Solving PDEs Using Conditional Generative Models

Chinmay Hegde

October 24, 2019

Iowa State → NYU

Today's talk

PDEs

Today's talk

PDEs

Generative Models

Today's talk

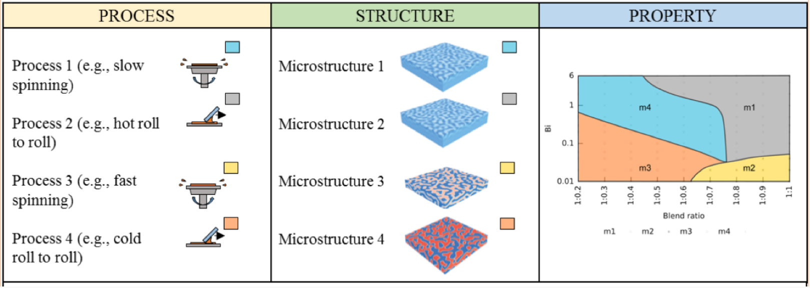
PDEs

Generative Models

Conditional

Application: Materials design

Materials design → exploring the space of plausible microstructures



Expensive, time-consuming; need to create **digital twin**

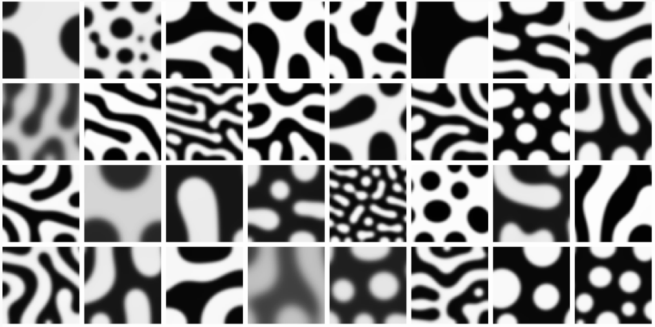
Application: Materials design

However, simulations are themselves extremely slow and costly.

Application: Materials design

However, simulations are themselves extremely slow and costly.

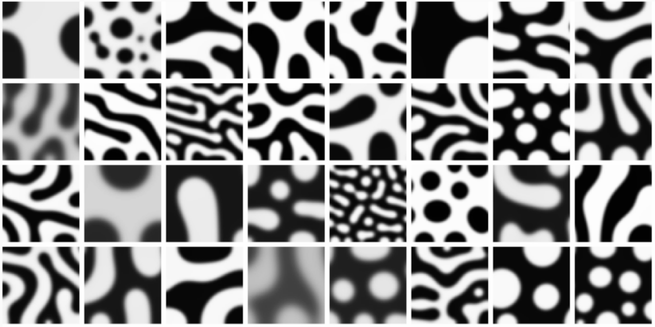
Ex: Two-component fluid mixtures that undergo a phase separation:
Solve the Cahn-Hilliard Equation $\rightarrow$ 4th order nonlinear PDE



Application: Materials design

However, simulations are themselves extremely slow and costly.

Ex: Two-component fluid mixtures that undergo a phase separation:
Solve the Cahn-Hilliard Equation $\rightarrow$ 4th order nonlinear PDE



Question: **Are there alternatives** to numerically solving PDEs?

Inverse problems

Canonical challenge in machine learning, scientific computing, engineering design

Inverse problems

Canonical challenge in machine learning, scientific computing, engineering design

$$y = \mathcal{A}(x) + \text{noise}.$$

Inverse problems

Canonical challenge in machine learning, scientific computing, engineering design

Parameters → Forward model → Observations

$$y = \mathcal{A}(x) + \text{noise}.$$

$\mathcal{A}$: given by nature, or manually designed

Inverse problems

Canonical challenge in machine learning, scientific computing, engineering design

$$y = \mathcal{A}(x) + \text{noise}.$$

$\mathcal{A}$: given by nature, or manually designed

Goal: Given y and (possibly) $\mathcal{A}$, recover an estimate of x.

Inverse problems

Canonical challenge in machine learning, scientific computing, engineering design

Parametersl → Forward model → Observations

Imaging / CV

- Imaging
- (De)compression
- Confocal microscopy
- MRI/CT

Inverse problems

Canonical challenge in machine learning, scientific computing, engineering design

Imaging / CV

- Imaging
- (De)compression
- Confocal microscopy
- MRI/CT

Systems, design, prediction

- System identification
- Inverse design
- Prognostics
- NDE

Classical approach

Estimation typically ill-posed; additional information / priors necessary

Classical approach

Estimation typically ill-posed; additional information / priors necessary

Estimation typically posed in terms of (constrained) optimization:

$$\widehat{x} = \min_x F(x|y, \mathcal{A})$$
$$\text{s.t.} \quad x \in S$$

Classical approach

Estimation typically ill-posed; additional information / priors necessary

Estimation typically posed in terms of (constrained) optimization:

$$\widehat{x} = \min_x F(x|y, \mathcal{A})$$
$$\text{s.t. } \; x \in S$$

S denotes hypothesis class (prior) for "true" x:

Classical approach

Estimation typically ill-posed; additional information / priors necessary

Estimation typically posed in terms of (constrained) optimization:

$$\widehat{x} = \min_{x} F(x|y, \mathcal{A})$$
$$\text{s.t.} \;\; x \in S$$

S denotes hypothesis class (prior) for "true" x:

- Bounded total variation
- Smoothness
- RKHS
- Sparsity in a basis/dictionary (synthesis, analysis, ...)
- ...

Neural Priors

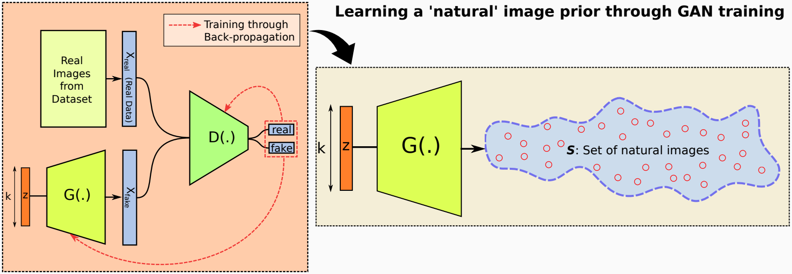
Given training data samples, learn a *neural generative prior* as the hypothesis class

Neural Priors

Given training data samples, learn a *neural generative prior* as the hypothesis class

Example prior: Generative Adversarial Networks (GANs)

[Goodfellow et al, 2014]



z: random variable; $G(z)$ is a convolutional neural network (CNN) that models the distribution of S

Promise of GANs

[Brock, Donahue, Simonyan, 2018]

BigGAN, $\dim(z) = 32$, $x = G(z)$, $\dim(x) = 256^2$

Promise of GANs

[Brock, Donahue, Simonyan, 2018]

BigGAN, dim$(z) = 32$, $x = G(z)$, dim$(x) = 256^2$

Using neural priors to solve PDEs

Intuition: Think of solving a PDE as a (non)linear mapping from one space (parameters/boundary conditions) to another space (domain)

Using neural priors to solve PDEs

Intuition: Think of solving a PDE as a (non)linear mapping from one space (parameters/boundary conditions) to another space (domain)

Why this is possible: Neural networks can approximate arbitrary high dimensional maps if architecture sufficiently wide/deep

[Cybenko '89], [Funahashi '89], [Hornik et al '89], [Kurkova '92]

Using neural priors to solve PDEs

Intuition: Think of solving a PDE as a (non)linear mapping from one space (parameters/boundary conditions) to another space (domain)

Why this is possible: Neural networks can approximate arbitrary high dimensional maps if architecture sufficiently wide/deep

[Cybenko '89], [Funahashi '89], [Hornik et al '89], [Kurkova '92]

More recent developments: Refinements to reasonable width but (very) deep networks

[Lu et al, '17], [Hardt-Ma '17], [Lin-Jegelka '18]

Using neural priors to solve PDEs

Intuition: Think of solving a PDE as a (non)linear mapping from one space (parameters/boundary conditions) to another space (domain)

Why this is possible: Neural networks can approximate arbitrary high dimensional maps if architecture sufficiently wide/deep

[Cybenko '89], [Funahashi '89], [Hornik et al '89], [Kurkova '92]

More recent developments: Refinements to reasonable width but (very) deep networks

[Lu et al, '17], [Hardt-Ma '17], [Lin-Jegelka '18]

Our approach: Train neural networks that can solve PDEs

Using neural priors to solve PDEs

Intuition: Think of solving a PDE as a (non)linear mapping from one space (parameters/boundary conditions) to another space (domain)

Why this is possible: Neural networks can approximate arbitrary high dimensional maps if architecture sufficiently wide/deep

[Cybenko '89], [Funahashi '89], [Hornik et al '89], [Kurkova '92]

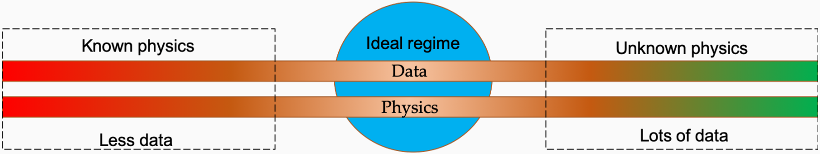
More recent developments: Refinements to reasonable width but (very) deep networks

[Lu et al, '17], [Hardt-Ma '17], [Lin-Jegelka '18]

Our approach: Train neural networks that can solve PDEs

But not in the standard way..

Learning based solutions to PDEs



[Raissi, Perdikaris, Karniadakis 2018]

[Yang, Zhang, Karniadakis 2018]

[Lu, Meng, Mao, Karniadakis 2019]

[Zhou, Zabaras, Koutsourelakis, Perdikaris 2019]

and many others ...

DiffNets: Formulation

Hybrid neural prior that combines data and physics

$$L_{\mathsf{Inv}}(W) = \|y - \mathcal{A}(x)\|^2, \quad x = G_W(z)$$

DiffNets: Formulation

Hybrid neural prior that combines data and physics

$$L_{\mathsf{Inv}}(W) = \|y - \mathcal{A}(x)\|^2, \quad x = G_W(z)$$

$$L_{\mathsf{GAN}}(W, \Psi) = \mathbb{E}_{x' \sim \mathbf{P}_{\mathsf{data}}}\left[\phi(D_\Psi(x'))\right] + \mathbb{E}_{x \sim \mathbf{P}_x}\left[\phi(-D_\Psi(x))\right]$$

Optimizing the min-max two-player game:

$$\min_W \max_\Psi L_{\mathsf{GAN}} + \mu L_{\mathsf{Inv}}$$

Example: Elliptic PDEs

Assume a stochastic elliptic PDE of the form:

$$\mathcal{A}(u) = f + \xi,$$

Example:

$$\nabla(K \circ \nabla(u)) = f + \xi, \quad \text{(Stochastic heat equation).}$$

Example: Elliptic PDEs

Assume a stochastic elliptic PDE of the form:

$$\mathcal{A}(u) = f + \xi,$$

Example:

$$\nabla(K \circ \nabla(u)) = f + \xi, \quad \text{(Stochastic heat equation)}.$$

Discretize (using finite differences):

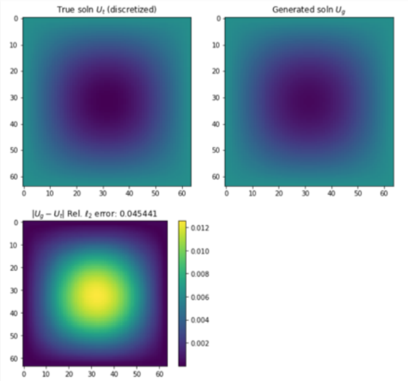
$$\nabla^2 u \approx \frac{1}{4h^2}\left(u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - 4u_{i,j}\right)$$

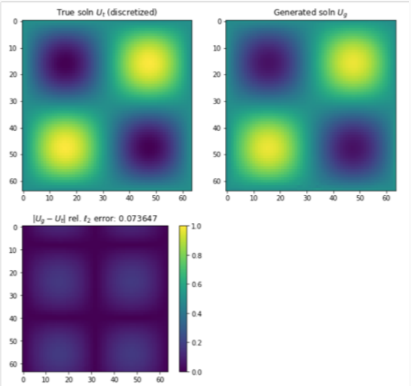
This gives a linear system of equations (assuming period BC):

$$(I - P_\Omega)Au = (I - P_\Omega)f + \xi,$$
$$P_\Omega u = P_\Omega b.$$

DiffNet: Elliptic PDEs

Periodic boundary conditions





Upshot: comparable accuracy as numerical solvers; solution for new boundary conditions only requires a forward pass through the generator CNN.

Theory: Linear PDEs

We can explicitly write down the solution space as:

$$u = \begin{bmatrix} A^{-1} P_{\Omega^c} f \\ P_{\Omega} b \end{bmatrix} + \begin{bmatrix} z \\ 0 \end{bmatrix} \tag{1}$$

where z is normally distributed with covariance $A^{-1} P_{\Omega^c} A^{(-1)^T}$.

Theory: Linear PDEs

We can explicitly write down the solution space as:

$$u = \begin{bmatrix} A^{-1} P_{\Omega^c} f \\ P_{\Omega} b \end{bmatrix} + \begin{bmatrix} z \\ 0 \end{bmatrix} \tag{1}$$

where z is normally distributed with covariance $A^{-1} P_{\Omega^c} A^{(-1)^T}$.

Setup:

- We do not know the form of the PDE (i.e., A is not known), but have access to training data sampled from distribution of u.
- We enforce boundary conditions as invariances
- Noise is gaussian

Theory: Linear PDEs

Generator: $z \mapsto \Theta z + \lambda$; Discriminator: $x \mapsto x^T P_{\Omega^c} \Psi P_{\Omega^c} x$

Solve: $\min_W \max_\Psi L_{\mathsf{GAN}} + \mu \mathbb{E}_{u \sim G_W(\mathcal{N})} \|P_\Omega(u - b)\|_2^2$.

Theory: Linear PDEs

Generator: $z \mapsto \Theta z + \lambda$; Discriminator: $x \mapsto x^T P_{\Omega^c} \Psi P_{\Omega^c} x$

Solve: $\min_W \max_\Psi L_{\mathsf{GAN}} + \mu \mathbb{E}_{u \sim G_W(\mathcal{N})} \|P_\Omega(u - b)\|_2^2$.

Theorem: At Nash equilibrium, we get:

$$\begin{aligned} \Psi_{\Omega^c} &= 0, \\ \Theta_\Omega &= 0, \\ \Theta_{\Omega^c} \Theta_{\Omega^c}^T &= A^{-1} P_{\Omega^c} A^{-1}. \end{aligned}$$

In other words: the generator

- provably learns unknown dynamics (inverse of $\mathcal{A}$) up to rotation
- provably enforces known constraints (boundary conditions)

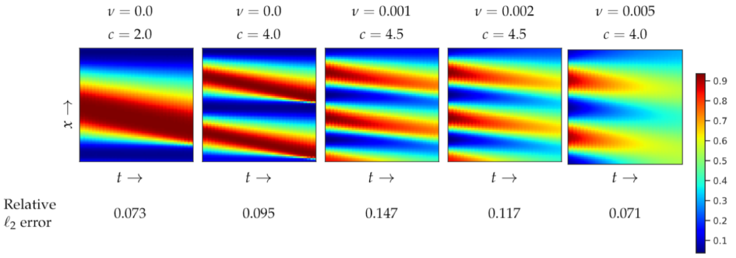
Solving nonlinear PDEs

Example: (In)viscid Burgers Equation

$$\frac{\partial u}{\partial t} + u \cdot \nabla u = (\nu + \xi)\nabla^2 u,$$
$$u_{t=0} = 1 - \cos \frac{2\pi c x}{L}$$

Example: (In)viscid Burgers Equation

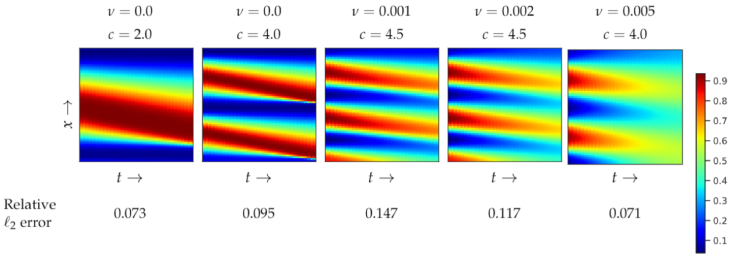
$$\frac{\partial u}{\partial t} + u \cdot \nabla u = (\nu + \xi)\nabla^2 u,$$
$$u_{t=0} = 1 - \cos \frac{2\pi c x}{L}$$



Solving nonlinear PDEs

Example: (In)viscid Burgers Equation

$$\frac{\partial u}{\partial t} + u \cdot \nabla u = (\nu + \xi)\nabla^2 u,$$
$$u_{t=0} = 1 - \cos \frac{2\pi c x}{L}$$



Upshot: Far quicker solutions (about 1500X speedup) than standard numerical solvers; effective surrogate in lieu of solving stochastic PDEs

Physics-aware conditional generative models

Going beyond PDEs: instead of enforcing PDE constraints, one can use other prior information (statistics, geometry, etc)

Physics-aware conditional generative models

Going beyond PDEs: instead of enforcing PDE constraints, one can use other prior information (statistics, geometry, etc)

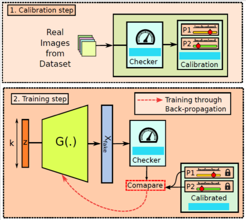
- Example invariance: mass fraction; $\alpha = \mathbf{E}_{\mathbf{r}} x(\mathbf{r})$
- Example invariance: 2-point correlation; $\beta(\mathbf{r}) = \mathbf{E}_{\mathbf{r_1},\mathbf{r_2}} x(\mathbf{r_1}) x(\mathbf{r_2})$
- Example invariance: connectivity or shape information

Physics-aware conditional generative models

Going beyond PDEs: instead of enforcing PDE constraints, one can use other prior information (statistics, geometry, etc)

- Example invariance: mass fraction; $\alpha = \mathbf{E}_\mathbf{r} x(\mathbf{r})$
- Example invariance: 2-point correlation; $\beta(\mathbf{r}) = \mathbf{E}_{\mathbf{r_1},\mathbf{r_2}} x(\mathbf{r_1}) x(\mathbf{r_2})$
- Example invariance: connectivity or shape information

Model: **Invariance Network** (InvNet). Same as above, but enforce above properties.



[Singh, Shah, G., Sarkar, H, NeurIPS WS 2018]

InvNets: Results

Invariance: P1 (volume fraction)

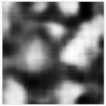
Desired: 0.42
Actual: 0.45

Desired: 0.52
Actual: 0.55

Desired: 0.61
Actual: 0.64

Desired: 0.72
Actual: 0.72

Invariance: 2 point correlation distance (in pixels)



Desired: 23 pix
Actual: 25 pix

Desired: 33 pix
Actual: 31 pix

Desired: 42 pix
Actual: 42 pix

Desired: 50 pix
Actual: 48 pix

InvNets: Results

Invariance: P1 (volume fraction)

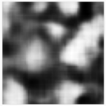
Desired: 0.42
Actual: 0.45

Desired: 0.52
Actual: 0.55

Desired: 0.61
Actual: 0.64

Desired: 0.72
Actual: 0.72

Invariance: 2 point correlation distance (in pixels)



Desired: 23 pix
Actual: 25 pix

Desired: 33 pix
Actual: 31 pix

Desired: 42 pix
Actual: 42 pix

Desired: 50 pix
Actual: 48 pix

Upshot: Far quicker microstructure reconstruction (nearly **1500X** speedup in amortized time)

Summary

This talk: Solving PDEs using neural networks.

- **Conditional generative models**
- **Wide Applicability**
- **(Preliminary) Theoretical guarantees**

Summary

This talk: Solving PDEs using neural networks.

- **Conditional generative models**
  Allow for user tuning of inputs/boundary conditions
- **Wide Applicability**
  Works for linear, non-linear PDEs
- **(Preliminary) Theoretical guarantees**
  Nash equilibrium provably learns the inverse

Open problems:

- Properly modeling grid mismatch (ideas from multi-scale modeling?)
- Faster training (ideas from PDE pre-conditioning?)
- Limited samples (ideas from transfer learning?)
- Guarantees (ideas from optimization and statistics?)

Thanks

Funded by the DARPA AIE (AIRA) Program; Dec 2018 - present.

- Baskar Ganapathysubramanian, Soumik Sarkar (ISU)
- Daniel Sparkman (AFRL)
- Payel Das, Youssef Mroueh (IBM)

Singh, Shah, Pokuri, Sarkar, G., H, "Physics-aware Deep Generative Models for Creating Synthetic Microstructures", Dec 2018

Shah, Joshi, Ghoshal, Pokuri, Sarkar, G., H, "Encoding Invariances in Deep Generative Models", June 2019

Joshi, Cho, Pokuri, Sarkar, G., H, "InvNet: Incorporating Statistical and Geometric Invariances in Generative Models", Sept 2019

Joshi, Shah, Sarkar, G., H, "Generative Models for Solving Partial Differential Equations", Dec 2019