

Deep learning-based posterior inference for inverse problems

Deep Ray

Department of Aerospace & Mechanical Engineering
University of Southern California

Email: deepray@usc.edu

Website: deepray.github.io

Jointly with Harisankar Ramaswamy, Dhruv Patel, Assad Oberai

Supported by ARO grant W911NF2010050, NRL grant N00173-19-P-1119, AIER-USC, CARC-USC

In-House Annual Math Symposium
IISER Bhopal
March 26, 2022

- ▶ Inverse problems and Bayesian inference
- ▶ Deep neural networks
- ▶ Conditional generative adversarial networks (cGANs)
- ▶ Deep posteriors

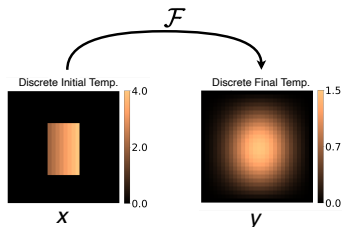
Consider a forward problem

$$\mathcal{F} : \mathbf{x} \in \Omega_x \mapsto \mathbf{y} \in \Omega_y, \quad \Omega_x \in \mathbb{R}^{N_x}, \quad \Omega_y \in \mathbb{R}^{N_y}$$

For example, the heat conduction PDE model for temperature field u

$$\begin{aligned} \frac{\partial u(\mathbf{s}, t)}{\partial t} - \nabla \cdot (\kappa(\mathbf{s}) \nabla u(\mathbf{s}, t)) &= f(\mathbf{s}), & \forall (\mathbf{s}, t) \in (0, 1)^2 \times (0, T] \\ u(\xi, 0) &= u_0(\mathbf{s}), & \forall \mathbf{s} \in (0, 1)^2 \\ u(\xi, t) &= 0, & \forall \mathbf{s} \in \partial(0, 1)^2 \times (0, T] \end{aligned}$$

Forward problem: Given $u_0(\mathbf{s})$ determine $u(\mathbf{s}, T)$



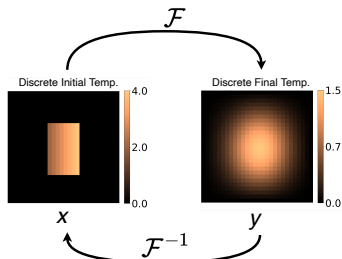
Consider a forward problem

$$\mathcal{F} : \mathbf{x} \in \Omega_x \mapsto \mathbf{y} \in \Omega_y, \quad \Omega_x \in \mathbb{R}^{N_x}, \quad \Omega_y \in \mathbb{R}^{N_y}$$

For example, the heat conduction PDE model for temperature field u

$$\begin{aligned} \frac{\partial u(\mathbf{s}, t)}{\partial t} - \nabla \cdot (\kappa(\mathbf{s}) \nabla u(\mathbf{s}, t)) &= f(\mathbf{s}), & \forall (\mathbf{s}, t) \in (0, 1)^2 \times (0, T] \\ u(\xi, 0) &= u_0(\mathbf{s}), & \forall \mathbf{s} \in (0, 1)^2 \\ u(\xi, t) &= 0, & \forall \mathbf{s} \in \partial(0, 1)^2 \times (0, T] \end{aligned}$$

Inverse problem: Given $u(\mathbf{s}, T)$ infer $u_0(\mathbf{s})$



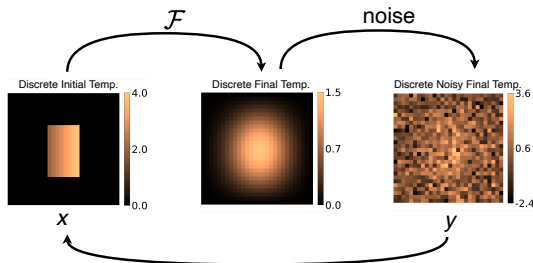
Consider a forward problem

$$\mathcal{F} : \mathbf{x} \in \Omega_x \mapsto \mathbf{y} \in \Omega_y, \quad \Omega_x \in \mathbb{R}^{N_x}, \quad \Omega_y \in \mathbb{R}^{N_y}$$

For example, the heat conduction PDE model for temperature field u

$$\begin{aligned} \frac{\partial u(\mathbf{s}, t)}{\partial t} - \nabla \cdot (\kappa(\mathbf{s}) \nabla u(\mathbf{s}, t)) &= f(\mathbf{s}), & \forall (\mathbf{s}, t) \in (0, 1)^2 \times (0, T] \\ u(\xi, 0) &= u_0(\mathbf{s}), & \forall \mathbf{s} \in (0, 1)^2 \\ u(\xi, t) &= 0, & \forall \mathbf{s} \in \partial(0, 1)^2 \times (0, T] \end{aligned}$$

Inverse problem: Given **noisy** $u(\mathbf{s}, T)$ infer $u_0(\mathbf{s})$

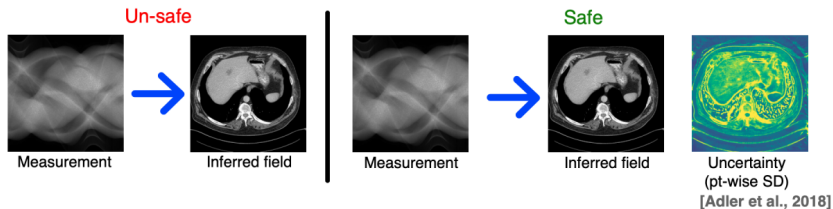


Challenges with inverse problems:

- ▶ Inverse map is **not well posed**.
- ▶ **Noisy measurements**.
- ▶ Need to encode **prior knowledge** about x .

Uncertainty in inferred field critical for applications with high-stake decisions.

Example: Medical imaging to detect liver lesions

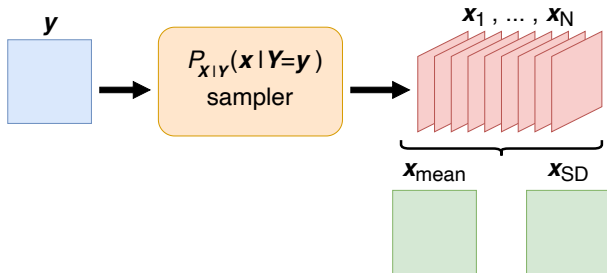


Assume \mathbf{x} and \mathbf{y} are modelled by random variables \mathbf{X} and \mathbf{Y} .

AIM: Given a measurement $\mathbf{Y} = \mathbf{y}$ approximate the conditional (posterior) distribution

$$P_{\mathbf{X}|\mathbf{Y}}(\mathbf{x}|\mathbf{Y} = \mathbf{y})$$

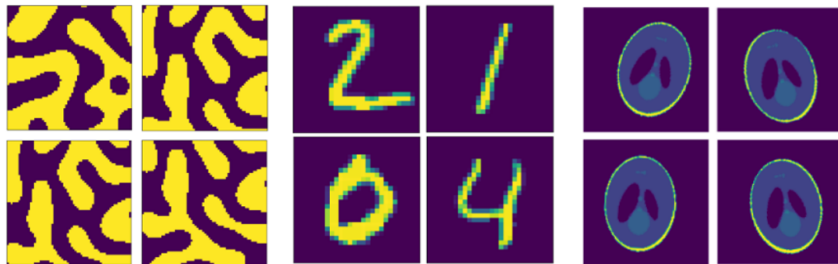
and sample from it.



Bayesian formulation: challenges

- ▶ Posterior sampling techniques, such as MCMC, are prohibitively expensive when N_x is large.
- ▶ Characterization of **priors for complex data**

Examples of prior data snapshots for \mathbf{x} :



Representing this data using simple distributions is **hard!**

Resolve both issues using deep learning

A neural network is a parametrized mapping

$$NN_{\theta} : \Omega_x \rightarrow \Omega_y$$

typically formed by alternating composition

$$NN_{\theta} := \mathcal{A}_{\theta_{L+1}}^{(L+1)} \circ \rho \circ \mathcal{A}_{\theta_L}^{(L)} \circ \rho \circ \mathcal{A}_{\theta_{L-1}}^{(L-1)} \circ \dots \circ \rho \circ \mathcal{A}_{\theta_1}^{(1)}$$

where

$\theta = \{\theta_k\}_{k=1}^L \rightarrow$ trainable weights and biases of the network

$\mathcal{A}_{\theta_k}^{(k)} \rightarrow$ parametrized affine transformation

$\rho \rightarrow$ non-linear activation function

Usage:

- ▶ Let \mathbf{x} and \mathbf{y} are related in some manner, say $\mathbf{y} = \mathbf{f}(\mathbf{x})$.
- ▶ We are only given $\mathcal{S} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$.
- ▶ NN_{θ} can be used to learn \mathbf{f} .

Consider a suitable measure

$$\mu : \Omega_x \times \Omega_y \times \Omega_x \times \Omega_y \rightarrow \mathbb{R}$$

s.t. $\mu(\mathbf{x}, \mathbf{y}, \underbrace{\mathbf{x}, NN_{\theta}(\mathbf{x})}_{=\hat{\mathbf{y}}})$ is the error/discrepancy between $(\mathbf{x}, \mathbf{y}) \in \mathcal{S}$ and $(\mathbf{x}, \hat{\mathbf{y}})$.

Define the loss/objective function

$$\Pi(\theta) = \frac{1}{N} \sum_{i=1}^N \mu(\mathbf{x}_i, \mathbf{y}_i, \mathbf{x}_i, NN_{\theta}(\mathbf{x}_i))$$

Solve the **non-convex** optimization problem

$$\theta^* = \arg \min_{\theta} \Pi(\theta)$$

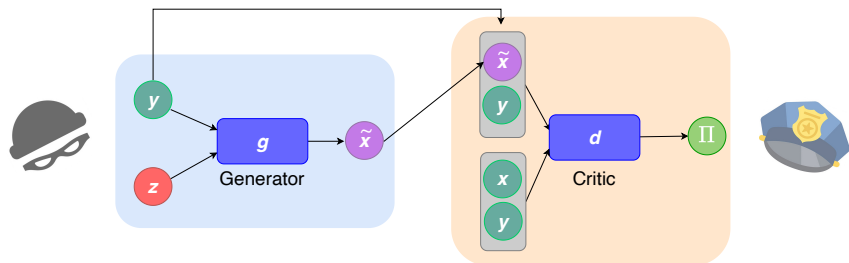
Then $NN_{\theta^*} \approx \mathbf{f}$

Also need to tune network **hyper-parameters**:

- Width
- Depth (L)
- Activation function ρ
- Optimizer
- Loss function
- Dataset

Conditional GANs

- ▶ Learning distributions conditioned on another field.
- ▶ Comprises two neural networks, g and d .
- ▶ Flipping role of \mathbf{x} and \mathbf{y} for inverse problems.



Generator network:

- ▶ $g : \Omega_z \times \Omega_y \rightarrow \Omega_x$.
- ▶ Latent variable $\mathbf{z} \sim P_z, N_z \ll N_x$.
- ▶ $(\mathbf{x}, \mathbf{y}) \sim P_{XY}$

Critic network:

- ▶ $d : \Omega_x \times \Omega_y \rightarrow \mathbb{R}$.
- ▶ $d(\mathbf{x}, \mathbf{y})$ large for real \mathbf{x} , small otherwise.

- ▶ Objective function

$$\Pi(\mathbf{g}, d) = \mathbb{E}_{\substack{(\mathbf{x}, \mathbf{y}) \sim P_{XY} \\ \mathbf{z} \sim P_Z}} [d(\mathbf{x}, \mathbf{y}) - d(\mathbf{g}(\mathbf{z}, \mathbf{y}), \mathbf{y})]$$

- ▶ \mathbf{g} and d determined (with constraint $\|d\|_{\text{Lip}} \leq 1$) through

$$(\mathbf{g}^*, d^*) = \arg \min_{\mathbf{g}} \arg \max_d \Pi(\mathbf{g}, d)$$

- ▶ Adler et al. (2018) proved that the minmax problem is equivalent to

$$\mathbf{g}^*(\cdot, \mathbf{y}) = \arg \min_{\mathbf{g}} W_1(P_{X|Y}, \mathbf{g}_{\#}(\cdot, \mathbf{y})P_Z) \quad \text{given } \mathbf{y} \sim p_Y$$

where W_1 is the Wasserstein-1 distance.

- ▶ Convergence in W_1 implies weak convergence

$$\mathbb{E}_{\mathbf{x} \sim P_{X|Y}} [\ell(\mathbf{x})] = \mathbb{E}_{\mathbf{z} \sim P_Z} [\ell(\mathbf{g}^*(\mathbf{z}, \mathbf{y}))], \quad \forall \ell \in C_b(\Omega_X).$$

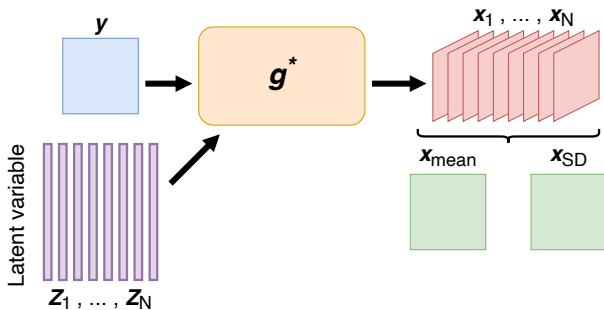
\implies conditional statistics converge!

Steps:

- ▶ Acquire samples $\mathcal{S}_x = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$, where $\mathbf{x}_i \sim P_X^{\text{prior}}$.
- ▶ Use forward map \mathcal{F} to generate paired dataset

$$\mathcal{S} = \{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_N, \mathbf{y}_N)\} \quad \text{where} \quad \mathbf{y}_n = \mathcal{F}(\mathbf{x}_n) + \text{noise.}$$

- ▶ Train a cGAN on \mathcal{S}
- ▶ Sample using g^* for a new test measurement \mathbf{y}
- ▶ Evaluate statistics using Monte Carlo



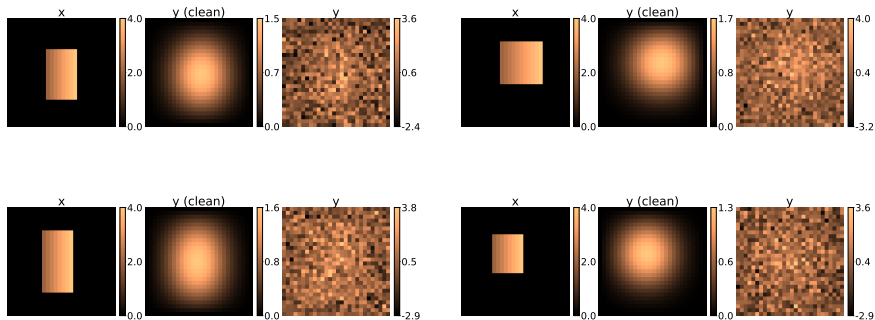
Consider the PDE

$$\begin{aligned}\frac{\partial u(\mathbf{s}, t)}{\partial t} - \nabla \cdot (\kappa(\mathbf{s}) \nabla u(\mathbf{s}, t)) &= f(\mathbf{s}), & \forall (\mathbf{s}, t) \in (0, 1)^2 \times (0, T] \\ u(\xi, 0) &= u_0(\mathbf{s}), & \forall \mathbf{s} \in (0, 1)^2 \\ u(\xi, t) &= 0, & \forall \mathbf{s} \in \partial(0, 1)^2 \times (0, T]\end{aligned}$$

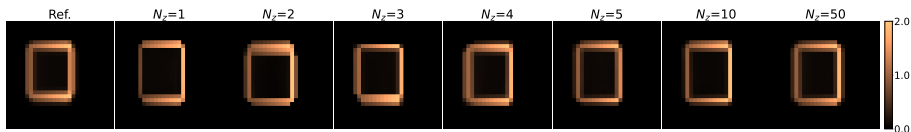
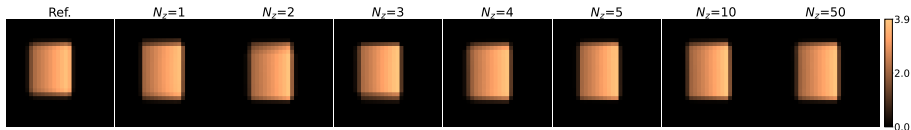
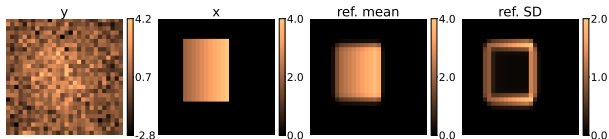
- ▶ \mathbf{x} : discrete initial temperature field.
- ▶ \mathbf{y} : noisy discrete final temperature field.
- ▶ \mathcal{F} : Finite difference solver for the PDE.
- ▶ We will assume a constant κ and f .

Assuming a \mathbf{x} to given by a rectangular inclusion and $N_x = N_y = 28 \times 28 = 784$

Training samples:

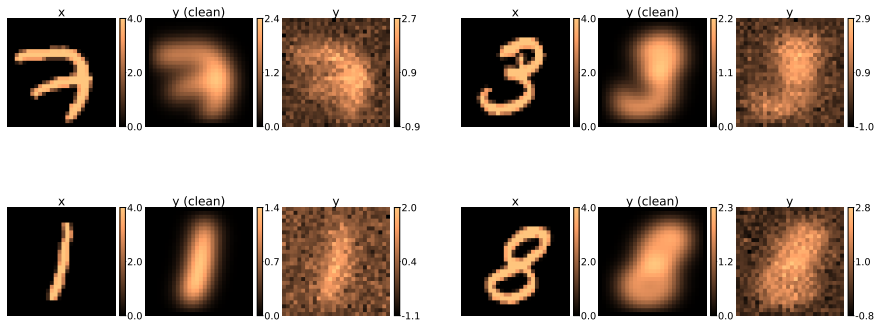


Testing trained cGAN (statistics with 800 \mathbf{z} samples)

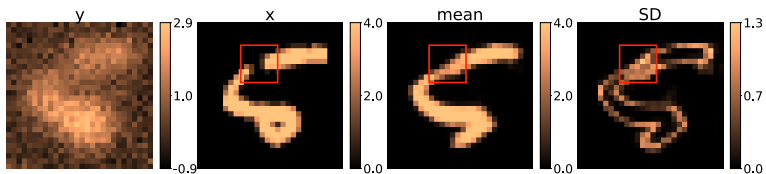
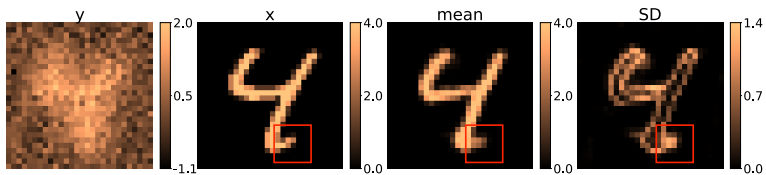
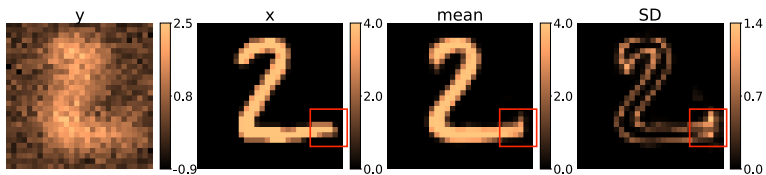


Assuming a \mathbf{x} to given by MNIST handwritten digits and $N_x = N_y = 28 \times 28 = 784$

Training samples:



Testing trained cGAN (statistics with 800 \mathbf{z} samples)

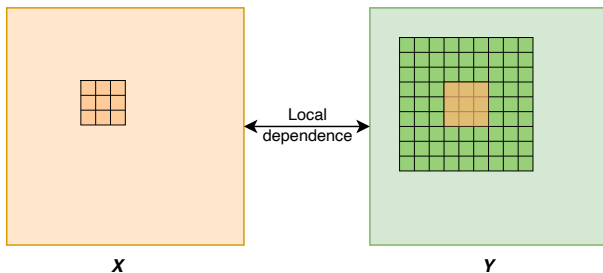


Generalization: Network trained on Set A gives good predictions on another distinct Set B, possibly sampled from a different distribution.

We can prove¹ the following: Assume

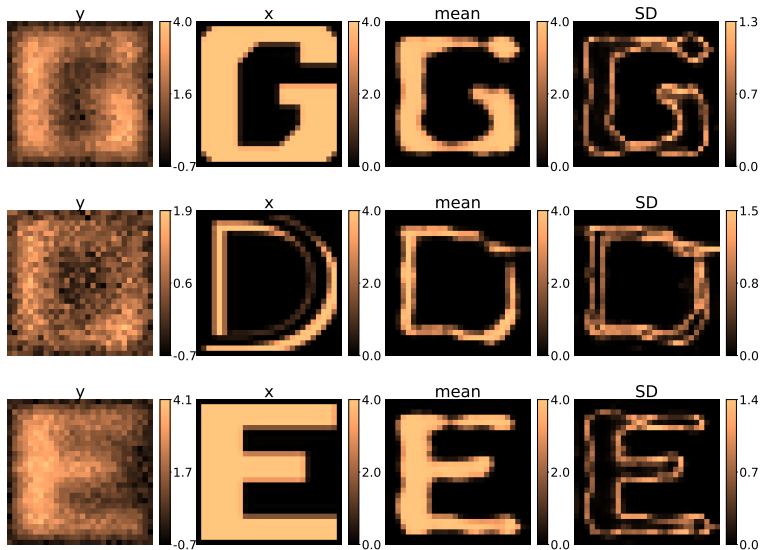
- ▶ The true (regularized) inverse map \mathcal{F}^{-1} is **spatially local**.
- ▶ Set A and Set B contain samples with **similar local spatial features**.
- ▶ A cGAN train on Set A, and is **also spatially local**.

Then the cGAN can generalize well to Set B.



1: *The efficacy and generalizability of conditional GANs for posterior inference in physics-based inverse problems* (D. Ray, D. Patel, H. Ramaswamy, A. A. Oberai); preprint 2022.

cGAN trained on MNIST, tested on notMNIST



- ▶ What do we gain?
 - ▶ Ability to represent and encode complex prior data.
 - ▶ Dimension reduction since $N_z \ll N_x$.
 - ▶ Sampling from cGAN is quick and easy.
- ▶ Generalizability – training on smaller dataset.
- ▶ Algorithm has been tested for many other physic-based applications.