Neural Inverse Operators for Solving PDE Inverse Problems

Yunan Yang

February 21, 2024

Department of Mathematics, Cornell University

Scientific Machine Learning: Theory and Algorithms workshop
February 21-23, 2024, The University of Maryland

The Collaborators

Roberto Molinaro | Björn Engquist | Siddhartha Mishra

The paper https://arxiv.org/pdf/2301.11167.pdf, also in ICML 2023.

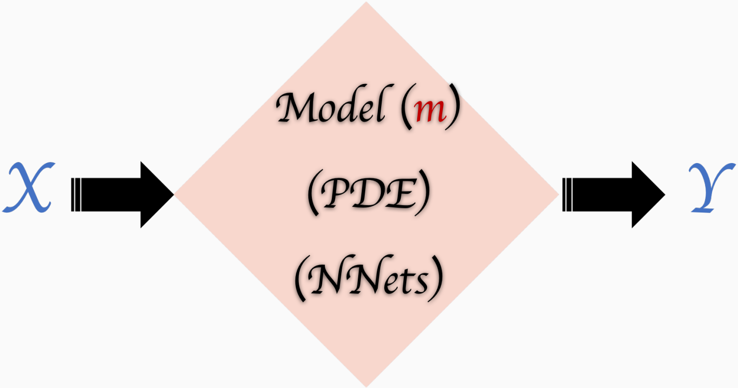
Inverse Problems

The modeling step (the forward problem)

Well-Known Inverse Problems:

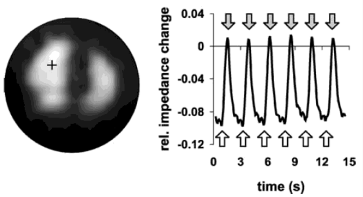
Locate Earthquake Source, Image the Black Hole, X-ray/CT/Ultrasound

General “Inverse Problems”



Inverse data matching problems aim at finding m such that the predicted outputs $(X, F(m))$ match given measured data (X, Y).

Calderón's Problem (Electrical Impedance Tomography, EIT)



— Wikipedia

$$\begin{cases} \nabla \cdot (a(x)\nabla u) = 0, & x \in \Omega \\ u(x) = \psi, & x \in \partial\Omega \end{cases}$$

Given "Dirichlet-to-Neumann" map

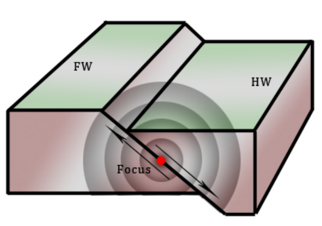
$$\Lambda_a : \mathcal{H}^{1/2}(\partial\Omega) \longrightarrow \mathcal{H}^{-1/2}(\partial\Omega)$$
$$\Lambda_a : \psi \longrightarrow a\nabla u_\psi \cdot \mathbf{n},$$

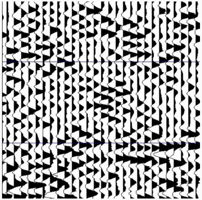
the goal is to find

$$a(x), \quad x \in \Omega.$$

Kohn, R. V., & Vogelius, M. (1987). Relaxation of a variational method for impedance computed tomography. CPAM.

Waveform Inversion (FWI)





— Wikipedia

$$\begin{cases} m(\mathbf{x}) \dfrac{\partial^2 u(\mathbf{x}, t)}{\partial t^2} - \triangle u(\mathbf{x}, t) = s(\mathbf{x}, t) \\ \text{Zero i.c. in half-space } \Omega \\ \text{Neumann b.c. on } \partial\Omega \end{cases}$$

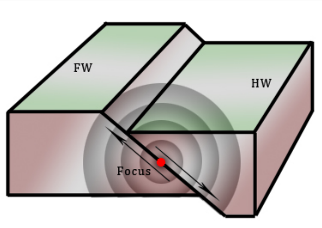
$m(\mathbf{x}) = \dfrac{1}{c(\mathbf{x})^2}$, $c(\mathbf{x})$ is the wave velocity

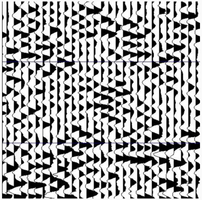
Given $u(x_r, y_r, z = 0, t)$ the goal is to find

$$m(x), \quad x \in \Omega.$$

Tarantola, A. (2005). Inverse problem theory and methods for model parameter estimation. SIAM.

Helmholtz Equation Based Inversion





— Wikipedia

$$\begin{cases} \triangle u + \omega^2 m(x) u = s(x, \omega) \\ \text{Neumann b.c. on } \partial\Omega \end{cases}$$

$m(\mathbf{x}) = \frac{1}{c(\mathbf{x})^2}$, $c(\mathbf{x})$ is the wave velocity

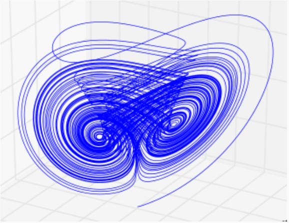
Given $u(x_r, y_r, z = 0; \omega)$ the goal is to find

$$m(x), \quad x \in \Omega.$$

Colton, David L., Rainer Kress. Inverse acoustic and electromagnetic scattering theory. Vol. 93. Berlin: Springer, 1998.

Modeling the Dynamics

"Chen" System [Chen-Ueta, 1999]



Y.-Nurbekyan-Negrini-Martin-Pasha, 2023. Optimal transport for parameter identification of chaotic dynamics via invariant measures. SIADS.

A general parameterized dynamical system may take the form

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix} = v(x, y, z; \underbrace{\sigma, \rho, \beta}_{\theta}) \approx v(\mathbf{x}, \theta)$$

where $v \approx v(\cdot, \theta)$ can be

- polynomials,
- basis functions,
- neural networks, and so on,

where θ corresponds to

- expansion coefficients,
- neural network weights, etc.

The Forward Model $F(m)$

F is given; we just find m (e.g., PDEs).

- Pro: We know the best (exact) forward problem!
- Con: The forward and inverse problems are so nonlinear!

OR

F is not known; we are free to choose (e.g., XXX-net).

- Pro: The freedom to modify it to a "better" map
  - Over-Parametrization;
  - Model Extension;
  - Model Reduction.
- Con: Trial and error to build the model

How to Solve $F(m) = g$

- Linear Inverse Problem, i.e., $Am = g$
  (often combined with numerical linear algebra)
  - Direct Method
  - Iterative Method
  - Optimization-Based Method (e.g., least-squares min)

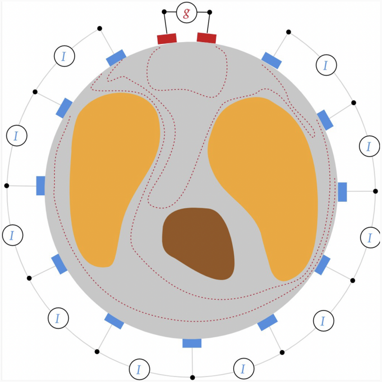
How to Solve $F(m) = g$

- Linear Inverse Problem, i.e., $Am = g$
(often combined with numerical linear algebra)
  - Direct Method
  - Iterative Method
  - Optimization-Based Method (e.g., least-squares min)

- Nonlinear Inverse Problem, $F(m) = g$
  - Direct Method (challenging to construct) (**in today's talk**)
  - Iterative Method (e.g., nonlinear GMRES)
  - Optimization Method

Learn a Direct Inverse Map

Example: Calderón's Problem



$$\begin{cases} \nabla \cdot (a(x)\nabla u) = 0, & x \in \Omega \\ u(x) = \psi, & x \in \partial\Omega \end{cases}$$

Given Dirichlet-to-Neumann map

$\Lambda_a : \mathcal{H}^{1/2}(\partial\Omega) \longrightarrow \mathcal{H}^{-1/2}(\partial\Omega)$

$\Lambda_a : \quad \psi \quad \longrightarrow a\nabla u_\psi \cdot \mathbf{n},$

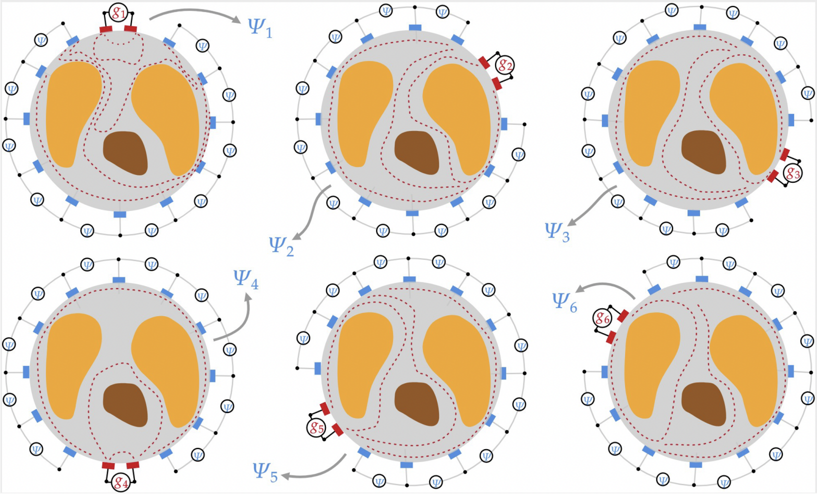
the goal is to find

$$a(x), \quad x \in \Omega.$$

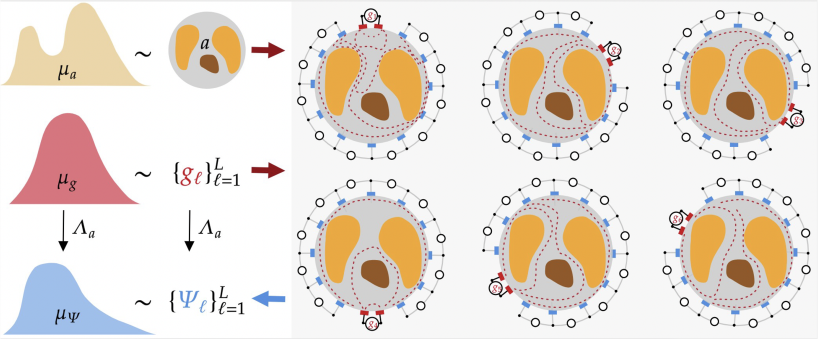
In suitable settings, it is provable that there exists an inverse problem with (log-) stability.

The Data Acquisition

Recall that the data is an **operator** on the continuous level



The Training Data Acquisition



We provide $\left(a^{(i)}, \{\Psi_\ell^{(i)}\}_\ell\right)$ as the training data for $i = 1, \ldots, n$ number of different parameter samples with $a^{(i)} \sim \mu_a$.

$$\{\Psi_\ell^{(i)}\}_\ell \approx \mu_\Psi = \Lambda_{a^{(i)}} \sharp \mu_g \,, \quad \mu_g \text{ fixed.}$$

Push-Forward Map

Consider π and μ as two probability measures on the domain X and Y, respectively. We say T is a mass-preserving push-forward map (i.e., $\mu = T\sharp\pi$) if

$$\boxed{\mu(A) = \pi\left(T^{-1}(A)\right)},$$

where $A \subseteq Y$ is an arbitrary Borel measurable set and $T^{-1}(A) \subseteq X$ denotes its preimage.

Push-Forward Map

Consider π and μ as two probability measures on the domain X and Y, respectively. We say T is a mass-preserving push-forward map (i.e., $\mu = T\sharp\pi$) if

$$\boxed{\mu(A) = \pi\left(T^{-1}(A)\right)},$$

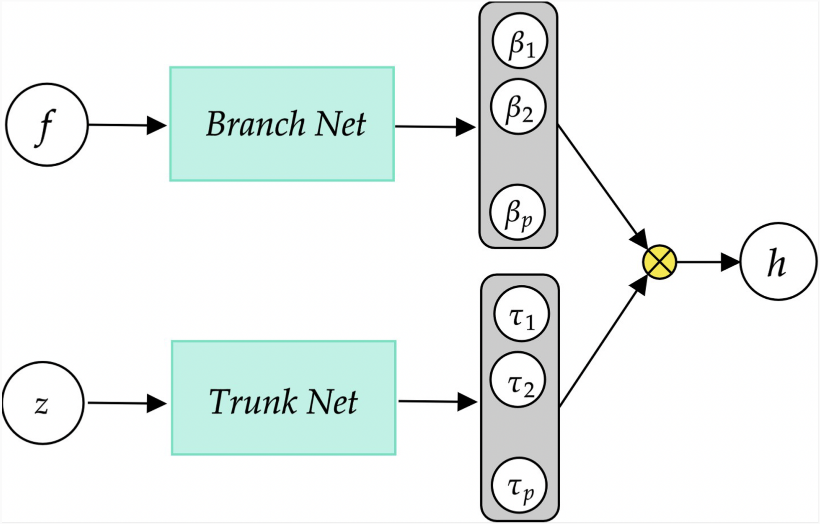
where $A \subseteq Y$ is an arbitrary Borel measurable set and $T^{-1}(A) \subseteq X$ denotes its preimage.

If $d\pi = p(x)dx$, $d\mu = q(x)dx$, we further have

$$\boxed{p(x)dx = q\left(T(x)\right)\left|\nabla_x T(x)dx\right|},$$

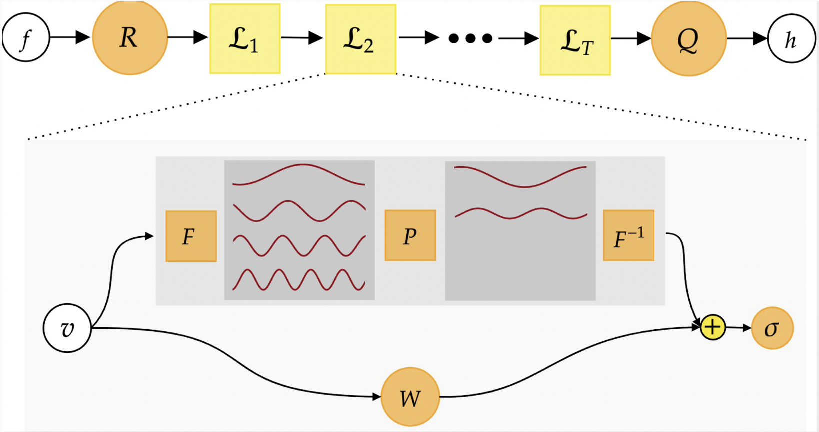
the well-known change of variable formula.

The Neural Network Architecture — DeepONet



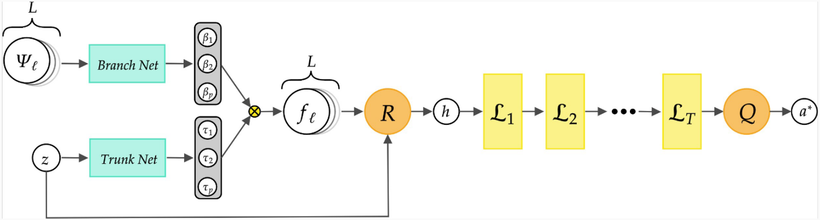
NN_1: DeepONet [Lu-Jin-Karniadakis,2019]

The Neural Network Architecture — Fourier Neutral Operator



NN_2: Fourier Neutral Operator (FNO) [Li et al., 2020]

The Neural Network Architecture



DeepONet FNO

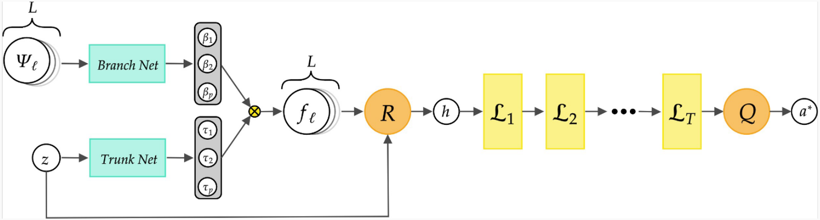
The proposed Neural Inverse Operator (NIO)

An Intuition:

DeepONet: $\{\Psi_\ell\} \mapsto \{f_\ell\}$ (analogy: $\underline{\{a\nabla u_\psi \cdot \mathbf{n}\} \text{ on } \partial\Omega}$ to $\underline{\{u_\psi\} \text{ on } \Omega}$)

FNO: $\{f_\ell\} \mapsto a$ (analogy: $\underline{\{u_\psi\} \text{ on } \Omega}$ to $\underline{a \text{ on } \Omega}$)

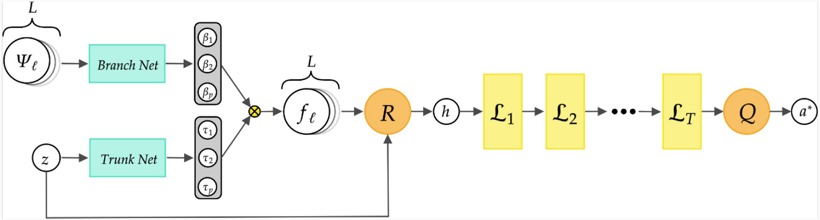
The Neural Network Architecture



DeepONet (NN_1) FNO (NN_2)

$$NIO\left(\Lambda_a \sharp \mu_g\right) = NN_2\left(\underbrace{NN_1 \sharp \underbrace{\left(\Lambda_a \sharp \mu_g\right)}_{\mu_\Psi}}_{\text{samples } \{f_\ell\}}\right) \to a.$$

The Neural Network Architecture

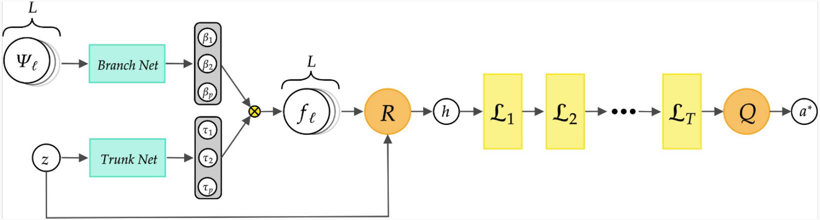


DeepONet (NN_1) FNO (NN_2)

$$NIO\left(\Lambda_a\sharp\mu_g\right) = NN_2\left(\underbrace{NN_1\sharp\underbrace{\left(\Lambda_a\sharp\mu_g\right)}_{\mu_\Psi}}_{\text{samples } \{f_\ell\}}\right) \to a.$$

One concern: NN does not know $\{\Psi_\ell\}$ are samples of μ_Ψ and similarly $\{f_\ell\}$ are samples of an underlying distribution.

The Neural Network Architecture



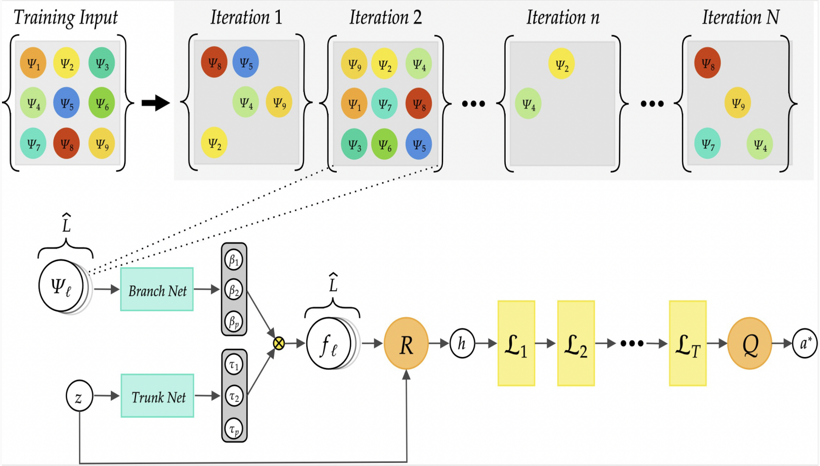
DeepONet (NN_1) FNO (NN_2)

$$NIO\left(\Lambda_a \sharp \mu_g\right) = NN_2\Bigg(\underbrace{NN_1 \sharp \underbrace{\left(\Lambda_a \sharp \mu_g\right)}_{\mu_\Psi}}_{\text{samples } \{f_\ell\}}\Bigg) \to a.$$

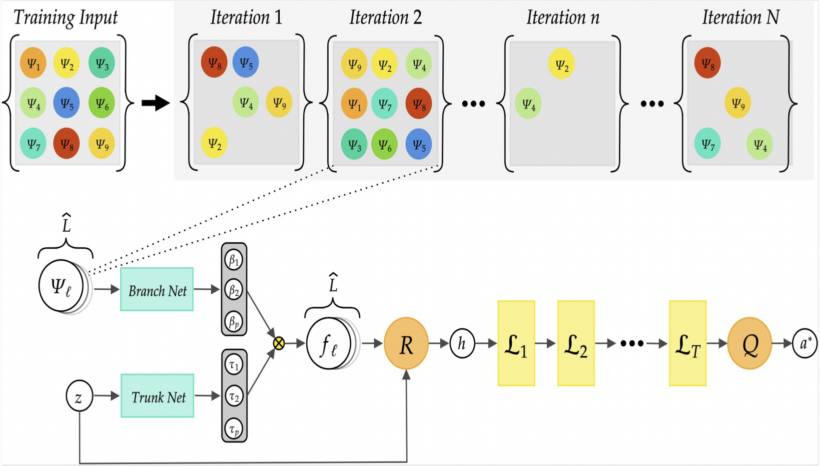
One concern: NN does not know $\{\Psi_\ell\}$ are samples of μ_Ψ and similarly $\{f_\ell\}$ are samples of an underlying distribution.

We want: (1) permutation invariant; (2) different *a* can have different *L*; (3) testing data can have a different *L*

The Training Scheme — Bagging — "Randomized Batching"



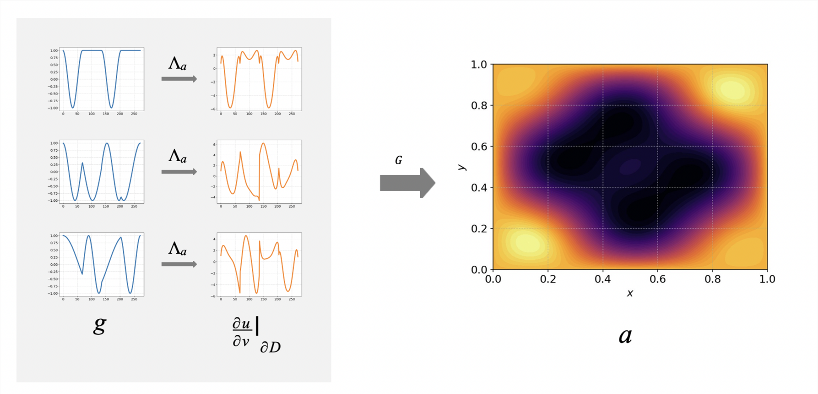
The Training Scheme — Bagging — “Randomized Batching”



Rich theoretical analysis in “Bagging” from statistical learning.

Numerical Results

EIT Examples



$$\begin{cases} \nabla \cdot (a(x)\nabla u) = 0, & x \in \Omega \\ u(x) = \psi, & x \in \partial\Omega \end{cases}$$

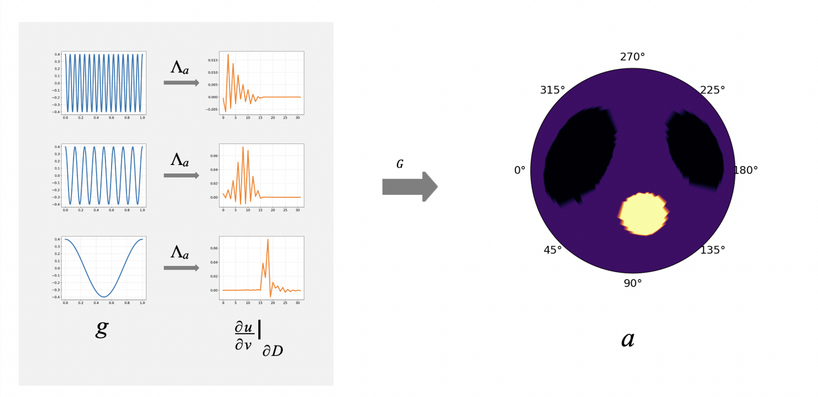
Given DtN map

$\Lambda_a : \mathcal{H}^{1/2}(\partial\Omega) \longrightarrow \mathcal{H}^{-1/2}(\partial\Omega)$

$\Lambda_a : \quad \psi \quad \longrightarrow a\nabla u_\psi \cdot \mathbf{n},$

the goal is to find $a(x)$, $x \in \Omega$.

EIT Examples



$$\begin{cases} \nabla \cdot (a(x)\nabla u) = 0, & x \in \Omega \\ u(x) = \psi, & x \in \partial\Omega \end{cases}$$

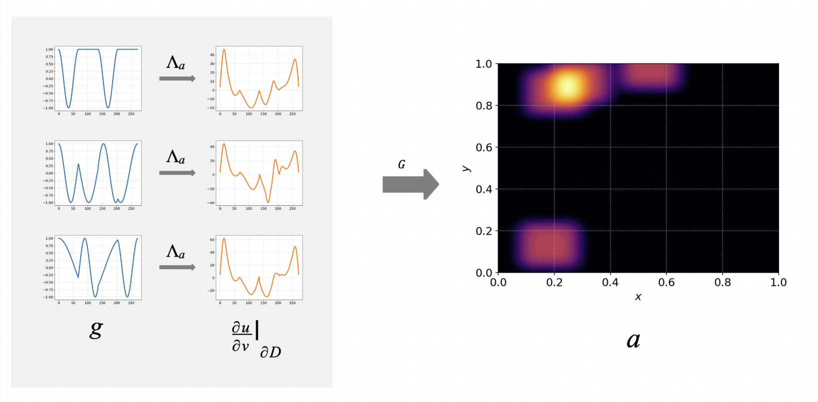
Given DtN map

$$\Lambda_a : \mathcal{H}^{1/2}(\partial\Omega) \longrightarrow \mathcal{H}^{-1/2}(\partial\Omega)$$

$$\Lambda_a : \quad \psi \longrightarrow a\nabla u_\psi \cdot \mathbf{n},$$

the goal is to find $a(x)$, $x \in \Omega$.

EIT Examples



$$\begin{cases} \nabla \cdot (a(x)\nabla u) = 0, & x \in \Omega \\ u(x) = \psi, & x \in \partial\Omega \end{cases}$$

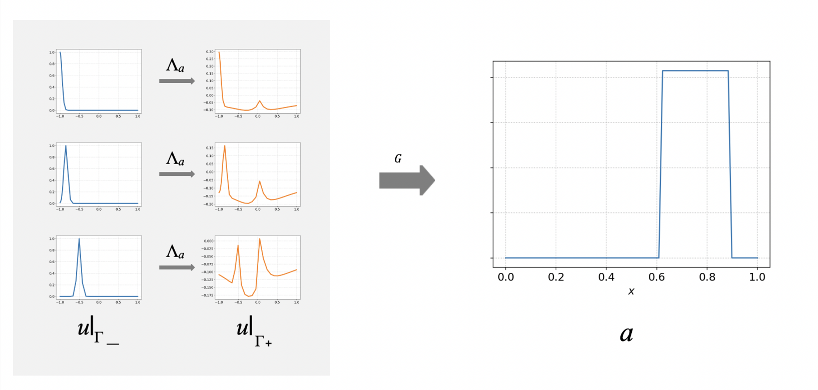
Given DtN map

$$\Lambda_a : \mathcal{H}^{1/2}(\partial\Omega) \longrightarrow \mathcal{H}^{-1/2}(\partial\Omega)$$

$$\Lambda_a : \quad \psi \quad \longrightarrow a\nabla u_\psi \cdot \mathbf{n},$$

the goal is to find $a(x)$, $x \in \Omega$.

RTE Inversion Examples



$$v \cdot \nabla_z u(z,v) + \sigma_a(z) u(z,v) = \frac{1}{\epsilon} a(z) Q[u],\ z \in D$$

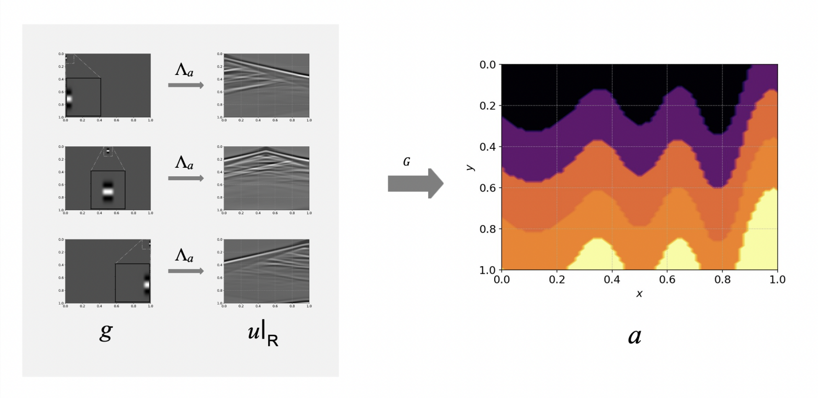
$$u(z,v) = \phi(z,v),\ z \in \Gamma_-$$

Given the Albedo operator

$$\Lambda_a : L^1(\Gamma_-) \mapsto L^1(\Gamma_+)$$

$$\Lambda_a : u\big|_{\Gamma_-} = \phi \mapsto u\big|_{\Gamma_+}$$

Wave Inversion Results



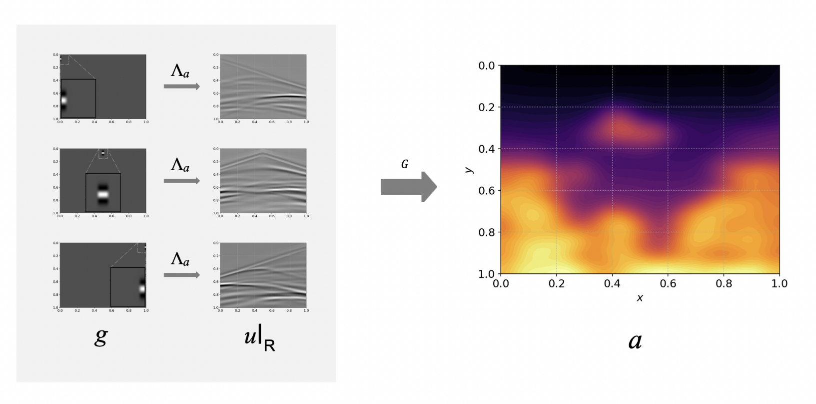
$$u_{tt}(t,z) + a(z)^2 \Delta u = s, \quad (z,t) \in D \times [0,T],$$

Given the *Source-to-Receiver* (StR) operator,

$$\Lambda_a : L^2([0,T] \times D) \mapsto L^2([0,T]; X_R),$$

$$\Lambda_a : s \mapsto u\big|_{[0,T]\times R},$$

Wave Inversion Results



$$u_{tt}(t,z) + a(z)^2 \Delta u = s,$$
$$(z,t) \in D \times [0,T],$$

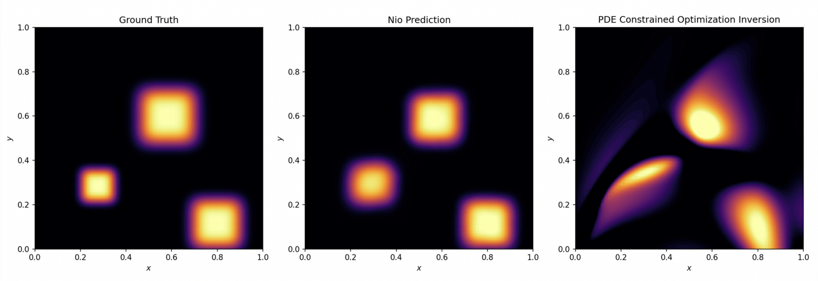
Given the *Source-to-Receiver* (StR) operator,

$$\Lambda_a : L^2([0,T] \times D) \mapsto L^2([0,T]; X_R),$$
$$\Lambda_a : s \mapsto u\big|_{[0,T]\times R},$$

Testing Performance Comparison

	DONet		FCNN		NIO	
	$L^1\downarrow$	$L^2\downarrow$	$L^1\downarrow$	$L^2\downarrow$	$L^1\downarrow$	$L^2\downarrow$
EIT Trigonometric	1.97%	2.36%	1.49%	1.82%	**0.85%**	**1.05%**
EIT Heart&Lungs	0.95%	3.69%	0.27%	1.62%	**0.18%**	**1.16%**
EIT Inclusion Detection	3.83%	7.41%	2.53%	7.55%	**1.07%**	**2.94%**
Optical Imaging	2.35%	4.35%	1.46%	3.71%	**1.1%**	**2.9%**
Seismic Imaging - CurveVel - A	3.98%	5.86%	**2.65%**	5.05%	2.71%	**4.71%**
Seismic Imaging - Style - A	3.82%	5.17%	3.12%	4.63%	**3.04%**	**4.36%**

Compare with PDE-Constrained Optimization



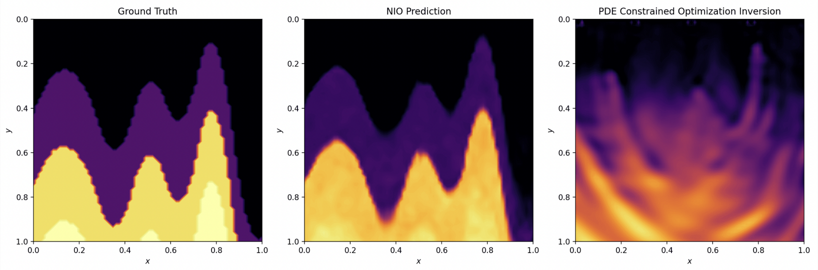
The ill-posed Calderón Problem (inverse Darcy flow)

$$\begin{cases} \nabla \cdot (a(x)\nabla u) = 0, & x \in \Omega \\ u(x) = \psi, & x \in \partial\Omega \end{cases}$$

$$\min_{a \in A(D)} \sum_{i=1}^{L} \text{dist}(\mathcal{F}_i(a), d_i^{\text{obs}}) \quad s.t. \text{ PDE constraints}$$

Difficulty in PDE-Constrained optimization: high wavenumber (i.e., edges)

Compare with PDE-Constrained Optimization



The full waveform inversion (FWI) problem

$$u_{tt}(t,z) + a^2(z)\Delta u = s,$$
$$(z,t) \in D \times [0,T],$$

$$\min_{a \in A(D)} \sum_{i=1}^{L} \text{dist}(\mathcal{F}_i(a), d_i^{\text{obs}}) \quad s.t. \text{ PDE constraints}$$

Difficulty in PDE-Constrained optimization: local minima

Conclusions

Summary and Future Directions

Summary

- We consider a large class of PDE-based inverse problems that are “solvable” only when providing a data operator (e.g., DtN, Albedo).

Summary and Future Directions

Summary

- We consider a large class of PDE-based inverse problems that are "solvable" only when providing a data operator (e.g., DtN, Albedo).
- Learn mappings from operators to functions to solve inverse problem.

Summary and Future Directions

Summary

- We consider a large class of PDE-based inverse problems that are "solvable" only when providing a data operator (e.g., DtN, Albedo).
- Learn mappings from operators to functions to solve inverse problem.
- Motivated by the inverse-problem nature, we proposed a novel architecture, termed **Neural Inverse Operator** (NIO), based on a composition of DeepONet and FNO.

Summary and Future Directions

Summary

- We consider a large class of PDE-based inverse problems that are "solvable" only when providing a data operator (e.g., DtN, Albedo).
- Learn mappings from operators to functions to solve inverse problem.
- Motivated by the inverse-problem nature, we proposed a novel architecture, termed **Neural Inverse Operator** (NIO), based on a composition of DeepONet and FNO.
- Random batching to inform FNO the input is an *empirical distribution*.

Summary and Future Directions

Summary

- We consider a large class of PDE-based inverse problems that are “solvable” only when providing a data operator (e.g., DtN, Albedo).
- Learn mappings from operators to functions to solve inverse problem.
- Motivated by the inverse-problem nature, we proposed a novel architecture, termed **Neural Inverse Operator** (NIO), based on a composition of DeepONet and FNO.
- Random batching to inform FNO the input is an *empirical distribution*.

Summary and Future Directions

Summary

- We consider a large class of PDE-based inverse problems that are “solvable” only when providing a data operator (e.g., DtN, Albedo).
- Learn mappings from operators to functions to solve inverse problem.
- Motivated by the inverse-problem nature, we proposed a novel architecture, termed **Neural Inverse Operator** (NIO), based on a composition of DeepONet and FNO.
- Random batching to inform FNO the input is an *empirical distribution*.

Future Directions

1. Conduct theoretical analysis on the generalization error

Summary and Future Directions

Summary

- We consider a large class of PDE-based inverse problems that are "solvable" only when providing a data operator (e.g., DtN, Albedo).
- Learn mappings from operators to functions to solve inverse problem.
- Motivated by the inverse-problem nature, we proposed a novel architecture, termed **Neural Inverse Operator** (NIO), based on a composition of DeepONet and FNO.
- Random batching to inform FNO the input is an *empirical distribution*.

Future Directions

1. Conduct theoretical analysis on the generalization error
2. How do different ways of representing Λ_a affect convergence?

Summary and Future Directions

Summary

- We consider a large class of PDE-based inverse problems that are "solvable" only when providing a data operator (e.g., DtN, Albedo).
- Learn mappings from operators to functions to solve inverse problem.
- Motivated by the inverse-problem nature, we proposed a novel architecture, termed **Neural Inverse Operator** (NIO), based on a composition of DeepONet and FNO.
- Random batching to inform FNO the input is an *empirical distribution*.

Future Directions

1. Conduct theoretical analysis on the generalization error
2. How do different ways of representing Λ_a affect convergence?
3. How does the PDE inverse problem stability improve using statistical learning-type of algorithms?

Acknowledgment

Thanks for your attention!

All my collaborators.