Simplifying Transformer-Based U-Net Neural Physics Simulators

Pietro Sittoni
Gran Sasso Science Institute
L'Aquila, Italy
pietro.sittoni@gssi.it

Francesco Tudisco
Gran Sasso Science Institute, L'Aquila, Italy
University of Edinburgh, UK
Miniml.AI

Abstract

U-Net–style architectures are widely adopted for modeling physical systems, as their multiscale structure enables efficient processing of high-resolution data while reflecting the hierarchical structure of many physical phenomena. The most successful U-Net-based neural physics simulators often combine convolutions, Fourier layers, or specialized transformer mechanisms such as windowed or axial attention; however, these structured components can limit adaptability across different spatial and spatiotemporal dimensions. In this work, we revisit transformer-based U-Nets with the goal of simplifying attention-based neural physics simulators without sacrificing multiscale efficiency or predictive accuracy. We introduce `UFlex`, an attention-based U-Net whose Transformer blocks remain close to the standard design and operate on a one-dimensional token sequence, allowing the same architecture to be applied across one-, two-, three-dimensional, and spatiotemporal regular-grid problems without major design changes. Evaluated on seven challenging benchmarks (four 2D and three 3D), `UFlex` scales to resolutions of up to 512×512 in 2D and $256 \times 128 \times 256$ in 3D, while reducing training memory and accelerating training compared to state-of-the-art transformer baselines, all while achieving state-of-the-art predictive accuracy. Code is available at `https://github.com/Pietrosittoni/UFlex`.

1 Introduction

Since their introduction by [79], Transformers have become a central architectural primitive across a wide range of modalities, including language processing [61], vision [24], and audio [8]. Their success is partly due to their implementation simplicity: Transformer blocks are built primarily from matrix multiplications, making them highly parallelizable and well-suited to modern hardware. At the same time, because they operate on sequences, Transformers provide a flexible interface for data of different structure and dimensionality. Together with their favorable scaling behavior [37], these properties have made attention-based models one of the most widely adopted neural architectures in modern machine learning.

These properties make attention-based models particularly appealing for neural physics simulation. Physical systems may be defined on one-, two-, or three-dimensional domains, may evolve over time, and may involve multiple coupled physical fields. A useful neural simulator should therefore be accurate and efficient, but also simple to implement and easy to reuse across problem dimensionalities. However, directly applying standard self-attention to high-resolution physical states is often impractical, since its memory complexity scales as $\mathcal{O}(n^2)$, where n is the number of processed tokens. This bottleneck is especially severe during training. In `PyTorch` [58], for instance, activations from the forward pass are stored in the computational graph to enable efficient gradient computation. Model-agnostic memory-saving strategies, such as gradient checkpointing [20] or gradient accumulation across multiple steps, can reduce memory usage but often significantly slow down optimization.

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

A common way to make attention feasible for high-resolution physical systems is to combine it with multiscale architectures. Many physical phenomena contain structure across a wide range of spatial scales: in the Navier–Stokes equations, for example, fine-scale features such as thin boundary layers, shocks, and turbulent eddies coexist with large-scale structures such as jets, vortices, and global flow patterns. U-Net–style architectures exploit this structure by progressively downsampling the representation in an encoder, increasing the effective receptive field and reducing the number of processed tokens, and then restoring spatial resolution in a decoder through upsampling and skip connections. This multiscale inductive bias underlies the original U-Net [64], convolutional architectures rooted in earlier work on neural visual processing and convolutional representations [26, 39], and several neural physics simulators, including SineNet [84], U-FNets [28], Poseidon [33], and PDE-Transformer [35].

Despite their strong performance, many state-of-the-art attention-based neural physics simulators rely on specialized attention mechanisms and highly engineered architectural components. MPP [51], for example, incorporates Axial-Attention [34] among other modifications. Poseidon [33] and PDE-Transformer [35] combine Swin-Transformer Attention [48] with U-Net–inspired designs. Fourier-based U-Net variants replace convolutional layers with Fourier neural operator layers [43]. These models achieve impressive results, but their performance often comes with substantial architectural specialization. In particular, moving from two-dimensional to three-dimensional simulations, or from square domains to rectangular or spatiotemporal domains, can require replacing core layers, inflating two-dimensional components to three dimensions [82, 53], or otherwise modifying significant parts of the implementation.

In this paper, we introduce `UFlex` by taking a different approach. Rather than designing a new, highly specialized attention mechanism, we keep each Transformer block as close as possible to the standard design—multi-head self-attention with RoPE, RMSNorm, a SwiGLU MLP, and residual connections—and place these blocks inside a U-Net hierarchy with downsampling, upsampling, and skip connections. We stress that the simplicity we target lies in the *blocks*: we do not use windowed, axial, or Fourier-based attention, whereas the overall U-shaped design is a deliberate architectural choice that controls memory and computation at high resolution. Crucially, `UFlex` operates on a one-dimensional token sequence throughout, so that the same building blocks are used across one-, two-, and three-dimensional problems, with or without temporal context. Inputs are mapped to a sequence of tokens, processed through the multiscale encoder–decoder, and mapped back to the target physical field.

This simplicity does not come at the cost of performance. We show that `UFlex` outperforms multiple state-of-the-art neural physics simulators across a broad set of benchmarks, including specialized attention-based models such as MPP [51], Poseidon [33], PDE-Transformer [35], and DPOT [30], as well as U-Net, Unisolver [86], and Transolver [81]. Our experiments cover diverse parametric and complex physical systems at resolutions ranging from 256×256 to 512×512 in 2D and from 64^3 to $256 \times 128 \times 256$ in 3D, corresponding to grid sizes from 65k to 8.5M points. These results suggest that a carefully designed U-Net built from standard attention blocks can achieve state-of-the-art accuracy while remaining substantially easier to adapt across simulation settings than more specialized architectures.

Overall, our main contributions are as follows:

- We propose `UFlex`, a multiscale neural physics simulator with a 1D latent token representation, built from Transformer blocks.
- We show that the same architecture applies to 2D and 3D regular-grid simulations, with or without temporal context, without major design changes, and that a model trained in 2D can be fine-tuned directly in 3D.
- We show that `UFlex` matches or outperforms state-of-the-art neural physics simulators on six of seven benchmarks, while requiring less training memory and wall-clock time.
- We provide ablation studies on the main architectural components of `UFlex`: downsampling and upsampling functions, skip-connection types, resampling factor, and model size.

2 Related Work

Machine Learning for Physics. Neural operators [6, 31], such as FNO [42, 77, 41, 45], Graph Neural Networks [16, 67], Equivariant Graph Neural Networks [76], or combinations of GNNs

and FNOs [44], are popular approaches for simulating physical systems. Other notable models include convolution-based architectures with channel attention [72] and U-Net–based models [80, 84]. PINNs [62] are another important class of methods, although they are often harder to train [38, 66]. Additional relevant work includes [68], which trains a tokenizer for solving in-context learning problems, as well as approaches that combine machine learning with classical solvers [78, 74, 10].

Transformers. In recent years, Transformers [79] have become the gold-standard architectural primitive across a wide range of modalities, including language [61, 23], vision [24], audio [8], video [3], physics [14], and multimodal settings such as text–image models [60] and scientific simulations involving spectra and images [57]. Moreover, Transformers are increasingly the dominant choice across different learning paradigms, such as self-supervision with Masked Autoencoders (MAE) [32], DINO [19, 55, 70], and JEPA [4, 5]. They are also widely used in diffusion models, both latent-space models [63, 59], including those applied to physical simulation [65], and pixel-space models [9, 22, 36]. More recently, flow-matching models have also emerged as a strong alternative to diffusion-based models [46, 25, 15, 11], and the state-of-the-art models are Transformer-based.

U-shaped Transformers. Several works combine Transformer blocks with a U-Net hierarchy. In image generation, U-ViT [9] uses long skip connections but no downsampling, while HDiT [22] and simple diffusion [36] rely on a 2D structured latent space. U-DiT [75] downsamples tokens inside a U-shaped diffusion Transformer, and PDE-Transformer [35] adapts a related design to physics simulation, reaching accuracy comparable to U-DiT at a substantially lower training cost [35]. These two architectures are the closest to `UFlex`, but differ in three main respects. First, both organize tokens on a 2D grid and rely on 2D-specific operators, namely PixelShuffle/Unshuffle-based resampling, shifted-window attention in PDE-Transformer, and pixel-unshuffled attention with depthwise convolutions in U-DiT; as a result, PDE-Transformer is restricted to 2D regular grids [35]. `UFlex` instead operates on a 1D token sequence, so the same backbone handles 2D, 3D, and spatiotemporal inputs, and a model trained in 2D can be fine-tuned directly in 3D (Section 7). Second, both models restrict attention to local windows or token subsets and add further structured components, such as channel-wise axial attention in PDE-Transformer and cosine-similarity attention, convolutional feed-forward layers, and re-parameterization in U-DiT; `UFlex` uses standard multi-head self-attention, global in space and causal in time, with RoPE, a SwiGLU MLP, and RMSNorm at every stage. Third, PDE-Transformer conditions only on the PDE type, whereas `UFlex` embeds a variable number of continuous physical parameters through adaRMSN-Zero. Moreover, the concatenation-based skip connections of U-DiT resemble the variant that we evaluated and discarded in our ablations (Section 5). Although several of these components (RoPE, RMSNorm, SwiGLU, patch tokenization) are standard in modern Transformers, our contribution is not a new building block, but a unified U-shaped Transformer for regular-grid physics simulation that handles different dimensionalities within a single framework.

Transformer-based models for physics simulations. Transformers are widely used in physics simulations and operator learning due to their ability to capture long-range dependencies. Early approaches include Galerkin-style and Fourier transformers [18], transformer methods for inverse boundary-value problems [27], temporal-attention models on reduced meshes [29], GNOT for irregular meshes and multi-input operators [31], and OFormer as a transformer variant for irregular meshes [40]. Transformer-based models have also demonstrated strong performance in weather forecasting, as seen in Aurora [14] and Pangu-Weather [13]. For PDEs on regular grids, several specialized architectures have been proposed, including Poseidon [33], MPP [51], and DPOT [30]. MPP and DPOT rely on in-context memory to autoregressively predict future solution snapshots, while other methods adopt parameter conditioning mechanisms, such as Unisolver [86] or PDE-Transformer [35]. For general geometries and irregular meshes, recent models include UPT [2], AB-UPT [1], Erwin [85], Transolver [81], and Transolver++ [50]. We view regular-grid and general-geometry simulators as two active and complementary research directions: the latter are more general in scope, whereas architectures specialized to regular grids typically enjoy more favorable computational scaling. Our focus is on the regular-grid setting; nonetheless, we compare against Transolver in Section G.3.

3 Model Design

Setting. We aim to learn a physical system from a set of physical states collected in sequences of n temporal trajectories, discretized both in space and time $\{\left(u_t^l\right)_{t=0}^{T(l)}\}_{l=1}^{n}$. The temporal horizon T can

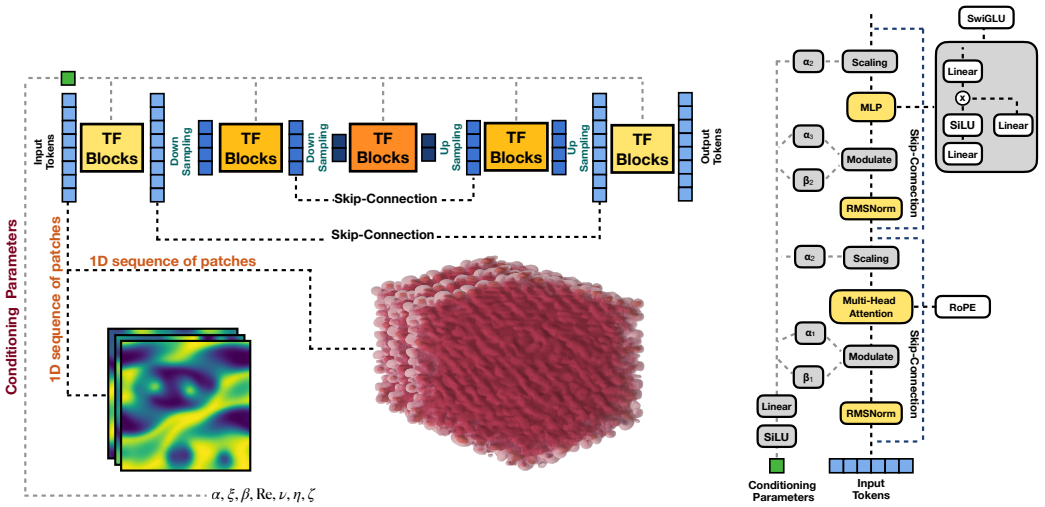


Figure 1: **Left:** `UFlex`'s architecture diagram; TF Blocks denotes Transformer Blocks. **Right:** diagram of a single Transformer block.

vary among the different trajectories, thus it depends on l; we drop l in the following for clarity. We train the model M_θ with the loss function

$$\mathcal{L}(\theta) = \mathbb{E}_{u_{t+1}|u_{t;m},u_{t;m}\sim p}\left[||M_\theta(u_{t;m}) - u_{t+1}||_2^2\right].$$

The variable m denotes the in-context memory. If $m = 0$, then $u_{t;m} = u_t$; if instead $m > 0$, then $u_{t;m} = [u_{t-m}, \ldots, u_t]$, meaning that it contains the current snapshot and the m previous ones. This autoregressive training paradigm allows us, during inference, to reconstruct the discretized trajectory starting from an initial condition u_0 (and, if $m > 0$, the next additional m memory snapshots) by *unrolling* the model, i.e., by repeatedly applying $u_{t+1} = M_\theta(u_{t;m})$, $t = t + 1$.

Architecture Rationale. We argue that Transformer blocks used within a U-Net do not require an explicit 2D or 3D latent space structure to model high-dimensional physical systems, since the most important inductive bias of U-Net–based models is their multiscale structure. Given a sequence of tokens $x_1, \ldots, x_n$, self-attention can learn relationships between any two tokens x_i and x_j, regardless of how the sequence is organized. Based on this rationale, we design `UFlex`, a Transformer-based model with a 1D latent representation that ensures adaptability to varying dimensionalities (e.g., 2D, 3D, and temporal domains) while retaining the memory and computational advantages of multiscale modeling. The architecture is illustrated in Figure 1.

Physical State Projection. Consider a 2D temporal physical state $u \in \mathbb{R}^{T\times H\times W\times C}$ or a 3D state $u \in \mathbb{R}^{T\times H\times W\times D\times C}$, where T denotes the temporal components, H denotes the height, W the width, D (in the 3D case) the depth, and C the number of channels, which may represent a scalar field (e.g., pressure), a vector field (e.g., velocity), or, in more complex scenarios, a tensor field (e.g., a stress tensor). As is common in Transformer architectures [24], we divide the state into a 1D sequence of patches $u \in \mathbb{R}^{L\times(CP_tP_hP_wP_d)}$, where (P_t, P_h, P_w, P_d) are the patch sizes along each physical dimension, and

$$L = \begin{cases} (T/P_t)(H/P_h)(W/P_w), & \text{for 2D states,} \\ (T/P_t)(H/P_h)(W/P_w)(D/P_d), & \text{for 3D states.} \end{cases}$$

In the 2D case, we set $P_d = 1$. The in-context memory snapshots $u_{t;m}$ are therefore folded directly into the patching operation: each spatio-temporal patch groups P_t consecutive snapshots, so that an arbitrary temporal context window is absorbed purely through patch construction, without a separate temporal module or any change to the backbone. Tokens are ordered by temporal patch, and when the input contains more than one temporal patch, we apply a block-causal attention mask: each token attends to all tokens of its own temporal patch and of earlier ones, so that attention is global in space and causal in time. Since downsampling merges consecutive tokens within the same temporal patch, this structure is preserved at every scale. We normalize the patches using an RMSNorm layer [83],

linearly project them into a latent space $\mathbb{R}^{L\times d}$ (where d is the latent dimension), and apply a second RMSNorm. After processing the tokens with the model, we decode the latent representation by first normalizing it, then linearly projecting it back into the physical space, and finally unpatchifying it. This converts the sequence of tokens back into the original physical state u.

Transformer Block. Our model employs a modern Transformer block that incorporates several recent improvements compared to the original architecture [79]. To break the permutation equivariance of Multi-Head Attention (MHA), we use Rotary Position Embeddings (RoPE) in its original formulation [71]. For the MLP, we adopt a SwiGLU [69] feed-forward block, a gated linear unit variant using the SiLU nonlinearity. We use RMSNorm as a pre-normalization layer throughout the network.

Many physical phenomena depend on scalar parameters that are not directly observable from the state u, such as the Mach number, Atwood number, or viscosity. Given a set of scalar parameters $p_1, \dots, p_r$, we embed each of them into the latent space using a two-layer MLP with a SiLU activation between the linear layers. We then aggregate the embeddings by summation, $c = \sum_{i=1}^{r} \phi_i(p_i)$, where ϕ_i denotes the MLP embedding of p_i.

To inject these physical parameters into the backbone model, we adopt a mechanism similar to the adaLN-Zero used in [59]. Before passing the normalized activations to the MHA and MLP components, we apply a modulation layer defined as

$$\texttt{modulate}(x, c) = \alpha(c) \circ x + \beta(c),$$

where $\circ$ denotes the Hadamard product, and $\alpha(c), \beta(c)$ are learned vectors. Additionally, we scale the outputs of both MHA and MLP using another learned vector $\alpha_2(c)$.

For each Transformer block, we apply a SiLU activation and project c linearly into a space of dimension six times the embedding dimension, which we then split into six separate components (three for MHA and three for the MLP). The main difference from the approach in [59] lies in our use of RMSNorm instead of LayerNorm [7] (hence we refer to our conditioning system as adaRMSN-Zero instead of adaLN-Zero), and in the use of multiple conditioning parameters, since [59] conditions on the diffusion time and the class label. Our method allows for the encoding of a variable number of parameters.

Downsampling & Upsampling Modules. The downsampling and upsampling stages modify the length of token sequences through two functions: $D\colon \mathbb{R}^{N\times d} \to \mathbb{R}^{(N/f)\times d}$ (downsampling), $U\colon \mathbb{R}^{N\times d} \to \mathbb{R}^{(N\cdot f)\times d}$ (upsampling), where $f > 1$ denotes the resampling factor. Given a latent representation $z \in \mathbb{R}^{N\times d}$, the downsampling stage applies D to produce a shorter representation $\tilde{z} \in \mathbb{R}^{(N/f)\times d}$. Similarly, the upsampling stage applies U to obtain a longer representation $\hat{z} \in \mathbb{R}^{(N\cdot f)\times d}$. In `UFlex`, D is a learnable linear projection that merges f consecutive tokens into one, U is a learnable linear operator that expands each token into f tokens, and encoder and decoder features at the same scale are combined through additive skip connections. We use $f = 4$ by default. These choices are supported by an extensive ablation study, reported in Section 5.

`UFlex` **Implementation Details.** We implement three model sizes, Small, Big, and Large, all with three stages, i.e., two downsampling and two upsampling operations (Figure 1), and a patch size of 8 ($16 \times 8 \times 16$ for the $256 \times 128 \times 256$ CE-RSG dataset). The Small and Big models use two Transformer blocks per stage, while the Large model uses three blocks per encoder and decoder stage and four in the bottleneck. Across all stages, the latent dimension is defined as `latent dimension` $= 64 \times$ `number of heads`, with the number of heads set to $[6, 8, 12]$ (Small), $[8, 12, 18]$ (Big), and $[12, 16, 24]$ (Large). We use an MLP ratio of four. After each down-/upsampling operation, both the tokens and the conditioning variables are linearly projected to the new latent space. We refer to the three variants as `UFlex-S` (57M parameters), `UFlex-B` (123M), and `UFlex-L` (374M). Unless otherwise specified, all results correspond to `UFlex-B`. For all experiments, we predict the next snapshot using only the previous one. For the low-resolution (64^3) 3D Compressible Navier–Stokes dataset, we instead use four previous snapshots as temporal context, grouped into temporal patches of size $P_t = 2$, following [17]. Our implementation is available at `https://github.com/Pietrosittoni/UFlex`.

Table 1: Datasets used in our experimental evaluation.

2D Dataset	Resolution	3D Dataset	Resolution
Active Matter (AM)	256×256	Compressible Navier-Stokes (CNS)	64^3
Planetswe (PSWE)	256×512	Rayleigh-Taylor Instability (RTI)	128^3
Viscoelastic Instability (VI)	512×512	Convective Envelope RSG (CE-RSG)	$256 \times 128 \times 256$
Compressible Navier-Stokes (CNS)	512×512		

Component	*Variant*	AM	PSWE	VI	CNS
Upsampling	Learnable	**0.344**	**0.244**	**0.099**	**0.087**
	Linear Int.	0.439	0.292	0.123	0.109
Downsampling	Learnable	**0.344**	**0.244**	**0.099**	**0.087**
	Max	0.353	0.254	0.113	0.090
	Average	0.363	0.263	0.115	0.094
Skip connection	Sum	**0.344**	**0.244**	**0.099**	**0.087**
	Learnable	0.350	0.248	0.106	0.090

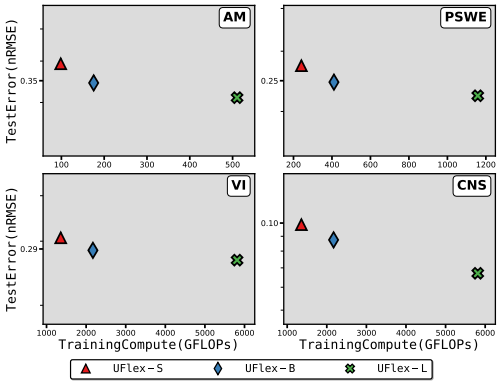


Figure 2: **Ablation studies and model scaling** on Active Matter (AM), Planetswe (PSWE), Viscoelastic Instability (VI), and Compressible Navier–Stokes (CNS). **Left:** upsampling, downsampling, and skip-connection ablations. **Right:** model scaling. We report the nRMSE rollout loss over the first 30% of the maximum temporal horizon, and we highlight in light blue our default choice and in **bold** the best test error.

4 Experimental Set-up

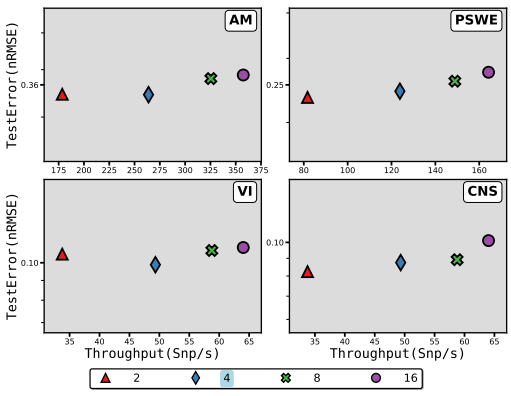
Training Details. For all the experiments, we use AdamW [49] with $\beta_2 = 0.95$, which differs from the original $\beta_2 = 0.999$, to avoid training instability [21, 56]. As a learning rate scheduler, we use a linear warm-up with 1000 training steps, from 0 to 5×10^{-4}. We then use a cosine scheduler to decrease the learning rate down to 1×10^{-6}. We did not use weight decay. We maintain an exponential moving average of the model weights using a decay factor of 0.998. Moreover, we clip the gradient to have a maximum norm of 1. Similar to [35], we use mixed precision training. Additional training details can be found in Appendix D.

Datasets. We evaluate `UFlex` and state-of-the-art baselines on seven different datasets, four 2D and three 3D, from The Well [54] and PDEBench [73]. Details are reported in Table 1 and Appendix A.

5 Ablation Studies

We run all ablation studies with `UFlex-B` on four 2D equations: Active Matter, Planetswe, Compressible Navier–Stokes, and Viscoelastic Instability. We ablate the down/upsampling functions, the type of skip connection between the encoder and decoder, and the resampling factor, and we study the model's parameter scalability.

Downsampling function. We compare three downsampling strategies: **Average**, **Max**, and a **Learnable** linear projection. As shown in Figure 2, the Learnable strategy consistently achieves the lowest test nRMSE across all datasets. It provides relative error reductions of $\sim$3–14% over fixed pooling, with the most significant gains on VI (12.4% improvement over Max, 13.9% over Average). On the AM and PSWE datasets, Learnable downsampling outperforms Average pooling by 5.2% and 7.2%, respectively. Fixed pooling remains competitive, particularly on CNS, where Max pooling (0.090) is relatively close to the Learnable result (0.087); however, the trainable approach offers a consistent performance advantage.



Factor f	*Datasets*					
	AM		PSWE		VI & CNS	
	TM (GB)	TS (s)	TM (GB)	TS (s)	TM (GB)	TS (s)
2	22	0.56	21	0.56	21	1.12
4	16	0.45	16	0.46	16	0.94
8	14	0.39	14	0.36	14	0.84
16	14	0.33	13	0.33	13	0.72

Figure 3: **Resampling factor** f. Comparison of four different resampling factors $(2, 4, 8, 16)$ across four datasets. **Left**: nRMSE rollout loss over the first 30% of the maximum temporal horizon. **Right:** Training Memory (TM) and Training Speed (TS). We highlight in light blue our default choice.

Upsampling function. We compare **Linear Interpolation**[1] against a **Learnable** linear operator that expands tokens into higher-resolution representations. As shown in Figure 2, the Learnable strategy significantly outperforms interpolation, achieving relative nRMSE reductions of 16% to 22% across all datasets. The most substantial gains occur on the AM and CNS datasets, where error drops from 0.439 to 0.344 (21.6%) and 0.109 to 0.087 (20.2%), respectively. In the VI and PSWE cases, the learnable approach improves performance by 19.5% and 16.4%. These results highlight that a trainable mapping is crucial for preserving high-frequency details during the decoding stage, offering a clear advantage over fixed interpolation.

Skip connections. We ablate the skip connection between the encoder and decoder, comparing a simple element-wise **sum** against a **learnable** concatenation-based projection. This learnable variant is close to the skip connections used in PDE-Transformer [35] and U-DiT [75]. As shown in Figure 2, both strategies yield similar performance, but the sum operation consistently achieves the lowest nRMSE. Specifically, it provides marginal improvements of 1.6% on PSWE ($0.248 \rightarrow 0.244$) and AM ($0.350 \rightarrow 0.344$), and up to 6.6% on VI ($0.106 \rightarrow 0.099$). On CNS, the sum operation outperforms the learnable alternative by 3.3% ($0.090 \rightarrow 0.087$). These results suggest that simple additive skip connections are sufficient for effective feature integration, offering superior performance without the additional parameter overhead of a learnable projection.

Resampling factor f**.** We evaluate resampling factors $f \in \{2, 4, 8, 16\}$ (Figure 3). Increasing f from 2 to 4 yields the highest relative throughput gain ($1.47\times$), while further increases show diminishing returns (e.g., $1.1\times$ from $f = 8$ to 16). Accuracy is non-monotonic: $f = 4$ achieves the lowest test error for Active Matter (nRMSE ≈ 0.32) and VI (nRMSE ~ 0.10). Even at $f = 16$, performance remains robust, with CNS error only rising from 0.06 ($f = 2$) to 0.10. Training efficiency follows a similar trend: moving from $f = 2$ to $f = 4$ reduces memory by 27% (22GB $\rightarrow$ 16GB) and accelerates AM training speed by 20% (0.56s $\rightarrow$ 0.45s). For VI & CNS, memory stabilizes at $\sim$ 14GB for $f \geq 8$, while gradient accumulation scales training time by approximately $2\times$ compared to AM/PSWE.

Model scaling. We evaluate three model sizes, `S` (57M), `B` (123M), and `L` (374M), analyzing test error against training compute. As shown in Figure 2, increasing the number of parameters consistently improves performance across all datasets. The most significant scaling benefit is observed in the CNS dataset, where test nRMSE drops from ~ 0.10 (`S`) to ~ 0.07 (`L`). For the remaining datasets, scaling effects are more uniform; for example, in AM, nRMSE improves from ~ 0.36 to ~ 0.33. These gains come with predictable resource trade-offs: moving from `S` to `L` increases training memory usage by $3.2\times$ (11GB $\rightarrow$ 35GB) and compute requirements by approximately $5\times$ (e.g., $100 \rightarrow 500$ GFLOPs for AM).

[1]We used `F.interpolate(x, size=new_seq_len, mode='linear', align_corners=False)`, where `new_seq_len` is the length of the upsampled sequence.

Dataset	*Model*	TE (nRMSE)	T (snp/s)	TS (s)	TC (GFLOPs)
AM	PDE-T	0.598	96	1.8	474
	Unisolver	1.060	107	2.4	312
	U-Net	0.782	145	2.9	822
	Poseidon	0.717	113	2.6	195
	MPP	0.459	64	3.38	719
	DPOT	0.623	150	1.25	700
	`UFlex-B`	**0.343**	**263**	**0.45**	175
PSWE	PDE-T	0.404	48	1.8	925
	Unisolver	0.423	80	2.1	358
	U-Net	0.348	72	2.8	1638
	Poseidon	0.300	58	2.5	387
	MPP	0.337	33	3.4	1434
	DPOT	0.682	75	1.18	1390
	`UFlex-B`	**0.243**	**123**	**0.46**	410
VI	PDE-T	0.482	25	3.72	3710
	Unisolver	0.362	**58**	3.9	792
	U-Net	0.288	36	5.8	6556
	Poseidon	0.367	29	5.2	1560
	MPP	0.100	16	6.6	5754
	DPOT	**0.091**	42	2.06	5410
	`UFlex-B`	0.100	49	**0.94**	2170
CNS	PDE-T	0.869	25	3.7	3710
	Unisolver	0.403	**58**	3.9	792
	U-Net	0.338	36	5.8	6556
	Poseidon	0.212	29	5.2	1560
	MPP	0.091	16	6.6	5754
	DPOT	0.122	42	2.06	5410
	`UFlex-B`	**0.087**	49	**0.94**	2170

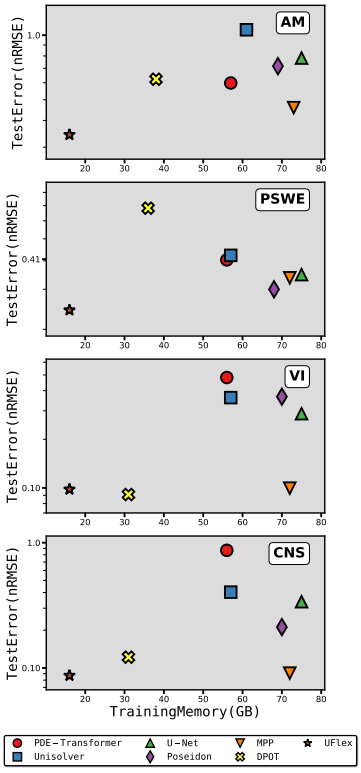


Figure 4: **Baseline 2D**. The table on the **left** reports Test Error (TE), i.e., the nRMSE rollout loss over the first 30% of the maximum temporal horizon, Throughput (T), Training Speed (TS), and Training Compute (TC). The plots on the **right** show the trade-off between Test Error and peak VRAM usage during training. Best and second-best performing models are indicated in **bold** and light gray, respectively.

6 Comparison with Other Works

6.1 2D

We evaluate the performance of our model against different state-of-the-art baselines: PDE-Transformer [35], Unisolver [86], U-Net [64], Poseidon [33], MPP [51], and DPOT [30]. All baselines are matched to approximately the same number of parameters as `UFlex-B`; we leave all other technical details of the baselines to Section B.1 and refer to Appendix E for the full rollout comparison.

Training Efficiency. As shown in Figure 4, `UFlex` significantly reduces resource overhead compared to state-of-the-art baselines. It utilizes the least training memory across all datasets; for example, in PSWE, it uses $\sim$ 15GB while DPOT, the closest competitor, consumes $\sim$ 35GB ($\sim 2\times$ more). Swin-based models like PDE-Transformer and Poseidon require over $4\times$ the memory of `UFlex`, reaching $\sim$ 70GB. Consequently, `UFlex` achieves superior training speeds, requiring only 0.45s–0.94s per step, which is $2.7\times$ to $7\times$ faster than MPP (3.38s–6.6s).

Generalization. `UFlex` achieves state-of-the-art performance on three of four datasets, with substantial nRMSE leads in AM (0.343 vs. MPP's 0.459) and PSWE (0.243 vs. Poseidon's 0.300). On the VI dataset, `UFlex` (0.100) is narrowly outperformed by DPOT (0.091), a 9% difference. This marginal gap is likely because DPOT and MPP, which require in-context memory, are unrolled over a shorter temporal horizon than the other baselines (including `UFlex`). On the VI dataset, their rollout

Dataset	*Model*	TE	T	TM	TC
CNS	MPP 3D	0.425	87	55	28
	DPOT 3D	0.219	169	28	401
	U-Net 3D	0.224	176	51	443
	`UFlex-B`	**0.197**	**241**	**14**	109
RTI	MPP 3D	0.147	43	55	226
	DPOT 3D	0.104	23	51	3171
	U-Net 3D	0.075	41	67	1515
	`UFlex-B`	**0.040**	**45**	**18**	1130
CE-RSG	MPP 3D	0.349	12	-	940
	DPOT 3D	0.434	6	-	2347
	U-Net 3D	0.315	14	-	1295
	`UFlex-B`	**0.256**	**36**	-	1388

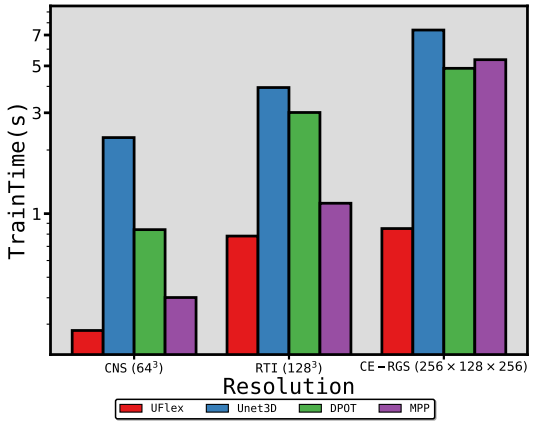


Figure 5: **Baseline 3D**. The table on the **left** reports four key performance metrics: Test Error (TE), i.e., the nRMSE rollout loss over the first 30% of the maximum temporal horizon, Throughput (T), Training Memory (TM), and Training Compute (TC). The figure on the **right** illustrates the scaling of training time relative to data resolution. Best and second-best performing models are indicated in **bold** and light gray, respectively.

within the first 30% of the maximum temporal horizon is three times shorter, which naturally biases the nRMSE evaluation in favor of DPOT and MPP. We study the robustness of this comparison across random seeds in Section G.2.

Inference Efficiency. `UFlex` achieves the highest throughput on AM (263 snp/s) and PSWE (123 snp/s). While Unisolver is more efficient at higher resolutions, it requires $3.5\times$ more inference memory than `UFlex`. As shown in Figure 3, `UFlex` can surpass Unisolver's throughput by increasing the factor f to 8 or 16, while maintaining a lower test error.

FLOPs Efficiency. `UFlex` does not always achieve the lowest raw Training Compute (TC), requiring 410 GFLOPs on PSWE compared to Poseidon's 387 GFLOPs. However, thanks to the less structured latent space of `UFlex`, which relies solely on matrix products rather than masking or other structured operations, it can process a larger number of FLOPs per second.

6.2 3D

We evaluate the performance of our model against different state-of-the-art baselines for 3D data: DPOT [30], MPP [51], and U-Net [64]. In this section, methods are compared under a matched computational budget (the same hardware and VRAM): since 3D MPP and 3D convolutional U-Nets are memory-intensive, the largest configurations that fit this budget are an AViT-ti MPP and a 20M-parameter U-Net, whereas `UFlex-B` has $\sim$120M parameters. To isolate the effect of model capacity from architectural design, we additionally report a parameter-matched comparison in Section G.1. We do not report the training memory usage on the CE-RSG dataset because the micro-batch sizes differ across models; we refer to Appendix D for additional discussion and to Appendix F for the full rollout comparison.

Training Efficiency. As illustrated in the right panel of Figure 5, `UFlex-B` significantly reduces training overhead compared to state-of-the-art 3D baselines. It achieves the fastest wall-clock time per training step across all resolutions, maintaining a sub-1s latency even for high-resolution datasets like CE-RSG ($256 \times 128 \times 256$). In terms of memory, `UFlex-B` utilizes the least training memory (TM) in all reported cases; for instance, in the CNS dataset, it requires only 14GB, which is $\sim 2\times$ less than DPOT (28GB) and nearly $4\times$ less than MPP (55GB).

Generalization. `UFlex-B` achieves state-of-the-art performance (nRMSE) on all three 3D datasets. It shows a particularly strong lead in the RTI dataset, where its Test Error of 0.040 is nearly $2\times$ lower than the second-best model, U-Net 3D (0.075). In the CNS dataset, `UFlex-B` (0.197) also surpasses DPOT (0.219), demonstrating its robustness in handling complex 3D spatial dynamics.

Model	TE (30%)	TE (45%)	TE (60%)	TE (75%)	TE (90%)	TE (100%)
`UFlex-B` (Fine-tuning)	**0.451**	**0.458**	**0.451**	**0.440**	**0.449**	**0.431**
`UFlex-B` (Scratch)	0.514	0.547	0.562	0.572	0.574	0.576

Table 2: **Transfer to 3D**. We report the test error on 3D CNS at different rollout horizons (30%, 45%, 60%, 75%, 90%, and 100% of the maximum horizon). The best-performing model for each column is highlighted in **bold**.

Inference Efficiency. The model achieves the highest inference throughput (T) across all benchmarks. In the CNS dataset, `UFlex-B` reaches 241 snp/s, representing a 37% increase over U-Net 3D (176 snp/s) and a nearly $3\times$ improvement over MPP 3D (87 snp/s). The higher throughput of `UFlex` relative to convolutional U-Nets is consistent with prior observations that Vision Transformers can be faster in practice than CNNs at comparable parameter counts, since attention and MLP blocks map to large, regular matrix multiplications that are easier to saturate on modern accelerators than convolutions [12]. This high throughput makes `UFlex-B` well suited for long-rollout 3D forecasting.

FLOPs Efficiency. Similarly to the 2D case, `UFlex-B` does not always report the lowest raw Training Compute (TC) in GFLOPs; for example, on RTI, MPP 3D requires fewer GFLOPs (226 vs. 1130). However, `UFlex-B` processes a larger number of FLOPs per second, which allows it to scale to high resolutions with lower wall-clock training time than models with lower theoretical FLOPs.

6.3 Additional Comparisons

In Appendix G, we further report a parameter-matched 3D comparison (Section G.1), a multi-seed evaluation (Section G.2), a comparison with the general-geometry Transformer Transolver (Section G.3), and a static prediction task (Section G.4); `UFlex` remains the best-performing model in all of them, except on VI, where it is within one standard deviation of DPOT.

7 Transferring from 2D to 3D

Transferring a model from a 2D setting to a 3D one is crucial for many practical applications, for instance, because training directly in 3D is significantly more computationally demanding. Conventional approaches often rely on structured architectures and require specialized techniques, such as inflation [82, 53], as also adopted in [51]. In contrast, `UFlex` allows us to bypass these techniques and directly transfer a model trained in 2D to a 3D setting. To evaluate this capability, we simulate a constrained computational budget scenario by limiting training to a small number of epochs (10 in our case). As shown in Table 2, the fine-tuned model demonstrates superior performance on the 3D CNS dataset.

8 Conclusion

We introduced `UFlex`, a multiscale neural physics simulator that places Transformer blocks close to the standard design inside a U-Net hierarchy operating on a 1D token sequence. Rather than relying on specialized mechanisms such as windowed attention, axial attention, Fourier layers, or dimension-specific operators, `UFlex` uses the same attention-based U-Net design across one-, two-, three-dimensional, and spatiotemporal regular-grid problems. Across a diverse set of physical simulation benchmarks, this architecture achieves state-of-the-art performance, both under compute-matched and parameter-matched comparisons, while remaining easy to adapt across resolutions and spatial dimensions.

Limitations. Our study focuses on regular-grid systems and does not address unstructured meshes, particle-based simulations, adaptive discretizations, or complex moving geometries. Like the regular-grid Transformer baselines, `UFlex` is trained and evaluated at a fixed resolution and is not resolution invariant. The multiscale design makes attention more practical at high resolution, but very large 3D or long-horizon spatiotemporal problems remain costly. Our contribution is empirical, without theoretical guarantees, and our experiments cover supervised learning from fixed datasets; limited data, distribution shift, online rollout stability, and explicit physical constraints remain future work.

Acknowledgments and Disclosure of Funding

This work received no third-party funding. The authors declare no competing interests.

References

[1] Benedikt Alkin, Maurits Bleeker, Richard Kurle, Tobias Kronlachner, Reinhard Sonnleitner, Matthias Dorfer, and Johannes Brandstetter. AB-UPT: Scaling neural CFD surrogates for high- fidelity automotive aerodynamics simulations via anchored- branched universal physics transformers. *Transactions on Machine Learning Research*, 2025.

[2] Benedikt Alkin, Andreas Fürst, Simon Lucas Schmid, Lukas Gruber, Markus Holzleitner, and Johannes Brandstetter. Universal physics transformers: A framework for efficiently scaling neural operators. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[3] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6836–6846, 2021.

[4] Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15619–15629, 2023.

[5] Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

[6] Kamyar Azizzadenesheli, Nikola Kovachki, Zongyi Li, Miguel Liu-Schiaffini, Jean Kossaifi, and Anima Anandkumar. Neural operators for accelerating scientific simulations and design. *Nature Reviews Physics*, 6(5):320–328, 2024.

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

[8] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in neural information processing systems*, 33:12449–12460, 2020.

[9] Fan Bao, Shen Nie, Kaiwen Xue, Yue Cao, Chongxuan Li, Hang Su, and Jun Zhu. All are worth words: A vit backbone for diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 22669–22679, 2023.

[10] Leah Bar and Nir Sochen. Unsupervised deep learning algorithm for pde-based forward and inverse problems. *arXiv preprint arXiv:1904.05417*, 2019.

[11] Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, Sumith Kulal, et al. Flux. 1 kontext: Flow matching for in-context image generation and editing in latent space. *arXiv e-prints*, pages arXiv–2506, 2025.

[12] Lucas Beyer. On the speed of ViTs and CNNs. Blog post, 2024.

[13] Kaifeng Bi, Lei Xie, Hongyu Zhang, et al. Accurate medium-range global weather forecasting with 3d neural networks. *Nature*, 619:533–538, 2023.

[14] Cristian Bodnar, Wessel P Bruinsma, Ana Lucic, Megan Stanley, Anna Allen, Johannes Brandstetter, Patrick Garvan, Maik Riechert, Jonathan A Weyn, Haiyu Dong, et al. A foundation model for the earth system. *Nature*, pages 1–8, 2025.

[15] Nicholas Matthew Boffi, Michael Samuel Albergo, and Eric Vanden-Eijnden. Flow map matching with stochastic interpolants: A mathematical framework for consistency models. *Transactions on Machine Learning Research*, 2025.

[16] Johannes Brandstetter, Daniel E Worrall, and Max Welling. Message passing neural pde solvers. In *International Conference on Learning Representations*, 2022.

[17] Ricardo Buitrago, Tanya Marwah, Albert Gu, and Andrej Risteski. On the benefits of memory for modeling time-dependent PDEs. In *The Thirteenth International Conference on Learning Representations*, 2025.

[18] Shuhao Cao. Choose a transformer: Fourier or galerkin. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[19] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021.

[20] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost, 2016.

[21] Enea Monzio Compagnoni, Tianlin Liu, Rustem Islamov, Frank Norbert Proske, Antonio Orvieto, and Aurelien Lucchi. Adaptive methods through the lens of SDEs: Theoretical insights on the role of noise. In *The Thirteenth International Conference on Learning Representations*, 2025.

[22] Katherine Crowson, Stefan Andreas Baumann, Alex Birch, Tanishq Mathew Abraham, Daniel Z Kaplan, and Enrico Shippole. Scalable high-resolution pixel-space image synthesis with hourglass diffusion transformers. In *Forty-first International Conference on Machine Learning*, 2024.

[23] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.

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

[25] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

[26] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, 1980.

[27] Ruchi Guo, Shuhao Cao, and Long Chen. Transformer meets boundary value inverse problems. In *The Eleventh International Conference on Learning Representations*, 2023.

[28] Jayesh K Gupta and Johannes Brandstetter. Towards multi-spatiotemporal-scale generalized PDE modeling. *Transactions on Machine Learning Research*, 2023.

[29] XU HAN, Han Gao, Tobias Pfaff, Jian-Xun Wang, and Liping Liu. Predicting physics in mesh-reduced space with temporal attention. In *International Conference on Learning Representations*, 2022.

[30] 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 *Forty-first International Conference on Machine Learning*, 2024.

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

[32] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022.

[33] Maximilian Herde, Bogdan Raonic, Tobias Rohner, Roger Käppeli, Roberto Molinaro, Emmanuel de Bezenac, and Siddhartha Mishra. Poseidon: Efficient foundation models for PDEs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[34] Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial attention in multidimensional transformers. *arXiv preprint arXiv:1912.12180*, 2019.

[35] Benjamin Holzschuh, Qiang Liu, Georg Kohl, and Nils Thuerey. PDE-transformer: Efficient and versatile transformers for physics simulations. In *Forty-second International Conference on Machine Learning*, 2025.

[36] Emiel Hoogeboom, Jonathan Heek, and Tim Salimans. simple diffusion: End-to-end diffusion for high resolution images. In *International Conference on Machine Learning*, pages 13213–13232. PMLR, 2023.

[37] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020.

[38] Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. *Advances in neural information processing systems*, 34:26548–26560, 2021.

[39] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[40] Zijie Li, Kazem Meidani, and Amir Barati Farimani. Transformer for partial differential equations' operator learning. *Transactions on Machine Learning Research*, 2023.

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

[42] Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, Anima Anandkumar, et al. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021.

[43] 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.

[44] Zongyi Li, Nikola Borislavov Kovachki, Chris Choy, Boyi Li, Jean Kossaifi, Shourya Prakash Otta, Mohammad Amin Nabian, Maximilian Stadler, Christian Hundt, Kamyar Azizzadenesheli, and Anima Anandkumar. Geometry-informed neural operator for large-scale 3d PDEs. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[45] Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *ACM/IMS Journal of Data Science*, 1(3):1–27, 2024.

[46] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

[47] Phillip Lippe, Bastiaan S. Veeling, Paris Perdikaris, Richard E Turner, and Johannes Brandstetter. PDE-refiner: Achieving accurate long rollouts with neural PDE solvers. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[48] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.

[49] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[50] Huakun Luo, Haixu Wu, Hang Zhou, Lanxiang Xing, Yichen Di, Jianmin Wang, and Mingsheng Long. Transolver++: An accurate neural solver for PDEs on million-scale geometries. In *Forty-second International Conference on Machine Learning*, 2025.

[51] Michael McCabe, Bruno Régaldo-Saint Blancard, Liam Parker, Ruben Ohana, Miles Cranmer, Alberto Bietti, Michael Eickenberg, Siavash Golkar, Geraud Krawezik, Francois Lanusse, et al. Multiple physics pretraining for spatiotemporal surrogate models. *Advances in Neural Information Processing Systems*, 37:119301–119335, 2024.

[52] Rudy Morel, Jiequn Han, and Edouard Oyallon. DISCO: learning to DISCover an evolution operator for multi-physics-agnostic prediction. In *Forty-second International Conference on Machine Learning*, 2025.

[53] Eric Nguyen, Karan Goel, Albert Gu, Gordon Downs, Preey Shah, Tri Dao, Stephen Baccus, and Christopher Ré. S4ND: Modeling images and videos as multidimensional signals with state spaces. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[54] Ruben Ohana, Michael McCabe, Lucas Thibaut Meyer, Rudy Morel, Fruzsina Julia Agocs, Miguel Beneitez, Marsha Berger, Blakesley Burkhart, Stuart B. Dalziel, Drummond Buschman Fielding, Daniel Fortunato, Jared A. Goldberg, Keiya Hirashima, Yan-Fei Jiang, Rich Kerswell, Suryanarayana Maddu, Jonah M. Miller, Payel Mukhopadhyay, Stefan S. Nixon, Jeff Shen, Romain Watteaux, Bruno Régaldo-Saint Blancard, François Rozet, Liam Holden Parker, Miles Cranmer, and Shirley Ho. The well: a large-scale collection of diverse physics simulations for machine learning. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024.

[55] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024. Featured Certification.

[56] Antonio Orvieto and Robert M. Gower. In search of adam's secret sauce. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[57] Liam Parker, Francois Lanusse, Siavash Golkar, Leopoldo Sarra, Miles Cranmer, Alberto Bietti, Michael Eickenberg, Geraud Krawezik, Michael McCabe, Rudy Morel, et al. Astroclip: a cross-modal foundation model for galaxies. *Monthly Notices of the Royal Astronomical Society*, 531(4):4990–5011, 2024.

[58] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[59] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.

[60] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[61] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[62] Maziar Raissi, Paris Perdikaris, and George 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, 2019.

[63] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[64] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[65] François Rozet, Ruben Ohana, Michael McCabe, Gilles Louppe, Francois Lanusse, and Shirley Ho. Lost in latent space: An empirical study of latent diffusion models for physics emulation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[66] Tim De Ryck, Florent Bonnet, Siddhartha Mishra, and Emmanuel de Bezenac. An operator preconditioning perspective on training in physics-informed machine learning. In *The Twelfth International Conference on Learning Representations*, 2024.

[67] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International conference on machine learning*, pages 8459–8468. PMLR, 2020.

[68] Louis Serrano, Armand Kassaï Koupaï, Thomas X Wang, Pierre ERBACHER, and Patrick Gallinari. Zebra: In-context generative pretraining for solving parametric PDEs. In *Forty-second International Conference on Machine Learning*, 2025.

[69] Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

[70] Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, Francisco Massa, Daniel Haziza, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal, Hervé Jégou, Patrick Labatut, and Piotr Bojanowski. Dinov3, 2025.

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

[72] Makoto Takamoto, Francesco Alesiani, and Mathias Niepert. Learning neural pde solvers with parameter-guided channel attention. In *International Conference on Machine Learning*, pages 33448–33467. PMLR, 2023.

[73] Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Daniel MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. Pdebench: An extensive benchmark for scientific machine learning. *Advances in Neural Information Processing Systems*, 35:1596–1611, 2022.

[74] Nils Thuerey, Philipp Holl, Maximilian Mueller, Patrick Schnell, Felix Trost, and Kiwon Um. Physics-based deep learning. *arXiv preprint arXiv:2109.05237*, 2021.

[75] Yuchuan Tian, Zhijun Tu, Hanting Chen, Jie Hu, Chao Xu, and Yunhe Wang. U-dits: Downsample tokens in u-shaped diffusion transformers. In *Advances in Neural Information Processing Systems*, 2024.

[76] Artur Toshev, Gianluca Galletti, Johannes Brandstetter, Stefan Adami, and Nikolaus A Adams. E (3) equivariant graph neural networks for particle-based fluid mechanics. In *ICLR 2023 Workshop on Physics for Machine Learning*, 2023.

[77] Alasdair Tran, Alexander Mathews, Lexing Xie, and Cheng Soon Ong. Factorized fourier neural operators. In *The Eleventh International Conference on Learning Representations*, 2023.

[78] Kiwon Um, Robert Brand, Yun Raymond Fei, Philipp Holl, and Nils Thuerey. Solver-in-the-loop: Learning from differentiable physics to interact with iterative pde-solvers. *Advances in neural information processing systems*, 33:6111–6122, 2020.

[79] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[80] Christopher Williams, Fabian Falck, George Deligiannidis, Christopher C. Holmes, Arnaud Doucet, and Saifuddin Syed. A unified framework for u-net design and analysis. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

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

[82] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *Proceedings of the European conference on computer vision (ECCV)*, pages 305–321, 2018.

[83] Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in neural information processing systems*, 32, 2019.

[84] Xuan Zhang, Jacob Helwig, Yuchao Lin, Yaochen Xie, Cong Fu, Stephan Wojtowytsch, and Shuiwang Ji. Sinenet: Learning temporal dynamics in time-dependent partial differential equations. In *The Twelfth International Conference on Learning Representations*, 2024.

[85] Maksim Zhdanov, Max Welling, and Jan-Willem van de Meent. Erwin: A tree-based hierarchical transformer for large-scale physical systems. In *Forty-second International Conference on Machine Learning*, 2025.

[86] Hang Zhou, Yuezhou Ma, Haixu Wu, Haowen Wang, and Mingsheng Long. Unisolver: PDE-conditional transformers towards universal neural PDE solvers. In *Forty-second International Conference on Machine Learning*, 2025.

A Dataset Details

All data used in this work come from two sources [54, 73]. We selected datasets based on the phenomenon's complexity, simulating research-frontier scenarios, and the datasets must allow evaluation of the model's computational and memory performance at multiple resolutions. Accordingly, we test the model in 2D at 256×256, 256×512, and 512×512, and in 3D at 64^3, 128^3, and $256 \times 128 \times 256$. In the following paragraphs, we provide the technical details on the phenomena and how the data was generated.

Temporal Resolution.

Dataset	*Active Matter (2D)*	*Planetswe (2D)*	*Viscoelastic Instability (2D)*	*Comp. NS (2D)*	*Comp. NS (3D)*	*Rayleigh–Taylor Instability (3D)*	*Convective Envelope RSG (3D)*
Temporal Horizon	81	201	20	21	21	119	100

Table 3: Maximum temporal horizon across different datasets. For all datasets, we use a temporal stride of 1 except the Planetswe, where we used a temporal stride of 5 to relieve the accumulation of error problem due to the extremely large number of temporal steps.

Active Matter. We study dense suspensions of self-propelled rod-like particles of length ℓ and diameter b, with large aspect ratio $\ell/b \gg 1$, immersed in an incompressible Stokes fluid of volume V. The system is described by a kinetic theory in terms of the particle distribution function $\Psi(x, p, t)$, where $x \in \mathbb{R}^2$ represents the spatial coordinates and $p \in \mathbb{S}^1$ denotes particle orientation. The evolution of Ψ is governed by the Smoluchowski equation

$$\frac{\partial \Psi}{\partial t} + \nabla_x \cdot (\dot{x}\Psi) + \nabla_p \cdot (\dot{p}\Psi) = 0,$$

with conformational fluxes

$$\dot{x} = u - d_T \nabla_x \log \Psi, \qquad \dot{p} = (I - pp) \cdot (\nabla u + 2\zeta D) \cdot p - d_R \nabla_p \log \Psi,$$

where $u(x, t)$ is the background fluid velocity, d_T and d_R are dimensionless translational and rotational diffusion coefficients, ζ is the strength of steric alignment interactions, and $D = \langle pp \rangle$ is the second-moment tensor. Macroscopic fields are obtained as moments of Ψ, including the concentration $c = \langle 1 \rangle$, the polarity field $n = \langle p \rangle / c$, and the nematic order parameter $Q = \langle pp \rangle / c$, with $\langle f \rangle = \int_{|p|=1} f \Psi \, dp$.

The particle dynamics are coupled to incompressible Stokes flow through

$$-\Delta u + \nabla P = \nabla \cdot \Sigma, \qquad \nabla \cdot u = 0,$$

where $P(x, t)$ is the fluid pressure and the stress tensor is given by

$$\Sigma = \alpha D + \beta S : E - 2\zeta\beta(D \cdot D - S : D).$$

Here α represents the active dipole strength, β is the particle density, $E = (\nabla u + \nabla u^\top)/2$ is the rate-of-strain tensor, and $S = \langle pppp \rangle$ is the fourth-order orientational moment. The stress includes contributions from activity, particle rigidity, and steric torques.

The system is solved numerically using a pseudo-spectral method with Fourier differentiation in both space and orientation, combined with a second-order implicit-explicit backward differentiation scheme (SBDF2), treating linear terms implicitly and nonlinear terms explicitly. Simulations are performed in a periodic square domain of size $L = 10$, with 256×256 spatial modes and 256 orientation modes, using a time step of $\Delta t = 0.0004$.

The physical parameters are varied as follows: $\alpha \in \{-1, -2, -3, -4, -5\}$, $\zeta \in \{1, 3, 5, 7, 9, 11, 13, 15, 17\}$, and $\beta = 0.8$. The observed fields in the dataset include the concentration (scalar), velocity (vector), orientation tensor (second-order tensor), and strain-rate tensor (second-order tensor), producing eleven input-output channels at a resolution of 256×256. The high-dimensional orientation space and the stiffness of the dynamics for large alignment strengths make direct simulation computationally expensive, motivating the development of surrogate models that can efficiently predict low-order moment dynamics over long time horizons.

Planetswe. We study the dynamics of incompressible fluid flows in regimes where horizontal length scales are much larger than vertical scales, allowing a depth-integrated two-dimensional

approximation known as the shallow water equations. These equations are obtained by integrating the incompressible Navier-Stokes equations in the vertical direction, leaving the fluid height $h(x,t)$ above the base as a key dynamical variable. In this dataset, we consider the rotating, forced, hyperviscous shallow water equations on a spherical domain, given by

$$\begin{aligned}\frac{\partial u}{\partial t} &= -u\cdot\nabla u - g\nabla h - \nu\nabla^4 u - 2\Omega\times u,\\ \frac{\partial h}{\partial t} &= -H\nabla\cdot u - \nabla\cdot(hu) - \nu\nabla^4 h + F(t,\phi,\theta),\end{aligned}$$

where u is the horizontal velocity, h is the fluid surface height, g is the gravitational acceleration, Ω is the planetary rotation vector, H is the mean fluid depth, and ∇^4 denotes a hyperviscosity operator with coefficient $\nu = 1.76\times10^{-10}$ chosen for numerical stability. The hyperviscosity term selectively damps small scales without significantly affecting the large-scale dynamics. The forcing term $F(t,\phi,\theta)$ introduces time-dependent seasonality with daily and annual cycles, mimicking realistic climatic variations on the Earth.

The initial conditions are derived from the 500 hPa pressure level of the ERA5 reanalysis dataset and incorporate realistic Earth topography. To ensure balanced initial states, the velocity and height fields are repeatedly simulated for short periods and projected onto hydrostatic balance before burning-in for half a model year. The subsequent simulations span three model years, with data recorded every model hour, resulting in 3024 snapshots per initial condition.

Simulations are performed using the spin-spherical harmonic pseudospectral method implemented in Dedalus, with a spatial grid oversampled by a factor of $3/2$ for anti-aliasing following Orszag's rule. The time integration employs a second-order IMEX Runge-Kutta scheme, and the time-step is dynamically adjusted according to the CFL condition. The spectral fields are interpolated onto an equiangular physical grid for analysis.

The dataset varies only in initial conditions, reflecting different atmospheric states while keeping physical parameters constant. The recorded fields include the velocity field u (vector) and the surface height h (scalar) at a resolution of 256×512. The long integration horizon, coupled with realistic topography and time-dependent forcing, presents significant challenges for long-term prediction of atmospheric dynamics.

Viscoelastic Instability. This dataset contains two-dimensional direct numerical simulations of dilute polymer solutions confined between two parallel plates. The flow exhibits multiple coexisting attractors: a laminar state, a steady arrowhead regime (SAR), a chaotic arrowhead regime (CAR), and a chaotic elasto-inertial turbulence regime (EIT). SAR corresponds to a traveling-wave solution, while CAR and EIT are visually similar; CAR features a weak polymer arrowhead structure along the mid-plane that is absent in EIT. The dataset also includes edge states, which are trajectories lying on the boundary between basins of attraction, possessing a single unstable direction and marking the transitions between distinct flow regimes. The spatial resolution of all fields is 512×512.

The flow is modeled using FENE-P viscoelastic fluids with velocity $\mathbf{u}=(u,v)$, pressure p, and polymer conformation tensor C. The governing non-dimensional equations read

$$\begin{aligned}\mathrm{Re}\left(\frac{\partial\mathbf{u}}{\partial t}+\mathbf{u}\cdot\nabla\mathbf{u}\right)+\nabla p &= \beta\Delta\mathbf{u}+(1-\beta)\nabla\cdot\mathbf{T}(C),\\ \nabla\cdot\mathbf{u} &= 0,\\ \mathbf{T}(C) &= \frac{1}{\mathrm{Wi}}\left(\frac{C}{1-\frac{\mathrm{tr}(C)-3}{L_{\max}^2}}-I\right),\\ \frac{\partial C}{\partial t}+(\mathbf{u}\cdot\nabla)C+\mathbf{T}(C) &= C\cdot\nabla\mathbf{u}+(\nabla\mathbf{u})^{\top}\cdot C+\epsilon\Delta C.\end{aligned}$$

Here, $\beta=\nu_s/\nu$ is the ratio of solvent to total viscosity, $L_{\max}$ is the maximum polymer chain extensibility, and $\epsilon=D/(U_b h)$ is the dimensionless polymer stress diffusivity. The Reynolds number $\mathrm{Re}=U_b h/\nu$ and Weissenberg number $\mathrm{Wi}=\tau U_b/h$ set the relative importance of inertia and polymer elasticity, with h the half-distance between plates, U_b the bulk velocity, and τ the polymer relaxation time.

Edge states are computed by bisection between initial conditions known to converge to distinct attractors, for example between laminar and EIT states or between EIT and SAR.

The parameters for this dataset are fixed as follows: $\mathrm{Re} = 1000$, $\mathrm{Wi} = 50$, $\beta = 0.9$, $\epsilon = 2 \times 10^{-6}$, and $L_{\max} = 70$. The recorded fields include the velocity vector $\mathbf{u}$, pressure p (scalar), and components of the positive-definite polymer conformation tensor C. Specifically, C_{xx}, C_{yy}, and C_{xy} are stored as tensor fields, while C_{zz} is recorded as a scalar field.

Compressible Navier-Stokes (2D & 3D). We investigate the temporal evolution of a compressible Newtonian fluid in d-dimensional space ($d \in \{2, 3\}$). The system is described by the state vector of conserved quantities, including mass density ρ, fluid velocity $v \in \mathbb{R}^d$, and total energy density $E = \epsilon + \rho v^2/2$, where ϵ is the internal energy. The dynamics are governed by the conservation laws of mass, momentum, and energy, expressed as the system of partial differential equations:

$$\begin{aligned} \partial_t \rho + \nabla \cdot (\rho v) &= 0, \\ \rho(\partial_t v + v \cdot \nabla v) &= -\nabla p + \eta \Delta v + (\zeta + \eta/3)\nabla(\nabla \cdot v), \\ \partial_t E + \nabla \cdot [(p + E)v - v \cdot \sigma'] &= 0, \end{aligned}$$

where p is the thermodynamic pressure, and σ' denotes the viscous stress tensor. The system is closed by the ideal gas equation of state, relating internal energy to pressure via $\epsilon = p/(\Gamma - 1)$ with the adiabatic index $\Gamma = 5/3$. The coefficients η and ζ represent the shear and bulk viscosity, respectively.

The numerical solution is computed using a finite volume method that handles the distinct regimes of the flow. The inviscid flux components are resolved using the second-order HLLC Riemann solver combined with the Monotonic Upstream-Centered Scheme for Conservation Laws (MUSCL) for spatial reconstruction to capture shock discontinuities accurately. The viscous terms are approximated using a standard second-order central difference scheme.

The physical parameters are varied to cover different regimes of compressibility and viscosity. The Mach number $M = |v|/c_s$ is selected as $M \in \{0.1, 1.0\}$ for the 2D simulations, while the 3D simulations are fixed at $M = 1.0$. The viscosity parameters are varied across $\eta, \zeta \in \{10^{-8}, 10^{-2}, 10^{-1}\}$, representing regimes from essentially inviscid Euler flow to highly viscous flow. The observed fields in the dataset include the density, velocity vector, and pressure. The 2D data are generated at a spatial resolution of 512×512, while the 3D data are provided at a resolution of 128^3. The dataset includes challenging initial conditions such as the "Riemann problem" (shock-tube), which spontaneously generates shocks and rarefaction waves, creating discontinuities that are difficult for surrogate models to resolve without producing Gibbs phenomena or violating conservation laws.

Rayleigh–Taylor Instability. The Rayleigh–Taylor instability occurs when two immiscible fluids of different densities are initially at rest, with the heavier fluid positioned above the lighter one. Any perturbation that displaces a heavier fluid parcel below a lighter one triggers the instability, as the misalignment between pressure gradients and density gradients generates vorticity, which amplifies the initial perturbation. As the instability grows, nonlinear effects dominate, leading to turbulent mixing between the two fluids. The spatial resolution of the dataset is 128^3.

The key dimensionless parameter governing the flow is the Atwood number, defined as

$$A = \frac{\rho_h - \rho_l}{\rho_h + \rho_l},$$

where ρ_h and ρ_l are the densities of the heavy and light fluids, respectively. The growth rate of the mixing layer can be characterized by

$$\alpha = \frac{\dot{L}^2}{4AgL},$$

where L is the width of the turbulent mixing zone and g is the gravitational acceleration.

The governing equations for miscible fluids with a common molecular diffusivity κ are the continuity, momentum, and incompressibility equations:

$$\begin{aligned} \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) &= 0, \\ \frac{\partial (\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u}\mathbf{u}) &= -\nabla p + \nabla \cdot \boldsymbol{\tau} + \rho \mathbf{g}, \\ \nabla \cdot \mathbf{u} &= -\kappa \nabla \cdot \left(\frac{\nabla \rho}{\rho} \right), \end{aligned}$$

with the deviatoric stress tensor defined as

$$\boldsymbol{\tau} = \rho\nu \left[\nabla \mathbf{u} + (\nabla \mathbf{u})^\top - \frac{2}{3}(\nabla \cdot \mathbf{u}) I \right],$$

where ν is the kinematic viscosity and I is the identity matrix.

The simulations are performed using the TURMIX3D code on a staggered marker-and-cell mesh with a Lagrange-plus-remap approach and a Helmholtz–Hodge decomposition for pressure. The spatial discretization is second-order accurate, using an upwind total variation diminishing scheme with Van Leer flux limiters, and time integration is performed with a second-order strong-stability-preserving Runge-Kutta method. The pressure equation is modified to account for variable density and non-zero divergence:

$$\nabla \cdot \left(\frac{1}{\rho^{n+1}} \nabla \left(\frac{\rho_l}{\rho_h - \rho_l} p^n \right) \right) = \frac{\rho_l}{\Delta t (\rho_h - \rho_l)} \nabla \cdot \left(\frac{(\rho \mathbf{u})^{\text{int}}}{\rho^{n+1}} + \kappa \frac{\nabla \rho^{n+1}}{\rho^{n+1}} \right),$$

which is solved using a red-black relaxation method combined with a V-cycle multigrid solver. The kinematic viscosity is rescaled dynamically to maintain the Kolmogorov scale near the grid resolution:

$$\nu(t) = \left(\frac{h}{2.1} \right)^4 \langle \epsilon \rangle^{1/3}, \quad \langle \epsilon \rangle = \frac{1}{\langle \rho \rangle L} \frac{d}{dt} \left(\langle \rho \rangle L (\langle E_p \rangle - \langle K \rangle) \right),$$

where $\langle \epsilon \rangle$ is the mean energy dissipation rate per unit mass and h is the mesh spacing.

The dataset includes simulations for five Atwood numbers, $A \in \{3/4, 1/2, 1/4, 1/8, 1/16\}$, with 13 different initializations. Initial conditions vary either by the mean μ and standard deviation σ of the initial Fourier spectrum $A(k)$ with $\mu \in \{1, 4, 16\}$ and $\sigma \in \{1/4, 1/2, 1\}$, or by varying the maximum phase of the Fourier components $\phi_{\max} \in \{\pi/128, \pi/8, \pi/2, \pi\}$ for a fixed mean $\mu = 16$ and $\sigma = 0.25$. The phase of each component is drawn randomly from the appropriate interval.

The recorded fields are the density ρ (scalar field) and the velocity $\mathbf{u}$ (vector field).

Convective Envelope of Red Supergiants. Massive stars in the red supergiant (RSG) phase exhibit turbulent, convective envelopes. To capture the inherently three-dimensional convective dynamics, we employ three-dimensional radiative hydrodynamic (3D RHD) simulations. These simulations are used to study phenomena such as the progenitors of supernova explosions and their early evolution, the calibration of mixing-length theory for one-dimensional stellar models, and surface granulation caused by large-scale convective plumes that influence photometric and interferometric observations. The dataset has a resolution of $256 \times 128 \times 256$ in spherical-polar coordinates.

The governing 3D RHD equations are

$$\begin{aligned}
\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) &= 0, \\
\frac{\partial (\rho \mathbf{v})}{\partial t} + \nabla \cdot (\rho \mathbf{v}\mathbf{v} + \mathbf{P}_{\text{gas}}) &= -\mathbf{G}_r - \rho \nabla \Phi, \\
\frac{\partial E}{\partial t} + \nabla \cdot [(E + P_{\text{gas}})\mathbf{v}] &= -c G_r^0 - \rho \mathbf{v} \cdot \nabla \Phi, \\
\frac{\partial I}{\partial t} + c\mathbf{n} \cdot \nabla I &= S(I, \mathbf{n}),
\end{aligned}$$

where ρ is the gas density, $\mathbf{v}$ is the flow velocity, $\mathbf{P}_{\text{gas}}$ is the gas pressure tensor, P_{gas} its scalar component, $E = E_g + \rho \mathbf{v}^2/2$ is the total energy with $E_g = 3P_{\text{gas}}/2$, I is the frequency-integrated intensity as a function of time, position, and propagation direction $\mathbf{n}$, and $\nabla\Phi$ is the gravitational acceleration with $\nabla\Phi = -Gm(r)/r^2$, where $m(r)$ is the mass enclosed within radius r. The radiation source term in the co-moving frame is

$$S_0(I_0, \mathbf{n}_0) = c\rho\kappa_a^P \left(\frac{c a_r T^4}{4\pi} - J_0 \right) + c\rho(\kappa_S + \kappa_a^R)(J_0 - I_0),$$

where κ_a^P and κ_a^R are the Planck and Rosseland mean absorption opacities, κ_S is the electron scattering opacity, and J_0 is the zeroth moment of the intensity.

The RHD equations are solved using the Godunov method implemented in Athena++, with a spherical-polar grid. The polar angle θ is discretized uniformly into 128 bins spanning $\pi/4$ to $3\pi/4$, and the

azimuthal angle ϕ into 256 bins spanning 0 to π, with periodic boundary conditions in both θ and ϕ. Ghost zones are used to enforce these boundary conditions. The radial direction uses a logarithmically spaced grid with 384 zones, giving $\delta r/r \approx 0.01$, extending sufficiently to capture wind structures or extended atmospheres.

The recorded fields include energy density E (scalar), gas density ρ (scalar), pressure P_{gas} (scalar), and velocity $\mathbf{v}$ (vector).

B Baseline

B.1 2D

We compare `UFlex` against six state-of-the-art models. For fairness, we match the number of trainable parameters to approximately 120M, corresponding to the size of `UFlex-B`. The only exception is Unisolver, for which we use a smaller parameter count due to its ViT-based architecture and complex conditioning mechanism. In the original work [86], the compressible Navier–Stokes dataset from PDEBench [73] is downsampled to a resolution of 128×128. In contrast, we retain the original resolution of 512×512. For Unisolver, we therefore use the largest model that fits within the same computational budget as the other baselines, i.e., up to the available VRAM capacity. We report below the resources that we used to implement all the baselines.

- **PDE-Transformer** [35]: We use the authors' implementation from the repository https://github.com/tum-pbs/pde-transformer, adopting their optimal hyperparameter settings. We used their Big model configuration (PDE-B).
- **Unisolver** [86]: We use the authors' implementation from the repository https://github.com/thuml/Unisolver, adopting their optimal hyperparameter settings.
- **U-Net**: We implement a U-Net with GELU as the activation function and Layer Normalization as the normalization layer. The network consists of four downsampling and upsampling stages, utilizing residual blocks with a depth of two. We additionally incorporate a conditioning mechanism similar to that used in [47]. We end up with a model with 140M parameters.
- **Poseidon** [33]: We use the authors' implementation from https://github.com/camlab-ethz/poseidon, along with the optimal hyperparameter configuration reported in the original paper. We used their Big model configuration (Poseidon-B). We adapt the model to rectangular domains to train it on Planetswe.
- **MPP** [51]: We use the authors' implementation from `https://github.com/PolymathicAI/multiple_physics_pretraining`, adopting the optimal hyperparameter configuration reported in the original paper. We used their Big model configuration (AVIT-B).
- **DPOT** [30]: We use the authors' implementation from https://github.com/HaoZhongkai/DPOT/tree/main, adopting the optimal hyperparameter configuration reported in the original paper. We used their medium model configuration (DPOT-Medium).

B.2 3D

We compare `UFlex` with three different state-of-the-art models: MPP [51], DPOT [30], and 3D U-Net.

- **DPOT** [30]: We use the authors' implementation from https://github.com/HaoZhongkai/DPOT/tree/main, adopting the optimal hyperparameter configuration reported in the original paper. We used their medium model configuration (DPOT-Medium).
- **MPP 3D** [51]: We modify the authors' implementation from `https://github.com/PolymathicAI/multiple_physics_pretraining`, adapting it for 3D data similarly to [52], adopting the optimal hyperparameter configuration reported in the original paper. We

used their tiny model configuration (AVIT-ti), as the model during training is memory-intensive. To ensure a fair comparison with the other baseline, we utilized the same computational resources.

- **U-Net 3D**: In 3D, convolutions become considerably more expensive than their 2D counterparts. Due to this, we use a 20M parameter U-Net to fit the model into the same computational resources as the other model. We implement a U-Net with GELU as the activation function and Layer Normalization as the normalization layer. The network consists of one downsampling stage and one upsampling stage, utilizing residual blocks with a depth of two.

C Metrics

Test Error. As Test error, we use the following rollout loss, the nRMSE. Therefore, the following loss for the j-th trajectory

$$\text{nRMSE}_j = \frac{1}{T}\sum_{t=1}^{T}\sqrt{\frac{\left\|\text{traj}^t_{\text{pred,j}} - \text{traj}^t_{\text{gt,j}}\right\|_2^2}{\left\|\text{traj}^t_{\text{gt,j}}\right\|_2^2}}$$

where T is the temporal horizon, $\text{traj}^t_{\text{pred},j}$ is the predicted trajectory at time t and $\text{traj}^t_{\text{gt},j}$ is the ground truth trajectory at time t. Then, we average the loss across all trajectories

$$\text{nRMSE} = \frac{1}{\mathcal{B}}\sum_{j=1}^{\mathcal{B}}\text{nRMSE}_j$$

where $\mathcal{B}$ is the number of all trajectories. **Train Speed.** We measure the wall-clock time of a single training step, including the forward pass, backward pass, and optimizer update, while excluding data loading and host-to-device transfers. Measurements are taken after an initial warm-up phase to remove startup effects, and GPU synchronization is enforced to ensure accurate timing. All experiments are conducted on the same machine and GPU with identical batch sizes and input resolutions on a single Nvidia A100.

Inference Throughput. We measure inference performance as the number of output snapshots produced per second, with the batch size fixed across all models. Inference is performed in evaluation mode with gradients disabled, and throughput is computed after a warm-up phase over multiple forward passes. All measurements are carried out on the same machine and hardware configuration on a single Nvidia A100.

GFLOPs. We report GFLOPs as the total number of floating-point operations executed per input sample, measured using PyTorch's operator-level FLOP counter on a single forward–backward pass with batch size one.

Train memory. We measure the peak memory usage on the VRAM during training.

D Additional Training details

For all the datasets, we used different batch sizes and epochs; we summarized them with the following table.

Dataset	*Active Matter (2D)*	*Planetswe (2D)*	*Viscoelastic Instability (2D)*	*Comp. NS (2D)*	*Comp. NS (3D)*	*Rayleigh–Taylor Instability (3D)*	*Convective Envelope RSG (3D)*
Batch size	64	32	32	32	64	32	32
Gradient accumulation steps	1	1	2	2	1	2	–
Epochs	100	5	25	10	75	50	50

Table 4: Additional training hyperparameters for the different simulation datasets.

By batch size, we mean the effective batch size. For example, a batch size of 32 with 2 accumulation steps means that we backpropagate twice using micro-batches of size 16, and update the weights every two backward passes. For the experiments on the Convective Envelope RSG dataset, the large

number of grid points necessitates the use of different models with varying numbers of gradient accumulation steps. For each model, we used the largest micro-batch size possible to fit with the same computational resources as the other models. DPOT 3D and U-Net 3D use 4 accumulation steps, MPP 3D uses 8 steps, while `UFlex` uses only 2 steps thanks to its memory efficiency. Moreover, to run the experiments, we increase the patch size of DPOT from 8 to 16 and of MPP from 16 to 32. Across all the experiments for MPP and DPOT, since these two models are designed to use in-context snapshots, we pass four snapshots as in-context temporal memory. We use z-score normalization for all the data, normalizing with mean and variance across each channel, so mean, variance $\in \mathbb{R}^C$, where C is the number of channels. Furthermore, to cope with outlier values in the Planetswe and Viscoelastic Instability datasets, we use a log normalization $x = \text{sign}(x) \log(1 + |x|)$ similar to [65].

E 2D Baseline Full Rollout performance

Dataset	*Model*	TE (30%)	TE (45%)	TE (60%)	TE (75%)	TE (90%)	TE (100%)
AM	PDE-T	0.598	0.813	0.954	1.031	1.089	1.114
	Unisolver	1.060	1.176	1.229	1.255	1.276	1.281
	U-Net	0.782	0.99	1.108	1.177	1.227	1.245
	Poseidon	0.717	0.933	1.072	1.152	1.212	1.237
	MPP	0.459	0.701	0.87	0.970	1.04	1.067
	DPOT	0.623	0.828	0.958	1.026	1.074	1.094
	`UFlex-B`	**0.343**	**0.57**	**0.771**	**0.887**	**0.967**	**0.999**
PSWE	PDE-T	0.404	0.486	0.537	0.706	0.583	0.595
	Unisolver	0.423	0.498	0.548	0.583	0.609	0.624
	U-Net	0.348	0.452	0.524	0.578	0.616	0.634
	Poseidon	0.300	0.396	0.457	0.506	0.541	0.561
	MPP	0.337	0.438	0.519	0.583	0.644	0.682
	DPOT	0.682	1.052	1.402	1.643	1.817	1.908
	`UFlex-B`	**0.243**	**0.319**	**0.380**	**0.434**	**0.473**	**0.493**
VI	PDE-T	0.482	0.503	0.517	0.525	0.533	0.535
	Unisolver	0.362	0.414	0.449	0.476	0.502	0.511
	U-Net	0.288	0.349	0.391	0.418	0.44	0.445
	Poseidon	0.367	0.422	0.454	0.474	0.488	0.493
	MPP	0.100	0.173	0.228	0.281	0.321	0.335
	DPOT	**0.091**	**0.142**	**0.181**	**0.221**	**0.256**	**0.268**
	`UFlex-B`	0.100	0.146	0.190	0.234	0.271	0.283
CNS	PDE-T	0.869	0.865	0.859	0.855	0.851	0.849
	Unisolver	0.403	0.501	0.580	0.645	0.7	0.728
	U-Net	0.338	0.465	0.571	0.658	0.734	0.781
	Poseidon	0.212	0.274	0.33	0.382	0.431	0.462
	MPP	0.091	0.133	0.175	0.217	0.26	0.288
	DPOT	0.122	0.19	0.256	0.317	0.372	0.406
	`UFlex-B`	**0.087**	**0.126**	**0.164**	**0.201**	**0.239**	**0.265**

Table 5: **Full Rollout performance Baseline 2D**. We report the test error at various rollout horizon (30%,45%,60%,75%,90%,100% of the maximum rollout horizon). Best and second-best performing models are indicated in **bold** and light gray, respectively.

As we can notice from Table 5 from the other temporal horizon, when it is higher than 30%, we can draw similar conclusions as for the 30% horizon. The only difference is that, in the VI case, `UFlex-B` has the same test error as MPP at the beginning of the rollout, but MPP accumulates more error than `UFlex-B` over longer horizons.

F 3D Baseline Full Rollout performance

Dataset	*Model*	TE (30%)	TE (45%)	TE (60%)	TE (75%)	TE (90%)	TE (100%)
CNS	MPP 3D	0.425	0.437	0.439	0.434	0.426	0.417
	DPOT 3D	0.219	0.235	0.247	0.260	0.266	0.269
	U-Net 3D	0.224	0.256	0.278	0.301	0.311	0.319
	`UFlex-B`	**0.197**	**0.215**	**0.230**	**0.248**	**0.257**	**0.264**
RTI	MPP 3D	0.147	0.17	0.188	0.199	0.206	0.209
	DPOT 3D	0.104	0.137	0.163	0.182	0.199	0.234
	U-Net 3D	0.075	0.114	0.151	0.184	0.212	0.263
	`UFlex-B`	**0.040**	**0.061**	**0.079**	**0.093**	**0.105**	**0.133**
CE-RSG	MPP 3D	0.349	0.454	0.537	0.603	0.660	0.691
	DPOT 3D	0.434	0.565	0.669	0.76	0.838	0.879
	U-Net 3D	0.315	0.410	0.490	0.557	0.613	0.643
	`UFlex-B`	**0.256**	**0.338**	**0.420**	**0.5**	**0.58**	**0.624**

Table 6: **Full Rollout performance Baseline 3D**. We report the test error at various rollout horizon (30%,45%,60%,75%,90%,100% of the maximum rollout horizon). Best and second-best performing models are indicated in **bold** and light gray, respectively.

As observed in Table 6 for the longer temporal horizons, when the rollout exceeds 30%, the conclusions are similar to those obtained for the 30% rollout horizon. Interestingly, on the RTI dataset, the U-Net model tends to accumulate more error compared to the other baselines; after 60% of the maximum horizon, the other two baselines outperform it.

G Additional Experiments

In this section, we report additional experiments that complement the comparisons in Section 6.

G.1 Parameter-Matched 3D Comparison

Due to the high memory requirements of the 3D convolutional U-Net and MPP, the models in Figure 5 (Section 6) have different sizes. To disentangle model capacity from architectural design, we compare the smaller `UFlex-S` (57M parameters) against MPP 3D and U-Net 3D. As reported in Table 7, `UFlex-S` still achieves the lowest test error at almost all rollout horizons, with the largest margins on RTI (0.043 vs. 0.075 at 30%) and CE-RSG (0.532 vs. 0.643 at 100%).

Dataset	*Model*	TE (30%)	TE (45%)	TE (60%)	TE (75%)	TE (90%)	TE (100%)
CNS	MPP 3D	0.425	0.437	0.439	0.434	0.426	0.417
	U-Net 3D	0.224	0.256	**0.278**	0.301	0.311	0.319
	`UFlex-S`	**0.215**	**0.255**	0.284	**0.300**	**0.309**	**0.313**
RTI	MPP 3D	0.147	0.170	0.188	0.199	0.206	0.209
	U-Net 3D	0.075	0.114	0.151	0.184	0.212	0.263
	`UFlex-S`	**0.043**	**0.081**	**0.117**	**0.146**	**0.169**	**0.184**
CE-RSG	MPP 3D	0.349	0.454	0.537	0.603	0.660	0.691
	U-Net 3D	0.315	0.410	0.490	0.557	0.613	0.643
	`UFlex-S`	**0.261**	**0.332**	**0.400**	**0.458**	**0.507**	**0.532**

Table 7: **Parameter-matched 3D comparison**. `UFlex-S` against MPP 3D and U-Net 3D, whose sizes are limited by the computational budget. We report the test error at different rollout horizons (% of the maximum horizon). Best and second-best performing models are indicated in **bold** and light gray, respectively.

G.2 Robustness to Random Seeds

To assess whether the reported margins are robust, we repeat the experiment with different random seeds on the Viscoelastic Instability dataset, which exhibits the smallest performance gap across methods. As shown in Table 8, `UFlex-B` and DPOT are within one standard deviation of each other at the beginning of the rollout, while both consistently outperform MPP at every horizon. `UFlex-B` also exhibits the lowest variance at the first horizon, and it achieves this accuracy while training $2.2\times$ faster than DPOT (Figure 4).

Model	TE (30%)	TE (45%)	TE (60%)	TE (75%)	TE (90%)	TE (100%)
MPP	0.107 ± 0.007	0.172 ± 0.001	0.224 ± 0.004	0.273 ± 0.008	0.311 ± 0.010	0.325 ± 0.010
DPOT	$\mathbf{0.096 \pm 0.005}$	$\mathbf{0.152 \pm 0.011}$	$\mathbf{0.195 \pm 0.013}$	$\mathbf{0.236 \pm 0.015}$	$\mathbf{0.272 \pm 0.016}$	$\mathbf{0.284 \pm 0.016}$
`UFlex-B`	0.098 ± 0.002	0.154 ± 0.008	0.202 ± 0.012	0.248 ± 0.014	0.286 ± 0.015	0.299 ± 0.016

Table 8: **Robustness to random seeds** on the Viscoelastic Instability dataset, where the gap across methods is the smallest. We report mean $\pm$ standard deviation of the test error over runs with different random seeds, at different rollout horizons. Best and second-best performing models are indicated in **bold** and light gray, respectively.

G.3 Comparison with a General-Geometry Transformer

Although our focus is on regular grids, we also compare `UFlex` with Transolver [81], a Transformer designed for general geometries and irregular meshes. On the four 2D datasets, `UFlex` substantially outperforms Transolver across all datasets and rollout horizons (Table 9), e.g., 0.243 vs. 1.231 on PSWE at 30%. This illustrates the well-known trade-off between generality and accuracy, and the advantages of architectures specialized for regular grids.

Dataset	*Model*	TE (30%)	TE (45%)	TE (60%)	TE (75%)	TE (90%)	TE (100%)
AM	Transolver	1.478	1.630	1.733	1.802	1.852	1.875
	`UFlex-B`	**0.343**	**0.570**	**0.771**	**0.887**	**0.967**	**0.999**
PSWE	Transolver	1.231	1.309	1.346	1.367	1.381	1.387
	`UFlex-B`	**0.243**	**0.319**	**0.380**	**0.434**	**0.473**	**0.493**
VI	Transolver	0.789	0.843	0.878	0.909	0.937	0.953
	`UFlex-B`	**0.100**	**0.146**	**0.190**	**0.234**	**0.271**	**0.283**
CNS	Transolver	0.512	0.531	0.539	0.542	0.546	0.547
	`UFlex-B`	**0.087**	**0.126**	**0.164**	**0.201**	**0.239**	**0.265**

Table 9: **Comparison with Transolver** on the 2D datasets. We report the test error at different rollout horizons (% of the maximum horizon). Best results are in **bold**.

Model	Relative Error
Unisolver	0.90
U-Net	0.79
Poseidon	0.79
PDE-Transformer	0.58
`UFlex-B`	**0.43**

Table 10: **Static prediction** on Active Matter: predicting the tenth time step directly from the initial condition.

G.4 Static Prediction

Although we focus on autoregressive time-evolution problems, the architecture of `UFlex` is not tied to temporal rollout. To verify this, we evaluate it on a static surrogate task, in which the model predicts the tenth time step of Active Matter directly from the initial condition. As shown in Table 10, `UFlex` also achieves the best performance in this setting (0.43 vs. 0.58 for PDE-Transformer), suggesting that its advantages are not limited to autoregressive time-evolution prediction.

H Rollout Visualization

In this section, we present the rollout visualization of the `UFlex-B` with the default choice of hyperparameters.

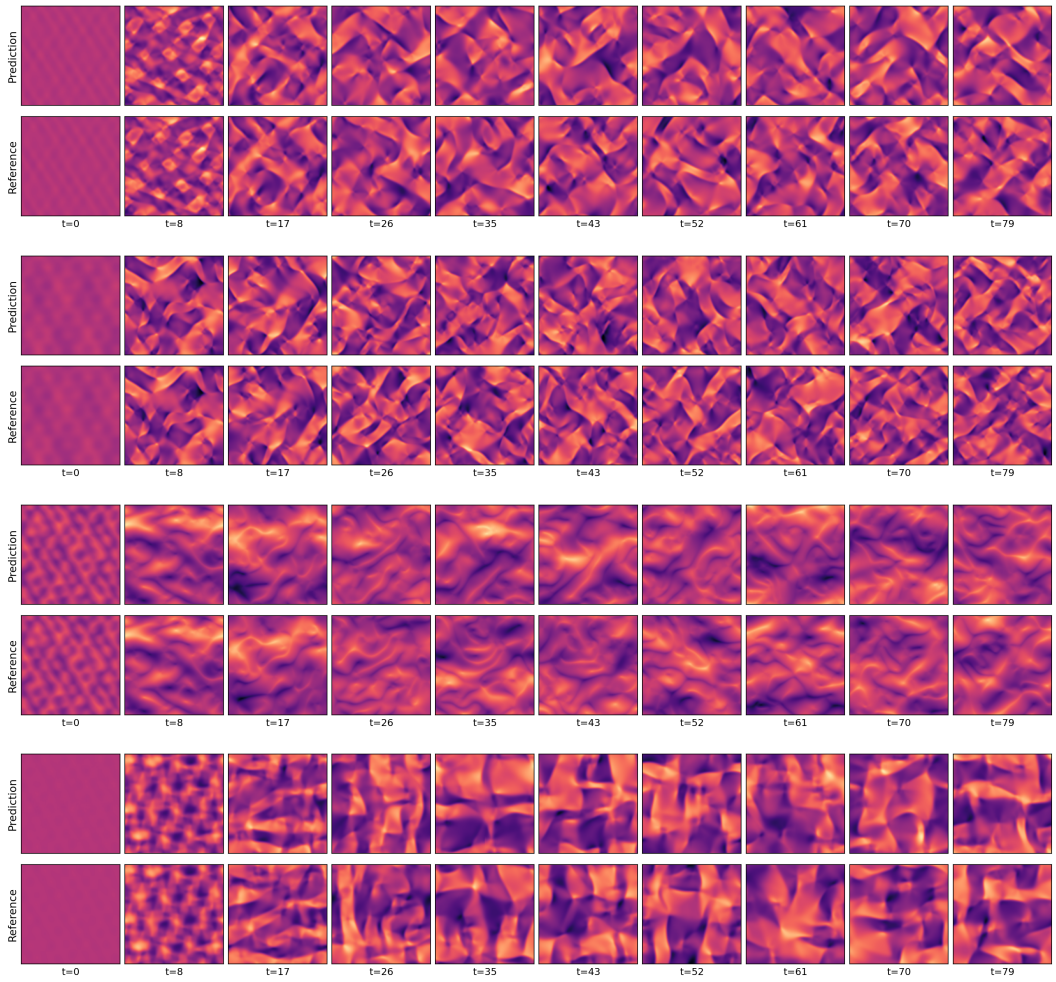


Figure 6: Active Matter rollout visualization

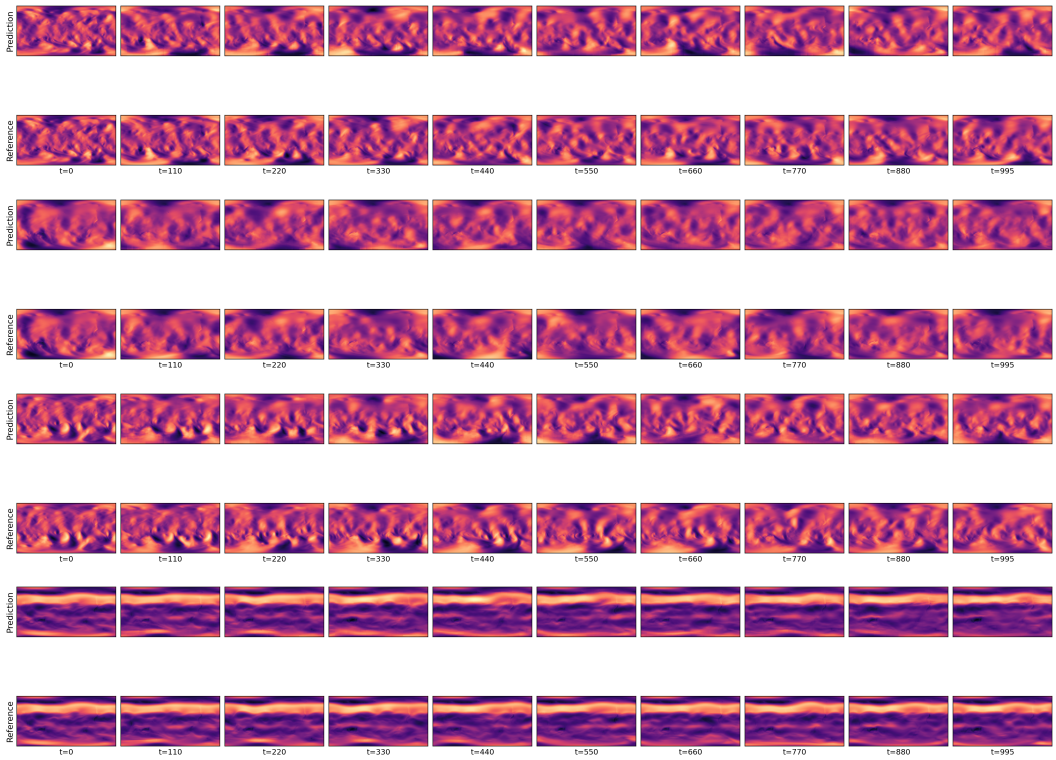


Figure 7: Planetswe rollout visualization

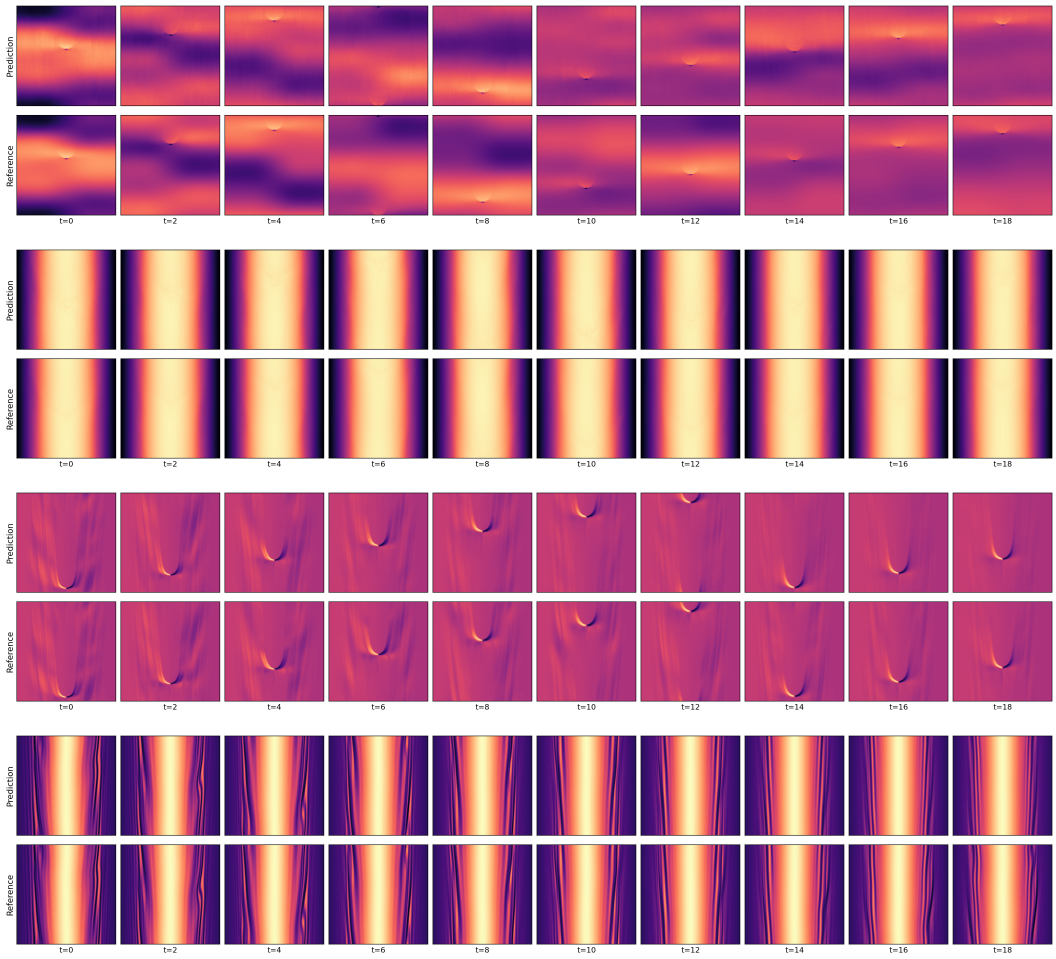


Figure 8: Viscoelastic Instability rollout visualization

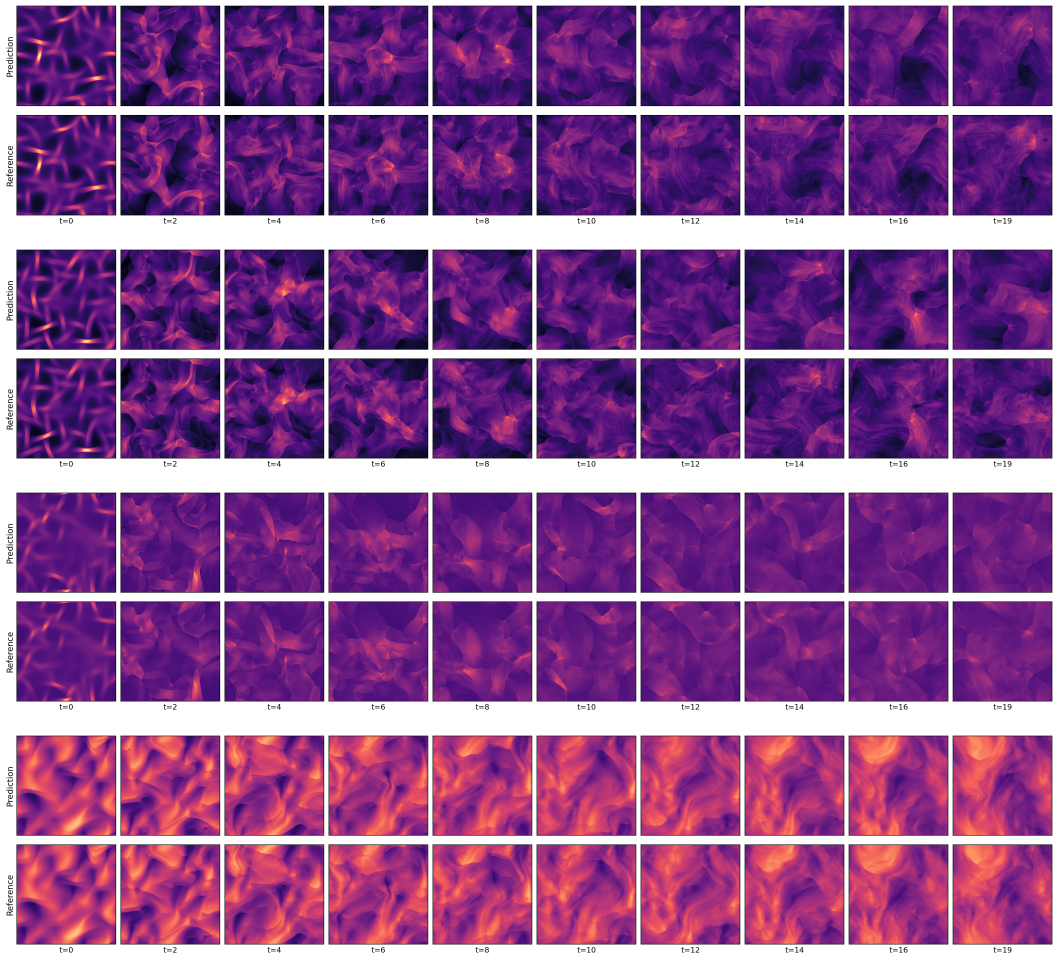


Figure 9: Compressible Navier-Stokes rollout visualization

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The claims made should match theoretical and experimental results

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We create a separate limitation section

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [N/A]

Justification: The paper does not include theoretical results.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: We discuss all the details for reproducing the experiments in the paper.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: We will provide our code for reproducing the paper, and we point out where to retrieve the data used that are publicly available.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: We specify in the paper all the details

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: We report mean $\pm$ standard deviation over runs with different random seeds for the dataset with the smallest performance gap across methods (Viscoelastic Instability, Section G.2). Due to the computational cost of training all baselines at high resolution, the remaining results are single runs.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: We report all the details in the main paper

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: We have read the NeurIPS Code of Ethics and confirm that the research presented in this paper conforms to its guidelines. To the best of our knowledge, the work does not involve ethical concerns related to human subjects, personal data, or harmful applications.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [N/A]

    Justification: There is no societal impact of the work performed.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: All original owners of assets are properly credited

    Guidelines:

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

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [Yes]

    Justification: The datasets, code, and trained models introduced in this work will be released together with documentation describing data generation, preprocessing, training procedures, licenses, intended use, and limitations. Since the datasets are synthetically generated, no personal data or consent procedures are involved. The assets will be anonymized during the review process.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: his work does not involve human subjects, crowdsourcing, user studies, or the collection of personal data. Therefore, IRB approval or equivalent ethical review was not required.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: Large language models were not used as part of the core methodology, experiments, or scientific contributions of this work.

    Guidelines:

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