ADAPTIVE TOKENIZATION FOR VISION TRANSFORMER PDE SIMULATION

Hanwen Wang
Graduate Group of Applied Mathematics
and Computational Science
University of Pennsylvania
Philadelphia, PA, 19104
wangh19@sas.upenn.edu

Paris G. Perdikaris
Department of Mechanical Engineering
and Applied Science
University of Pennsylvania
Philadelphia, PA, 19104
pgp@seas.upenn.edu

ABSTRACT

Vision Transformers have been increasingly adopted as neural-operator surrogates for simulating physical systems governed by partial differential equations (PDEs). However, the quadratic computational complexity of self-attention with respect to the number of tokens limits their scalability to fine discretizations in space and time. To address this challenge, we present Iterative Adaptive Patchification (IAP), a novel fixed-budget adaptive patching scheme tailored for ViT-based autoregressive PDE simulation. IAP dynamically allocates tokens to regions of high complexity, while enforcing a strict global token budget. This design ensures that training and inference costs can be flexibly adjusted during training or inference. We validate IAP on a suite of 2D and 3D PDE benchmarks, demonstrating that it achieves performance comparable to vanilla ViTs with fixed patch sizes across a range of token budgets, and outperforms the vanilla ViT baseline significantly on one 3D benchmark. The analysis of the compute-accuracy tradeoff also shows that the model is robust against post-training budget adjustments. The partial codebase [1] for IAP implementation is available on Github[2].

1 INTRODUCTION

Deep learning has rapidly emerged as a promising paradigm for simulating and forecasting physical systems governed by partial differential equations (PDEs). Traditional numerical solvers, though accurate, are often prohibitively expensive for long horizons or high-dimensional domains. They also rely on hand-crafted adjustments that are specific to the problems. In contrast, neural surrogates such as physics-informed neural networks (PINNs)(Raissi et al., 2019), DeepONets(Lu et al., 2021; Wang et al., 2021), Fourier Neural Operators (FNOs)(Li et al., 2021), and other operator-learning frameworks(Alkin et al., 2024; Wang et al., 2025) have shown the ability to approximate PDE dynamics at significantly lower cost, thanks to the advancement of parallel computing software and hardware, as well as the generalizability without excessive hand-crafting, from the flexibility of the deep learning architecture. With neural operators, autoregressive sequence models are particularly attractive: by iteratively predicting the next state from previous ones, they provide a natural framework for long-horizon PDE rollout, enabling both forecasting and acceleration of classical solvers. Recently, Vision Transformers (ViTs)(Kolesnikov et al., 2021) have proven themselves as strong candidates for this task. Their self-attention mechanism(Vaswani et al., 2017) captures global correlations across both spatial and temporal domains, allowing them to model both local nonlinear interactions and long-range transport phenomena. Yet, this expressivity comes at the price of quadratic complexity with respect to the sequence length, especially for PDE simulations with fine discretization or high dimensionality. Thus, the scalability bottleneck of dot-product attention remains a fundamental obstacle to deploying ViTs for high-resolution PDE forecasting. While various efficient attention approximations, such as linear attention (Katharopoulos et al., 2020), have

[1] As an ongoing work, we only release code that is crucial for the perception of the IAP method to foster understanding.

[2] https://github.com/PredictiveIntelligenceLab/iterative_adaptive_patchification.git

been proposed in the NLP and vision communities and adopted for PDE modeling(Sankaran et al., 2025), we tackle this challenge from a different perspective: instead of approximating the attention mechanism, we propose to adaptively reduce the sequence length by patchifying the input into variable-sized patches that are dynamically allocated according to the local complexity of the dynamics. This approach allows us to maintain the full expressivity of dot-product attention while significantly reducing the computational burden brought by the forceful fixed patch size, enabling ViTs to scale to higher resolutions and dimensions in PDE simulation tasks.

Background and Related Work. Prior work has attempted to alleviate this difficulty by introducing adaptivity at the patchification stage. Recursive strategies inspired by Adaptive Mesh Refinement (AMR) (Berger & Oliger, 1984; Berger & Colella, 1989) subdivide patches until local indicators, such as edge counts, entropy, gradient fluctuation, etc., fall below a tolerance threshold. These methods begin with a set of large patches covering the entire input domain and recursively subdivide existing patches into equal-sized children according to vision-based or physics-informed criteria. For example, the AMR Transformer (Xu et al., 2025) leverages domain-specific physical indicators such as velocity gradients, vorticity, etc, to determine whether a patch should be further refined. The Adaptive Patching Framework (APF) (Zhang et al., 2024) adopts an image-processing heuristic, applying the Canny edge detector to guide recursive subdivision. The Adaptive Patch Transformer (APT)(Choudhury et al., 2025) utilizes entropy as a complexity measure. Likewise, MATEY (Zhang et al., 2024) uses variance within the patch as an indicator to determine whether a patch should be further refined. Learnable scoring approaches, such as the Differentiable Dynamic Adaptive Region Tokenizer (DART)(Yin et al., 2025), have also been proposed to enable end-to-end training of the patchification process with a learned scoring network. Although these methods effectively concentrate resolution where the dynamics are complex, they produce sequence lengths that vary across samples, complicating batching, memory allocation, and stability during autoregressive rollout on hardware accelerators that require statically shaped tensors. During the training, sequences of different lengths have to be padded to the longest length for the batch processing of the backbone, effectively negating the benefit of the reduction of the sequence length. Furthermore, the subdivision decision of these methods is based on a local threshold, which, as a hyperparameter, requires careful tuning per dataset and normalization so that the number of patches per sample falls into a reasonable range. This sensitivity to the threshold can limit the robustness of the patchification across different datasets and timesteps, especially for PDE simulations where the dynamics can vary significantly across time and space.

Other architectures such as the Perceiver (Jaegle et al., 2021; Lee et al., 2019; Jaegle et al., 2022) circumvent this issue by maintaining a fixed pool of latent learnable tokens that cross-attend to the inputs. While this yields a constant compute budget, the latent tokens are randomly initialized and independent of the input structure, which can limit their ability to capture fine-scale, data-dependent features that are largely local. On the other hand, the Controllable Patching from Mukhopadhyay et al. (2025) uses the convolution-based patch embeddings of different strides and kernel sizes, during training, to introduce a set of token budgets and train the backbone to be robust against different token counts, to introduce the compute elasticity at inference time. However, the patchification is still fixed and non-adaptive to the local dynamics.

To address these limitations, we propose the Iterative Adaptive Patchification (IAP), a fixed-budget patching strategy. Instead of recursively subdividing until thresholds are satisfied, IAP iteratively selects the most informative patches—those whose image gradient norms are highest—and refines them. In this way, IAP partitions the image into a fixed number of patches, while still dynamically allocating tokens to regions of different behavior. Furthermore, unlike Perceiver-style models where the latent tokens are free parameters, IAP produces tokens that remain closely tied to the underlying image content. By combining gradient-driven adaptivity with budgeted refinement, IAP unifies the efficiency of adaptive patching with the scalability demands of transformer architectures.

Contributions.

1. IAP uses a global refinement procedure, where the subdivision decision of individual patch is made by comparing its score with other patches, instead of using a local threshold. This relative subdivision criterion removes the burden of tuning the threshold for each dataset, and allows the patchification to be more robust across different datasets and timesteps.

2. The number of tokens is strictly controlled with no variance, and IAP allows real time adjustment of it. The model can also be trained and inferred with different token budgets without retraining.
3. For one 3D benchmark with localized dynamics, we show that the IAP significantly outperforms the ViT with fixed patch size, with far fewer tokens.

Next, we detail the IAP patchification algorithm and the cross-attention decoder used to reconstruct full-resolution predictions.

2 METHOD

Let the input be $X \in \mathbb{R}^{H\times W}$ where H, W are the spatial dimensions; the method extends directly to D spatial dimensions. We constrain the maximum patch size to be $p_{\max}$, the minimum patch size to be $p_{\min}$, and the total number of patches to be $T_{\max}$.

Gradient-based splitting score. Inspired by the classic Canny edge detector(Canny, 1986), we define a per-pixel splitting score s using a multidimensional Sobel operator. The Sobel operator uses finite differences to calculate the image gradient along the given axis over the neighbors of the evaluated pixel, and uses a weighted average to smooth it. Appendix B details the computation of the per-pixel score map $s \in \mathbb{R}^{H\times W}$. We assume that regions with higher image gradients are more likely to contain complex dynamics and thus deserve finer patches.

Given a patch P with integer vertices $v = (v_0, v_1)$ and edge lengths $\ell = v_1 - v_0$, we define the patch score as the sum of the per-pixel score within the patch P, i.e., $S(P) = \sum_{u\in P} s(u)$. Patches whose edge lengths are less than or equal to $p_{\min}$ are not eligible for further splitting, and their scores are set to $-\infty$.

IAP starts by partitioning the image into non-overlapping patches of size $p_{\max} \times p_{\max}$, yielding an initial patch count of $T_0 = \frac{H\cdot W}{p_{\max}^2}$. Then, at each refinement step, the top-K patches with the highest scores are selected and split into 2^D children at their midpoints. The scores of the new children are computed and inserted back into a preallocated, score-sorted list of candidate patches. This process is repeated for a fixed number of iterations until the target token budget $T_{\max}$ is reached. Algorithms 1 and 2 in Appendix A summarize the entire IAP procedure. An example of the IAP partitioning result is shown in Figure 1c. Once the input grid is partitioned into variable-sized

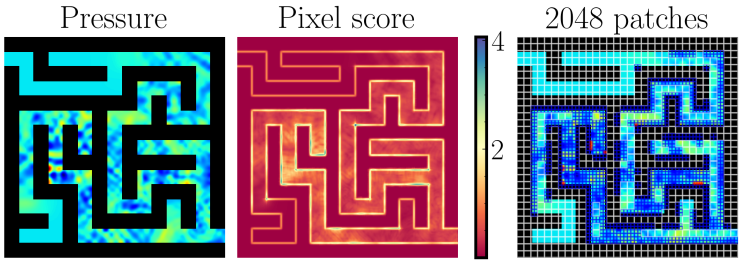


Figure 1: A test sample (Figure 1a) from `acoustic_scattering_maze` illustrating the gradient map and the Iterative Adaptive Patchification outcome. Figure 1b shows the estimated image-gradient norm computed with the Sobel kernel. Figure 1c shows the IAP partition with 2048 variable-sized patches.

patches, we embed each patch into a token of uniform embedding size using patch embeddings for different sizes, denoted as $\mathcal{V} \in \mathbb{R}^{|\mathcal{V}|\times F}$, where $|\mathcal{V}|$ is the number of tokens and F is the embedding dimension. We use the centroid of each patch as its position, which is then encoded using the Rotary Position Embedding (RoPE) method(Su et al., 2024) for positional augmentation.

IAP introduces additional computation due to dynamic patch selection, which relies on a heap-like data structure Williams (1964) that is difficult to implement efficiently on GPUs. This creates a gap between the theoretical compute advantage (Appendix F) and observed wall-clock time (Ap-

pendix G); however, this overhead is amortized by the attention and MLP computations as the backbone scales, making it asymptotically negligible for larger models.

Cross-Attention Decoder. PDE simulations often exhibit time-varying dynamics, making it unsafe to assume that the dynamics remain confined to a fixed region. To avoid this assumption, we employ a cross-attention decoder that operates at the minimal patch resolution. Given token embeddings $\mathcal{V} \in \mathbb{R}^{|\mathcal{V}| \times F}$, we initialize a *regular latent grid* of embeddings $V_o \in \mathbb{R}^{H' \times W' \times F}$, where $H' = H/p_{\min}, W' = W/p_{\min}$, i.e., each latent embedding corresponds to a $p_{\min} \times p_{\min}$ region in the input. Thus, each latent embedding $V_o(i, j) \in \mathbb{R}^F$ corresponds exactly to a unique $p_{\min} \times p_{\min}$ region in the input, ensuring that the initial tiling of the latent grid is well defined. We further add positional encodings to V_o based on its relative position within the enclosing IAP patch. The complete decoder query construction via tiling can be found in Appendix C.

Finally, the latent grid V_o is refined through *cross-attention* with the token set V, and then projected back to reconstruct the output image of the full resolution. This process resembles the decoder design in CViT(Wang et al., 2025), however, we remove the interpolation-based feature extraction for simplicity. From now on, we denote the ViT with IAP head and cross-attention decoder as IAP. A theoretical compute analysis of the IAP architecture is provided in Appendix F. Given the fixed token budget, and size S for the D-dimensional input grid, the computational complexity of the IAP backbone is constant, thanks to the fixed token budget. The cross-attention decoder has a complexity of $\mathcal{O}(S^D)$, whereas the ViT with fixed patch size has a complexity of $\mathcal{O}(S^{2D})$. While the compute overhead (Appendix G) still exists since the heap-like data structure is not efficiently implemented on hardware accelerators, it will be amortized by the scaling of the self-attention backbone.

3 EXPERIMENTS

Experiment Setup. We perform experiments on the benchmark problems and splits defined in The Well (Ohana et al., 2024). The suite comprises diverse spatiotemporal simulation domains, each evolving from sharply localized or structured initial conditions under physically motivated PDEs or continuum models. These datasets are thus well suited for evaluating autoregressive prediction models under nonlinear and multi-scale dynamics. We adopt the provided train/test splits and report performance using the canonical metric, Variance-weighted Relative Mean Squared Error (VRMSE), averaged across channels and samples. Several 2D benchmarks, `active_matter` (AM), `acoustic_scattering_maze` (ASM), `turbulent_radiative_layer_2D` (TRL2D), and `viscoelastic_instability` (VI), whose resolutions allow convenient division into 1024 or 4096 regular patches, have been chosen. In addition, we include a 3D benchmark `supernova_explosion_64` (SNE64) to showcase the efficacy of the IAP on 3D problems. Training specifications are summarized in Appendix E.

Results. The test VRMSEs of IAP and ViT are shown in Table 1. IAP achieves comparable performance to ViT when the token counts of the two architectures are close, and in some cases outperforms the corresponding fixed-patch-size ViT baseline. The most prominent improvement over the

Architecture	AM (256^2)		ASM (256^2)		TRL2D (128×384)		VI (512^2)		**SNE64** (64^3)	
	# Tokens	VRMSE	# Tokens	VRMSE	# Tokens	VRMSE	# Tokens	VRMSE	# Tokens	VRMSE
ViT	1024	0.0267	1024	0.0124	768	0.1751	1024	0.1043		
	4096	**0.0201**	4096	**0.0075**	3072	<u>0.1448</u>	4096	<u>0.0815</u>	4096	0.3426
IAP	1024	0.0279	1024	0.0138	1020	0.1577	1024	0.0938	1016	0.2846
	2044	0.0254	2044	0.0103	2040	0.1478	2044	0.0851	2024	<u>0.2204</u>
	4096	<u>0.0209</u>	4096	<u>0.0082</u>	4092	**0.1418**	4096	**0.0775**	4096	**0.1794**

Table 1: Variance-weighted RMSE (lower is better) on The Well benchmarks. Columns report the token budget (left) and VRMSE (right) for each dataset. Best results are bold; second-best are underlined.

ViT baseline comes from the 3D supernova explosion dataset, with a resolution of $64 \times 64 \times 64$. In this case, any change to the regular patch size results in a drastic reduction or increase in the number of tokens. Furthermore, this particular dataset simulates an explosion in a compressed monatomic

ideal gas; the explosion originates near the center of the simulation box and occupies only a small fraction of the total volume. Figure 2 shows that the IAP algorithm captures the explosion by assigning finer patches near the center and coarser patches in the outer regions, which are generally less dynamic.

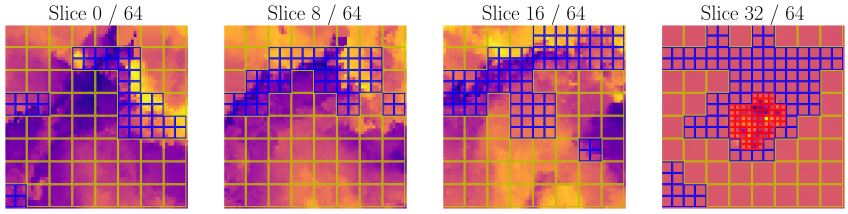


Figure 2: Visualization of the IAP patches for a channel of the `supernova_explosion_64` dataset, at different spatial slices, using an IAP setting with 2048 patches.

4 Summary and Discussion

In this work, we present the Iterative Adaptive Patchification (IAP) scheme, a novel fixed-budget adaptive patching strategy for Vision Transformer-based autoregressive PDE simulation. By iteratively refining the most salient patches based on a gradient-derived score map, IAP dynamically allocates tokens to regions of high complexity while coarsening smooth areas, all while enforcing a strict global token budget. We show that the IAP encoder can achieve comparable performance to the vanilla ViTs with fixed patch sizes on 2D PDE benchmarks, and significantly outperform its ViT counterpart on one 3D benchmark, with far fewer tokens. Crucially, IAP enables flexible trade-offs between accuracy and compute cost (Appendix H) at inference time by varying the token budget without retraining, making it a practical and efficient choice for high-resolution PDE forecasting tasks.

5 Limitations & Future work

Spatio-temporal and physics informed subdivision score The image-gradient-based score map with mean aggregation from classical computer vision methods may not be optimal for all types of PDE dynamics. For example, in Figure 1b, since a binary mask of the `acoustic_scattering_maze` is fed into the model as an input channel, large score is assigned to the maze wall. Consequently, as shown in Figure 1c, 10 and 11, fine patches are assigned along the maze wall, which has large image gradient values due to the broadcasted binary mask channel, regardless of whether the wave dynamic has reach the wall parts yet. Similarly as shown in Appendix L, individual channels may have unique gradient patterns that differ from that of the others, or with themselves at different time steps. Naïvely averaging the gradient score across channels may introduce imbalance of features from different channels. Future research could investigate either using hand-crafted physics-informed indicators for known PDE formulations, or learning a parametric scoring function end-to-end with lightweight vision models, to better capture the salient features relevant to the specific dynamics being learned.

Implementation challenges On the implementation side, due to the lack of dynamic shape support on current hardware accelerators, the IAP method still requires preallocating memory for the maximum token budget during training and inference; therefore, its theoretical efficiency in Appendix F is not fully reflected in actual wall-clock time in Appendix G. Future work could explore more efficient memory management techniques or leverage emerging hardware capabilities to better accommodate dynamic token counts without excessive overhead.

REFERENCES

Benedikt Alkin, Andreas Fürst, Simon Schmid, Lukas Gruber, Markus Holzleitner, and Johannes Brandstetter. Universal Physics Transformers: A Framework For Efficiently Scaling Neural Operators. *Advances in Neural Information Processing Systems*, 37:25152–25194, December 2024. doi: 10.52202/079017-0793. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/2cd36d327f33d47b372d4711edd08de0-Abstract-Conference.html.

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

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

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/jax-ml/jax.

John Canny. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, November 1986. ISSN 1939-3539. doi: 10.1109/TPAMI.1986.4767851. URL https://ieeexplore.ieee.org/abstract/document/4767851.

Rohan Choudhury, JungEun Kim, Jinhyung Park, Eunho Yang, László A. Jeni, and Kris M. Kitani. Accelerating Vision Transformers with Adaptive Patch Sizes, October 2025. URL http://arxiv.org/abs/2510.18091. arXiv:2510.18091 [cs].

Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General Perception with Iterative Attention. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 4651–4664. PMLR, July 2021. URL https://proceedings.mlr.press/v139/jaegle21a.html.

Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, Olivier Hénaff, Matthew M. Botvinick, Andrew Zisserman, Oriol Vinyals, and João Carreira. Perceiver IO: A General Architecture for Structured Inputs & Outputs, March 2022. URL http://arxiv.org/abs/2107.14795. arXiv:2107.14795 [cs].

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

Alexander Kolesnikov, Alexey Dosovitskiy, Dirk Weissenborn, Georg Heigold, Jakob Uszkoreit, Lucas Beyer, Matthias Minderer, Mostafa Dehghani, Neil Houlsby, Sylvain Gelly, Thomas Unterthiner, and Xiaohua Zhai. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. 2021.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 3744–3753. PMLR, May 2019. URL https://proceedings.mlr.press/v97/lee19d.html.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier Neural Operator for Parametric Partial Differential Equations, May 2021. URL http://arxiv.org/abs/2010.08895. arXiv:2010.08895 [cs].

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, March 2021. ISSN 2522-5839. doi: 10.1038/s42256-021-00302-5. URL https://www.nature.com/articles/s42256-021-00302-5.

Payel Mukhopadhyay, Michael McCabe, Ruben Ohana, and Miles Cranmer. Controllable Patching for Compute-Adaptive Surrogate Modeling of Partial Differential Equations, July 2025. URL http://arxiv.org/abs/2507.09264. arXiv:2507.09264 [cs] version: 1.

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 R. Blancard, François Rozet, Liam H. Parker, Miles Cranmer, and Shirley Ho. The Well: a Large-Scale Collection of Diverse Physics Simulations for Machine Learning. *Advances in Neural Information Processing Systems*, 37:44989–45037, December 2024. doi: 10.52202/079017-1430. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/4f9a5acd91ac76569f2fe291b1f4772b-Abstract-Datasets_and_Benchmarks_Track.html.

B. T. Polyak and A. B. Juditsky. Acceleration of Stochastic Approximation by Averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, July 1992. ISSN 0363-0129. doi: 10.1137/0330046. URL https://epubs.siam.org/doi/abs/10.1137/0330046.

M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, February 2019. ISSN 0021-9991. doi: 10.1016/j.jcp.2018.10.045. URL https://www.sciencedirect.com/science/article/pii/S0021999118307125.

Shyam Sankaran, Hanwen Wang, and Paris Perdikaris. WaveLiT: A Parameter-Efficient Architecture for Neural PDE Solvers. October 2025. URL https://openreview.net/forum?id=ZtRkmCzmrN#discussion.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with Rotary Position Embedding. *Neurocomputing*, 568:127063, February 2024. ISSN 0925-2312. doi: 10.1016/j.neucom.2023.127063. URL https://www.sciencedirect.com/science/article/pii/S0925231223011864.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://papers.nips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Sifan Wang, Hanwen Wang, and Paris Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed DeepONets. *Science Advances*, 7(40):eabi8605, September 2021. doi: 10.1126/sciadv.abi8605. URL https://www.science.org/doi/full/10.1126/sciadv.abi8605.

Sifan Wang, Jacob H. Seidman, Shyam Sankaran, Hanwen Wang, George J. Pappas, and Paris Perdikaris. CViT: Continuous Vision Transformer for Operator Learning, February 2025. URL http://arxiv.org/abs/2405.13998. arXiv:2405.13998 [cs].

John William Joseph Williams. Algorithm 232: heapsort. *Communications of the ACM*, 7(6):347–348, 1964.

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. pp. 5804–5813, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Xu_AMR-Transformer_Enabling_Efficient_Long-range_Interaction_for_Complex_Neural_Fluid_Simulation_CVPR_2025_paper.html.

Shicheng Yin, Kaixuan Yin, Yang Liu, Weixing Chen, and Liang Lin. DART: Differentiable Dynamic Adaptive Region Tokenizer for Vision Foundation Models, September 2025. URL http://arxiv.org/abs/2506.10390. arXiv:2506.10390 [cs].

Enzhi Zhang, Isaac Lyngaas, Peng Chen, Xiao Wang, Jun Igarashi, Yuankai Huo, Masaharu Munetomo, and Mohamed Wahib. Adaptive Patching for High-resolution Image Segmentation with Transformers. In *SC24: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–16, November 2024. doi: 10.1109/SC41406.2024.00082. URL https://ieeexplore.ieee.org/abstract/document/10793226.

A Iterative Adaptive Patchification Algorithm

The Best-first refinement step described in Algorithm 1 is the core of the Iterative Adaptive Patchification (IAP) algorithm. At each iteration, the top-K patches with the highest splitting scores $S(P)$ are selected for refinement, where $S(P)$ is defined as the sum of the per-pixel score within patch P. Each split increases the token count by $\Delta = K\,(2^D - 1)$. Prior to running the iterative procedure,

Algorithm 1 Best-first refinement step

1: Initialization begins with a tiling of patches of size `max_length`
2: **for** each refinement step **do**
3: Select the top-K = `split_per_step` patches by $S(P)$
4: Split each into 2^D children at midpoints $m = (v_0 + v_1)//2$
5: Compute scores only for the new children
6: Insert them into a preallocated, score-sorted list of candidates
7: **end for**

the input grid is partitioned into patches of the maximum allowable size, yielding an initial patch count $T_0 = \frac{H \cdot W}{p_{\max}^2}$. Given a target budget $T_{\max}$ and initial patch count T_0, the number of refinement iterations is

$$\texttt{num\_split} \;=\; \Big\lfloor \frac{T_{\max} - T_0}{\Delta} \Big\rfloor. \tag{1}$$

Algorithm 2 Iterative Adaptive Patchification

Require: Image x, token budget T, patch parameters (`min_length`, `max_length`, `split_per_step`, `pow`)
1: Compute the per-pixel score map s via Sobel filters
2: Initialize the candidate list $\mathcal{V}$ using patches of size $p_{\max}$
3: **for** $t = 1$ to `num_split` **do**
4: Perform best-first refinement in Algorithm 1
5: **end for**
6: **return** the vertices of the selected patches $\mathcal{V}$

B Sobel Operator for Image Gradient Estimation

Given the pixel value $X(h, w)$, where h, w are the height and width coordinates, the image partial derivatives at location (h, w) are calculated as

$$\frac{\partial}{\partial h}X(h,w) = \sum_{i=-k}^{k} S_i \sum_{j=-k}^{k} d_j X(h-j, w-i), \frac{\partial}{\partial w}X(h,w) = \sum_{i=-k}^{k} S_i \sum_{j=-k}^{k} d_j X(h-i, w-j), \tag{2}$$

the derivative filter $d \in \mathbb{R}^K$ is computed from a symmetric $(-k, \ldots, k)$ centered finite-difference stencil, and the smoothing filter is the binomial vector $S \in \mathbb{R}^K$ with entries $\binom{K-1}{j}$. The resulting partial derivative is an average of the partial derivatives in the local neighborhood, weighted by the smoothing vector, and its linear nature allows us to write the summation as a convolution operator with kernel size K. Convolving X with the Sobel operator yields a gradient tensor $\nabla X \in \mathbb{R}^{H \times W \times D}$ (with $D = 2$ for images), and the per-pixel splitting score for a pixel at location (h, w) is calculated as the ℓ_2 norm of the image gradient

$$s(h, w) = ||\nabla X(h, w)||_2. \tag{3}$$

For an input signal $X \in \mathbb{R}^{H \times W \times C}$ with C channels, we first normalize each channel across the entire image, then calculate the pixel score map channel-wise and average across channels to yield a final scalar score. An example of the process is illustrated in Figure 1b.

C Decoder Latent Construction

First, each latent embedding is constructed from its enclosing patch token $V_{\pi(i,j)}$, where $\pi(i,j)$ denotes the index of the patch covering location (i,j). To encode relative positional information within the enclosing IAP patch, we introduce a learned gating vector $g(o_{i,j}) \in \mathbb{R}^F$ and bias vector $b(o_{i,j}) \in \mathbb{R}^F$, both dependent on the relative offset $o_{i,j}$ of (i,j) within the enclosing patch. Formally,

$$V_o(i,j) = \sigma(g(o_{i,j})) \odot V_{\pi(i,j)} + b(o_{i,j}) \equiv \text{diag}(\sigma(g(o_{i,j})))V_{\pi(i,j)} + b(o_{i,j}), \tag{4}$$

where $\odot$ denotes elementwise multiplication and σ is the sigmoid function; this serves as an approximation to the regular depatchification process that can be conveniently implemented for parallel computation even with variable-sized incoming patches. However, the ablation study in Appendix D does not indicate a performance gain with the addition of this in-patch relative positional embedding.

D Decoder ablation

# Tokens	1024	2048	4096
With latent	0.15774	0.14778	0.14182
Without latent	0.15975	0.14736	0.14152

Table 2: Decoder ablation results comparing configurations with and without in-patch learnable latent embeddings for `turbulent_radiative_layer_2D`.

E Training Specifications

The models are trained on an NVIDIA H200 GPU using JAX(Bradbury et al., 2018). We use the AdamW optimizer with linear warm-up and exponential decay, and augment the optimizer with Polyak averaging(Polyak & Juditsky, 1992) of step size 0.01, as it improves generalization and therefore test performance. We only use MSE loss during training. Complete optimizer hyperparameters, dataloader settings, and logging defaults are summarized in Table 3. Rotary Position Embeddings (RoPE) remain enabled throughout both the encoder and decoder stacks so that positional phase information is shared between ViT and IAP.

Hyperparameter	Value
Optimizer	AdamW
Weight decay	5e-2
Gradient clipping	Global norm 1.0
Initial learning rate	1×10^{-6}
Warmup schedule	Linear to 5×10^{-4} over 2500 steps
Decay schedule	Exponential, factor 0.95 every 5000 steps
Minimum learning rate	1×10^{-5}
Loss function	Mean-squared error
EMA step size	0.01
Number of iterations	2.5e5

Table 3: Optimization settings shared by all experiments.

Table 4 summarizes the fixed architectural hyperparameters used by the IAP models in this work.

Hyperparameter	Value
Encoder layers	7 (self-attention)
Decoder layers	1 (cross-attention)
Attention heads	6
QKV dimension	384
MLP expansion ratio	2
Split-per-step	4 candidate patches
Latent patch size / min patch size	2
Positional encoding	RoPE

Table 4: Transformer hyperparameters held fixed across most IAP experiments.

Due to the high-resolution of the VI benchmark, we double the minimum and maximum lengths allowed for the IAP algorithm, as well as the latent decoder patch size so that the cross-attention decoder does not incur excessive computational cost. For the IAP model with only 1024 tokens, we double the maximum length so that the token budget is attainable with sufficient coarse patches. For the 3D benchmark, we change the QKV dimension to 432 so that the per-head embedding dimension is divisible by 3 in order to use the 3D axial RoPE. Table 5 lists the Vision Transformer baselines that serve as fixed-token references in Table 1.

Hyperparameter	Value
Encoder layers	8
Attention heads	6
QKV dimension	384
MLP expansion ratio	2
Patch size	$\{2, 4, 8\}$ (per experiment)
Positional encoding	RoPE
Decoder	None (pure encoder)

Table 5: ViT baseline hyperparameters.

F Compute Complexity Analysis

With embedding dimension fixed, the compute cost of a cross-attention block for Q queries and K key-value pairs is proportional to QK. We first approximate the complexity of such a cross-attention decoder as

$$\mathcal{C}(Q, K) \propto QK, \tag{5}$$

in suitable units. Let the scale of the incoming signal grid be S, the dimension of the grid be D, and the number of tokens be $|\mathcal{V}|$. For a Vision Transformer model with an IAP patchification head, $L - 1$ self-attention blocks, and one cross-attention decoder block, the overall compute cost is

$$\mathcal{C}_{\text{IAP}} \propto (L-1)|\mathcal{V}|^2 + \left(\frac{S}{p_{\min}}\right)^D |\mathcal{V}|, \tag{6}$$

whereas a vanilla Vision Transformer with L self-attention blocks has a compute cost of

$$\mathcal{C}_{\text{ViT}} \propto L\left(\frac{S}{p}\right)^{2D}. \tag{7}$$

The actual compute cost may vary due to differences in the accelerator hardware or the driver and software versions.

G Training Wall-clock Comparison

Architecture	AM (256^2)		ASM (256^2)		TRL2D (128×384)		VI (512^2)		SNE64 (64^3)	
	# Tokens	Time (s)	# Tokens	Time (s)	# Tokens	Time (s)	# Tokens	Time (s)	# Tokens	Time (s)
ViT	1024	653.1	1024	354.0	768	280.0	1024	2243.1		
	4096	2243.3	4096	2221.0	3072	1553.3	4096	2549.9	4096	3130.2
IAP	1024	912.1	1024	792.1	1020	669.7	1024	2314.9	1016	2260.8
	2044	1451.2	2044	1319.6	2040	1191.7	2044	2309.3	2024	3329.6
	4096	3239.4	4096	3067.1	4092	2859.4	4096	3632.1	4096	6445.6

Table 6: Training time in seconds per 10,000 steps for the ViT and IAP runs reported in Table 1.

Table 6 reports the wall-clock training time per 10,000 steps for different models. Due to the dynamic nature of the iterative adaptive partition algorithm, and because the transformer backbone is lightweight, the reduction in compute from using fewer tokens is not fully reflected in the measured wall-clock time, especially for these high-resolution benchmarks. Further improvements such as custom CUDA kernel implementation and better asynchronous dispatch support may help close the gap between the theoretical compute cost of the transformer backbone and the observed runtime at the same token count.

H VRMSE Trends Across Evaluation Budgets

Figure 3 reports the VRMSE of each IAP run across the range of evaluation-token budgets, which may not be native to the training configuration of each model.

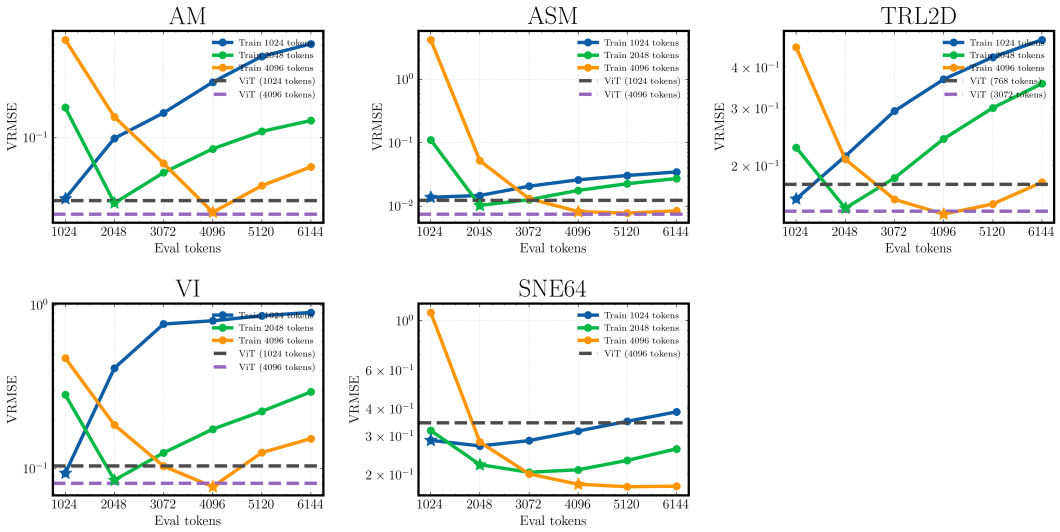


Figure 3: VRMSE across evaluation budgets for all benchmarks. Each panel overlays the IAP runs and ViT baselines for the corresponding dataset.

Table 7 records the training tokens, VRMSE observed at those training budgets, and the per-evaluation VRMSEs associated with each IAP run visualized in Figure 3. Dataset names are merged across rows to highlight the common evaluation sets within each benchmark.

Dataset	# Tokens Trained	VRMSE @ Train	1024	2048	3072	4096	5120	6144
ASM	1024	0.014	0.014	0.015	0.021	0.026	0.031	0.035
	2048	0.010	0.111	0.010	0.013	0.018	0.023	0.027
	4096	0.008	4.22	0.052	0.013	0.008	0.008	0.008
AM	1024	0.028	0.028	0.099	0.168	0.321	0.552	0.717
	2048	0.025	0.188	0.025	0.048	0.079	0.114	0.144
	4096	0.021	0.782	0.155	0.058	0.021	0.037	0.054
TRL2D	1024	0.158	0.158	0.213	0.293	0.366	0.428	0.483
	2048	0.148	0.227	0.148	0.183	0.241	0.299	0.355
	4096	0.142	0.458	0.208	0.157	0.142	0.152	0.177
VI	1024	0.094	0.094	0.412	0.767	0.804	0.862	0.904
	2048	0.085	0.283	0.085	0.125	0.174	0.224	0.295
	4096	0.078	0.476	0.185	0.104	0.078	0.125	0.153
SNE64	1024	0.285	0.285	0.268	0.284	0.314	0.348	0.384
	2048	0.220	0.315	0.220	0.203	0.209	0.230	0.260
	4096	0.179	1.091	0.279	0.200	0.179	0.175	0.176

Table 7: Evaluation VRMSEs for the IAP runs visualized in Figure 3.

I Reproducibility Statement

We provide detailed training specifications in Appendix E, including optimization hyperparameters. The datasets are publicly available as described in Section 3. We will release the code base and checkpoints upon request.

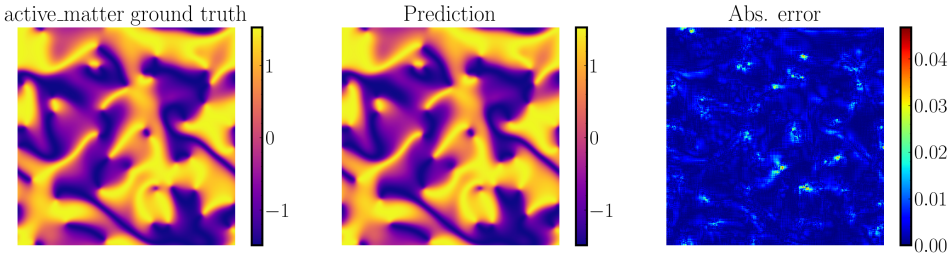
Due to the computational cost, we only evaluate each model once per configuration without repeating across different random seeds. To estimate uncertainty due to random initialization, we run the ViT baseline with a patch size of 4 and an IAP model with 2048 tokens four times with different random seeds on the `turbulent_radiative_layer_2D` dataset. For the IAP model with 2048 tokens, the VRMSE has a sample mean of 0.1491625 and sample standard deviation of 0.0009732, and the ViT model with a patch size of 4 (3072 tokens) has a sample mean of 0.144515 and sample standard deviation of 0.0010843.

J Usage of the Large Language Models

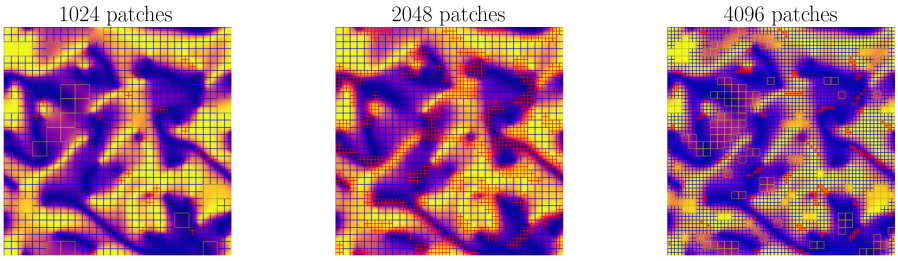
We used Large Language Models (LLMs) mainly for proofreading and improving the readability of this paper. We also used LLMs for code autocompletion while writing certain parts of the code (primarily for visualization) under human supervision; all program executions were performed by the authors.

K VISUALIZATION

In this section, we showcase the predictions of our IAP model with 4096 tokens on the 2D samples, as well as how IAP partitions the images differently with variable token budget.

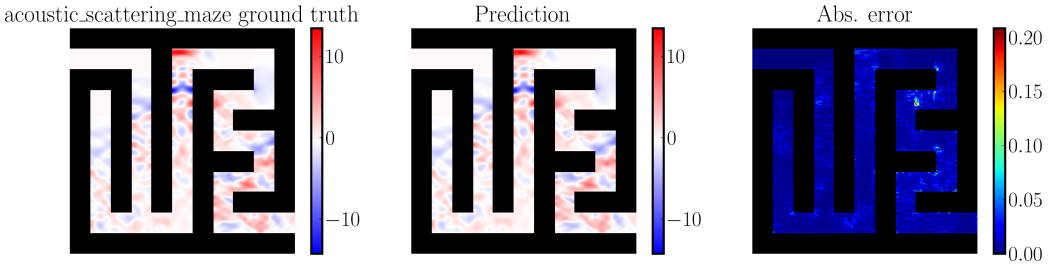


(a) Predictions

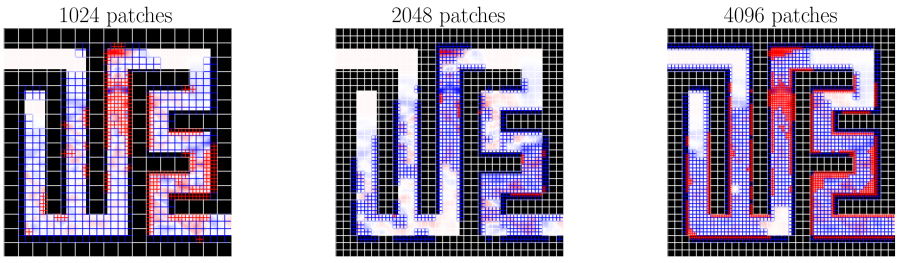


(b) Patchification

Figure 4: Active Matter visualizations across three evaluation-token budgets.

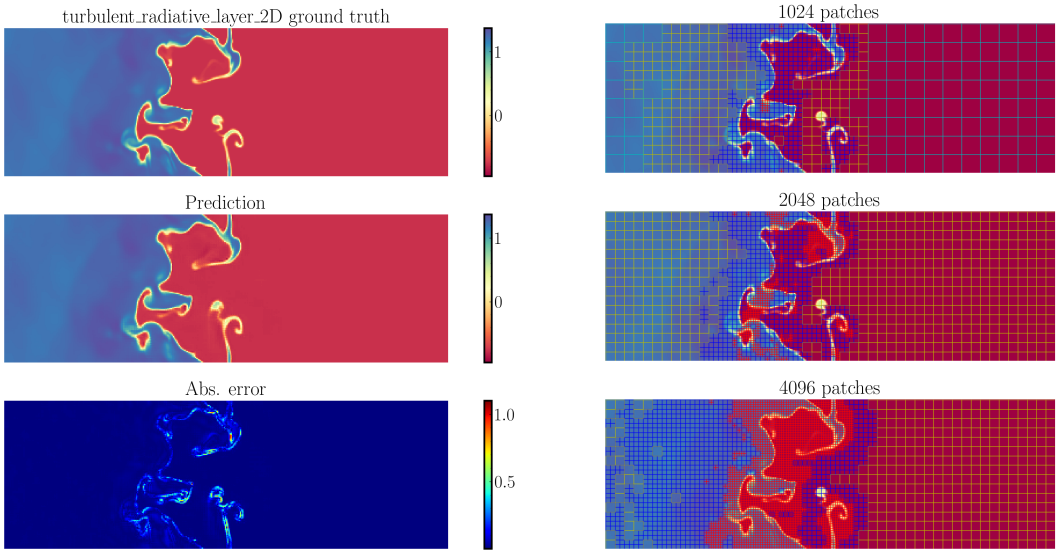


(a) Predictions



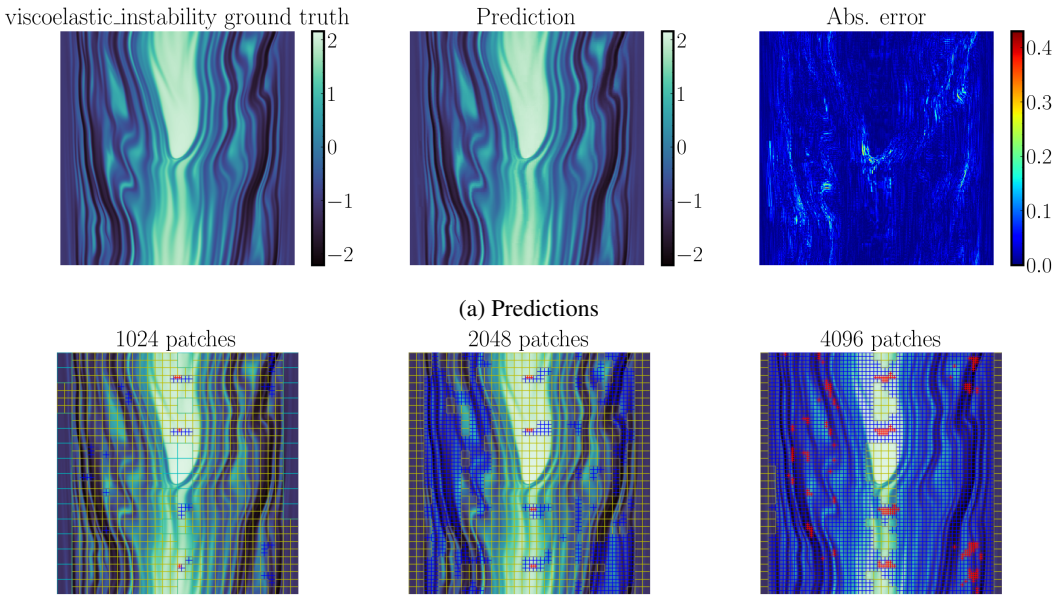
(b) Patchification

Figure 5: Acoustic Scattering Maze visualizations across three evaluation-token budgets.



(a) Predictions (b) Patchification

Figure 6: Turbulent Radiative Layer 2D visualizations across three evaluation-token budgets.



(a) Predictions

(b) Patchification

Figure 7: Viscoelastic Instability visualizations across three evaluation-token budgets.

L Gradient Score Maps

We visualize the per-channel, per-timestep gradient score maps computed by the Sobel operator (Appendix B), as well as their channel-averaged aggregations used by IAP for adaptive patchification.

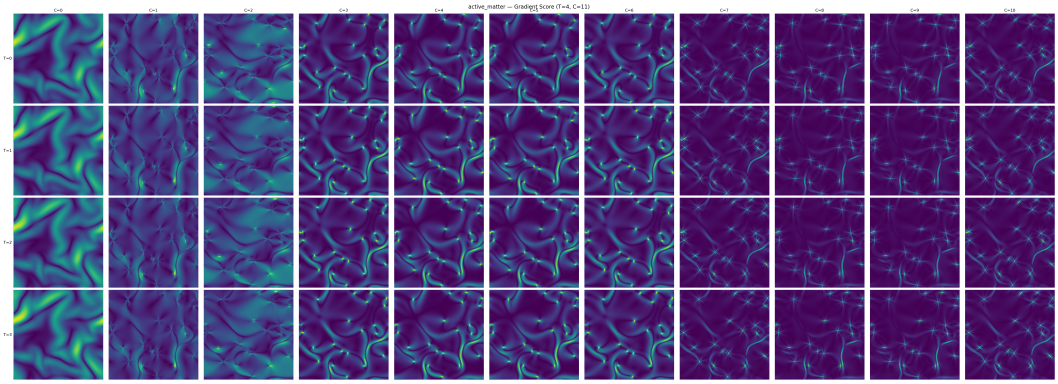


Figure 8: Per-channel, per-timestep gradient score maps for `active_matter`.

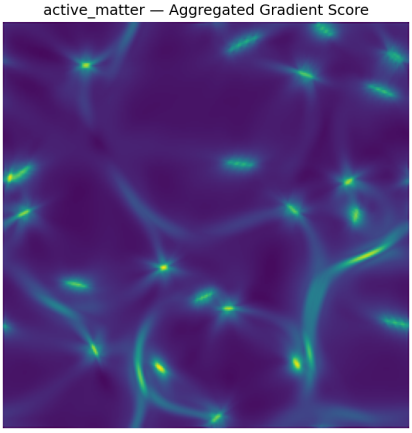


Figure 9: Aggregated gradient score map used by IAP for `active_matter`.

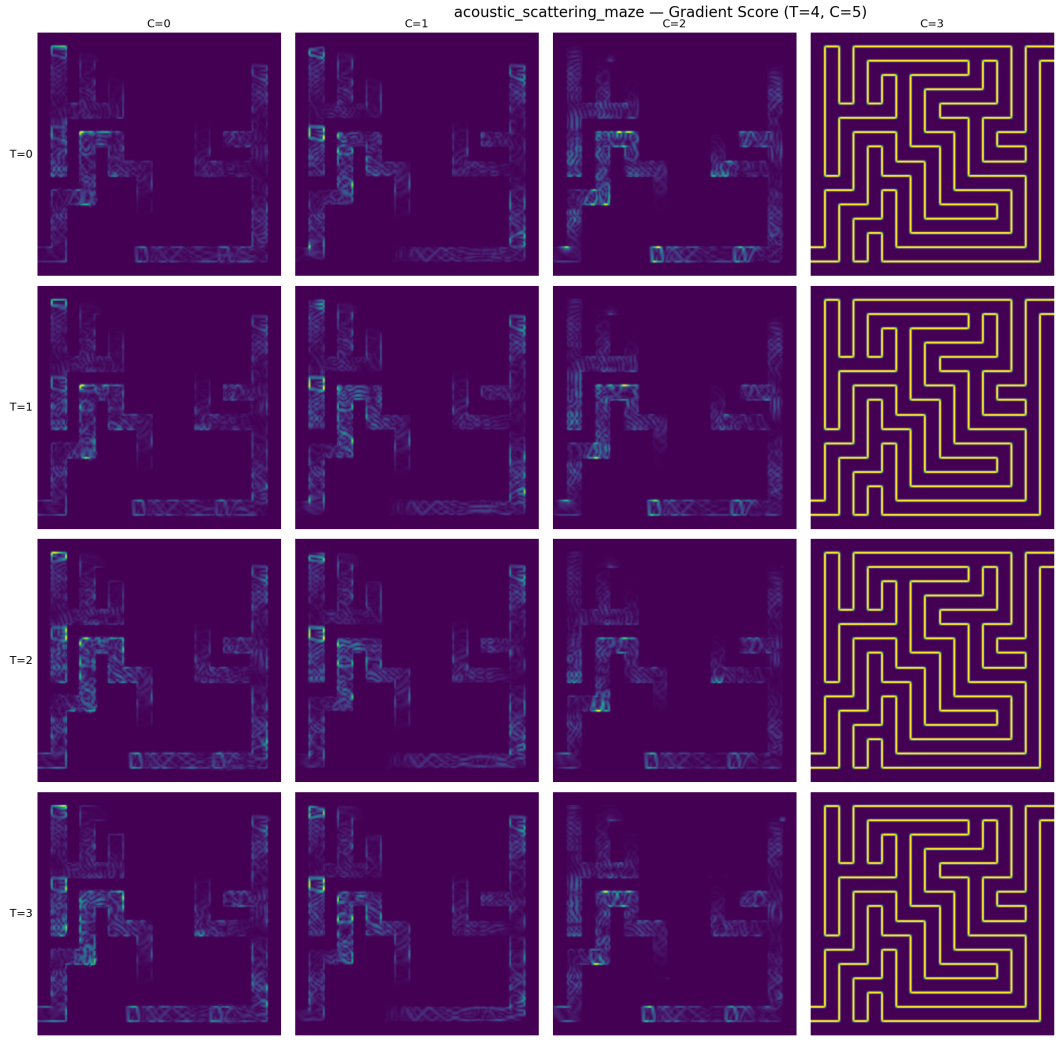


Figure 10: Per-channel, per-timestep gradient score maps for `acoustic_scattering_maze`. The last channel corresponds to the binary mask of the maze wall, which has large gradient values along the wall boundary.

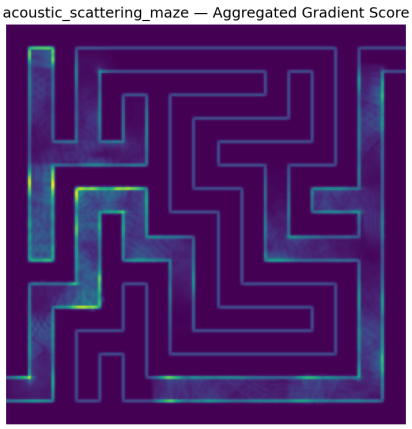


Figure 11: Aggregated gradient score map used by IAP for `acoustic_scattering_maze`.

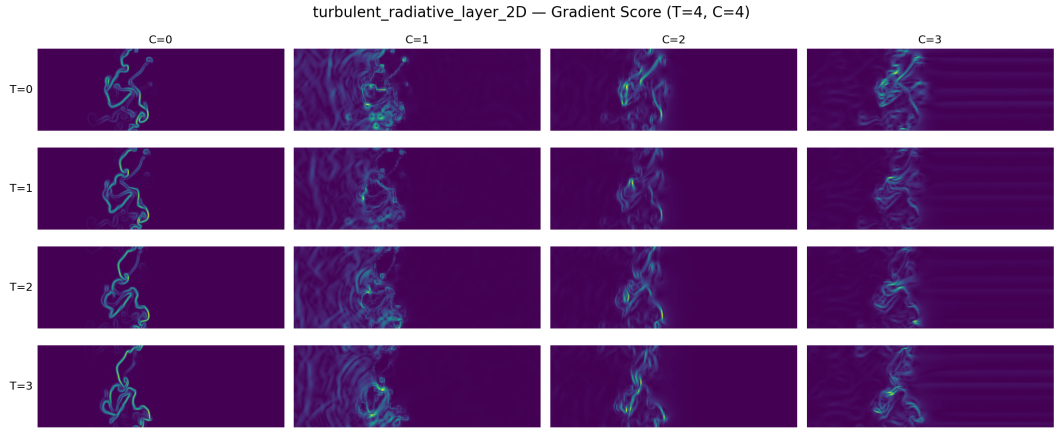


Figure 12: Per-channel, per-timestep gradient score maps for `turbulent_radiative_layer_2D`.

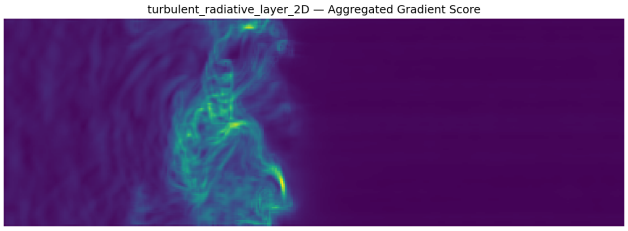


Figure 13: Aggregated gradient score map used by IAP for `turbulent_radiative_layer_2D`.

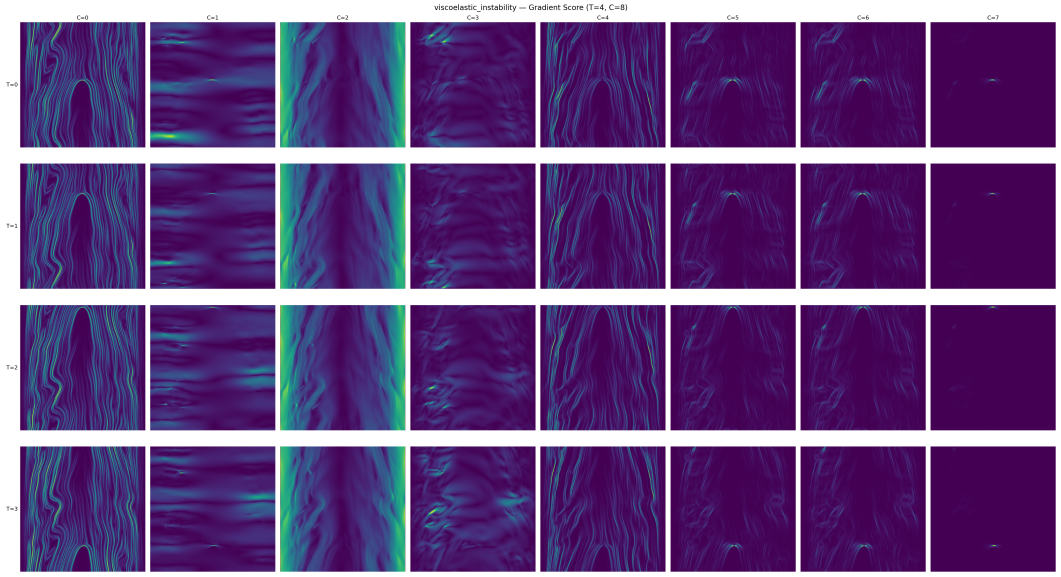


Figure 14: Per-channel, per-timestep gradient score maps for `viscoelastic_instability`.

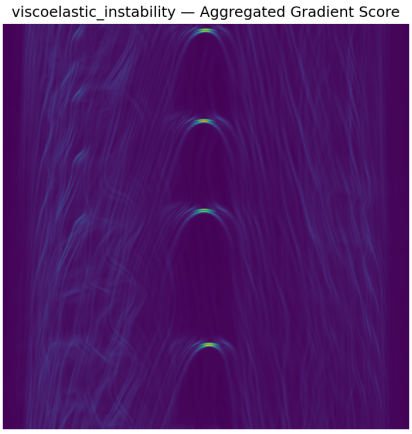


Figure 15: Aggregated gradient score map used by IAP for `viscoelastic_instability`.