Towards efficient deep operator learning for forward and inverse PDEs

Ke Chen
Mathematics
University of Maryland

February 21, Scientific Machine Learning: Theory and Algorithm, Brin Mathematics Research Center

Introduction to inverse problems

$$\mathcal{L}_a u = 0$$
$$\mathcal{B} u = g$$

a is the parameter of interest
g is the boundary condition
u is the PDE solution

Forward Problem

given PDE parameters and boundary condition,
find the PDE solution u

Inverse Problem

given multiple measurements of solution u,
find PDE parameter a

Solving inverse problems

Physical Model

$$\mathcal{L}_a u = 0$$
$$\mathcal{B} u = g$$

Data Collection

choose experiment setup g_i

obtain measurement data $h_i = \mathcal{M} u_i$

repeat for g_{i+1}

Reconstruction

given dataset $(g_i, h_i)_{i=1}^n$

find the parameter function a

Applications

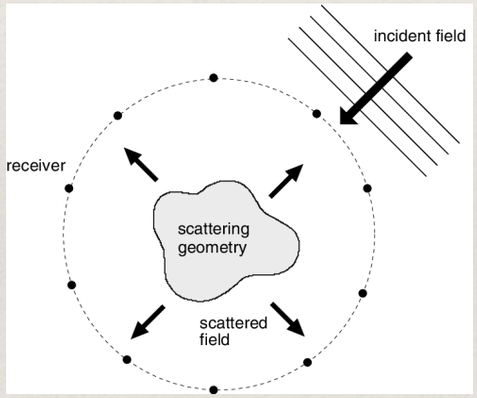
Optical Tomography (photon transport)

Y. Hoshi, Progress in Brain Research, 2016

$$\begin{cases} v \cdot \nabla_x f(x,v) = \sigma_s(x)\mathcal{L}[f](x,v) - \sigma_a(x) f(x,v) \\ f|_{\Gamma_-} = \phi(x,v) \end{cases}$$

photon intensity -> attenuation and scattering

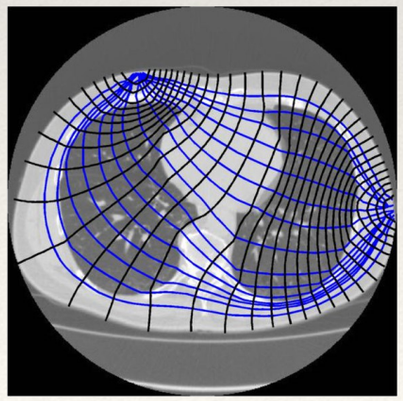
Full wave inversion



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

far field wave -> wave speed function

Electric Impedance Tomography: Calderón problem

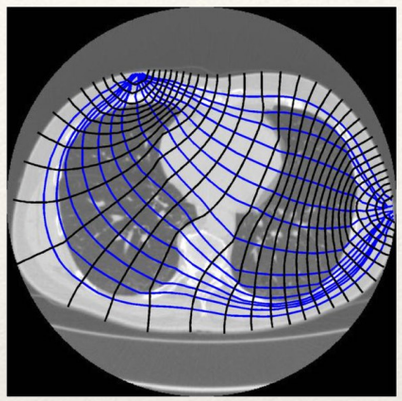


$$-\nabla \cdot (a(x)\nabla u(x)) = 0$$

electric current -> conductivity function

Challenges

EIT: Calderón problem



$$-\nabla \cdot (a(x)\nabla u(x)) = 0$$
$$u|_{\partial\Omega} = g$$
$$\Lambda_a : g \mapsto h := a\frac{\partial u}{\partial n}|_{\partial\Omega}$$

Uniqueness: Sylvester and Uhlmann, 1987, Haberman and Tataru 2011
Stability: logarithmic estimate Alessandrini, 1988

$$\|a_1 - a_2\| \leq \frac{C}{|\log \|\Lambda_{a_1} - \Lambda_{a_2}\||^{\sigma}} + \|\Lambda_{a_1} - \Lambda_{a_2}\|$$

- Challenges

Most inverse problems are nonlinear and ill-posed.
A large number of data is needed to obtain uniqueness.
prior information is needed to tackle the ill-posedness and noises.

Solving inverse problems

Given experimental measurement data:

$$S_a = \{(g_i, h_i),\ i = 1, \ldots, n \mid h_i = \Lambda_a g_i + \text{noise}\}$$

g_i: setup of the i-th experiment

h_i: measurements of the i-th experiment

Λ_a: PDE model

- Optimization method

$$\min \frac{1}{n} \sum_i \|\Lambda_a g_i - h_i\|^2 + R(a)$$

$$a_0 \xrightarrow{\text{solve PDE 2n times}} a_1 \xrightarrow{\text{solve PDE 2n times}} \cdots \xrightarrow{\text{solve PDE 2n times}} a_N$$

a large number N of iteration is needed due to ill-posedness
each iteration requires a large number n of PDE solves for unique reconstruction
each PDE solve requires re-assembling due to updating PDE parameters

Solving inverse problems

Given experimental measurement data:

$$S_a = \{(g_i, h_i),\ i = 1, \ldots, n \mid h_i = \Lambda_a g_i + \text{noise}\}$$

setup of the i-th experiment

PDE model

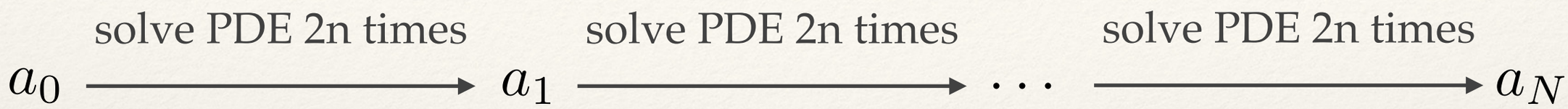
measurements of the i-th experiment

- Bayesian method $\pi(a \mid S_a) \sim \pi_{\text{prior}}(a)\pi(S_a \mid a)$

$$a_0 \xrightarrow{\text{MCMC sampling}} a_1 \xrightarrow{\text{MCMC sampling}} \cdots \xrightarrow{\text{MCMC sampling}} a_N$$

N is around 250 in practice to obtain posterior samples
each MCMC sampling = 2n PDE solves
each iteration requires a large number n of PDE solves for unique reconstruction
each PDE solve requires re-assembling due to updating PDE parameters

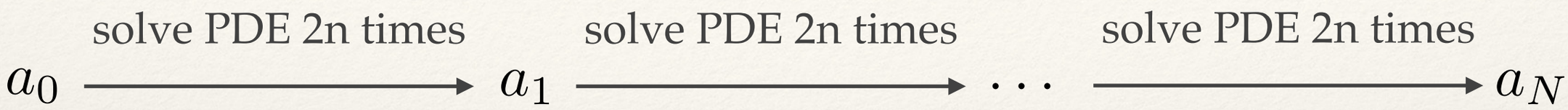
Operator learning



# of iterations	x	# of PDE solves	=	large computation
$log(\frac{1}{\varepsilon})$ or worse		n x (forward + adjoint PDE) n = # of measurements		a large number of PDE solves and assembles

Construction of PDE solvers and solving PDEs are expensive.
A fast parametric PDE solver is needed.

Learning the forward PDE



# of iterations	x	# of PDE solves	=	large computation
$log(\frac{1}{\varepsilon})$ or worse		n x (forward + adjoint PDE) n = # of measurements		large number of PDE solves and assembles

neural network surrogate for the forward PDE operator

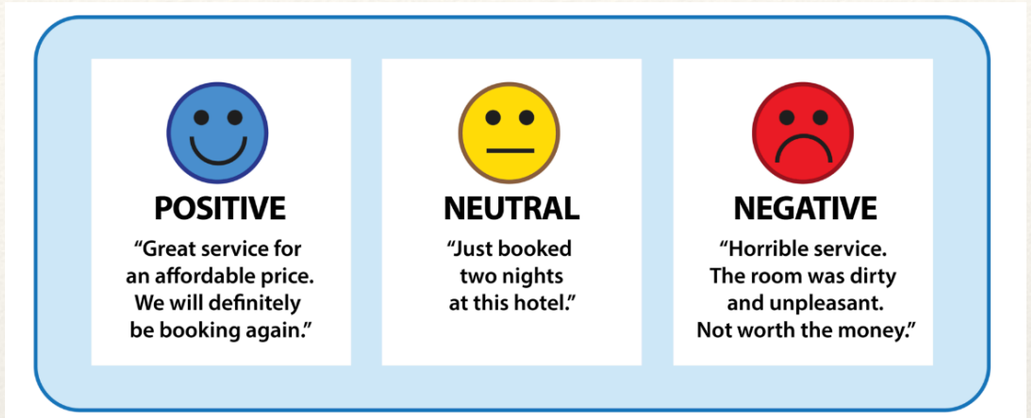
- A neural network is used to learn the "forward PDE operator"
- the forward operator maps (PDE parameter, experiment setup) to measurement
- the NN will be evaluated many times to obtain the target parameter function

Learning the forward PDE operator

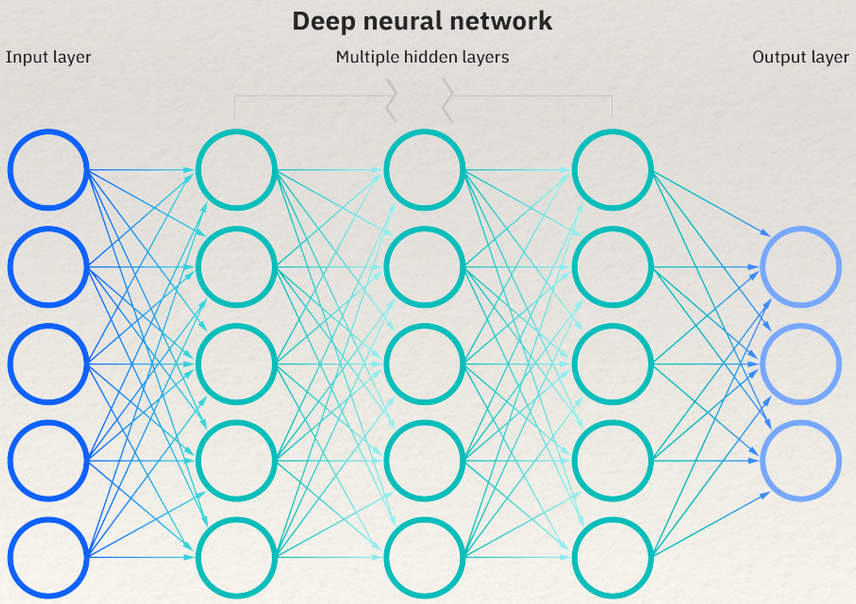
Deep learning

Classification:
Chihuahua or muffin

Large Language model:
ChatGPT



Natural Language Processing:
Sentiment Analysis



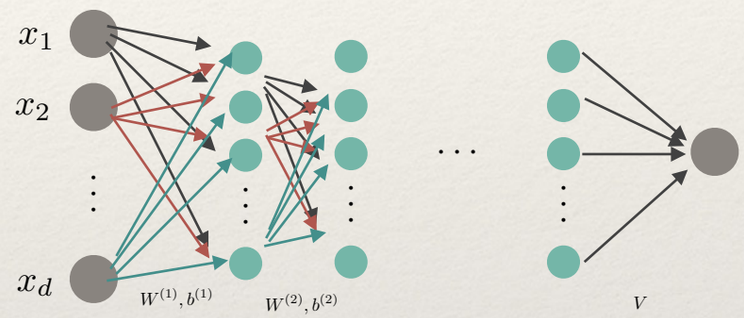
- a flexible representation for nonlinear functions
- automatic feature finder
- hardware support: CUDA, GPUs
- efficient training: SGD, ADAM,…

Learning a function

Goal:

Given a randomly generated data set $\{x_i, y_i = f(x_i)\}_{i=1}^n$ for an unknown function f

Find a DNN $h(x;\theta)$ that best fits f



Methods: training stage

Train a neural network via an optimization problem

$$\theta^* = arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \|h(x_i;\theta) - y_i\|^2$$

Validation: testing stage

Test the trained neural network $h(\cdot;\theta^*)$ on an unseen randomly generated data set

Discretization: encoder and decoder

We need encoder and decoder to discretize the PDE operator

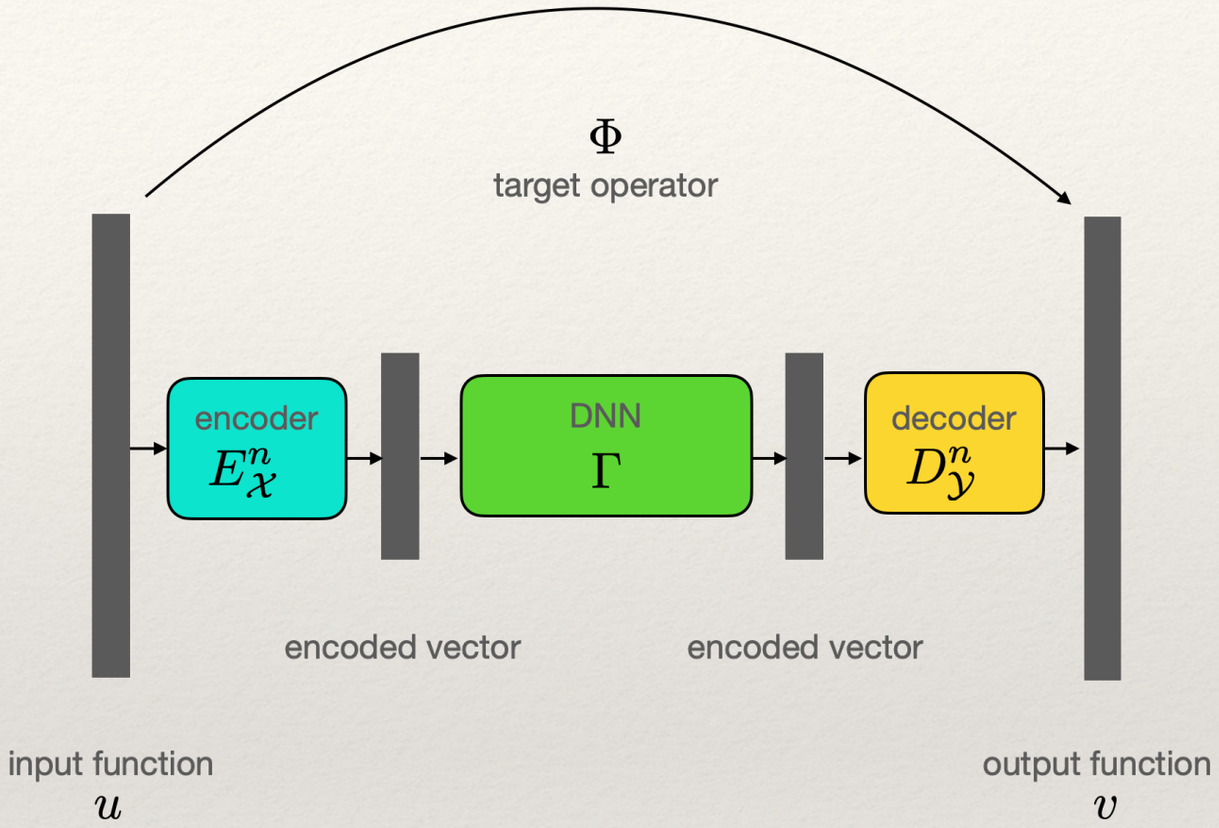
$$\Phi : \mathcal{X} \to \mathcal{Y}$$

Typical encoder-decoder method:

- PCA (SVD)
- truncated Fourier basis
- data-driven encoder-decoder
- …

$$E_{\mathcal{X}} : \mathcal{X} \to \mathbb{R}^{d_{\mathcal{X}}} \qquad D_{\mathcal{X}} : \mathbb{R}^{d_{\mathcal{X}}} \to \mathcal{X}$$

$$E_{\mathcal{Y}} : \mathcal{Y} \to \mathbb{R}^{d_{\mathcal{Y}}} \qquad D_{\mathcal{Y}} : \mathbb{R}^{d_{\mathcal{Y}}} \to \mathcal{Y}$$



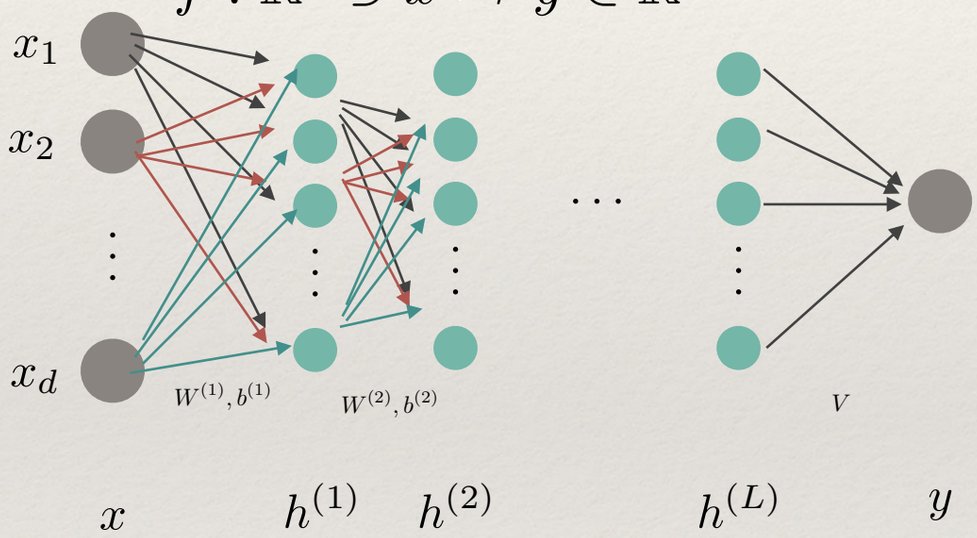
Training a PDE operator:

$$\Gamma_{\mathrm{NN}} \in \underset{\Gamma \in \mathcal{F}_{NN}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \|\Gamma \circ E_{\mathcal{X}}^n(u_i) - E_{\mathcal{Y}}^n(v_i)\|_2^2$$

Challenges: Curse of Dimensionality

function learning

$$f : \mathbb{R}^d \ni x \mapsto y \in \mathbb{R}$$



Learning error is at the order

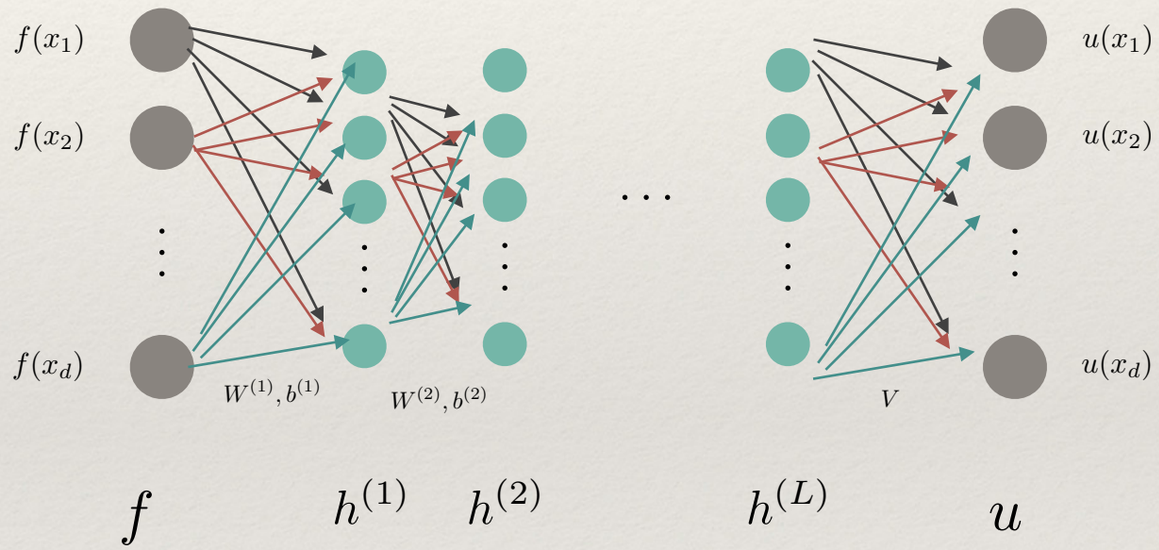
$$\mathcal{O}(n^{-\frac{1}{d}})$$

n is the number of training data

d is the input dimension

PDE operator learning

$$\Phi : L^2(\Omega) \ni f \mapsto u \in H^2(\Omega)$$



infinite dimension d implies infinitely large NN sizes!

d is the number of discretization points, not the domain dimension of PDEs

Main results: learning a Lipschitz operator

Theorem (informal version)

Consider a Lipschitz operator $\Phi : \mathcal{X} \to \mathcal{Y}$ with pre-trained encoders and decoders, there exists a neural network Γ_{NN} such that

the number of neurons is bounded by $\mathcal{O}\left(d_{\mathcal{Y}}^{\frac{4-d_{\mathcal{X}}}{4+2d_{\mathcal{X}}}} n^{\frac{d_{\mathcal{X}}}{4+2d_{\mathcal{X}}}}\right)$

and the generalization error $\mathbb{E}_S \mathbb{E}_u \|D_{\mathcal{Y}} \circ \Gamma_{\mathrm{NN}} \circ E_{\mathcal{Y}}(u) - \Phi(u)\|^2$

is bounded by

$$\mathcal{O}\left(n^{-\frac{2}{2+d_{\mathcal{X}}}} \log n\right) + \mathcal{O}(n^{-1}) + \text{encoding errors}$$

CoD: large number of training data is required for a large input dimension

C, C. Wang, H. Yang, TMLR 2022

Main results: learning a Lipschitz operator

Theorem (informal version)

Consider a Lipschitz operator $\Phi : \mathcal{X} \to \mathcal{Y}$ with pre-trained encoders and decoders, if the operator is either of low dimension or low complexity then there exists a neural network Γ_{NN} such that

the number of neurons is bounded by $\mathcal{O}\left(d_{\mathcal{Y}}^{\frac{4-d_0}{4+2d_0}} n^{\frac{d_0}{4+2d_0}}\right)$

and the generalization error $\mathbb{E}_S \mathbb{E}_u \|D_{\mathcal{Y}} \circ \Gamma_{\mathrm{NN}} \circ E_{\mathcal{Y}}(u) - \Phi(u)\|^2$

is bounded by

$$\mathcal{O}\left(n^{-\frac{2}{2+d_0}} \log n\right) + \mathcal{O}(n^{-1}) + \text{encoding errors}$$

The CoD caused by the input dimension is removed

$d_0 \ll d_{\mathcal{X}}$ depends on the structure of PDE

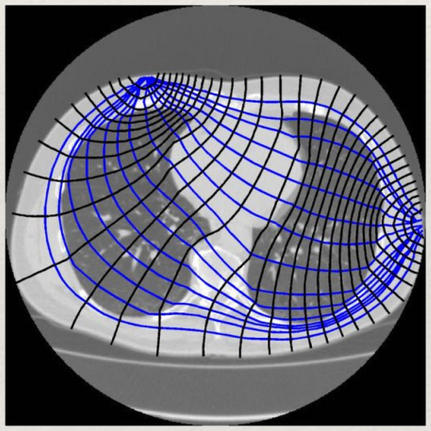
C, C. Wang, H. Yang, TMLR 2022

Low dimension structure

An operator is of low dimension d_0 if its domain is a d_0 dimensional manifold.

Examples: Electric impedance tomography

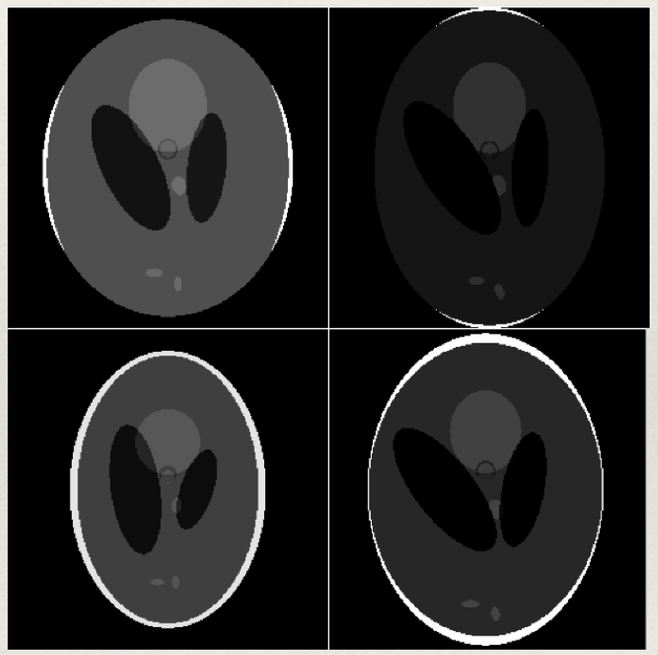
Conductivity functions of elliptic equation:



$$\begin{cases} -\mathrm{div}(a(x)\nabla_x u(x)) = 0\,, & \text{in } \Omega \subset \mathbb{R}^2, \\ u = f\,, & \text{on } \partial\Omega. \end{cases}$$

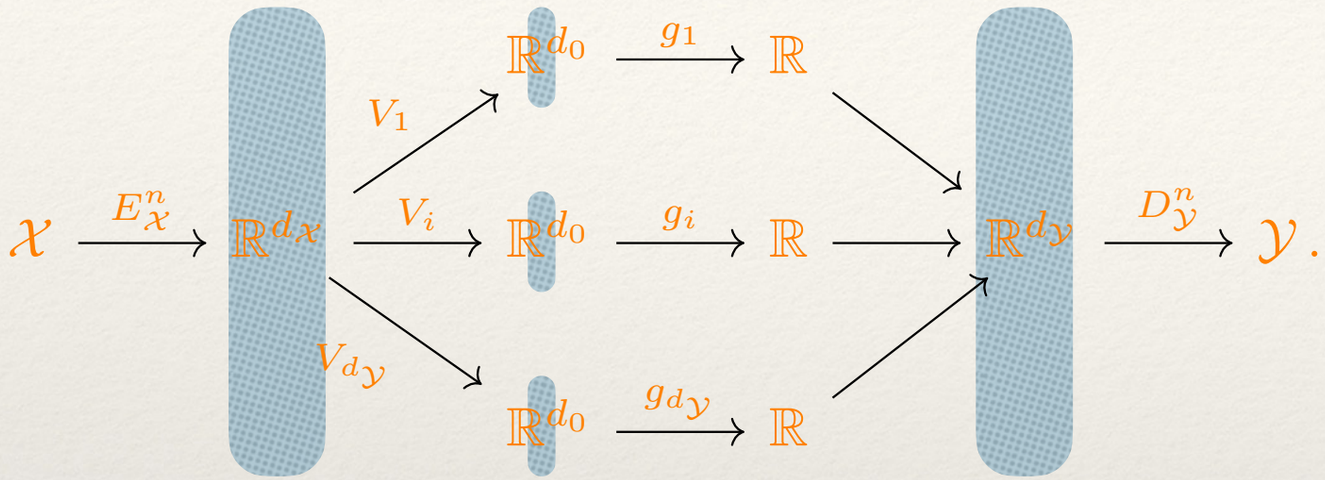
$$\Phi : a \mapsto u$$

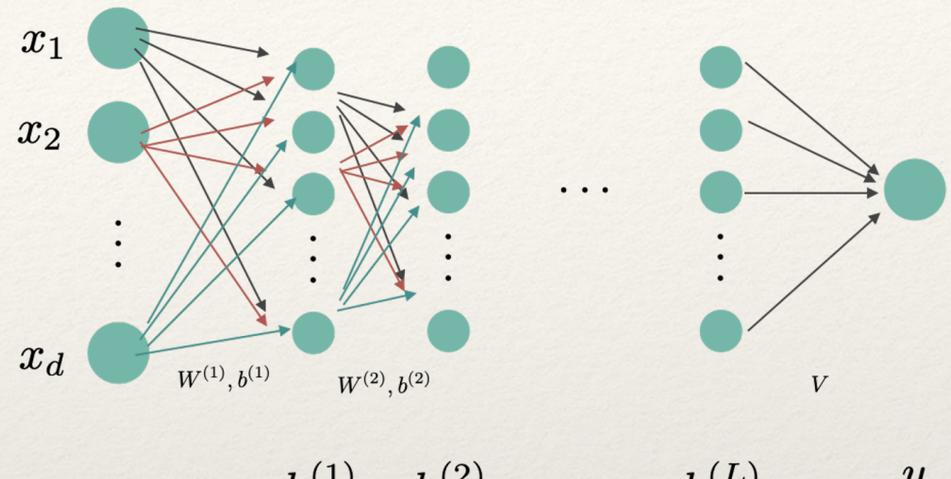
Shepp-Logan Phantoms



- The phantoms consist of ellipses of different shapes, angles, and locations.
- Each ellipse is parametrized by five numbers.

Low complexity structure

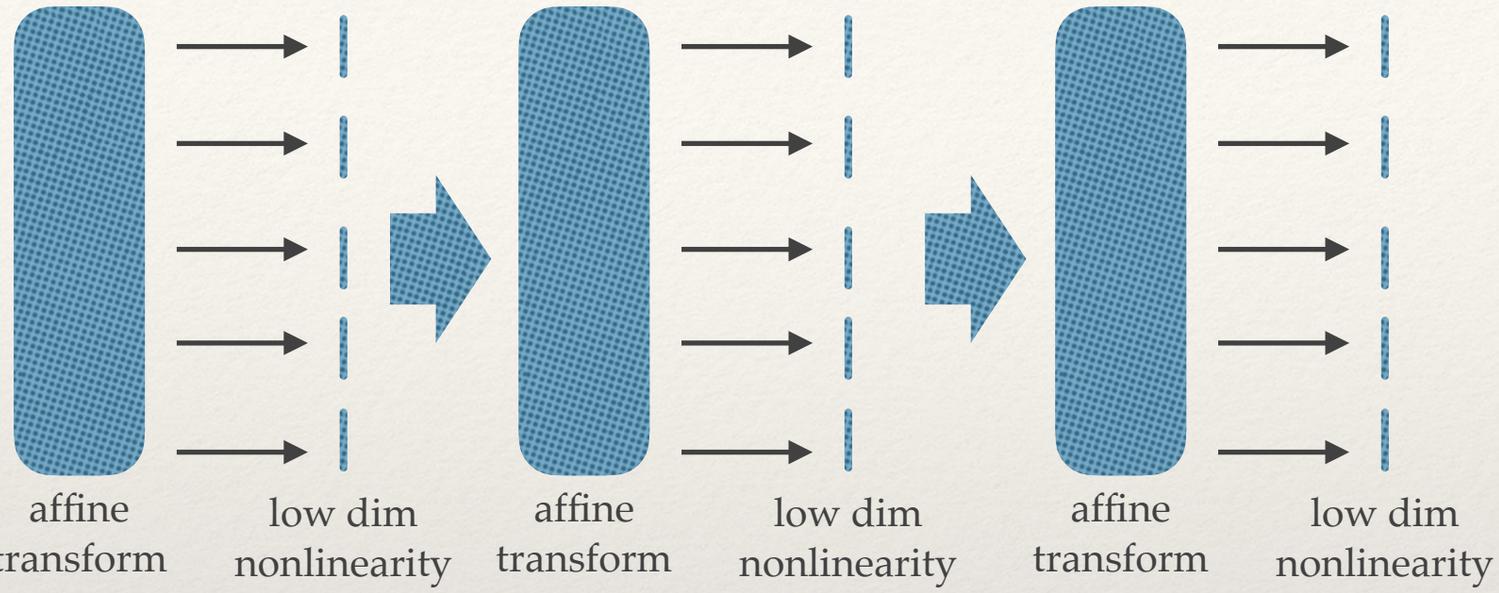




one-layer low complexity operator feedforward NN

- Low complexity structure means compositions of affine transform and low dimensional nonlinear functions.
- FNN can be considered a low-complexity operator with $d_0 = 1$

Low complexity structure



Examples: Burgers equation

$$u_t + uu_x = \kappa u_{xx}$$
$$u(x,0) = u_0(x)$$

Cole-Hopf transformation
$u = -2\kappa(\ln v)_x$

$$v_t = \kappa v_{xx}$$
$$v(x,0) = v_0(x)$$

The solution operator $\Phi : u_0 \mapsto u(\cdot, T)$ has is of low complexity with $d_0 = 2$

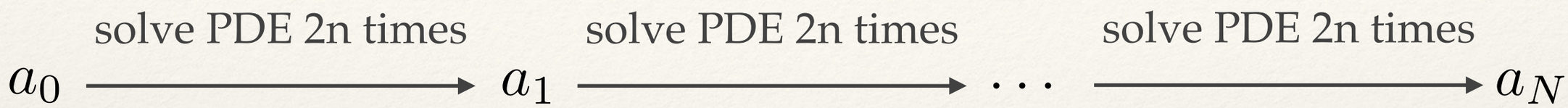
$$v_0 = \exp\left(-\frac{1}{2\kappa}\int_0^x u_0(s)ds\right)$$
one dim nonlinearity

$$v = \mathcal{K}(\cdot, T) * v_0$$
no nonlinearity

$$u(x,T) = -2\kappa\frac{v_x(x,T)}{v(x,T)}$$
two dim nonlinearity

Learning the inverse PDE operator

Learning the inverse PDE



# of iterations	x	# of PDE solves	=	large computation
$log(\frac{1}{\varepsilon})$ or worse		n x (forward + adjoint PDE) n = # of measurements		large number of PDE solves and assembles

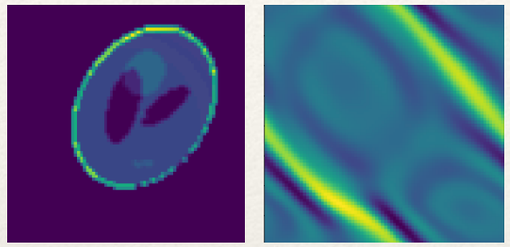
neural network surrogate of the inverse PDE operator

- A neural network is used to learn the "inverse operator"
- the inverse operator maps all measurement data to the target parameter
- the NN will be evaluated only once to obtain a reconstructed image

Learning forward and inverse operators

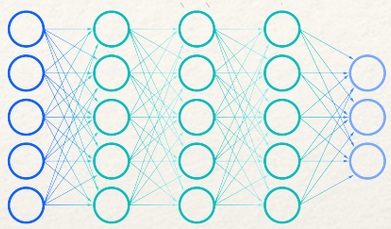
Learning forward PDE

data



(a_i, m_i)

training

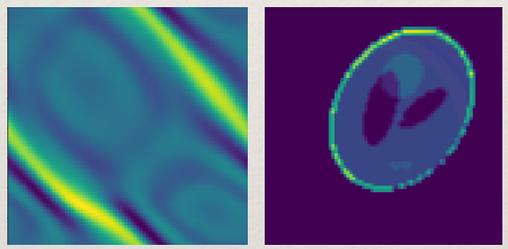


model

$\approx \quad f : a \mapsto m$

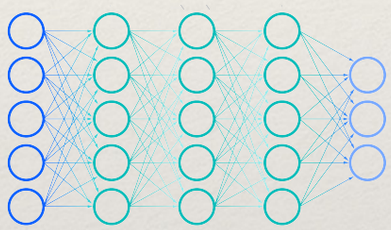
Learning inverse PDE

data



(m_i, a_i)

training

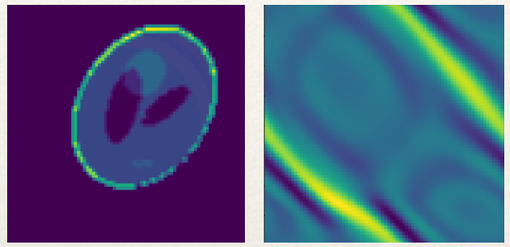


model

$\approx \quad f^{-1} : m \mapsto a$

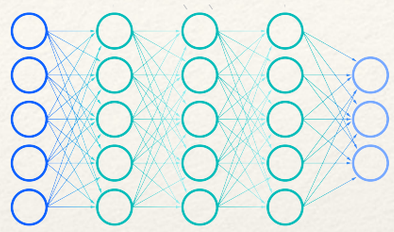
How to regularize the inverse problem

Learning forward PDE

data 

(a_i, m_i)

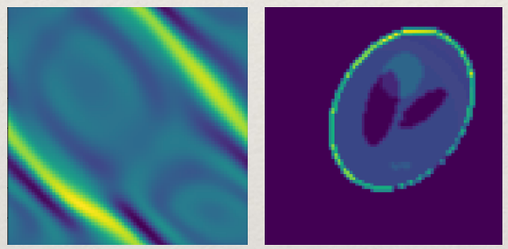
training



model

$\approx \quad f : a \mapsto m$

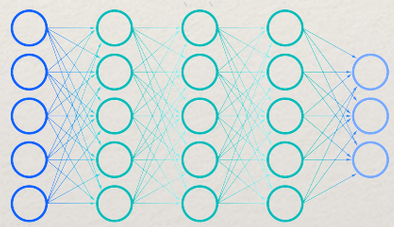
Learning inverse PDE

data 

(m_i, a_i)

training

penalize NN output



model

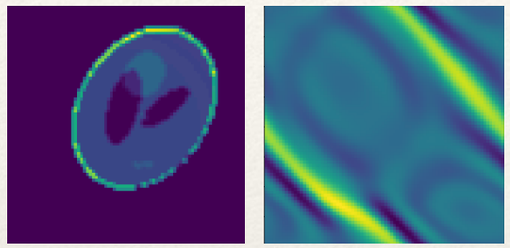
design special NN

$\approx \quad f^{-1} : m \mapsto a$

hard to learn by NN due to ill-posedness

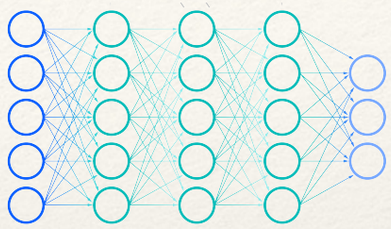
Training on regularized data

Learning forward PDE

data 

(a_i, m_i)

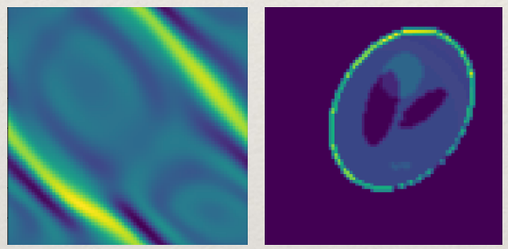
training



model

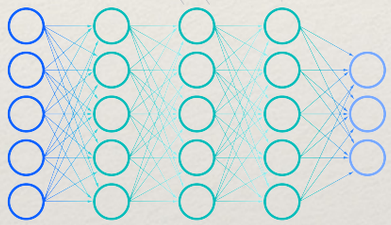
$\approx \quad f : a \mapsto m$

Learning inverse PDE

data 

(m_i, a_i)

training



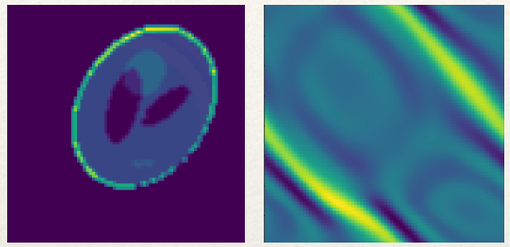
model

$\approx \quad f^{-1} : m \mapsto a$

Regularize the training data

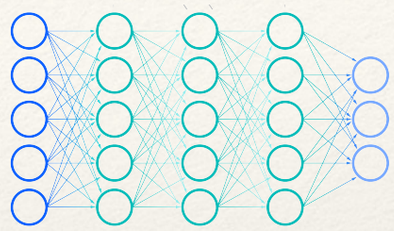
Data-Regularized Operator Learning

Learning forward PDE

data 

(a_i, m_i)

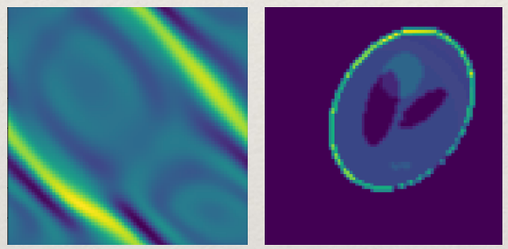
training



model

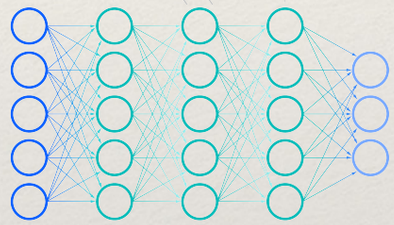
$\approx \quad f : a \mapsto m$

Learning inverse PDE

data 

(m_i, a_i)

training



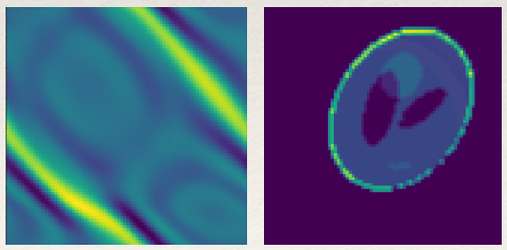
model

$\approx \quad f^{-1} : m \mapsto a$

hard to learn

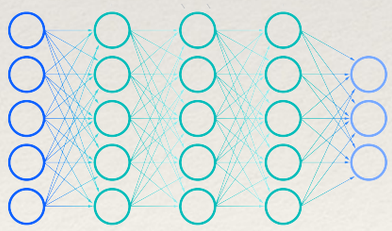
sometimes not well-defined

Learning regularized inverse PDE

data 

$(m_i, \hat{a}_i)$

training



model

$\approx f_{\text{reg}}^{-1} : m \mapsto \hat{a}$

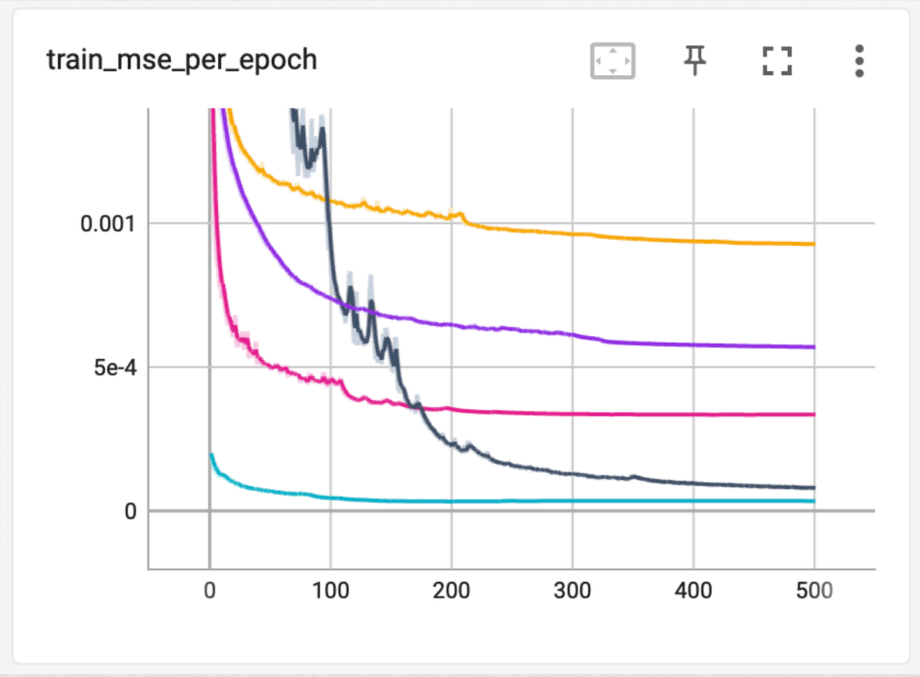
better regularity

easy to learn

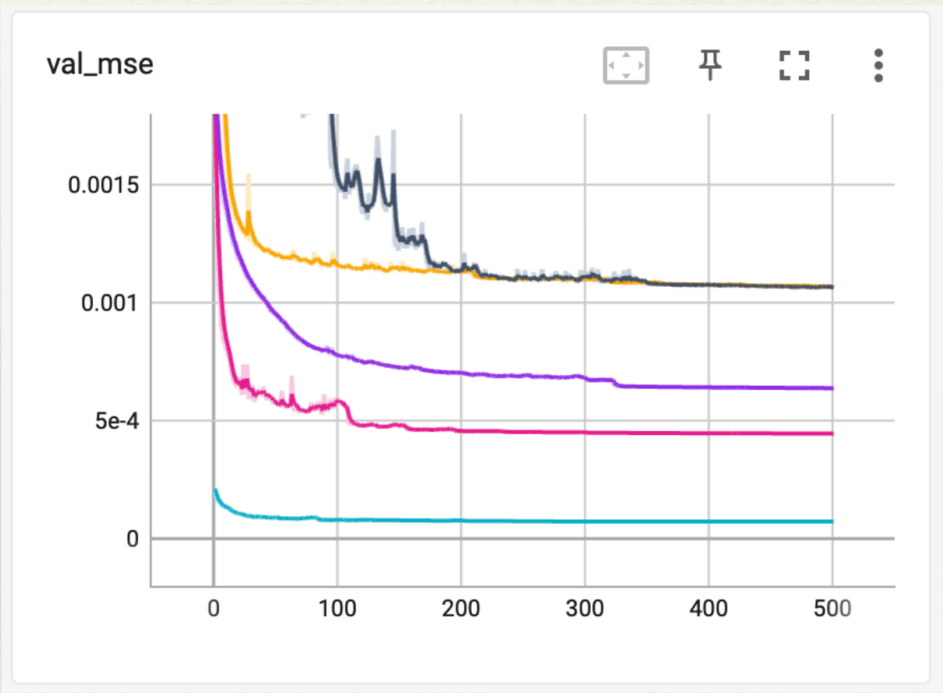
Learning with regularized data

Lasso as an example $\hat{x} = \arg\min \frac{1}{2}\|Ax - b\|^2 + \lambda\|x\|_1$.

target operator $f_{\text{reg}}^{-1} : b \mapsto \hat{x}$



training error



testing error

Black curve: learning on raw data

other curves: learning on various regularized data

learning on regularized data -> better generalization

Tikhonov regularization

Tikhonov Regularization

$$\hat{a} = \arg\min \|f(a) - m\|^2 + \lambda \mathcal{R}(a)\,.$$

Example: LASSO $$\hat{x} = \arg\min \frac{1}{2}\|Ax - b\|^2 + \lambda \|x\|_1\,.$$

Theorem. If the LASSO problem satisfies the non-degeneracy condition for all $\hat{x}$, then the regularized inverse $f_{\text{reg}}^{-1} : b \mapsto \hat{x}$ is Lipschitz with the following Lipschitz constant

$$L \leq \frac{\text{Cond}(A)}{\sigma_{\min}(A)} + \frac{1}{\sigma_{\min}^2(A)}\,.$$

Non-degeneracy condition: the LASSO problem is non-degenerate at $\hat{x}$ if
1. A_I has full column rank, $I = \text{supp}(\hat{x})$
2. $\|A_{I^C}^\top (b - A_I \hat{x}_I)\|_\infty < \lambda$

C, C. Wang, H. Yang 2023

Bayesian Inversion

Bayesian Regularization

$$q(a \mid m) = \frac{1}{C} p_0(a) \exp\left(-\frac{\|m - f(a)\|^2_{\Gamma_\varepsilon}}{2}\right),$$

Example: empirical mean estimate $\hat{a} := \mathbb{E}_{q(a \mid m)}[a]$

Maximum a posterior estimate $\hat{a} := \arg\max_a q(a \mid m)$

Theorem. Consider the regularized inverse $f^{-1}_{\text{reg}} : m \mapsto \hat{a}$, where $\hat{a}$ is the empirical mean, if

1. the measurement noises ε are independently sampled from Gaussian $\mathcal{N}(0, \Gamma_\varepsilon)$
2. the posterior $q(a|m)$ has a bounded second moment for all measurements
3. the forward map f is distinguishable, $\sup_a \left| J_f^{-1}(a) \right| \leq C_f$

Then the regularized inverse is Lipschitz with Lipschitz constant

$$L \leq C_\varepsilon B^{1/2} C_f^{1/2}$$

C, C. Wang, H. Yang 2023

Contributions & Summary

- In the forward problem setting, we theoretically explained why the low complexity and low dimension structure of PDE operators do not suffer from CoD.
- Instead of adding regularization in the training stage, we propose to train NN on a regularized data set for inverse problems
- We show that the regularized inverse map is Lipschitz for the LASSO regularization and Bayesian regularization (empirical mean estimate).

Thank you!

Questions