LEARNING GENERAL-PURPOSE CNN-BASED SIMULATORS FOR ASTROPHYSICAL TURBULENCE

Alvaro Sanchez-Gonzalez,[*,1] **Kimberly Stachenfeld**,[*,1] **Drummond B. Fielding**,[2]
Dmitrii Kochkov,[3] **Miles Cranmer**,[4] **Tobias Pfaff**,[1] **Jonathan Godwin**,[1] **Can Cui**,[2]
Shirley Ho,[2] **Peter Battaglia**[1]

[1]DeepMind, London, UK [2]Center for Computational Astrophysics, Flatiron Institute, New York, NY
[3]Google Research, Cambridge, MA [4]Princeton University, Princeton, NJ
[*]Equal contribution. `alvarosg@google.com`, `stachenfeld@google.com`

ABSTRACT

Given the rise of machine learning (ML) for simulation, an important question is: to what extent can learned models supplement or replace traditional simulators? Here we develop a fully learned simulator model based on domain-general Convolutional Neural Network (CNN) methods, and study its performance on a range of turbulence problems in astrophysics. We compare the learned model to specialized PDE solvers in terms of spatial and temporal resolution, numerical stability, and generalization performance. We find that the learned models outperform coarsened solvers on certain metrics, particularly in their ability to preserve high-frequency information at low resolution, and describe ways to improve generalization beyond the training distribution. To our knowledge, our model is the first to be trained on `Athena++` (a state-of-the-art simulator widely used in computational fluid dynamics and magneto-hydrodynamics), and more generally, the first fully-learned astrophysical turbulence simulator.

1 INTRODUCTION

Many scientific disciplines depend on simulating turbulent fluid dynamics, from engineering applications like aeronautics [18] and medicine [19], to scientific domains ranging from small-scale molecular dynamics [23] to large-scale simulation of galaxies for astrophysics [4]. Powerful specialized PDE solvers have a rigorous foundation in numerical analysis and maintain accuracy over long integrations. However, because turbulent dynamics span several length scales, these solvers require high-resolution grids and therefore substantial computational resources. Otherwise, simulated dynamics quickly depart from the underlying equations, often leading to the loss of high frequency details.

Learned simulators are a promising avenue for maintaining accuracy at low resolutions because they can adapt to capture the effective dynamics of larger scales on coarse grids. Learned simulators vary in the relative extents to which they incorporate components from classical solvers, like closures and subgrid discretization [5, 7, 16, 25, 10], versus more pure ML methods which are gaining traction in scientific, engineering, and graphics domains [9, 14, 21, 26, 13, 12, 17]. Advantages of the pure ML route are that the same model can be used without specialized knowledge of the domain and that they do not require interfacing with solvers. However, fully learned simulators often work well only when conditions are similar to the training distribution, which can limit their stability over time and their generalization capabilities.

Here we use a CNN model that is not specifically specialized for turbulence simulation to study how training choices lead to trade-offs in stability, generalization, and efficiency. We find that the model can fit a variety of simulated turbulence datasets, and that training options (corrupting inputs with noise, temporal downsampling) affect stability, generalization outside the training distribution, preservation of scientifically relevant conserved quantities, and high-frequency information. We compare the learned simulator to a PDE solver run at coarser grids, and find that the learned simulator outperforms the solver at the same resolution, and more accurately captures high-frequency structure even at times outside what the model saw during training.

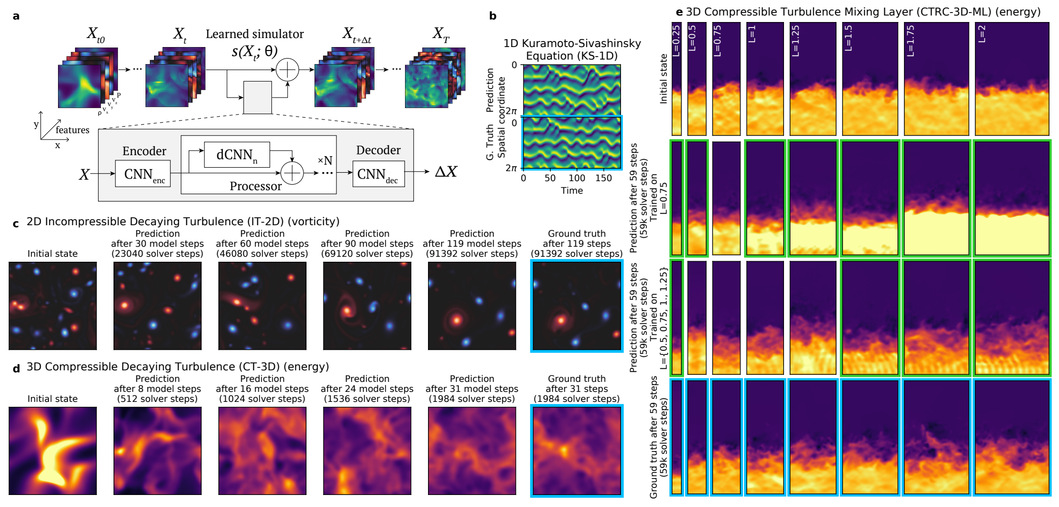

Figure 1: (a) Learned simulator schematic. (b-e) Predicted trajectories using the learned simulator. Blue frames indicate ground truth, and green frames indicate out-of-distribution generalization. (b) KS-1D: The learned simulator produces a plausible trajectory that follows the ground truth closely for the first 150 steps (t < 75). (c) IT-2D: The learned model prediction remains accurate after 119 model steps (91,392 solver steps). (d) CT-3D: The learned model prediction remains accurate after 31 model steps (1984 solver steps). (e) CTRD-3D-ML. Predicted state after 59 model steps (59,000 solver steps) on multiple box sizes, for models trained on a single box size ($L = 0.75$) (row 2) and a range of box sizes ($L \in \{0.5, 0.75, 1., 1.25\}$) (row 3). Videos available at tinyurl.com/dl4sturb.

2 Experimental Turbulent Domains

We use four turbulence PDEs (Fig. 1b-e) (see Appendix):
1D Kuramoto-Sivashinsky Equation (KS-1D): A PDE that generates unstable, chaotic dynamics in 1D, solved using Fourier spectral method [11, 22].
2D Incompressible Decaying Turbulence (IT-2D): Fluid flow under Navier-Stokes in which small-scale eddies decay into large-scale structures due to the inverse energy cascade, relevant to planetary atmospheric flow, solved using Direct Numerical Simulation [10].
3D Compressible Decaying Turbulence (CT-3D): Transonic turbulent flow under Navier-Stokes assuming adiabatic equation of state, common in astrophysics [15], solved with `Athena++` [24].
3D Compressible Turbulence with Radiative Cooling Mixing Layer (CTRC-3D-ML): Turbulent mixing from the Kelvin-Helmholtz instability, caused by velocity differences across the interface between fluids of different densities, solved with `Athena++`. Mixing involves strong cooling, leading to net flow from the low-density phase into the mixing layer, and is relevant to galaxy formation [6].

3 Model

Learned Simulation Framework on Grids Our objective is to learn a simulation model s that maps X_t to $X_{t+\Delta t}$. We let $X_t \in \mathcal{X}$ be a n+1 dimensional tensor specifying the state variables for each point on a grid with n spatial dimensions and one feature dimension at time t. Applying physical dynamics over K time steps yields a trajectory of states $(X_{t_0}, ..., X_{t_K})$. A simulation $s : \mathcal{X} \rightarrow \mathcal{X}$ maps a state X_t to a future state $\tilde{X}_{t+\Delta t} = s(X_t)$. We denote a simulated "rollout" trajectory as $(X_{t_0}, \tilde{X}_{t_1}, ..., \tilde{X}_{t_K})$, where X_{t_0} represents initial conditions given as input, and simulated future states are computed iteratively. Our learnable simulator (Fig. 1a), uses a neural network NN (with weights θ) to model one-step of the simulation as $\tilde{X}_{t+\Delta t} = s(X_t; \theta) = X_t + \text{NN}(X_t; \theta)$.

Neural Network Architecture The model is comprised of a single-CNN encoder, a processor network, and a single-CNN decoder. The processor consists of $N = 4$ dilated CNN blocks (dCNN_n) with unshared weights connected in series and residual connections [27, 8]. Each block consists of 7 dilated CNN layers with dilation rates of (1, 2, 4, 8, 4, 2, 1). Residual connections help

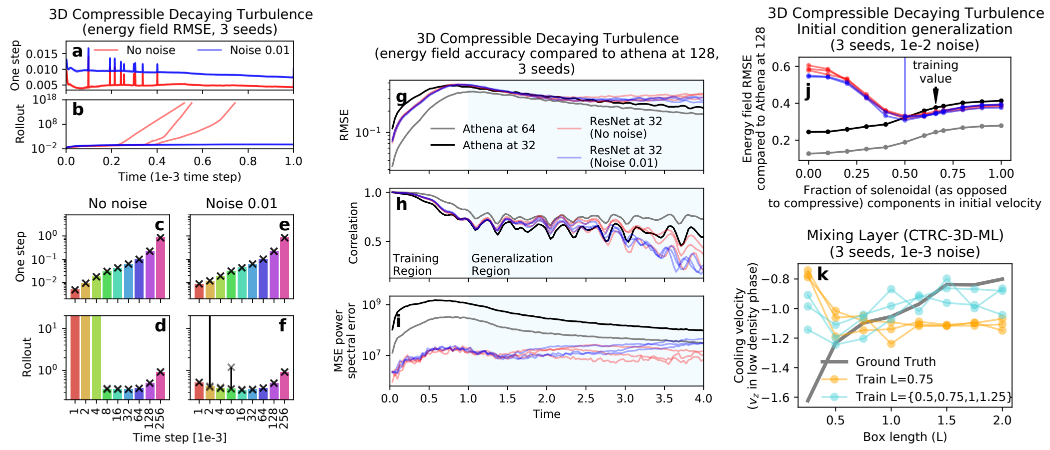

Figure 2: (a-b) One-step and rollout error for a model trained with $\sigma = 0.01$ and without noise $\sigma = 0$. (c-f) Temporal coarsening impact on one-step and rollout error. Little downsampling <8 yields lower one-step errors (c), however requires taking more model steps during a rollout which can cause instability (d). These instabilities can be mitigated by training with noise (f). Higher coarsening >32 increases one-step and rollout error. (g-i) Energy MSE error, correlation, and power spectrum error as a function of time, for rollouts of learned models trained on high-resolution Athena data (128^3) downsampled to 32^3, and for Athena ran at lower spatial resolutions (64^3, 32^3). (j) Generalization to different initial conditions. The model generalizes well to initial velocities with higher solenoidal, but not to higher compressive components. (k) Cooling velocity generalization as function of the box size in the mixing layer. Training on multiple sizes improves performance.

avoid vanishing gradients, and dilations provide longer-range communication while preserving local structure. We use periodic padding for spatial axes with periodic boundary conditions, and fixed padding for spatial axes with fixed boundary conditions.

Training We use an L2 loss $\ell(X_t, X_{t+\Delta t}) = \text{MSE}(\text{NN}(X_t;\theta), \Delta X)$ to optimize parameters θ. Input X_t and target $\Delta X = X_{t+\Delta t} - X_t$ features are normalized to be zero-mean and unit variance. Optionally, we trained with Gaussian random noise with fixed variance σ added to the input X_t of the loss function, as this has been shown to improve stability of rollouts and prevent error accumulation by training the model to correct for small errors [20, 21, 17]. See Appendix for more details.

4 Results

We evaluate our learned simulators in terms of stability, performance, efficiency, and generalization. Results are evaluated on held-out test trajectories sampled from the same distribution used for training (except for generalization sections). All units are dimensionless.

Domain generality Unlike numerical solvers, which are PDE specific, learned simulator models can be designed to be reusable. Here, the same general-purpose architecture and loss, trained on each of the domains, learns to capture a range of qualitatively diverse turbulent dynamics (Fig. 1b-e, videos: tinyurl.com/dl4sturb), using as few as 27 training trajectories (CT-3D).

Stability While scientific simulators are typically designed to be stable over time, a common failure mode in learned models is that small errors can accumulate over rollouts and lead to a domain shift. We speculate this is because, as the model is fed its most recent prediction back in as input for predicting future steps, its distribution of input states begins to deviate from that experienced at training, where it fails to generalize and instead makes arbitrarily poor predictions. We found that adding Gaussian noise $\sigma = 0.01$ to the inputs X_t during training led to less accurate one-step predictions, but more stable trajectories (Fig. 2a,b), presumably because the training distribution has broader support and the model is optimized to map deviant inputs back to the training distribution.

Temporal coarsening An advantage of learned simulators is that they can exploit a larger step size than the numerical solver, as they can learn to compensate for errors in finite-difference approximations of the time derivatives. As expected, while the one-step error is at its lowest when using smaller time steps, the rollout error has an optimal time step at around 32 (Fig. 2c-f). This demonstrates the trade-off between large time steps (> 32), as predictions become more challenging, or small time steps (< 16), which require more simulator steps, often yielding unstable models (Fig. 2d), although these may still be stabilized to some extent with training noise (Fig. 2f).

Spatial coarsening Numerical solvers are known to lose high frequency information about dynamics (e.g. due to numerical viscosity) when applied to grids that are too coarse, which learned models may be sufficiently expressive to capture. We compare the energy predictions of our model run at a resolution of 32^3 on ground truth solver data downsampled from 128^3 to those produced by the solver at low resolutions of 32^3 and 64^3. All grids are downsampled to 32^3 for the comparison. The learned simulators outperform both the same- and higher-resolution `Athena++` rollouts in terms of the Power Spectral Density, as the `Athena++` simulators lose high frequency components that the learned simulators preserve (see videos, link in Fig. 1 caption). We further find that, in terms of MSE and correlation on the training region ($t < 1$ in Fig. 2g-i) the learned model running at 32^3 slightly outperforms `Athena++` at the same resolution of 32^3 but not `Athena++` at 64^3.

Running time Learned simulators can accelerate simulations not only by coarsening in time and space, but also by running off-the-shelf on specialized GPU hardware, unlike e.g. `Athena++` which is CPU specific. By downsampling in time ($\Delta t = 0.5 \rightarrow 32$) and space ($128^3 \rightarrow 32^3$) the CT-3D learned model running on a single GPU speeds up the wall-time simulation time by a factor of 1000 compared to `Athena++` running on an 8-core CPU (More details in Appendix).

Constraint satisfaction and preserved quantities Traditional solvers often implement constraints to preserve conserved quantities. Learned simulators, however, will not necessarily learn such constraints. We observe that training with noise helps preserves local constraints (e.g. keeping the divergence of the incompressible fluid near 0), but does not always prevent slow drift in global quantities (e.g. mean value of the signal in the KS equation, or total energy/mass/momentum in 3D turbulence), possibly due to the local nature of the convolutional model (Supp. Fig. A.1).

Generalization to longer rollouts We evaluate the stability of the model for rollouts longer than those in the training data. In the case of KS-1D, which has stationary dynamics, the model remains stable for longer trajectories (Supp. Fig. A.5). On the other hand, in CT-3D, for which the dynamics are not stationary under decaying turbulence, even 32^3 `Athena++` overtakes the learned models in terms of both MSE and and correlation ($t > 1$ in Fig. 2g-i).

Generalization to different initial conditions We varied the ratio of solenoidal to compressive components in the initial velocity field in CT-3D (Fig. 2j) and found the model generalizes well to more solenoidal but not to more compressive initial conditions, possibly due to faster turbulence decay under compressive conditions.

Generalization to larger boxes We tested the generalization capability of the CTRD-3D-ML model to boxes with different size L (Fig. 1e) and measured the predicted cooling velocity (small laminar flow in the low density phase perpendicular to the interface, see Appendix for more details). The cooling velocity is of scientific relevance and known to depend on the box size. We found that unless the model is trained on a range of sizes, the predicted cooling velocity does not follow the right trend, and even when trained on a range of sizes, generalization outside the training range is not reliable across seeds (Fig. 2k). We speculate that achieving this form of generalization would require stronger inductive biases and/or more sophisticated dataset engineering.

5 Conclusions

Fully learned turbulence simulators can learn directly from data, require less specialized engineering, run efficiently on general-purpose hardware, and can be applied to diverse astrophysics environments. We find that our learned simulators outperform comparably coarse solvers (in particular, preserving high frequency structure), and that training noise and temporal coarsening improve stability. However, out-of-distribution generalization remains a challenge for our general-purpose models. Future work should explore improving generalization by extending work on theoretically-motivated state representations and more powerful general-purpose inductive biases.

Acknowledgements

We thank Irina Higgins for providing feedback on an earlier version of the manuscript. D.B.F. is supported by the Simons Foundation through the Flatiron Institute.

References

[1] Guido Boffetta and Robert E. Ecke. Two-dimensional turbulence. *Annual Review of Fluid Mechanics*, 44(1):427–451, 2012. doi: 10.1146/annurev-fluid-120710-101240. URL https://doi.org/10.1146/annurev-fluid-120710-101240.

[2] Guido Boffetta and Robert E Ecke. Two-dimensional turbulence. *Annual Review of Fluid Mechanics*, 44:427–451, 2012.

[3] Axel Brandenburg and Åke Nordlund. Astrophysical turbulence modeling. *Reports on Progress in Physics*, 74(4):046901, April 2011. doi: 10.1088/0034-4885/74/4/046901.

[4] V. M. Canuto and J. Christensen-Dalsgaard. Turbulence in astrophysics: Stars. *Annual Review of Fluid Mechanics*, 30(1):167–198, 1998. doi: 10.1146/annurev.fluid.30.1.167.

[5] Karthik Duraisamy, Gianluca Iaccarino, and Heng Xiao. Turbulence modeling in the age of data. *Annual Review of Fluid Mechanics*, 51(1):357–377, Jan 2019. ISSN 1545-4479. doi: 10.1146/annurev-fluid-010518-040547. URL http://dx.doi.org/10.1146/annurev-fluid-010518-040547.

[6] Drummond B. Fielding, Eve C. Ostriker, Greg L. Bryan, and Adam S. Jermyn. Multiphase gas and the fractal nature of radiative turbulent mixing layers. *The Astrophysical Journal*, 894(2):L24, May 2020. ISSN 2041-8213. doi: 10.3847/2041-8213/ab8d2c. URL http://dx.doi.org/10.3847/2041-8213/ab8d2c.

[7] Jonathan B. Freund, Jonathan F. MacArt, and Justin Sirignano. DPM: A deep learning PDE augmentation method (with application to large-eddy simulation), 2019.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[9] Byungsoo Kim, Vinicius C. Azevedo, Nils Thuerey, Theodore Kim, Markus H. Gross, and Barbara Solenthaler. Deep fluids: A generative network for parameterized fluid simulations. *CoRR*, abs/1806.02071, 2018. URL http://arxiv.org/abs/1806.02071.

[10] Dmitrii Kochkov, Jamie A. Smith, Ayya Alieva, Qing Wang, Michael P. Brenner, and Stephan Hoyer. Machine learning accelerated computational fluid dynamics, 2021.

[11] Yoshiki Kuramoto. Diffusion-Induced Chaos in Reaction Systems. *Progress of Theoretical Physics Supplement*, 64:346–367, 02 1978. ISSN 0375-9687. doi: 10.1143/PTPS.64.346. URL https://doi.org/10.1143/PTPS.64.346.

[12] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations, 2020.

[13] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations, 2020.

[14] Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature Communications*, 9(1):4950, 2018. ISSN 2041-1723. doi: 10.1038/s41467-018-07210-0. URL https://doi.org/10.1038/s41467-018-07210-0.

[15] Steven A. Orszag and G. S. Patterson. Numerical Simulation of Three-Dimensional Homogeneous Isotropic Turbulence. *Physical Review Letters*, 28(2):76–79, January 1972. doi: 10.1103/PhysRevLett.28.76.

[16] Jaideep Pathak, Mustafa Mustafa, Karthik Kashinath, Emmanuel Motheau, Thorsten Kurth, and Marcus Day. Using machine learning to augment coarse-grid computational fluid dynamics simulations, 2020.

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

[18] Li Rhie, C Chow. Numerical study of the turbulent flow past an airfoil with trailing edge separation. *AIAA Journal*, 21(11):1525–1532, 1983.

[19] Ahmed M. Sallam and Ned H. C. Hwang. Human red blood cell hemolysis in a turbulent shear flow: Contribution of reynolds shear stresses. *Biorheology*, 21(6):783–797, 1984.

[20] Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin Riedmiller, Raia Hadsell, and Peter Battaglia. Graph networks as learnable physics engines for inference and control. In *International Conference on Machine Learning*, pp. 4470–4479. PMLR, 2018.

[21] 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*, pp. 8459–8468. PMLR, 2020.

[22] G.I. Sivashinsky. Nonlinear analysis of hydrodynamic instability in laminar flames—i. derivation of basic equations. *Acta Astronautica*, 4(11):1177–1206, 1977. ISSN 0094-5765. doi: https://doi.org/10.1016/0094-5765(77)90096-0. URL https://www.sciencedirect.com/science/article/pii/0094576577900960.

[23] E. R. Smith. A molecular dynamics simulation of the turbulent couette minimal flow unit. *Physics of Fluids*, 27(11):115105, 2015. doi: 10.1063/1.4935213.

[24] James M. Stone, Kengo Tomida, Christopher J. White, and Kyle G. Felker. The athena++ adaptive mesh refinement framework: Design and magnetohydrodynamic solvers. *The Astrophysical Journal Supplement Series*, 249(1):4, Jun 2020. ISSN 1538-4365. doi: 10.3847/1538-4365/ab929b. URL http://dx.doi.org/10.3847/1538-4365/ab929b.

[25] Kiwon Um, Robert Brand, Yun, Fei, Philipp Holl, and Nils Thuerey. Solver-in-the-Loop: Learning from differentiable physics to interact with iterative PDE-solvers, 2021.

[26] Rui Wang, Karthik Kashinath, Mustafa Mustafa, Adrian Albert, and Rose Yu. Towards physics-informed deep learning for turbulent flow prediction, 2020.

[27] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions, 2016.

A Appendix

Hyperparameter	Value
Kernel size	3
Latent size	48
Activation	ReLU
Dilated block depth	7
Dilated block dilations	(1, 2, 4, 8, 4, 2, 1)
# processor blocks N	4
Shared processors?	No
Loss	MSE

Table A.1: Model hyperparameters.

	KS Equation	Incompressible Decaying	Compressible Decaying	Compressible Radiative Cooling Mixing Layer
Numerical Solver	Fourier Method	DNS	`Athena++`	`Athena++`
# Spatial dims	1	2	3	3
# Features	1	2	5	5
Features	v_x	v_x, v_y	ρ, v_x, v_y, v_z, P	ρ, v_x, v_y, v_z, P
Box size				
L_x	2π	2π	1	0.25 to 2
L_y	n/a	2π	1	0.25 to 2
L_z	n/a	n/a	1	3
Grid element size				
Solver	2π / 256	2π / 576	1 / 128	1 / 128
Learned model	2π / 64	2π / 48	1 / 32	1 / 32
(relative to solver)	4x	12x	4x	4x
Warm-up duration	75	500	0.05	2.32
Trajectory duration	181	400	1	7.226
Time step				
Solver	0.5	0.00436	0.0005	0.00012 to 0.00014
Learned model	0.5	3.35	0.032	0.12
(relative to solver)	1x	768x	64x	1000x to 875x
# Trajectories				
Training	1000	190	27	20 if $L_x = 0.75$ 5 if $L_x \neq 0.75$
Validation	100	10	4	1 per L_x
Test	100	10	4	1 per L_x
Training details				
Early stopping?	No	No	No	Yes
Batch size	32	8	1	1 (4 if multisize)
Noise	1e-2	1e-4	1e-2	1e-3

Table A.2: Dataset details. ρ refers to density, P tp pressure, and v_x, v_y, v_z to velocity components. *Warm-up* refers to the initial transient from initial conditions which is influenced by the underlying numerical scheme and discarded from evaluation and training. Figures are dimensionless.

A.1 Additional dataset details

A.1.1 1D Kuramoto-Sivashinsky (KS) Equation

This is a well-studied 1D PDE that generates unstable, chaotic dynamics in 1 dimension [11, 22] with periodic boundaries. The ground truth simulations are computed using the Fourier Spectral Method. Initial condition was set to $\cos(w_1 x + \phi_1)(1 + \sin(w_2 x + \phi_2))$, where x ranges from 0 to 2π, ϕ_i are sampled uniformly from $[0, 2\pi)$, and w_1 are integers sampled uniformly from $[1, 12)$. We perform a *warmup* from this initial condition for 75 simulation time units. This dataset does not have special relevance to astrophysics; however, it is a well-studied chaotic equation that is useful for assessing the ability of our learned models to capture highly unstable nonlinear dynamics.

A.1.2 2D Incompressible Decaying Turbulence

This models fluid flow described by the Navier-Stokes equations in which small-scale eddies decay into the large-scale structures due to the inverse energy cascade [2]. The underlying simulations were performed by solving the incompressible Navier-Stokes equations using a Direct Numerical Simulation (DNS) finite-volume solver. Boundaries along both dimensions are periodic, and the initial conditions consist of random velocity fields with small-scale variation. Initial conditions for different trajectories were obtained by sampling a high-resolution velocity field from a log-normal distribution of amplitude 1 and wavenumber $k = 10$. We perform a *warmup* for 500 simulation time units, this is done to discard the transient flow that is heavily influenced by the underlying numerical scheme. These simulations are scientifically relevant to open questions in atmospheric flows on solar system planets and extrasolar planets [1].

A.1.3 3D Compressible Decaying Turbulence

This models decaying transonic Navier-Stokes turbulent flow in a 3D cubic box with periodic boundary conditions [e.g., 15]. These simulations adopt an adiabatic equation of state with a constant adiabatic index $\gamma = 5/3$. Simulations were carried out with `Athena++` [24]. The initial turbulence is driven on scales $\geq L$, the size of the box. The turbulent driving pattern in the initial condition is split into its compressive and solenoidal components using a Helmholtz decomposition. The relative strength of these two components is varied from purely compressive to purely solenoidal. The initial driving pattern results in a root-mean-squared velocity of $\sqrt{2}c_s$, where $c_s^2 = (5/3)(P/\rho)$ is the sound speed of the fluid. The initial conditions are varied across trajectories by randomizing the phase of the spectral components, leading to different pattern in real space. We perform a *warmup* for 0.05 simulation time units. Compressible turbulence is ubiquitous in astrophysical environments, so understanding the dynamics and properties of these flows on small and large scales plays a crucial role in regulating planet, star, black hole, and galaxy formation [3].

A.1.4 3D Compressible Radiative Cooling Mixing Layer Dynamics

These simulations model the interplay of radiative cooling and mixing that results from turbulence driven by the Kelvin-Helmholtz instability, which can arise when there is velocity difference across the interface between two fluids of different densities. This process is common in essentially all aspects of galaxy formation [6] and has close parallels to many processes in atmospheric flows. The simulations are set up as a boundary problem initialized with a low-density fluid ($\rho = 0.01$) on the top half of the domain ($z > 0$) moving in the positive x direction ($v_x = 2.04, v_y = v_z = 0$), and a high-density fluid ($\rho = 1.$) moving in the negative x direction in the bottom half of the domain ($v_x = -2.04, v_y = v_z = 0$). The initial pressure is set to 1. The x and y boundaries are periodic, while the z boundary (perpendicular to the interface) is fixed to the initial conditions. To break the symmetry, small perturbations to v_z are added around the boundary. This perturbations are changed across trajectories by randomizing the phase of the spectral components, leading to different pattern in real space. We perform a *warmup* for 2.32 simulation time units. Simulations were carried out with `Athena++` [24].

This data presents a few unique challenges compared to the others. First, it was the only domain with fixed boundary conditions. Second, because turbulent dynamics are limited to the vicinity of the mixing layer, and because the behavior of the fluid above, at, and below the mixing layer is markedly different, there is less data representative of each type of fluid dynamic behavior.

A.2 Additional model details

The learned simulator consisted of a CNN encoder, a dilated CNN processor with residual skip connections, and a CNN decoder. The parameters of the model are listed in Table A.1.

CNN parameters All individual CNNs use a kernel size of 3 and have 48 output channels (except the decoder, which has an output channel for each feature). Each individual CNN layer in the processor is immediately followed by a rectified linear unit (ReLU) activation function. The Encoder CNN and the Decoder CNNs do not use activations.

Dilated connections The dilation rate in the CNN filter introduces space between each element in the filter. Whereas a standard CNN filter with kernel 3 would operate over 3 adjacent pixels, a dilated CNN filter with kernel 3 and dilation 2 will operate over 3 pixels spaced at intervals of 2 in each dimension, spanning a field 5 pixels wide. This increases the scale of the CNN kernels while preserving the number of parameters and the resolution.

CNN padding For each dimension in $\mathcal{X}$ with periodic boundary conditions, we implemented periodic padding. For dimensions with a fixed boundary condition, we forced the boundary to a value rather than letting the model predict it, and masked the loss so the model was not trained to predict boundary conditions. The tensor was padded with repetitions of the boundary value. We also augmented the input state with a feature that indicated fixed-boundary versus non-boundary states using a one-hot vector, so the model could distinguish them.

A.3 Additional training details

Training noise In some cases, we trained with Gaussian random noise with fixed variance σ added to the input X_t of the loss function. Note that this not only affects the input to the neural network, but also slightly modifies the target $\Delta X = X_{t+\Delta t} - X_t$ (and also impacts the variance used to normalize targets).

Loss At training time we sample pairs of input-output states (separated by the model time step) from the trajectories, and perform gradient updates based on a single step of the model. We do not rollout the model during training.

Optimization We optimized the loss using an Adam optimizer. We trained the models for up to 10M steps, with exponential learning rate decay annealed from $1e-4$ to $1e-7$ in the first 6M steps. Models usually reached convergence at around 5M steps. Training took up to a week on an NVIDIA V100 GPU.

Hyper-parameter optimization We did not perform exhaustive hyper-parameter optimization, except on the scale of the noise (which we scanned for each domain) and some informal tuning of the depth of the dilated blocks (to make sure the network was deep enough for the most challenging domain). Note that achieving optimal performance on test trajectories from the training distribution was not part of the scope of this work, and there are likely hyperparameters that would improve performance over our results.

Validation All of the research was performed by looking at performance on the validation sets. The test set was completely held-out until final evaluation prior to writing the paper.

Early stopping For the Mixing Layer Turbulence (CTRD-3D-ML), which was more prone to overfitting, we used early stopping based on the validation performance. For all other models we simply evaluated the model as it was at the end of training.

A.4 Additional results

Three instances of each model were trained 3 times using 3 different initialization seeds. Bar plots indicate median performance and bars indicate min-max performance. We chose the energy field

$E = \frac{1}{2}\rho v^2 + \frac{3}{2}P$ as the quantitative metric of 3D turbulence for the main text, as it summarizes performance on all state variables.

Figures and videos for 3D environments (CTRD-3D-ML and CT-3D) show a single slice of the 3D grid at $y = 0$, displaying the x and z coordinates in the horizontal and vertical axis, respectively.

Downsampling in time Fig. A.2 as well as videos (tinyurl.com/dl4sturb) show models for KS-1D, IT-2D, and CT-3D models trained and working well on a wide range of time step sizes. Note that for a fair comparison in Fig. 2d,f performance is averaged across only time steps that are predicted for all models (e.g. multiples of the largest time step, 256).

Downsampling in space All comparisons across spatial resolutions are always obtained by first downsampling the data into a common 32^3 grid, using an approach that preserves mass, momentum, and energy (Same approach used by `Athena++`). Fig. A.3 shows the downsampling comparisons equivalent to Fig. 2g-i for each of the 5 state variables independently.

Running time For 1 simulation time unit of CT-3D, the CPU runtime for `Athena++` with 8 CPU processors is $\sim$4s, $\sim$60s, and $\sim$1000s for 32^3, 64^3, and 128^3 resolutions, respectively (quartic scaling, as the time step is also scaled with the resolution to compensate for numerical viscosity). In comparison, the learned model's runtime is 1s on an NVIDIA V100 GPU, and 20-30s on a 8-core CPU. Note that the learned model runs at reduced spatial and temporal resolution, but preserves the dynamics of the high resolution 128^3 `Athena++` simulation. Simulations in `Athena++` may be faster if implemented for GPUs. However, because scientific simulators like `Athena++` are specialized, each simulator's GPU implementation requires specialized engineering effort, whereas learned models can take advantage of methods designed more generally for deep learning.

Cooling velocity Having reliable models that generalize to larger boxes increases the applicability of learned models to scientific domains by enabling experiments in regions of parameter space that would otherwise be prohibitively expensive to simulate. For example, for CTRC-3D-ML, the dynamics undergo a transition when the box width increases relative to the cooling length ($v_{\text{turb}} t_{\text{cool}}$), which is difficult to study because simulations for these widths essentially prohibitive. Thus, we want to understand what affects the learned simulator's ability to generalize to a range of box widths (the length L_x and width L_y of the input tensor X) not previously seen in the training data. For CTRD-3D-ML, we can look at the cooling velocity, the average in-flowing velocity at the low density fluid boundary that develops as a means to resupply the energy that has been radiated away in the mixing layer. Cooling velocity is a useful metric because it is a scalar quantity that depends on the box width. Furthermore, understanding how turbulent dynamics at a mixing layer pull heat from the surroundings is scientifically relevant for questions in astrophysics [6]. We evaluate generalization to different box sizes ($L_x = L_y \in [0.25, 0.5, 0.75, 1.0, 1.25, 1.5, 1.75, 2.0]$) (Fig. 2k). We find that the model trained on box length $L_x = L_y = 0.75$ (orange) is not able to produce trajectories with the correct cooling velocity for other box sizes. However, we also find that augmenting the dataset with data from a range of box sizes ($L_x = L_y \in [0.5, 0.75, 1.0, 1.25]$) improves accuracy of the cooling velocity estimate for the unseen box lengths, although the generalization outside the training domain remains imperfect and unreliable across seeds. We speculate that achieving this form of generalization would require stronger inductive biases and/or more sophisticated dataset engineering.

KS-1D generalization Figs. A.4 and A.5 show generalization to larger domains and longer trajectories. While the learned simulations do not perfectly capture the ground truth, we see that qualitative features of turbulence are preserved across the rollout.

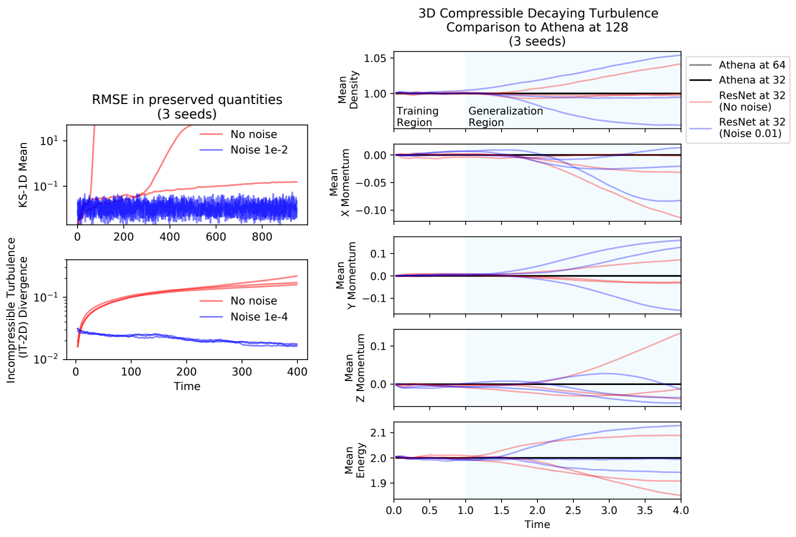

Figure A.1: (left) KS equation (KS-1D) net velocity error, and Incompressible Turbulence (IT-2D) divergence error as function of model step. Training with noise helps keeping the values bounded to be close to 0. (right) Preservation of the 5 conserved quantities in 3D Compressible Turbulence (CT-3D) as a function of time. In this case, training with noise does not completely prevent drift of the conserved quantities.

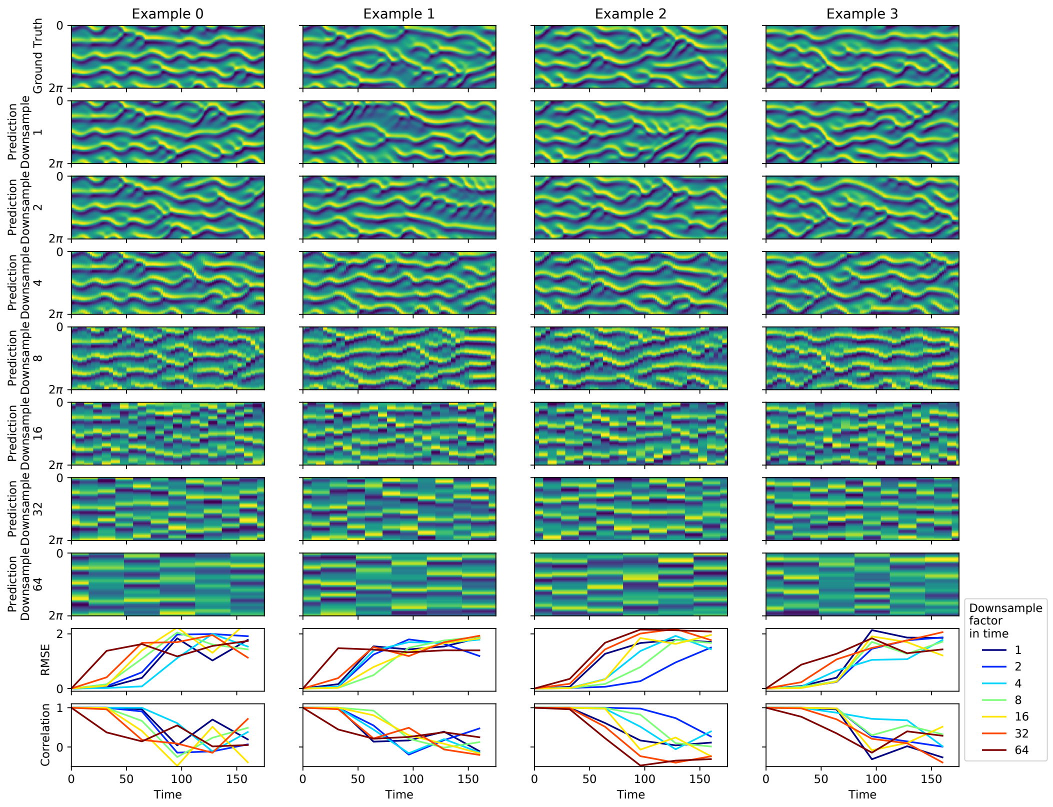

Figure A.2: (top) Sample trajectories from the model trained on the KS-1D dataset at different temporal downsampling factors. (bottom) MSE and correlation performance of ML models for different downsampling factors in time as a function of simulation steps.

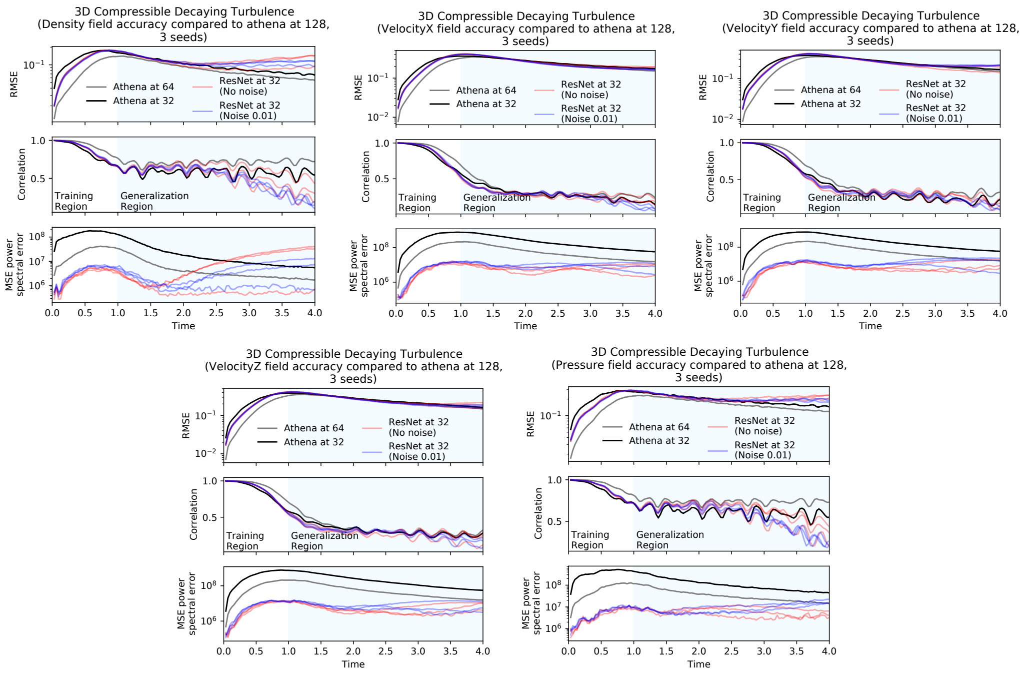

Figure A.3: (CT-3D) Per-variable (Density, Velocity X, Velocity Y, Velocity Z and Pressure) MSE error, correlation, and power spectrum error as a function of time, for rollouts of learned models trained on high-resolution Athena data (128^3) downsampled to 32^3, and for Athena ran at lower spatial resolutions (64^3, 32^3).

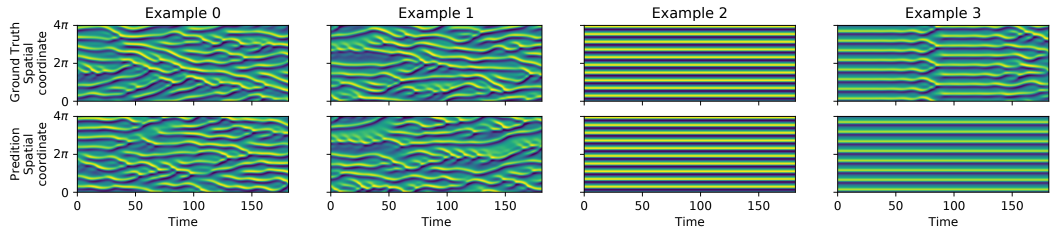

Figure A.4: KS-1D model generalizing to larger spatial domains (trained on domains 2π wide).

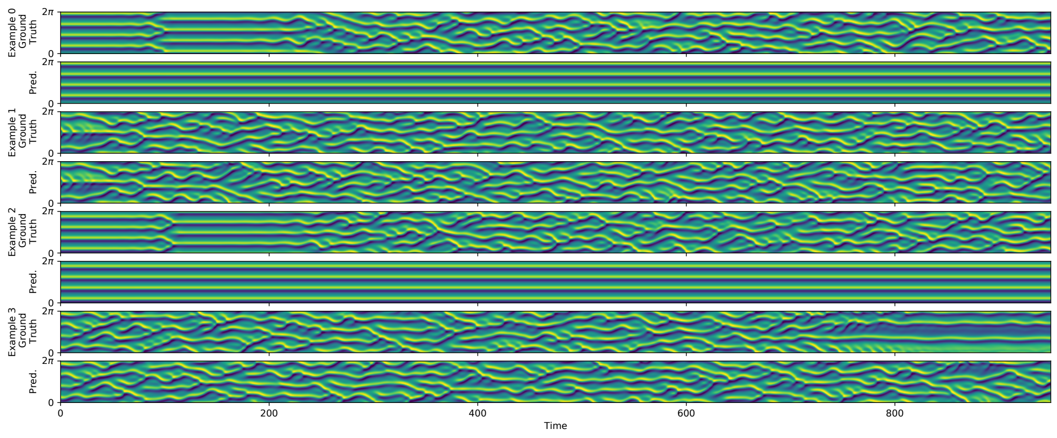

Figure A.5: KS-1D model generalizing to longer trajectories (trained on trajectories with 181 simulation time units).