Fundamentals of Physics Informed Machine Learning

Yunchao Yang, PhD

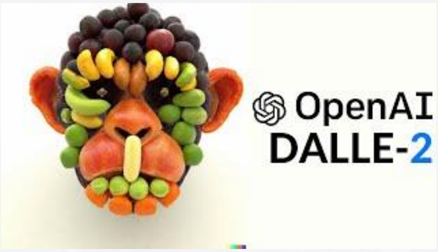
AI Research Facilitator

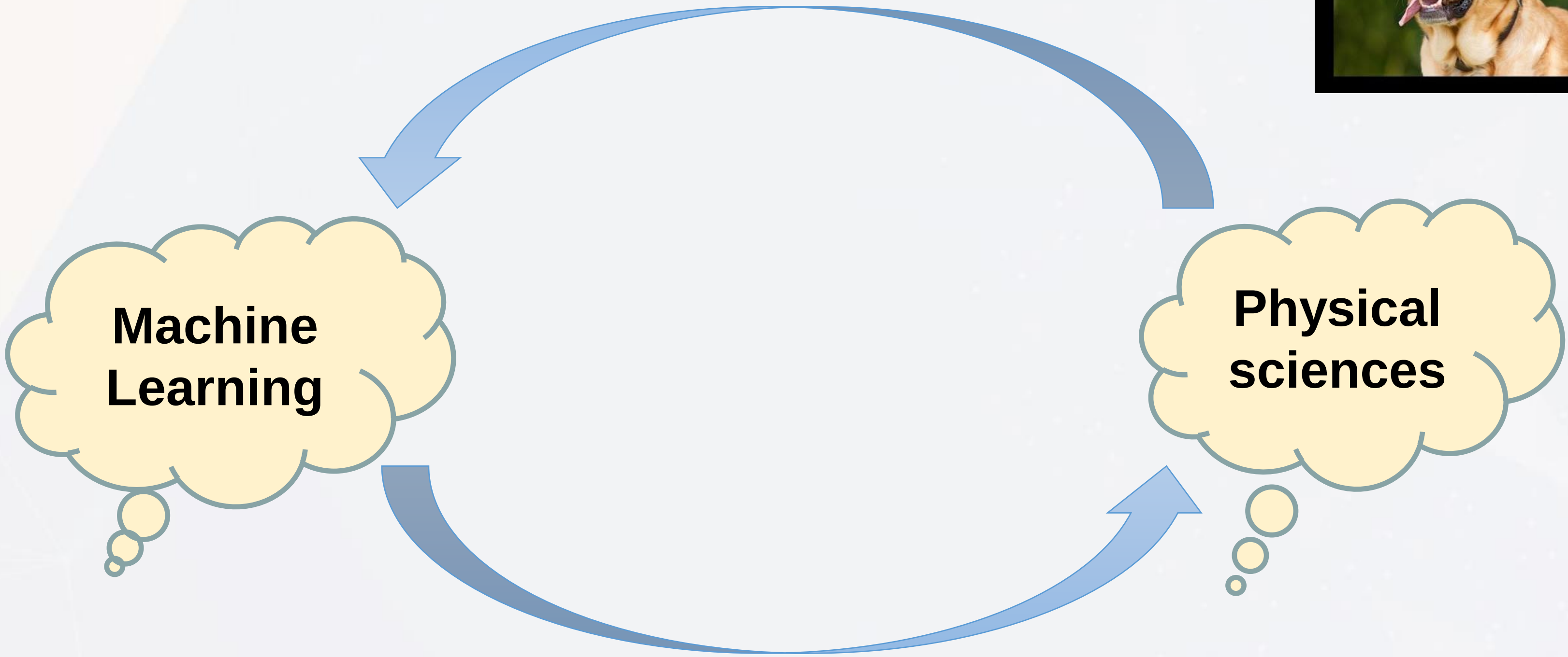
UF Research Computing

yunchaoyang@ufl.edu

Interplay of Machine Learning and Physics

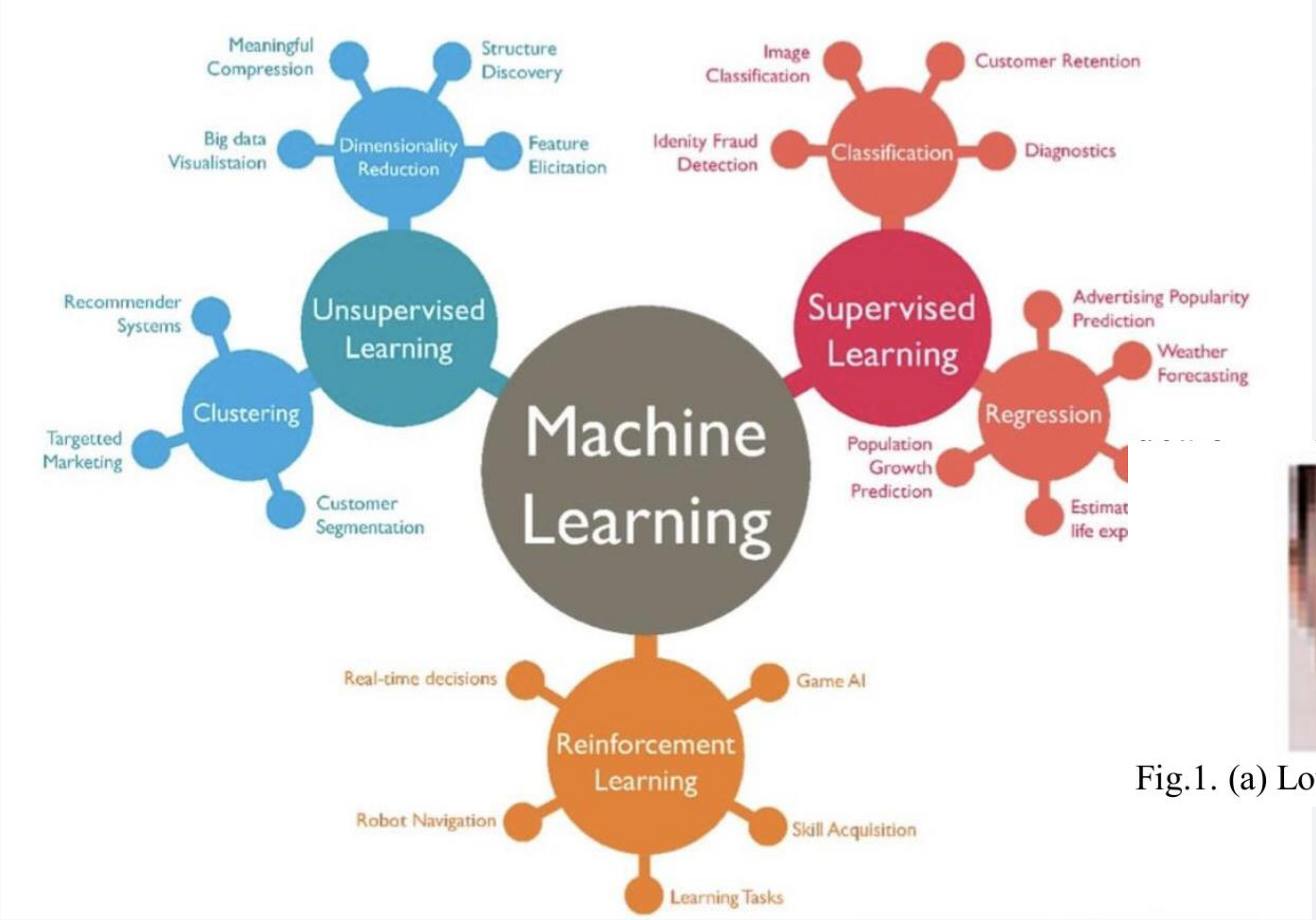
- Inspire new algorithms from physical insights: e.g. diffusion model
- human-understandable insights from Interpreting ML results





- provide a scientific tool for discovering elusive patterns within physical sciences
- data-driven solution of complex science & engineering problems
- applications of machine learning techniques to physical sciences is growly rapidly

Machine learning/Deep Learning algorithms



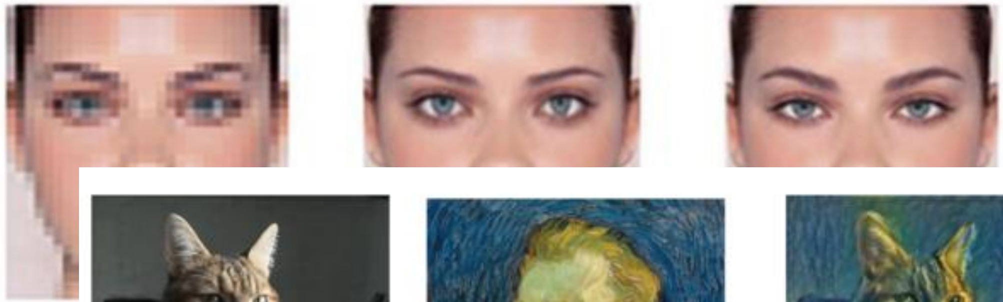


Fig.1. (a) Low

Physical systems and PDEs

The dynamic performance of a physical system is obtained by utilizing the physical laws of mechanical, electrical, fluid and thermodynamic systems. The physical systems are generally modeled with partial differential equations (PDE).

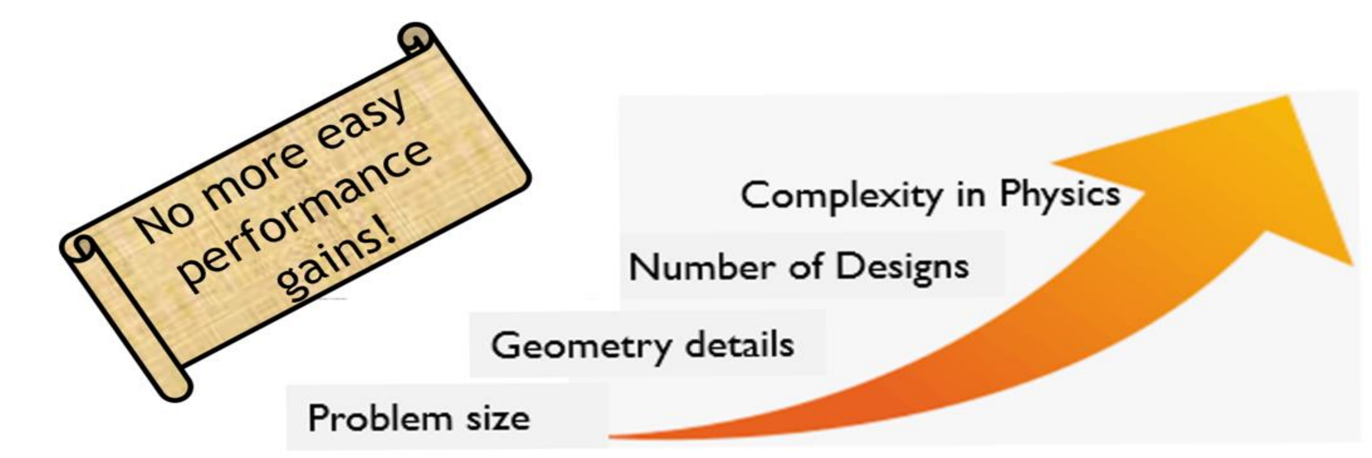
- most PDEs cannot be solved analytically in real applications
  - Heat equation $\frac{\partial u}{\partial t} = \Delta u$
  - Wave equation $\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}.$
  - Laplace's equation $\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2} = 0.$
  - Poisson's equation $\nabla^2 \varphi = f.$
  - Burgers' equation $\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2}.$
  - Navier-Stokes equation $\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} - \nu \nabla^2 \mathbf{u} = -\frac{1}{\rho}\nabla p + \mathbf{g}.$

Bottleneck in Traditional Scientific Computing

Traditional numerical methods:

Numerical analysis and algebra

- Finite difference
- Finite Element
- Finite Volume
- Runge–Kutta methods



Source: NVIDIA

limitations:

- Computationally Expensive
- Domain Discretization Techniques
- Not suitable for Data-assimilation or Inverse problems

The integration of Data, Deep Neural Networks, and Physical Laws - PINN

- Physics
  - Partial differential equations (PDE)
    - Governing equations
    - Conservation law
  - Boundary and initial conditions
- Neural Networks
  - Function approximation

$$G(x) = \sum_{j=1}^{N} \alpha_j \sigma\left(y_j \cdot x + \theta_j\right)$$

- Data
  - available observations

Physics-Informed Neural Networks (PINNs): A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations

M Raissi, P Perdikaris, GE Karniadakis, Journal of Computational Physics 378, 686-707

How about Black-box Deep Neural Networks?

- Scientific problems are often under-constrained
  - Complex, dynamic, and non-stationary relationships
  - A large number of variables with a small number of samples
- Standard methods for evaluating ML models (e.g., cross-validation) will fail
  - Easy to learn spurious relationships that look deceptively good on training and test datasets
  - But lead to poor generalization outside the available data
- Interpretability is an important end-goal (esp. in scientific problems)
  - How can we open the black-box of DNN results?
- Need to explain or discover the underlying mechanisms of process to
  - Form a basis for scientific advancements
  - Safeguard against the learning of non-generalizable patterns

Karpatne, DLPS, 2017

Problem setup

- Parameterized, nonlinear PDE(s)

$$u_t + \mathcal{N}[u; \lambda] = 0, x \in \Omega \subset \mathbb{R}^D, \; t \in [0, T]; \quad (\cdot)_t = \frac{\partial (\cdot)}{\partial t}$$

- where u(t, x) denotes the latent (hidden) solution, N [·; λ] is a nonlinear operator parametrized by λ.
- The above setup covers a wide range of PDEs in applied mathematics, including conservation laws, diffusion, convection–diffusion, etc.

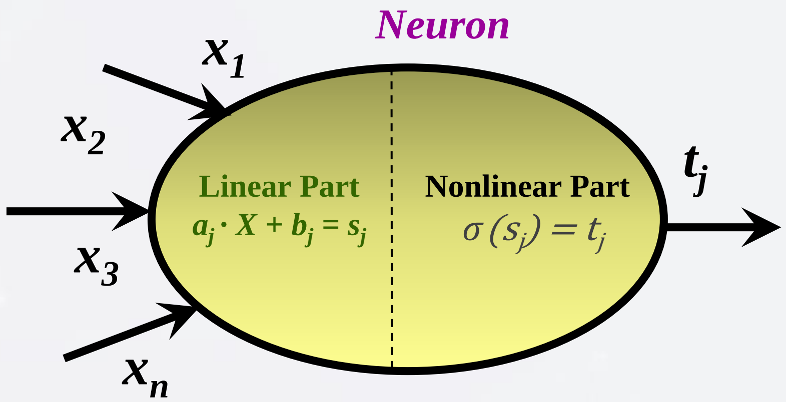
- For example: Burger’s equations:

$$\mathcal{N}[u; \lambda] = \lambda_1 u u_x - \lambda_2 u_{xx} \text{ and } \lambda = (\lambda_1, \lambda_2); \quad (\cdot)_x = \frac{\partial (\cdot)}{\partial x} \quad (\cdot)_{xx} = \frac{\partial^2 (\cdot)}{\partial x^2}$$

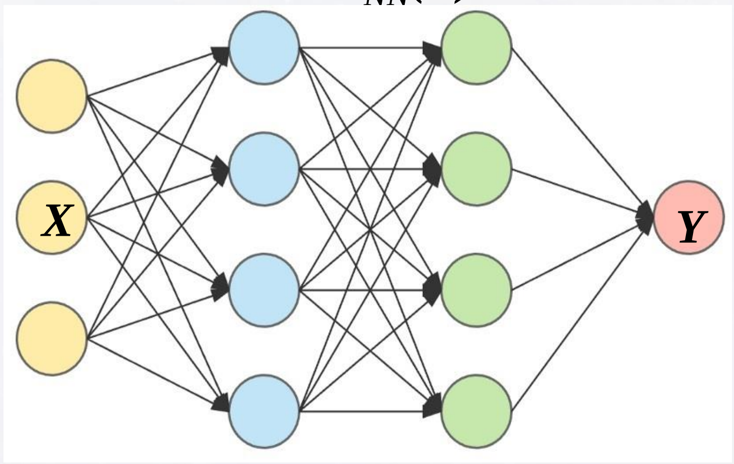
Neural Networks = Function Approximation

Try to find a mapping function: $Y = f(X;\theta)$

- Polynomial: $a_1 + a_2 x + a_3 x^2 + \cdots$
- Nonlinear: $1 + \frac{a_1 \tanh(a_2 x)}{a_3 x - \tanh(a_4 x)}$
- Neural Network: $W_3\sigma(W_2\sigma(W_1 x + b_1) + b_2) + b_3$.
  - Neural Networks are universal approximators which work well in high dimensions
  - Train the weights (W, b)



$Y = F_{NN}(X)$



Introduction: PINNs

$$u_t + \mathcal{N}[u; \lambda] = 0, x \in \Omega \subset \mathbb{R}^D, \ t \in [0, T]; \quad (\cdot)_t = \frac{\partial (\cdot)}{\partial t}$$

- Data-driven solution
  - λ Given, the goal is to find NN(t, x) = u(t,x)
- Data-driven discovery of PDEs
  - Find λ that best describes observations u (t_i , x_j)

M Raissi, P Perdikaris, GE Karniadakis,
Journal of Computational Physics 378, 686-707

PINN: Data-driven solution

- Rewrite the PDE as $f(u; t, x) = 0$

$$f(u; t, x) \doteq u_t + \mathcal{N}[u], \quad \text{along with} \quad u = u_\theta(t, x)$$

- Along with the above constraint (+ AD) this gives *Physics-informed neural network* **parameterized by θ**

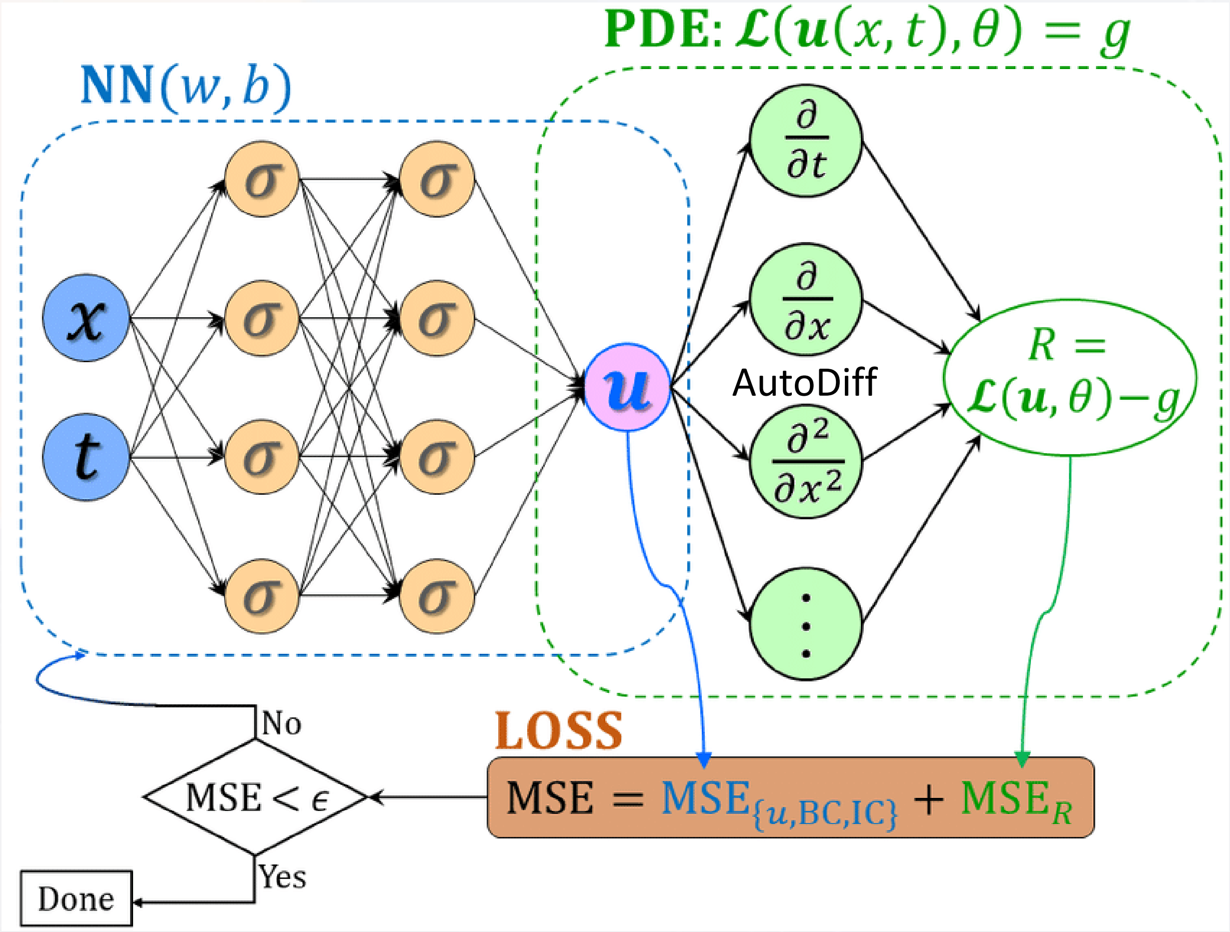
$$\boxed{\mathcal{L} = \mathcal{L}_u + \mathcal{L}_f}$$

$$\mathcal{L}_u = \frac{1}{N_u} \sum_{i=1}^{N_u} \left| u\left(t_u^i, x_u^i\right) - u^i \right|^2 ; \quad \mathcal{L}_f = \frac{1}{N_f} \sum_{i=1}^{N_f} \left| f\left(t_f^i, x_f^i\right) \right|^2$$

Data loss **Physical loss**

PINN:Data-driven solution

- Network architecture



- Losses:
  - Physical loss
    - Use automatic differentiation to calculate derivatives
    - Governing equation
  - Data loss
    - IC/BC
    - Labeled data ((observation), optional)

PINN: data-driven discovery of PDE

- Given noisy and incomplete measurements z of the state of the system, the data-driven discovery of PDE results in computing the unknown state $u(t,x)$ and learning model parameter λ that best describe the observed data.

$$u_t + N[u; \lambda] = 0, \quad x \in \Omega, \quad t \in [0, T]$$

- Define: $f := u_t + N[u; \lambda] = 0$

- Treat λ as a learnable parameter in the neural network

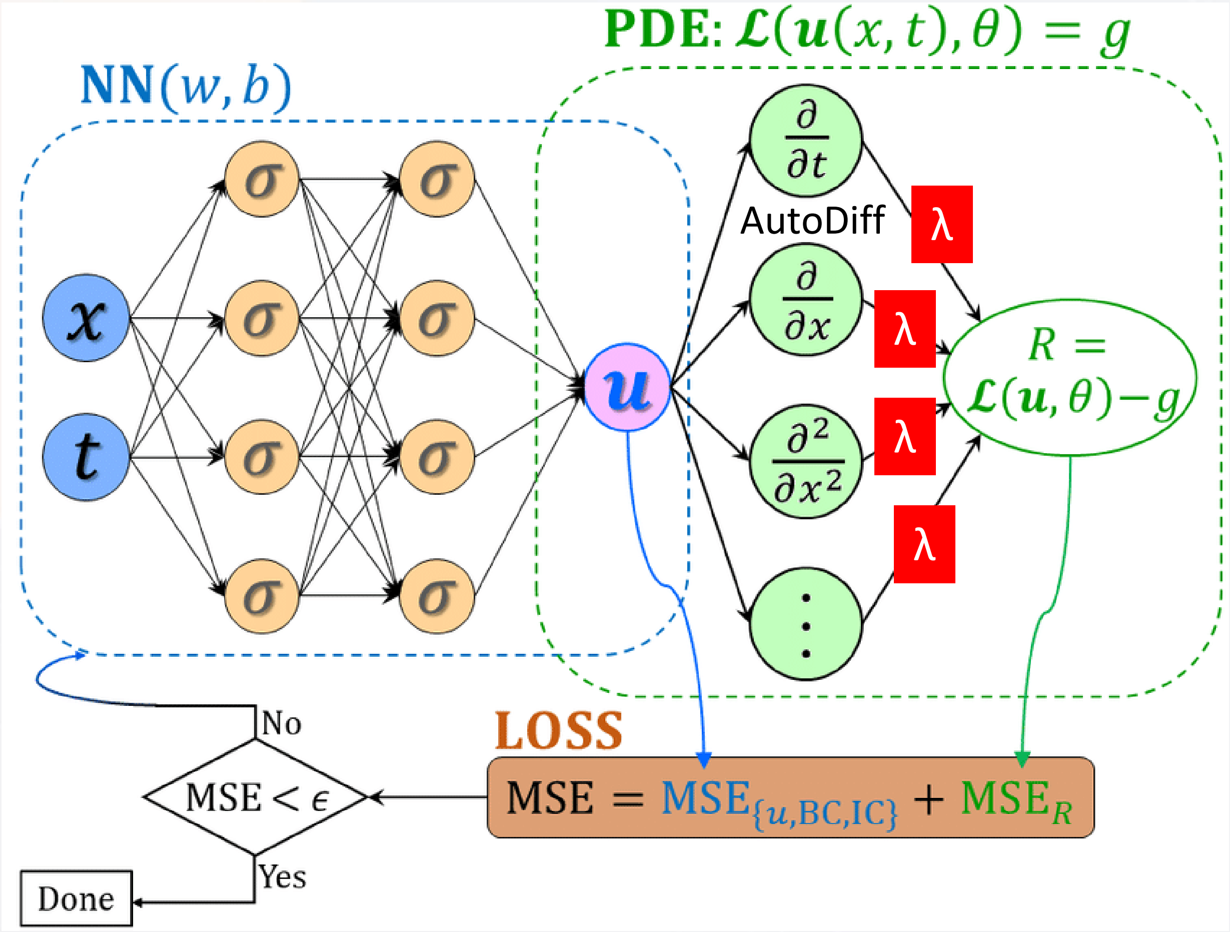
- This network is to approximate $\boldsymbol{u(t,x)}$. Then the parameters of $\boldsymbol{u(t,x)}$ and λ can be learned by minimizing the same loss function

$$L_{tot} = Lu + Lf$$

M Raissi, P Perdikaris, GE Karniadakis, Journal of Computational Physics 378, 686-707

PINN: data-driven discovery of PDE

- Network architecture



- Losses:
  - Physical loss (with λ)
    - Use automatic differentiation to calculate derivatives
    - Governing equation
  - Data loss
    - IC/BC
    - Labeled data ((observation), optional)
- λ can be learned by minimizing losses during backpropagation

Code demo

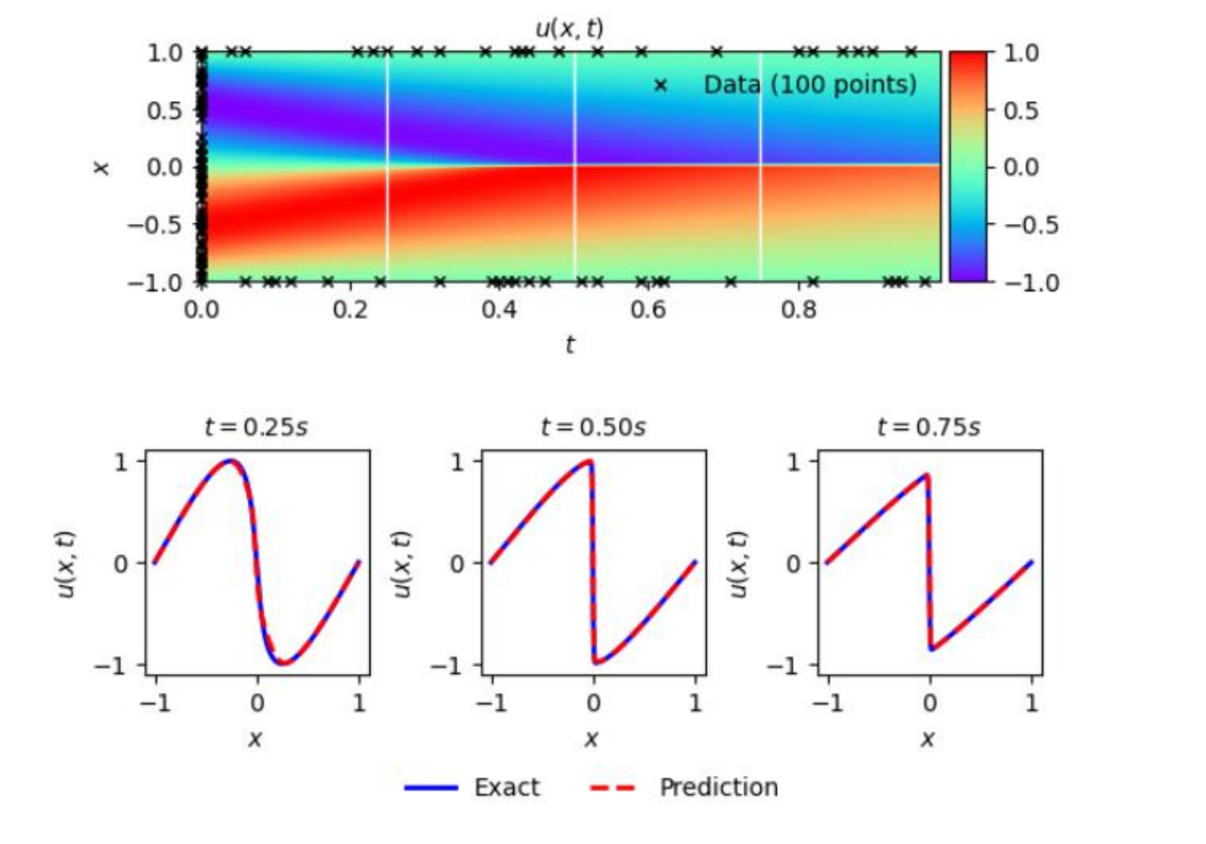
- PINN solution of 1D Burgers equation in PyTorch

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}$$

$$x \in [-1, 1]$$
$$t \in [0, 1]$$

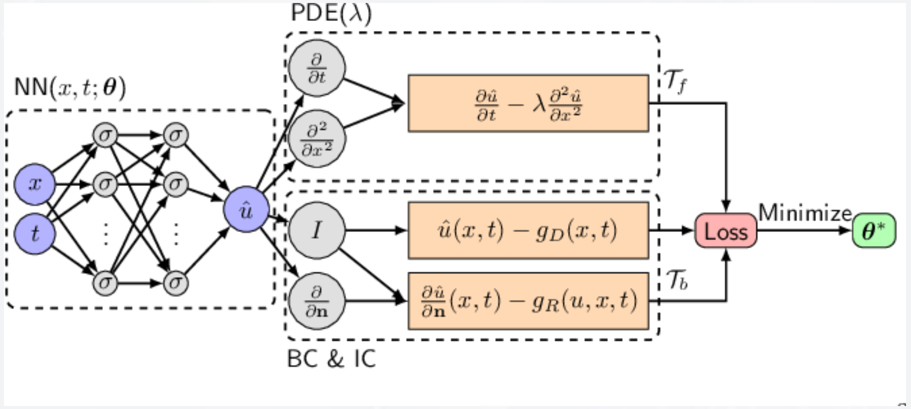
IC/BC

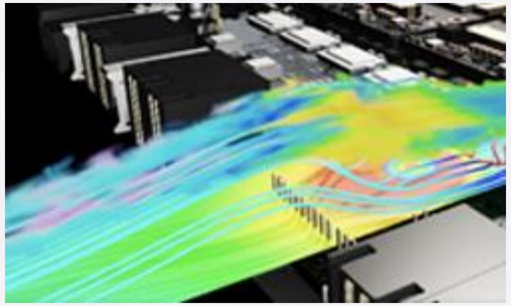
$$t = 0, u = 0$$
$$x = -1, u = 0$$
$$x = 1, u = 0$$



Available PINN libraries

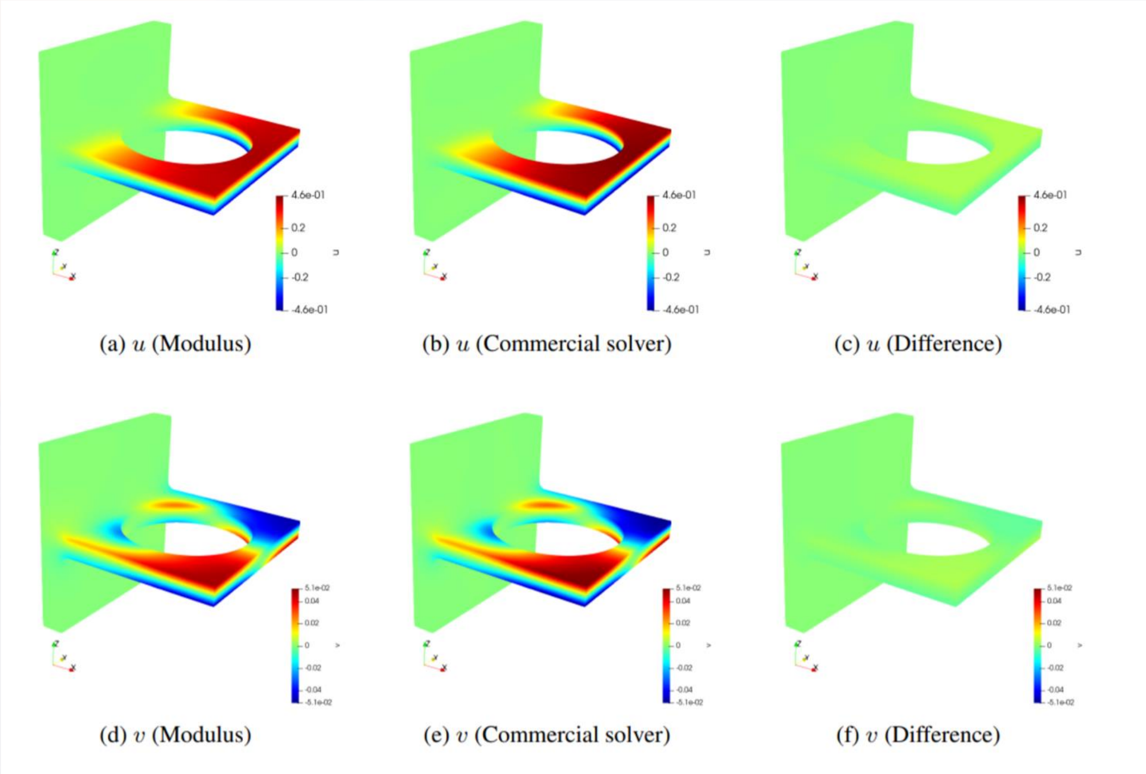
- [DeepXDE]()
- [NVIDIA Modulus]()
- PyTorch or Tensorflow implementations





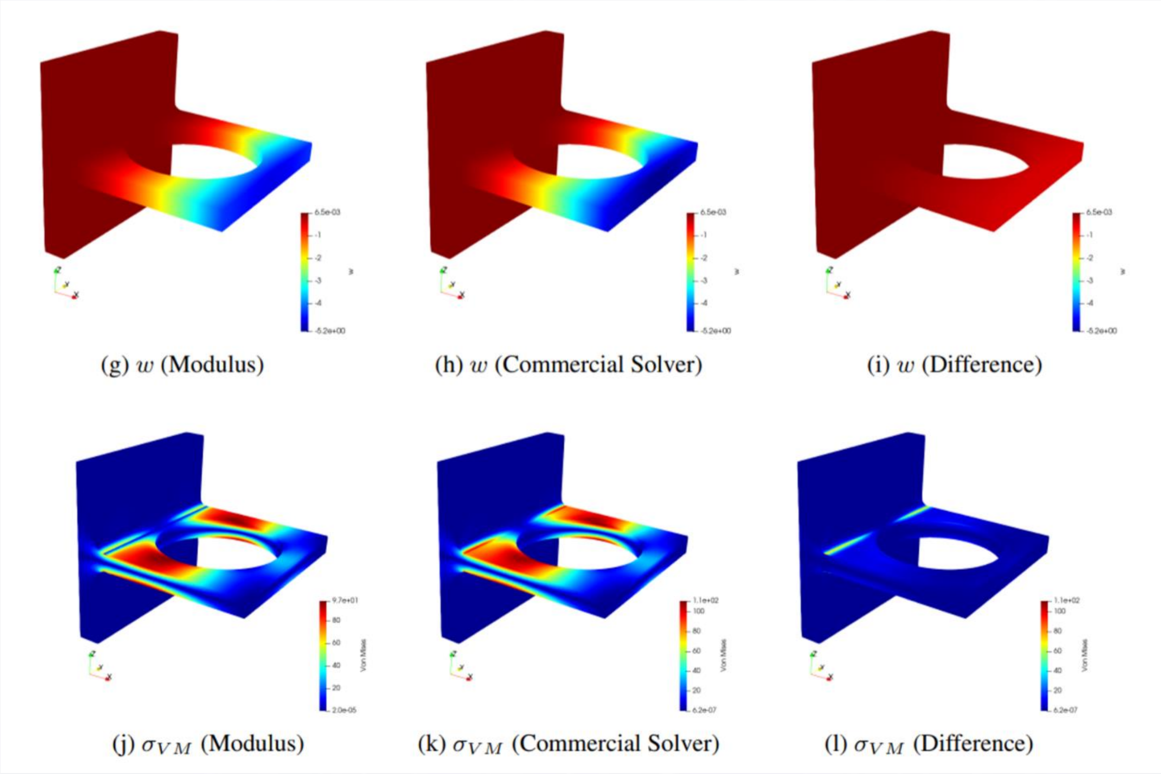
What is Modulus?

- Modulus is a PDE solver
    - Like traditional solvers such as Finite Element, Finite Difference, Finite Volume, and Spectral solvers, Modulus can solve PDEs



(a) u (Modulus) (b) u (Commercial solver) (c) u (Difference)

(d) v (Modulus) (e) v (Commercial solver) (f) v (Difference)



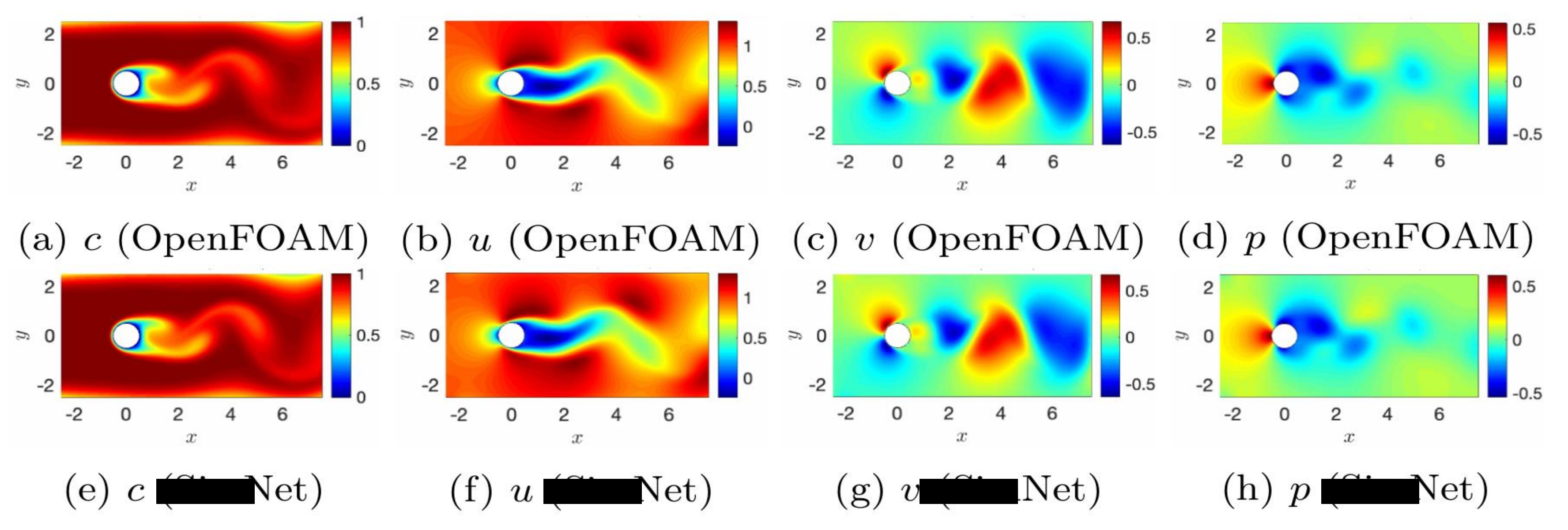
(g) w (Modulus) (h) w (Commercial Solver) (i) w (Difference)

(j) σ_{VM} (Modulus) (k) σ_{VM} (Commercial Solver) (l) σ_{VM} (Difference)

A comparison between Modulus and commercial solver results for a bracket deflection example. Linear elasticity equations are solved here.

What is Modulus?

- Modulus is a solver for inverse problems
  - Many applications in science and engineering involve inferring unknown system characteristics given measured data from sensors or imaging.
  - By combining data and physics, Modulus can effectively solve inverse problems.



A comparison between Modulus and OpenFOAM results for the flow velocity, pressure and passive scalar concentration fields. Modulus has inferred the velocity and pressure fields using scattered data from passive scalar concentration.

Modulus Resources

- Download Now: https://developer.nvidia.com/modulus-downloads
- Webpage: https://developer.nvidia.com/modulus
- Documentation: https://sw-docs-dgx-station.nvidia.com/deeplearning/modulus/index.html
- Developer Forum: https://forums.developer.nvidia.com/c/physics-simulation
- Demos:
  - Accelerating Extreme Weather Prediction with FourCastNet
  - Siemens Energy HRSG Digital Twin Simulation Using NVIDIA Modulus and Omniverse
  - Accelerating Scientific & Engineering Simulation Workflows with AI
  - Flow Physics Quantification in an Aneurysm Using NVIDIA Modulus
- Blogs:
  - AI and Machine Learning in Physics
  - Using NVIDIA Modulus and Omniverse Wind Farm Digital Twin for Siemens Gamesa (using NVIDIA Modulus and Omniverse)
  - Siemens Energy Taps NVIDIA to Develop Industrial Digital Twin of Power Plant in Omniverse and Modulus
  - Using Hybrid Physics-Informed Neural Networks for Digital Twins in Prognosis and Health Management
  - Using Physics-Informed Deep Learning for Transport in Porous Media

Recent progress of Physics informed Learning

- **PINN family**
  - **sPINN**
  - **fPINN**
  - **xPINN**
- **Fourier Network**
- **Fourier Neural Operator**
- **Physics Informed Neural Operator (PINO)**
- **DeepONet (Deep Operator Networks)**
- **…**

https://github.com/idrl-lab/PINNpapers