Physics-informed machine learning

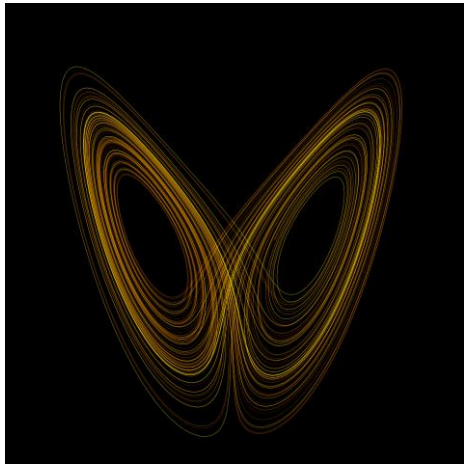
Zhongkai Hao

Content

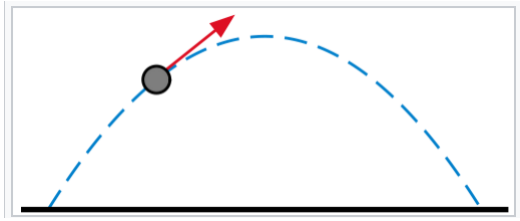
- Motivation
- Methods to combine physics and ML
- Observational biases
- Inductive biases
- Learning biases
- Advantages
- Examples and important applications
- Limitations
- Softwares
- Future directions

Some knowledge about physical systems

- Algebraic equations – kinematics, some stable or simple systems, graphics
- Dynamical systems– kinematics, population in bio
- ODE or ODEs— systems of mass points, rigid body, Hamilton systems
- PDE or PDEs— fields, Electric-Magnetic fields, fluid fields, quantum chemistry
- Statistical systems—systems with uncertainty

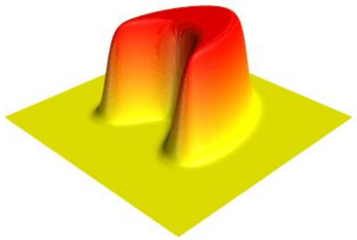


Dynamical systems



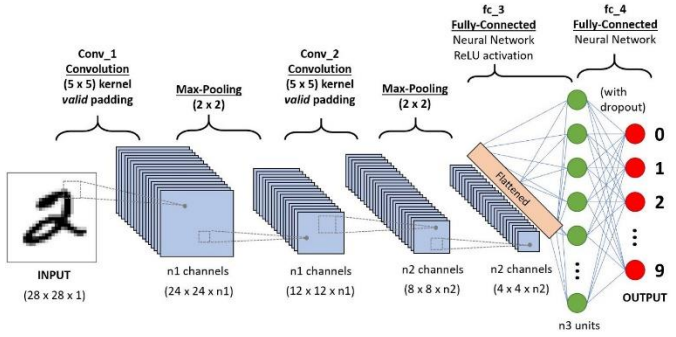
$$m\frac{\mathrm{d}^2x(t)}{\mathrm{d}t^2} = F(x(t))$$

ODE systems

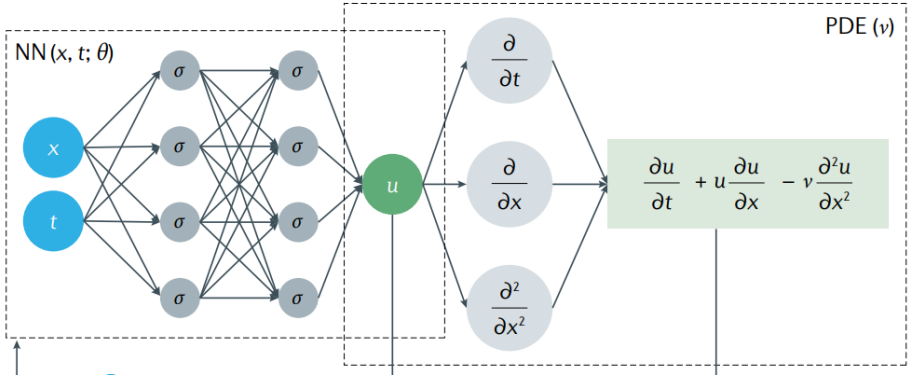


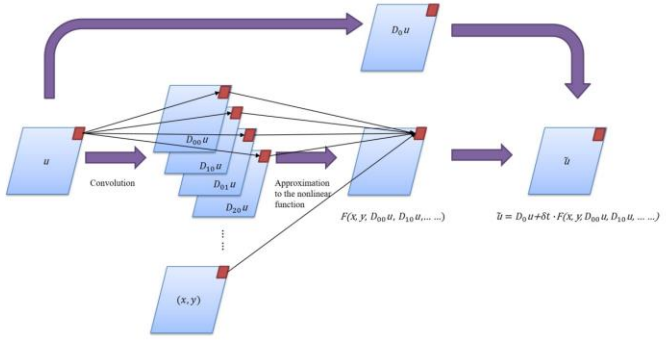
PDE systems

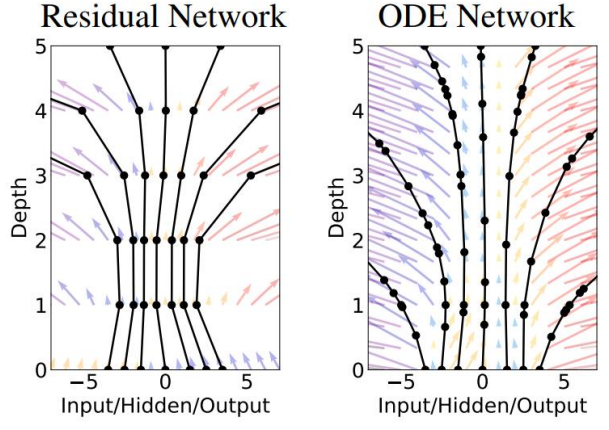
Motivation for combing physics and machine learning



- Physics as (inductive) biases for better machine learning models like CNN, RNN compared with MLP
- Use neural networks to solve physical systems (PINN)
- Use machine learning to estimate parameters or physical laws
- Explore connections between NN architectures and physics (NeuralODE/ Mean field theory for NN)

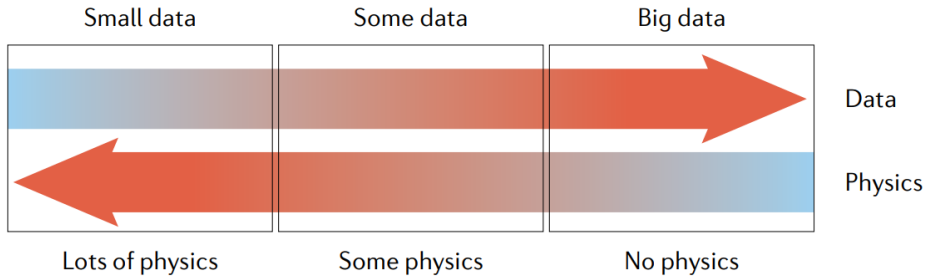


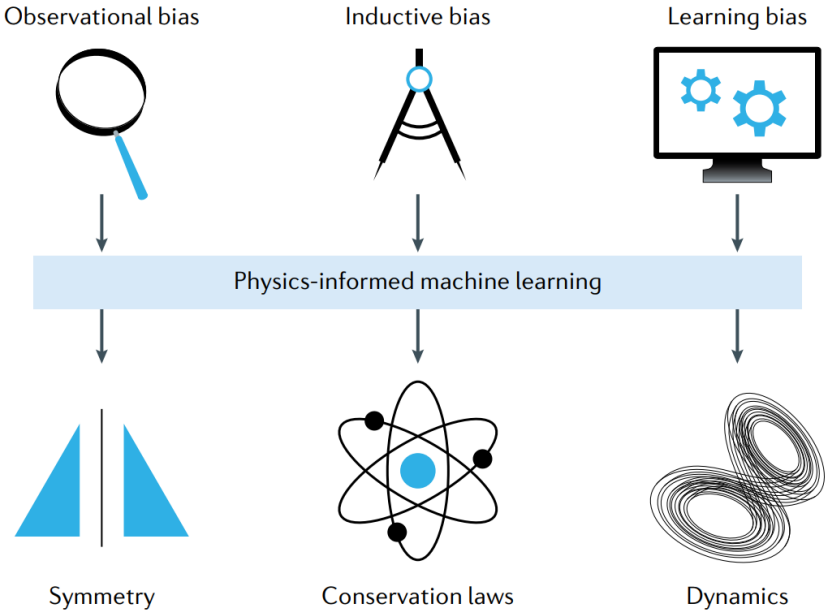




How to embed physics in ML

- Observational biases (use sufficient data with augmentation to fit a model)
- Inductive biases (specialized NN to embed some prior knowledge)
- Learning biases (use loss as constraint)
- Hybrid approaches(finding physical laws, combine with numeric methods)





Observational biases

- Main idea– Learns from big data, collect or augment data using symmetry

- CNN for image recognition
- CT image recognition
- Material property prediction
- MPNN for molecular property prediction

Inductive biases

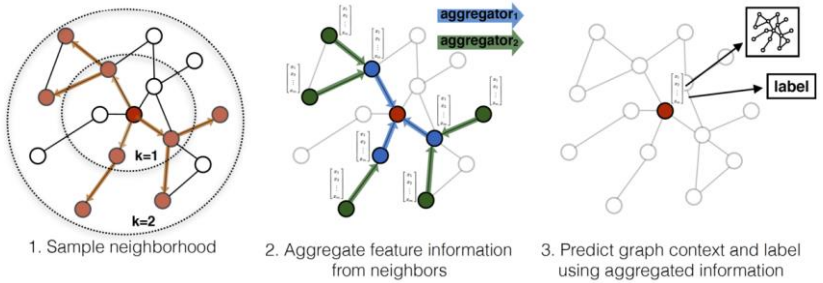
- Main idea– specialized NN architectures to embed prior knowledge
- Limitations– based on well-defined physics or symmetry groups

Fitting data (forward problems)

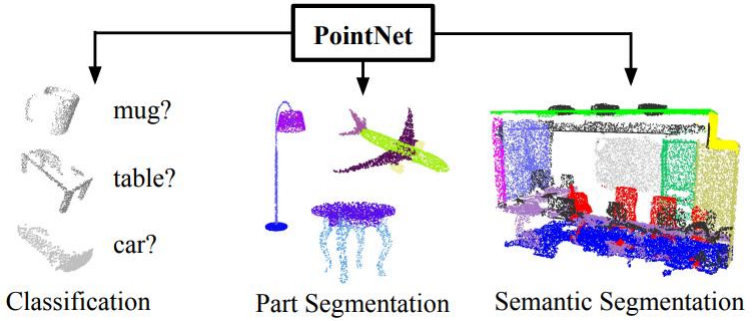
- CNN and more general CNN– translation/rotation/gauge symmetry invariance/equivariance
- GNN– permutation invariance/equivariance
- DTNN/SchNet—spatial translation/rotation invariance

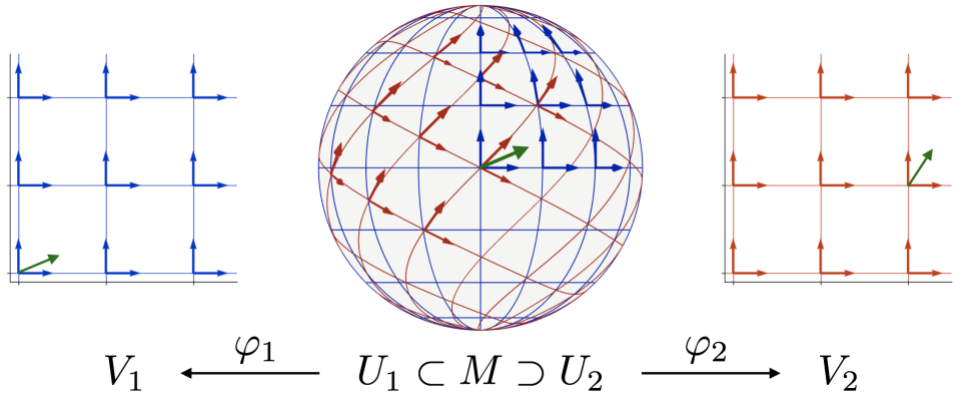
Fitting system parameters (inverse problems)

Inductive biases



- Convolutional NN/ Graph NN/PointNet
- GNN: message passing for permutation invariant while considering local connections
- PointNet: global pooling for permutation invariant
- Gauge equivalent NN on mainfolds
- SympNets– learns systems that preserves symplectic mappings



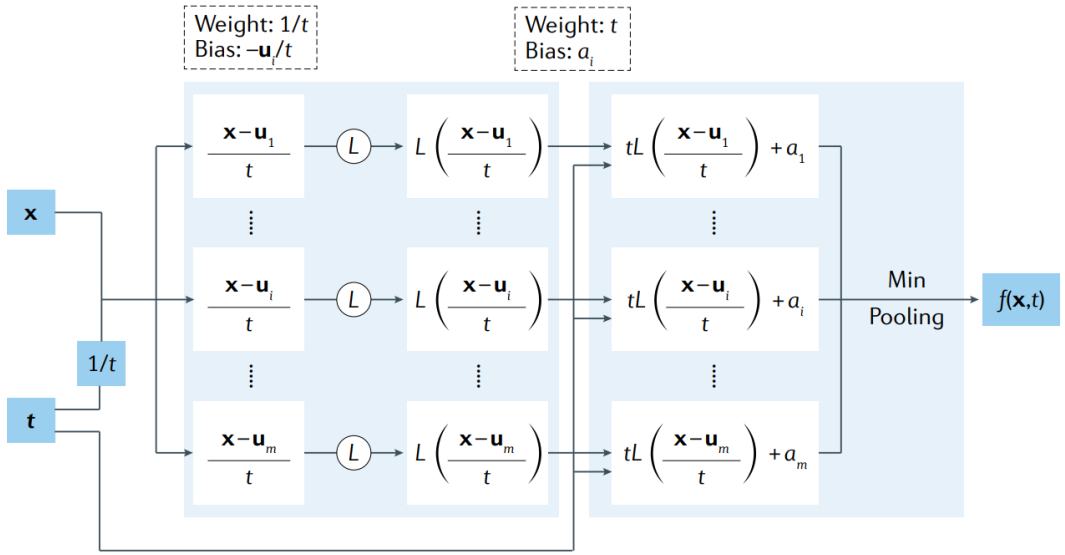


Inductive biases

- Lax-Oleinik Formula
- Some NN are natural solutions of Hamilton-Jacobi equations

$$\begin{cases} \dfrac{\partial f}{\partial t}(x,t) + H(\nabla_x f(x,t)) = 0 & x \in \mathbb{R}^n, t \in (0,+\infty), \\ f(x,0) = J(x) & x \in \mathbb{R}^n, \end{cases}$$

- This motivate us to find other cases that NN are solutions of PDE.



Learning biases

- Main idea—add regularization terms to loss function

- Examples
- Deep Galerkin Methods
- PINN(physics informed neural networks)

- InvNet(use a invariance checker network in GAN to keep invariance)
- ContactNet(use a special parameterization to deal with discontinuity in contact)
- Use Bayesian networks for uncertainty quantification
- Add penalty to enforce Lyapunov stability

Deep Galerkin Methods

- Consider the PDE below

$$\begin{aligned}
&\frac{\partial u}{\partial t}(t,x) + \mathcal{L}u(t,x) = 0, \quad (t,x) \in [0,T] \times \Omega, \\
&u(t=0,x) = u_0(x), \\
&u(t,x) = g(t,x), \quad x \in \partial\Omega,
\end{aligned}$$

- We use a network $f(t,x;\theta)$ to approximate the solution and use this loss to optimize it

$$J(f) = \left\| \frac{\partial f}{\partial t}(t,x;\theta) + \mathcal{L}f(t,x;\theta) \right\|^2_{[0,T]\times\Omega,\nu_1} + \|f(t,x;\theta) - g(t,x)\|^2_{[0,T]\times\partial\Omega,\nu_2} + \|f(0,x;\theta) - u_0(x)\|^2_{\Omega,\nu_3}.$$

- When the dimension is large, we use Monte-Carlo sampling to estimate the integral

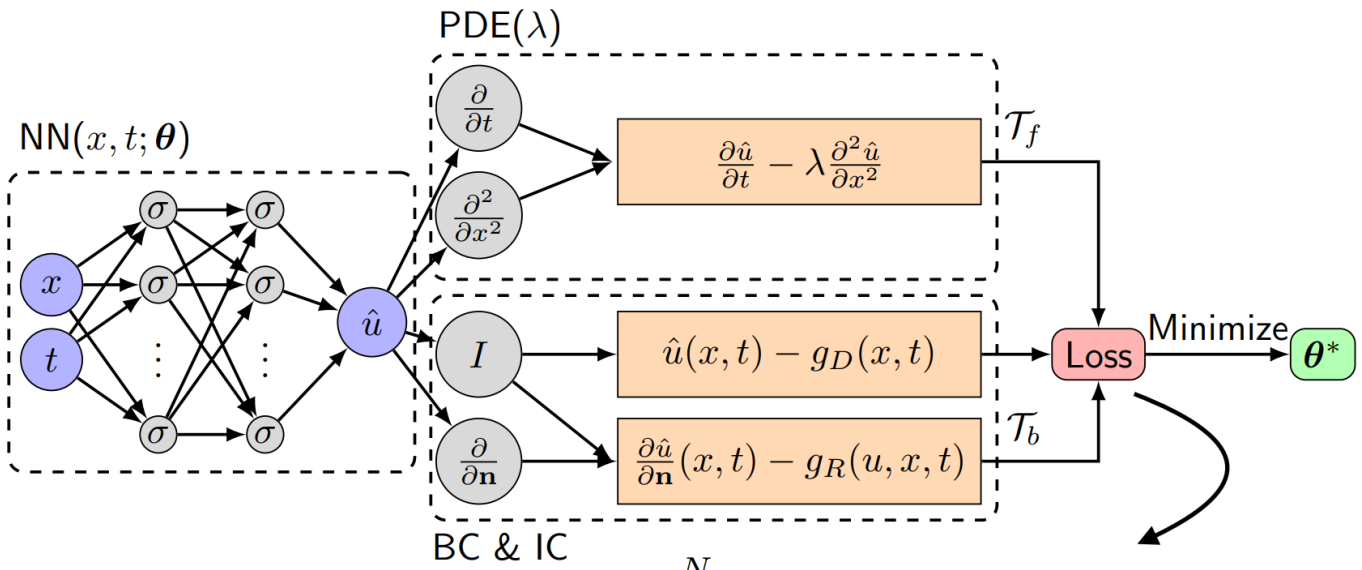
$$G(\theta_n, s_n) = \left(\frac{\partial f}{\partial t}(t_n,x_n;\theta_n) + \mathcal{L}f(t_n,x_n;\theta_n) \right)^2 + \left(f(\tau_n,z_n;\theta_n) - g(\tau_n,z_n) \right)^2 + \left(f(0,w_n;\theta_n) - u_0(w_n) \right)^2.$$

- Then $\mathbb{E}\left[\nabla_\theta G(\theta_n, s_n) \middle| \theta_n\right] = \nabla_\theta J(f(\cdot;\theta_n))$.

https://arxiv.org/pdf/1708.07469.pdf

Physics informed neural networks

- Main idea – use regularization to solve the PDE/ODE
- A flexible framework to handle both data and prior knowledge



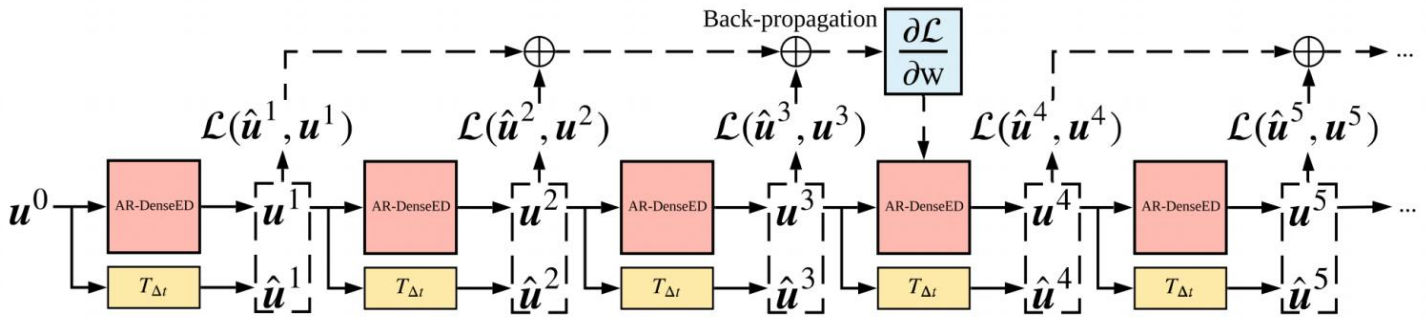
$$\mathcal{L}(\theta) := \underbrace{\frac{1}{N_u}\sum_{i=1}^{N_u}[\boldsymbol{u}_i - f_\theta(\boldsymbol{x}_i)]^2}_{\text{Data fit}} + \underbrace{\frac{1}{\lambda}\mathcal{R}[f_\theta(\boldsymbol{x})]}_{\text{Physics regularization}}$$

https://www.seas.upenn.edu/~cis522/slides/CIS522_Lecture11T.pdf

Uncertainty Quantification of PINN

- Use multi-step method for learning physics constraints

$$\begin{aligned}\boldsymbol{u}(\boldsymbol{x},t)_t + F\left(\boldsymbol{x},\boldsymbol{u}\left(\boldsymbol{x},t\right)\right) = 0, \quad \boldsymbol{x}\in\Omega,\ t\in[0,T],\\ \mathcal{B}(\boldsymbol{u}) = b\left(\boldsymbol{x},t\right), \quad \boldsymbol{x}\in\Gamma,\\ \boldsymbol{u}(\boldsymbol{x},0)\sim p\left(\boldsymbol{u}(\boldsymbol{x},0)\right),\end{aligned}$$



- Use SWAG for sampling posterior , in SWAG

$$\hat{\boldsymbol{u}}^i = f\left(\boldsymbol{\chi}^i,\mathbf{w}\right)+\epsilon, \quad p\left(\epsilon\right)=\mathcal{N}\left(\epsilon\,|0,\beta^{-1}\boldsymbol{I}_d\right),$$

$$\begin{aligned}p\left(\hat{\boldsymbol{u}}^i|\boldsymbol{\chi}^i,\mathbf{w},\beta\right) &= \mathcal{N}\left(\hat{\boldsymbol{u}}^i|f\left(\boldsymbol{\chi}^i,\mathbf{w}\right),\beta^{-1}\boldsymbol{I}_d\right),\\ &= \mathcal{N}\left(T_{\Delta t}\left(\boldsymbol{\mathcal{U}}^i,F_{\Delta x}\right)|f\left(\boldsymbol{\chi}^i,\mathbf{w}\right),\beta^{-1}\boldsymbol{I}_d\right).\end{aligned}$$

$$p(\boldsymbol{\theta}|\mathcal{S})\sim\mathcal{N}\left(\boldsymbol{\theta}_{SWA},\boldsymbol{\Sigma}_{SWA}\right), \quad \boldsymbol{\theta}\equiv\{\mathbf{w},ln(\beta)\},$$

Hybrid Approaches

- DeepONet for operator learning
- Learning unknown physics (e.g. constitutive laws for non-Newtonian fluids)
- Embed numerical method into a NN
- FermiNet that parameterized wavefunction obey Fermi-Dirac statistics
- Graph Operator Networks that incorporates graphs to model particle interactions

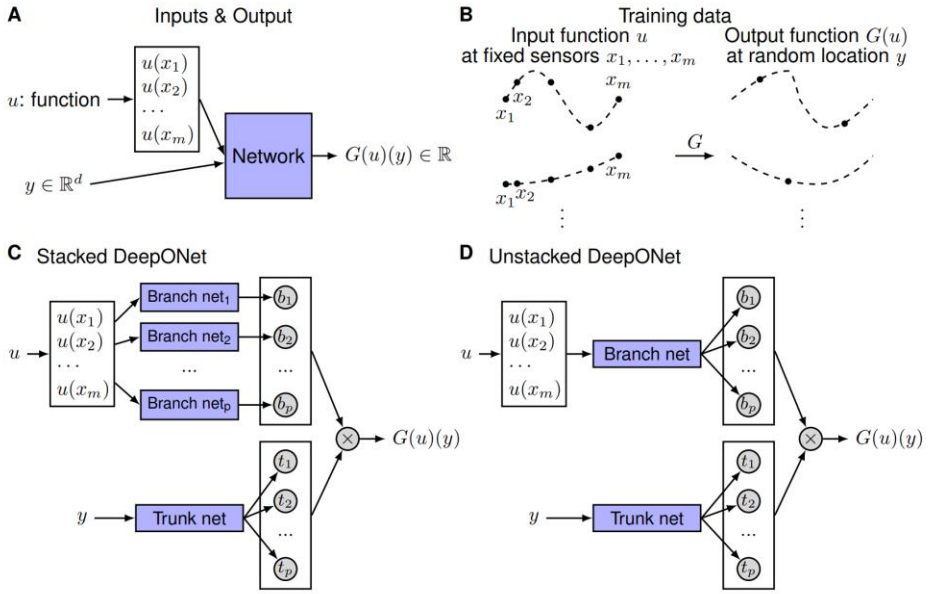
DeepONet

- Learning operator from data

Theorem 1 (Universal Approximation Theorem for Operator). *Suppose that σ is a continuous non-polynomial function, X is a Banach Space, $K_1 \subset X$, $K_2 \subset \mathbb{R}^d$ are two compact sets in X and $\mathbb{R}^d$, respectively, V is a compact set in $C(K_1)$, G is a nonlinear continuous operator, which maps V into $C(K_2)$. Then for any $\epsilon > 0$, there are positive integers n, p, m, constants $c_i^k, \xi_{ij}^k, \theta_i^k, \zeta_k \in \mathbb{R}$, $w_k \in \mathbb{R}^d$, $x_j \in K_1$, $i = 1, \dots, n$, $k = 1, \dots, p$, $j = 1, \dots, m$, such that*

$$\left| G(u)(y) - \sum_{k=1}^{p} \underbrace{\sum_{i=1}^{n} c_i^k \sigma \left(\sum_{j=1}^{m} \xi_{ij}^k u(x_j) + \theta_i^k \right)}_{branch} \underbrace{\sigma(w_k \cdot y + \zeta_k)}_{trunk} \right| < \epsilon \qquad (1)$$

holds for all $u \in V$ and $y \in K_2$.



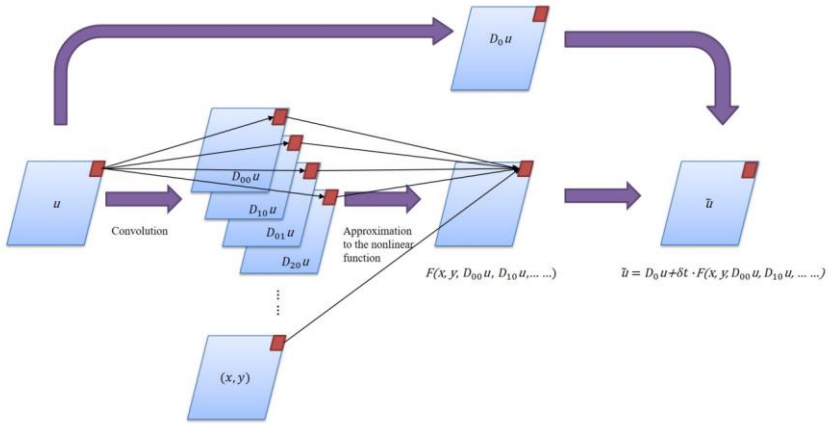
- Input: a datasets of functions f, probe points y, output $G(u)(y)$
- Examples:

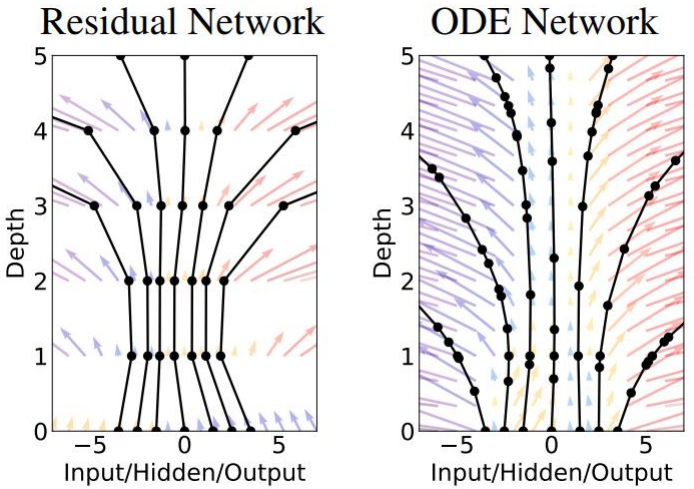
$$\begin{cases} \frac{d}{dx}\boldsymbol{s}(x) = \boldsymbol{g}(\boldsymbol{s}(x), u(x), x) \\ \boldsymbol{s}(a) = \boldsymbol{s_0} \end{cases}, \qquad (Gu)(x) = \boldsymbol{s_0} + \int_a^x \boldsymbol{g}((Gu)(t), u(t), t)dt.$$

Connections to numerical methods

- Main idea—Some NN architectures can be viewed as numerical methods

- PDE-Net—Convolutional (NN) are finite differential stencils of PDE discretizations
- ResNet—Euler discretization for some ODE
- NeuralODE—bridge NN and ODE





PDE-Net—Convolution as numeric differentiation discretization (ICML'18)

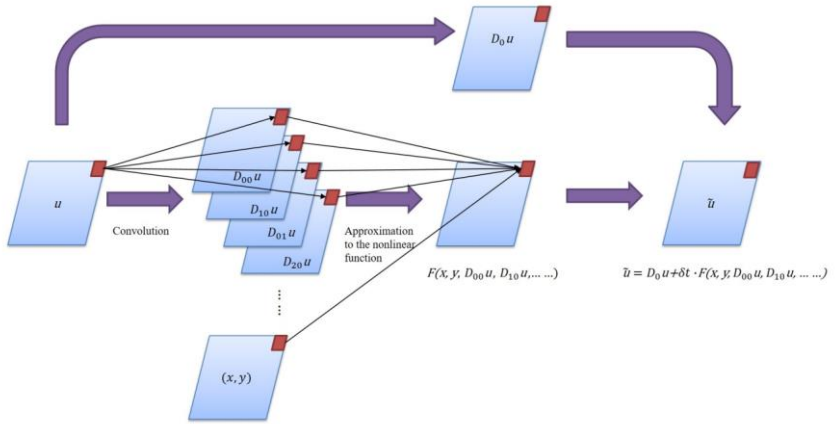
- Numerical diff operator equals to convolutional filter

1D filter: $\vec{D}_x^2 = [1 \quad -2 \quad 1]$,

2D filter: $\mathbf{D}_{xy}^2 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$.

- Train $F(.)$ to fit the data

$$\tilde{u}(t_{i+1}, \cdot) = D_0 u(t_i, \cdot) + \Delta t \cdot F(x, y, D_{00}u, D_{10}u, D_{01}u, D_{20}u, D_{11}u, D_{02}u, \ldots).$$



Neural ODE (NeurIPS 18)

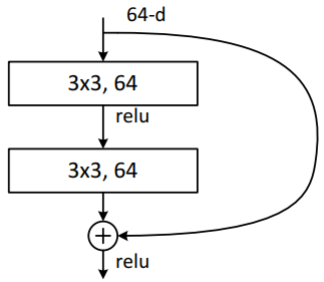
- ResNet architecture

$$\mathbf{h}_{t+1} = \mathbf{h}_t + f(\mathbf{h}_t, \theta_t)$$

- NeuralODE architecture

$$\frac{d\mathbf{h}(t)}{dt} = f(\mathbf{h}(t), t, \theta)$$

- T parameters are reduced to 1
- To optimize NODE, the backpropagation is also an ODE



$$\frac{d\mathbf{a}(t)}{dt} = -\mathbf{a}(t)^\mathsf{T} \frac{\partial f(\mathbf{z}(t), t, \theta)}{\partial \mathbf{z}}$$

$$\frac{dL}{d\theta} = -\int_{t_1}^{t_0} \mathbf{a}(t)^\mathsf{T} \frac{\partial f(\mathbf{z}(t), t, \theta)}{\partial \theta} dt$$

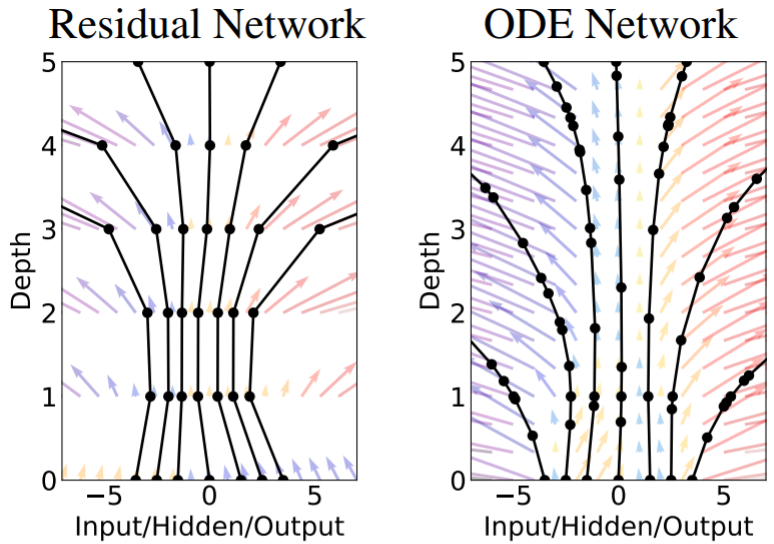
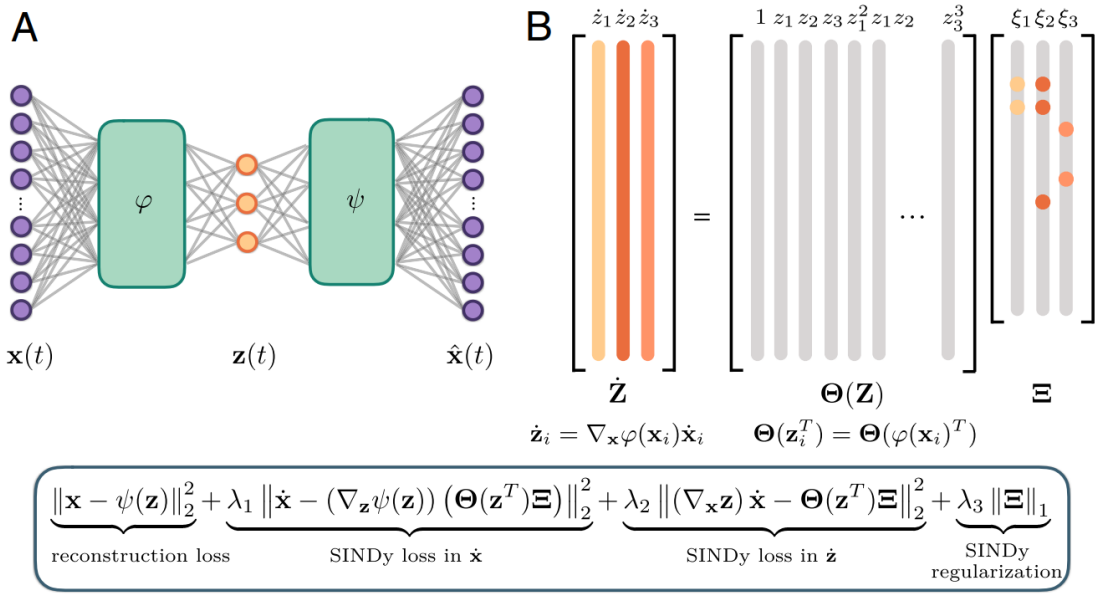
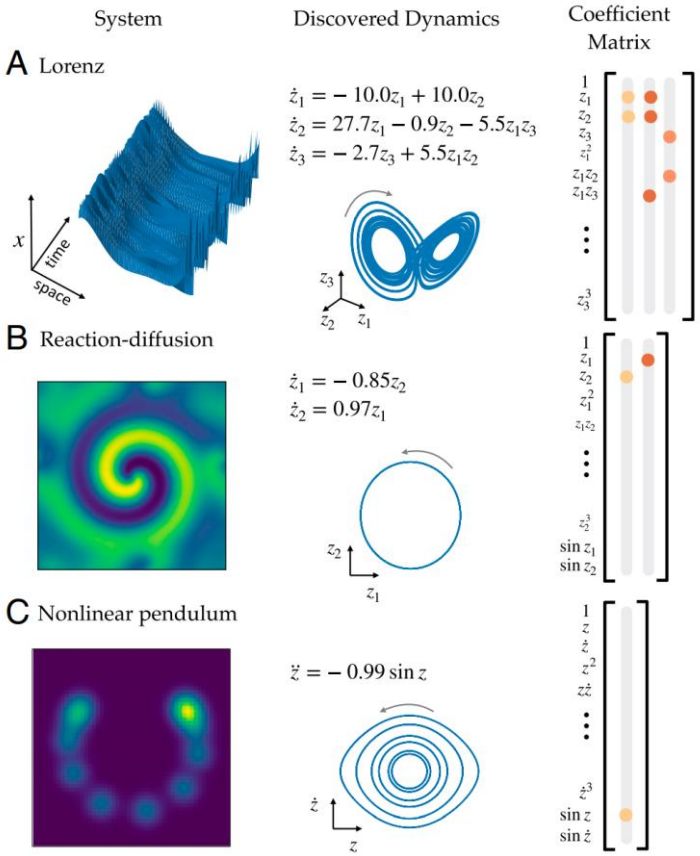


Figure 1: *Left:* A Residual network defines a discrete sequence of finite transformations. *Right:* A ODE network defines a vector field, which continuously transforms the state. *Both:* Circles represent evaluation locations.

Data driven discovery of governing equations (PNAS19)

- Finding terms of ODE for physical systems
- Use an auto-encoder architecture from data to data
- Use L1- regularization term to discovery dynamic terms





Advantages

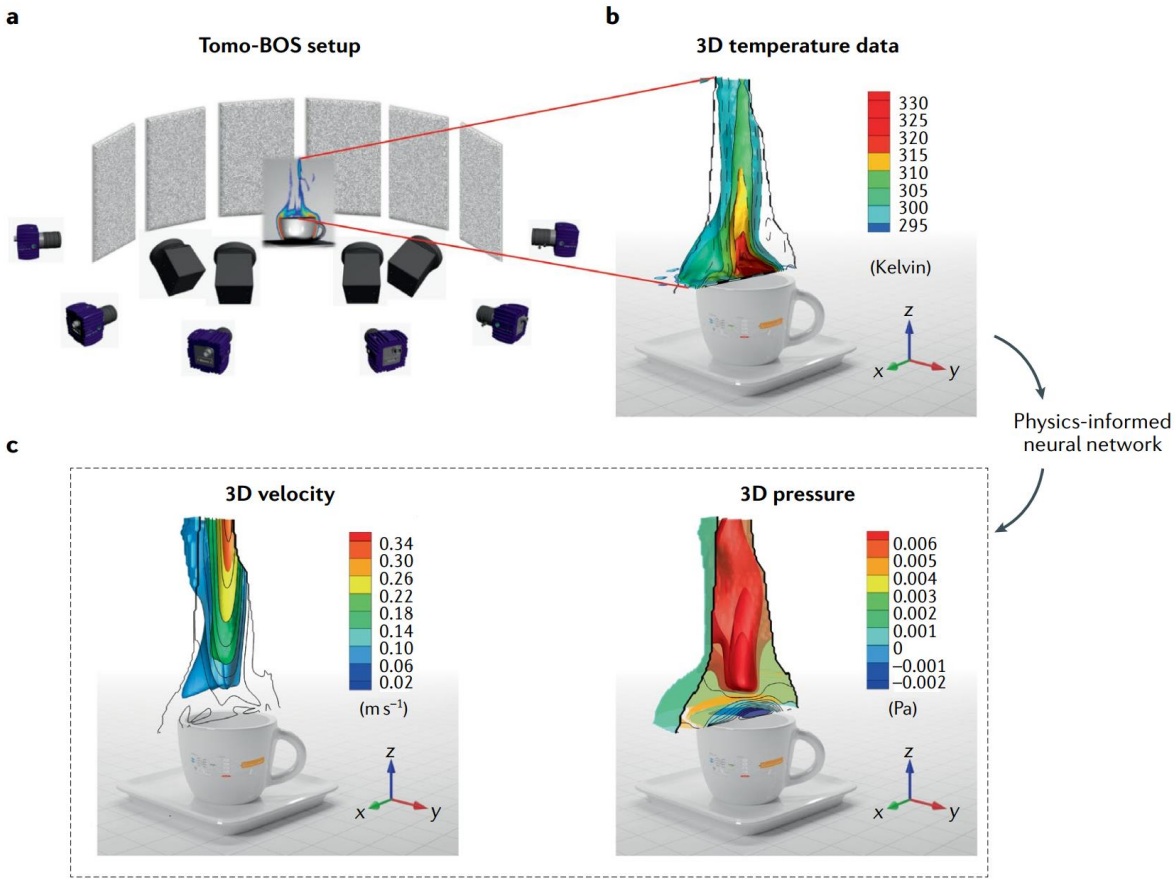
- Incomplete/ imperfect data, mesh free
- High dimensionality
- Stronger generalization with small data
- Uncertainty quantification
- Understanding deep learning from physics perspective

Applications highlights

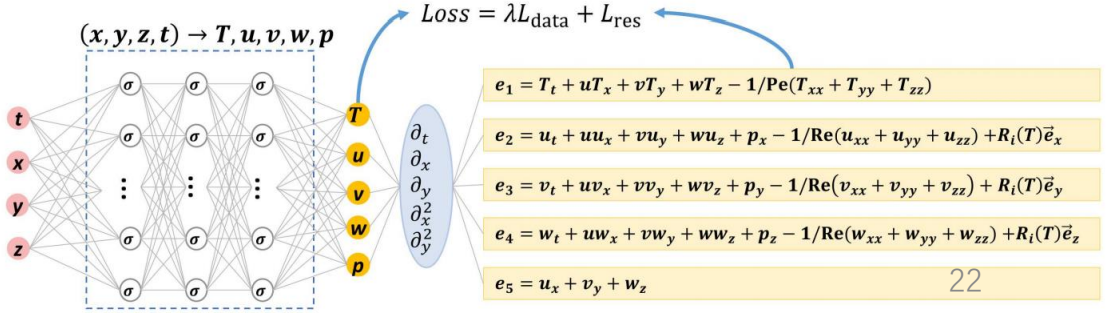
- Flow over espresso cup
- 4D flow MRI data
- Uncovering edge plasma dynamics from partial observations
- Studying transitions between metastable states of a distribution
- Thermodynamically consistent PINNs
- Quantum chemistry and molecular simulation
- Material Science
- Geographics

Applications– Flow over espresso cup

- Input: cameras recording the distortion of dot-patterns caused by air density variation
- Processing: Derive 3D temperature field
- Learning: use PINN to recover the velocity and pressure fields



- PINN details: use space and time coordinates as input , output the velocity and pressure fields, trained by minimizing temperature loss and conservation loss



Applications—4D MRI data

- Use PINN that obeys NS equation to reconstruct the velocity and pressure fields

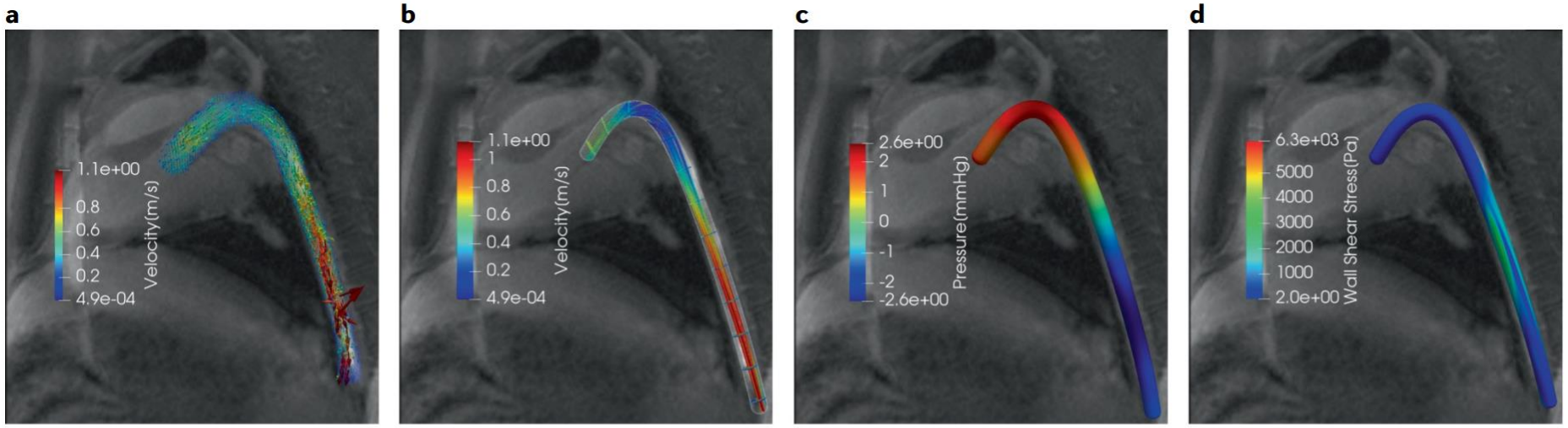


Fig. 3 | **Physics-informed filtering of in-vivo 4D-flow magnetic resonance imaging data of blood flow in a porcine descending aorta.** Physics-informed neural network (PINN) models can be used to de-noise and reconstruct clinical magnetic resonance imaging (MRI) data of blood velocity, while constraining this reconstruction to respect the underlying physical laws of momentum and mass conservation, as described by the incompressible Navier–Stokes equations. Moreover, a trained PINN model has the potential to aid the automatic segmentation of the arterial wall geometry and to infer important biomarkers such as blood pressure and wall shear stresses. **a** | Snapshot of in-vivo 4D-flow MRI measurements. **b–d** | A PINN reconstruction of the velocity field (panel **b**), pressure (panel **c**), arterial wall surface geometry and wall shear stresses (panel **d**). The 4D-flow MRI data were acquired by E. Hwuang and W. Witschey at the University of Pennsylvania. The PINN inference and visualization were performed by S. Wang, G. Kissas and P. Perdikaris at the University of Pennsylvania.

Applications—Uncovering edge plasma dynamics

- Learn the electric potential and field from single test discharge using PINN

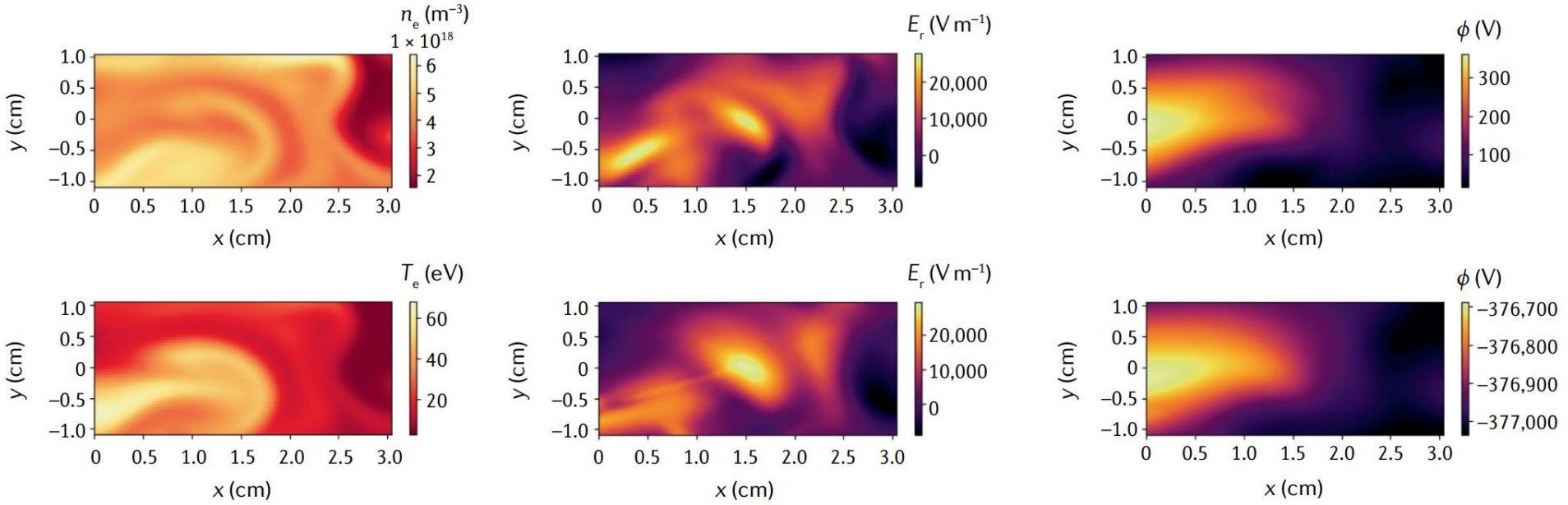


Fig. 4 | **Uncovering edge plasma dynamics.** One of the most intensely studied aspects of magnetic confinement fusion is edge plasma behaviour, which is critical to reactor performance and operation. The drift-reduced Braginskii two-fluid theory has for decades been widely used to model edge plasmas, with varying success. Using a 3D magnetized two-fluid model, physics-informed neural networks (PINNs) can be used to accurately reconstruct[141] the unknown turbulent electric field (middle panel) and underlying electric potential (right panel), directly from partial observations of the plasma's electron density and temperature from a single test discharge (left panel). The top row shows the reference target solution, while the bottom row depicts the PINN model's prediction. These 2D synthetic measurements of electron density and temperature over the duration of a single plasma discharge constitute the only physical dynamics observed by the PINNs from the 3D collisional plasma exhibiting blob-like filaments. ϕ, electric potential; E_r, electric field; n_e, electron density; T_e, electron temperature. Figure courtesy of A. Matthews, MIT.

Thermodynamically consistent PINN

- For PDE
- Traditional PINN loss function

$$\begin{aligned} \partial_t \boldsymbol{u} + \nabla \cdot \boldsymbol{F}(\boldsymbol{u}) = 0 \qquad & x, t \in \Omega, \text{ for all } i \\ \boldsymbol{u} = \boldsymbol{u}_0 \qquad & t = 0 \\ \boldsymbol{F}(\boldsymbol{u}) \cdot \hat{n} = g \qquad & x \in \Gamma_- \end{aligned}$$

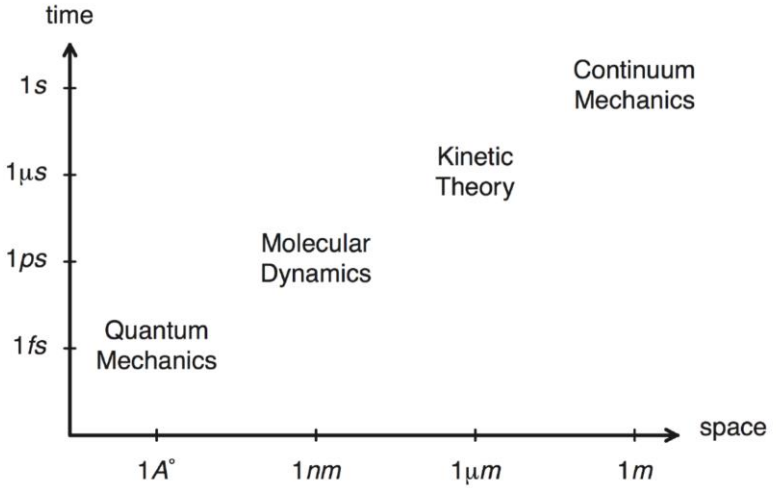
$$\mathcal{L}_{\text{PINN}} = ||\partial_t \boldsymbol{u} + \nabla \cdot \boldsymbol{F}(\boldsymbol{u})||^2_{\ell_2(\mathcal{D}_{int})} + \epsilon_0 ||\boldsymbol{u} - \boldsymbol{u}_0||^2_{\ell_2(\mathcal{D}_{IC})} + \epsilon_\Gamma ||\boldsymbol{F}(\boldsymbol{u}) \cdot \hat{n} - g||^2_{\ell_2(\mathcal{D}_{BC})},$$

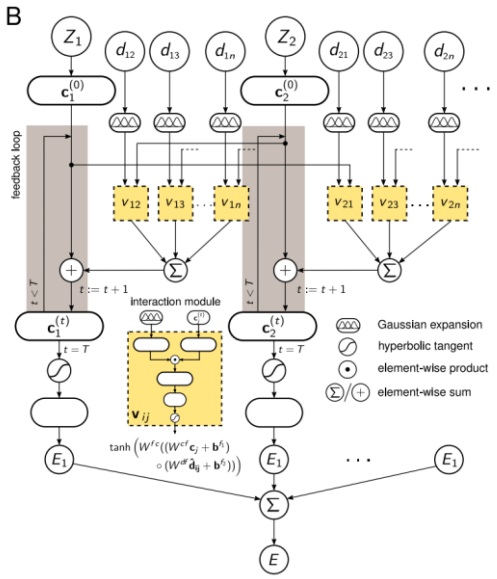
- Problems—in hyperbolic settings
- The authors define a extended flux, use a single loss to solve the PDE
- Control Volume PINN

$$\mathcal{L}_{cvPINN} = \sum_{c=1}^{N_c} \left| \int_{f \in \partial c} \widetilde{\boldsymbol{F}} \cdot d\boldsymbol{A} \right|^2. \qquad \widetilde{\boldsymbol{F}} = \begin{cases} \hat{\boldsymbol{F}}(\mathcal{NN}), & \text{if } \boldsymbol{x} \in \Omega \\ g\hat{n}, & \text{if } \boldsymbol{x} \in \Gamma_- \end{cases}$$

Applications—Quantum Chemistry and Molecular Simulations

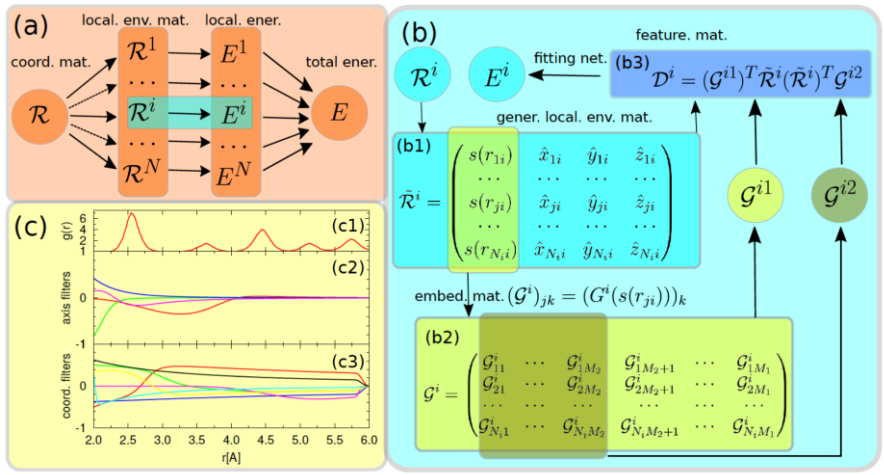
- Traditional levels of molecular simulation
- Combine phys and ML
- Data driven – use data to fit patterns (DTNN,SchNet)
- Physics driven—use NN to solve (DP, FermiNet)
- Generative Models





DTNN

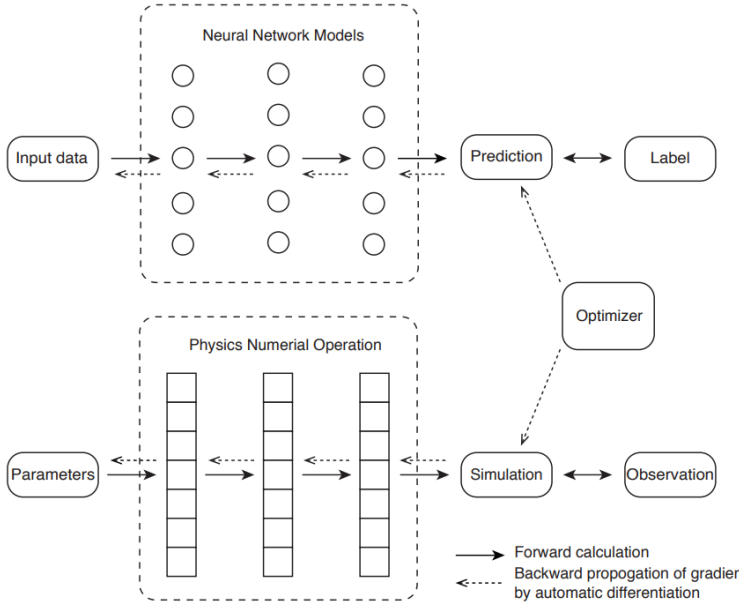
The whole sub-network consists of an encoding net $\mathcal{D}^i(\mathcal{R}^i)$ and a fitting net $E^i(\mathcal{D}^i)$.



Deep Potential

Applications—Material Science and Geoscience

- Different from molecular simulation—Some tasks involve more complex data (e.g. visual)
- Identify and precisely characterize a surface breaking crack in a metal plate
- Extracting mechanical properties of 3D-printed materials via instrumented indentation using multi-fidelity NN
- estimates subsurface properties, such as rock permeability and porosity, from seismic data by coupling NNs
- Combine NN and numerical solver for a wide class of seismic inversion problems, such as velocity estimation, earthquake location retrieval.
- More applications on material property prediction



Precipitation nowcasting using GAN (Nature)

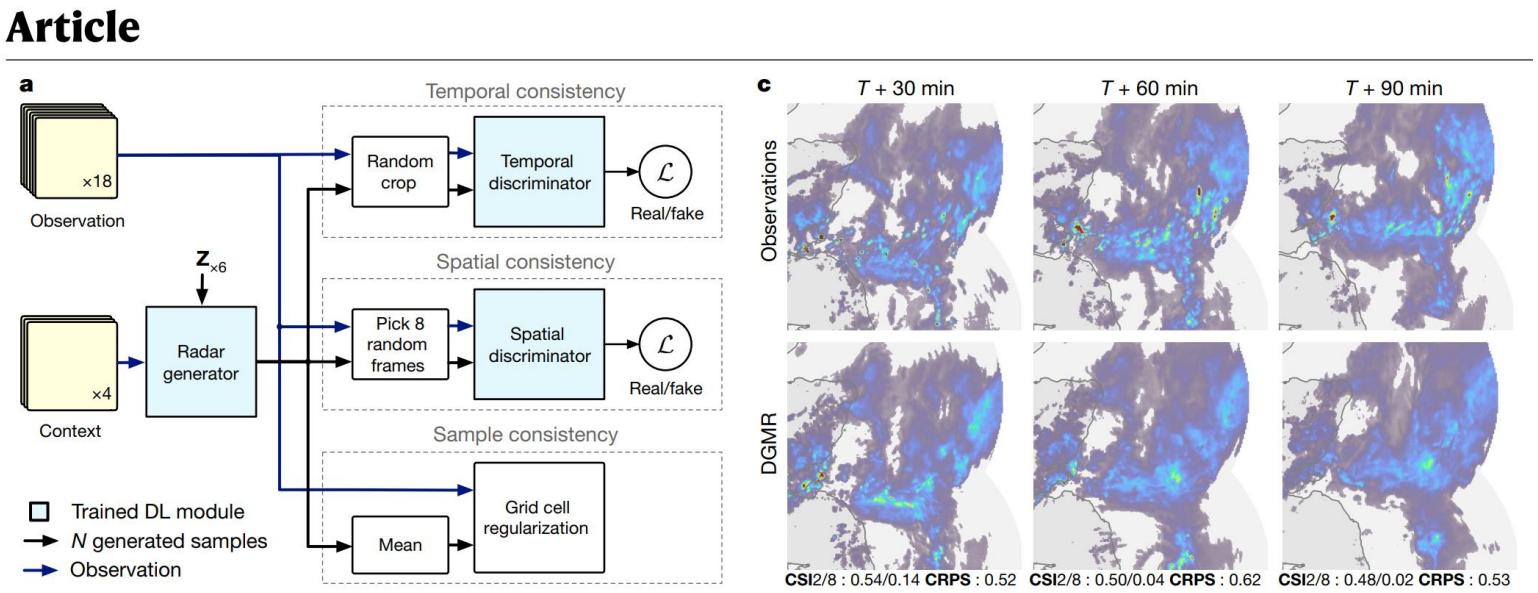
- Using a GAN to fit a generative model for precipitation data
- Model formulation

$$P(\mathbf{X}_{M+1:M+N} \mid \mathbf{X}_{1:M}) = \int P(\mathbf{X}_{M+1:M+N} \mid \mathbf{Z}, \mathbf{X}_{1:M}, \boldsymbol{\theta}) P(\mathbf{Z} \mid \mathbf{X}_{1:M}) \mathrm{d}\mathbf{Z}.$$

- Objective function

$$\mathcal{L}_G(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{X}_{1:M+N}}[\mathbb{E}_{\mathbf{Z}}[D(G_{\boldsymbol{\theta}}(\mathbf{Z}; \mathbf{X}_{1:M})) + T(\{\mathbf{X}_{1:M}; G_{\boldsymbol{\theta}}(\mathbf{Z}; \mathbf{X}_{1:M})\})] - \lambda \mathcal{L}_R(\boldsymbol{\theta})];$$

$$\mathcal{L}_R(\boldsymbol{\theta}) = \frac{1}{HWN} ||(\mathbb{E}_{\mathbf{Z}}[G_{\boldsymbol{\theta}}(\mathbf{Z}; \mathbf{X}_{1:M})] - \mathbf{X}_{M+1:M+N}]) \odot w(\mathbf{X}_{M+1:M+N})||_1.$$



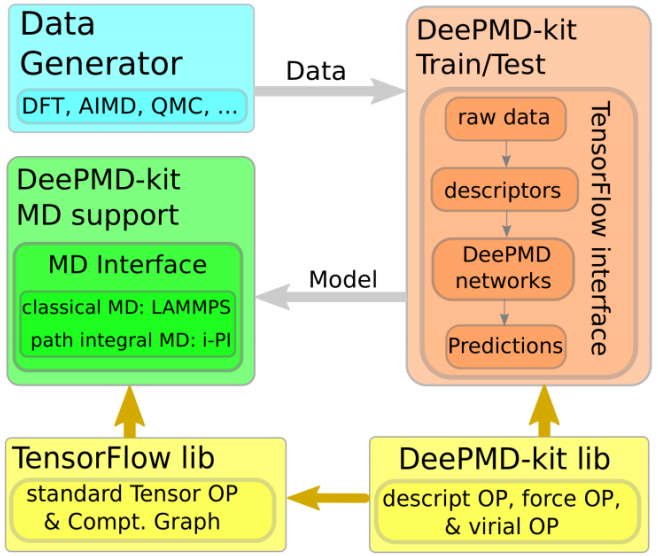
Limitations

- Difficulty in learning high frequency functions /F-principle, multiscale problems
- Algorithms and architectures limitations/ complex loss, unstable training, activation function
- Benchmarks , datasets and metrics/ data collection, for phys and chem
- No theoretical framework for PINN(like error/ well-posedness/training analysis)
- Code platforms, PINN requires higher/fractional order derivatives, scalable algo for large scale problem

Software

- PINN code platforms

Software name	Usage	Language	Backend	Ref.
DeepXDE	Solver	Python	TensorFlow	154
SimNet	Solver	Python	TensorFlow	155
PyDEns	Solver	Python	TensorFlow	156
NeuroDiffEq	Solver	Python	PyTorch	157
NeuralPDE	Solver	Julia	Julia	158
SciANN	Wrapper	Python	TensorFlow	159
ADCME	Wrapper	Julia	TensorFlow	160
GPyTorch	Wrapper	Python	PyTorch	161
Neural Tangents	Wrapper	Python	JAX	162



- TensorFlow: efficient network operators
- LAMMPS, i-PI; MPI/GPU support.

- Related code platforms—DeepPMD-kit, Taichi, DGL model zoo(MPNN, SchNet)
- Recommend libs—DeepXDE

Future directions

- Digital twins/ a concept first to describe the digital copy of an engine manufactured in their factories, by assimilating real measurements to calibrate computational models, a digital twin aims to replicate the behavior of a living or non-living physical entity in silico (sim2real, real2sim?)
- Fusion/transformation/interpretations between models
- Searching for intrinsic variables, representations

Strengths of ML and Phys

Machin Learning

- Data Driven
- Deep Neural Networks
- Optimization toolkits

Physics

- Principle equation
- Numerical/ Analytical solver
- Empirical laws or structural laws for systems

Thanks

References

- Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A.& Vandergheynst, P. Geometric deep learning: going beyond Euclidean data. IEEE Signal Process. Mag. 34,18–42 (2017).
- Bruna, J. & Mallat, S. Invariant scattering convolution networks. IEEE Trans. Pattern Anal. Mach. Intell. 35, 1872–1886 (2013)
- Jin, P., Zhang, Z., Zhu, A., Tang, Y. & Karniadakis, G. E.SympNets: intrinsic structure-preserving symplecticnetworks for identifying Hamiltonian systems. Neural Netw. 132, 166–179 (2020).
- Quantum-Chemical Insights from Deep Tensor Neural Networks https://arxiv.org/abs/1609.08259
- Sirignano, J. & Spiliopoulos, K. DGM: a deep learning algorithm for solving partial differential equations. J. Comput. Phys. 375, 1339–1364 (2018).
- Raissi, M., Perdikaris, P. & Karniadakis, G. E.Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J. Comput. Phys. 378, 686–707 (2019).
- Wu, J. L. et al. Enforcing statistical constraints in generative adversarial networks for modeling chaotic dynamical systems. J. Comput. Phys. 406, 109209 (2020).
- Shah, V. et al. Encoding invariances in deep generative models. Preprint at arXiv https://arxiv.org/abs/1906.01626 (2019).
- Lanthaler, S., Mishra, S. & Karniadakis, G. E. Error estimates for DeepONets: a deep learning framework in infinite dimensions. Preprint at arXiv https://arxiv.org/abs/2102.09618 (2021).

References

- Pfau, D., Spencer, J. S., Matthews, A. G. &Foulkes, W. M. C. Ab initio solution of the manyelectron Schrödinger equation with deep neural networks. Phys. Rev. Res. 2, 033429 (2020).
- Li, Z. et al. Multipole graph neural operator for parametric partial differential equations. in Adv. Neural Inf. Process. Syst. (2020).
- Darbon, J. & Meng, T. On some neural network architectures that can represent viscosity solutions of certain high dimensional Hamilton-Jacobi partial differential equations. J. Comput. Phys. 425, 109907(2021)
- Long, Z., Lu, Y., Ma, X. & Dong, B. PDE-Net: learning PDEs from data. Proc. Int. Conf. Mach. Learn. 80, 3208–3216 (2018).
- Raissi, M., Yazdani, A. & Karniadakis, G. E. Hidden fluid mechanics: learning velocity and pressure fields from flow visualizations. Science 367, 1026–1030 (2020).
- Mathews, A., Francisquez, M., Hughes, J. & Hatch, D.Uncovering edge plasma dynamics via deep learning from partial observations. Preprint at arXiv https://arxiv.org/abs/2009.05005 (2020)
- Shukla, K., Di Leoni, P. C., Blackshire, J., Sparkman, D. & Karniadakis, G. E. Physics-informed neural network for ultrasound nondestructive quantification of surface breaking cracks. J. Nondestruct. Eval. 39, 1–20 (2020).
- Lu, L. et al. Extraction of mechanical properties of materials through deep learning from instrumented indentation. Proc. Natl Acad. Sci. USA 117, 7052–7062 (2020).

References

- Zhu, W., Xu, K., Darve, E. & Beroza, G. C. A general approach to seismic inversion with automatic differentiation. Preprint at arXiv https://arxiv.org/abs/2003.06027 (2020).
- Li, D., Xu, K., Harris, J. M. & Darve, E. Coupled time-lapse full-waveform inversion for subsurface flow problems using intrusive automatic differentiation. Water Resour. Res. 56, e2019WR027032 (2020)
- http://web.math.princeton.edu/~weinan/IPAM.pdf
- http://web.math.princeton.edu/~weinan/control.pdf
- http://web.math.princeton.edu/~weinan/ICIAM.pdf