Physics Informed Neural Networks (PINNs)

Srivatsa Kundurthy

Physics Informed Machine Learning for…

- Less computational cost
- Less complex formulas and algorithms
- Dealing with data uncertainty
- Faster processing for real-time situations
- DL: Automatic feature extraction when dealing with muli-fidelity data

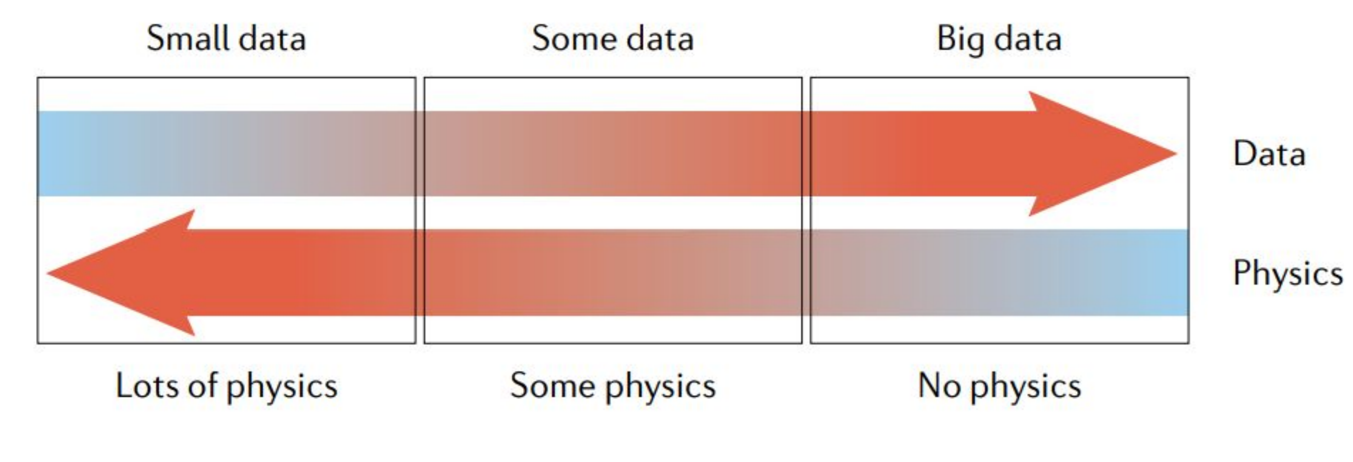
A Problem

- Machine Learning Models depending only on observational data may generalize poorly

The PINN Approach

PINNs – Physics Informed Neural Networks

- Mathematical models + observational data
- Generalize better due to embedded understanding of physical world
- Good for the “middle case”



How do we embed physical laws?

Observational Biases

- Data tends to reflect the underlying physical phenomena
- Learn functions that reflect data structure

Inductive Biases

- The learning algorithm makes certain assumptions
- Exploit network architecture to favor physical laws
- A direct approach? - Constraining network to physically viable solutions

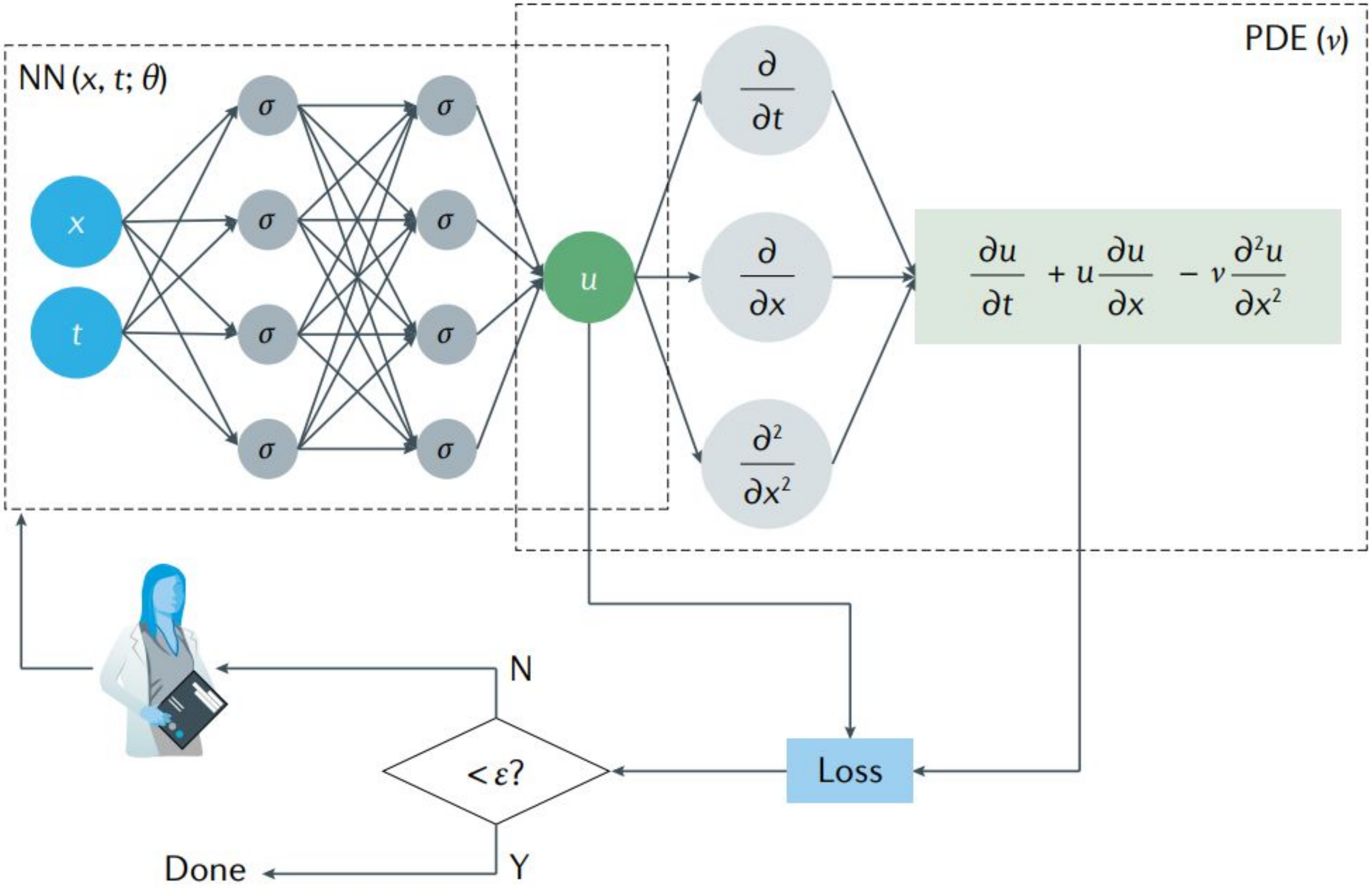
Learning Biases

- A “soft” approach
- Utilize appropriate loss function, inference algorithm, etc. to favor physics-supported solutions

PINN Architecture

PINN Architecture

- PINNs tend to focus on learning biases
- Loss function incorporating both PDE and observational data
- Situation where we have a known PDE and experimental data



NN $(x, t; \theta)$
PDE (ν)
x
t
σ
u
$\frac{\partial}{\partial t}$
$\frac{\partial}{\partial x}$
$\frac{\partial^2}{\partial x^2}$
$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \nu\frac{\partial^2 u}{\partial x^2}$
Loss
$< \varepsilon ?$
N
Y
Done

Loss

$$\mathcal{L} = w_{\text{data}}\mathcal{L}_{\text{data}} + w_{\text{PDE}}\mathcal{L}_{\text{PDE}},$$

where

$$\mathcal{L}_{\text{data}} = \frac{1}{N_{\text{data}}}\sum_{i=1}^{N_{\text{data}}} (u(x_i, t_i) - u_i)^2 \quad \text{and}$$

$$\mathcal{L}_{\text{PDE}} = \frac{1}{N_{\text{PDE}}}\sum_{j=1}^{N_{\text{PDE}}} \left(\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \nu\frac{\partial^2 u}{\partial x^2}\right)^2 \Big|_{(x_j, t_j)}.$$

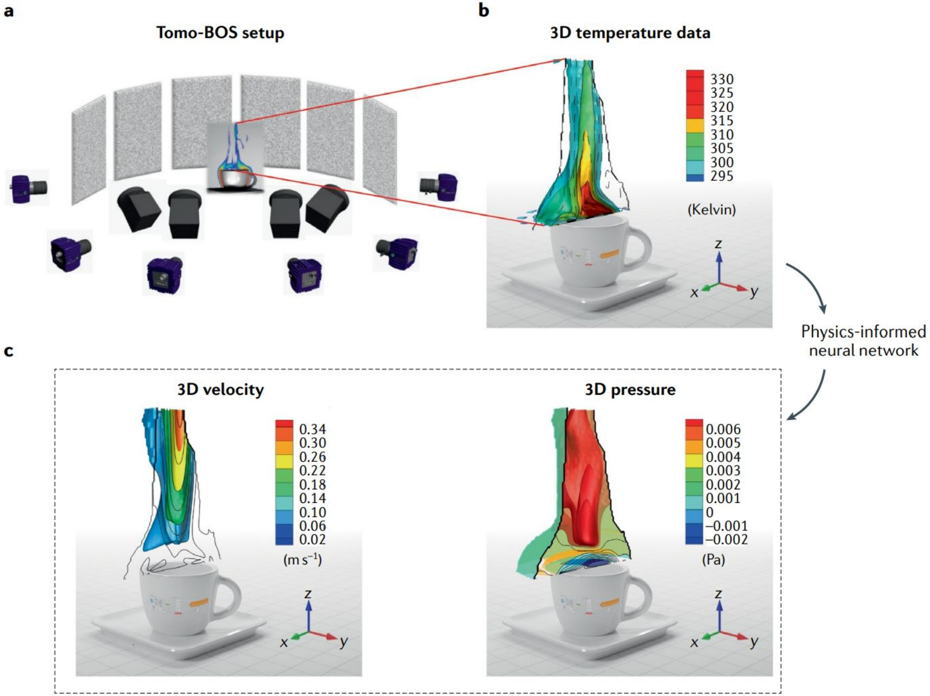
Connections

- Classical numerical methods show many analogs to PINN methods

PINN Strengths

- Work best on inverse problems and poorly formulated situations
  - Traditional solvers are working better with forward, well-posed problems
- Incomplete Models, missing/outlier data
  - “Middle Case”
- Requiring less data to train due to embedding of physics

Example – Flow over Espresso Cup



Libraries: DeepXDE

- DL library for differential equation modeling
- Research tool: Ready-to-go PINN solver
- Also described as an “educational tool” suitable for coursework

Related Work

- DeepFNets - Neural Networks for Functional Approximation
- DeepONets - Neural Networks for Operator Approximation
- Multi-physics applications
  - “Digital Twins”

Sources

- https://www.brown.edu/research/projects/crunch/sites/brown.edu.research.projects.crunch/files/uploads/Nature-REviews_GK.pdf: Review of PIML (Karniadakis et al.)
- https://arxiv.org/pdf/1907.04502.pdf: DeepXDE: A Deep Learning Library for Solving Differential Equations (Lu et al.)
- https://www.youtube.com/watch?v=QV1fVttZ6YE&t=3468s: From PINNs to DeepONets (Talk, Karniadakis)
- https://maziarraissi.github.io/PINNs/: Physics Informed Deep Learning (Raissi et al.)