TAMING PRETRAINED PHYSICS TOKENISERS UNDER DOMAIN SHIFT

Anonymous authors
Paper under double-blind review

ABSTRACT

Pretrained tokenisers offer a promising paradigm for making physics foundation models more modular and flexible, yet it remains unclear when their representations help downstream emulation and how robustly they transfer across physical systems. We systematically study these questions across three physical systems from The Well and a range of tokeniser pretraining settings, comparing emulators trained from scratch with those initialised using pretrained tokenisers. We find that pretraining improves next-step prediction and autoregressive rollout quality, while also making emulators more robust to downstream data scarcity. Under domain shift, successful transfer depends on allowing the pretrained tokeniser to adapt to the downstream system. We further find that reusing weights from a semantically related field improves transfer to physical fields unseen during pretraining. Together, these results provide practical principles for adapting and reusing pretrained physics tokenisers in downstream emulation.

1 INTRODUCTION

State-of-the-art high-resolution image and video generators are commonly trained in two stages. First, a convolutional tokeniser is pretrained to compress pixels into a compact latent representation, and a transformer then learns to predict in that latent space (Esser et al., 2021; Chang et al., 2022; Yu et al., 2023; 2024; Tschannen et al., 2024; NVIDIA et al., 2025). This separation reflects the hypothesis that convolutions suffice for capturing low-level, high-frequency structure, reserving transformers for higher-level representations where global interactions become important (Esser et al., 2021). This view is consistent with observations that transformers learn convolution-like structures at small scales (Dosovitskiy et al., 2021).

In this paradigm, the tokeniser is trained once and reused across many models. Physics foundation models are trained differently. MPP, DPOT, Poseidon, and Walrus train their encoders jointly with the dynamics model, from scratch, for each model (McCabe et al., 2024; Hao et al., 2024; Herde et al., 2024; McCabe et al., 2025). A physics tokeniser that could instead be pretrained once and reused would make these models modular.

This makes tokeniser pretraining an appealing prospect for physics foundation models, as it is for image and video modelling. However, its success in image and video modelling does not show whether a pretrained tokeniser can be reused across physical systems. For natural images and videos, the three colour channels have consistent semantics across datasets, whereas in physical systems, channels represent different physical fields, each with its own semantic meaning, tensorial character (scalar, vector, tensor), and units. More importantly, the physical fields themselves vary across datasets. Euler flow evolves density, energy, pressure, and momentum, whereas Rayleigh-Bénard convection evolves buoyancy, pressure, and velocity. Moreover, different physical systems are governed by different equations and can therefore exhibit different small-scale structures—the features that the tokeniser must learn to represent. It is therefore unclear whether a tokeniser pretrained on a set of physical systems will generalise to unseen systems, potentially limiting its reusability.

Our contributions. We systematically study pretrained physics tokenisers under both in-domain and out-of-domain transfer across three physical systems. We investigate their effect on next-step prediction and autoregressive rollouts, their behaviour under data scarcity, and whether semantic relationships between physical fields can facilitate transfer. Our main findings are:

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

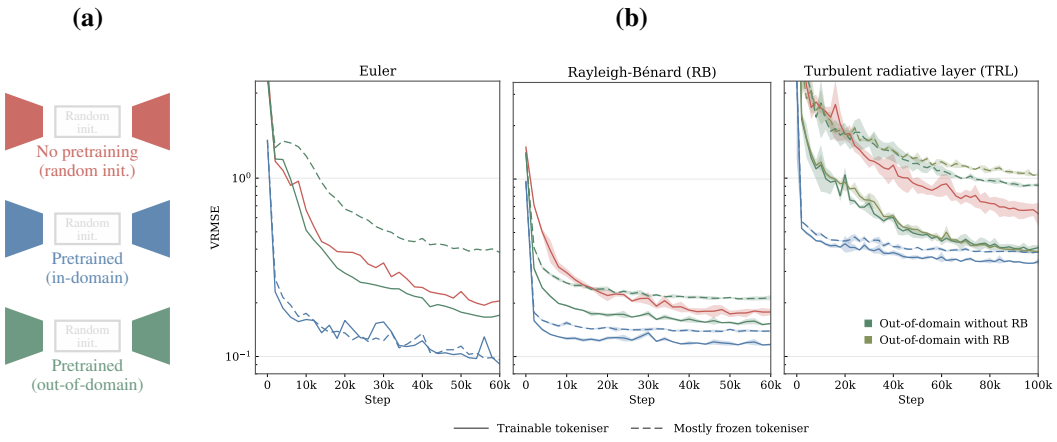

Figure 1: (a) The emulator consists of a convolutional tokeniser and a transformer-based processor. We compare its performance under three tokeniser pretraining settings: no pretraining, in-domain, and out-of-domain. The processor is always initialised randomly. (b) Validation next-step VRMSE learning curves for three experiment sets: Euler, RB, and TRL. Emulators with pretrained tokenisers reach lower errors than those without pretraining. For out-of-domain tokenisers, lower errors are achieved only when the tokeniser is trainable. In the RB and TRL panels, shaded regions indicate $\pm 1\sigma$ across three random initialisations. Euler experiments use a single initialisation.

1. **Pretraining improves emulator training.** Pretrained tokenisers accelerate convergence and reduce final next-step prediction error. These benefits extend to out-of-domain pretraining when the tokeniser remains trainable, while freezing an out-of-domain tokeniser can substantially degrade performance.
2. **Pretraining improves autoregressive rollouts.** In-domain pretraining consistently improves rollout stability, and freezing an in-domain tokeniser can further improve long-horizon stability despite not yielding the lowest next-step error.
3. **Pretraining becomes increasingly valuable under data scarcity.** As downstream training data decrease, pretrained tokenisers degrade substantially less than training from scratch, with the largest relative benefits appearing in the most data-scarce regimes.
4. **Semantic relationships between physical fields facilitate transfer.** When a target field is absent during pretraining, initialising its tokeniser head from a semantically related pretrained field improves performance, whereas copying from an unrelated field provides little or no benefit.

2 Methods

2.1 Data

We use *The Well* (Ohana et al., 2025), a large-scale collection of diverse physics simulations designed for machine learning applications. We use five 2D datasets: Euler multiquadrants, Rayleigh-Bénard (RB) convection, shear flow, active matter, and turbulent radiative layer (TRL). Each dataset contains training, validation, and test splits. We further divide the training set into two disjoint halves, using one exclusively for tokeniser pretraining and the other exclusively for emulator training.

Each example is a sequence of T physical states extracted from a simulation trajectory and is represented as $\mathbf{x} \in \mathbb{R}^{C \times T \times H \times W}$, where C denotes the number of physical fields. The physical fields (e.g., pressure, density, and momentum) vary across datasets. We use sequences of length $T = 10$, with a temporal stride[1] sampled uniformly from $\{1, 2, 3\}$. Additional information about the datasets, including their dimensions, number of examples, and physical fields, is provided in Appendix A.

[1]The temporal stride denotes the spacing between consecutive frames, measured in simulation time steps; a stride of k selects every k-th simulation frame.

2.2 EXPERIMENTS

Our experiments compare emulators under three tokeniser pretraining settings (Figure 1a):

- **No pretraining**: The tokeniser and the rest of the emulator are randomly initialised.
- **In-domain**: The tokeniser is pretrained on the same physical system as the emulator.[2]
- **Out-of-domain** The tokeniser is pretrained on a mixture of physical systems different from the emulator's.

We conduct three experiment sets, each using one of the Euler, RB, or TRL datasets as the emulation target. Figure 2 summarises the experiment sets, including the emulation dataset, its physical fields, and the out-of-domain dataset mixture.

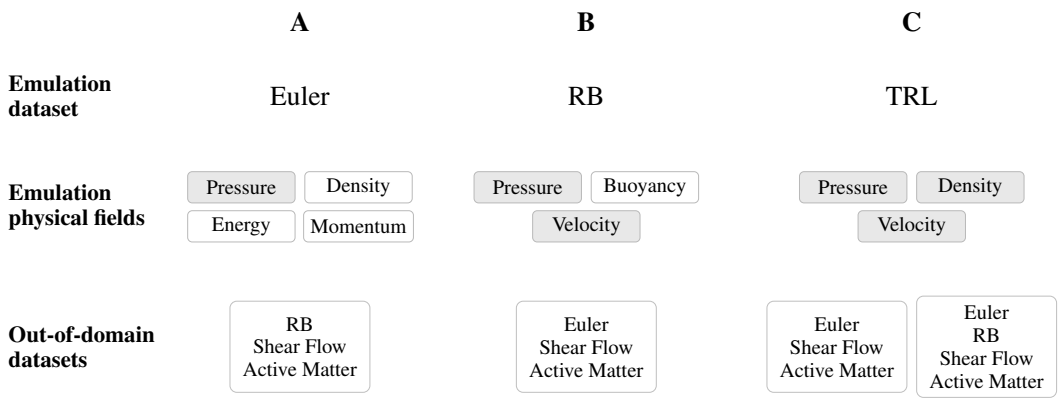

Figure 2: Overview of the experiment sets. Shaded physical fields are the ones that also appear in the out-of-domain pretraining data. TRL is trained using two different out-of-domain mixtures.

2.3 EVALUATION METRICS

VRMSE. We assess reconstruction quality using variance-normalised root mean squared error:

$$\mathrm{VRMSE} = \frac{\|\mathbf{x} - \hat{\mathbf{x}}\|_2}{\sigma_{\mathbf{x}}}, \tag{1}$$

where $\sigma_{\mathbf{x}}$ is the standard deviation of the target field $\mathbf{x}$ (computed over all pixels). This measures reconstruction error relative to the variability in the target. We compute VRMSE per timeframe and per physical field, then average across both to obtain aggregate metrics.

Decorrelation time. To measure how long an autoregressive rollout remains correlated with the true trajectory, we compute the Pearson correlation $r_t = \mathrm{PearsonCorr}(\hat{\mathbf{x}}_t, \mathbf{x}_t)$ between the predicted and true fields over all spatial locations at each rollout timestep t.

The decorrelation time is the index of the first rollout timestep at which the correlation falls below a threshold r^*. If the correlation never falls below r^*, the value is the rollout length T_{roll}:

$$\tau_{\text{dec}}(r^*) = \min\big(\{t : r_t < r^*\} \cup \{T_{\text{roll}}\}\big). \tag{2}$$

We compute decorrelation time per physical field and report the average across fields.

Normalised error power spectrum (NEPS). To check at which spatial scales the errors lie, we also report a spectral error, the power spectrum of the residual divided by that of the true field, within low, mid and high wavenumber bands (Appendix D).

Next-step metrics are evaluated on 2,560 validation examples for Euler and RB and 828 for TRL. Rollout metrics are evaluated on 500, 175, and 9 validation trajectories for Euler, RB, and TRL, respectively.

[2]Since the tokeniser and emulator are trained on disjoint data splits, an in-domain tokeniser has never seen the trajectories its emulator is trained on.

2.4 Training Objective

The emulator predicts the temporal evolution of physical systems. Given a sequence of physical states $\mathbf{x}_{0:t-1}$, it predicts the next frame $\hat{\mathbf{x}}_t$ and is trained to minimise mean absolute error (MAE):

$$\ell_{\text{Emulator}} \propto \|\mathbf{x}_t - \hat{\mathbf{x}}_t\|_1 \tag{3}$$

The tokeniser is trained with an autoencoding objective, minimising mean squared error (MSE) between the reconstruction $\hat{\mathbf{x}}_{0:t-1}$ and input sequence $\mathbf{x}_{0:t-1}$:

$$\ell_{\text{Tokeniser}} \propto \|\mathbf{x}_{0:t-1} - \hat{\mathbf{x}}_{0:t-1}\|_2^2 \tag{4}$$

2.5 Architecture

The emulator consists of two components: (i) a tokeniser that compresses spatiotemporal data into compact latent representations, and (ii) a transformer-based processor that operates on these tokens to predict future timesteps.

Processor. We use the processor architecture from Walrus (McCabe et al., 2025), which employs factorised spatial and temporal attention with axial positional encodings and a causal temporal structure. We refer readers to McCabe et al. (2025) for detailed architectural specifications.

Tokeniser. We use a simplified version of MAGVIT-2 (Yu et al., 2024), retaining the causal-convolutional encoder-decoder backbone but removing vector quantisation, adversarial and perceptual losses, and adaptive group normalisation. The tokeniser is trained solely with MSE reconstruction on continuous, non-quantised latents. This architecture supports runtime-adjustable spatiotemporal compressions for causal convolutions. Details are provided in Appendix B.

The tokeniser handles different physical fields using field-specific input and output heads around a shared backbone. Each field is mapped into the shared representation by its input head and reconstructed by its output head. This allows the same tokeniser to operate across datasets containing different sets of physical fields.

The tokeniser's encoder progressively increases channel dimensions from the number of physical fields to 16, 32, and 64 channels, then compresses to a bottleneck of 18 latent channels. The decoder mirrors this structure. The processor consists of six transformer blocks with embedding dimension 1088. A fully connected layer projects the tokeniser output to the processor dimension. Table 1 summarises the parameter counts for each component.

Table 1: Parameter counts for the tokeniser and downstream components.

Tokeniser	Processor & Projection	Total
5 M	100 M	105 M

2.6 Training

Freezing strategy. During emulator training, the processor and projection layers are always trainable. For the tokeniser, we consider two training strategies. In the *trainable* setting, all tokeniser parameters are updated jointly with the rest of the emulator. In the *mostly frozen* setting, the core of the tokeniser is frozen, while the layers directly interfacing with pixel and latent space remain trainable. Specifically, these comprise the encoder head, decoder head, and bottleneck layers, while all intermediate blocks remain frozen. Table 2 reports the number of trainable parameters under each strategy. In the *mostly frozen* setting, only 2% of the tokeniser parameters remain trainable.

From each 10-frame sequence described in Subsection 2.1, the first 9 frames are used for autoencoding during tokeniser pretraining and as context for predicting the 10th frame during emulator training. When pretraining on multiple datasets, we sample each dataset with equal probability. We pretrain tokenisers for 170k steps using the SOAP optimiser (Vyas et al., 2025). Emulators are trained for 60k steps for Euler and RB and 100k steps for TRL, using the AdamW optimiser (Loshchilov & Hutter, 2019) with a learning-rate scheduler. Both stages use distributed data parallel (DDP) training on 8

Table 2: Parameter training strategies and corresponding trainable parameter budgets.

Strategy	Tokeniser	# Trainable	Trainable Fraction
Trainable (🔓)	5 M	5 M	100 %
Mostly frozen (🔒)		85 k	2 %

GPUs with an effective batch size of 16 (two examples per GPU). The complete training recipe is provided in Appendix C. Tokeniser pretraining is performed on NVIDIA H100 or GH200 GPUs, while emulator training is performed on AMD Instinct MI355X or NVIDIA GH200 GPUs.

3 Results

3.1 Next-step prediction

Figure 1b shows validation next-step VRMSE throughout emulator training. Across all three systems, both in-domain and out-of-domain pretraining accelerate convergence and reduce error relative to no pretraining when the tokeniser remains trainable. Freezing an in-domain tokeniser preserves much of the benefit of pretraining. In contrast, freezing an out-of-domain tokeniser substantially degrades performance, resulting in higher final errors than no pretraining on all three systems.

Table 3 reports validation next-step VRMSE at the end of emulator training. With a trainable tokeniser, in-domain pretraining reduces final VRMSE relative to no pretraining by 56% on Euler, 34% on RB, and 47% on TRL, while out-of-domain pretraining reduces it by 17%, 14%, and 35%, respectively. Frozen in-domain tokenisers retain much of this improvement, whereas frozen out-of-domain tokenisers perform worse than no pretraining across all systems. In RB and TRL, the largest variation across random initialisations occurs for no pretraining and frozen out-of-domain pretraining.

Table 3: Next-step validation VRMSE at the end of emulator training. Values for RB and TRL are mean $\pm 1\sigma$ across three random initialisations; while Euler experiments use a single initialisation. The 🔓 and 🔒 symbols indicate *trainable* and *mostly frozen* tokenisers, respectively. Lower is better.

	Euler	RB	TRL
🔓 **No pretraining**	0.205	0.178 ± 0.007	0.633 ± 0.094
🔓 **In-domain**	0.091	0.117 ± 0.002	0.334 ± 0.002
🔒 **In-domain**	0.096	0.139 ± 0.002	0.385 ± 0.001
🔓 **Out-of-domain**	0.170	0.154 ± 0.001	0.414 ± 0.007
🔒 **Out-of-domain**	0.384	0.215 ± 0.007	0.914 ± 0.032

To examine how prediction errors are distributed across spatial scales, we also report the normalised error power spectrum in Appendix D.

3.2 Rollout quality

Figure 3 shows validation decorrelation time throughout emulator training for a correlation threshold of $r^* = 0.8$. Across all three systems, in-domain pretraining increases decorrelation time relative to no pretraining, with predicted trajectories remaining correlated with the ground truth for longer. The effect of out-of-domain pretraining is less consistent: keeping the tokeniser trainable increases decorrelation time for Euler and TRL, but not for RB, while mostly freezing it substantially reduces decorrelation time relative to its trainable counterpart.

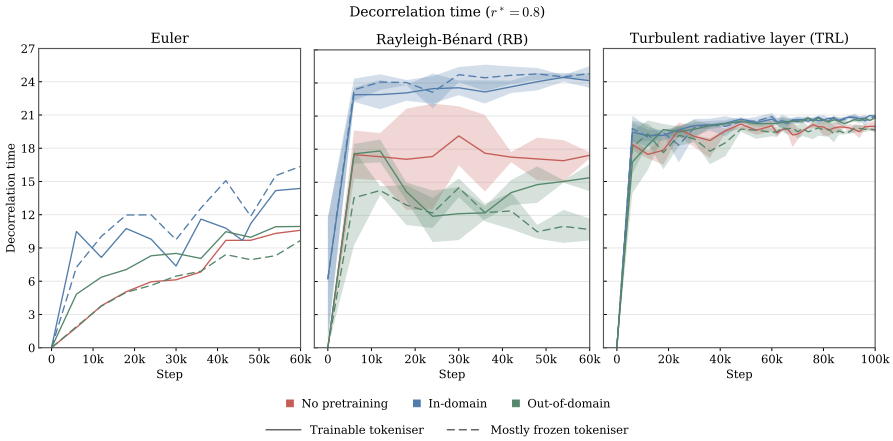

Figure 3: Decorrelation time learning curves on the validation set for different pretraining and freezing settings. Each panel shows one of the Euler, RB, and TRL experiment sets.

Interestingly, for Euler and RB, frozen in-domain tokenisers achieve longer decorrelation times than their trainable counterparts. This contrasts with next-step prediction, where trainable in-domain tokenisers achieve lower errors, and suggests that freezing a well-aligned pretrained representation may act as a form of regularisation during autoregressive rollouts.

3.3 Behaviour under data scarcity

We next investigate whether pretraining becomes more valuable when downstream training data are limited. We repeat the RB experiments using 2% (2,674 examples) and 0.3% (382 examples) of the original training data, and the TRL experiments using 10% (276 examples). All other training settings are kept unchanged.

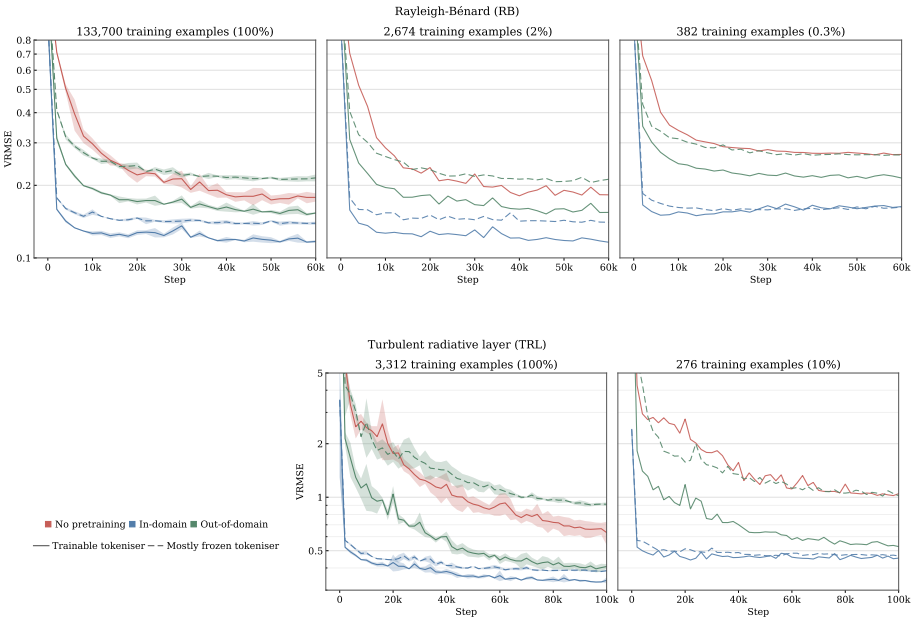

Figure 4: Validation next-step VRMSE learning curves under data scarcity for RB (top) and TRL (bottom). Columns correspond to progressively smaller emulator training sets, with the number and percentage of training examples shown above each panel. Shaded regions in the 100% panel indicate 1σ across three random initialisations.

Figure 4 shows that the advantage of pretraining grows as downstream training data become scarce. In both RB and TRL, models trained without pretraining degrade substantially as the amount of training data decreases, whereas models with in-domain pretrained tokenisers remain comparatively robust. This is particularly pronounced in the most data-scarce RB setting, where both trainable and mostly frozen in-domain tokenisers achieve substantially lower error than no pretraining. In TRL, trainable out-of-domain pretraining also degrades less under data scarcity than no pretraining.

3.4 Semantic transfer between physical fields

One possible limitation of out-of-domain pretraining is incomplete overlap in physical fields: some fields required by the downstream emulator are absent during pretraining, and therefore lack pretrained field-specific heads. We investigate whether knowledge from pretrained fields can be transferred to such unseen fields through their semantic relationships.

For each of the Euler and RB experiment sets, we select one such unseen target field and compare two weight-copying strategies: copying from a semantically related field and copying from an unrelated field. We consider fields related when they represent physically analogous quantities or play similar roles in the dynamical evolution equations. Specifically, for Euler, we initialise the momentum head from either velocity (related) or pressure (unrelated), while for RB, we initialise the buoyancy head from either tracer (related) or pressure (unrelated). In Euler, momentum is directly related to velocity, whereas pressure represents a distinct physical quantity. In RB, buoyancy and tracer are both advected scalar fields, whereas pressure plays a distinct role in the dynamics.

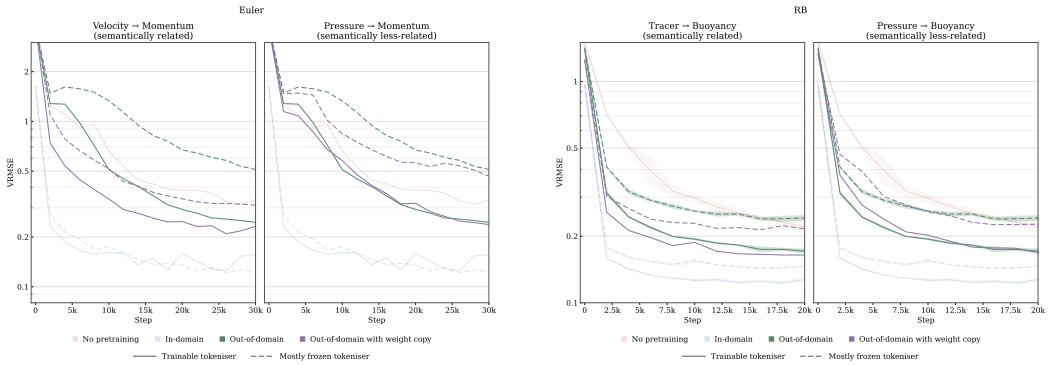

Figure 5: Validation next-step VRMSE learning curves for weight-copying experiments on Euler (left) and RB (right). For Euler, the momentum head is initialised from velocity (related) or pressure (unrelated); for RB, the buoyancy head is initialised from tracer (related) or pressure (unrelated). The original out-of-domain initialisation is shown for comparison.

Figure 5 shows the resulting next-step VRMSE learning curves. Copying weights from a related field improves performance over the standard out-of-domain initialisation in both experiment sets. In contrast, copying from an unrelated field provides no improvement and can even degrade performance. These results suggest that physical similarity between fields provides a useful prior for transferring tokeniser-head weights to fields unseen during pretraining.

4 Related work

Latent emulators with task-trained autoencoders. Several PDE emulators learn an autoencoder alongside latent dynamics, including LE-PDE (Wu et al., 2022), the latent neural PDE solver of Li et al. (2025), and AROMA (Serrano et al., 2024). LE-PDE finds that pretraining the autoencoder for reconstruction and freezing it before training the dynamics performs worse than joint training; LNOP reports a similar result despite pretraining its autoencoder across multiple PDE systems (Wu et al., 2022; Wang & Wang, 2025). Consistently, our trainable in-domain tokenisers outperform their mostly frozen counterparts, although both outperform training from scratch (Table 3).

Pretraining and transfer in physics foundation models. Physics foundation models including MPP, DPOT, Poseidon, UPT, CoDA-NO, and Walrus pretrain across physical systems and transfer to downstream tasks (McCabe et al., 2024; Hao et al., 2024; Herde et al., 2024; Alkin et al., 2024; Rahman et al., 2024; McCabe et al., 2025). These approaches transfer the representation and dynamics components together, rather than isolating the contribution of tokeniser pretraining. Masliaev et al. (2025) instead freeze a pretrained operator core and learn new input and output adapters for downstream problems. Zhou & Farimani (2024) pretrain PDE encoders with masked autoencoding and transfer them to downstream tasks, while Phaedra (Lingsch et al., 2026) develops a discrete tokeniser for physical fields and demonstrates reconstruction on unseen PDE families.

Reusing pretrained tokenisers. In vision, tokenisers such as VQGAN, MAGVIT-2, and Cosmos are pretrained and reused by downstream generators (Esser et al., 2021; Yu et al., 2024; NVIDIA et al., 2025). Recent work also investigates whether these tokenisers should remain frozen or be adapted jointly with the generator, including ETT (Wang et al., 2025) and REPA-E (Leng et al., 2025). PhysiX (Nguyen et al., 2025) transfers this paradigm to physics by initialising from a Cosmos video tokeniser and training on The Well. Unlike PhysiX, our experiments vary the tokeniser initialisation while keeping the emulator initialisation fixed, allowing us to isolate the effect of tokeniser pretraining.

Initialising weights for new inputs. Extending pretrained models to variables unseen during pretraining requires initialising new input and output parameters. ClimaX uses variable-specific embeddings and introduces new embeddings for unseen variables (Nguyen et al., 2023), while Aurora reuses pretrained wind embeddings when adapting to new ocean-wave variables (Bodnar et al., 2025). Lehmann et al. (2026) further show that new variables are better predicted from Aurora's latent representations when they are physically related to variables seen during pretraining. Similar ideas appear in language models, where WECHSEL and FOCUS initialise embeddings for new tokens using semantically related pretrained tokens (Minixhofer et al., 2022; Dobler & de Melo, 2023). Our weight-copying experiment applies this principle to field-specific heads of a physics tokeniser, directly comparing related and unrelated source fields.

5 Discussion

5.1 Key Findings

Our results show that physics tokenisers can be pretrained and successfully reused for downstream emulation. In-domain pretraining consistently improves next-step prediction and rollout quality, while also making emulators more robust to limited downstream data. Out-of-domain tokenisers can also transfer successfully, but require adaptation: keeping them trainable improves next-step prediction over training from scratch across all three systems, whereas mostly freezing them substantially degrades performance. Together, these results suggest that reusable physics tokenisers are feasible, but that representations pretrained on different physical systems should not generally be treated as fixed feature extractors.

That said, freezing has an interesting effect when pretraining and downstream domains are aligned. Frozen in-domain tokenisers retain much of their next-step advantage and, for Euler and RB, produce more stable autoregressive rollouts than their trainable counterparts, suggesting that restricting adaptation can sometimes benefit long-horizon prediction. Our field-transfer experiments further show that incomplete field overlap need not require learning unseen field heads entirely from scratch: initialisation from a physically related pretrained field improves performance over the default out-of-domain initialisation in both Euler and RB. This suggests that physical relationships between fields provide a useful prior when adapting tokenisers to new systems.

These findings motivate pretrained physics tokenisers as reusable resources: a single tokeniser pretrained across diverse systems could provide useful initialisation for many downstream emulators, particularly when downstream data are scarce.

5.2 Limitations

Our experiments focus on three downstream physical systems and a single tokeniser architecture, leaving the generality across broader classes of physical systems and tokeniser designs to future

work. While RB and TRL results are averaged across three random initialisations, computational constraints limited Euler to a single initialisation. The semantic-transfer study is similarly focused: we test two target fields, momentum in Euler and buoyancy in RB, with two source initialisations for each. Extending these experiments to a broader range of physical fields and relationships would help establish how generally semantic similarity can guide transfer.

Tokeniser pretraining introduces an upfront computational cost, which can be amortised through reuse across downstream models; quantifying this trade-off at scale remains for future work. In addition, some learning curves are still improving at the end of training, leaving room for further gains with longer training. Finally, we evaluate tokeniser transfer only for autoregressive emulation; whether the same benefits extend to other downstream tasks remains to explore.

5.3 Future Directions

A central question is how to construct a general-purpose physics tokeniser. While our results show that multi-system out-of-domain pretraining can transfer successfully, it does not outperform in-domain pretraining. Future work should investigate how the composition and diversity of the pretraining data affect downstream transfer, and whether these benefits improve with larger pretraining datasets and tokenisers.

Our study also fixes the tokeniser architecture and evaluates transfer only for autoregressive emulation. Exploring alternative tokeniser architectures and objectives, and extending evaluation to other downstream tasks, will help determine how broadly pretrained physics representations can be reused. Finally, our semantic-transfer results motivate more systematic approaches for adapting tokenisers to unseen physical fields, including methods that explicitly exploit relationships between physical variables.

6 Conclusion

We investigate whether pretrained physics tokenisers can serve as reusable components for physical emulation. Across three physical systems, we find that pretraining improves downstream performance and makes emulators more robust to data scarcity. Tokenisers pretrained on other physical systems can also transfer successfully when allowed to adapt, while our field-copy experiments show that physical relationships between fields can provide a useful prior when adapting to unseen variables. Together, these results support a modular approach to physics emulation in which tokenisers are pretrained once and subsequently adapted and reused across downstream systems.

REFERENCES

Benedikt Alkin, Andreas Fürst, Simon Schmid, Lukas Gruber, Markus Holzleitner, and Johannes Brandstetter. Universal physics transformers: A framework for efficiently scaling neural operators. In Advances in Neural Information Processing Systems, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/2cd36d327f33d47b372d4711edd08de0-Abstract-Conference.html.

Cristian Bodnar, Wessel P. Bruinsma, Ana Lucic, Megan Stanley, Anna Allen, Johannes Brandstetter, Patrick Garvan, Maik Riechert, Jonathan A. Weyn, Haiyu Dong, Jayesh K. Gupta, Kit Thambiratnam, Alexander T. Archibald, Chun-Chieh Wu, Elizabeth Heider, Max Welling, Richard E. Turner, and Paris Perdikaris. A foundation model for the earth system. Nature, 641(8065):1180–1187, 2025.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T. Freeman. Maskgit: Masked generative image transformer, 2022. URL https://arxiv.org/abs/2202.04200.

Konstantin Dobler and Gerard de Melo. FOCUS: Effective embedding initialization for monolingual specialization of multilingual models. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pp. 13440–13454, 2023.

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, 2021. URL https://arxiv.org/abs/2010.11929.

Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis, 2021. URL https://arxiv.org/abs/2012.09841.

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, 2024. URL https://arxiv.org/abs/2403.03542.

Maximilian Herde, Bogdan Raonić, Tobias Rohner, Roger Käppeli, Roberto Molinaro, Emmanuel de Bézenac, and Siddhartha Mishra. Poseidon: Efficient foundation models for pdes, 2024. URL https://arxiv.org/abs/2405.19101.

Fanny Lehmann, Firat Ozdemir, Benedikt Soja, Torsten Hoefler, Siddhartha Mishra, and Sebastian Schemm. Fine-tuning a weather foundation model with lightweight decoders for unseen physical processes. Journal of Geophysical Research: Machine Learning and Computation, 3(4), 2026.

Xingjian Leng, Jaskirat Singh, Yunzhong Hou, Zhenchang Xing, Saining Xie, and Liang Zheng. REPA-E: Unlocking VAE for end-to-end tuning with latent diffusion transformers. In Proceedings of the IEEE/CVF International Conference on Computer Vision, 2025. URL https://openaccess.thecvf.com/content/ICCV2025/html/Leng_REPA-E_Unlocking_VAE_for_End-to-End_Tuning_of_Latent_Diffusion_Transformers_ICCV_2025_paper.html.

Zijie Li, Saurabh Patil, Francis Ogoke, Dule Shu, Wilson Zhen, Michael Schneier, John R. Buchanan, and Amir Barati Farimani. Latent neural PDE solver: a reduced-order modelling framework for partial differential equations. Journal of Computational Physics, 524:113705, 2025. doi: 10.1016/j.jcp.2024.113705. URL https://doi.org/10.1016/j.jcp.2024.113705.

Levi Lingsch, Georgios Kissas, Johannes Jakubik, and Siddhartha Mishra. Phaedra: Learning high-fidelity discrete tokenization for the physical science, 2026.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

Mikhail Masliaev, Dmitry Gusarov, Ilya Markov, and Alexander Hvatov. Towards universal neural operators through multiphysics pretraining, 2025. NeurIPS 2025 Workshop on Machine Learning and the Physical Sciences.

Michael McCabe, Bruno Régaldo-Saint Blancard, Liam Holden Parker, Ruben Ohana, Miles Cranmer, Alberto Bietti, Michael Eickenberg, Siavash Golkar, Geraud Krawezik, Francois Lanusse, Mariel Pettee, Tiberiu Tesileanu, Kyunghyun Cho, and Shirley Ho. Multiple physics pretraining for physical surrogate models, 2024. URL https://arxiv.org/abs/2310.02994.

Michael McCabe, Payel Mukhopadhyay, Tanya Marwah, Bruno Regaldo-Saint Blancard, Francois Rozet, Cristiana Diaconu, Lucas Meyer, Kaze W. K. Wong, Hadi Sotoudeh, Alberto Bietti, Irina Espejo, Rio Fear, Siavash Golkar, Tom Hehir, Keiya Hirashima, Geraud Krawezik, Francois Lanusse, Rudy Morel, Ruben Ohana, Liam Parker, Mariel Pettee, Jeff Shen, Kyunghyun Cho, Miles Cranmer, and Shirley Ho. Walrus: A cross-domain foundation model for continuum dynamics, 2025. URL https://arxiv.org/abs/2511.15684.

Benjamin Minixhofer, Fabian Paischer, and Navid Rekabsaz. WECHSEL: Effective initialization of subword embeddings for cross-lingual transfer of monolingual language models. In Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 3992–4006, 2022.

Payel Mukhopadhyay, Michael McCabe, Ruben Ohana, and Miles Cranmer. Compute-adaptive surrogate modeling of partial differential equations. In ICLR 2025 Workshop on Machine Learning for Multi-Phase Phenomena (MLMP), March 2025. URL https://openreview.net/forum?id=YM3koX4nHp. Poster.

Tung Nguyen, Johannes Brandstetter, Ashish Kapoor, Jayesh K Gupta, and Aditya Grover. ClimaX: A foundation model for weather and climate. In Proceedings of the 40th International Conference on Machine Learning, volume 202 of Proceedings of Machine Learning Research, pp. 25904–25938, 2023.

Tung Nguyen, Arsh Koneru, Shufan Li, and Aditya Grover. PhysiX: A foundation model for physics simulations, 2025. URL https://arxiv.org/abs/2506.17774.

NVIDIA, :, Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, Daniel Dworakowski, Jiaojiao Fan, Michele Fenzi, Francesco Ferroni, Sanja Fidler, Dieter Fox, Songwei Ge, Yunhao Ge, Jinwei Gu, Siddharth Gururani, Ethan He, Jiahui Huang, Jacob Huffman, Pooya Jannaty, Jingyi Jin, Seung Wook Kim, Gergely Klár, Grace Lam, Shiyi Lan, Laura Leal-Taixe, Anqi Li, Zhaoshuo Li, Chen-Hsuan Lin, Tsung-Yi Lin, Huan Ling, Ming-Yu Liu, Xian Liu, Alice Luo, Qianli Ma, Hanzi Mao, Kaichun Mo, Arsalan Mousavian, Seungjun Nah, Sriharsha Niverty, David Page, Despoina Paschalidou, Zeeshan Patel, Lindsey Pavao, Morteza Ramezanali, Fitsum Reda, Xiaowei Ren, Vasanth Rao Naik Sabavat, Ed Schmerling, Stella Shi, Bartosz Stefaniak, Shitao Tang, Lyne Tchapmi, Przemek Tredak, Wei-Cheng Tseng, Jibin Varghese, Hao Wang, Haoxiang Wang, Heng Wang, Ting-Chun Wang, Fangyin Wei, Xinyue Wei, Jay Zhangjie Wu, Jiashu Xu, Wei Yang, Lin Yen-Chen, Xiaohui Zeng, Yu Zeng, Jing Zhang, Qinsheng Zhang, Yuxuan Zhang, Qingqing Zhao, and Artur Zolkowski. Cosmos world foundation model platform for physical ai, 2025. URL https://arxiv.org/abs/2501.03575.

Ruben Ohana, Michael McCabe, Lucas Meyer, Rudy Morel, Fruzsina J. Agocs, Miguel Beneitez, Marsha Berger, Blakesley Burkhart, Keaton Burns, Stuart B. Dalziel, Drummond B. Fielding, Daniel Fortunato, Jared A. Goldberg, Keiya Hirashima, Yan-Fei Jiang, Rich R. Kerswell, Suryanarayana Maddu, Jonah Miller, Payel Mukhopadhyay, Stefan S. Nixon, Jeff Shen, Romain Watteaux, Bruno Régaldo-Saint Blancard, François Rozet, Liam H. Parker, Miles Cranmer, and Shirley Ho. The well: a large-scale collection of diverse physics simulations for machine learning, 2025. URL https://arxiv.org/abs/2412.00568.

Md Ashiqur Rahman, Robert Joseph George, Mogab Elleithy, Daniel Leibovici, Zongyi Li, Boris Bonev, Colin White, Julius Berner, Raymond A. Yeh, Jean Kossaifi, Kamyar Azizzadenesheli, and Anima Anandkumar. Pretraining codomain attention neural operators for solving multiphysics PDEs. In Advances in Neural Information Processing Systems, volume 37, pp. 104035–104064, 2024.

Louis Serrano, Thomas X Wang, Etienne Le Naour, Jean-Noël Vittaut, and Patrick Gallinari. AROMA: Preserving spatial structure for latent PDE modeling with local neural fields. In Advances in Neural Information Processing Systems,

2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/185a120a3f709187e68bd092e6098851-Abstract-Conference.html.

Michael Tschannen, Cian Eastwood, and Fabian Mentzer. Givt: Generative infinite-vocabulary transformers, 2024. URL https://arxiv.org/abs/2312.02116.

Nikhil Vyas, Depen Morwani, Rosie Zhao, Mujin Kwun, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham Kakade. Soap: Improving and stabilizing shampoo using adam, 2025. URL https://arxiv.org/abs/2409.11321.

Tian Wang and Chuang Wang. Latent neural operator pretraining for solving time-dependent PDEs. In Neural Information Processing, ICONIP 2024, volume 2282 of Communications in Computer and Information Science, pp. 163–178. Springer, 2025.

Wenxuan Wang, Fan Zhang, Yufeng Cui, Haiwen Diao, Zhuoyan Luo, Huchuan Lu, Jing Liu, and Xinlong Wang. End-to-end vision tokenizer tuning, 2025. URL https://arxiv.org/abs/2505.10562.

Tailin Wu, Takashi Maruyama, and Jure Leskovec. Learning to accelerate partial differential equations via latent global evolution. In Advances in Neural Information Processing Systems, volume 35, pp. 2240–2253, 2022.

Lijun Yu, Yong Cheng, Kihyuk Sohn, José Lezama, Han Zhang, Huiwen Chang, Alexander G. Hauptmann, Ming-Hsuan Yang, Yuan Hao, Irfan Essa, and Lu Jiang. Magvit: Masked generative video transformer, 2023. URL https://arxiv.org/abs/2212.05199.

Lijun Yu, José Lezama, Nitesh B. Gundavarapu, Luca Versari, Kihyuk Sohn, David Minnen, Yong Cheng, Vighnesh Birodkar, Agrim Gupta, Xiuye Gu, Alexander G. Hauptmann, Boqing Gong, Ming-Hsuan Yang, Irfan Essa, David A. Ross, and Lu Jiang. Language model beats diffusion – tokenizer is key to visual generation, 2024. URL https://arxiv.org/abs/2310.05737.

Anthony Zhou and Amir Barati Farimani. Masked autoencoders are PDE learners. Transactions on Machine Learning Research, 2024. URL https://arxiv.org/abs/2403.17728.

A DATA

We provide additional details on the datasets used in our experiments. Table 4 summarises their spatial dimensions, training-set sizes, and physical fields. The training sizes correspond to 10-frame examples with temporal stride 1, before the disjoint split used for tokeniser and emulator training.

Table 4: Dataset sizes, spatial dimensions, and available physical fields. Euler sample counts combine the open- and periodic-boundary-condition datasets.

		Euler	**RB**	**Shear Flow**	**Active Matter**	**TRL**
	Training samples	736k	267k	171k	12.6k	6.6k
	Validation samples	92k	33.4k	21.4k	1.7k	828
	Dimensions	512×512	512×128	256×512	256×256	128×384
Field Type	**Field Name**					
Scalar	Energy	✓				
	Density	✓				✓
	Pressure	✓	✓	✓		✓
	Buoyancy		✓			
	Tracer			✓		
	Concentration				✓	
Vector	Momentum	✓				
	Velocity		✓	✓	✓	✓
Tensor	Orientation				✓	
	Strain-Rate				✓	

Euler multiquadrants. Evolution of different gases starting with piecewise constant initial data in quadrants. The evolution can give rise to shocks, rarefaction waves, contact discontinuities, interaction with each other and domain walls. The evolved fields are density ρ, energy e, pressure p, and velocity components u and v, and the system follows Euler equations for a compressible gas. The fields are sampled on a uniform 512×512 Cartesian grid over 100 time steps. The dataset comprises 20 parameter configurations, varying in boundary condition type (open or periodic) and gas constant γ, with 500 trajectories per configuration.

Rayleigh-Bénard convection. Simulations of classical Rayleigh–Bénard convection, where a horizontal fluid layer heated from below develops convective rolls (Bénard cells) due to thermal buoyancy forces. The evolved fields include buoyancy b, pressure p, and velocity components (u, v), governed by the Boussinesq approximation of the Navier–Stokes equations. The data are sampled on a 512×128 uniform Cartesian grid over 200 time steps, across 35 parameter sets and 50 trajectories per parameter set.

Shear flow. Continuous deformation of adjacent fluid layers sliding past each other with different velocities. The evolved fields are tracer s, pressure p, and velocity components u and v, and the system is governed by incompressible Navier-Stokes equations and is sampled on a uniform 256×512 Cartesian grid over 200 time steps, across 28 flow configurations and 40 trajectories per configuration.

Active matter. Simulation of a continuum model for biological active matter, comprising rod-like particles immersed in a Stokes fluid. The evolved fields include a tracer-like concentration field c, fluid velocity (u, v), an orientation tensor (describing alignment of the active agents), and a strain-rate tensor. The data is sampled on a uniform 256×256 Cartesian grid over 81 time steps, across 45 parameter sets and 5 trajectories per parameter set.

Turbulent Radiative Layer: Simulation of shear-driven mixing between cold, dense gas and hot, dilute gas; an astrophysically motivated setup relevant to multiphase interfaces in the interstellar and circumgalactic medium. The evolved fields include density ρ, pressure p, and velocity components (u, v), governed by compressible hydrodynamics with radiative cooling. The dataset is sampled on a 384×128 Cartesian grid over 101 time steps. It spans 9 cooling-rate configurations and 10 trajectories per configuration.

B TOKENISER ARCHITECTURE AND FLEXIBLE COMPRESSION

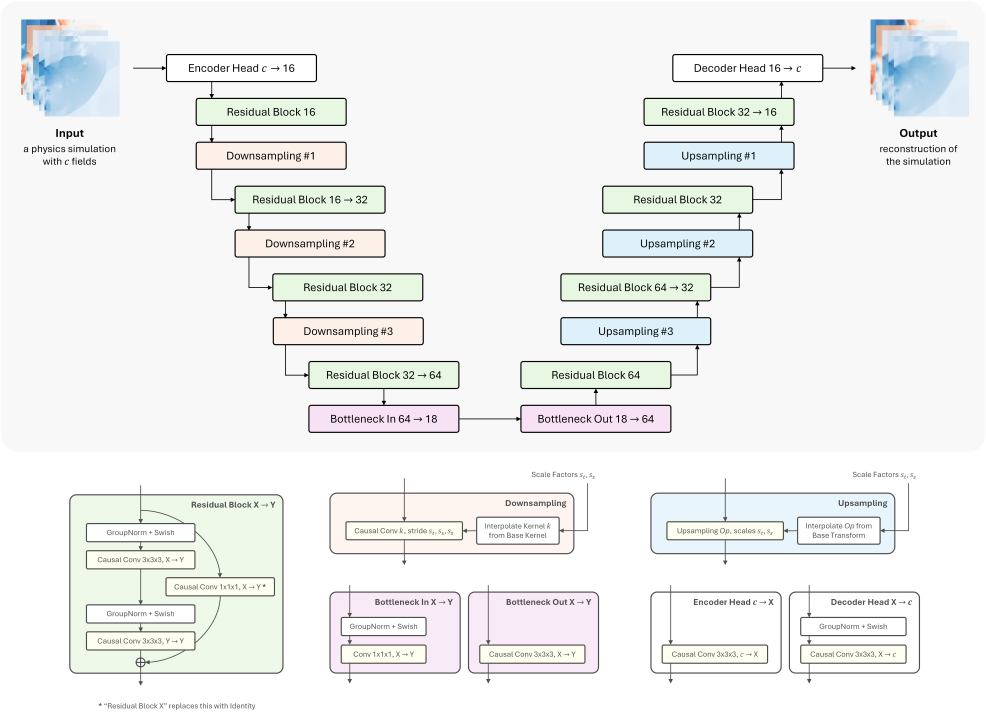

Figure 6: MAGVIT-Simple architecture.

MAGVIT-Simple autoencodes temporal sequences of 2D or 3D simulations $\mathbf{x} \in \mathbb{R}^{T \times S_1 \times \cdots \times S_k}$, where $k \in \{2, 3\}$. The temporal size T is set to "1 + a power of two", and the spatial sizes S_i are powers of two. Figure 6 illustrates the tokeniser architecture.

We employ causal convolutions, as introduced in MAGVIT-2, which enable joint image-video tokenisation and have been shown to outperform non-causal convolutions and attention-based blocks Yu et al. (2024): for 2D simulations, a 3D causal convolutional layer with kernel size (k_t, k_s, k_s) is applied to the input[3]. Spatial padding is applied symmetrically to both ends of each spatial dimension and is configured to preserve the input's spatial size. Temporal padding is applied only at the beginning (i.e., $k_t - 1$ leading frames), with no padding at the end. This preserves the temporal size of the input and ensures that each output timeframe depends only on the current and preceding frames. Padding values are set to zero.

The encoder and decoder heads use adaptive causal 3D convolutions that support a variable number of input physical fields. A single convolutional kernel is learned for the maximum number of fields; at runtime, rows and columns corresponding to active fields are selected from the weight matrix, and the output is rescaled by $\sqrt{C_{\text{total}}/C_{\text{active}}}$ to compensate for reduced fan-in. This follows the field embedding approach from MPP (McCabe et al., 2024).

For downsampling, we follow the idea of convolutional kernel modulation introduced in Mukhopadhyay et al. (2025) and extend it to causal convolutions. Let the set of scaling factors in a downsampling layer be denoted by $\{ (s_t, s_s)_i \}$, where each pair (s_t, s_s) consists of temporal and spatial strides, both chosen as powers of two. We use strided causal convolutions with adaptive kernel sizes to perform downsampling.

[3]Operating on 3D simulations would require using 4D causal convolutions.

Given a scaling pair (s_t, s_s), we define the spatial and temporal kernel sizes as

$$k_s = s_s \qquad k_t = \begin{cases} 1 + s_t, & \text{if } s_t > 1, \\ s_t, & \text{if } s_t = 1. \end{cases} \tag{5}$$

The stride is set to the corresponding scaling factor. This results in non-overlapping convolutions in space and a one-frame overlap in time. The kernel weights are obtained by interpolating from a base kernel, corresponding to the largest scaling factors $(s_{t,\text{max}}, s_{s,\text{max}})$ and thus having the largest size.

For flexible upsampling, we explore two strategies: causal transposed convolutions and the depth-to-space (pixel shuffle) operation. The former follows the same principle as flexible strided convolutions, with an additional convolution at the end to adjust the output dimensions. In the latter approach, the number of channels is first expanded by the product of the scaling factors, $s_t \times s_s^k$, via a causal convolution. The resulting tensor is then rearranged, redistributing channel values into the spatiotemporal dimensions.

To enable flexible upsampling with depth-to-space, we define the base convolutional kernel that upsamples by the largest scaling factor. For any smaller scaling factor that divides the base, we first perform base upsampling and then subsample the output to achieve the desired lower dimensionality. In some cases, this behaviour can be reproduced more efficiently by subsampling the base convolutional kernel directly, thereby avoiding redundant computation. One such case arises when the subsampling strategy selects every

$$\eta = \left(\frac{s_{t,\text{base}}}{s_t} \ , \ \frac{s_{s,\text{base}}}{s_s} \right)$$

pixel in the output - a strategy we adopt in our flexible depth-to-space layers. Finally, because this operation introduces additional time frames, we follow MAGVIT-2 (Yu et al., 2024) and discard the first $s_t - 1$ frames from the output.

C Detailed Training Recipe

This appendix provides the complete set of hyperparameters for both training stages. High-level training details are described in Subsection 2.6, and the tokeniser architecture is described in Appendix B.

Table 5 summarises the training hyperparameters for both stages.

Table 5: Training hyperparameters for tokeniser pretraining and emulator training.

Hyperparameter	Tokeniser Pretraining	Emulator Training
Optimiser		
Optimiser	SOAP	AdamW
[Peak] learning rate	5×10^{-4}	5×10^{-5}
Betas	$(0.95, 0.95)$	$(0.9, 0.999)$
Weight decay	0.01	10^{-4}
ϵ	10^{-8}	10^{-10}
LR scheduler	—	Inv. sqrt (see below)
Training		
Max steps	170,000	60,000 or 100,000
Gradient clipping	0.5	5.0
Distribution strategy	DDP	FSDP

Learning rate schedule. For tokeniser pretraining, no learning rate scheduler is used; the learning rate remains constant at 5×10^{-4} throughout training.

For emulator training, we use an inverse-square-root schedule with linear warmup and square-root cooldown, stepping once per epoch. Let e denote the current epoch (1-indexed), with total epochs $E = 100$, warmup duration $W = 10$, cooldown duration $C = 10$, warmup starting factor $\alpha = 0.1$, and decay offset $\kappa = 128$. The schedule has three phases:

Warmup ($e \leq W$): The learning rate increases linearly:

$$\mathrm{lr}(e) = \mathrm{lr}_0 \left(\alpha + (1 - \alpha) \frac{e}{W} \right). \tag{6}$$

Decay ($W < e \leq E - C$): The learning rate follows an inverse-square-root decay:

$$\mathrm{lr}(e) = \frac{\mathrm{lr}_0}{1 + \sqrt{s + \kappa} - \sqrt{\kappa}}, \tag{7}$$

where $s = e - W$ is the number of epochs since warmup ended.

Cooldown ($e > E - C$): The learning rate ramps down to zero with a square-root profile:

$$\mathrm{lr}(e) = \mathrm{lr}_{\text{end}} \left(1 - \sqrt{\frac{t}{C}} \right), \tag{8}$$

where $t = e - (E - C)$ counts epochs into cooldown and lr_{end} is the learning rate at the end of decay.

Data loading and recycling. Each GPU processes a disjoint shard of the training data. For tokeniser pretraining, each shard contains 2000 unique batches (of size 2). Once all unique batches have been consumed, the data loader reseeds its random number generator and begins iterating over the dataset again, allowing resampling in subsequent passes.

Convolution details. Encoder and decoder heads use causal 3D convolutions with kernel size $(k_t, k_s, k_s) = (3, 3, 3)$ and stride 1. Temporal padding is applied only at the past (leading $k_t - 1 = 2$ frames) so that each output time step depends only on current and previous inputs; spatial padding is symmetric (1 pixel per side). All residual blocks use causal 3D convolutions with kernel size 3 and stride 1. The encoder bottleneck uses a $1 \times 1 \times 1$ convolution to project to the latent dimension; the decoder bottleneck uses kernel size 3.

In downsampling layers, stride equals the chosen compression (s_t, s_s, s_s) (see Table 6). The convolution kernel size is tied to the stride: temporal size is $k_t = 1 + s_t$ when $s_t > 1$ and $k_t = s_t$ when $s_t = 1$; spatial size is $k_s = s_s$. Thus for the flexible layers the temporal kernel is 1 or 3 and the spatial kernel is 2 or 4. Spatial padding in these layers is "valid" (no padding).

Upsampling layers use causal $3 \times 3 \times 3$ convolutions. The convolution uses the same padding as elsewhere: causal in time (two leading frames, no trailing) and symmetric in space (one pixel on each side) so that spatial size is preserved before the depth-to-space rearrangement.

Compression factors. Table 6 summarises the temporal and spatial compression factors at each downsampling depth of the tokeniser (see Appendix B for the flexible compression mechanism). During training, the tokeniser uniformly samples one factor from each set at every forward pass; during validation, the scales are fixed to $(1, 2)$, $(2, 2)$, $(2, 2)$ for depths #1–#3 respectively.

Table 6: Temporal and spatial compression factors at each tokeniser depth. Values in braces indicate the set from which a factor is uniformly sampled during training.

Depth	Temporal Compression	Spatial Compression
#1	1	2
#2	{1, 2}	{2, 4}
#3	{1, 2}	{2, 4}

Processor configuration. As noted in Subsection 2.6, the processor has an embedding dimension of 1088 and the tokeniser bottleneck dimension is 18. A linear projection maps between the 18-dimensional tokeniser output and 1088-dimensional processor input, with a symmetric projection mapping back after the final block. Each of the six processor blocks consists of full spatial attention followed by axial temporal attention, both using 16 attention heads. Spatial attention employs a SwiGLU MLP with $4\times$ expansion ratio (hidden dimension 4352) and Rotary Position Embeddings (RoPE) with axial frequencies. Temporal attention uses causal masking and learned relative position biases. Block normalisation is RMSGroupNorm and stochastic depth (drop path) is applied with a linearly increasing rate from 0 to 0.05 across blocks.

Input normalisation. In both training stages, each input sample is normalised before being fed to the tokeniser. We compute the root mean square (RMS) of each physical field over all time steps and spatial locations, and divide by this value (clamped below at $\epsilon = 10^{-7}$ for numerical stability). The inverse transform is applied to the output. This samplewise normalisation operates independently per sample and per field.

D ADDITIONAL NEXT-FRAME METRIC

Spatial metrics do not reveal how well a model captures structure at different scales. We therefore examine spectral errors by analysing the power spectrum of the residual field. For fields $\mathbf{x}$ and $\hat{\mathbf{x}}$, we compute the power spectrum of the residual $\mathbf{x} - \hat{\mathbf{x}}$ by binning Fourier modes $\mathbf{k}'$ according to their wavenumber magnitude $|\mathbf{k}'|$ and averaging the squared Fourier coefficients within each bin B_k:

$$P_{\mathbf{x}-\hat{\mathbf{x}}}(k) = \frac{1}{|B_k|} \sum_{\mathbf{k}' \in B_k} \left| \mathcal{F}\{\mathbf{x} - \hat{\mathbf{x}}\}(\mathbf{k}') \right|^2 . \tag{9}$$

This power spectrum quantifies the contribution of each wavenumber band to the total pixel-space MSE via Plancherel's theorem: the total MSE equals $\frac{1}{N^2} \sum_k |B_k| \, P_{\mathbf{x}-\hat{\mathbf{x}}}(k)$, where N^2 is the number of spatial pixels. To obtain a scale-invariant quality metric, we normalise by the true field's power spectrum at each scale:

$$\text{NEPS}(k) = \frac{P_{\mathbf{x}-\hat{\mathbf{x}}}(k)}{P_{\mathbf{x}}(k)}, \tag{10}$$

where $P_{\mathbf{x}}(k)$ is the power spectrum of the true field, computed analogously to $P_{\mathbf{x}-\hat{\mathbf{x}}}(k)$. This normalised error power spectrum (NEPS) measures the ratio of error power to signal power at each scale. Since each frequency band uses its own normalisation, NEPS values are not additive across scales but provide a relative quality metric independent of the natural energy distribution. A value of $\text{NEPS}(k) = 0.1$ indicates the error power is 10% of the signal power at scale k, while $\text{NEPS}(k) = 1.0$ means the error power equals the signal power at that scale. We partition wavenumbers into three frequency ranges (low, mid, high) and report NEPS for each range. We compute NEPS per timeframe and per field, obtaining aggregate metrics by averaging across timeframes and/or fields.

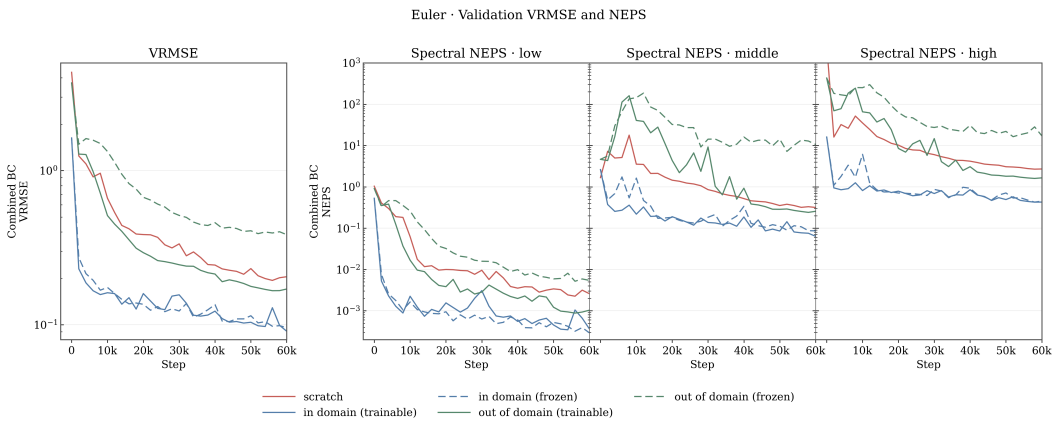

Figure 7: Validation next-step VRMSE and NEPS for Euler.

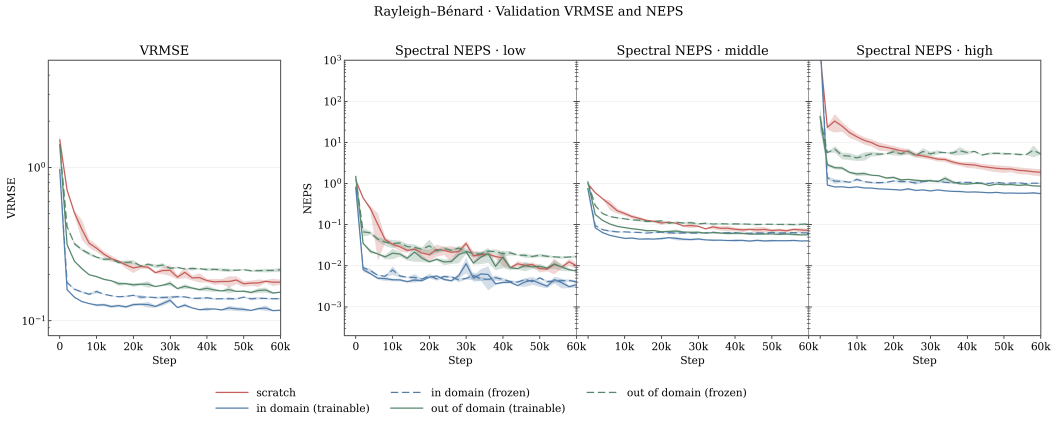

Figure 8: Validation next-step VRMSE and NEPS for RB.

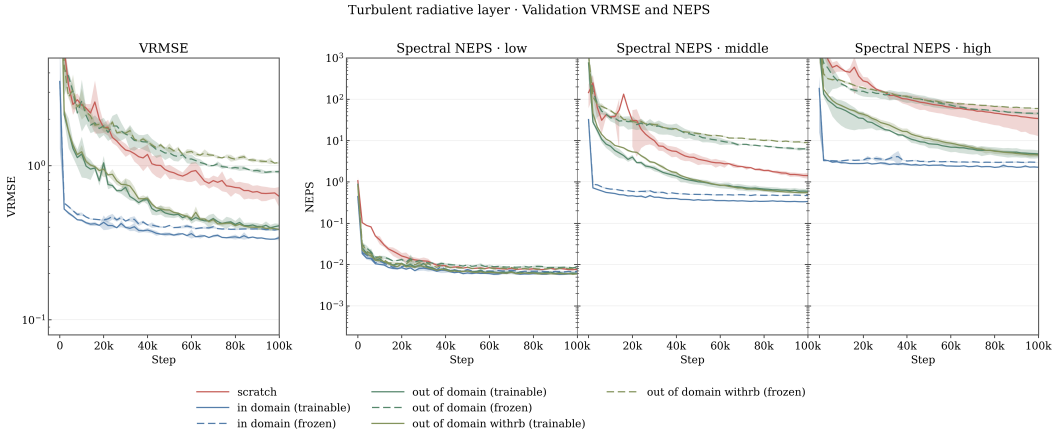

Figure 9: Validation next-step VRMSE and NEPS for TRL.

E AI ASSISTANCE

Large language models were used to assist with editing, rephrasing, and improving the clarity and organisation of the manuscript. All scientific content, experimental design, analysis, interpretation of results, and final editorial decisions were made by the authors.