Learning Spatio-temporal Dynamics via NNs

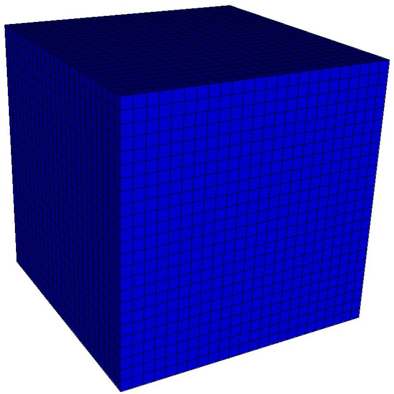
With application to data driven approaches for solving PDEs

STOR 891 Minji Kim

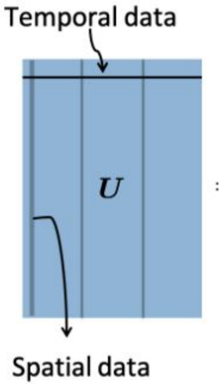
Figure: librom.net

Application: Data driven approaches for solving PDEs

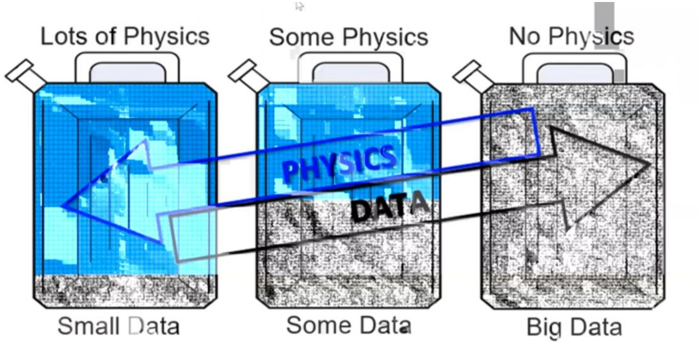
- Data-driven approaches for solving differential equations often share similar objectives to those we have seen in high-dimensional time series.
- One caveat for this talk: Nothing stochastic. But some stochastic extensions to SPDEs exist.



$u(x,t)$



Contents



- Goal: Data-driven approaches for physics simulation and ST dynamics
- As we adopt a data-driven approach, two things emerge: the **projection** into the latent space and the evolution based on the latent state **dynamics**
- Contents
  - Classic Physics Informed Approaches
  - From Linear **Projection** to Autoencoder
  - Learning Latent **Dynamics**
  - Recent Approaches

Classic (Physics-Informed) Approaches

Data driven approaches for solving PDEs

- PDEs can be written in ODE form by applying spatial discretization.

Data driven approaches for solving PDEs

- PDEs can be written in ODE form by applying spatial discretization.

E.g. Viscid 1D Burgers Equation : solve u(x,t) such that

$$\frac{\partial u}{\partial t} + u \cdot \frac{\partial u}{\partial x} = \frac{1}{\mu}\frac{\partial^2 u}{\partial x^2}, \; x \in \Omega = [0,2], \; t \in [0,T],$$

with some initial condition and boundary condition $u(0,t) = u(2,t) = 0.$

Spatially discretize the x into uniform grid points for fixed time t:

$$x_i = (i-1)\Delta x, \; \Delta x = 2/(n_x - 1), \; u_i = u(x_i, t; \mu), \; U = (u_2, \ldots, u_{n_x-2})^\top$$

$$\frac{dU}{dt} = -\frac{1}{\Delta x}(MU \odot U) + \frac{1}{\mu(\Delta x)^2} DU =: f(U), \; M = \begin{pmatrix} 1 & & & \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{pmatrix}, \; D = \begin{pmatrix} -2 & 1 & & \\ 1 & -2 & 1 & \\ & & \ddots & \\ & & 1 & -2 \end{pmatrix}$$

Basic Structure

- PDEs can be written in ODE form by applying spatial discretization.

- Governing Physics (ODE):

$$\frac{du}{dt} = f(u, t\,;\,\mu),\;\; u, f \in \mathbb{R}^N$$

- Full Order Model:

Backward time integrator

Solve time-discretization: $\text{for } t_1,\ \ldots t_n \text{ and } u_k := \hat{U}(t_k),\ u_k = u_{k-1} + \Delta t\, f(u_k,\, t_k;\, \mu)$

Physics - Informed

Linear Subspace Projection

- Governing Physics (ODE):

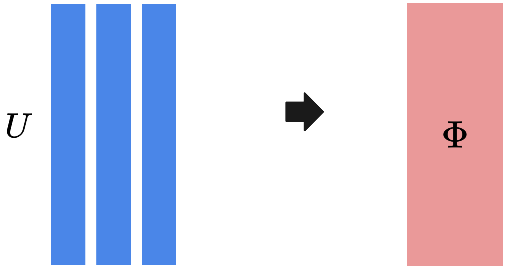
$$\frac{du}{dt} = f(u, t; \mu), \ u, f \in \mathbb{R}^N$$

- Reduced Order Model:

Linear subspace projection: project $u_{t_k} \in \mathbb{R}^N$ into a latent reduced space $\hat{u}_{t_k} \in \mathbb{R}^r$,

Collect data across several μ

Obtain projection matrix by SVD



$$u = u_{ref} + \Phi \hat{u}$$

Linear Subspace Projection

- Governing Physics (ODE):

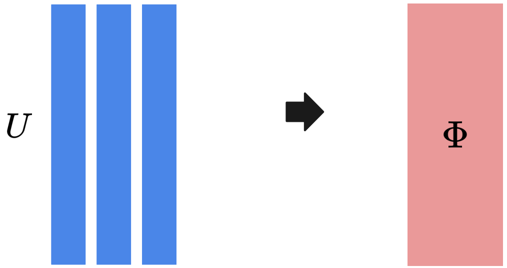
$$\frac{du}{dt} = f(u, t; \mu), \ u, f \in \mathbb{R}^N$$

- Reduced Order Model:

Linear subspace projection: project $u_k \in \mathbb{R}^N$ into a latent reduced space $\hat{u}_k \in \mathbb{R}^r$,

Collect data across several μ

Obtain projection matrix by SVD



Data-Driven

Physics - Informed

$$u = u_{ref} + \Phi \hat{u}$$

$$\frac{d\hat{u}}{dt} = \Phi^\top f(u_{ref} + \Phi \hat{u}, t; \mu)$$

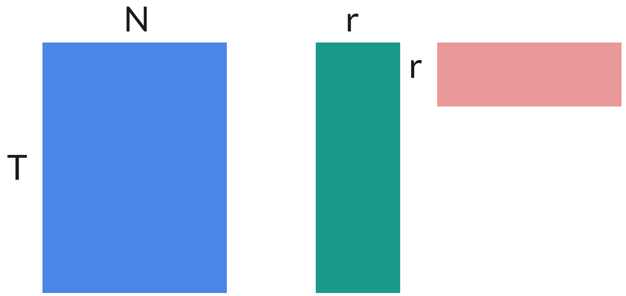
evolve within the latent space

$$\hat{u}_k = \hat{u}_{k-1} + \Delta t \, \Phi^\top f(u_{ref} + \Phi \hat{u}_k, t_k; \mu)$$

From Linear Projection to Autoencoder

Linear Factor Model versus AutoEncoder

$$\hat{X} = F\Lambda^{\top}$$



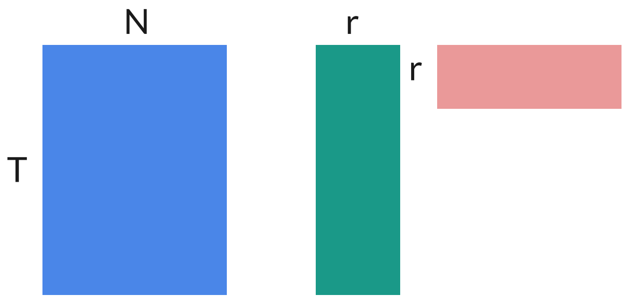
Latent Factor space

$$X_t \in \mathbb{R}^N \rightarrow \hat{f}_t \in \mathbb{R}^r \rightarrow \hat{f}_{t+1} = \Phi \hat{f}_t \rightarrow \hat{X}_{t+1} = \hat{\Lambda} \hat{f}_{t+1}$$

$$\hat{\Lambda} = \sqrt{N}\,\hat{U}_r \text{, where } \hat{U}_r \text{ is the first r eigenvectors of } \hat{\Sigma} = \frac{1}{T} X^{\top} X$$

Linear Factor Model versus AutoEncoder

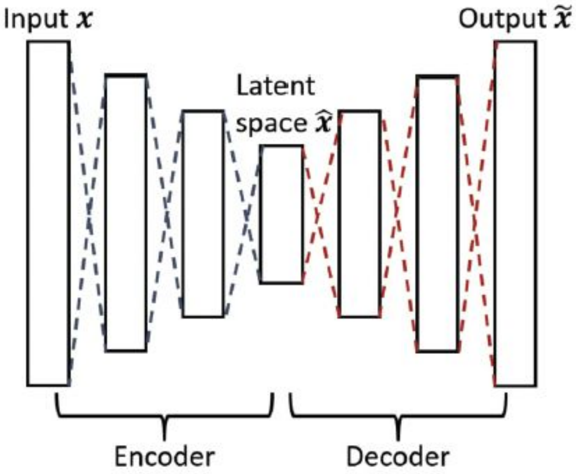
$$\hat{X} = F\Lambda^\top$$



Latent Factor space

$$X_t \in \mathbb{R}^N \to \hat{f}_t \in \mathbb{R}^r \to \hat{f}_{t+1} = \Phi \hat{f}_t \to \hat{X}_{t+1} = \hat{\Lambda} \hat{f}_{t+1}$$

$$\hat{\Lambda} = \sqrt{N}\,\hat{U}_r\,,\ \text{where}\ \hat{U}_r\ \text{is the first r eigenvectors of}\ \hat{\Sigma} = \frac{1}{T}X^\top X$$

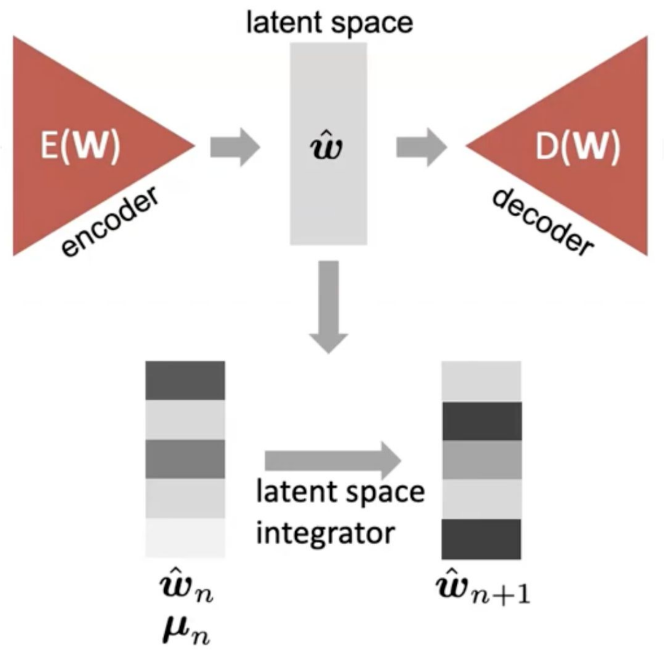


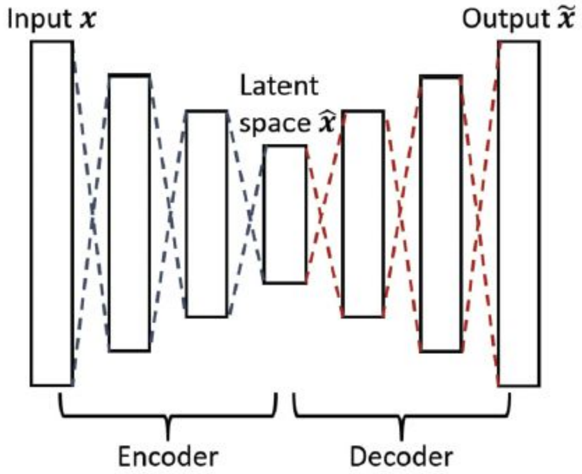
$$\mathbb{R}^r \ni \hat{x} = E_\theta(x)\,,\ D_\phi(\hat{x}) = \tilde{x} \in \mathbb{R}^N$$

Let $\hat{x}_t$ evolve within the latent space, and recover as $D(\hat{x}_{t+k})$

its (time-derivative) dynamics will be learned

Linear Factor Model versus AutoEncoder





$$\mathbb{R}^r \ni \hat{x} = E_\theta(x)\,,\; D_\phi(\hat{x}) = \tilde{x} \in \mathbb{R}^N$$

Let $\hat{x}_t$ evolve within the latent space, and recover as $D(\hat{x}_{t+k})$

its (time-derivative) dynamics will be learned

From Linear Model to Neural Network

- Learning the latent space manifold via autoencoder:

  Let $E_\theta : \mathbb{R}^N \to \mathbb{R}^r$ be an "Encoder" and $D_\phi : \mathbb{R}^r \to \mathbb{R}^N$ be a "Decoder" functions

Linear subspace projection:

$$u \approx \tilde{u} = u_{ref} + \Phi\hat{u}, \quad \Phi \in \mathbb{R}^{N\times r}$$

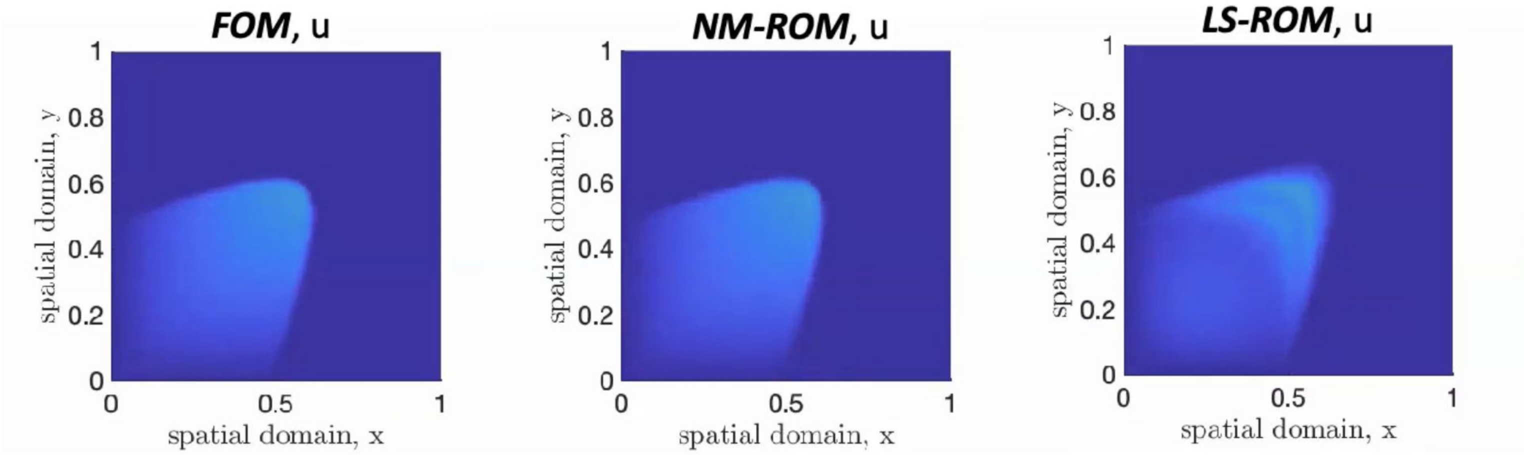
$$\frac{du}{dt} \approx \Phi^\top \frac{d\hat{u}}{dt} = f(u_{ref} + \Phi\hat{u},\, t;\, \mu)$$

Nonlinear projection:

$$u \approx \tilde{u} = u_{ref} + D_\phi(\hat{u})$$

$$\frac{du}{dt} \approx J_D(\hat{u})\frac{d\hat{u}}{dt} = f(u_{ref} + D_\phi(\hat{u}), t; \mu)$$

From Linear Model to Neural Network



latent space
dimension of 5

method	NM-ROM	LS-ROM
max. rel. error (%)	0.93	34.4
speed-up	11.6	26.8

Autoencoders

- Learning the latent space manifold via autoencoder:

  Let $E_\theta : \mathbb{R}^N \to \mathbb{R}^r$ be an "Encoder" and $D_\phi : \mathbb{R}^r \to \mathbb{R}^N$ be a "Decoder" functions

- Loss:

  $u \approx E_\theta(D_\phi(u))$ may not work well.

  - Add a regularization term, or learn Denoising autoencoder, which became a key concept in generative models

    $L(u, D(E(u))) + \Omega(E(u))$ $\quad\quad$ $L(u, D(E(\tilde{u}))), \tilde{u} \sim p_{data}(u)$

Autoencoders

- Learning the latent space manifold via autoencoder:

  Let $E_\theta : \mathbb{R}^N \to \mathbb{R}^r$ be an "Encoder" and $D_\phi : \mathbb{R}^r \to \mathbb{R}^N$ be a "Decoder" functions

- Loss:

  $u \approx E_\theta(D_\phi(u))$ may not work well.

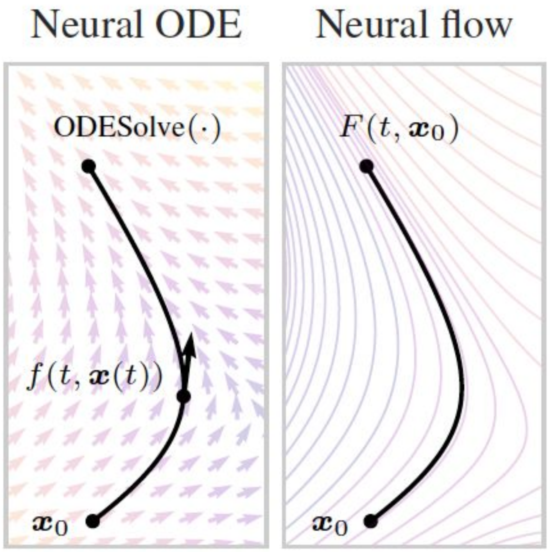
  - Add a regularization term, or learn Denoising autoencoder, which became a key concept in generative models

  - Auto-decoding: use a Decoder to encode by solving $\hat{u}_t = \arg\min_\alpha \left(\| u_t - D_\phi(\alpha) \|^2\right)$

Learning Latent Dynamics

Chen et al. (2019), Bilos et al. (2021)

Neural ODEs



$$\frac{d\,\hat{u}}{d\,t} = f_\theta(\hat{u},\, t\,;\, \mu),\;\; \hat{u}, f \in \mathbb{R}^r$$

- Instead of specifying a discrete sequence of hidden layers, Neural ODEs parameterize the derivative of the hidden state using a neural network
- The function f describes (determines) how the latent state changes in time, which is completely unknown.
- Apply numerical time integrators to evolve u_hat. Alternatively, one can directly learn $F_\psi \; s.t. \; \frac{dF}{dt} = f$

Neural ODEs

Learning:

- Combined with Decoder,

$$\min_{\theta} \mathbb{E}||\, z_t - \left(z_0 + \int_0^t f_\theta(z_\tau) d\tau \right) ||^2$$

$$s.t.\ z_{[0,T]}\ ,\ \phi\ =\ \arg\min\ \mathbb{E}_{t,x,\mu}\, \|\, u_t(x)\ -\ D_\phi(z_t)(x)\|^2$$

Generative latent function time-series model

- Once f_θ is learned, we can evolve the latent state via ODE solver.
- Each trajectory is determined from initial latent state z_{t_0} and a set of latent dynamics shared across all time series.
- A generative model can be obtained by sampling $z_{t_0} \sim p(z_{t_0})$
- Then,

$$z_{t_1},\ \ldots z_{t_n} = \text{ODEsolve}\left(z_{t_0},\ f,\ \theta;\ t_1, \ldots, t_n\right)$$

$$u_{t_n}\ =\ D_\phi(z_{t_n})$$

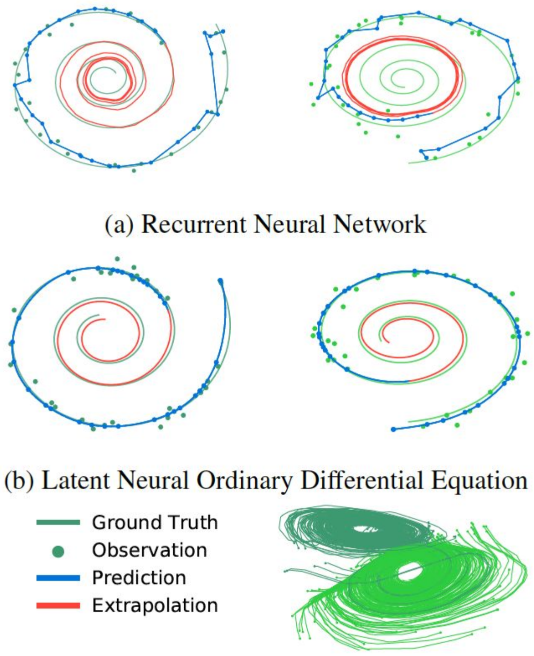
Time-series Latent ODE Experiment

A. RNN with 25 hidden units

B. NODE model with 4-dimensional latent space, f and a decoder parametrized by one-hidden-layer with 20 hidden units respectively.

A dataset of 1000 2-dimensional spirals (clockwise and counter-clockwise), each starting at a different point, sampled at 100 timesteps, with gaussian noise added.

(a, b) Reconstruction and extrapolation, (c) A projection of 4-dimensional latent ODE trajectories onto their first two dimensions, colored by the two different direction



(a) Recurrent Neural Network

(b) Latent Neural Ordinary Differential Equation

(c) Latent Trajectories

Time-series Latent ODE Experiment

- Data-space trajectories decoded from varying one dimension of z_{t_0}
- The latent trajectories change smoothly as a function of the initial point z_{t_0}, switching from clockwise to counter clockwise
- Color indicates progression through time, starting at purple.

Extra Topics on Recent Advances in Scientific Machine Learning

Recent advances in Scientific Machine Learning

Space and time continuous models and their generalization towards robustness in extrapolation.

1. Operator Learning: DeepONet (DO) and Neural Operator (NO)
   - ➔ Given function (e.g. initial condition) as input, return solution function.
2. Implicit Neural Representation (INR)
   - ➔ INR is a coordinate-based neural networks, using sinusoidal filters to capture signals.
   - ➔ DINo (Yin 2023) integrates INR as a decoder to approximate functions independently of the observation grid, while leveraging latent state dynamics to model the temporal evolution of solution states.

NN as a Function Operator

$$\frac{du}{dt} = f(u, t; \mu), \; u, f \in \mathbb{R}^N$$

$$s.t. \; u(x, 0) = g(x)$$

- Initial Value Problem.
- DeepONet
  - Integrates two distinct NNs
  - Branch network is a vector valued NN $\mathbf{c}(\cdot;\theta) = (c_0(\cdot;\theta), \ldots, c_N(\cdot;\theta))^\top$
  - Trunk network is a vector valued NN defined on $\Omega_y \subset \mathbb{R}^{d_y}$, $\phi(\cdot;\psi) = (1, \phi_1(\cdot;\psi), \ldots, \phi_N(\cdot;\psi))^\top$

➔ $u[g;\Theta](x) := \phi^\top(x;\psi)\,\mathbf{c}(g;\theta), \; \Theta = \{\psi, \theta\}$

Ha et al. (2016), Sitzmann et al. (2020)

Learning parameters

DNN: $\min_{\Theta} \mathcal{F}(X;\Theta), \quad \Theta = \{W, b\}$

input layer: vector x
input dimension: m

output layer: y = σ(Wx+b)
output dimension: m'

Simple NN model with one layer
Learns W and b

HyperDNN: $\min_{\Phi} \mathcal{F}(X;\Theta) = \mathcal{F}(X;\mathcal{H}(C;\Phi)).$

- Hypernetworks are neural networks that generate **weights** for another neural network.

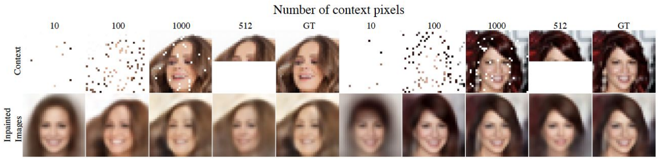
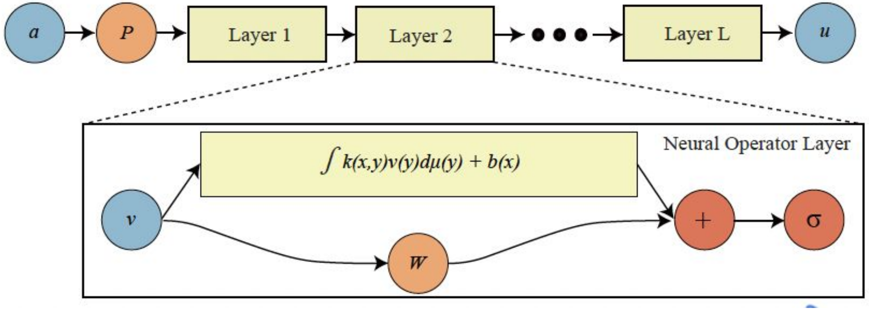


Figure 6: Generalizing across implicit functions parameterized by SIRENs on the CelebA dataset [49]. Image inpainting results are shown for various numbers of context pixels in O_j.

$$C = I_\theta(x, y),\ I_\theta : \mathbb{R}^2 \to \mathbb{R}^3$$

Learn an image by getting coordinates as input and three color channels as output. Then, image is a function itself.
Learn multiple images by learning parameter θ_i for each image.

Neural Operator



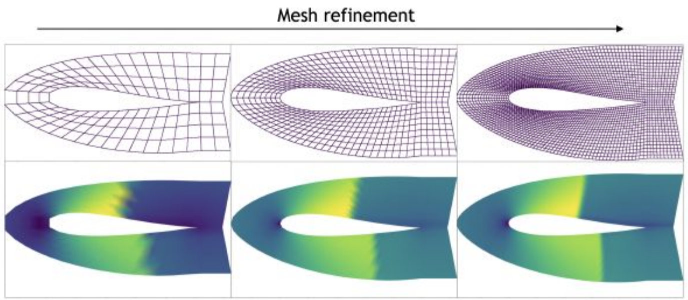


Figure 1: Discretization Invariance
An discretization-invariant operator has convergent predictions on a mesh refinement.

Learns the kernel function

$e.g.\ k(x,y) = \kappa(x-y)1_{B(x,r)}(y), \kappa_\psi : D \to \mathbb{R}^{n\times m}$

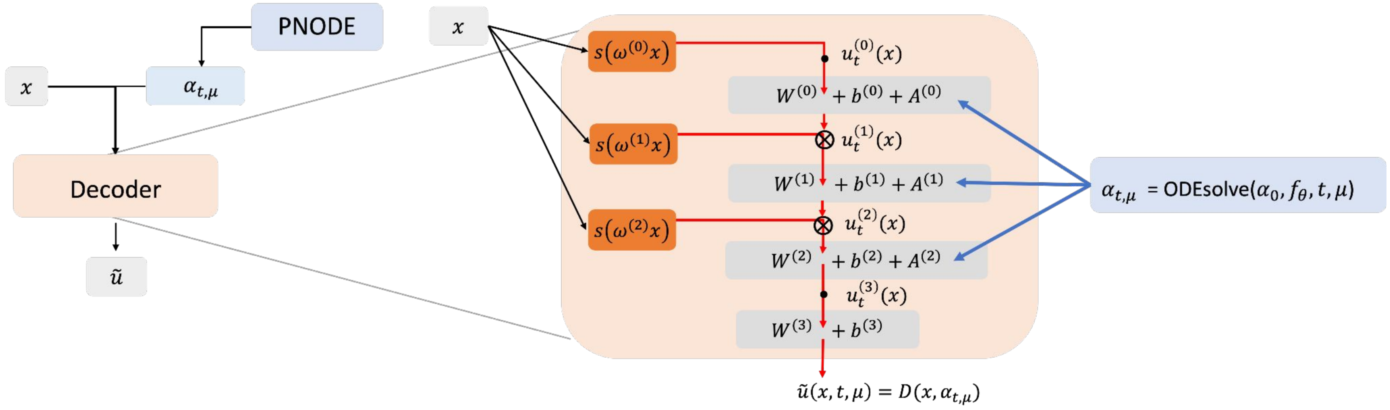
Then, the integral operator is a Convolution ftn

Idea of spatial generalization:

Learns ψ for κ_ψ, θ for b_θ, $W \in \mathbb{R}^{n\times m}$

Nothing depends on J

Implicit Neural Representation



- The coordinate-based neural network solution (decoder output) is conditionally defined based on the low-dimensional latent state.
- The parametrized neural ODE (PNODE) learns different trajectories of latent states for each PDE parameter.
- Sinusoidal filters are used to construct Fourier basis and efficiently capture the spatial signal.

References

- Y. Kim, Y. Choi, D. Widemann, T. Zohdi, A fast and accurate physics-informed neural network reduced order model with shallow masked autoencoder, *Journal of Computational Physics*, 451, 110841, **2022**
- I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, *MIT press*, **2016**
- J. Park, P. Florence, J. Straub, R. Newcombe, S. Lovegrove, DeepSDF: Learning continuous signed distance functions for shape representation, *CVPR*, pp. 165–174, **2019**
- R. Chen, Y. Rubanova, J. Bettencourt, D. Duvenaud, Neural ordinary differential equations, *NeurIPS* 31, pp. 6572–6583, **2018**
- M. Bilos, J. Sommer, S. Rangapuram, T. Januschowski, S. Gunnemann, Neural Flows: Efficient Alternative to Neural ODEs, *NeurIPS*, **2021**
- Lu, L., Jin, P., Pang, G. *et al.* Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nat Mach Intell* 3, pp. 218–229, **2021**
- S. Lee and Y. Shin, On the training and generalization of deep operator networks, arXiv preprint, **2023**
- D. Ha, A. Dai, Q. Le, HyperNetworks, *ICLR*, **2017**
- V. Sitzmann, J. Martel, A. Bergman, D. Lindell, G. Wetzstein, Implicit neural representations with periodic activation functions, *NeurIPS* 33, pp. 7462–7473, **2020**
- N. Kovachki, Z. Li, B. Liu, K. Azizzadeneshell, K. Bhattacharya, A. Stuart, A. Anandkumar, Neural operator: Learning maps between function spaces, *Journal of Machine Learning Research* 24, **2023**
- Y. Yin, M. Kirchmeyer, J. Franceschi, A. Rakotomamonjy, and P. Gallinari, Continuous PDE dynamics forecasting with implicit neural representations, *In The Eleventh International Conference on Learning Representations*, **2023**
- M. Kim, T. Wen, K. Lee, Y. Choi, Physics-informed reduced order model with conditional neural fields, *NeurIPS 2024 Workshop on Machine Learning and the Physical Sciences*, **2024**