A Discrete–Continuous Curriculum Learning (DCCL) Framework for Stable Long-Horizon PDE Surrogates

Lalit Ghule
Ansys Inc.
lghule@ansys.com

Akanksh Shetty
Innovative Numerics LLC.
akanksh.shetty@innovativenumerics.com

Morgane Bourgeois
Ansys Inc.
morgane.bourgeois@ansys.com

Abstract

Deep surrogates for transient dynamics often diverge over long horizons due to compounding error. Curriculum learning mitigates exposure bias by gradually replacing teacher forcing with self-rollouts, but discrete switching can induce optimization instabilities. We propose a *discrete–continuous curriculum learning* (DCCL) framework that blends model predictions with ground truth via a convex combination. On a 2D vorticity dataset, DCCL consistently reduces long-horizon rollout error compared to self-rollout and discrete curriculum baselines for both UNet and Fourier Neural Operator (FNO) backbones, with pronounced gains in extrapolation.

1 Introduction

Neural surrogates promise major speedups for transient Partial Differential Equations (PDE) simulation, but maintaining long-horizon accuracy is hard as autoregressive errors compound beyond the training window. Despite advances in physics-informed training (1) and operator learning (e.g., FNO) (2; 3), stability over extended rollouts remains the bottleneck (4; 5; 6; 7).

In sequence modeling, these systems face exposure bias - training conditions on ground truth and inference on model outputs. Teacher forcing stabilizes optimization yet exacerbates this mismatch (8), while scheduled sampling / curriculum learning mitigates it by intermittently feeding back predictions (9; 10; 11; 12; 13; 14; 15). Prior work has ported this Natural Language Processing (NLP) idea to fluid dynamics, showing that discrete curricula improve long-horizon rollouts for UNet and FNO on 2D vorticity (2; 16; 17; 4; 5). Notably, these techniques originate in NLP where outputs are discrete tokens and continuity of the input distribution across steps is not itself a modeling requirement. In contrast, PDE state trajectories are continuous in space and time, so abrupt probability "switches" can induce sharp distribution shifts during training.

This paper introduces a discrete–continuous curriculum learning (DCCL) that replaces hard toggles with a convex blend of predictions and ground truth governed by an annealed mixing coefficient. Contributions: (i) a simple curriculum that reduces exposure bias without abrupt switching; (ii) a controlled comparison of UNet and FNO under identical training budgets and schedules; and (iii) consistent reductions in rollout error, with the largest gains beyond the training horizon.

Machine Learning and the Physical Sciences Workshop, NeurIPS 2025.

2 Method

2.1 Setup

We study supervised sequence prediction for transient dynamics. Let $\mathbf{x}_t$ denote the state at time t. A predictor f_θ maps a fixed history of k frames to the next state. At each training step we warm-start with the ground-truth history $\mathbf{x}_{t-k+1:t}$, roll out the model for a horizon H_r, and optimize sequence-wise error over the rollout.

2.2 Training strategies

1) Self-Rollout (baseline). At rollout step τ, predict $\hat{\mathbf{x}}_{t+\tau} = f_\theta(\textit{inputs})$ and feed that prediction as the next input:

$$\texttt{next\_input} \leftarrow \hat{\mathbf{x}}_{t+\tau}.$$

This mirrors inference but is vulnerable to compounding error over long horizons.

2) Discrete Curriculum Learning (DCL; scheduled sampling). Let s denote the global training step (update) and let $e(s) \in [0, 1]$ be the curriculum intensity. At rollout step τ, sample $u \sim \mathcal{U}(0, 1)$ and set

$$\texttt{next\_input} \leftarrow \begin{cases} \mathbf{x}_{t+\tau}, & u > e(s), \\ \texttt{detach}(\hat{\mathbf{x}}_{t+\tau}), & u \le e(s) \quad \text{(with stop-gradient).} \end{cases}$$

We use a linear schedule $e(s)$ that increases from 0 to 1 over training. Prior work shows such a curriculum stabilizes training and improves long-horizon accuracy for UNet and FNO on 2-D vorticity.

3) Discrete–Continuous Curriculum Learning (DCCL; ours) DCL's binary switch can make the input distribution change abruptly. We retain the same stochastic gate but, on the $u \le e(s)$ condition, replace the discrete choice of prediction as input with a convex blend: $\texttt{next\_input} = e(s) \cdot \text{prediction} + \big(1 - e(s)\big) \cdot \text{ground truth}$, with $e(s)$ rising linearly from 0 to 1. Early in training ($e \approx 0$) inputs are almost entirely ground truth; late in training ($e \to 1$) they are predominantly model-driven, but the transition is smooth. (DCCL gates with prob. $e(s)$ and blends with weight $e(s)$; expected prediction weight $= e(s)^2$.)

At each step:

1. Draw $u \sim \mathcal{U}(0, 1)$.
2. If $u > e(s)$: use the ground truth, $\mathbf{x}_{t+\tau}$.
3. Else: use a convex combination, $e(s)\, \hat{\mathbf{x}}_{t+\tau} + \big(1 - e(s)\big)\, \mathbf{x}_{t+\tau}$.

Algorithm 1: TrainRollout (per epoch)

Input: DataLoader $\mathcal{B}$, model f_θ, loss ℓ (MSE), optimizer, curriculum value e
foreach *batch* $(\textit{inputs}, \textit{targets}) \in \mathcal{B}$ **do**
  move *inputs*, *targets* to device;
  $L \leftarrow 0$; $T \leftarrow$ number of rollout steps;
  for *step* $= 1$ **to** T **do**
    predicted $\leftarrow f_\theta(\textit{inputs})$;
    $L \leftarrow L + \ell(\textit{predicted}, \textit{targets}[:, \textit{step}])$;
    inputs $\leftarrow$ `NextInput`(*inputs*, *predicted*, *targets*$[:, \textit{step}]$, e);
  $L \leftarrow L/T$;
  `optimizer.zero_grad();` backprop L; `optimizer.step();`

Algorithm 2: NextInput: DCCL

Input: *inputs*, *predicted*, *target*, curriculum value e
sample $u \sim \mathcal{U}(0, 1)$;
if $u \leq e$ **then**
    weighted $\leftarrow e \cdot$ `detach(predicted)` $+ (1 - e) \cdot$ *target*;
    return `concat( inputs[:, 1:], weighted );`
else
    return `concat( inputs[:, 1:], target );`

2.3 Implementation details

We keep data processing, optimizer, rollout horizon per update, and evaluation protocol identical across strategies; only the NextInput function differs.

Data. We consider 2D vorticity evolution on a unit torus with spatial resolution 64×64 and viscosity $\nu = 10^{-3}$, producing trajectories of $T = 70$ discrete time steps. The first 50 steps are used solely to construct training pairs, and the final 20 steps (51–70) are reserved for extrapolation performance evaluation. From the 50-step prefix, we generate supervised examples in a rolling-window manner: each input is a history of $k = 10$ consecutive fields, and the model is trained to predict the subsequent 20 steps autoregressively.

Backbones. We employ UNet and Fourier Neural Operator (FNO) backbones with 3.43M parameters and 1.18M parameters respectively.

Training. The predictor consumes a history of length $k = 10$ and is optimized to generate the next 20 time steps, one step at a time; during training, we unroll the predictor for 20 steps per update. We use Adam for 500 epochs with a base learning rate of 5×10^{-4} and a StepLR scheduler (step=50, γ=0.5). The curriculum variables follow a linear schedule that increases e from 0 to 1 over the course of training. All experiments were conducted on a single NVIDIA QUADRO RTX 6000, with each run taking approximately 16 hours to complete.

3 Results

To get an idea of the baseline performance, we trained a UNet with *self-rollout*. As expected, its error curve grows the fastest after the training horizon. Figure 1 shows rollout MSE versus time step with a dashed line at step 50 marking the onset of extrapolation; DCCL exhibits the slowest error growth in late steps for *both* backbones. Figure 2 shows UNet rollouts in the extrapolation region and visually reflects the same trend: DCCL better preserves the phase and amplitude of coherent structures, reducing late-time smearing and drift.

4 Discussion and limitations

Why DCCL helps. Scheduled sampling (DCL) reduces exposure bias but introduces abrupt distribution switches when it replaces ground truth with predictions during training. In continuous-state PDE surrogates, such discontinuities can hinder optimization and amplify long-horizon drift (18). *DCCL keeps the same stochastic gate yet blends prediction and ground truth when the gate is active*

Table 1: Rollout error (MSE). "Full" denotes the entire 70-step rollout; "Extrap." denotes steps > 50.

	Full		Extrap.	
Backbone	DCL	**DCCL (ours)**	DCL	**DCCL (ours)**
UNet	2.06e^{-3}	$\mathbf{1.79\text{e}^{-3}}$	4.77e^{-3}	$\mathbf{3.83\text{e}^{-3}}$
FNO	1.26e^{-3}	$\mathbf{6.08\text{e}^{-4}}$	2.23e^{-3}	$\mathbf{1.22\text{e}^{-3}}$

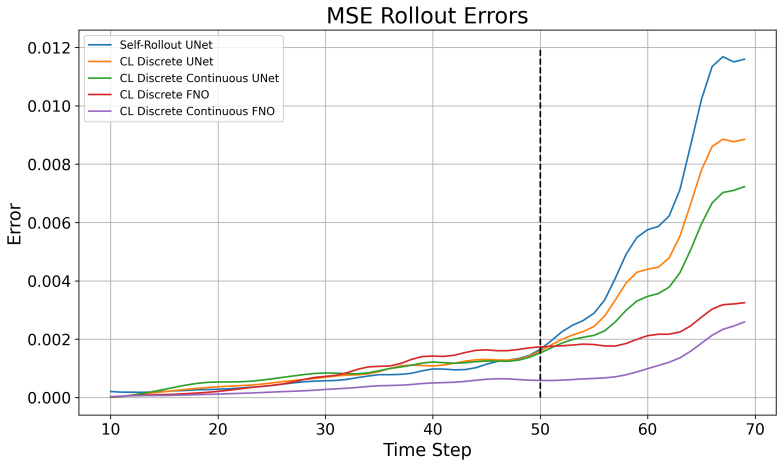

Figure 1: Rollout MSE versus time; dashed line marks extrapolation onset (step 50).

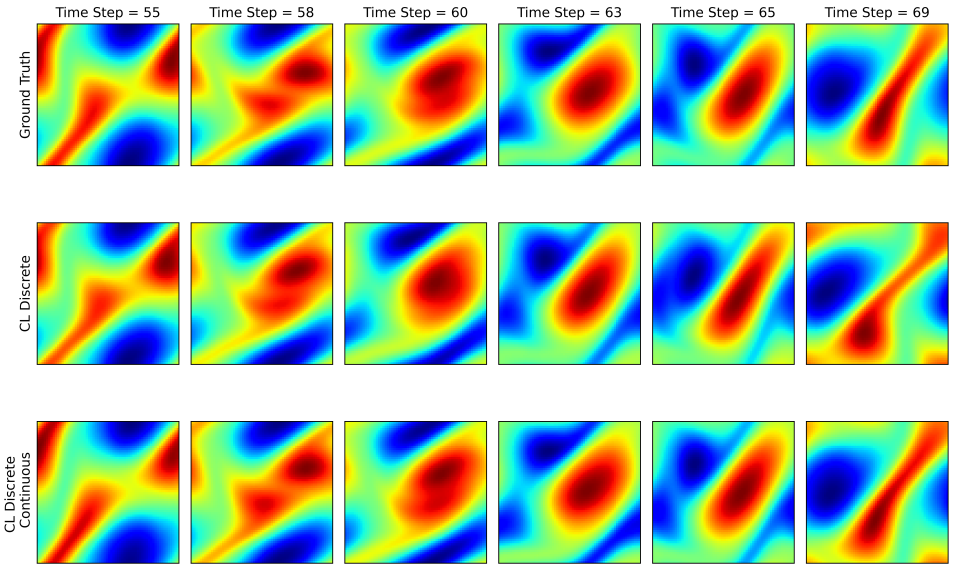

Figure 2: Qualitative comparison (UNet). Top: ground truth; middle: DCL; bottom: DCCL (ours).

(i.e., when $u \leq e(s)$): This simple convex combination smooths the training-time state distribution toward the test-time distribution, yielding slower error accumulation (Fig. 1) and lower extrapolation MSE (Table 1). In practice, we detach the prediction inside the blend (Alg. 2) to prevent degenerate gradients.

Benefit to the transient-dynamics community. DCCL is a *training policy* that: (i) is architecture-agnostic, (ii) requires *no* changes to data, loss, or rollout budgets, and (iii) adds negligible compute. Because it directly targets exposure bias while respecting the continuity of physical states, it is immediately useful for researchers building surrogates across *all types of transient dynamics*, including but not limited to fluids, reaction–diffusion, plasmas and weather.

Limitations and scope. The convex blend can slightly perturb physical invariants during training (e.g., energy or circulation) (19), even though inference uses pure model predictions. Mitigations include invariant-preserving projections, divergence/energy penalties, or pairing DCCL with spectral/energy regularization. Our experiments focus on 2-D vorticity at fixed resolution and time step;

extending to multi-physics, stiffer regimes, and adaptive time stepping is future work. We used a linear mixing schedule; schedule design and per-step adaptivity remain open knobs (20).

5 Future work

Future work will include broadening evaluation beyond a single 2D vorticity dataset to multiple PDE families such as incompressible and compressible flows, reaction–diffusion, and shallow water systems. It is worth investigating the effect of DCCL on longer horizons, with measurements of error growth and spectral drift well beyond the training window. Resolution sensitivity deserves systematic study through grid sweeps, cross-resolution transfer, and tests on unstructured and adaptive meshes. Extending to 3D configurations will help assess stability in more complex regimes. On the modeling side, architectures beyond UNet and FNO, including graph and mesh based surrogates such as MeshGraphNet offer a way to probe representation dependence. Finally, exploring non-linear schedules for the mixing coefficient $e(s)$, together with analyses of how the convex blend influences exposure bias and late time drift, may yield additional gains in rollout stability.

6 Conclusion

We presented *Discrete–Continuous Curriculum Learning (DCCL)*, which replaces the binary switch in scheduled sampling with a convex blend between prediction and ground truth (applied when $u \leq e(s)$). On a standard 2-D vorticity benchmark and two backbones (UNet, FNO), DCCL consistently improves long-horizon accuracy: it lowers both full-trajectory and extrapolation MSE relative to a discrete curriculum, with the largest margins for FNO (Table 1) and visibly slower error growth over time (Fig. 1). As a drop-in, budget-neutral change, DCCL is straightforward to integrate into existing codebases and should benefit the broader research community developing surrogates for transient dynamics across domains.

References

[1] M. Raissi, P. Perdikaris, G. E. Karniadakis, "Physics-Informed Neural Networks," *J. Comput. Phys.*, 2019.

[2] Z. Li et al., "Fourier Neural Operator for Parametric PDEs," arXiv, 2020.

[3] K. Azizzadenesheli et al., "Neural Operators for Accelerating Scientific Simulations," arXiv, 2023.

[4] P. Lippe, B. S. Veeling, P. Perdikaris, R. E. Turner, J. Brandstetter, "PDE-Refiner: Achieving Accurate Long Rollouts with Neural PDE Solvers," *NeurIPS*, 2023.

[5] M. McCabe, P. Harrington, S. Subramanian, J. Brown, "Towards Stability of Autoregressive Neural Operators," *Transactions on Machine Learning Research (TMLR)*, 2024.

[6] Z. Ye, C.-S. Zhang, W. Wang, "Recurrent Neural Operators: Stable Long-Term PDE Prediction," arXiv, 2025.

[7] Cai et al., "Towards long rollout of neural operators with local attention and flow-matching-inspired correction," *NeurIPS Workshop on Machine Learning and the Physical Sciences (ML4PS)*, 2024.

[8] R. J. Williams, D. Zipser, "A Learning Algorithm for Continually Running Fully Recurrent Neural Networks," 1989.

[9] S. Bengio, O. Vinyals, N. Jaitly, N. Shazeer, "Scheduled Sampling for Sequence Prediction with RNNs," arXiv, 2015.

[10] A. Venkatraman, M. Hebert, J. A. Bagnell, "Improving Multi-Step Prediction of Learned Time-Series Models," arXiv, 2015.

[11] A. M. Lamb, A. Goyal, Y. Zhang, et al., "Professor Forcing: A New Algorithm for Training Recurrent Networks," *NeurIPS*, 2016.

[12] F. Huszár, "How (not) to train your generative model: Scheduled sampling, likelihood, adversary?" arXiv, 2015.

[13] Z. Cen, Y. Liu, S. Zeng, et al., "Bridging the Training-Inference Gap in LLMs by Leveraging Self-Generated Tokens," *TMLR*, 2025.

[14] X. Huang, Z. Li, G. He, M. Zhou, E. Shechtman, "Self Forcing: Bridging the Train-Test Gap in Autoregressive Video Diffusion," arXiv, 2025.

[15] S. Ross, G. J. Gordon, J. A. Bagnell, "A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning (DAgger)," *AISTATS*, 2011.

[16] O. Ronneberger, P. Fischer, T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," arXiv, 2015.

[17] L. Ghule, R. Ranade, J. Pathak, "NLP-Inspired Training Mechanics for Modeling Transient Dynamics," arXiv, 2022.

[18] Deo and Jaiman, "Harnessing Loss Decomposition for Long-Horizon Wave Predictions via Deep Neural Networks," *NeurIPS Workshop on Machine Learning and the Physical Sciences (ML4PS)*, 2024.

[19] Zhang et al., "OceanCastNet: A deep learning ocean wave model with energy conservation," arXiv, 2024.

[20] Y. Guan, J. Liu, Z. Li, "Uncovering the Spectral Bias of Fourier Neural Operator," arXiv:2212.02844, 2022.