Bayesian inference using Generative Adversarial Networks

Deep Ray
Department of Mathematics
University of Maryland, College Park

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

RIT: ML for Rare Events
March 31, 2023

UNIVERSITY OF MARYLAND

"We call two problems inverses of one another if the formulation of each involves all or part of the solution of the other. Often, for historical reasons, one of the two problems has been studied extensively for sometime, while the other is newer and not so well understood. In such cases, the former is called the direct problem, while the latter is called the inverse problem."

– Joseph Keller, 1976

Forward and inverse problems

Consider the phenomena of heat flow through some material.

Modelled using the **heat equation**:

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

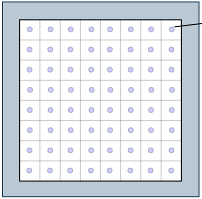
$$u(\boldsymbol{s},0) = u_0(\boldsymbol{s})$$

where

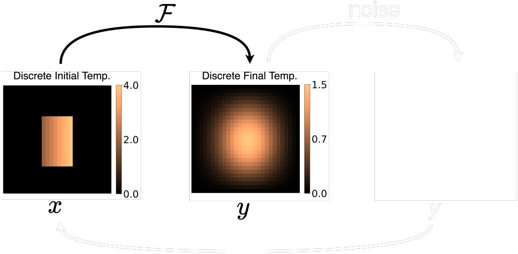
$u(\boldsymbol{s},t) \rightarrow$ temperature at location $\boldsymbol{s}$ at time t

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

$\kappa \rightarrow$ thermal conductivity of material



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**:

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

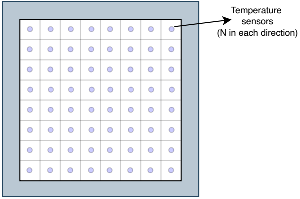
$$u(\boldsymbol{s},0) = u_0(\boldsymbol{s})$$

where

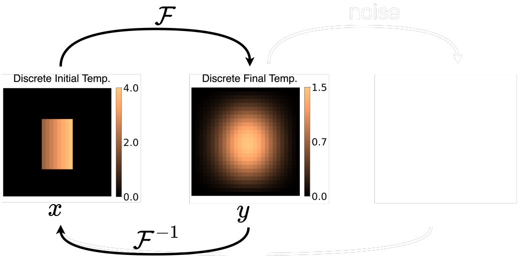
$u(\boldsymbol{s},t) \rightarrow$ temperature at location $\boldsymbol{s}$ at time t

$u_0(\boldsymbol{s}) \rightarrow$ 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 $u_0(\boldsymbol{s})$



Consider the phenomena of heat flow through some material.

Modelled using the **heat equation**:

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

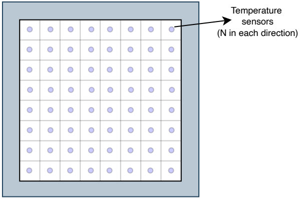
$$u(\boldsymbol{s},0) = u_0(\boldsymbol{s})$$

where

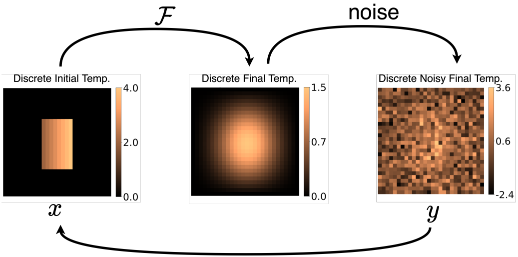
$u(\boldsymbol{s},t) \rightarrow$ temperature at location $\boldsymbol{s}$ at time t

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

$\kappa \rightarrow$ thermal conductivity of material



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

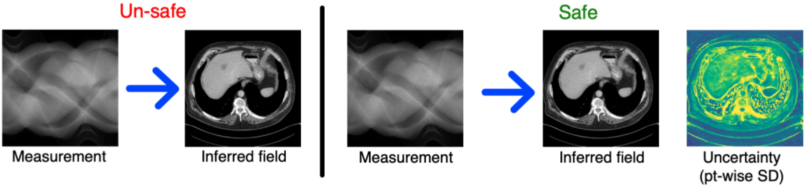


Challenges with inverse problems:

- Inverse map is not well posed.
- Noisy measurements from direct problem.
- Need to encode prior knowledge about inferred field.

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

Example: Medical imaging to detect liver lesions



[Adler et al., 2018]

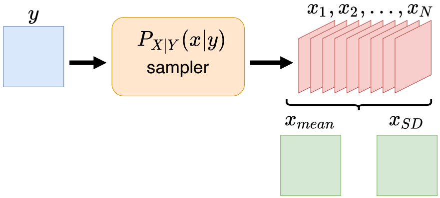
Assume $\boldsymbol{x} \in \Omega_X$ and $\boldsymbol{y} \in \Omega_Y$ are modeled by random variables $\boldsymbol{X}$ and $\boldsymbol{Y}$.

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 $\boldsymbol{y}$

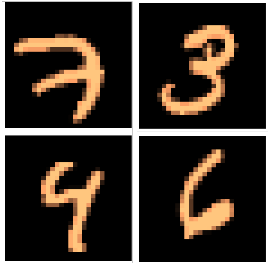
We want to generate samples:

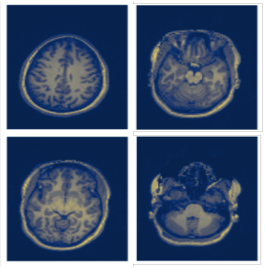


Bayesian framework: challenges

- 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:



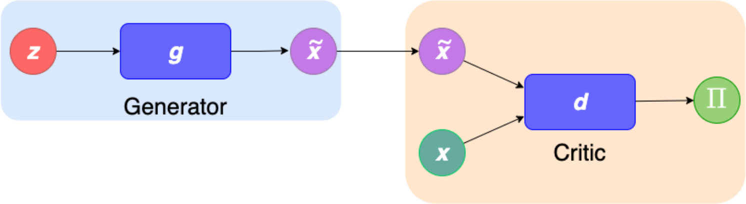


Representing this data using simple distributions is hard!

Resolve both issues using deep learning

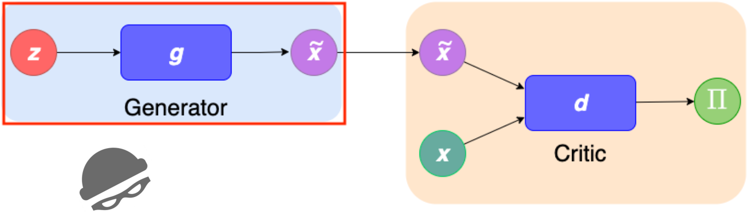
How do we learn a target (unknown) distribution $P_{\boldsymbol{X}}$ and sample from it?

Designed by Goodfellow et al. (2014).



Two networks with some suitable architectures.

Designed by Goodfellow et al. (2014).

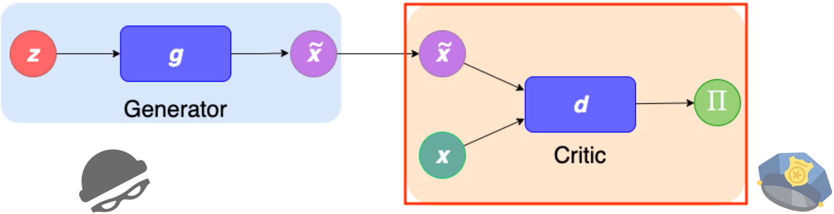


Two networks with some suitable architectures.

Generator network $\boldsymbol{g}$:

- Generates fake samples $\tilde{\boldsymbol{x}}$
- $\boldsymbol{g} : \Omega_Z \to \Omega_X$.
- Latent variable $\boldsymbol{z} \in \Omega_Z \subset \mathbb{R}^{N_z}$.
- $\boldsymbol{z} \sim P_Z$ simple distribution, e.g. Gaussian.
- $N_z \ll N_x$.

Designed by Goodfellow et al. (2014).



Two networks with some suitable architectures.

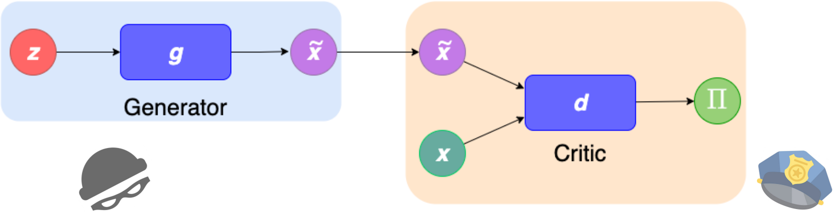
Generator network $\boldsymbol{g}$:

- Generates fake samples $\tilde{\boldsymbol{x}}$
- $\boldsymbol{g} : \Omega_Z \to \Omega_X$.
- Latent variable $\boldsymbol{z} \in \Omega_Z \subset \mathbb{R}^{N_z}$.
- $\boldsymbol{z} \sim P_Z$ simple distribution, e.g. Gaussian.
- $N_z \ll N_x$.

Critic network d:

- Distinguishes fake samples from real
- $d : \Omega_X \to \mathbb{R}$.
- $\boldsymbol{x} \sim P_X$.
- $d(\boldsymbol{x})$ large for $\boldsymbol{x} \sim P_X$, small otherwise.

Designed by Goodfellow et al. (2014).



Two networks with some suitable architectures.

Generator network $\boldsymbol{g}$:

- Generates fake samples $\tilde{\boldsymbol{x}}$
- $\boldsymbol{g} : \Omega_Z \to \Omega_X$.
- Latent variable $\boldsymbol{z} \in \Omega_Z \subset \mathbb{R}^{N_z}$.
- $\boldsymbol{z} \sim P_Z$ simple distribution, e.g. Gaussian.
- $N_z \ll N_x$.

Critic network d:

- Distinguishes fake samples from real
- $d : \Omega_X \to \mathbb{R}$.
- $\boldsymbol{x} \sim P_X$.
- $d(\boldsymbol{x})$ large for $\boldsymbol{x} \sim P_X$, small otherwise.

For a metric $\mathcal{M}$ on $\mathcal{P}(\Omega_X)$, define the loss

$$\Pi(\boldsymbol{g}, d) = \mathcal{M}(P_X, g_\# P_Z).$$

Solve the MinMax problem

$$(\boldsymbol{g}^*, d^*) = \arg\min_{\boldsymbol{g}} \arg\max_{d} \Pi(\boldsymbol{g}, d) \quad \longrightarrow \quad \text{Adversarial Training}$$

Proposed by Arjovsky et al. (2017), using the Wasserstein-1 metric

$$W_1(P_1, P_2) = \inf_{\gamma \in J(P_1, P_2)} \mathop{\mathbb{E}}_{(\boldsymbol{x}_1, \boldsymbol{x}_2) \sim \gamma} [\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|]$$

Using the Kantorovich-Rubinstein dual characterization, we have

$$W_1(P_1, P_2) = \sup_{\|f\|_{\mathsf{Lip}} \leq 1} \left(\mathop{\mathbb{E}}_{\boldsymbol{x} \sim P_1} [f(\boldsymbol{x})] - \mathop{\mathbb{E}}_{\boldsymbol{x} \sim P_2} [f(\boldsymbol{x})] \right)$$

Proposed by Arjovsky et al. (2017), using the Wasserstein-1 metric

$$W_1(P_1, P_2) = \inf_{\gamma \in J(P_1, P_2)} \mathop{\mathbb{E}}_{(\boldsymbol{x}_1, \boldsymbol{x}_2) \sim \gamma} [\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|]$$

Using the Kantorovich-Rubinstein dual characterization, we have

$$W_1(P_1, P_2) = \sup_{\|f\|_{\mathsf{Lip}} \leq 1} \left(\mathop{\mathbb{E}}_{\boldsymbol{x} \sim P_1} [f(\boldsymbol{x})] - \mathop{\mathbb{E}}_{\boldsymbol{x} \sim P_2} [f(\boldsymbol{x})] \right)$$

Set the loss function as

$$\Pi(\boldsymbol{g}, d) = \mathop{\mathbb{E}}_{\boldsymbol{x} \sim P_X} [d(\boldsymbol{x})] - \mathop{\mathbb{E}}_{\boldsymbol{z} \sim P_Z} [d(\boldsymbol{g}(\boldsymbol{z}))]$$

Under the constraint $\|d\|_{\mathsf{Lip}} \leq 1$, find

$$d^*(\boldsymbol{g}) = \arg\max_d \Pi(\boldsymbol{g}, d) \quad \Longrightarrow \quad \Pi(\boldsymbol{g}, d^*(\boldsymbol{g})) = W_1(P_X, \boldsymbol{g}_\# P_Z)$$

Proposed by Arjovsky et al. (2017), using the Wasserstein-1 metric

$$W_1(P_1, P_2) = \inf_{\gamma \in J(P_1, P_2)} \mathop{\mathbb{E}}_{(\boldsymbol{x}_1, \boldsymbol{x}_2) \sim \gamma} [\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|]$$

Using the Kantorovich-Rubinstein dual characterization, we have

$$W_1(P_1, P_2) = \sup_{\|f\|_{\mathsf{Lip}} \leq 1} \left(\mathop{\mathbb{E}}_{\boldsymbol{x} \sim P_1} [f(\boldsymbol{x})] - \mathop{\mathbb{E}}_{\boldsymbol{x} \sim P_2} [f(\boldsymbol{x})] \right)$$

Set the loss function as

$$\Pi(\boldsymbol{g}, d) = \mathop{\mathbb{E}}_{\boldsymbol{x} \sim P_X} [d(\boldsymbol{x})] - \mathop{\mathbb{E}}_{\boldsymbol{z} \sim P_Z} [d(\boldsymbol{g}(\boldsymbol{z}))]$$

Under the constraint $\|d\|_{\mathsf{Lip}} \leq 1$, find

$$d^*(\boldsymbol{g}) = \arg\max_d \Pi(\boldsymbol{g}, d) \quad \Longrightarrow \quad \Pi(\boldsymbol{g}, d^*(\boldsymbol{g})) = W_1(P_X, \boldsymbol{g}_\# P_Z)$$

Thus, for the optimal generator $\boldsymbol{g}^*$

$$\boldsymbol{g}^* = \arg\min_{\boldsymbol{g}} W_1(P_X, \boldsymbol{g}_\# P_Z)$$

Proposed by Arjovsky et al. (2017), using the Wasserstein-1 metric

$$W_1(P_1, P_2) = \inf_{\gamma \in J(P_1,P_2)} \mathop{\mathbb{E}}_{(\boldsymbol{x}_1,\boldsymbol{x}_2)\sim\gamma} [\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|]$$

Using the Kantorovich-Rubinstein dual characterization, we have

$$W_1(P_1, P_2) = \sup_{\|f\|_{\mathsf{Lip}} \leq 1} \left(\mathop{\mathbb{E}}_{\boldsymbol{x}\sim P_1} [f(\boldsymbol{x})] - \mathop{\mathbb{E}}_{\boldsymbol{x}\sim P_2} [f(\boldsymbol{x})] \right)$$

Set the loss function as

$$\Pi(\boldsymbol{g}, d) = \mathop{\mathbb{E}}_{\boldsymbol{x}\sim P_X} [d(\boldsymbol{x})] - \mathop{\mathbb{E}}_{\boldsymbol{z}\sim P_Z} [d(\boldsymbol{g}(\boldsymbol{z}))]$$

Under the constraint $\|d\|_{\mathsf{Lip}} \leq 1$, find

$$d^*(\boldsymbol{g}) = \arg\max_d \Pi(\boldsymbol{g}, d) \quad \Longrightarrow \quad \Pi(\boldsymbol{g}, d^*(\boldsymbol{g})) = W_1(P_X, \boldsymbol{g}_\# P_Z)$$

Thus, for the optimal generator $\boldsymbol{g}^*$

$$\boldsymbol{g}^* = \arg\min_{\boldsymbol{g}} W_1(P_X, \boldsymbol{g}_\# P_Z)$$

If for a sequence $\{g_n^*\}$, $W_1(P_X, \boldsymbol{g}^*_{n\#} P_Z) \to 0 \implies$ weak convergence of measures

$$\mathop{\mathbb{E}}_{\boldsymbol{z}\sim P_Z} [\ell(\boldsymbol{g}_n^*(\boldsymbol{z}))] \to \mathop{\mathbb{E}}_{\boldsymbol{x}\sim P_X} [\ell(\boldsymbol{x})], \quad \forall\, \ell \in C_b(\Omega_X)$$

$\longrightarrow$ moments converge.

In practice, at the discrete level

- Generate/obtain the finite dataset $\mathcal{S} = \{\boldsymbol{x}_i : \boldsymbol{x}_i \in \Omega_X,\ 1 \leq i \leq n\}$.
- Compute expectations using Monte Carlo

$$\mathop{\mathbb{E}}_{\boldsymbol{x} \sim P_X}[d(\boldsymbol{x})] \approx \frac{1}{n}\sum_{i=1}^{n} d(\boldsymbol{x}_i), \qquad \mathop{\mathbb{E}}_{\boldsymbol{z} \sim P_Z}[d(\boldsymbol{g}(\boldsymbol{z}))] \approx \frac{1}{n}\sum_{i=1, \boldsymbol{z}_i \sim P_Z}^{n} d(\boldsymbol{g}(\boldsymbol{z}_i))$$

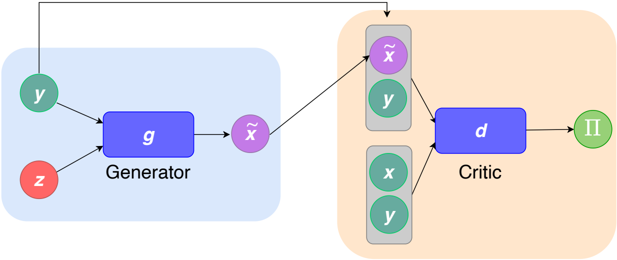
- Iterative solve the MinMax problem:
  - Take N (typically $N \geq 4$) optimization steps for d
  - Take 1 optimization step for $\boldsymbol{g}$
- Add a gradient penalty term (Gulrajani, 2017) to constraint d to be 1-Lipschitz

$$\lambda \frac{1}{n}\sum_{j=1}^{n}(\|\nabla_{\boldsymbol{x}} d(\boldsymbol{x}_j)\| - 1)^2$$

Results by Karras et al. (2018) from NVIDIA.
CELEBA-HQ dataset, $N_z = 512$, $N_x = 1024 \times 1024 \times 3 = 3.14 \times 10^6$
$\longrightarrow$ dimension reduction!

But we want to learn a conditional distribution $P_{\boldsymbol{X}|\boldsymbol{Y}}(\boldsymbol{x}|\boldsymbol{y})$.
Can we still use a GAN?

Learning distributions conditioned on another field.



Generator network:

- $\boldsymbol{g} : \Omega_Z \times \Omega_Y \to \Omega_X$.
- Also takes measurement $\boldsymbol{y}$ as input.
- $(\boldsymbol{x}, \boldsymbol{y})$ 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

$$\boldsymbol{X}^{\boldsymbol{g}} = \boldsymbol{g}(\boldsymbol{Z}, \boldsymbol{y}), \text{ with disitribution } P^{\boldsymbol{g}}_{\boldsymbol{X}|\boldsymbol{Y}} = \boldsymbol{g}(., \boldsymbol{y})_{\#} P_{\boldsymbol{Z}}.$$

- 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

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

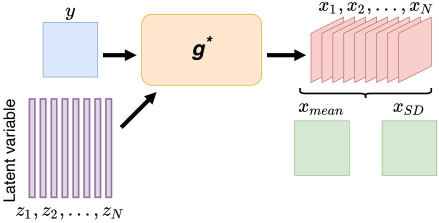
- Adler et al. (2018) proved that the minmax problem is equivalent to

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

where W_1 is the Wasserstein-1 distance.

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



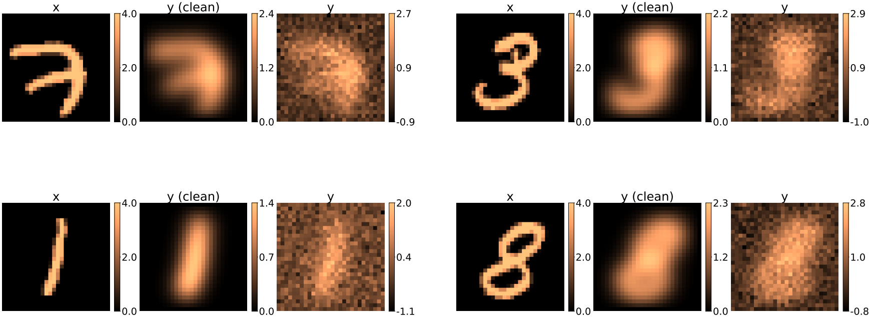
How well does it work for physics-based problems?

- 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 (an image).
- $\boldsymbol{Y}$: Noisy final temperature at the sensors (also an image).
- 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.

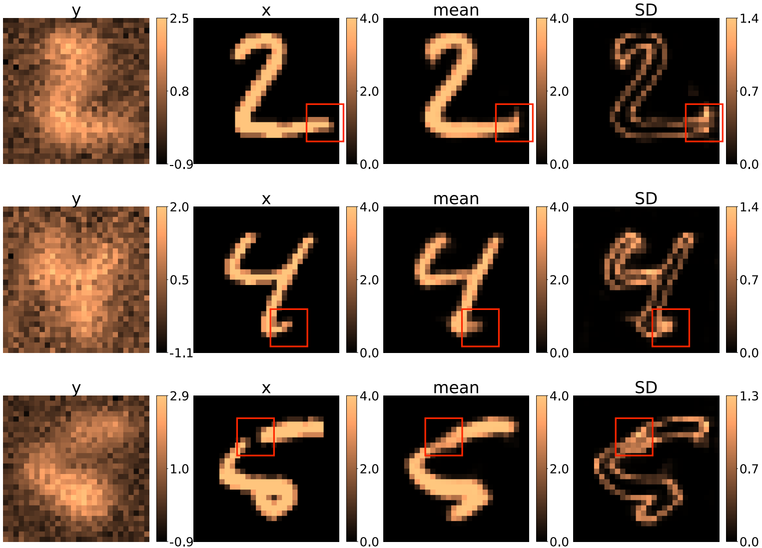
Assuming $\boldsymbol{X}$ to given by MNIST handwritten digits: u_0 takes the value 4.0 on the digit and 0.0 everywhere else.

Training samples:



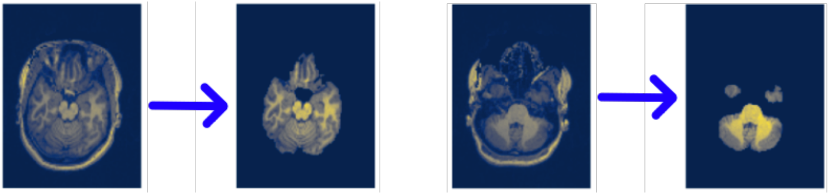
- Trained a network with dimension $N_z = 100$ for latent variable $\boldsymbol{Z}$. Note that $N_x = 784$.
- We don't have clean y in real problems!

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.

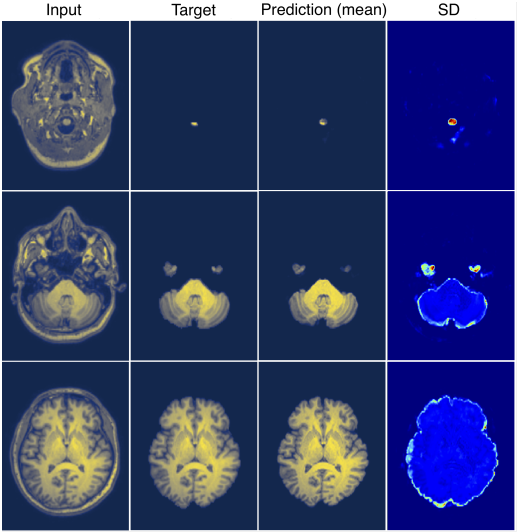


Setup

- $\boldsymbol{Y}$: Image of the head: original MRI
- $\boldsymbol{X}$: Image of the brain (what we infer)
- 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 unseen head images



De-raining images

- Removal of weather artifacts (rain, mist, etc) is critical in autonomous driving/aviation systems.
- $\boldsymbol{Y}$: Rainy image
- $\boldsymbol{X}$: De-rained/clean image
- Sufficient real clean/rainy images not available.
- Synthetic pairs generated through another GAN.
- cGAN trained on images of size $256 \times 256 \implies 65,536$ pixels.
- Latent dimension $N_z = 100$.

De-raining images

Testing on real rainy images

Rainy

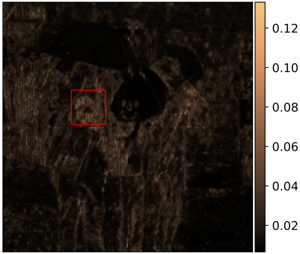
Real image of size 512 x 512.

De-rainer applied in 4 patches.

De-rainer unsure if letters are rain streaks or not.

De-rained

SD



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
De-raining	65,536	100	655.36

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

Questions?