Learning latent field dynamics of PDEs

Dmitrii Kochkov *
Google Research
dkochkov@google.com

Alvaro Sanchez-Gonzalez
DeepMind

Jamie Smith
Google Research

Tobias Pfaff
DeepMind

Peter Battaglia
DeepMind

Michael P. Brenner
School of Engineering and Applied Sciences, Harvard University
Google Research

Abstract

We present a new approach to learning surrogate models for simulation of complex physical systems described by nonlinear partial differential equations. It aims to capture three features of PDEs: locality, time continuity and formation of elementary patterns in the solution by learning a local, low dimensional latent representation and corresponding time evolution. We show that our method achieves top performance and competitive inference speed compared to baseline methods while operating with a 4-times more compact representation. Since the models learn local representations of the solution, they generalize to different system sizes that feature qualitatively different behavior without retraining.

1 Introduction

Predicting the evolution of a complex physical system through numerical simulation is a central problem throughout engineering and the physical sciences. In many cases, the differential equations governing the system are well understood, however, resolving the small spatiotemporal features required to solve these equations still presents significant computational challenges. Recently there has been a surge of interest in applying machine learning (ML) to address these challenges. There are roughly three distinct directions emerging in the literature: (1) discovery of governing equations directly from data [3, 20, 12]; (2) development of augmented numerical methods [2, 21, 7, 15, 18]; (3) surrogate modeling [4, 19, 1, 14, 13, 5].

Here we consider a class of surrogate models that incorporate three prominent PDE priors:

- *Locality*: time evolution at every point x depends only on its immediate neighborhood.
- *Time continuity*: solutions to partial differential equations change continuously with time.
- *Formation of patterns*: emergence of smallest scales that build up the solution manifold.

We present our results by evaluating the performance of this family of models on the chaotic Kuramoto-Sivashinsky equation [11, 16].

2 Background

In this work we are interested in capturing the dynamics of a physical system that is described as a partial differential equation of the form Eq. (1), where $\mathbf{u}(x, t)$ represents the configuration of the

*corresponding author: dkochkov@google.com

Third Workshop on Machine Learning and the Physical Sciences (NeurIPS 2020), Vancouver, Canada.

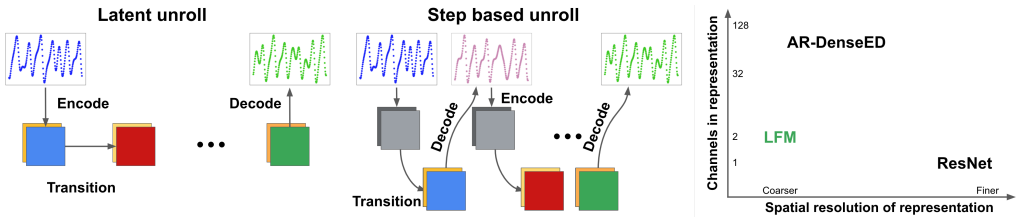

Figure 1: (left) Pictorial representation of a latent unroll scheme. (center) Step based unroll scheme that obtains intermediate results at fixed intervals. (right) A comparison of representation sizes used by different models for computation of time evolution, x-axis represent spatial resolution and y-axis represents the number of channels (local dimensionality).

system at position x at time t.

$$\frac{\partial \mathbf{u}(x,t)}{\partial t} = f(\mathbf{u}, \nabla \mathbf{u}, \ldots, \nabla^n \mathbf{u}) \tag{1}$$

Partial differential equations describe a wide variety of complex phenomena, ranging from turbulence in Navier-Stokes equations to pattern formation in reaction-diffusion equation. For many systems of practical interest, dissipation leads to the emergence of an inertial manifold [17, 9].

Traditional numerical methods capture the dynamics via iterative updates, computing the evolving state at every time step. This is typically accomplished by discretizing the solution on a spatial grid and specifying a functional "basis" such as polynomials, hat-basis or Fourier modes. To represent the solution of the underlying PDE, numerical methods must maintain fine enough discretization to accurately represent the smallest length scales of the solution. This requirement causes computational degrees of freedom to far exceed the dimension of the inertial manifold. Furthermore, the choice of discretization affects the time step that can be taken at each iteration without introducing large errors, increasing computational cost.

In this work we formulate a class of ML models that alleviate some of these obstacles by using learned representations of the solution manifold that (1) preserve spatial structure of the solution and *locality* of interactions; (2) evolve *continuously in time* and (3) efficiently parameterize emergent *patterns* in the solution.

3 Construction and Training of the Latent Field Models

3.1 The Encode-Transition-Decode Architecture

The *latent field model* (LFM) consists of three components: an *encoder* $\mathcal{E}$, a *decoder* $\mathcal{D}$, and a *derivative model* $\mathcal{F}$. Together, they define the relationship between the physical states of the system $\mathbf{u}(t)$, corresponding latent states $\mathbf{z}(t)$ and their time derivatives $\dot{\mathbf{z}}$ (Eq. 2).

$$\mathbf{z}(t) = \mathcal{E}(\mathbf{u}(t)) \qquad \dot{\mathbf{z}}(t) = \mathcal{F}(\mathbf{z}(t)) \qquad \tilde{\mathbf{u}}(t) = \mathcal{D}(\mathbf{z}(t)) \tag{2}$$

To predict the future state $\tilde{\mathbf{u}}(t)$ of the system given an initial configuration $\mathbf{u}(0)$,

1. The initial state is encoded: $\mathbf{z}(0) = \mathcal{E}(\mathbf{u}(0))$
2. The system is evolved by numerical integration of $\dot{\mathbf{z}}(t) = \mathcal{F}(\mathbf{z}(t))$ on the $[0, t]$ interval.
3. The final latent state is decoded to a physical state: $\tilde{\mathbf{u}}(t) = \mathcal{D}(\mathbf{z}(t))$

Note that time evolution occurs in latent space. This differs from the standard approach where time evolution happens iteratively in the physical state (analogous to applying *encoder-transition-decoder* at every step). The difference between these approaches is illustrated in Fig. 1.

For all components ($\mathcal{E}$, $\mathcal{D}$, $\mathcal{F}$) we use fully convolutional architectures with periodic boundary conditions. This preserves the spatial structure of the representation and ensures that time dynamics depend only on the immediate neighbors of each spatial point.

3.2 Loss Functions and Training

For a 'perfect' model, we would have

$$\mathcal{D}(\mathcal{E}(\mathbf{u}))(t) = \mathbf{u}(t) \qquad J_{\mathcal{E}}(\mathbf{u})\dot{\mathbf{u}}(t) = \dot{\mathbf{z}}(t) \qquad J_{\mathcal{D}}(\mathbf{z})\dot{\mathbf{z}}(t) = \dot{\mathbf{u}}(t)$$

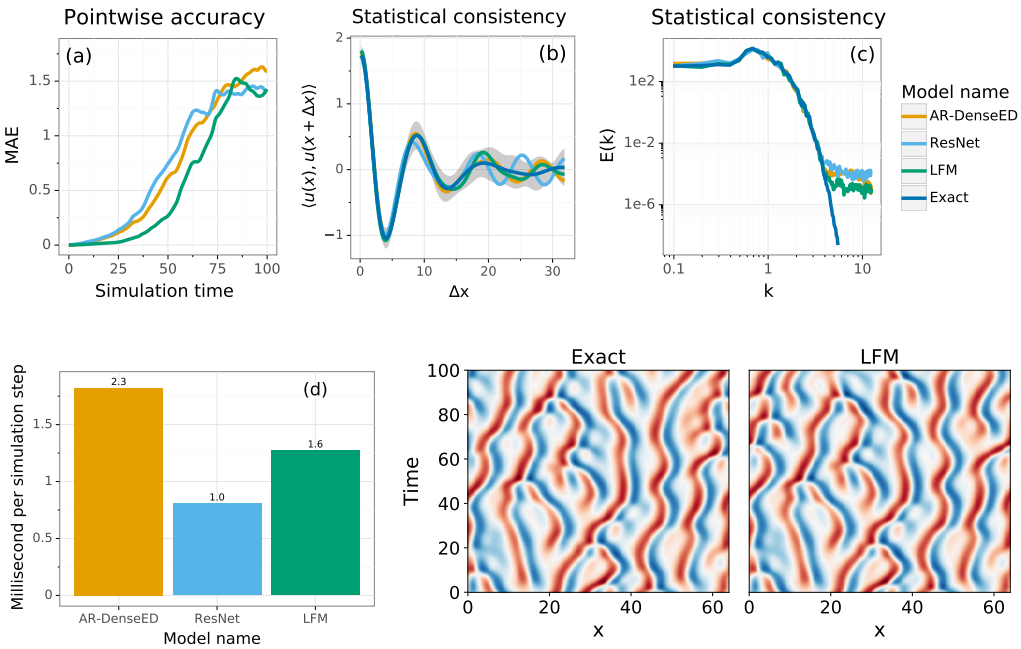

Figure 2: LFM achieves the highest accuracy and fast inference while operating in reduced representation. (a) Mean absolute error as a function of simulation time; (b) velocity correlation function at $t = 90$; (c) energy spectrum at $t = 90$. The deviation at high frequencies corresponds to modes of irrelevant amplitude; (d) wall-clock time used to predict 1000 time steps ahead; (bot right) LFM model unroll and reference evaluation data respectively.

where $J_{\mathcal{E}}(\mathbf{u})$ is the Jacobian of the encoder at $\mathbf{u}$, $J_{\mathcal{D}}(\mathbf{z})$ is the Jacobian of the decoder at $\mathbf{z}$ and $\dot{\mathbf{z}}(t)$ is the time derivative of the latent state computed as $\mathcal{F}(\mathbf{z}(t))$. This motivates our loss function:

$$L = L_1 + L_2 + L_3 \tag{3}$$

$$L_1 = \text{MSE}(\mathbf{u}, \mathcal{D}(\mathcal{E}((\mathbf{u})))) \quad L_2 = \text{MSE}(\dot{\mathbf{z}}, J_{\mathcal{E}}(\mathbf{u})\dot{\mathbf{u}}) \quad L_3 = \text{MSE}(\dot{\mathbf{u}}, J_{\mathcal{D}}(\mathbf{z})\dot{\mathbf{z}})$$

This method has been used for equation discovery in [3]. Note that *no time evolution is performed during training*. This avoids the expense of propagating gradients through the ODE integrator.

4 Related work

The idea of learning a surrogate model from time-series data for complex physical systems has been actively addressed in the recent literature [4, 19, 1, 14, 13, 5, 8]. One of the central themes in this line of research is the search for generic priors that boost accuracy of the resulting surrogates.

PDE-net [13] proposed a ResNet-like architecture capturing the time continuity of PDEs. Later replaced with differentiable integrators applied for problems with partial [1] and unstructured [8] data. Works [19, 14, 4] investigated priors originating in numerical methods such as scale separation [19], Koopman operators [14] and Lyapunov stability [4].

Our approach of using local latent representations to represent and time evolve a dynamical system is most closely compared to AR-DenseED architecture [5] and Lyapunov-stable model [4], both of which use an encoder-decoder structures. In contrast, we consider the scheme of obtaining a model trajectory by unrolling the dynamics purely in the latent space. In addition, compared to [4] we consider nonlinear dynamics, necessary for chaotic systems. Compared to AR-DenseED we additionally include a time continuity prior on the latent space.

5 Experiments and results

We train and evaluate the performance of latent field models and relevant baselines on the Kuramoto-Sivashinsky [11, 16] equation (Eq. 4). This system is known to have a low dimensional inertial

manifold, and to exhibit chaotic behavior.

$$\partial_t \mathbf{u} + \mathbf{u}\partial_x \mathbf{u} + \partial_x^2 \mathbf{u} + \partial_x^4 \mathbf{u} = 0 \tag{4}$$

We use a dataset that consists of trajectories and time derivatives of the solutions in the stationary chaotic regime generated using a high resolution simulator. See Appendix A for details.

Due to the chaotic nature of the problem, even close-to-perfect models would eventually diverge from the reference solution. This suggests an evaluation strategy that separately probes short term accuracy and long term consistency. Short term accuracy is evaluated point-wise, while long time scale behavior is examined based on statistically invariant quantities such as energy spectrum (Eq. 6) or (autocorrelation function Eq. 7). We also measure efficiency of the surrogates by recording a wall-clock time required to predict the state of the system at a future time.

By varying the size of the latent representation we found that LFM model can accurately capture the dynamics on 8 times coarser resolution, while using only 2 latent dimension for each spatial location. This amounts to 4-fold memory savings arising from a more compact representation of the state.

We now compare the performance of LFM to AR-DenseED [5] and a ResNet [6] architectures. For AR-DenseED we used parameters suggested in the original paper [5] and for ResNet we adjusted the number of parameters to approximately match the capacity of the LFM model. See Appendix B for details. At training, the models ingest either mini-batches of states and their time derivatives or mini-batches of trajectories of length T, depending on the training method (see Appendix C for baseline training). For each model we performed a hyperparameter sweep (see C for details) and compare the performance of configurations that achieved lowest cumulative pointwise error on the training data over the course of 100 time slices.

The evaluation results of the best performing models for each architecture are shown in Fig. 2. They compare both pointwise accuracy and statistical consistency. Our statistical consistency evaluation suggests that all models accurately capture the hidden structure of the chaotic dynamics. From the pointwise accuracy perspective we find that our LFM model remains close to the ground truth solution for longer than other models. See Fig. 2 (bottom right). While pointwise error is a practical and important for applications, we found that with current training methods and models it is unreliable for model ranking. This concern is motivated by the large variation of model performance across the random initializations, suggesting a lack of training pressure on error accumulation - a critical property for surrogate models. In Fig. 2 (d) we compare model efficiency. We find all models competitive with ResNet being fastest by a small margin.

Because our latent representation has spatial structure, it is possible to apply the model trained on one system size to domains of different size. This is a good "stress test" since larger domains develop spatiotemporal chaos of greater complexity. We evaluate the LFM model on large and small grids starting from an out of distribution initial condition to assess the performance on the transient phase and the equilibrium states (Fig. 3). The model correctly captures the transient and settles in the expected regime at longer times. This generalization is made possible because the model only relies on local updates that are intrinsically present in a single turbulent dataset.

6 Conclusions

In this work we have explored models that learn local latent representations of a dynamical system and corresponding time evolution. We demonstrated that it is possible to obtain low dimensional representation of the solution manifold that uses 4 times fewer parameters to represent the system, achieves top performance and is among fastest at inference models. We additionally demonstrated that local model generalize to systems of different size, capturing the change in the behavior of the solution.

7 Broader Impact

This work contributes to the fields of scientific computing and simulation. Our datasets consist only of snapshots of the state of a physical system. As a result, we don't believe this work has any implications for privacy or fairness.

As with any work in the physical sciences, we need to be aware of potentially harmful applications, such as military technology. However, we believe that advances in scientific computing are generally positive developments that advance quality of life.

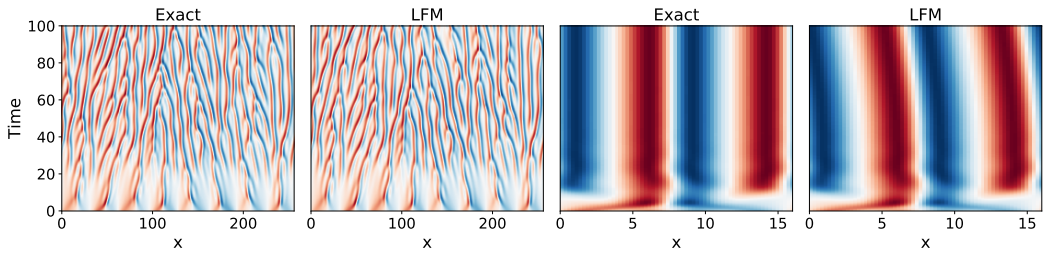

Figure 3: (left) Model and ground truth trajectories on a larger domain that show that model correctly captures the transient phase and settles into chaotic regime as expected. (right) Same for smaller domain that instead settles in a fixed pattern.Be Note that the model has not seen any stationary solutions in the training data.

References

[1] I. Ayed, E. de Bézenac, A. Pajot, J. Brajard, and P. Gallinari. Learning dynamical systems from partial observations. *arXiv preprint arXiv:1902.11136*, 2019.

[2] Y. Bar-Sinai, S. Hoyer, J. Hickey, and M. P. Brenner. Learning data-driven discretizations for partial differential equations. *Proceedings of the National Academy of Sciences*, 116(31):15344–15349, 2019.

[3] K. Champion, B. Lusch, J. N. Kutz, and S. L. Brunton. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences*, 116(45):22445–22451, 2019.

[4] N. B. Erichson, M. Muehlebach, and M. W. Mahoney. Physics-informed autoencoders for lyapunov-stable fluid flow prediction. *arXiv preprint arXiv:1905.10866*, 2019.

[5] N. Geneva and N. Zabaras. Modeling the dynamics of pde systems with physics-constrained deep auto-regressive networks. *Journal of Computational Physics*, 403:109056, 2020.

[6] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[7] J.-T. Hsieh, S. Zhao, S. Eismann, L. Mirabella, and S. Ermon. Learning neural pde solvers with convergence guarantees. *arXiv preprint arXiv:1906.01200*, 2019.

[8] V. Iakovlev, M. Heinonen, and H. Lähdesmäki. Learning continuous-time pdes from sparse data with graph neural networks. *arXiv preprint arXiv:2006.08956*, 2020.

[9] M. S. Jolly, R. Rosa, R. Temam, et al. Evaluating the dimension of an inertial manifold for the kuramoto-sivashinsky equation. *Advances in Differential Equations*, 5(1-3):31–66, 2000.

[10] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[11] Y. Kuramoto and T. Tsuzuki. Persistent propagation of concentration waves in dissipative media far from thermal equilibrium. *Progress of theoretical physics*, 55(2):356–369, 1976.

[12] S. Lee, M. Kooshkbaghi, K. Spiliotis, C. I. Siettos, and I. G. Kevrekidis. Coarse-scale pdes from fine-scale observations via machine learning. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 30(1):013141, 2020.

[13] Z. Long, Y. Lu, X. Ma, and B. Dong. Pde-net: Learning pdes from data. In *International Conference on Machine Learning*, pages 3208–3216, 2018.

[14] S. E. Otto and C. W. Rowley. Linearly recurrent autoencoder networks for learning dynamics. *SIAM Journal on Applied Dynamical Systems*, 18(1):558–593, 2019.

[15] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[16] G. Sivashinsky. Nonlinear analysis of hydrodynamic instability in laminar flames—i. derivation of basic equations. *AcAau*, 4(11):1177–1206, 1977.

[17] E. S. Titi. On approximate inertial manifolds to the navier-stokes equations. *Journal of mathematical analysis and applications*, 149(2):540–557, 1990.

[18] K. Um, P. Holl, R. Brand, N. Thuerey, et al. Solver-in-the-loop: Learning from differentiable physics to interact with iterative pde-solvers. *arXiv preprint arXiv:2007.00016*, 2020.

[19] R. Wang, K. Kashinath, M. Mustafa, A. Albert, and R. Yu. Towards physics-informed deep learning for turbulent flow prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1457–1466, 2020.

[20] B. Xiong, H. Fu, F. Xu, and Y. Jin. Data-driven discovery of partial differential equations for multiple-physics electromagnetic problem. *arXiv preprint arXiv:1910.13531*, 2019.

[21] J. Zhuang, D. Kochkov, Y. Bar-Sinai, M. P. Brenner, and S. Hoyer. Learned discretizations for passive scalar advection in a 2-d turbulent flow. *arXiv preprint arXiv:2004.05477*, 2020.

A Dataset details

Training and evaluation data were generated using a high resolution numerical solver. For one dimensional Kuramoto-Sivasinsky equation we used a spectral method to compute time derivatives of the solution in combination with an adaptive time stepping method DormandPrince from scipy library. The solution was obtained on a grid of size 256 corresponding to the domain size $L = 64$. In this regime the solution is chaotic with maximum Lyapunov exponent around 0.08. The data trajectories of length 100.0 time units were selected from the random initial solutions after discarding the transient phase. The time separation between the sequential time slices in the dataset was chosen to be 0.1. The resulting dataset has a mean of zero and variance of 1.3. Mean and standard deviation of the time derivatives is 0.31.

B Architecture details

All architectural components of the reported low dimensional latent field (LDLF) are based on composition of convolutional towers with periodic padding. The encoder architecture consists of a tower with strides $[1, 2, 1, 2, 1, 2]$, 64 channels in hidden layers and 2 output channels. Decoder has a transposed architecture that has only differs in the final layer that has 1 channel and has kernel size of 12. All activation function in the encoder and decoder networks are ReLU. Time derivative network contained 4 layers with kernel sizes $[3, 1, 1, 1]$ and channels $[64, 32, 32, 2]$ interleaved with tanh nonlinearities. Time stepping was performed using an adaptive Dormand-Prince integrator.

Other models trained for comparison included ResNet and AR-DenseED architectures. For ResNet we best performing model used 3 blocks with 64 hidden channels and 4 layers each. For AR-DenseED architecture we followed the design provided by the authors. For both baselines the batch normalization modules were omitted.

C Training and evaluation parameters

During training we have performed a hyperparameter sweep and reported results from the best performing configurations for each architecture. All models were optimized using Adam[10] optimizer with $b_1 = 0.9$ and $b_2 = 0.99$. Learning rate was explored in the range $[0.01, 0.0005]$. We included multiple initialization seeds for all models in the attempt to determine the best case performance for each model. AR-DenseED and ResNet were trained using unroll loss (Eq. 5), where we have varied the unroll length T in the range $[1, \ldots, 5]]$.

$$L(x, y) = \sum_{t_i}^{t_T} \text{MSE}(u(t_i), \tilde{u}(t_i)) \tag{5}$$

For evaluation of statistical consistency we used autocorrelation (Eq. 7) and energy spectrum (Eq. 6)

$$E(k_j) = \frac{1}{2}|\mathbf{u}(k_j)|^2 \qquad \mathbf{u}(k_j) = \sum_{i=0}^{i=N} \mathbf{u}(x_i) e^{-2\pi i \frac{k_j x_i}{L}} \tag{6}$$

$$\langle \mathbf{u}(x), \mathbf{u}(x + \Delta x) \rangle = \sum_{i,y} \mathbf{u}(x_i(t)) \mathbf{u}(x_i(t) + \Delta x) \tag{7}$$

To evaluate generalization on larger and smaller grids we used domains of size $L = 256$ and $L = 16$ respectively, obtaining ground truth results using the same solver as used for dataset generation (A). The initial conditions used to probe the transient phases are:

```
y_large = (1.2*np.sin(2*x)+0.13*np.sin(5*x)+
           0.23*np.cos(3*x)+0.4*np.sin(0.25*x+0.34))

y_small = (1.2*np.sin(4*x+0.3)+0.13*np.sin(8*x+0.9)+
           0.23*np.cos(12*x-0.6)+0.4*np.sin(12*x+0.34))
```

To measure performance we used 1000 time steps on a grid with resolution 1024 to moderately load the accelerators and avoid unwanted instrumentation effects. Execution time for all models was measured on Nvidia V100 GPU.