Deep learning aided Bayesian inference

Deep Ray
Department of Aerospace & Mechanical Engineering
University of Southern California

Email: deepray@usc.edu
Website: deepray.github.io

REU Exposure Seminar
University of Maryland, July 27, 2022

USC Viterbi
School of Engineering

Outline

- Probability, Monte Carlo and Bayes theorem
- Forward, inverse problems and the Bayesian framework
- Neural networks
- Conditional generative adversarial networks (cGANs)
- Applications:
  - Academic example: heat equation
  - Real life example: MRI brain extraction

Probability basics

- X is an $\mathbb{R}$-valued random variable with distribution P_X, e.g. uniform $\mathcal{U}([a,b])$, Gaussian $N(\mu, \sigma^2)$.
- **Expectations:** For a function $f : \mathbb{R} \to \mathbb{R}$, the expectation of $f(X)$ is

$$\mathop{\mathbb{E}}_{X \sim P_X}[f(X)] = \int_{\mathbb{R}} f(x) P_X(x) \mathrm{d}x.$$

For $f(x) = x$, we get the mean

$$\mu_X := \mathop{\mathbb{E}}_{X \sim P_X}[X] = \int_{\mathbb{R}} x P_X(x) \mathrm{d}x.$$

For $f(x) = (x - \mu_X)^2$, we get the variance

$$Var(X) := \mathop{\mathbb{E}}_{X \sim P_X}\left[(X - \mu_X)^2\right] = \int_{\mathbb{R}} (x - \mu_X)^2 P_X(x) \mathrm{d}x.$$

The standard deviation is $SD(X) := \sqrt{Var(X)}$.

- $\boldsymbol{X} = (X_1, X_2)$ is an $\mathbb{R}^2$-valued (2 dimensional) random variable with the **joint** distribution $P_{\boldsymbol{X}} = P_{X_1 X_2}$, e.g. Gaussian $N(\boldsymbol{\mu}, \Sigma)$.
- **Expectations:** For a function $f : \mathbb{R}^2 \to \mathbb{R}$

$$\mathop{\mathbb{E}}_{\boldsymbol{X} \sim P_{\boldsymbol{X}}} [f(\boldsymbol{X})] = \int f(\boldsymbol{x}) P_{\boldsymbol{X}}(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}.$$

For a a vector-valued function $\boldsymbol{f} : \mathbb{R}^2 \to \mathbb{R}^k$ with $\boldsymbol{f}(\boldsymbol{x}) = [f_1(\boldsymbol{x}), \cdots, f_k(\boldsymbol{x})]$,

$$\mathop{\mathbb{E}}_{\boldsymbol{X} \sim P_{\boldsymbol{X}}} [\boldsymbol{f}(\boldsymbol{X})] = \left[\mathop{\mathbb{E}}_{\boldsymbol{X} \sim P_{\boldsymbol{X}}} [f_1(\boldsymbol{X})], \cdots, \mathop{\mathbb{E}}_{\boldsymbol{X} \sim P_{\boldsymbol{X}}} [f_k(\boldsymbol{X})] \right] \in \mathbb{R}^k.$$

In particular, $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{x}$ gives 2-dimensional mean

$$\mu_{\boldsymbol{X}} := \mathop{\mathbb{E}}_{\boldsymbol{X} \sim P_{\boldsymbol{X}}} [\boldsymbol{X}] = [\mu_1, \mu_2].$$

The covariance matrix $\Sigma_{\boldsymbol{X}} \in \mathbb{R}^{2\times 2}$ is given by

$$\Sigma_{\boldsymbol{X}} := \begin{bmatrix} Var(X_1) & Cov(X_1, X_2) \\ Cov(X_1, X_2) & Var(X_2) \end{bmatrix}.$$

Probability basics

- **Marginals:** The probability distributions of each component

$$P_{X_1}(x_1) = \int_{\mathbb{R}} P_{\mathbf{X}}(x_1, x_2)\mathrm{d}x_2, \quad P_{X_2}(x_2) = \int_{\mathbb{R}} P_{\mathbf{X}}(x_1, x_2)\mathrm{d}x_1.$$

- **Conditionals**: The conditional probability distribution of X_1 given $X_2 = x_2$

$$P_{X_1|X_2}(x_1|x_2) = \frac{P_{X_1 X_2}(x_1, x_2)}{P_{X_2}(x_2)}.$$

  Similarly, the conditional probability distribution of X_2 given $X_1 = x_1$

$$P_{X_2|X_1}(x_2|x_1) = \frac{P_{X_1 X_2}(x_1, x_2)}{P_{X_1}(x_1)}.$$

  The conditional expectation: given $X_2 = x_2$

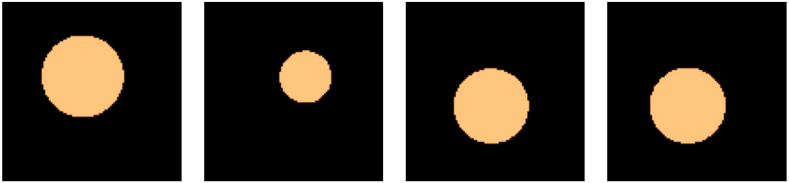
$$\mathop{\mathbb{E}}_{X_1 \sim P_{X_1|X_2}}[f(X_1)] = \int f(x_1) P_{X_1|X_2}(x_1|x_2)\mathrm{d}x_1.$$

- **Bayes' theorem:** Relates both conditionals

$$P_{X_1|X_2}(x_1|x_2) = \frac{P_{X_2|X_1}(x_2|x_1) P_{X_1}(x_1)}{P_{X_2}(x_2)}.$$

Images as random variables

Let $\mathcal{S} = \{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots\}$ be a set of images of shape $N \times N$. For example:



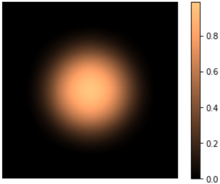
- Each image $\boldsymbol{x}^{(i)}$ can be seen as a realization of an N^2 dimensional random variable $\boldsymbol{X}$.
- Each pixel is a random variable X_i, for $1 \leq i \leq N^2$.
- We can calculate pixelwise mean and standard deviation of pixel intensity.

- Calculating $\underset{\boldsymbol{X}\sim P_{\boldsymbol{X}}}{\mathbb{E}}[f(\boldsymbol{X})] = \int f(\boldsymbol{x}) P_X(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}$ may be non-trivial.
- Approximate the integral using a suitable **quadrature**.
- Monte Carlo is one such strategy:
  - Choose i.i.d. samples $\{\boldsymbol{x}^{(i)}\}_{i=1}^{N}$
  - Calculate the estimator

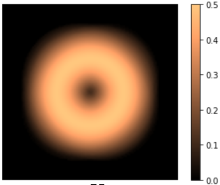
$$E_N[f] = \frac{1}{N}\sum_{i=1}^{N} f(\boldsymbol{x}^{(i)}) \approx \underset{\boldsymbol{X}\sim P_{\boldsymbol{X}}}{\mathbb{E}}[f(\boldsymbol{X})]$$

  - Error = $\mathcal{O}(\frac{1}{\sqrt{N}})$
- Useful when all we have is data

For example, with $N = 10,000$ random circle image



$\boldsymbol{X}_{mean}$



$\boldsymbol{X}_{SD}$

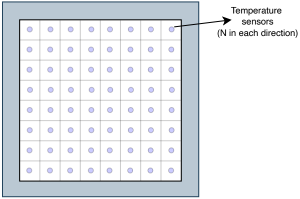
Consider the phenomena of heat flow through some material.

Modelled using the **heat equation**:

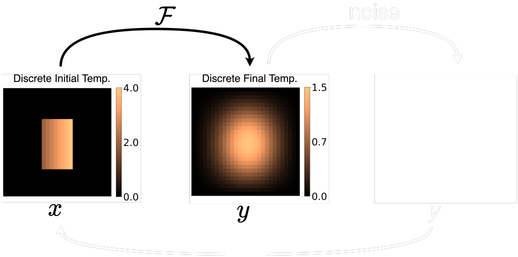
$$\frac{\partial u(\boldsymbol{s},t)}{\partial t} - \kappa \Delta u(\boldsymbol{s},t)) = 0$$
$$u(\boldsymbol{s},0) = u_0(\boldsymbol{s})$$

where

$u(\boldsymbol{s},t) \rightarrow$ temperature at location $\boldsymbol{s}$ at time t
$u_0(\boldsymbol{s}) \rightarrow$ initial temperature at location $\boldsymbol{s}$
$\kappa \rightarrow$ thermal conductivity of material



Forward problem $\mathcal{F}$: Given $u_0(\boldsymbol{s})$ at the sensor nodes determine $u(\boldsymbol{s},T)$



Consider the phenomena of heat flow through some material.

Modelled using the **heat equation**:

$$\frac{\partial u(\boldsymbol{s},t)}{\partial t} - \kappa \Delta u(\boldsymbol{s},t)) = 0$$

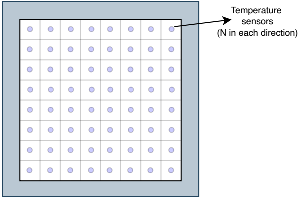
$$u(\boldsymbol{s},0) = u_0(\boldsymbol{s})$$

where

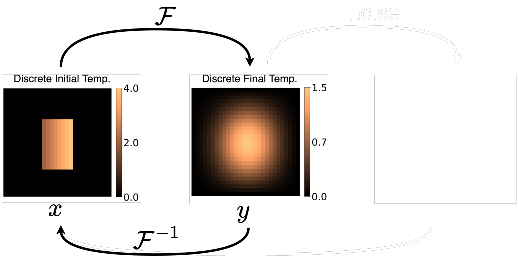
$u(\boldsymbol{s},t) \rightarrow$ temperature at location $\boldsymbol{s}$ at time t

$u_0(\boldsymbol{s}) \rightarrow$ initial temperature at location $\boldsymbol{s}$

$\kappa \rightarrow$ thermal conductivity of material



Inverse problem $\mathcal{F}^{-1}$: Given $u(\boldsymbol{s},T)$ at the sensor nodes infer $u_0(\boldsymbol{s})$



Forward and inverse problems

Consider the phenomena of heat flow through some material.

Modelled using the **heat equation**:

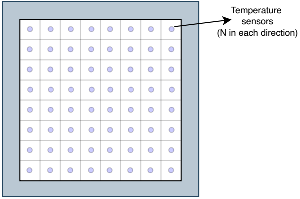
$$\frac{\partial u(\boldsymbol{s},t)}{\partial t} - \kappa \Delta u(\boldsymbol{s},t)) = 0$$
$$u(\boldsymbol{s},0) = u_0(\boldsymbol{s})$$

where

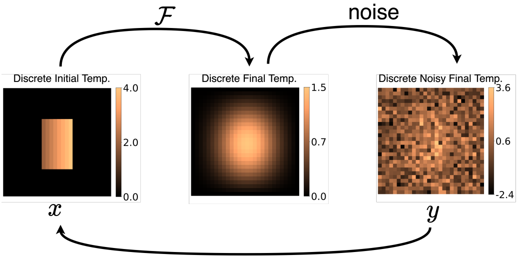
$u(\boldsymbol{s},t) \rightarrow$ temperature at location $\boldsymbol{s}$ at time t

$u_0(\boldsymbol{s}) \rightarrow$ initial temperature at location $\boldsymbol{s}$

$\kappa \rightarrow$ thermal conductivity of material



Inverse problem $\mathcal{F}^{-1}$: Given noisy $u(\boldsymbol{s},T)$ infer $u_0(\boldsymbol{s})$

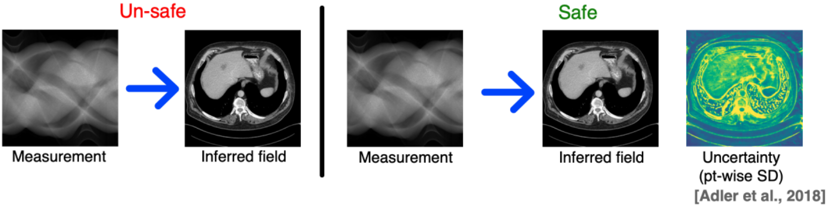


Challenges with inverse problems:

- Inverse map is not well posed.
- Noisy measurements.
- Need to encode prior knowledge about the inferred field x (e.g. initial temperature).

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

Example: Medical imaging to detect liver lesions



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

AIM: Given a measurement $\boldsymbol{Y} = \boldsymbol{y}$ approximate the conditional distribution

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

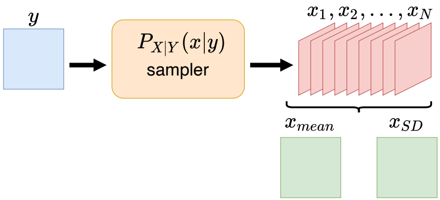
$P_{\boldsymbol{X}}(\boldsymbol{x})$: prior distribution

$P_{\boldsymbol{Y}}(\boldsymbol{y})$: evidence

$P_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{y}|\boldsymbol{x})$: likelihood of observing $\boldsymbol{y}$ given $\boldsymbol{x}$

$P_{\boldsymbol{X}|\boldsymbol{Y}}(\boldsymbol{x}|\boldsymbol{y})$: posterior distribution given $\boldsymbol{y}$

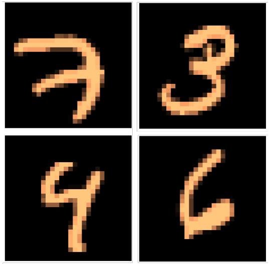
We want to generate samples:

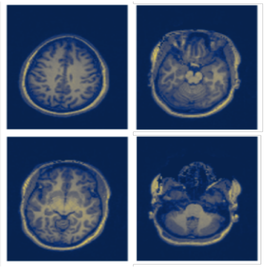


Bayesian formulation: challenges

- Posterior sampling techniques, such as Markov Chain Monte Carlo, are prohibitively expensive when dimension of $\boldsymbol{X}$ is large.
- Characterization of priors for complex data

For example, $\boldsymbol{x}$ data might look like:





Representing this data using simple distributions is hard!

Resolve both issues using deep learning

A neural network is a parametrized mapping

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

typically formed by alternating composition

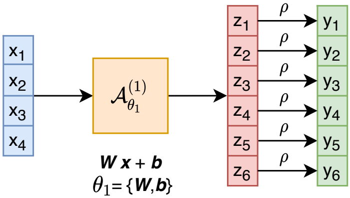
$$NN_{\boldsymbol{\theta}} := \rho \circ \mathcal{A}^{(L+1)}_{\boldsymbol{\theta}_{L+1}} \circ \rho \circ \mathcal{A}^{(L)}_{\boldsymbol{\theta}_L} \circ \rho \circ \mathcal{A}^{(L-1)}_{\boldsymbol{\theta}_{L-1}} \circ \cdots \circ \rho \circ \mathcal{A}^{(1)}_{\boldsymbol{\theta}_1}$$

where

$\boldsymbol{\theta} = \{\boldsymbol{\theta}_k\}_{k=1}^{L} \longrightarrow$ trainable weights and biases of the network

$\mathcal{A}^{(k)}_{\boldsymbol{\theta}_k} \longrightarrow$ parametrized affine transformation

$\rho \longrightarrow$ non-linear activation function



A neural network is a parametrized mapping

$$NN_{\boldsymbol{\theta}} : \Omega_x \to \Omega_y$$

typically formed by alternating composition

$$NN_{\boldsymbol{\theta}} := \rho \circ \mathcal{A}^{(L+1)}_{\boldsymbol{\theta}_{L+1}} \circ \rho \circ \mathcal{A}^{(L)}_{\boldsymbol{\theta}_{L}} \circ \rho \circ \mathcal{A}^{(L-1)}_{\boldsymbol{\theta}_{L-1}} \circ \cdots \circ \rho \circ \mathcal{A}^{(1)}_{\boldsymbol{\theta}_{1}}$$

where

$$\begin{aligned} \boldsymbol{\theta} = \{\boldsymbol{\theta}_k\}_{k=1}^{L} &\longrightarrow \text{ trainable weights and biases of the network} \\ \mathcal{A}^{(k)}_{\boldsymbol{\theta}_k} &\longrightarrow \text{ parametrized affine transformation} \\ \rho &\longrightarrow \text{ non-linear activation function} \end{aligned}$$

Usage:

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

Consider a suitable measure

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

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

Define the loss/objective function

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

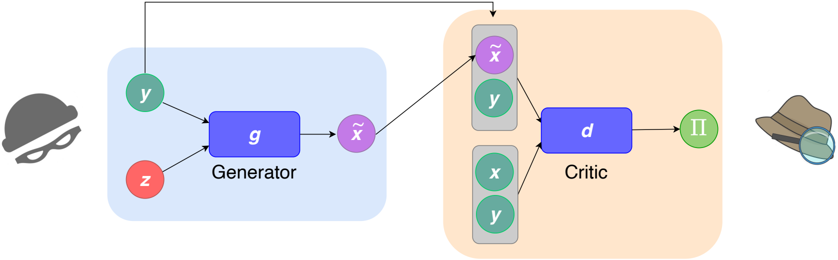
Solve the optimization problem

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

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

Also need to tune network hyper-parameters:
• Width • Depth (L) • Activation function ρ • Optimizer
• Loss function • Dataset

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



Generator network:

- $\boldsymbol{g} : \Omega_z \times \Omega_y \to \Omega_x$.
- Latent variable $\boldsymbol{Z} \sim P_{\boldsymbol{Z}}$, e.g. $N(0, \boldsymbol{I})$. Also $N_z \ll N_x$.
- $(\boldsymbol{x}, \boldsymbol{y})$ sampled from true P_{XY}

Critic network:

- $d : \Omega_x \times \Omega_y \to \mathbb{R}$.
- d tries to detect fake samples.
- $d(\boldsymbol{x}, \boldsymbol{y})$ large for real $\boldsymbol{x}$, small otherwise.

Conditional generative adversarial network (cGAN)

- Given $\boldsymbol{Y} = \boldsymbol{y}$ and $\boldsymbol{Z} \sim P_{\boldsymbol{Z}}$ we get a random variable

$$\boldsymbol{X}^{\boldsymbol{g}} = \boldsymbol{g}(\boldsymbol{Z}, \boldsymbol{y}), \quad \boldsymbol{X}^{\boldsymbol{g}} \sim P^{\boldsymbol{g}}_{\boldsymbol{X}|\boldsymbol{Y}}.$$

- Objective function

$$\Pi(\boldsymbol{g}, d) = \mathop{\mathbb{E}}_{\substack{(\boldsymbol{X},\boldsymbol{Y})\sim P_{\boldsymbol{X}\boldsymbol{Y}} \\ \boldsymbol{Z}\sim P_{\boldsymbol{Z}}}} \left[d(\boldsymbol{X}, \boldsymbol{Y}) - d\big(\boldsymbol{g}(\boldsymbol{Z}, \boldsymbol{Y}), \boldsymbol{Y}\big) \right]$$

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

$$(\boldsymbol{g}^*, d^*) = \arg\min_{\boldsymbol{g}} \; \arg\max_{d} \Pi(\boldsymbol{g}, d)$$

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

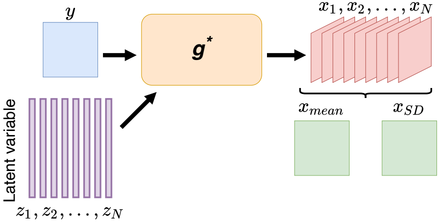
$$\boldsymbol{g}^* = \arg\min_{\boldsymbol{g}} \mathop{\mathbb{E}}_{\boldsymbol{Y}\sim P_{\boldsymbol{Y}}} \left[W_1(P_{\boldsymbol{X}|\boldsymbol{Y}}, P^{\boldsymbol{g}}_{\boldsymbol{X}|\boldsymbol{Y}}) \right]$$

where W_1 is the Wasserstein-1 distance.

Posterior sampling using cGANs

Steps:

- Acquire samples $\{\boldsymbol{x}_1, ..., \boldsymbol{x}_N\}$, where $\boldsymbol{x}_i$ are sampled from $P_{\boldsymbol{X}}$.
- For each $\boldsymbol{x}_i$ acquire the measurement $\boldsymbol{y}_i$ (using $\mathcal{F}$). $\boldsymbol{y}_i$ will be a sample from $P_{\boldsymbol{Y}|\boldsymbol{X}}$
- Construct the paired set $\mathcal{S} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), ..., (\boldsymbol{x}_N, \boldsymbol{y}_N)\}$. $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ can be seen as sampled from true joint $P_{\boldsymbol{X}\boldsymbol{Y}}$.
- Train a cGAN on $\mathcal{S}$.
- For a new test measurement $\boldsymbol{y}$, generate samples using $\boldsymbol{g}^*$.
- Evaluate statistics using Monte Carlo

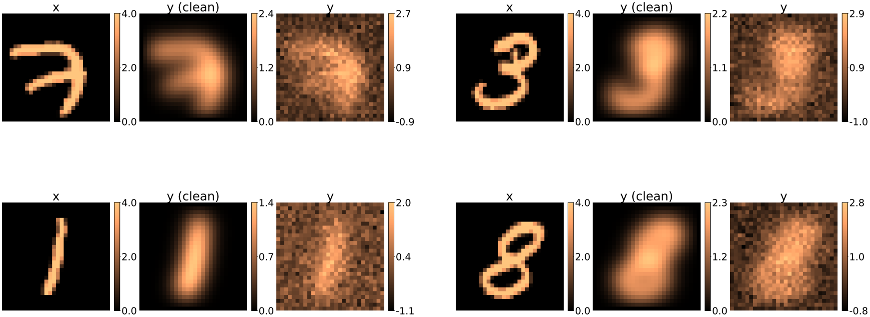


- We want to solve the inverse heat conduction problem.
- Assume the space has $N_x = 28 \times 28 = 784$ sensors.
- $\boldsymbol{X}$: Initial temperature at the sensors.
- $\boldsymbol{Y}$: Noisy final temperature at the sensors.
- We assume $\kappa = 0.2$
- $(\boldsymbol{x}, \boldsymbol{y})$ pairs obtained by numerically solving the forward problem $\mathcal{F}$ and artificially adding noise.
- We need to assume some prior distribution on $\boldsymbol{X}$.

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.

Assuming $\boldsymbol{X}$ to given by MNIST handwritten digits: u_0 takes the value 4.0 on the digit and 0.0 everywhere else.

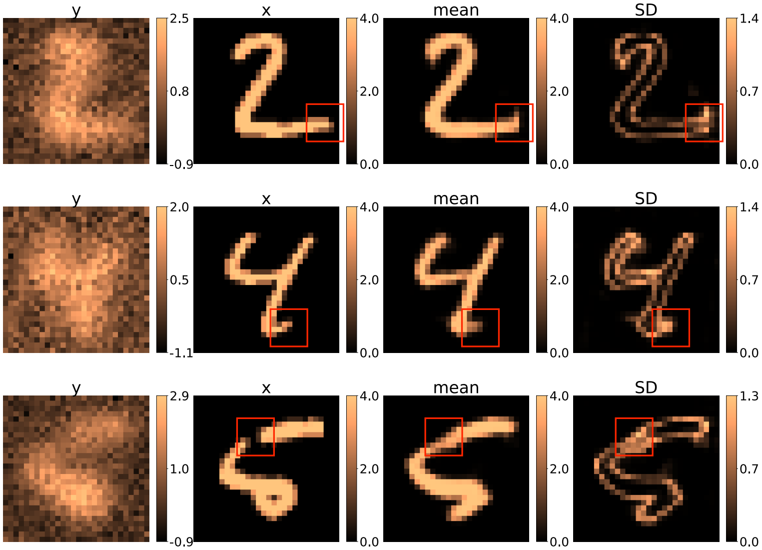
Training samples:



- Trained a network with dimension $N_z = 100$ for latent variable $\boldsymbol{Z}$. Note that $N_x = 784$.
- We don't have clean y in real problems!

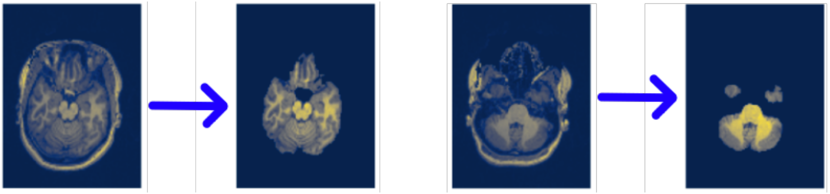
Inverse heat equation: testing

Testing trained generator g^*. Monte Carlo statistics with 800 z samples.



MRI brain extraction problem

- MRI images of the head used in several downstream task:
  - Quantifying grey and white matter
  - Monitoring neurological diseases, e.g., Alzheimer's
  - Estimating brain athrophy
- MRI images need to undergo ”skull stripping”: eliminating everything other than the brain.
- Manual process can be tedious and subjective.
- Existing algorithms do not quantify uncertainty.



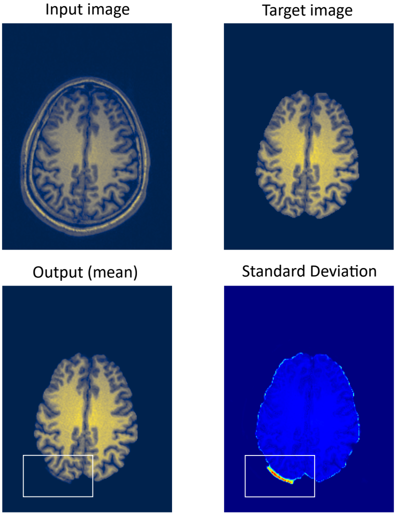
MRI brain extraction problem

Setup

- $\boldsymbol{X}$: Image of the brain (what we infer)
- $\boldsymbol{Y}$: Image of the head: original MRI
- Each image has shape $256 \times 192 \implies 49,152$ pixels
- Publicly available NFBS paired dataset used for training (no explicit $\mathcal{F}$)
- cGAN trained with latent dimension $N_z = 256$.

Probabilistic Brain Extraction in MR Images via Conditional Generative Adversarial Networks (A. Moazami, D. Ray, D. Pelletier, A. A. Oberai); preprint 2022.

Testing on a new sample



Conclusion

- Conditional GANs can be used to approximate and sample from conditional distributions.
- Required paired data to train – supervised learning model.
- What do we gain?
  - Ability to represent and encode complex data.
  - Dimension reduction since $N_z \ll N_x$.

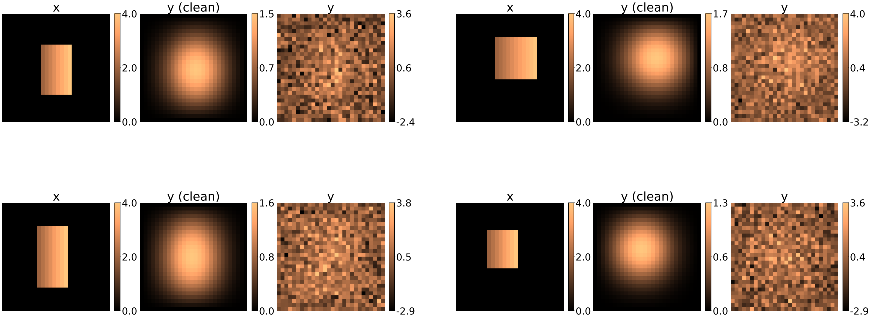
	N_x	N_z	**Dim. compression**
Inverse heat equation	784	100	7.84
Brain extraction	49,152	256	192

  - Once trained, sampling from cGAN is quick and easy.
- Algorithm tested for many other physics-based and medical applications.

Questions?

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

Training samples:



We never actually have clean $\boldsymbol{y}$!

Testing trained cGAN (statistics with 800 z samples)