Google Research

CEMRACS 23
Summer school

Leonardo Zepeda-Núñez, Senior Research Scientist

Google Research

Scientific Machine Learning

Section description

What is this lecture about

3 Main Objectives:

What is our take on SciML

Provide an introduction to Hyper-Networks and Diffusion Models

Introduce tools adapted to SciML

Outline

1 Block: Perspectives in SciML

Who we are, and what we do

Learning dynamics with HyperNetworks

2 Block Generative Modeling

- Classical Sampling Methods
- Diffusion Models

3 Block Probabilistic Modeling via Generative AI.

Outline

Computer Lab: Led by Martin Guerra (University of Wisconsin-Madison)

JAX as accelerated numpy and JAX transformations

Langevin Dynamics

Diffusion Models

https://github.com/google/jax

Block 1: Who we are and what we do

Google Research

Bio (short)

Non-standard path (very usual nowadays)

Born and raised in the Atacama Desert, Chile

Diploma ,*Ecole Polytechnique* X2006, France

Master Numerical Analysis and PDEs, *Université de Paris VI*

Ph.D. Mathematics, *MIT*

Visiting Assistant Professor, *UC Irvine*

Postdoc, *Lawrence Berkeley National Lab/UC Berkeley*

Assistant Professor of Mathematics, *UW-Madison*

Senior Research Scientist, Google Research

Spectral Methods for Navier Stokes

Shape Optimization

Fast Methods

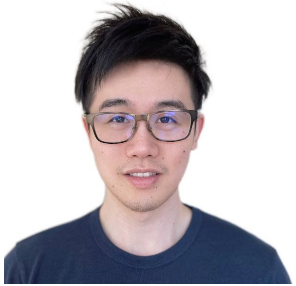
Wave propagation/Inverse problems

Quantum Chemistry

Scientific Machine Learning

SciML for Weather and Climate

Who we are



Zhong Yi Wan

Anudhyan Boral

Leonardo Zepeda-Núñez

Fei Sha

Mission Foundational technologies that drive **efficient modeling** of large-scale, high-stake, and **computationally intensive** physical systems

What we do

Upstream ML Research

Probabilistic Modelling

- M. A. Finzi, A. Boral, A. G. Wilson, F. Sha, L. Zepeda-Núñez, User-defined Event Sampling and Uncertainty Quantification in Diffusion Models for Physical Dynamical Systems, ICML 2023
- Z. Y. Wan, R. Baptista, Y. Chen, J. Anderson, A. Boral, F. Sha, L. Zepeda-Núñez. Debias Coarsely, Sample Conditionally: Statistical Downscaling through Optimal Transport and Probabilistic Diffusion Models, submitted to NeurIPS 2023

Dynamical Systems

- A. Boral, ZY Wan, L Zepeda-Núñez, J. Lottes, Q. Wang, Y. Chen, J. Anderson, F Sha. Neural Ideal Large Eddy Simulation: Modeling Turbulence with Neural Stochastic Differential Equations, submitted to NeurIPS 2023
- Z. Y. Wan, L. Zepeda-Núñez, A. Boral, F. Sha. Evolve Smoothly, Fit Consistently: Learning Smooth Latent Dynamics For Advection-Dominated Systems, ICLR 2023
- G. Dresdner, D. Kochkov, P. Norgaard, L. Zepeda-Núñez, J. A. Smith, M. Brenner, S. Hoyer. Learning to correct spectral methods for simulating turbulent flows. TMLR 2023.

Machine Learning by tasks

Machine learning can be roughly divided into 3 buckets:

Classification	Learning a **Partition** of a domain	Meshing techniques
Regression	Learning a **Map**	Approximating functions Solving ODEs/PDEs Approximating dynamics
Generation	Learning a **Distribution**	Solving SDEs Sampling from Distributions Uncertainty quantification

Classical Problems in Numerical Analysis / Computational Maths

Difference**: Much Higher Dimension!!**

Scientific Machine Learning

Focus: **High** Dimensional problems

Two stage approach:

- Use Numerical Analysis / Computational Maths insight to enhance ML techniques.
- Use ML techniques to solve Scientific Problems (high dimensional ones!)

Learning Dynamics with Machine Learning

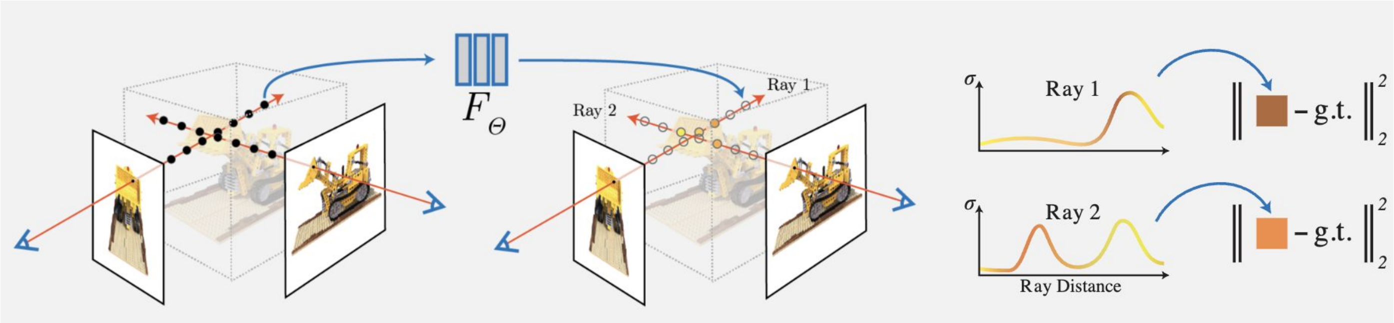
Different approaches

Physics Driven

Solving PDEs (seen Tuesday)
Learning operators (seen Wednesday)

Data-Driven

Reduced order models (seen Monday)
HyperNetworks + NeRF -> Neural Radiance Fields (this morning!)



https://www.matthewtancik.com/nerf

Google Research

Learning Dynamics with Machine Learning

Different approaches

Physics Driven

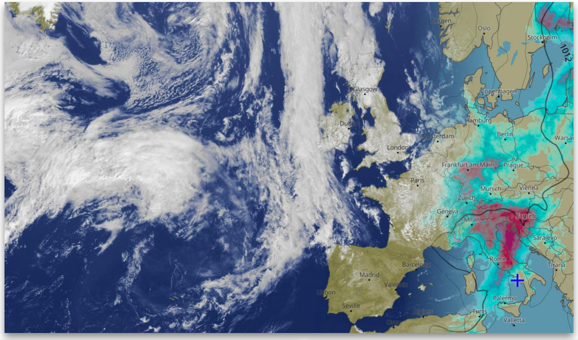
Solving PDEs (seen Tuesday)
Learning operators (seen Wednesday)

Data-Driven

Reduced order models (seen Monday)
HyperNetworks + NeRF -> Neural Radiance Fields (this morning!)

https://www.matthewtancik.com/nerf

Modeling Complex Time-dependent Systems



[Photo Credits: MeteoBlue, CNBC, MIT News]

Prediction problem

Given	present state	$u(x, 0)$
Target	future state after	$u(x, t)$

Main Assumption

- No “exact” PDE model / solver available

Emphasis

- Learn model from data
- Incorporate physics-based inductive biases
- Sampling and inference efficiency

Setup

- **Time dependent (but unknown) PDE**

$$\begin{cases} \partial_t u(x,t) &= \mathcal{F}[u(x,t)], \\ u(x,0) &= u_0. \end{cases}$$

- **Snapshot**

$$u(t_i) := u(x, t_i)$$

- **Trajectories**

$$u_0 \sim \mathcal{D} \qquad\qquad [u_0, u(t_1), u(t_2), \ldots, u(t_{n-1}), u(T)]$$

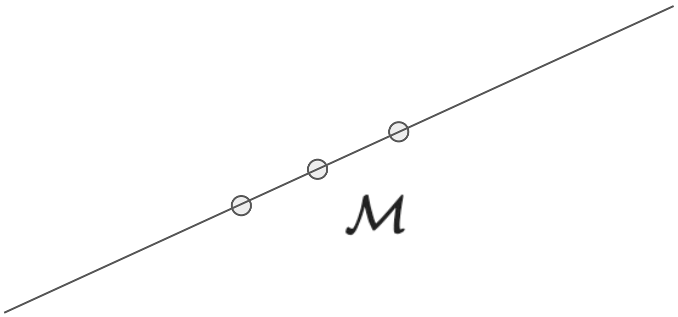
- **Objectives:** from only trajectory data,
  - can we learn a **compressed/latent representation** of a state?
  - can we learn the **dynamics**?

Kolmogorov n-widths

$$d_N(\mathcal{M}) := \inf_{V_N \subset H} \sup_{u \in \mathcal{M}} \inf_{v_N \in V_N} \|u - v_N\|$$

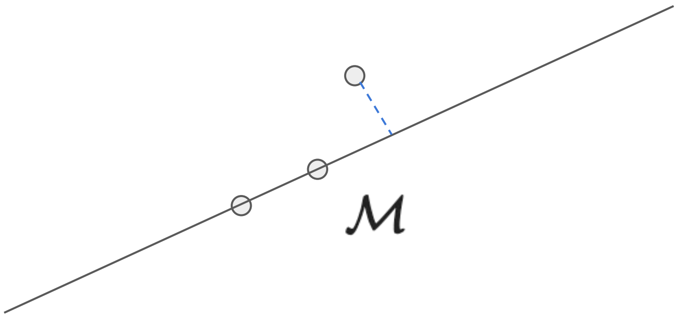
Kolmogorov n-widths

$$d_N(\mathcal{M}) := \inf_{V_N \subset H} \sup_{u \in \mathcal{M}} \inf_{v_N \in V_N} \|u - v_N\|$$



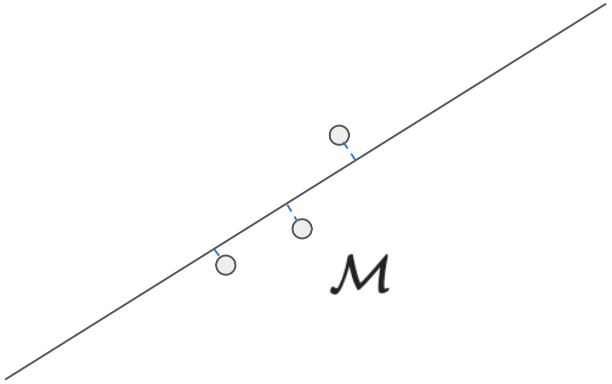
Kolmogorov n-widths

$$d_N(\mathcal{M}) := \inf_{V_N \subset H} \sup_{u \in \mathcal{M}} \inf_{v_N \in V_N} \|u - v_N\|$$



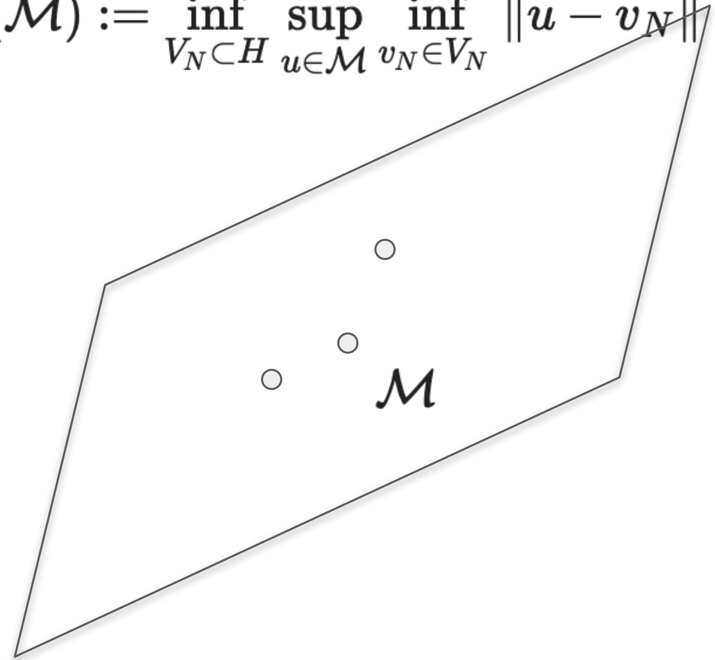
Kolmogorov n-widths

$$d_N(\mathcal{M}) := \inf_{V_N \subset H} \sup_{u \in \mathcal{M}} \inf_{v_N \in V_N} \|u - v_N\|$$



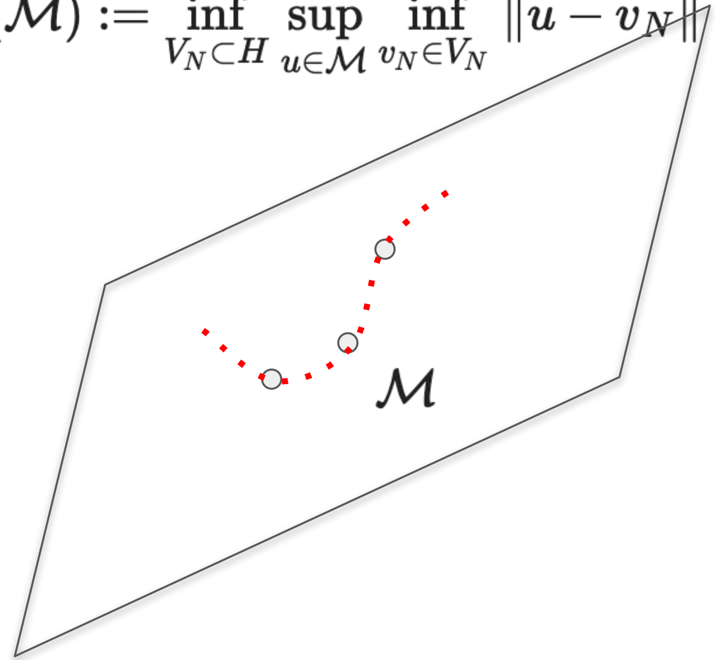
Kolmogorov n-widths

$$d_N(\mathcal{M}) := \inf_{V_N \subset H} \sup_{u \in \mathcal{M}} \inf_{v_N \in V_N} \|u - v_N\|$$



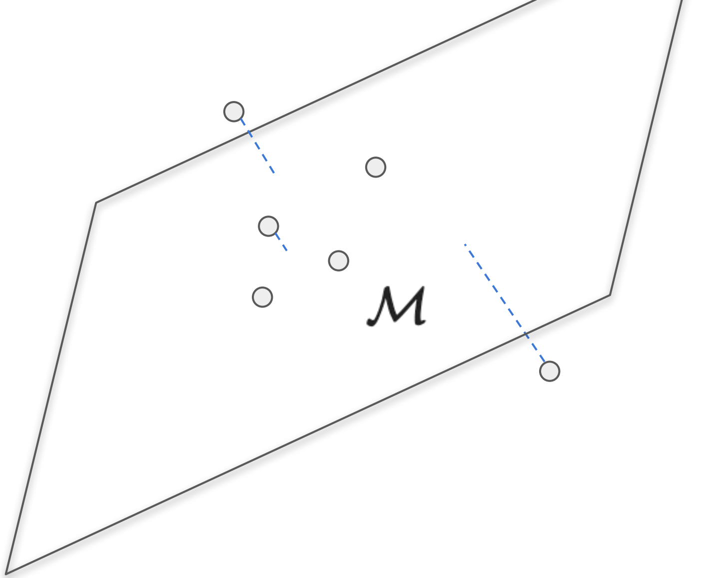
Kolmogorov n-widths

$$d_N(\mathcal{M}) := \inf_{V_N \subset H} \sup_{u \in \mathcal{M}} \inf_{v_N \in V_N} \|u - v_N\|$$



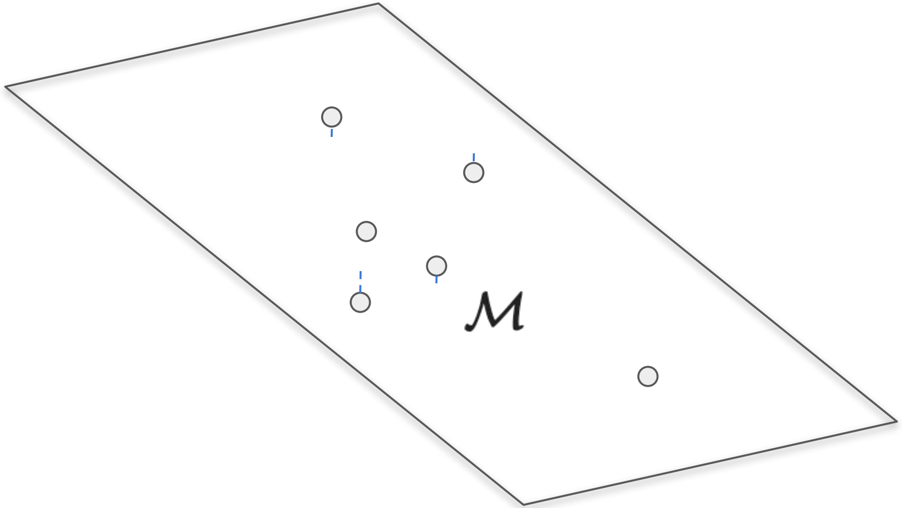
Kolmogorov n-widths

$$d_N(\mathcal{M}) := \inf_{V_N \subset H} \sup_{u \in \mathcal{M}} \inf_{v_N \in V_N} \|u - v_N\|$$



Kolmogorov n-widths

$$d_N(\mathcal{M}) := \inf_{V_N \subset H} \sup_{u \in \mathcal{M}} \inf_{v_N \in V_N} \|u - v_N\|$$

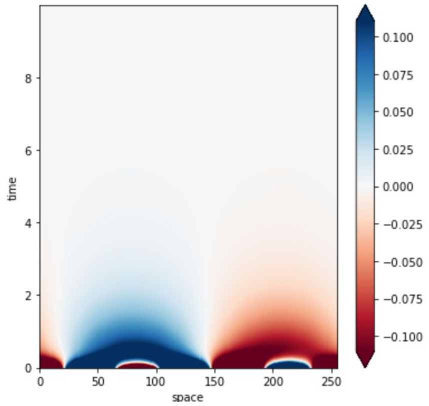


Kolmogorov n-widths and advection-dominated systems

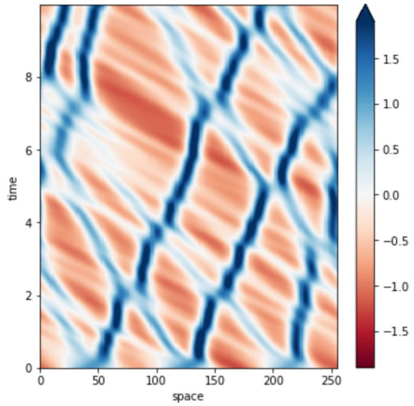
$$d_N(\mathcal{M}) := \inf_{V_N \subset H} \sup_{u \in \mathcal{M}} \inf_{v_N \in V_N} \|u - v_N\|$$

$$\mathcal{M} = [u_0, u(t_1), u(t_2), \ldots, u(t_{n-1}), u(T)]$$

$$\partial_t u = \partial_{xx} u$$



$$\partial_t u = -3\partial_x(u^2) - \partial_{xxx} u$$

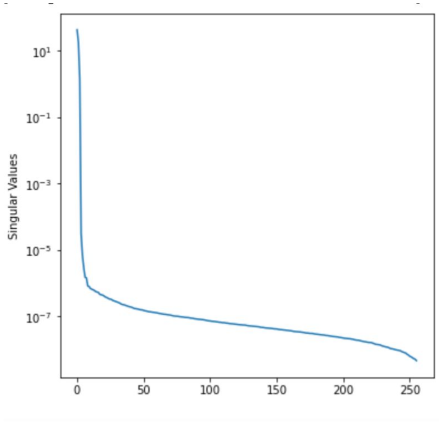


Kolmogorov n-widths and advection-dominated systems

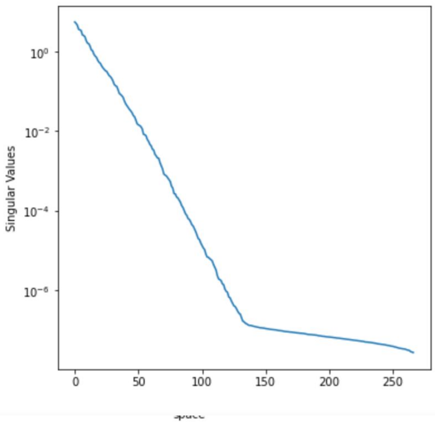
$$d_N(\mathcal{M}) := \inf_{V_N \subset H} \sup_{u \in \mathcal{M}} \inf_{v_N \in V_N} \|u - v_N\|$$

$$\mathcal{M} = [u_0, u(t_1), u(t_2), \ldots, u(t_{n-1}), u(T)]$$

$$\partial_t u = \partial_{xx} u$$



$$\partial_t u = -3\partial_x(u^2) - \partial_{xxx} u$$

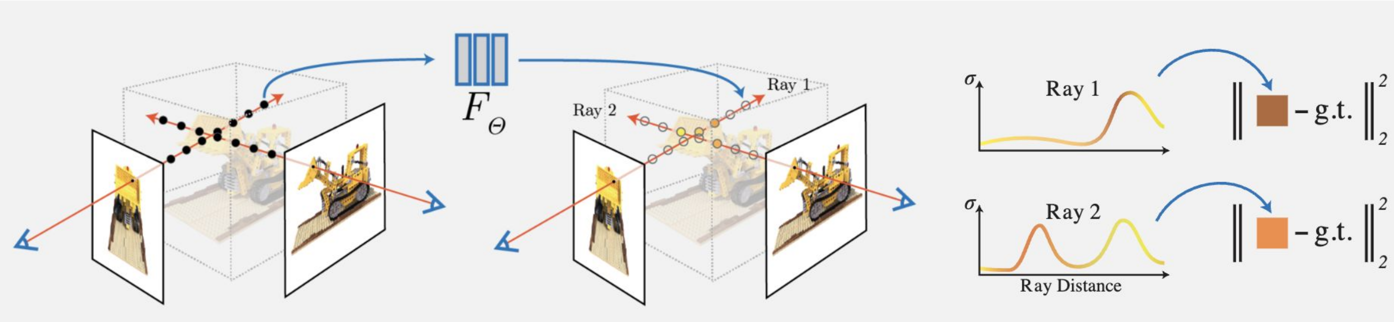


NeRF

Google Research

Using a parametric functional Anzats in the form of a Neural Networks

Evolution of parameters, aka, neural network weights



https://www.matthewtancik.com/nerf

Google Research

Machine Learning Methods: An Active Field

Hybrid Physics-ML

Mishra, 2018
Bar-Sinai et al., 2019
Kochkov et al., 2021
List et al., 2022
Bruno et al., 2021
Frezat et al., 2022
Dresdner et al., 2022

Operator Learning

Li et al., 2021
Tran et al., 2021
Fan et al., 2019
Li et al., 2020
Lu et al., 2021

Neural Ansatz

Raissi et al., 2019
Eivazi et al., 2021
E & Yu, 2018
Gao et al., 2022
Zang et al., 2020
de Avila Belbute-Peres et al.,2022
Bruna et al., 2022

Purely Learned Surrogates

Ronneberger et al., 2015
Wang et al., 2020
Sanchez-Gonzalez et al., 2020
Stachenfeld et al., 2022
Ayed et al., 2019

Machine Learning Methods: An Active Field

Hybrid Physics-ML
Mishra, 2018
Bar-Sinai et al., 2019
Kochkov et al., 2021
List et al., 2022
Bruno et al., 2021
Frezat et al., 2022
Dresdner et al., 2022

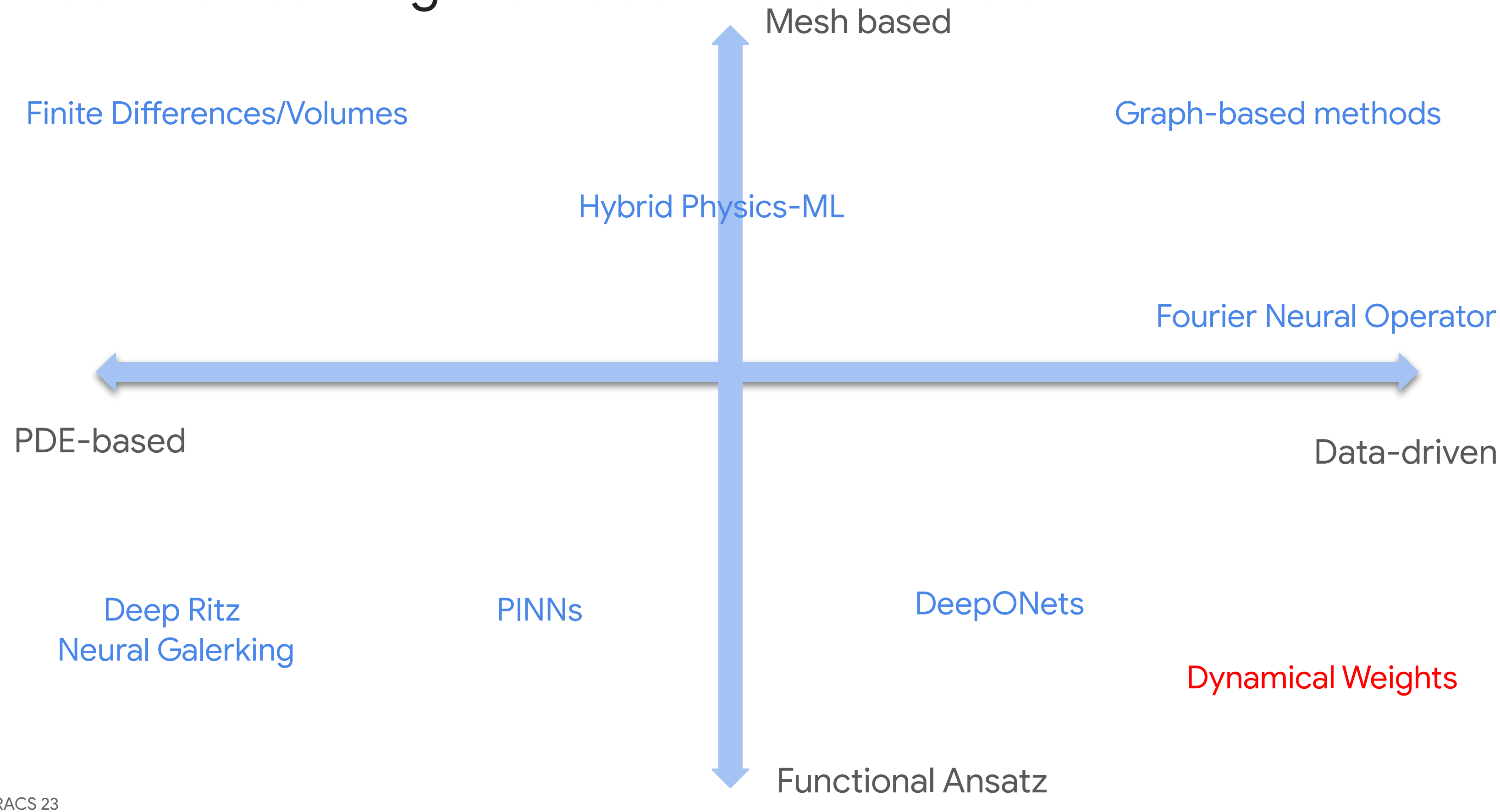
Operator Learning:
Li et al., 2021
Tran et al., 2021
Fan et al., 2019
Li et al., 2020
Lu et al., 2021

Neural Ansatz
Raissi et al., 2019
Eivazi et al., 2021
E & Yu, 2018
Gao et al., 2022
Zang et al., 2020
de Avila Belbute-Peres et al.,2022
Bruna et al., 2022

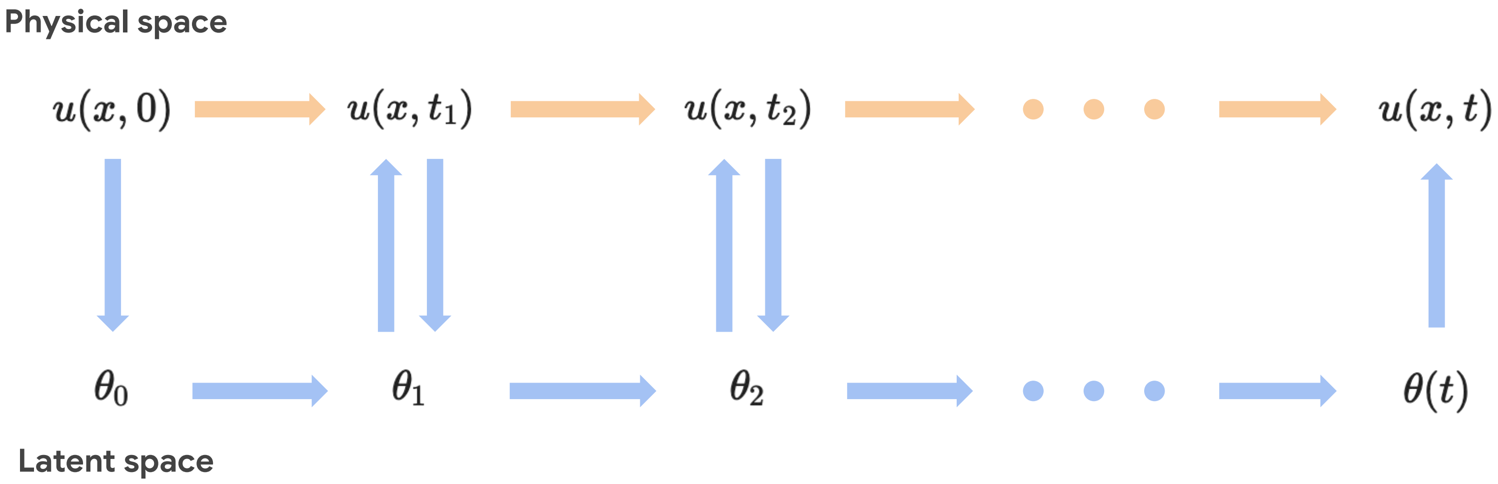
Dynamical Weights

Purely Learned Surrogates
Ronneberger et al., 2015
Wang et al., 2020
Sanchez-Gonzalez et al., 2020
Stachenfeld et al., 2022
Ayed et al., 2019

Machine Learning Methods: An Active Field

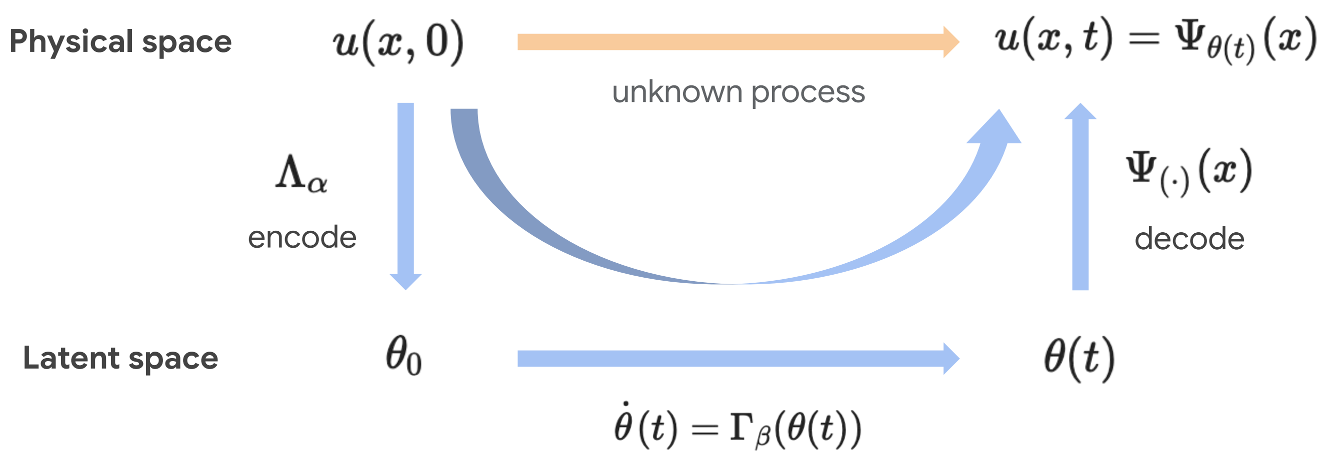


Encode-Process-Decode



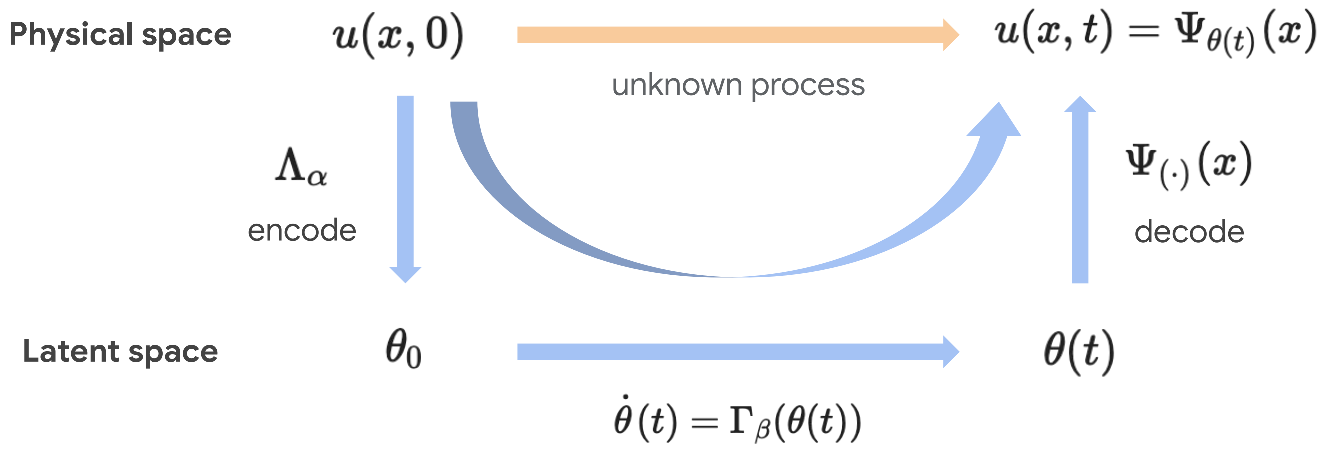
- Requires an uniform discretization in time
- Needs to go back to the ambient space at each time step
- Usually becomes unstable

Framework



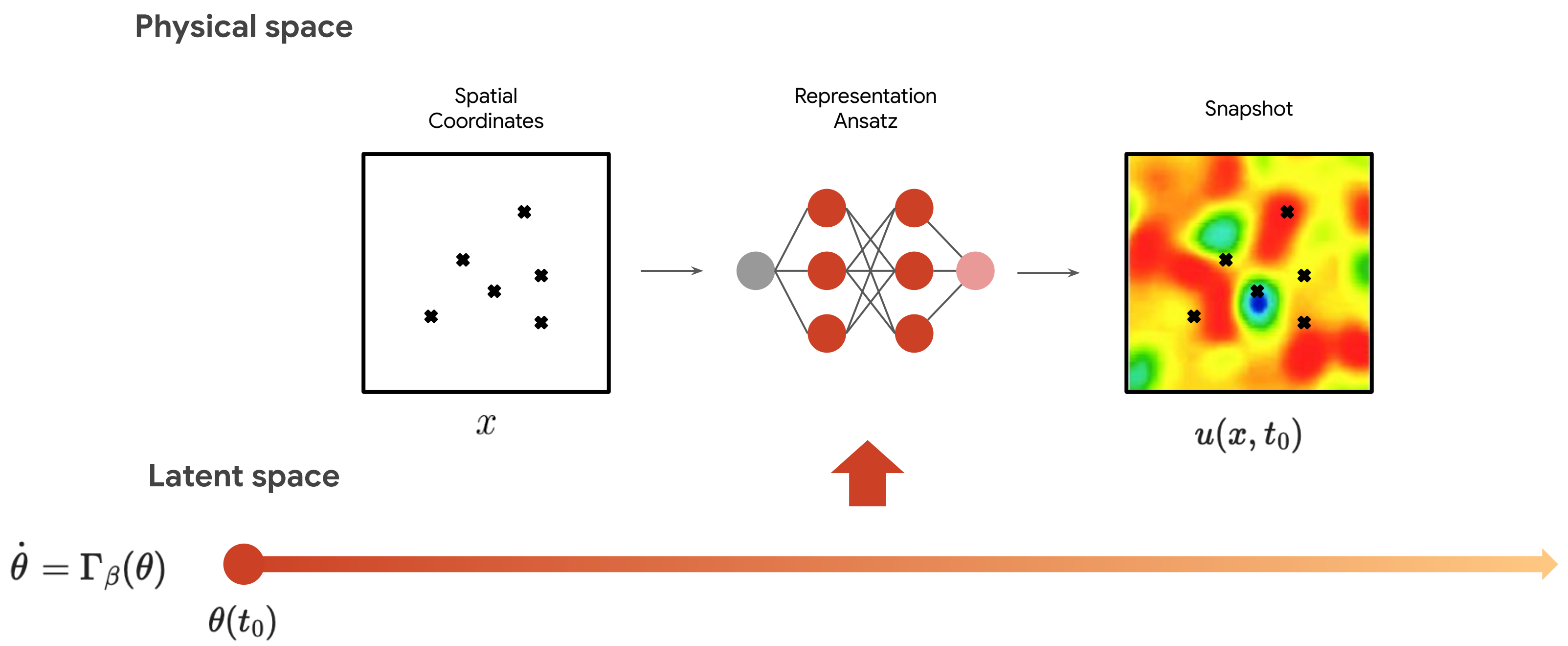
- Model components fully learned from data
- Continuous in both space and time
- Encode once and rollout for as long as needed

Framework



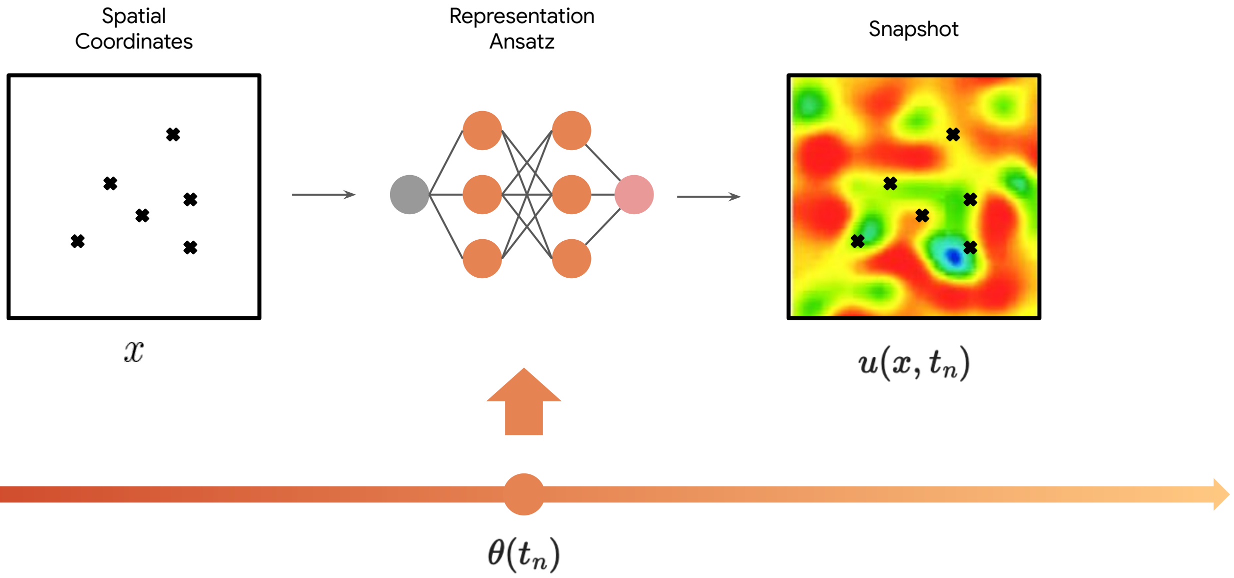
- Latent space - weights of a neural network
- Encoder training with smoothness inducing regularization
- Neural ODE model for latent dynamics

Network Weights as Time-Evolving Latent States



Network Weights as Time-Evolving Latent States

Physical space

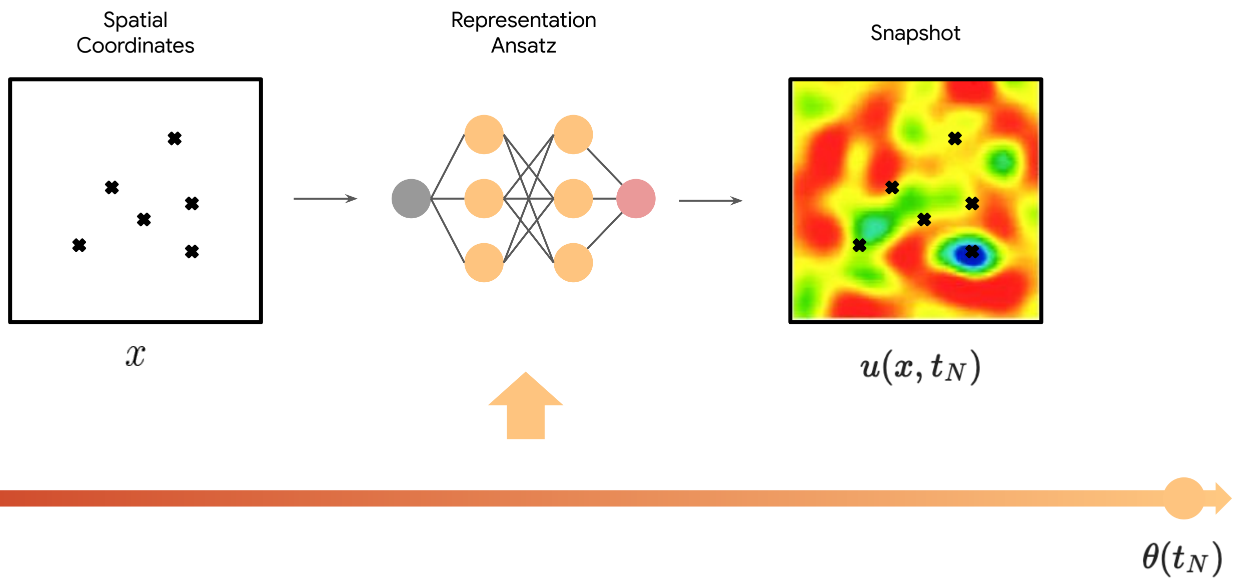


Latent space

$$\dot{\theta} = \Gamma_\beta(\theta)$$

Network Weights as Time-Evolving Latent States

Physical space



Latent space

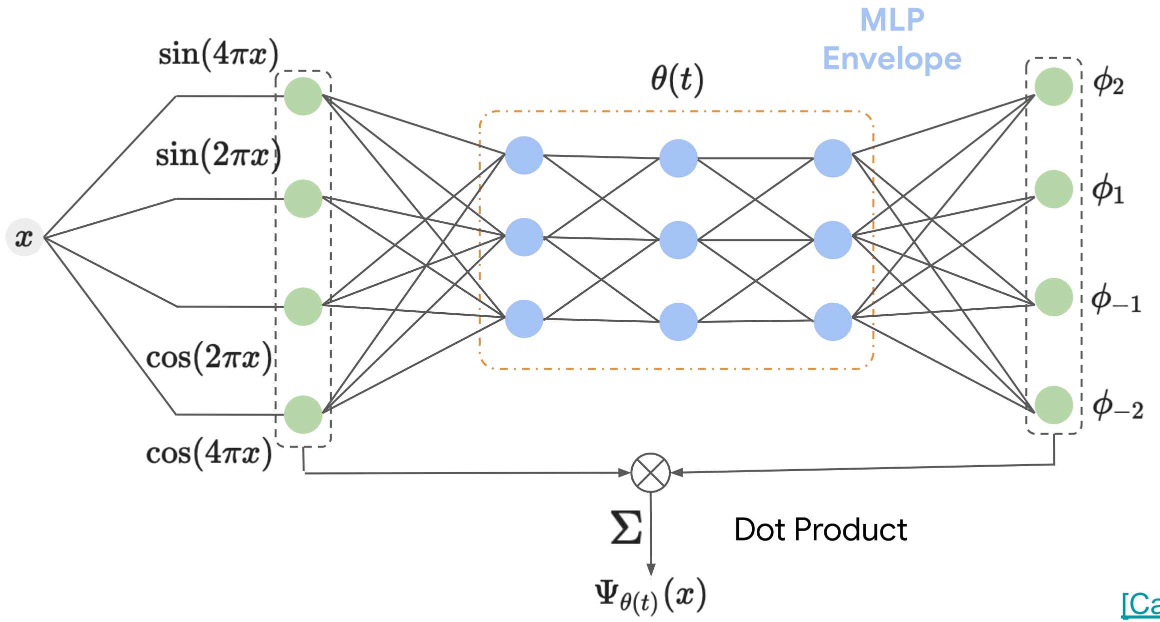
$$\dot{\theta} = \Gamma_\beta(\theta)$$

Representation Ansatz

- Tailored to the problem
- Slight over-parameterization for better sampling efficiency

Nonlinear Fourier Ansatz

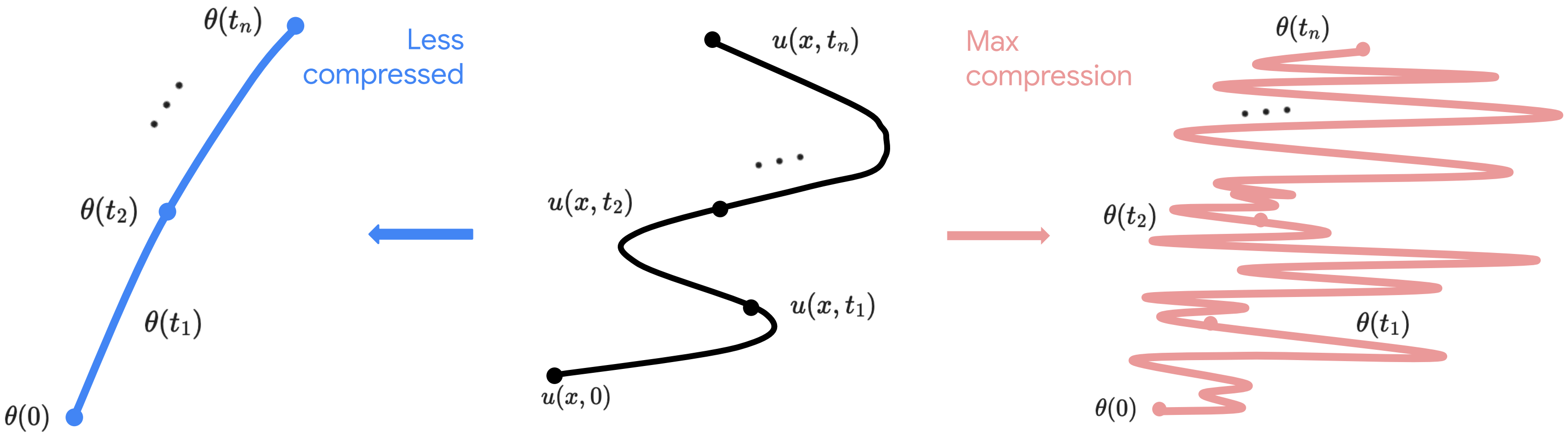
$$u(x,t) = \Psi_{\theta(t)}(x) = Re\left(\sum_{k=-K}^{K} \phi_k(x;\theta(t)) \exp\left(\frac{2i\pi kx}{L}\right)\right) \qquad K = \mathcal{O}(1) \qquad (\text{e.g.} = 3)$$



- Baked-in periodic boundary conditions
- Suitable for a wide class of advection-dominated systems

[Cai, Li, Liu]

Representation vs. Dynamics Efficiency



- Not pursue the most low-dimensional space and instead go for a better balance with dynamics efficiency
- Enforced during encoder training, decoupled from learning the latent dynamics

Encoder Training

Solution: train the encoder on the full trajectory

$$
\begin{tikzcd}[column sep=huge, row sep=large]
u(x,0) \arrow[d, shift right, "\Lambda_\alpha"'] \arrow[d, shift left, "\Psi_{(\cdot)}(x)"] & u(x,t_1) \arrow[d, shift left] & u(x,t) \\
\theta_0 \arrow[r, shift left] \arrow[r, shift right, "\dot{\theta}(t) = \Gamma_\beta(\theta(t))"'] & \theta(t_1) \arrow[u, shift left] \arrow[r, shift left] \arrow[r, shift right, "\dot{\theta}(t) = \Gamma_\beta(\theta(t))"'] & \theta(t) \arrow[u, shift left] \arrow[u, shift right]
\end{tikzcd}
$$

Diagram has to **commute**: following $\rightarrow$ or $\rightarrow$ should be the same

Google Research

Encoder Training

Solution: train the encoder on the full trajectory

$$\mathcal{L}(u) = \mathcal{L}_{\text{reconstruction}}(u) + \gamma \mathcal{L}_{\text{consistency}}(\Lambda(u))$$

Loss Function

snapshot reconstruction

$$\mathcal{L}_{\text{reconstruct}}(u) = \|u - \Psi(\Lambda(u))\|^2$$

encoding re-encoding

$$\mathcal{L}_{\text{consistency}}(\theta) = \|\theta - \Lambda(\Psi(\theta))\|^2$$

Ideal Method

Solution: train the encoder on the full trajectory

$$u = \Psi(\theta) \qquad \theta = \Lambda(u)$$

We don't know the dynamics!

$$\dot{u} = \nabla_\theta \Psi(\theta)\dot{\theta} \qquad \dot{\theta} = \nabla_u \Lambda(u)\boxed{\dot{u}}$$

$$\dot{u} = \nabla_\theta \Psi(\theta)\nabla_u \Lambda(u)\dot{u} \qquad \dot{\theta} = \nabla_u \Lambda(u)\nabla_\theta \Psi(\theta)\dot{\theta}$$

$$\downarrow \qquad \downarrow$$

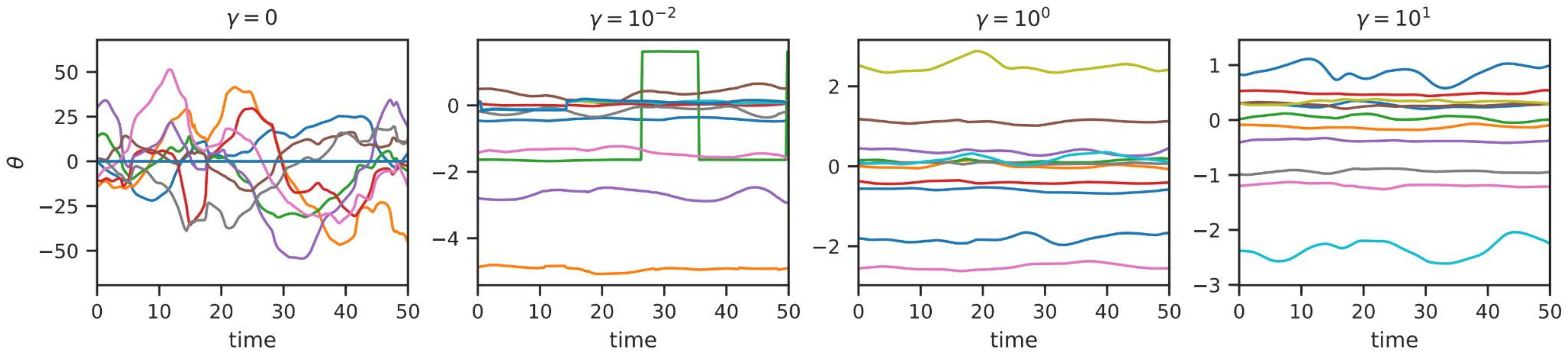
$$u(t) = \Psi(\Lambda(u(t))) \qquad \theta(t) = \Lambda(\Psi(\theta))$$

$$\mathcal{L}_{\text{reconstruct}}(u) = \|u - \Psi(\Lambda(u))\|^2 \qquad \mathcal{L}_{\text{consistency}}(\theta) = \|\theta - \Lambda(\Psi(\theta))\|^2$$

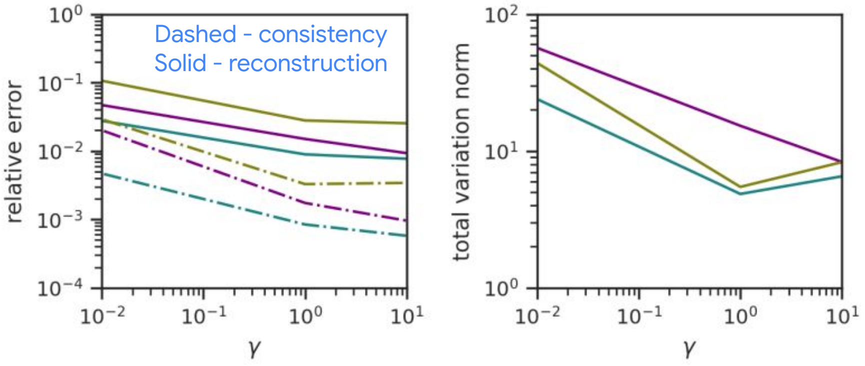
Smooth-inducing Regularization

Results shown for
Kuramoto-Sivashinsky (KS) system

Latent space trajectories at various regularization strengths:



Helps **reducing reconstruction error** while **inducing smoothness**



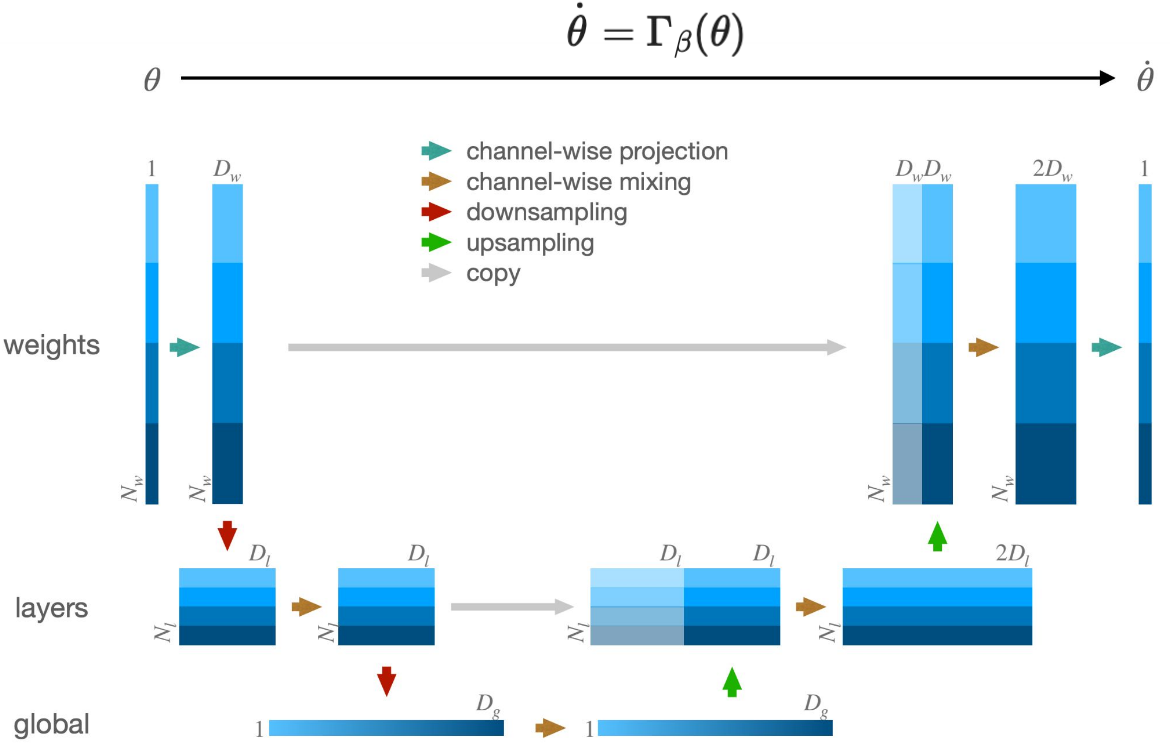
Colors = different learning rates

Learning Latent Dynamics

$$\mathcal{L}_{\mathrm{ODE}}(\beta) = \sum_n \left\| \theta_{t_n} - \tilde{\theta}_{t_n} \right\|^2,$$

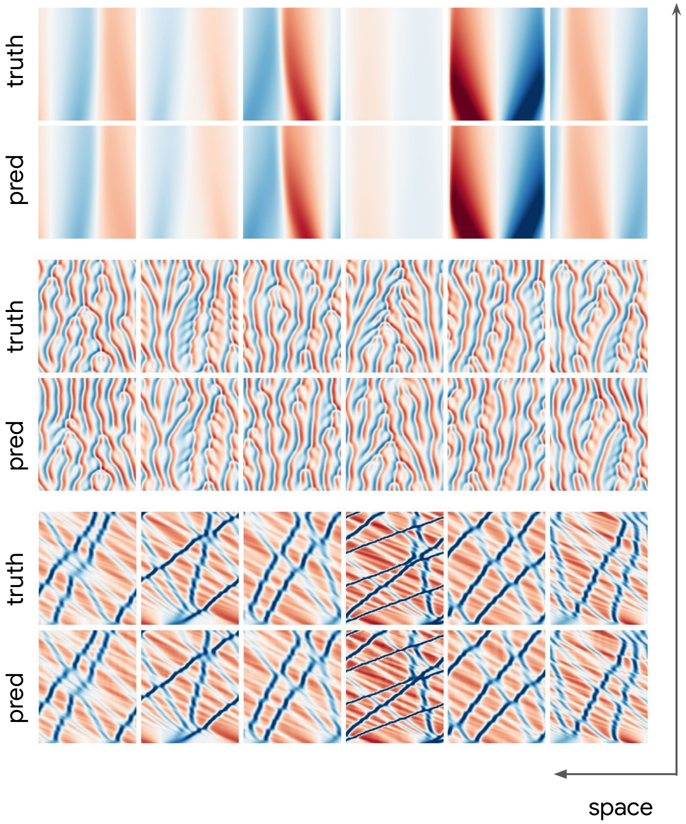
$$\tilde{\theta}_{t_n} = \theta_{t_0} + \int_{t_0}^{t_n} \Gamma_\beta(\theta(t))\, dt,$$

Multi-step MSE loss
(frozen encoder)



Google Research

Sample Rollouts (different ICs)



Viscous Burgers (VB)

[Dissipative]

Kuramoto-Sivashinsky (KS)

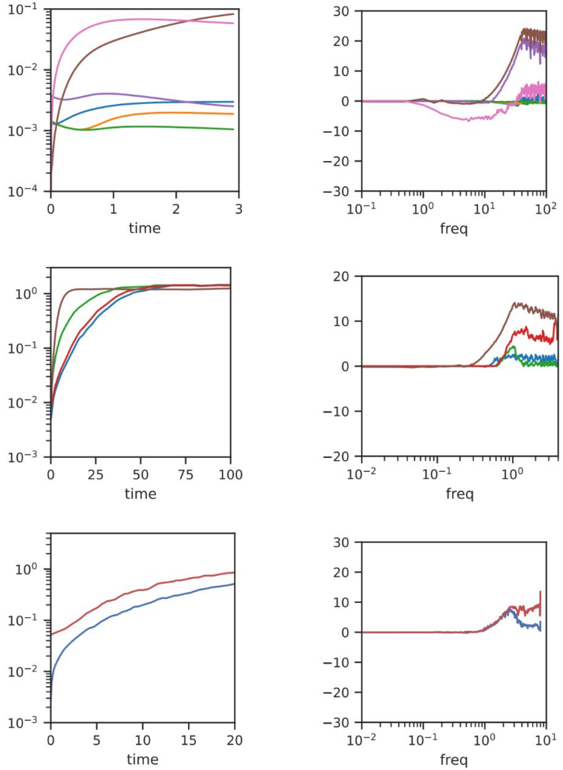
[Chaotic]

Korteweg-De Vries (KdV)

[Wave Dispersion]

Relative RMSE vs lead time

Log Energy Ratio against ground truth



Legend Ours-NFA: — NG-Lo: — NG-Hi: — Ours-NDV: — DeepONet: — FNO: —; DMD: —

Efficient Inference

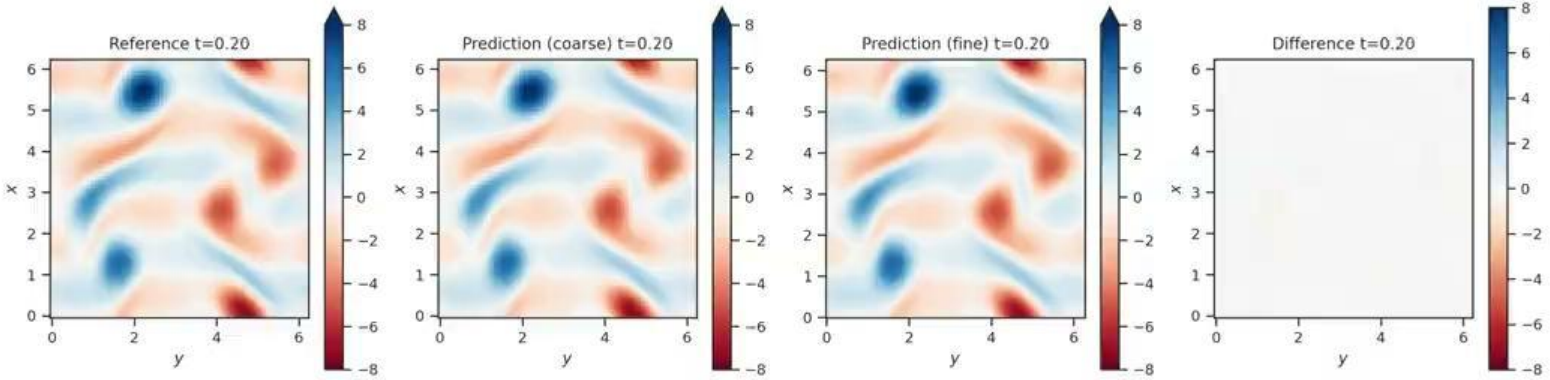
WCT = wall clock time; NFE = number of function evaluations

	VB				KS				KdV			
	WCT		NFE		WCT		NFE		WCT		NFE	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
Solver	8.4	0.5	-	-	5.2	0.3	-	-	1041	13.1	-	-
Ours-NFA	3.8	0.9	16.0	4.1	1.7	2.2	5.3	0.2	52.1	18.4	164.7	40.3
Ours-NDV	-	-	-	-	1.1	0.6	7.1	2.7	18.0	5.4	71.4	19.5
NG-Lo	30.2	8.5	27.5	10.8	167.9	21.4	114.7	13.3	-	-	-	-
NG-Hi	49.6	7.7	27.5	10.8	319.4	33.5	111.7	11.7	35670	29035	11651	9502
FNO	18.1	14.5	100.0	0.0	0.4	0.4	5.0	0.0	-	-	-	-
DMD	2.4	0.4	100.0	0.0	-	-	-	-	-	-	-	-

Solver: pseudo-spectral w/ same resolution;
NFA: nonlinear Fourier ansatz
NDV: neural decoder variant
NG-Hi/Lo: high/low sampling Neural Galerkin
FNO: Fourier neural operator
DMD: dynamic mode decomposition

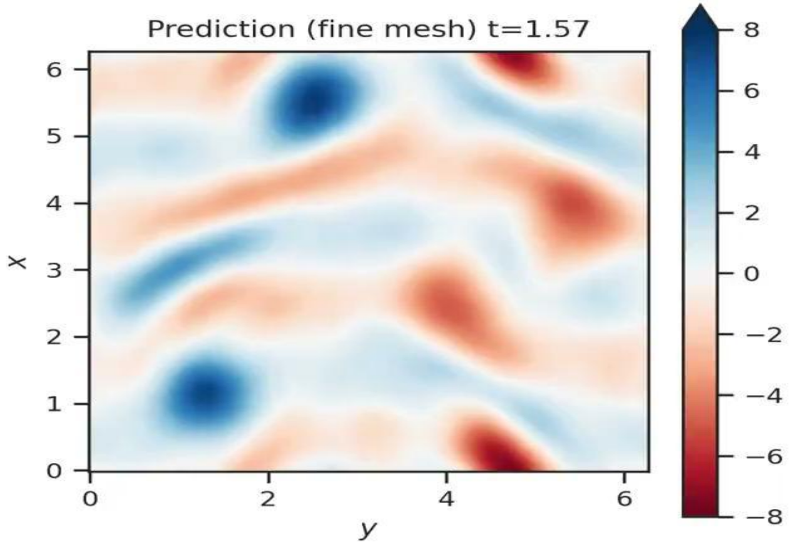
low NFE counts by adaptive-step integrator because we have smoother latent trajectories!

More Challenging Example: Kolmogorov Flow



Google Research

Indefinite Stability in Rollouts



Conclusions

More details available in the preprint (arXiv:2301.10391), and

github repository (https://github.com/google-research/swirl-dynamics)

- Correct physically pertinent ansatz for high-accuracy compression and good sampling efficiency
- Smooth latent space trajectories via consistency regularization
- Data-driven learning of the latent-space dynamics that is long-term stable
- Efficient inference from smooth trajectories

Block 2: Generative Modeling

Classical Sampling Techniques

Objective: Sample from a given target distribution $p(x)$

Main Idea: Sample from an **easy distribution** and transform the sample to the **target distribution**

Classical Methods:

- Inverse Transform Sampler
- Rejection Sampler
- Langevin Dynamics

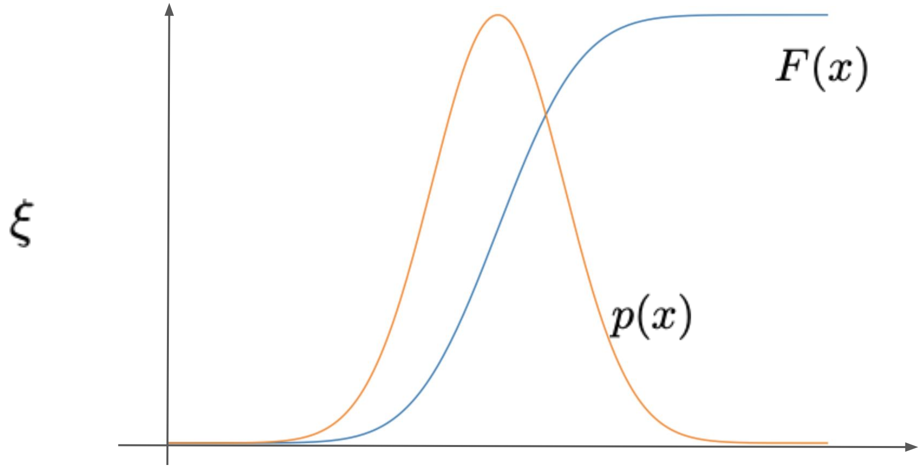
Machine Learning Methods

- Generative Adversarial Networks
- Variational Autoencoders
- Normalizing Flows
- Diffusion Models

Inverse Transformation

$$X \sim p(x) \mapsto F(x) = \int_{-\infty}^{x} p(x)dx \qquad \text{then} \qquad F(X) \sim \mathcal{U}[0,1]$$

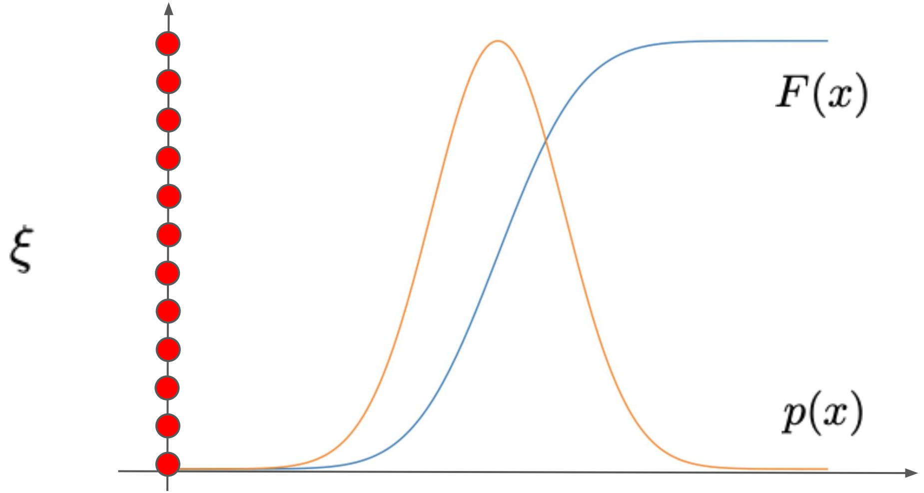
$$\xi \sim \mathcal{U}[0,1] \qquad \text{then} \qquad T(\xi) = F^{-1}(\xi) \sim p(x)$$



Inverse Transformation

$$X \sim p(x) \mapsto F(x) = \int_{-\infty}^{x} p(x)dx \qquad \text{then} \qquad F(X) \sim \mathcal{U}[0,1]$$

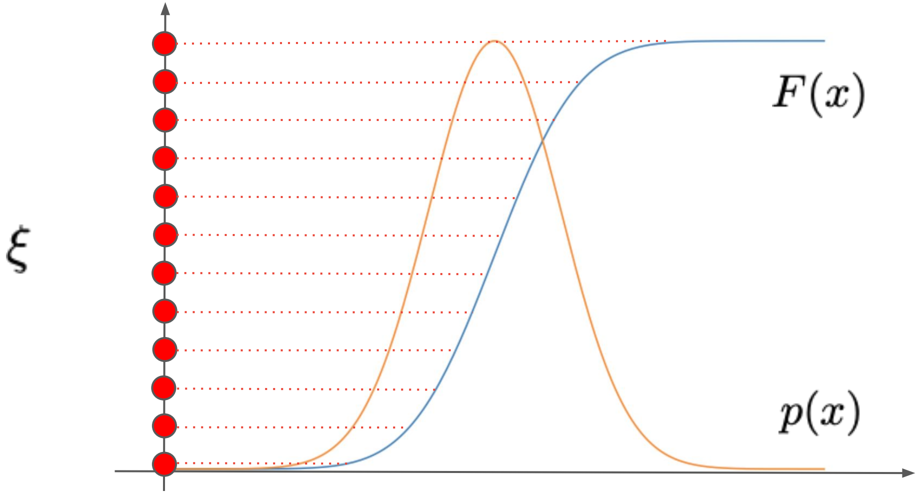
$$\xi \sim \mathcal{U}[0,1] \qquad \text{then} \qquad T(\xi) = F^{-1}(\xi) \sim p(x)$$



Inverse Transformation

$$X \sim p(x) \mapsto F(x) = \int_{-\infty}^{x} p(x)dx \qquad \text{then} \qquad F(X) \sim \mathcal{U}[0,1]$$

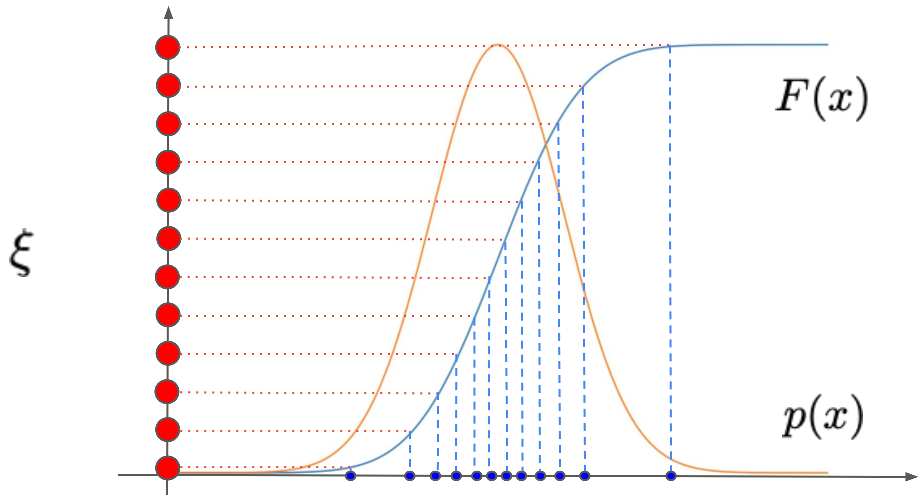
$$\xi \sim \mathcal{U}[0,1] \qquad \text{then} \qquad T(\xi) = F^{-1}(\xi) \sim p(x)$$



Inverse Transformation

$$X \sim p(x) \mapsto F(x) = \int_{-\infty}^{x} p(x)dx \qquad \text{then} \qquad F(X) \sim \mathcal{U}[0,1]$$

$$\xi \sim \mathcal{U}[0,1] \qquad \text{then} \qquad T(\xi) = F^{-1}(\xi) \sim p(x)$$



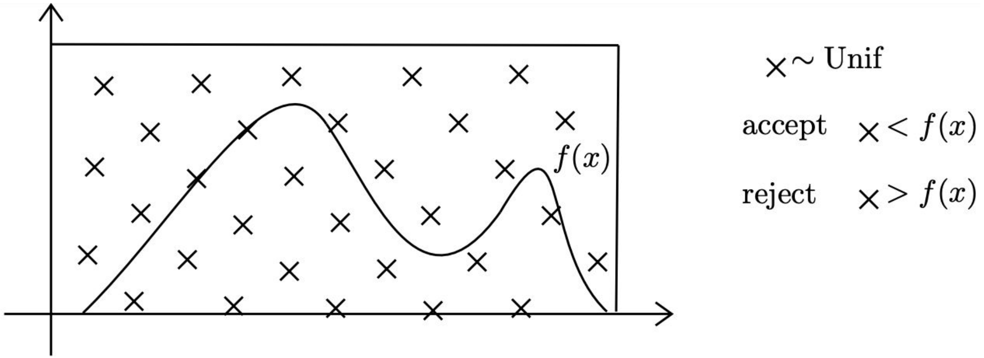
Rejection Sampling

Beyond 1D

Main Idea: Throw darts to a canvas

Accept the ones inside

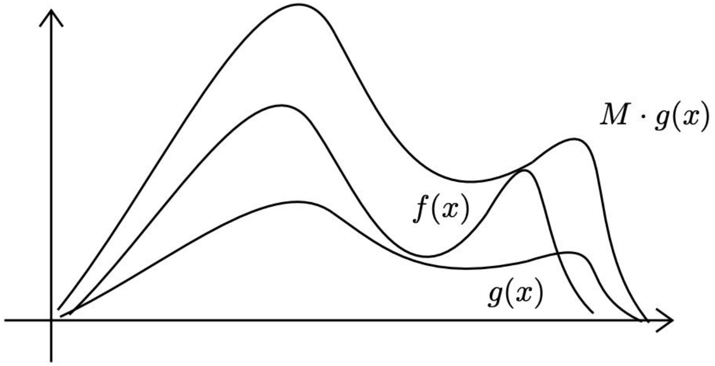
Reject the rest



Main Idea: Draw samples from $g(x)$

Accept $\dfrac{f(x)}{Mg(x)}$

Reject $\dfrac{Mg(x) - f(x)}{Mg(x)}$



Langevin Dynamics

Main Idea: Use an SDE, and look for the steady density.

Instead of proposal density work directly with the target density

$$dX_t = \underbrace{-\nabla V(X_t)dt}_{\text{Drift term}} + \underbrace{\sqrt{2}dW_t}_{\text{Noise term}}$$

Noisy version of flow equation

$$\dot{x} = -\nabla V(x)$$

Langevin Dynamics

Main Idea: Compute stationary (or steady) measure (depends only on the potential!)

$$X_t \sim p(x,t)$$

$$p(x,t) \to p_\infty(x) \qquad \text{as} \qquad t \to \infty$$

Euler-Maruyama

Main Idea: Simplest discretization in time

$$dX_t = \nabla \log p(X)dt + \sqrt{2}dW_t$$

$$\Downarrow$$

$$X_{n+1} = X_n + \nabla \log p(X_n)\Delta t + \sqrt{2\Delta t}\xi_n$$

$$\xi_n \sim \mathcal{N}(0, 1)$$

Langevin Dynamics Monte Carlo

Main Idea: Use the Langevin Dynamics as a proposal distribution and add a rejection step.

$$\tilde{X}_{n+1} = X_n + \nabla \log p(X_n)\Delta t + \sqrt{2\Delta t}\xi_n$$

$$\alpha = \min\left\{1, \frac{p(\tilde{X}_{n+1})q(X_n|\tilde{X}_{n+1})}{p(X_n)q(\tilde{X}_{n+1}|X_n)}\right\} \qquad q(x'|x) = \exp\left(-\frac{1}{4\Delta}\|x' - x - \Delta t\nabla \log p(x)\|^2\right)$$

$$X_{n+1} = \begin{cases} \tilde{X}_{n+1}, & \text{if } U < \alpha, \\ X_n, & \text{if } U \geq \alpha. \end{cases} \qquad U \sim \mathcal{U}(0,1)$$

Block 3: Diffusion models and Applications to Scientific Computing

Diffusion models

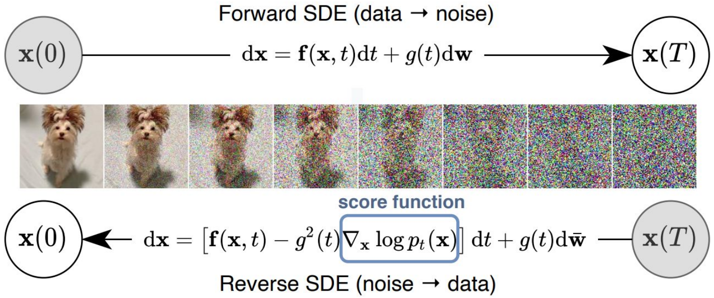
Main Idea: Sample by introducing noise/denoising

Sample from noise and transform the sample

Why? No access to the score function! **Only samples**

A cute corgi lives in a house made out of sushi.

[Saharia et al, 2020]



[Song et al, 2020]

3 Descriptions

Probability flow

SDE ODE

Feynman-Kac

Method of characteristics

PDE

Google Research

3 Different approaches

There are 3 main approaches that focus on approximating a different quantity (usually via a neural network)

Score functions $\nabla_x \log p_t(x)$

Denoiser $D(x, \sigma) \rightarrow x_0, \qquad x = x_0 + \epsilon_\sigma$

Noise from Sample $\varepsilon(x, \sigma), \qquad \dfrac{x - \varepsilon(x, \sigma)}{\sigma^2} = x_0$

They are asymptotically equivalent!

Luo, C. Understanding Diffusion Models: A Unified Perspective, Arxiv 2208.11970

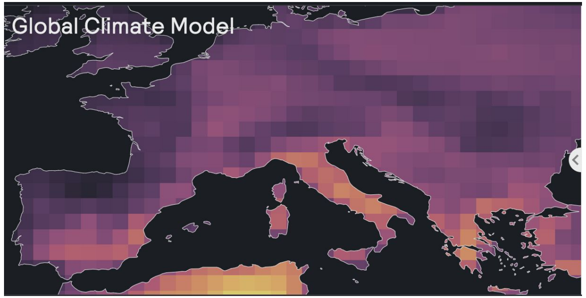
Application: Climate Downscaling

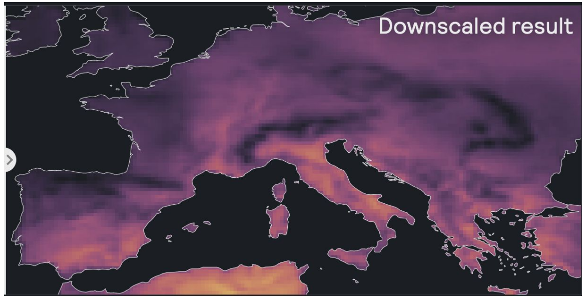
Motivation

Climate models: very coarse resolution ⟹ Biases due to the **lack of small** scale dynamics

Lack the level of **granularity** for local studies

E.g. Is the likelihood of extreme heat and wildfire in Marseille increasing?





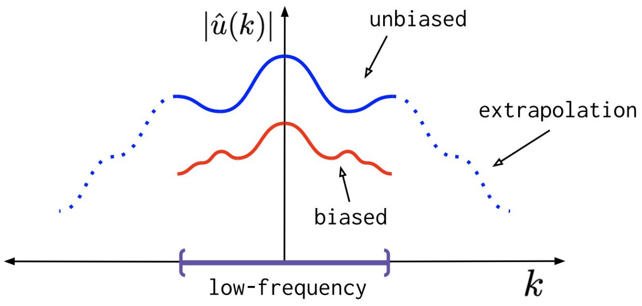
https://carbonplan.org/research/cmip6-downscaling-explainer

Why is it hard?

Solution:

Statistical Downscaling:

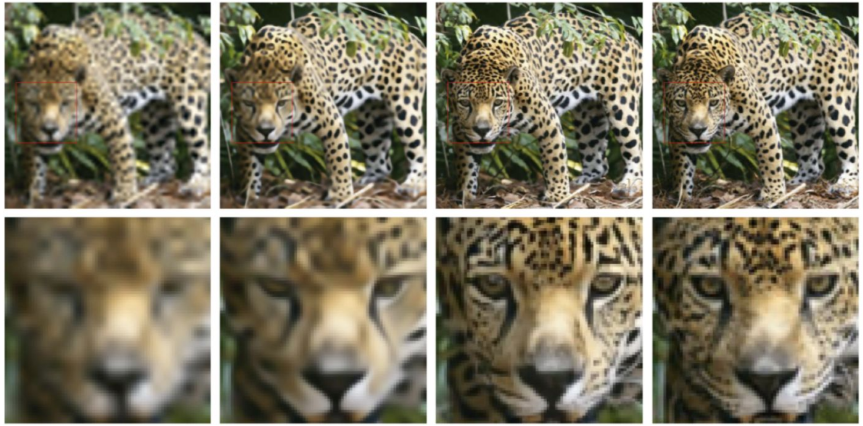
transform data from **low**-resolution to **high**-resolution



Two issues are entangled:

bias

super-resolution



Main difficulty:

lack of **paired data**

https://iterative-refinement.github.io/

2-step framework

Probabilistic framework:

"Conditional sampling from a high-quality prior".

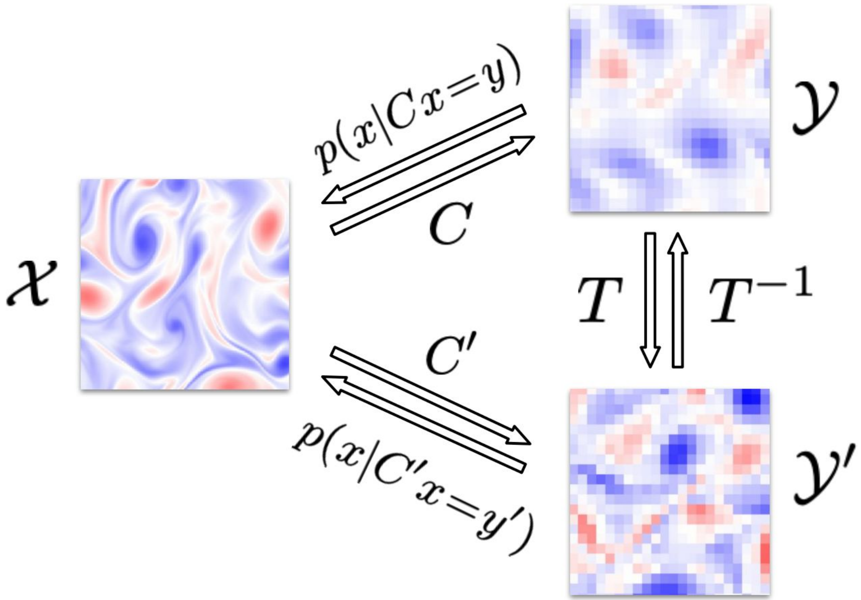
$$p(x|Cx=y)$$

But map C is **unknown**

Factorization 👉 $C = T^{-1} \circ C'$

Preserves statistical information

$$(T^{-1} \circ C')_\sharp \mu_X = \mu_Y$$



Diffusion model

Learn a prior $p(x)$ of the high-resolution data.

Why?

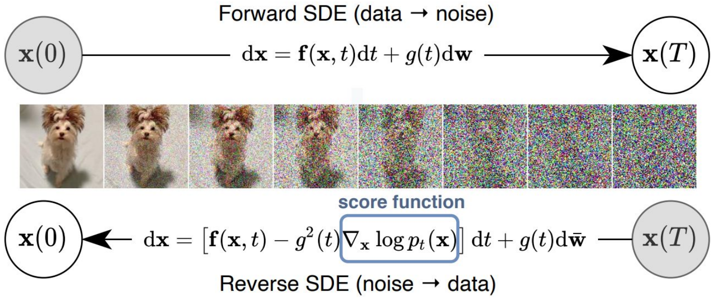
High-quality samples

High **coverage** of the distribution

On-demand sampling from the prior

A cute corgi lives in a house made out of sushi.

[Saharia et al, 2020]



[Song et al, 2020]

Diffusion model - conditional sampling

Two main approaches: train-time conditioning

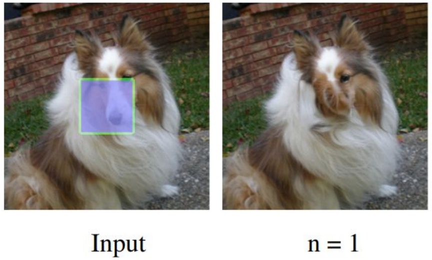
inference-time conditioning

Train a prior unconditionally $p(x)$

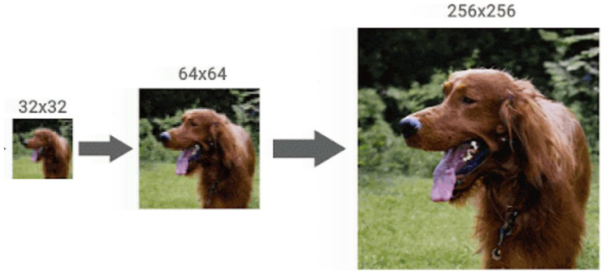
Then conditional sampling at inference (example inpainting)

$$p(x|C'x=y)$$

This procedure can be used for super-resolution



[Lugmayr et al. 2021]

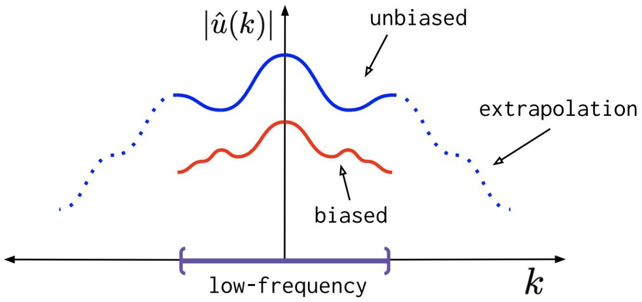


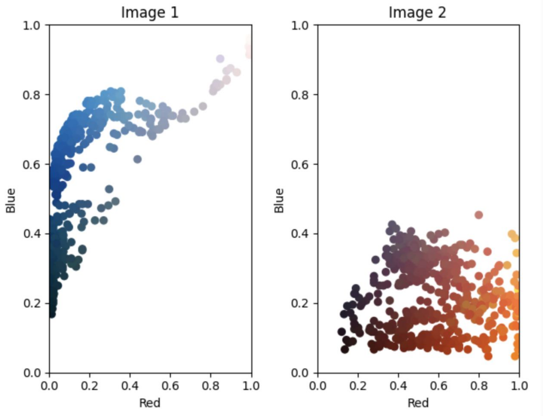
[Ho et al. 2021]

Debiasing - Optimal transport

There is a **bias** in the samples

super-resolving can only do Fourier **extrapolation**

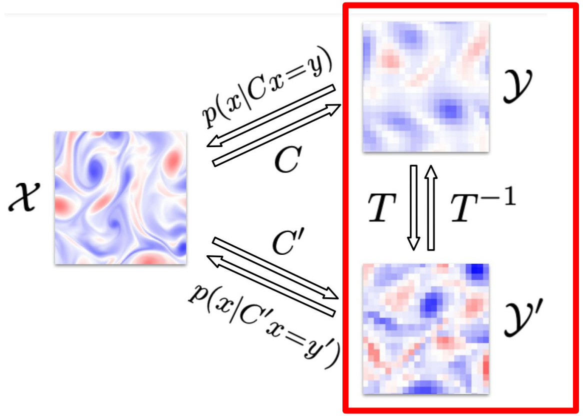




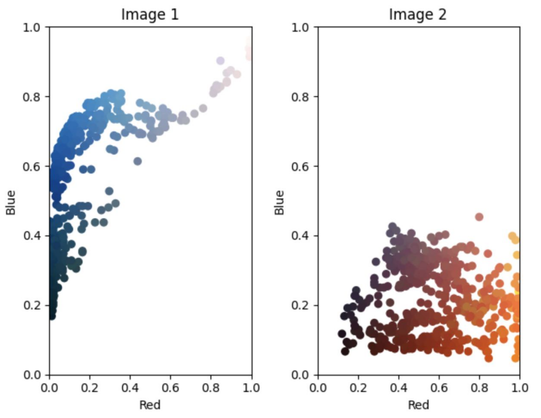
https://pythonot.github.io/

Debiasing - Optimal transport

How to fix the bias? Compute debiasing map



$$\min_T \left\{ \int c(y, T(y)) d\mu_Y(y) : T_\sharp \mu_Y = C'_\sharp \mu_X \right\}$$



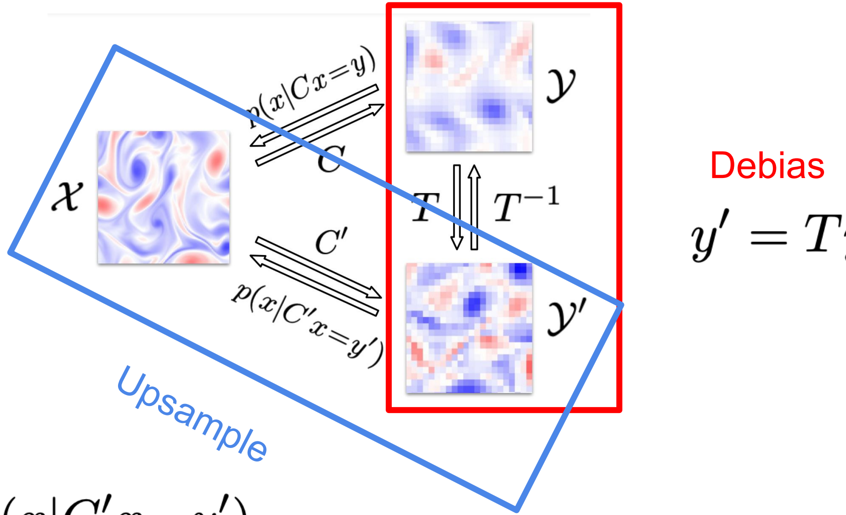
https://pythonot.github.io/

Downscaling = Debiasing $\Longrightarrow$ Upsample

Two sequential steps: Debias

Upsample

Goal: $p(x|Cx=y)$



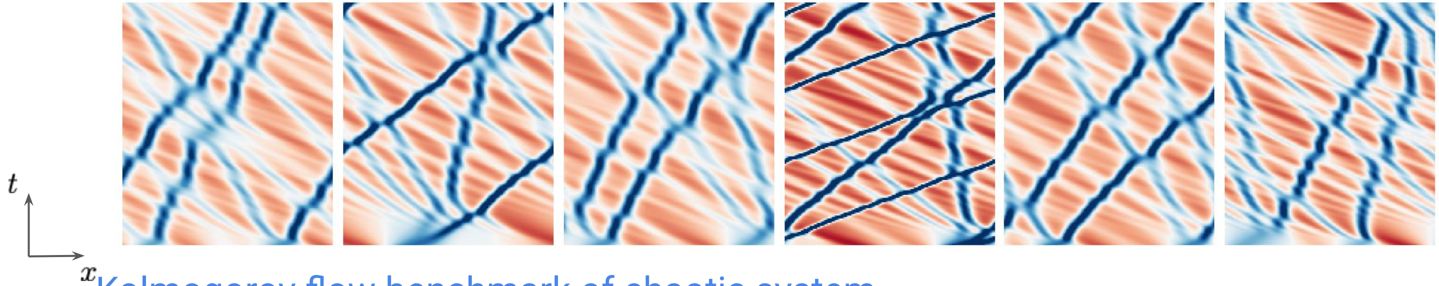
$y' = Ty$

$p(x|C'x=y')$

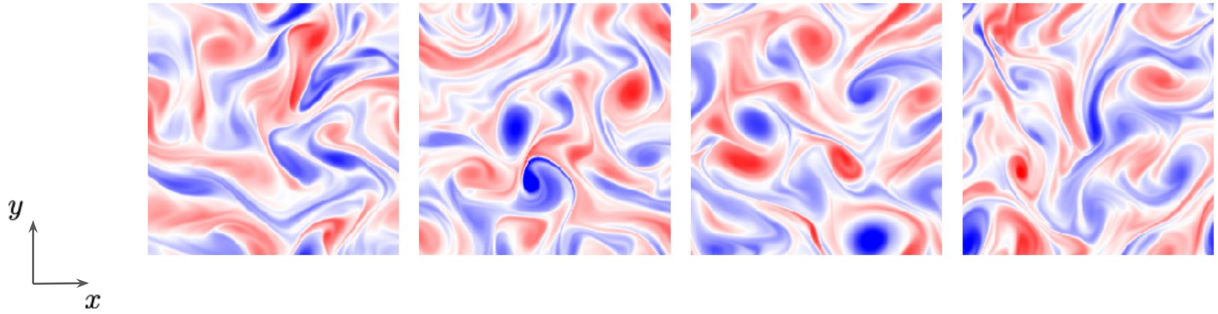
Google Research

Results

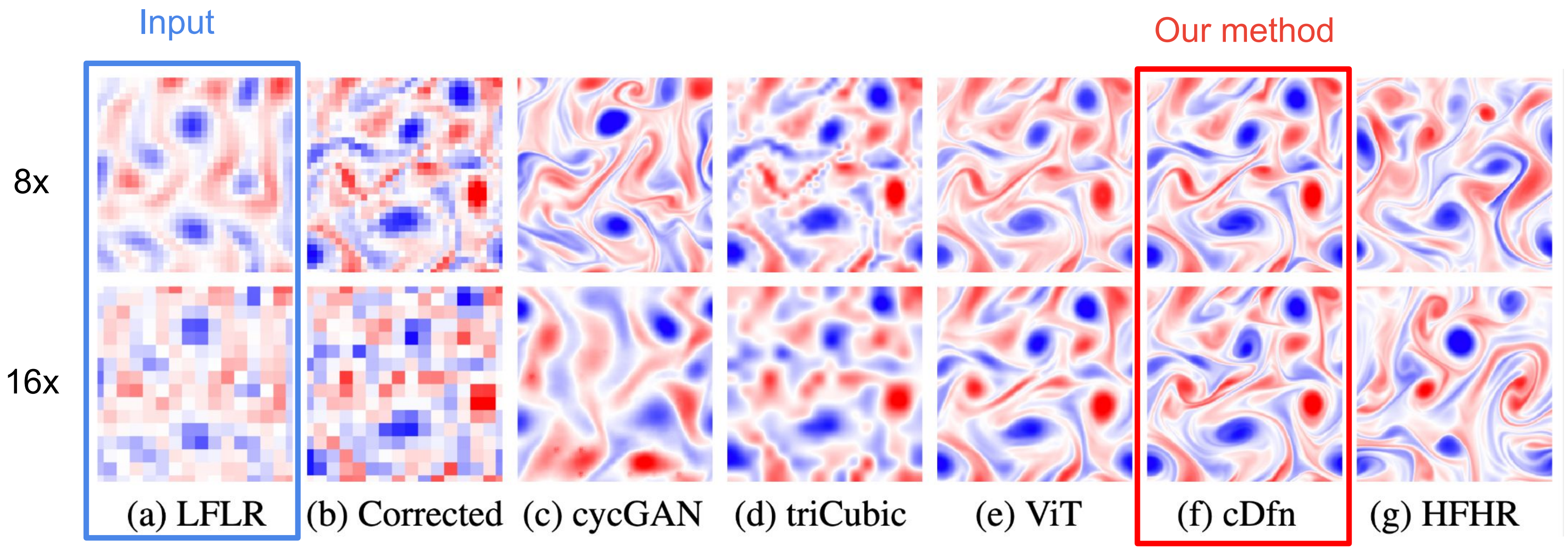
Data: Kuramoto-Sivashinsky system: simplest chaotic one-dimensional system.



Kolmogorov flow benchmark of chaotic system



Results


Input
Our method
8x
16x
(a) LFLR
(b) Corrected
(c) cycGAN
(d) triCubic
(e) ViT
(f) cDfn
(g) HFHR

Google Research

Results

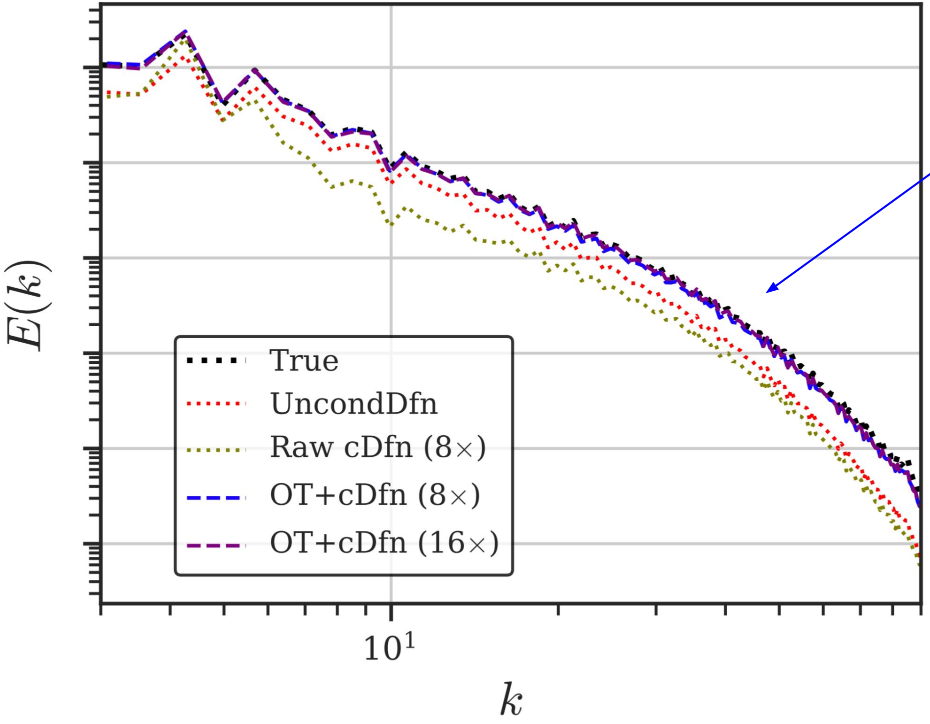
Energy Spectrum:

The **energy** at each Fourier mode / wavenumber

Provides a measure of the **spatial features** (up to translations) in the snapshot

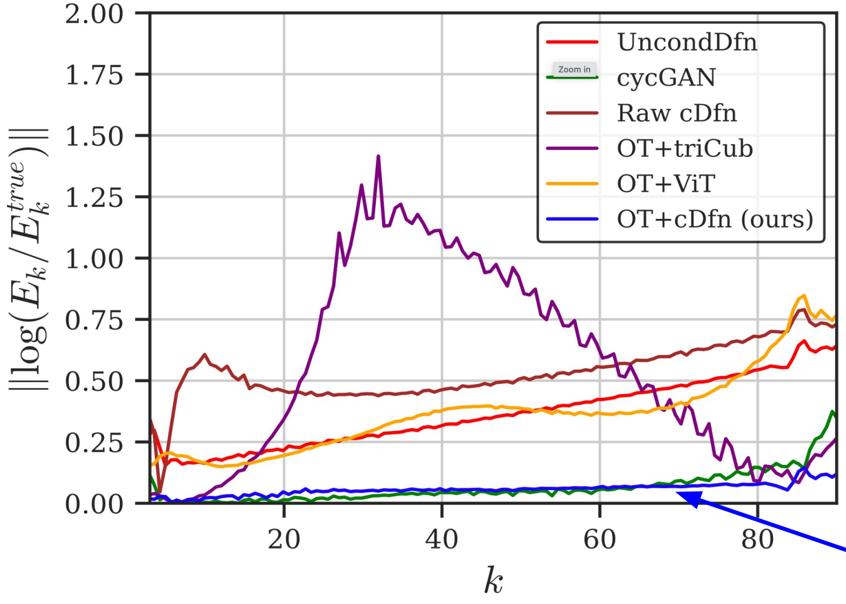
$$E(k) = \sum_{|\underline{k}|=k} |\hat{u}(\underline{k})|^2 = \sum_{|\underline{k}|=k} \left| \sum_i u(x_i) \exp(-j2\pi \underline{k} \cdot x_i / L) \right|^2$$

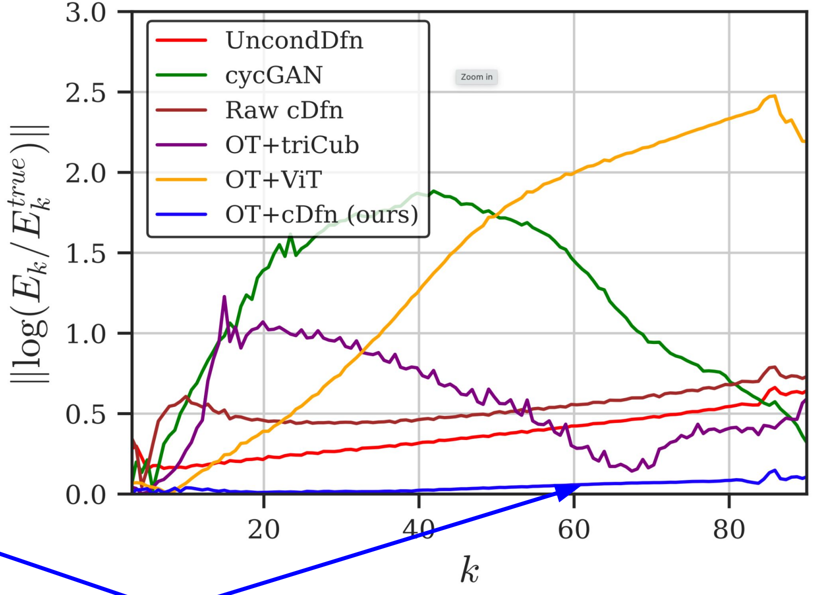
Results



Near perfect match to ground-truth

Results: Spectral Energy





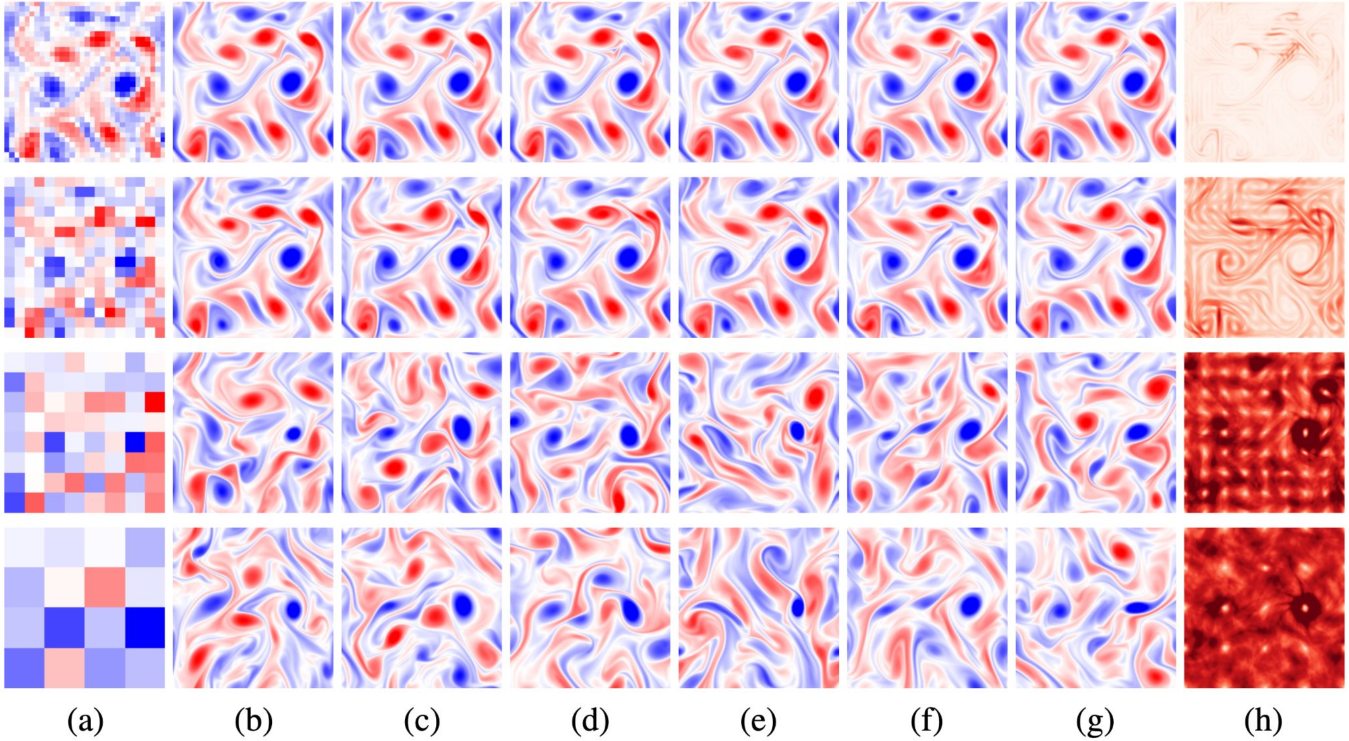
*Zero is a perfect match.

Our method has the the smallest error

Results: Spectral Energy

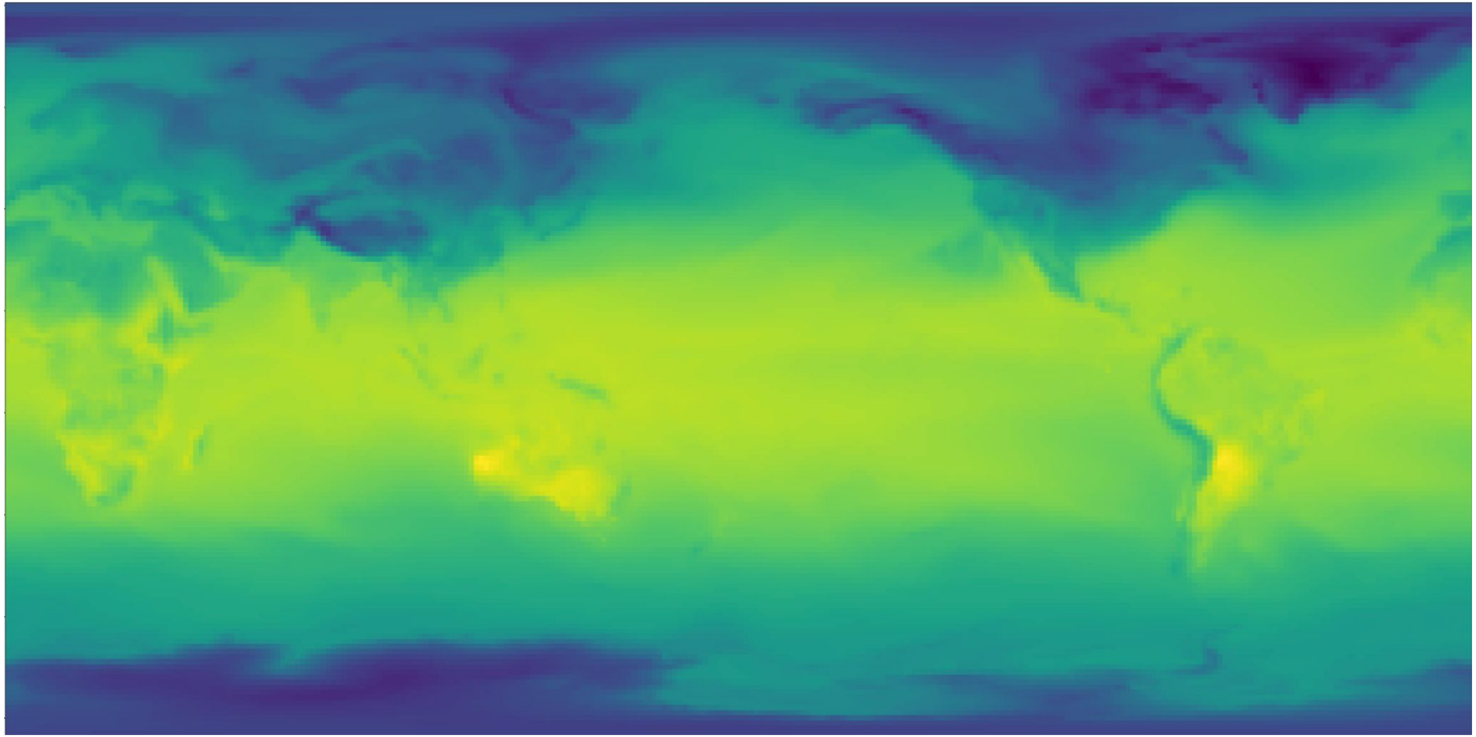
Metric	NS 8× downsample					NS 16× downsample				
	cyc GAN	Raw cDfn	OT+ triCub	OT+ ViT	OT+ cDfn	cyc GAN	Raw cDfn	OT+ triCub	OT+ ViT	OT+ cDfn
Constraint RMSE ↓	-	0.001	0	1.52	0.001	-	0.001	0	0.72	0.001
Sample Variability ↑	0	0.27	0	0	0.36	0	1.07	0	0	1.56
MELR (unweighted) ↓	0.08	0.79	0.52	0.38	**0.06**	1.14	0.54	0.55	1.38	**0.05**
MELR (weighted) ↓	0.05	0.37	0.06	0.18	**0.02**	0.28	0.30	0.13	0.09	**0.02**
KDE-KLD ↓	1.62	73.16	1.46	1.72	**1.40**	2.05	93.87	7.30	1.67	**0.83**

Results: Variability



Google Research

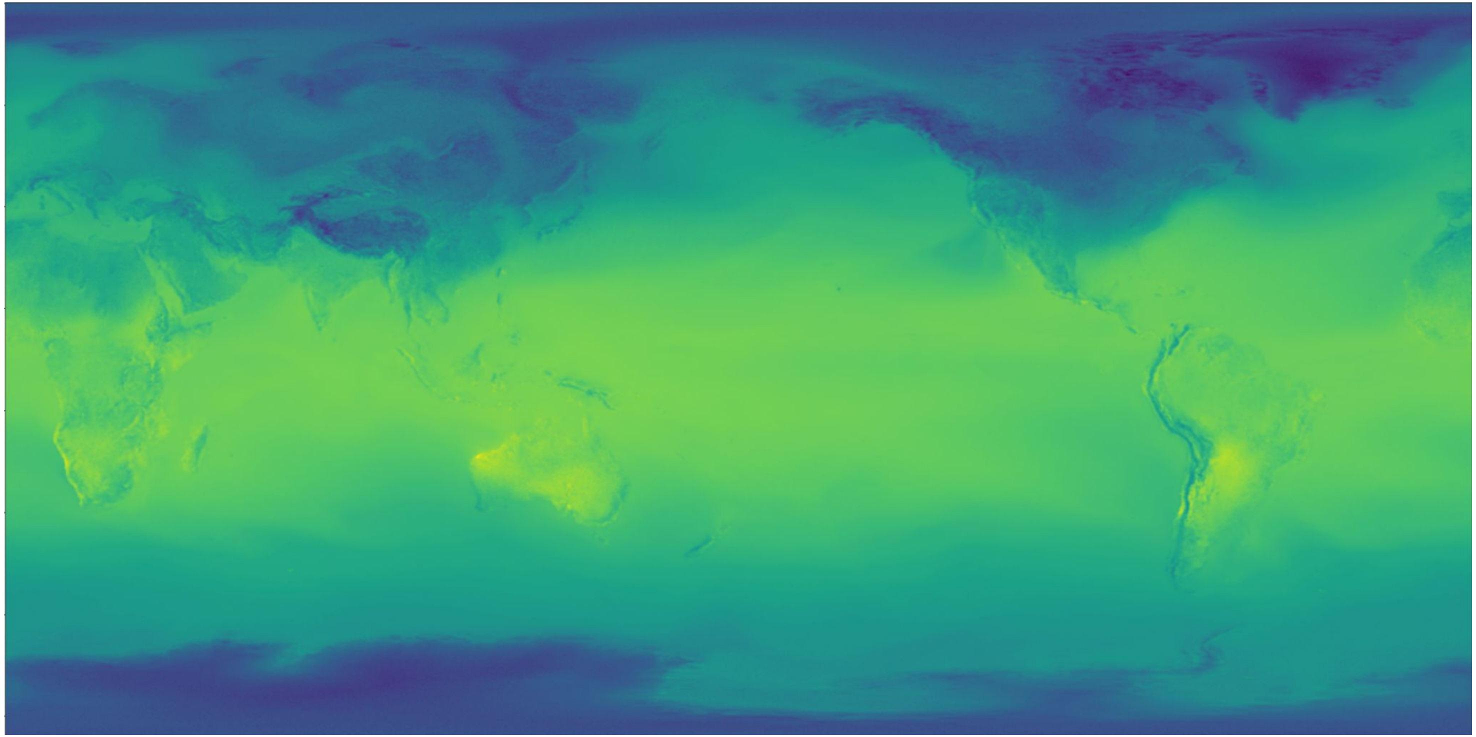
Climate Applications (WIP)



CESM simulation snapshot

Google Research

Climate Applications (WIP)



Unbiased and downscaled snapshot

Google Research

Conclusion

New probabilistic framework:

- **High-fidelity** samples with **correct biases**
- **Robust** debiasing based on a optimality conditions
- One prior rule-them all
  - We only train one diffusion model
  - **Inference-time conditioning**