Lecture 8: Solving PDEs via Operator Learning

Haizhao Yang
Department of Mathematics
University of Maryland College Park

2022 Summer Mini Course
Tianyuan Mathematical Center in Central China

Why Operator Learning?

Broad applications

- Reduced order modeling: learning operators in lower dim
- Solving parametric PDEs
- Solving inverse problems
- Density function theory: potential function to density function
- Phase retrieval: data to images
- Image processing: image to image
- Predictive data science: historical states to future states

Probably most mappings are high-dim or even infinite-dim

Example 1: Burgers equation

$$\partial_t u(x,t) + \partial_x(u^2(x,t)/2) = \nu \partial_{xx} u(x,t), \quad x \in (0,1), t \in (0,1]$$
$$u(x,0) = u_0(x)$$

- Periodic boundary conditions
- $\nu = 0.1$: a given viscosity coefficient
- Applications in fluid mechanics, nonlinear acoustics, gas dynamics, and traffic flow
- Goal: learn the mapping from $u_0(x)$ to $u(x,1)$.

Example 2: the steady-state of the 2D Darcy Flow equation

$$
\begin{aligned}
-\nabla \cdot (a(x)\nabla u(x)) &= f(x), \quad x \in (0,1)^2 \\
u(x) &= 0, \quad x \in \partial(0,1)^2
\end{aligned}
$$

- f: a given forcing function
- Applications in modeling the pressure of subsurface flow, the deformation and the electric potential of materials
- Goal: learn the forward mapping from $a(x)$ to $u(x)$.

Why Discretization-Invariant

Main concern in applications

- Good accuracy
- Low cost

Heterogeneous data structures in practice

- No discretization-invariance: repeated and expensive training
- Discretization-invariance: training once is enough

Learning Mathematical Operators

Notations

- Function spaces $\mathcal{X}$ and $\mathcal{Y}$, e.g., $\mathbb{R}$-valued over domain $\Omega \subset \mathbb{R}^D$
- Operator $\Psi : \mathcal{X} \to \mathcal{Y}$
- Data samples $\mathcal{S} = \{u_i, v_i\}_{i=1}^{2n}$ with

$$v_i = \Psi(u_i) + \epsilon_i,$$

where $u_i \overset{\text{i.i.d.}}{\sim} \gamma$ and $\epsilon_i \overset{\text{i.i.d.}}{\sim} \mu$

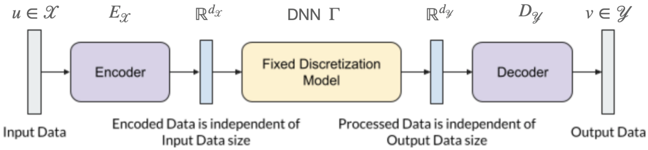
Goal

- Learn Ψ from samples $\mathcal{S}$

Method

- Deep neural networks $\Psi^n(u; \theta)$ as parametrization
- Supervised learning to find $\Psi^n(\cdot; \theta^*) \approx \Psi(\cdot)$

Operator Learning with **Fixed** Input and Output Sizes



Most methods:

Encoder-decoder of $\mathcal{X}$

- $D_{\mathcal{Y}} \circ E_{\mathcal{X}} \approx I$, $E_{\mathcal{X}} : \mathcal{X} \to \mathbb{R}^{d_{\mathcal{X}}}$, $D_{\mathcal{Y}} : \mathbb{R}^{d_{\mathcal{X}}} \to cX$
- Encoder $E_{\mathcal{X}}$: sampling, basis expansion, PCA, etc.
- Decoder $D_{\mathcal{X}}$: interpolation, basis expansion, PCA, etc.

Encoder-decoder of $\mathcal{Y}$

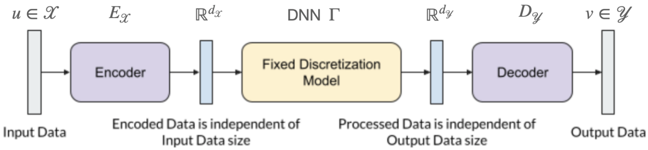
- Similar

Learning

- A DNN $\Gamma \approx \bar{\Psi} : \mathbb{R}^{d_{\mathcal{X}}} \to \mathbb{R}^{d_{\mathcal{Y}}}$
- $D_{\mathcal{Y}} \circ \Gamma \circ E_{\mathcal{X}} \approx \Psi : \mathcal{X} \to \mathcal{Y}$

Repeated and expensive re-training if $d_{\mathcal{X}}$ or $d_{\mathcal{Y}}$ changes

Operator Learning with Only a **Fixed** Input Size



DeepOnet: Chen & Chen, 1995; Lu, Jin, and Karniadakis, 2019:

$$v(z) = \Psi^n(u;\theta)(z) = \sum_{j=1}^{d_{\mathcal{Y}}} \alpha_j(E_{\mathcal{X}}(u);\theta)\psi_j(z;\theta)$$

- Encoder $E_{\mathcal{X}} : u \in \mathcal{X} \to E_{\mathcal{X}}(u) \in \mathbb{R}^{d_{\mathcal{X}}}$ via sampling
- DNN $\Gamma : E_{\mathcal{X}}(u) \in \mathbb{R}^{d_{\mathcal{X}}} \to \alpha \in \mathbb{R}^{d_{\mathcal{Y}}}$
- Decoder $D_{\mathcal{Y}} : \alpha \in \mathbb{R}^{d_{\mathcal{Y}}} \to v \in \mathcal{Y}$ using basis functions $\{\psi_j(z;\theta)\}_{j=1}^{d_{\mathcal{Y}}}$

Learning

- $D_{\mathcal{Y}} \circ \Gamma \circ E_{\mathcal{X}} \approx \Psi : \mathcal{X} \to \mathcal{Y}$

Repeated and expensive re-training if $d_{\mathcal{X}}$ changes

Discretization Invariant Operator Learning

Li, Kovachki, Azizzadenesheli, Liu, Bhattacharya, Stuart, Anandkumar, 2020

Deep neural network parametrization of $v = \Psi(u)$

$$v(z) = \Psi^n(u;\theta)(z) = Q_\theta \circ \mathcal{K}_\theta^L \circ \cdots \circ \mathcal{K}_\theta^1 \circ P_\theta(u)(z)$$

- Mapping $u \in \mathcal{X}$ to $v(z) \in \mathcal{Y}$ defined for $z \in \Omega_{\mathcal{Y}}$
- P_θ and Q_θ: pointwise linear transform
- $\mathcal{K}_\theta^j$: nonlinear integral transform
- $\Psi^n(u;\theta) \approx \Psi(u)$ via least squares

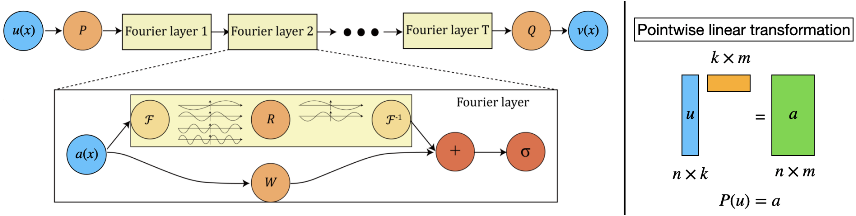
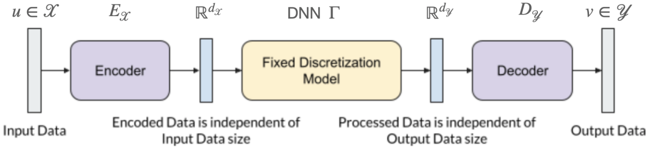


Figure: An illustration of Fourier Neural Operator (FNO) by Li et al. P, Q, R, and W are pointwise linear transformation.

Discretization Invariant Operator Learning

Ong, Shen, Y., arXiv:2203.05142
Sparsity: Key to discretization-invariance

Our idea 1 of network construction



Encoder and decoder

- Discretization-invariant
- Capture intrinsic dimension (sparsity)

Fixed discretization model

- Powerful expressivity
- Deep neural network (DNN)

Discretization Invariant Operator Learning

Ong, Shen, Y., arXiv:2203.05142

Nonlinear integral transforms as encoder and decoder

$$v(y) = \int_{\Omega_{\mathcal{X}}} \phi(u(x), x, y; \theta) u(x) dx$$

- Mapping $u \in \mathcal{X}$ to $v(y) \in \mathcal{Y}$ defined for $y \in \Omega_{\mathcal{Y}}$
- Kernel ϕ is a DNN parametrized by θ
- $\int_{\Omega_{\mathcal{X}}}$ is discretized according to the discrete $u(x)$

Discretization Invariant Operator Learning

Ong, Shen, Y., arXiv:2203.05142

Why integral-kernel-based encoder and decoder?

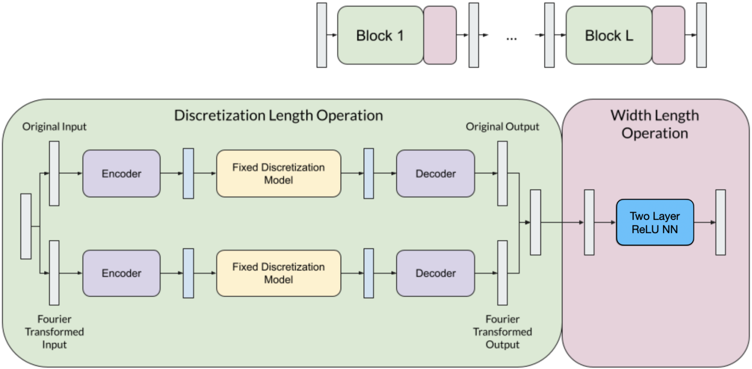
$$v(y) = \int_\Omega \phi(u(x), x, y; \theta) u(x) dx$$

- DNN expressivity: Fourier, Wavelet, other integral operators
- Data driven sparsity, i.e., DNN-based PCA

Discretization Invariant Operator Learning

Our idea 2 of network construction

- Parallel blocks (e.g., spatial and frequency domains)
- Post-processing ReLU NN
- Deep network via densely connected composition



Discretization Invariant Operator Learning

Our idea 3 for randomized data augmentation

Loss function

$$\min_{\theta} \mathbb{E}_{(u,v)\sim p_{data}} \mathbb{E}_S \left[\mathcal{L}\left(\Psi(u;\theta), v\right) + \lambda \mathcal{L}\left(\Psi(S(u);\theta), S(v)\right) \right]$$

- $\Psi(u;\theta)$ discretization-invariant neural network
- $\mathcal{L}(\cdot,\cdot)$: typical loss function, e.g., $\mathcal{L}(x,y) = ||x-y||^2$
- Random interpolation operator S
- p_{data}: joint distribution of (u,v) in $\mathcal{X} \times \mathcal{Y}$
- $\lambda > 0$

Numerical Comparison

Existing methods

- **UNet**, Ronneberger et al., MICCAI, 2015
- **DeepOnet**, Lu et al., Nature Machine Intelligence, 2021
- **FNO** (Fourier Neural Operator), Li et al., ICLR 2021
- **FT** (Fourier Transformer) and **GT** (Galerkin Transformer), S. Cao, NeurIPS, 2021

Examples

- Prediction
- Forward problems
- Inverse problems
- Signal processing

Numerical Comparison

Prediction of future states

Example 1: Burgers equation:

$$\partial_t u(x,t) + \partial_x(u^2(x,t)/2) = \nu \partial_{xx} u(x,t), \quad x \in (0,1), t \in (0,1]$$
$$u(x,0) = u_0(x)$$

- Periodic boundary conditions
- $\nu = 0.1$: a given viscosity coefficient
- Applications in fluid mechanics, nonlinear acoustics, gas dynamics, and traffic flow
- Goal: learn the mapping from $u_0(x)$ to $u(x,1)$.

Numerical Comparison

Example 1: Burgers equation:

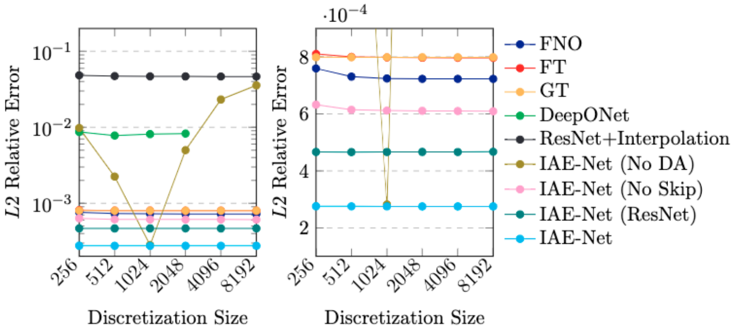


Figure: $L2$ relative error with $\nu = 1e^{-1}$ (Left) and its closeup (Right). Models are trained with $s = 1024$ and tested on the other resolutions.

Numerical Comparison

Forward problem

Example 2: the steady-state of the 2D Darcy Flow equation:

$$-\nabla \cdot (a(x)\nabla u(x)) = f(x), \quad x \in (0,1)^2$$
$$u(x) = 0, \quad x \in \partial(0,1)^2$$

- f: a given forcing function
- Applications in modeling the pressure of subsurface flow, the deformation and the electric potential of materials
- Goal: learn the forward mapping from $a(x)$ to $u(x)$.

Numerical Comparison

Example 2: the steady-state of the 2D Darcy Flow equation:

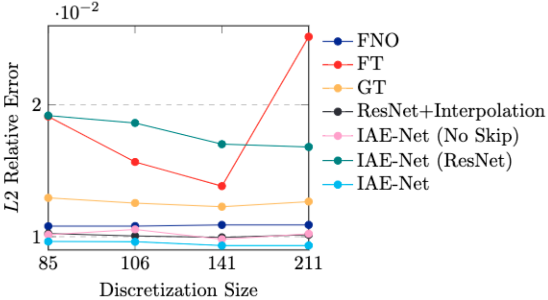


Figure: $L2$ relative error. Models are trained with $s = 141$ size training data and tested on the other resolutions.

Numerical Comparison

Inverse problem

Example 3: inverse scattering.

- Applications: non-destructive testing, medical imaging, seismic imaging, etc.
- Helmholtz equation

$$\left(-\nabla - \frac{\omega^2}{c(x)^2}\right) u(x) = 0$$

  with a given frequency ω and unknown speed $c(x)$
- Introduce

$$\frac{\omega^2}{c(x)^2} = \frac{\omega^2}{c_0(x)^2} + \eta(x), \qquad L_0 = -\nabla - \frac{\omega^2}{c_0(x)^2}$$

  with $c_0(x)$ given in applications
- Helmholtz equation:

$$\left(-\nabla - \frac{\omega^2}{c(x)^2}\right) u(x) = (L_0 - \eta(x))u(x) = 0$$

  as a parametric PDE with parameter η
- Goal: learn the mapping from $u(x)$ at sensor locations to $\eta(x)$

Numerical Comparison

Inverse problem
Example 3: inverse scattering.

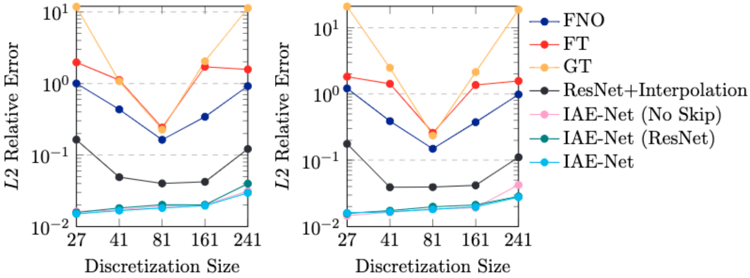


Figure: $L2$ relative error for the forward (Left) and inverse (Right) problem. Model is trained with $s = 81$ and tested on different resolutions.

Numerical Comparison

Image/signal processing
Example 4: blind source separation.

- Applications in image processing, medical imaging, audio signal, health measurement

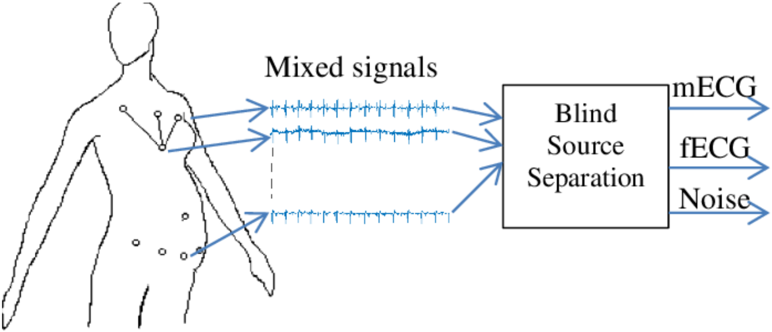


Figure: Extracting fetal ECG from mother's measurement plays an important role in diagnosing fetus's health. Figure credited to Bensafia et al.

Numerical Comparison

Example 4: blind source separation.

Table: Trained with size $s = 2000$ and tested on different resolutions for zero-shot generalization.

Model Name	**500**	**1000**	**2000**	**4000**
FNO	24.75%	16.76%	15.97%	18.23%
GT	27.24%	18.97%	17.75%	19.2%
DeepONet†			99.99%	
Unet	101%	68.78%	8.274%	69.85%
ResNet + Interpolation	43.73%	32.13%	31.16%	31.92%
IAE-Net (No Skip)	10.68%	8.723%	7.904%	8.153%
IAE-Net (ResNet)	9.924%	7.925%	7.15%	7.192%
IAE-Net	8.638%	7.048%	6.802%	6.848%