambda}{\mu}\right)^2 E,$$

for positive λ, μ, ν, with $L_s' = \lambda L_s$ and $\Delta t' = \mu \Delta t$. Under it $U_E' = (\lambda/\mu)\, U_E$, so the normalized fields and τ are unchanged, and scale-related stencils have identical network inputs. The predicted normalized residual is returned to physical units by the input stencil's constants $\bar{\rho}$, $\bar{\rho}\, U_E$ and $\bar{\rho}\, U_E^2$.

A.2.3 RESTRICTION AND PROLONGATION

Two operators move values between adjacent levels of the MultiGrid (Section 3.2).

Restriction. A 2×2 mean pool is applied over the whole finer grid. Cells in the one level coarser grid are marked if the finer grid has real values at their position. Masked coarse cells then take the mean of its four children values. The coarser levels of a MultiGrid are the exact restrictions of the finest, so the conserved quantities agree across levels.

Prolongation. Each coarse cell is subdivided into four equal children using a conservative, limited linear reconstruction, applied componentwise to the conserved variables. For a coarse-cell average $u_{i,j}$, define the limited differences

$$\sigma^x_{i,j} = \operatorname{minmod}\left(u_{i+1,j} - u_{i,j},\, u_{i,j} - u_{i-1,j}\right),$$
$$\sigma^y_{i,j} = \operatorname{minmod}\left(u_{i,j+1} - u_{i,j},\, u_{i,j} - u_{i,j-1}\right).$$

The minmod function returns the argument of smaller magnitude when both arguments have the same sign, and zero otherwise (LeVeque, 2002). Neighbor indices are periodic at the domain boundaries. Averaging the linear reconstruction over each child gives

$$u^{a,b}_{i,j} = u_{i,j} + \frac{a}{4}\sigma^x_{i,j} + \frac{b}{4}\sigma^y_{i,j}, \qquad a, b \in \{-1, +1\},$$

where a and b identify the child's position relative to the parent center. The factors of $1/4$ correspond to the displacement of each child center by one quarter of the coarse-cell width along each axis.

The child averages satisfy

$$\frac{1}{4}\sum_{a,b\in\{-1,+1\}} u^{a,b}_{i,j} = u_{i,j},$$

so prolongation preserves the integral of each conserved variable over the parent cell. Each child value also lies within the range of the parent and its four face-adjacent coarse neighbors, preventing interpolation overshoots beyond this local range. Since $\operatorname{minmod}(ca, cb) = c\,\operatorname{minmod}(a, b)$ for $c > 0$, prolongation commutes with positive componentwise rescaling of the conserved variables.

A.3 Experiment Setup

A.3.1 Data

We use the Well's compressible Euler dataset (Ohana et al., 2024)[1] which provides the numerical solution of the two-dimensional compressible Euler equations (Appendix A.2.1) for mass density, two momentum densities and total energy density, closed by an ideal gas law with ratio of specific heats γ, together with per-field mean and standard deviation over its training split used for the data z-score normalization in Section 3. The initial condition is a Riemann problem with piecewise constant states in quadrants on a periodic domain, whose interfaces launch shocks, contacts and rarefactions.

The full dataset spans ten values of γ. Its $\gamma = 1.4$ subset holds 400, 50 and 50 trajectories for training, validation and testing. Each trajectory has 101 frames, the initial condition and 100 steps of $\Delta t = 0.015\,\mathrm{s}$, a horizon of $1.5\,\mathrm{s}$. A frame holds the four conserved fields, the density ρ, the two components of the momentum $\mathbf{m}$ and the energy E, and the primitive pressure p, on a uniform 512×512 grid over a square domain of side $1\,\mathrm{m}$.

We use the $\gamma = 1.4$ subset to decouple solving from system identification, since a network trained on all ten values must identify γ from the frame as well as advance the state. One frame in and one frame out gives 100 pairs per trajectory, 40,000 for training and 5,000 for test. PARA uses the four conserved fields and recovers p through the ideal gas law, while the whole-frame networks use all five.

Dependence map. Figure 6 shows the dependence map of a patch, the one-step travel of information from every cell of the training frame toward the patch, beside the density and the signal speed field of that frame. For a cell x, let $d(x)$ be the distance in cells to the nearest cell of the patch, the short way round the periodic domain, and $n(x)$ the unit vector along it. The distance information travels from x in one step is $(u(x) \cdot n(x) + c(x))\,\Delta t/\Delta x$, the fastest characteristic speed toward the patch times the time step, in cells, with c the sound speed and Δx the cell size. A cell is in the

[1] https://polymathic-ai.org/the_well/datasets/euler_multi_quadrants_periodicBC/

domain of dependence of the patch when its arrow reaches the patch, that is when its travel is at least $d(x)$. The reach $(|u| + c)\,\Delta t/\Delta x$ bounds this travel in every direction. Over the training split it has a median of 12.1 cells and a 90th percentile of 17.9 (Table 7), so a halo of 16 cells holds the reach of most cells.

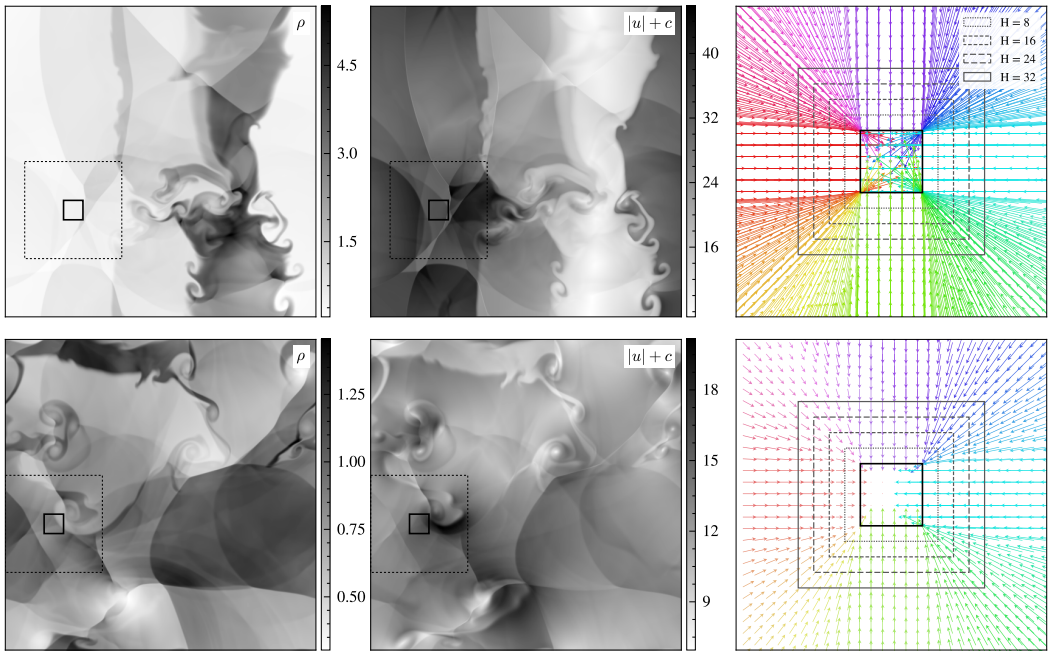


Figure 6: The dependence map of two patches. The top and bottom rows are the training frames and patches with the most and the median number of cells outside the $H = 16$ stencil reaching the patch in one step, the patch in black. Left, the density ρ. Middle, the signal speed $|u| + c$ in cells per step. Right, the 160 cells around the patch, the dashed square on the left two panels, with the travel of every cell toward the patch in one step. An arrow that reaches the patch marks a cell whose state the patch depends on at the next step. Gray squares are the stencil outlines for $H = 8$, 16, 24 and 32.

Signal reach. Table 7 gives the distribution of the one-step signal reach over the training split, and Figure 7 its largest value in every test frame together with the sharpest density step.

Table 7: **One-step signal reach over the training split.** For every cell of the 40,000 training frames, the distance its signal travels in one step, $(|u| + c)\,\Delta t/\Delta x$ cells, with u the velocity, c the sound speed at $\gamma = 1.4$, $\Delta t = 0.015$ and $\Delta x = 1/512$. Percentiles are read from a histogram with bins of 0.01 cells. The reach is a per-cell bound that ignores direction, which the dependence map of Figure 6 accounts for.

Percentile	Reach (cells)
50	12.05
90	17.93
99	28.77
99.9	35.53
99.99	41.94
maximum	65.22

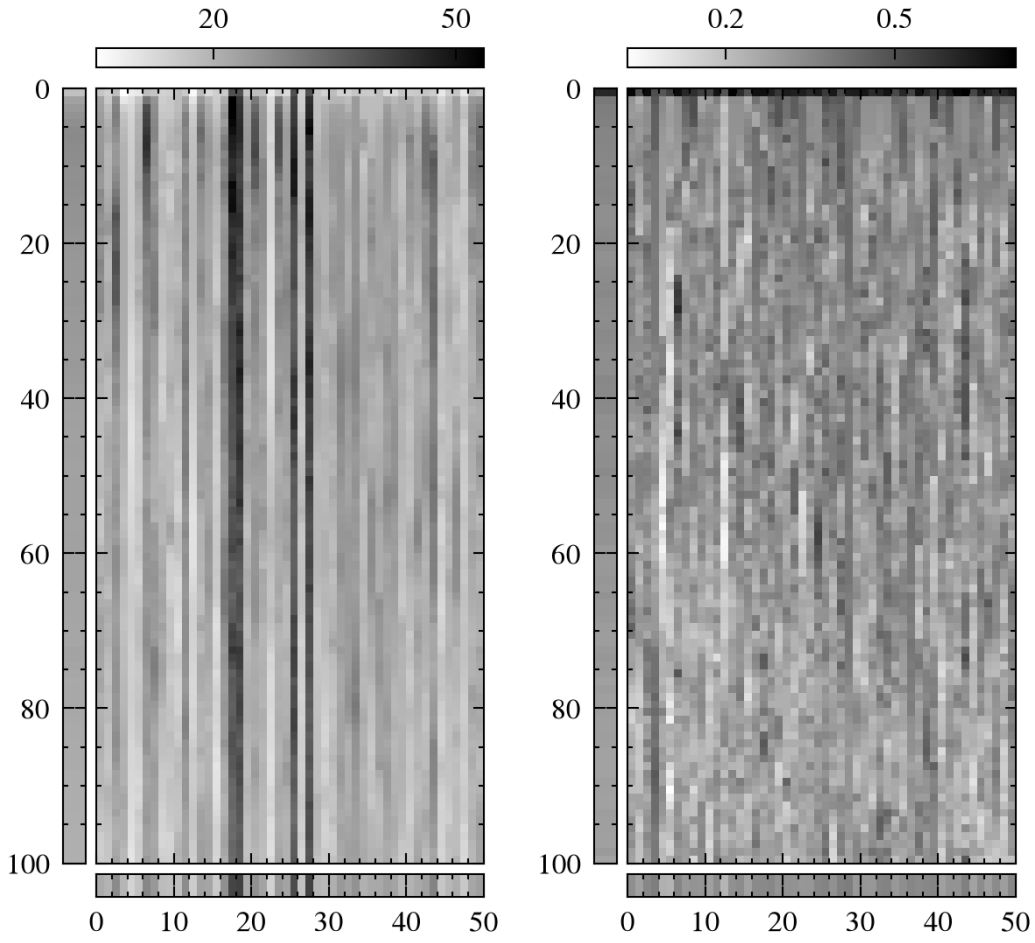


Figure 7: **Signal reach and density step of every test frame.** Each column is a test trajectory and each row a frame. Left, the largest signal reach in the frame, the maximum over cells of $(|u| + c)\,\Delta t/\Delta x$ in cells. Right, the sharpest density step, the largest difference between neighboring cells divided by the frame's largest density. In the reach map on the left, trajectories 17, 27, 25, 18 and 6 carry the fastest signals, with maxima of 55.5, 53.1, 49.2, 47.3 and 42.5 cells a step, surpassing the halo width $H = 16$. In the density step map on the right, the largest steps all occur at the first frame, with maxima of 0.963, 0.963, 0.913, 0.908 and 0.864 of the peak density on trajectories 13, 43, 42, 0 and 2, and fade to about 0.3 within a few frames.

A.3.2 TRAINING PROCEDURE

Table 8 gives the training procedures of the whole-frame and PARA training procedures. The whole-frame schedule warms up linearly over 5 epochs and then decays by cosine to a tenth of the peak rate, while the PARA schedule decays by cosine to zero. Three exceptions apply to scOT. The whole-frame scOT uses a learning rate of 5×10^{-4} and trains in float32 rather than float16 with a loss scaler, and the PARA scOT trains at a batch of 512 stencils. Both losses are the mean square error in normalized units. The whole-frame checkpoint is the best validation VRMSE over 80 frames and the PARA checkpoint the best validation MSE over 248 pairs. Each of the 40,000 frame pairs yields 340 stencils, and the PARA conditioning is four level channels under statistical normalization and one channel of τ under scale normalization.

Table 8: **Training procedures of the two forms.** The top block is shared, the middle block is each form's training procedure, and the bottom block is what each form requires.

Setting	Whole frame	PARA
epochs	40	40
loss	MSE	MSE
GPU	RTX 4090	RTX 4090
optimizer	AdamW	Adam
learning rate	10^{-3}	3×10^{-4}
schedule	warmup, cosine	cosine
weight decay	10^{-2}	0
precision	float16	bfloat16
checkpoint	validation VRMSE	validation MSE
sample	frame pair	stencil pair
samples per epoch	40,000	13.6 M
batch	4 frames	64 stencils
channels in, out	5, 5	4 + conditioning, 4
target	next frame	patch residual

Over forty epochs, the whole-frame U-Net takes 400 thousand gradient steps at a batch of 4 frames and trains in 32 hours, and the PARA U-Net takes 8.5 million steps at a batch of 64 stencils and trains in 85 hours (Table 9).

Table 9: **Training cost at 40 epochs over the 40,000 training pairs behind Table 1.** Whole frame trains through the Flowers repository at batch 4 frames, 10,000 steps per epoch. PARA trains at batch 64 stencils, 212,500 steps per epoch, since one pair yields 340 stencils over the four levels. The PARA scOT (†) trains at batch 512 and eager as it is launch bound at batch 64. Step is the wall clock between consecutive gradient steps of the same epoch, validation excluded. Wall time includes the validation pass of every epoch. The whole-frame rows train eager, the scOT with mixed precision off (‡) and the other three under Flowers' float16. PARA trains under bfloat16 with the network compiled, the scOT excepted. All runs on one RTX 4090 NVIDIA card.

model	parameters	wall time (h)	step (ms)	steps
Whole Frame				
CNextU-Net	18,571,565	15.4	137.1	400,000
Flower	17,336,565	24.0	214.7	400,000
scOT ‡	17,706,166	24.3	217.8	400,000
U-Net	17,518,661	31.9	284.7	400,000
PARA				
CNextU-Net	18,572,320	70.1	29.6	8,500,000
U-Net	17,521,156	85.1	35.9	8,500,000
Flower	17,329,684	85.5	36.1	8,500,000
scOT †	17,706,964	90.7	302.8	1,075,000

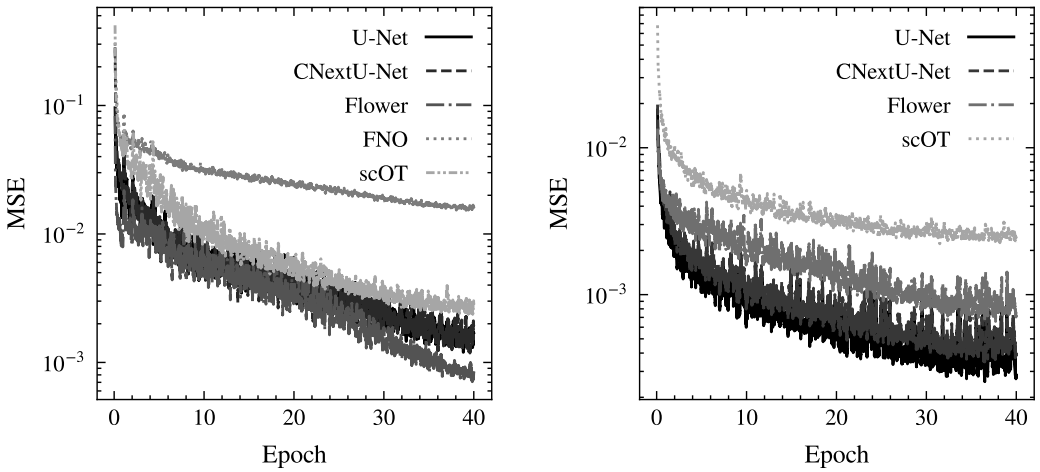


Figure 8: **Training loss of the forty-epoch runs of Table 1.** Left, whole frame, the mean square error of the statistically normalized next frame. Right, PARA, the mean square error of the statistically normalized change of the patch. Each curve is the training loss averaged over a tenth of an epoch, on a log axis against the epoch. Curves compare within a panel and not across, since the two panels measure different quantities in different units, and an epoch is 10,000 gradient steps on the left and 212,500 on the right, 26,875 for the PARA scOT at batch 512.

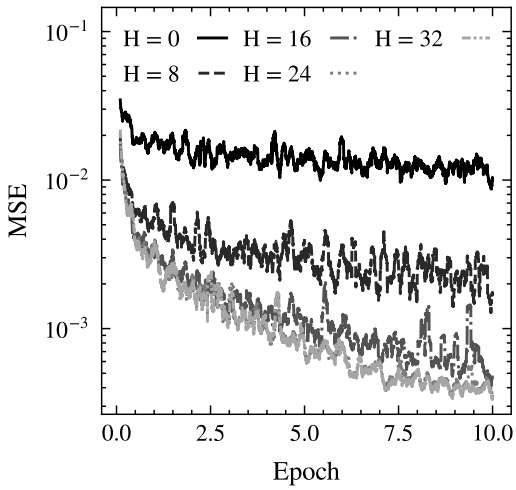


Figure 9: **Training loss of the halo width ablation.** The ten-epoch PARA U-Net at $H = 0$, 8, 16, 24 and 32, the mean square error of the statistically normalized change of the patch, averaged over a tenth of an epoch, on a log axis against the epoch. The runs at $H = 24$ and 32 train without level 1, whose 64 cells are narrower than their stencil, so their loss is over 336 stencils per pair against 340. Without a halo the loss stops falling within the first epoch, at $H = 8$ it falls to a fifth of that, and from $H = 16$ the three curves lie on top of each other, the same saturation as the VRMSE of Table 3.

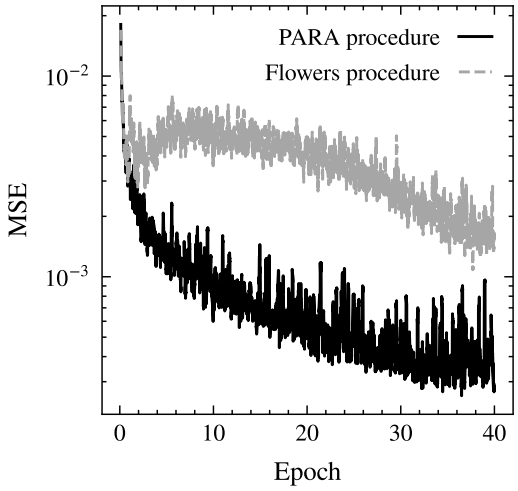


Figure 10: **Training loss of the PARA U-Net under the two procedures.** The same network and stencils, $P = 32$, $H = 16$, forty epochs, the mean square error of the statistically normalized change of the patch, averaged over a tenth of an epoch, on a log axis against the epoch. Under the whole-frame procedure the loss climbs through the five epochs of warmup as the rate rises to 10^{-3} and falls from there, ending about five times above the PARA procedure's, the gap of Table 15.

A.3.3 Evaluations

We report the variance-scaled root mean squared error (VRMSE) (Ohana et al., 2024), which normalizes the error of each field by the variance of its ground truth. This makes errors comparable across fields of different magnitude, such as density and energy. For a predicted field v and ground truth u,

$$\mathrm{VRMSE}(u, v) = \left(\frac{\langle |u - v|^2 \rangle}{\langle |u - \bar{u}|^2 \rangle + \epsilon} \right)^{1/2}, \tag{4}$$

where $\langle \cdot \rangle$ denotes the mean over the spatial domain, $\bar{u}$ is the spatial mean of u, and $\epsilon = 10^{-7}$ prevents division by zero when the ground truth is spatially constant.

Every frame of the test set is predicted one step from the ground-truth frame before it, 5000 pairs. The ground truth becomes a TreeGrid with every patch at one level l, the network advances it, and the prediction is the finest level of the completed tree, so every level is scored on the same 512^2 grid. We score the Well's five fields, density, the two momenta, energy and pressure, with the Well's implementation, and report the mean over fields and pairs. The whole-frame networks predict the pressure directly, while PARA recovers it from its four predicted conserved fields through the ideal gas law.

A.4 Architecture

We use four backbones, the U-Net (Ronneberger et al., 2015), CNextU-Net (Ohana et al., 2024), Flower (Muser et al., 2026), the best published whole-frame model on it, and scOT (Herde et al., 2024). We do not use AMR-Transformer (Xu et al., 2025).[2]

A network is run in one of two configurations. Whole-frame, it takes the five fields on the 512^2 frame and predicts the next frame. In PARA, it takes the four conserved fields on the 64^2 stencil, $P = 32$ and $H = 16$, with $L = 4$ one-hot level channels and predicts the residual of stencil. Which networks run in which configuration is stated with each result. Training details are in Appendix A.3.2.

[2]Its Shockwave results are on an unreleased 128^2 dataset from an unnamed solver, and the released code uses only the velocity-gradient criterion of the paper's four, so neither its numbers nor its method can be reproduced. Its Figure 1 gains involve no learned component. They compare an adaptive partition of a 1024^2 field against a regular 512^2 grid, both scored against the field itself, and a quadtree of block means alone reproduces the trade-off.

A.4.1 U-NET

The U-Net (Ronneberger et al., 2015) is an encoder-decoder architecture with a bottleneck at the coarsest resolution, and each encoder stage above it connected by a skip connection to the decoder stage at the same resolution. Our U-Net spans four resolutions. Each of our encoder and decoder stages is composed of a block of two zero-padded 3×3 convolutions, each followed by group normalization with two groups (Wu & He, 2018) and a Gaussian error linear unit (GELU) (Hendrycks & Gimpel, 2016). The first encoder stage lifts the input channels to $W = 96$. Each later encoder stage halves the resolution by 2×2 max pooling and doubles the channels. Each decoder stage doubles the resolution by bilinear upsampling, concatenates the encoder output at that resolution, and halves the channels. A final 1×1 convolution with no activation maps the $W = 96$ channels to the output fields.

Table 10 gives the whole-frame and PARA forms. The whole-frame U-Net takes the five fields, ρ, $\mathbf{m}$, E and p, on the 512^2 frame and returns the five fields on the next frame. The PARA U-Net takes the four conserved fields, ρ, $\mathbf{m}$ and E, on the stencil with the conditioning channels appended, four one-hot level channels under statistical normalization or one channel of τ under scale normalization, and returns the residual of the four fields on the stencil, of which the patch is kept. The two networks differ only in the first and last convolutions, which is the small gap in their parameter counts.

Table 10: The U-Net in its whole-frame and PARA forms. Every stage is two rounds of a zero-padded 3×3 convolution, group normalization and GELU. The two differ only in the input and output channels and in the resolutions the four stages span. The PARA column is shown for $P = 32$ and $H = 16$ under statistical normalization, whose input is $(5, 64, 64)$ under scale normalization.

Hyperparameter	whole frame	PARA
input	$(5, 512, 512)$	$(8, 64, 64)$
output	$(5, 512, 512)$	$(4, 64, 64)$
resolutions	512^2, 256^2, 128^2, 64^2	64^2, 32^2, 16^2, 8^2
channels	96, 192, 384, 768	96, 192, 384, 768
parameters	17,518,661	17,521,156

A.4.2 CNEXTU-NET

The CNextU-Net (Ohana et al., 2024) is the U-Net shape with ConvNeXt blocks (Liu et al., 2022b) in place of the convolution pairs. A block is a depthwise 7×7 convolution, layer normalization (Ba et al., 2016) over the channels, a linear layer expanding the channels fourfold, GELU, a linear layer back, a learned per-channel scale initialized at 10^{-6}, and a residual addition. A 3×3 convolution lifts the input channels to $W = 42$. The encoder has four stages of two blocks, each stage ending in a normalization and a 2×2 stride-2 convolution that halves the resolution and doubles the channels. One block sits at the neck. The decoder mirrors the encoder, each stage beginning with a normalization and a 2×2 stride-2 transposed convolution that doubles the resolution and halves the channels, the encoder output at that resolution concatenated and reduced by a 1×1 convolution before its two blocks. A final 3×3 convolution maps the $W = 42$ channels to the output fields. Our stencil version is the Flowers repository's port of the Well's model (Muser et al., 2026) reduced to two dimensions, without conditioning, stochastic depth or gradient checkpointing, and otherwise as published.

Table 11 gives the whole-frame and PARA forms. The whole-frame CNextU-Net takes the five fields, ρ, $\mathbf{m}$, E and p, on the 512^2 frame and returns the five fields on the next frame. The PARA CNextU-Net takes the four conserved fields, ρ, $\mathbf{m}$ and E, on the stencil with the conditioning channels appended, four one-hot level channels under statistical normalization or one channel of τ under scale normalization, and returns the residual of the four fields on the stencil, of which the patch is kept. The two differ only in the first and last convolutions.

A.4.3 FLOWER

Flower (Muser et al., 2026) is a U-Net architecture whose blocks are built on a self-warp instead of a convolution. Its block is a self-warp, a 1×1 residual projection, group normalization with 40 groups and GELU. The self-warp predicts a displacement field per head from its input with two 1×1

Table 11: The CNextU-Net in its whole-frame and PARA forms. Every stage is two ConvNeXt blocks, one at the neck. The two differ only in the input and output channels and in the resolutions the stages span. The PARA column is shown for $P = 32$ and $H = 16$ under statistical normalization, whose input is $(5, 64, 64)$ under scale normalization.

Hyperparameter	whole frame	PARA
input	$(5, 512, 512)$	$(8, 64, 64)$
output	$(5, 512, 512)$	$(4, 64, 64)$
resolutions	512^2, 256^2, 128^2, 64^2, 32^2	64^2, 32^2, 16^2, 8^2, 4^2
channels	42, 84, 168, 336, 672	42, 84, 168, 336, 672
parameters	18,570,807	18,572,320

convolutions, projects the input by a 1×1 convolution, and samples the projection at the displaced positions by bilinear interpolation, $u(x) \mapsto u(x + d(x))$, with $W/4 = 40$ heads at $W = 160$. The receptive field is learned, not set by a kernel. Two coordinate channels are concatenated to the fields, and a 1×1 lift takes the result to W. The encoder has three levels of one block, each followed by a 2×2 stride-2 convolution and ReLU that halves the resolution and doubles the channels, then one block at the bottleneck. The decoder mirrors it, one block at the coarse resolution that reduces the channels to those of the encoder output at the next resolution, a 2×2 stride-2 transposed convolution and ReLU that doubles the resolution, then that encoder output concatenated, which doubles the channels. Two 1×1 convolutions with a ReLU between map the $2W$ channels to the output fields.

Table 12 gives the whole-frame and PARA forms. The whole-frame Flower is the Flowers repository's model. Its warps wrap periodically, the dataset's boundary condition read at build time, and it is conditioned on γ through feature-wise linear modulation, which the unconditioned run needed to train. The stencil version is our port reduced to two dimensions without conditioning or dropout, and with one change for the stencil, its warps do not wrap, a stencil's edge is the halo and not a periodic boundary, so samples beyond it read zero. The whole-frame Flower takes the five fields, ρ, $\mathbf{m}$, E and p, on the 512^2 frame and returns the five fields on the next frame. The PARA Flower takes the four conserved fields, ρ, $\mathbf{m}$ and E, on the stencil with the conditioning channels appended, four one-hot level channels under statistical normalization or one channel of τ under scale normalization, and returns the residual of the four fields on the stencil, of which the patch is kept. Both forms also take two coordinate channels.

Table 12: Flower in its whole-frame and PARA forms. Every level is one block of a self-warp with 40 heads, group normalization and GELU. The PARA column is shown for $P = 32$ and $H = 16$ under statistical normalization, whose input is $(5 + 2, 64, 64)$ under scale normalization.

Hyperparameter	whole frame	PARA
input	$(5 + 2, 512, 512)$	$(8 + 2, 64, 64)$
output	$(5, 512, 512)$	$(4, 64, 64)$
resolutions	512^2, 256^2, 128^2, 64^2	64^2, 32^2, 16^2, 8^2
channels	160, 320, 640, 1280	160, 320, 64
heads	40	40
conditioning	γ, FiLM	none
parameters	17,329,362	17,329,684

A.4.4 SCOT

scOT (Herde et al., 2024) is a Swin-v2 transformer (Liu et al., 2022a) in a U-Net shape, taken from the Flowers repository, which shares one definition between the whole-frame and stencil runs. A 4×4 stride-4 convolution embeds each 4×4 block of cells as one token of width $W = 32$, followed by layer normalization. The encoder has four stages of eight Swin-v2 layers, at widths W, $2W$, $4W$, $8W$ and 2, 4, 8, 16 heads. Each layer is multi-head self-attention within 7×7 windows of tokens, the windows shifted by three tokens in every second layer, then a two-layer MLP of ratio four with GELU, both with a residual and post normalization, and stochastic depth rising linearly to 0.1.

Position enters through Swin-v2's relative position bias, there is no absolute position embedding. Between stages a patch merging concatenates each 2×2 block of tokens and maps it linearly to twice the width. The decoder mirrors the encoder, a patch unmerging maps each token linearly to a 2×2 block at half the width, and the encoder's output at that resolution is added, not concatenated, after one ConvNeXt block (Liu et al., 2022b) of a 7×7 depthwise convolution, layer normalization and an MLP of ratio four. A 4×4 stride-4 transposed convolution recovers the cells from the tokens and a 5×5 convolution mixes them.

Table 13 gives the whole-frame and PARA forms. Both are the yaml's setting, width 32, depths 8, 8, 8, 8, window 7, without conditioning. A token grid that is not a multiple of the window is padded with zero tokens, and a grid smaller than the window shrinks the window to the grid with no shift, so the whole frame attends over 7×7 windows at every stage, while the stencil's two coarsest stages, 4×4 and 2×2 tokens, attend over the whole grid. Neither version wraps at the frame's edge. The whole-frame scOT trains in full precision, since under the whole-frame procedure's float16 mixed precision its attention over 128×128 tokens overflows and the loss scaler skips every step. The stencil scOT trains at batch 512 in place of 64, since the network is launch-bound at the stencil's 256 tokens and 64 would have taken 22 days. The whole-frame scOT takes the five fields, ρ, $\mathbf{m}$, E and p, on the 512^2 frame and returns the five fields on the next frame. The PARA scOT takes the four conserved fields, ρ, $\mathbf{m}$ and E, on the stencil with the conditioning channels appended, four one-hot level channels under statistical normalization or one channel of τ under scale normalization, and returns the residual of the four fields on the stencil, of which the patch is kept.

Table 13: scOT in its whole-frame and PARA forms. Every stage is eight Swin-v2 layers with 7×7 windows, and the skips pass through one ConvNeXt block. The PARA column is shown for $P = 32$ and $H = 16$ under statistical normalization, whose input is $(5, 64, 64)$ under scale normalization.

Hyperparameter	whole frame	PARA
input	$(5, 512, 512)$	$(8, 64, 64)$
output	$(5, 512, 512)$	$(4, 64, 64)$
token grids	128^2, 64^2, 32^2, 16^2	16^2, 8^2, 4^2, 2^2
widths	32, 64, 128, 256	32, 64, 128, 256
heads	2, 4, 8, 16	2, 4, 8, 16
window	7	7, 7, 4, 2
precision	float32	bfloat16
batch	4 frames	512 stencils
parameters	17,706,166	17,706,964

A.5 Additional results

A.5.1 One-step error distribution

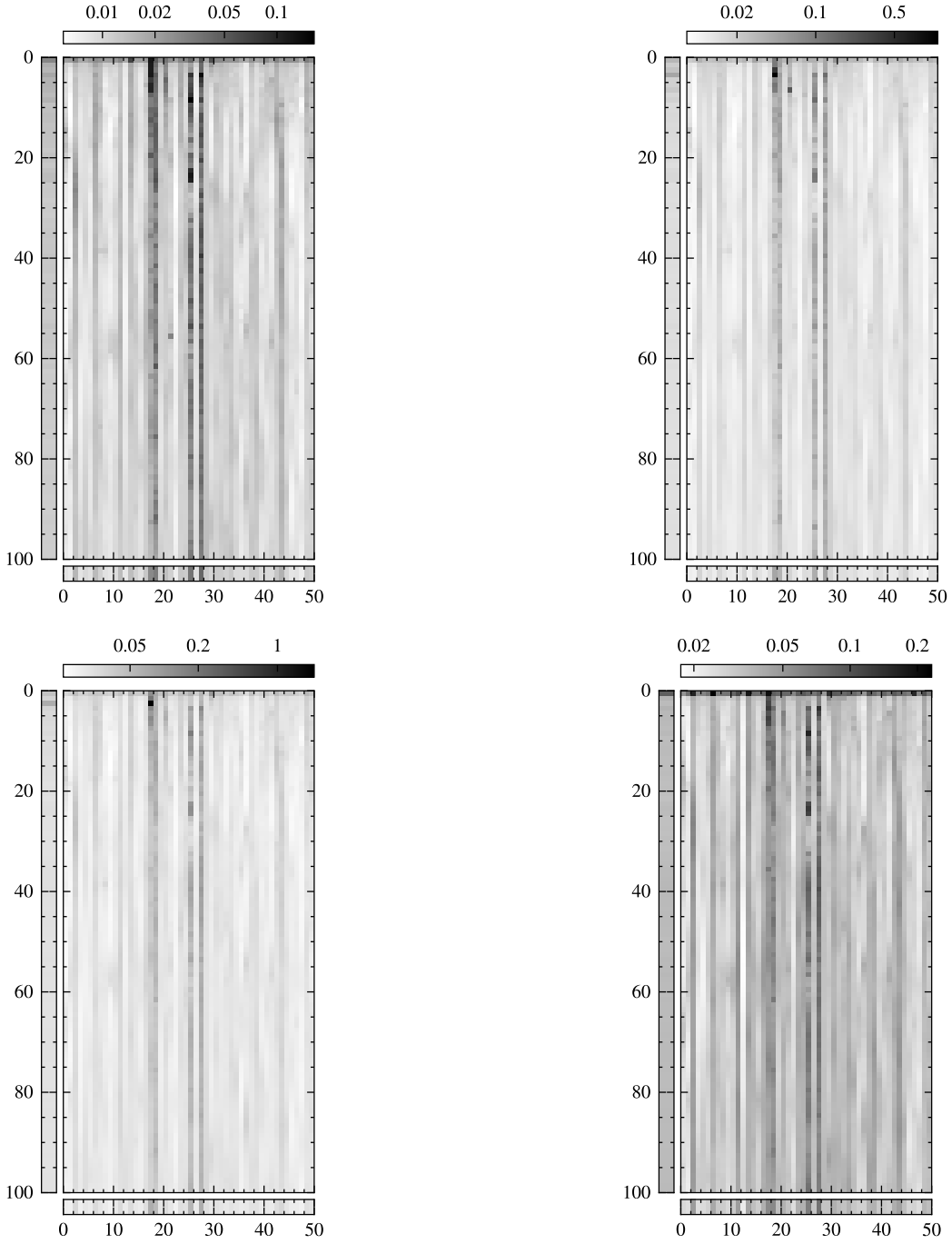


Figure 11: **One-step VRMSE of every test pair for the four PARA backbones of Table 1.** Top left U-Net, top right CNextU-Net, bottom left Flower, bottom right scOT. Each map holds the 50 test trajectories along the horizontal axis and their 100 frames down the vertical, one cell per pair, level 4, on a log color scale. Every prediction starts from the true frame, so the vertical axis is the frame index and not a rollout step. The strip below a map is the mean over frames per trajectory, the strip to its left the mean over trajectories per frame, on the map's color scale. Each map's color range is its own, from its smallest to its largest VRMSE.

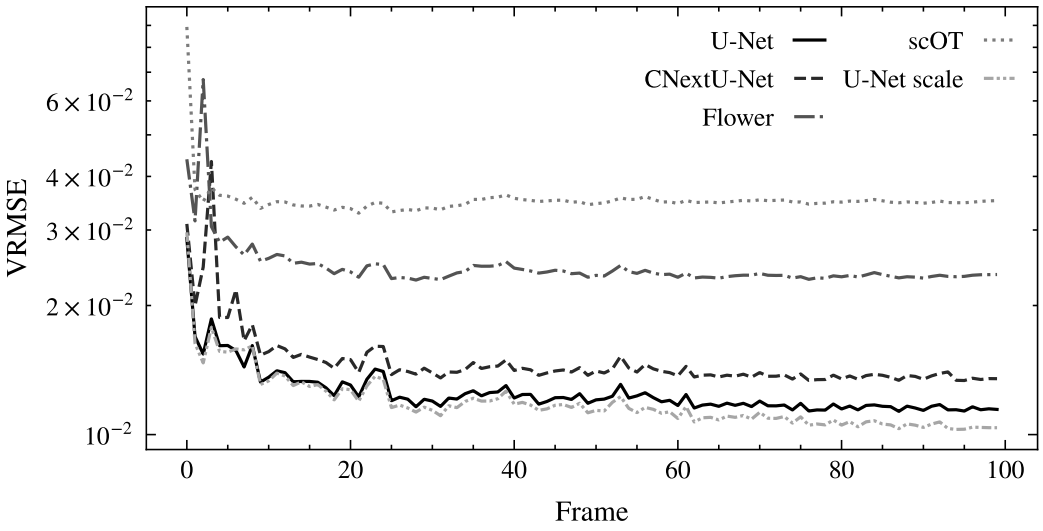


Figure 12: **One-step VRMSE by frame index.** The mean over the 50 test trajectories of the level-4 VRMSE at each frame, for the four PARA backbones of Table 1 under statistical normalization and the U-Net under the data-free scale normalization of Table 4. Every prediction starts from the true frame, so the horizontal axis is the position in the trajectory and not a rollout step. Log scale.

A.5.2 ROLLOUT

Table 14: **Rollout VRMSE, mean and median.** The medians of Table 2, over the same 50 test trajectories and the frames of each range. A mean far above its median marks a few diverging trajectories, as for the PARA scOT over frames 1 to 20. The published rows are medians over test trajectories on all ten γ of one rollout from frame 17, run through both frame ranges without restart.

	Whole frame		PARA	
Backbone	mean	median	mean	median
Frames 1–20				
U-Net	0.651	0.644	0.110	0.081
CNextU-Net	0.700	0.689	0.144	0.100
Flower	0.207	0.181	0.169	0.151
scOT	0.634	0.624	1.46	0.178
Published, all γ				
Walrus, 6 frames (McCabe et al., 2026)	–	0.0720	–	–
Poseidon-L, 1 frame (McCabe et al., 2026)	–	0.0682	–	–
Frames 21–60				
U-Net	0.761	0.794	0.180	0.162
CNextU-Net	0.830	0.831	0.222	0.210
Flower	0.320	0.311	0.297	0.290
scOT	0.770	0.793	0.300	0.246
Published, all γ				
Walrus, 6 frames (McCabe et al., 2026)	–	0.3408	–	–
Poseidon-L, 1 frame (McCabe et al., 2026)	–	0.2807	–	–

A.5.3 SENSITIVITY MAPS

Figure 4 shows how the trained network uses its halo, through the gradient of the patch residual with respect to each input cell, averaged over the ring of cells at each distance from the patch (Figure 13). For $H \geq 16$ the sensitivity falls by an order of magnitude over 16 cells, and the curves coincide where they overlap, so the network weights a cell by its distance and not by its halo width. For

$H = 8$, where 5 percent of patches are contained, the sensitivity is flat over the halo and rises toward its edge, so the network leans on the boundary of what it sees. Every curve rises at its halo edge, from a factor of two at $H = 8$ to a small tail at $H = 32$, a trace of the domain of dependence the stencil cuts off.

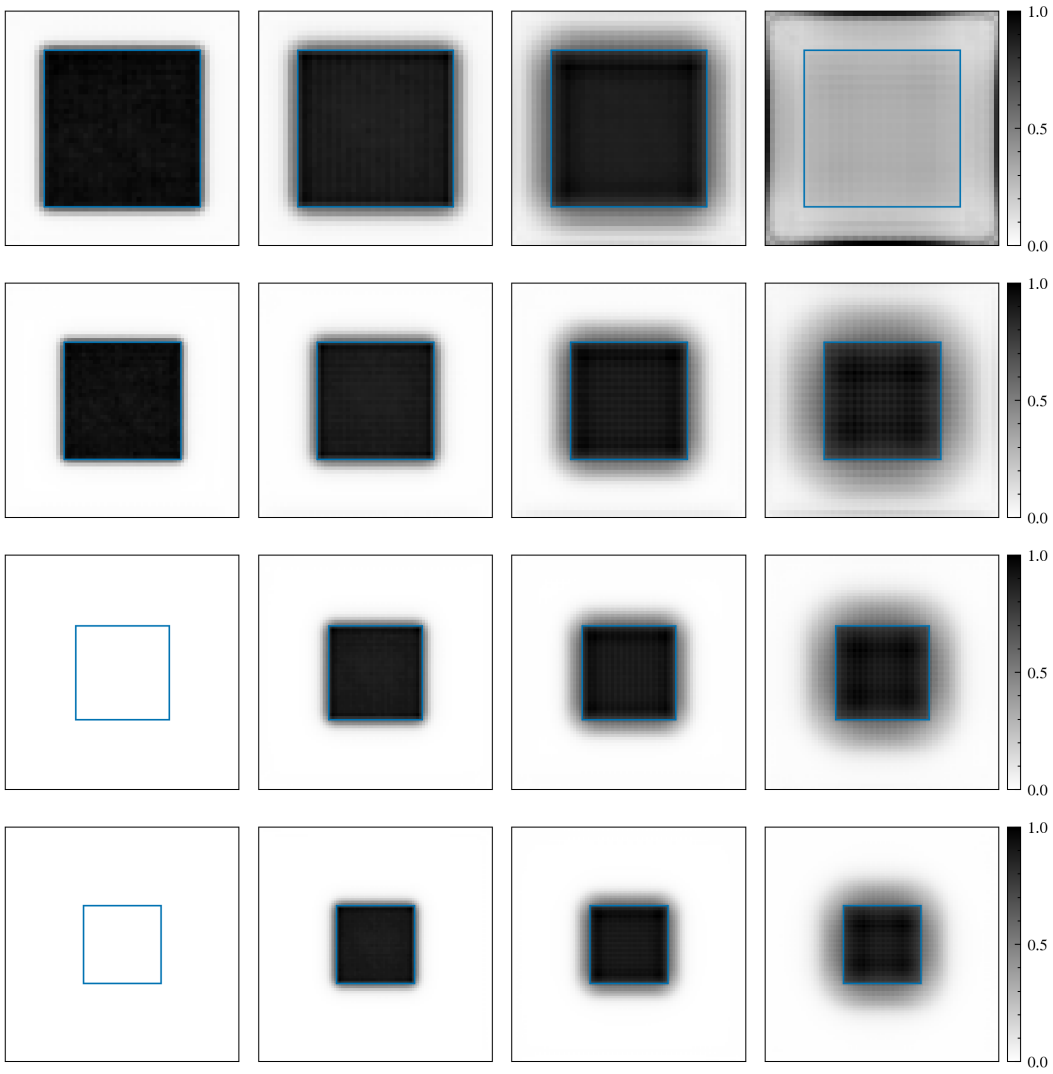


Figure 13: **Sensitivity maps of the PARA U-Net** at ten epochs for the halo widths of Table 3, $H = 8$, 16, 24 and 32 top to bottom, levels 1 to 4 left to right. Level 1 is absent for $H \geq 24$, where the stencil is wider than the level. Each map is the absolute gradient of the patch residual's L_1 norm with respect to each input cell, averaged over the four input fields and over every stencil of that level in 500 test frames, and divided by its own peak. The blue square marks the patch. Figure 4 averages the level-4 maps over square rings outward from the patch, with $r = 0$ the patch's own boundary ring.

A.5.4 ADAPTIVE ROLLOUT

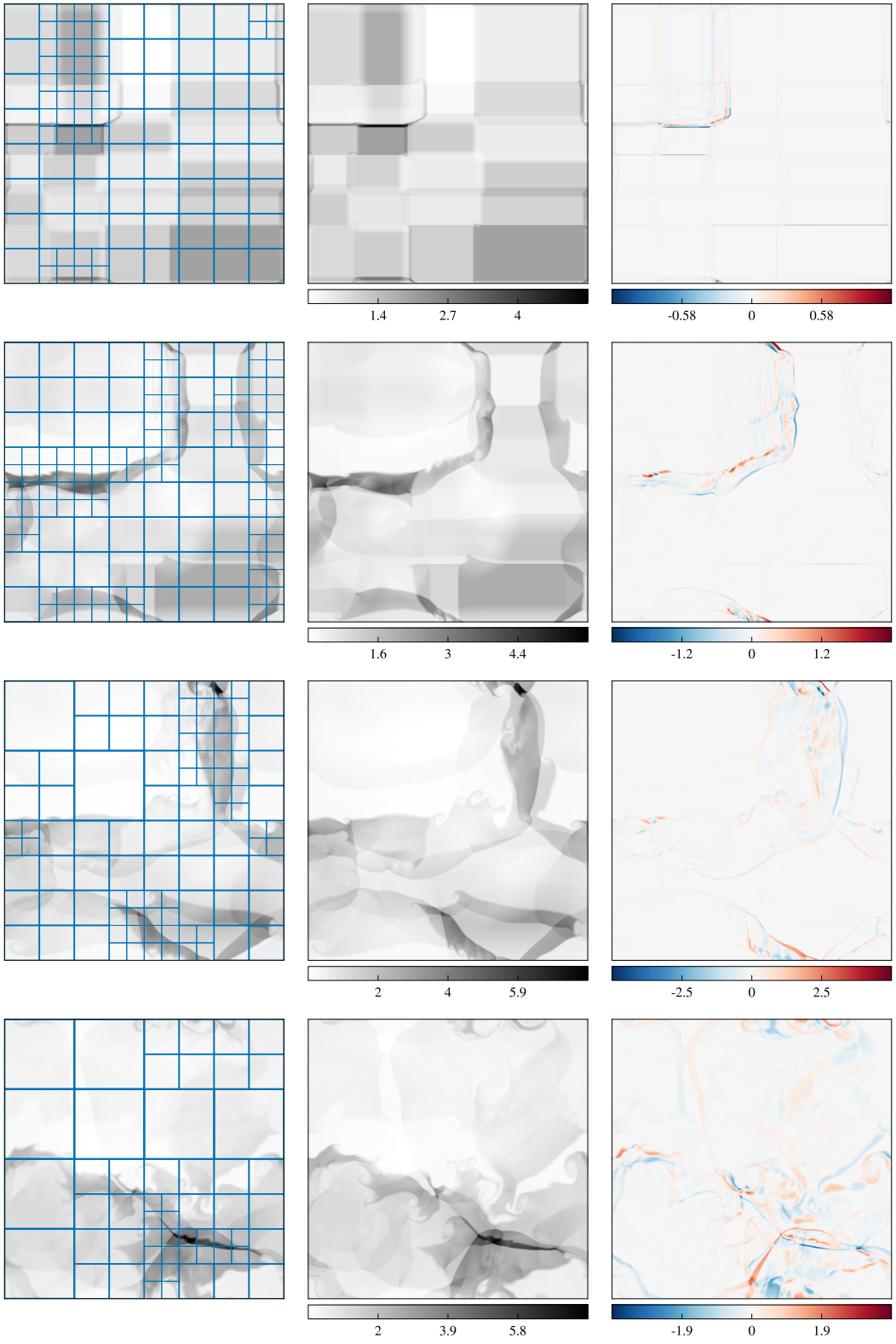


Figure 14: **Adaptive rollout, frames 1 to 20.** The scale-normalized U-Net on test trajectory 17 from the truth at frame 0, at frames 1, 5, 10 and 20, top to bottom. Left the predicted density with the layout of that step, thick lines for coarse patches and thin for fine, middle the truth, right their difference. The thresholds are 0.01, 0.05 and 0.15. Each row has its own color range.

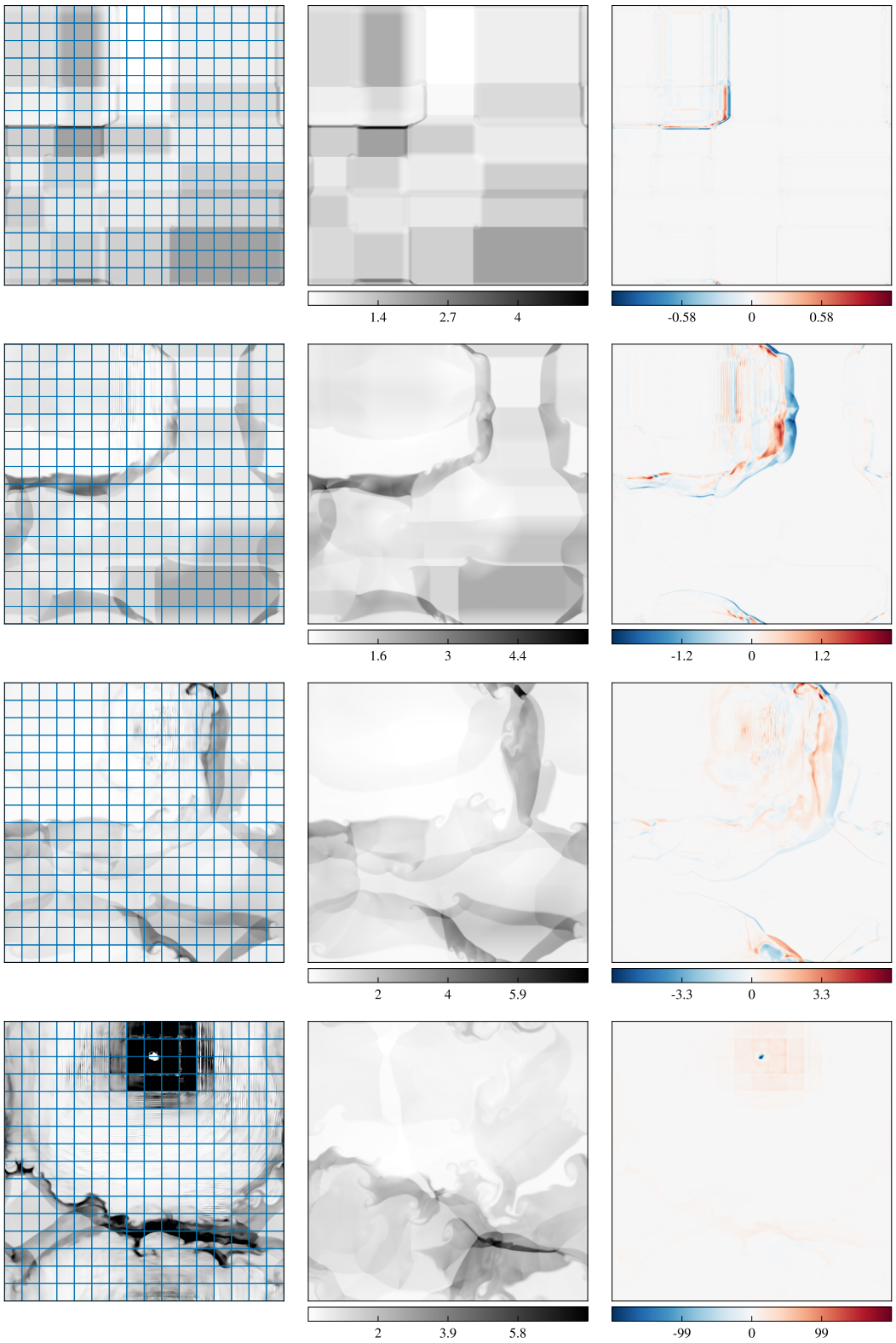


Figure 15: **The scale-normalized U-Net's uniform rollout diverging.** Test trajectory 17 from the truth at frame 0, the density at frames 1, 5, 10 and 20, top to bottom, on the uniform grid. Left the prediction, middle the truth, right their difference, each row on its own color range. One-cell stripes appear in the smooth low-density region behind the leading front by frame 5 and grow into the divergence of Table 5.

A.5.5 TRAINING PROCEDURE AND SEED

Table 15: **PARA has the lower error under either training procedure.** One-step VRMSE on the test split at the finest level of the same four-scale U-Net at 17.5 M parameters, trained for forty epochs under each training procedure of Table 8. Down a column the training procedure is shared, and PARA lowers the error by a factor of 1.5 under the whole-frame procedure and 2.7 under its own. Along a row the form is shared, and the training procedure changes the whole-frame error by 7 percent and the PARA error by a factor of 2.0.

Form	Whole-frame procedure	PARA procedure
Whole frame	0.0366	0.0342
PARA	**0.0248**	**0.0126**

Table 16: **Seed variation.** One-step and rollout VRMSE of the whole-frame U-Net and the scale-normalized PARA U-Net, each trained with two seeds, on the uniform grid. Seed 1 is the run reported in the main text. The last column is seed 1 minus seed 2 as a percentage of seed 1, negative where seed 2 has the higher error. The large difference of the scale-normalized U-Net over frames 1 to 20 comes from the divergence of its uniform rollout on trajectory 17 (Table 5).x

	Seed 1	Seed 2	Difference (%)
Whole-frame U-Net			
One step	0.0366	0.0352	3.83
Frames 1–20	0.6514	0.6267	3.79
Frames 21–60	0.7606	0.8837	−16.18
PARA U-Net, scale normalization			
One step	0.0120	0.0120	−0.11
Frames 1–20	3.4748	0.7100	79.57
Frames 21–60	0.1706	0.1815	−6.39