香港科技大学(广州)
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY (GUANGZHOU)

FUNH 5000:

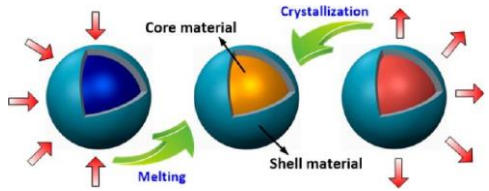
AI in Advanced Materials

Zecheng GAN

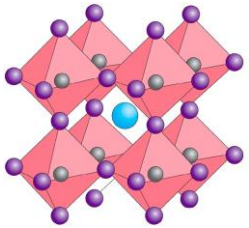
Advanced Materials Thrust
The Hong Kong University of Science and Technology (Guangzhou)

Advanced Materials for Sustainable Economy

Phase-Change Material

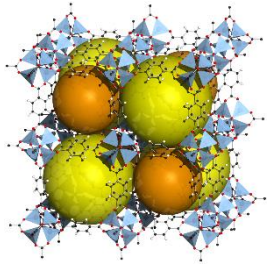


Perovskite

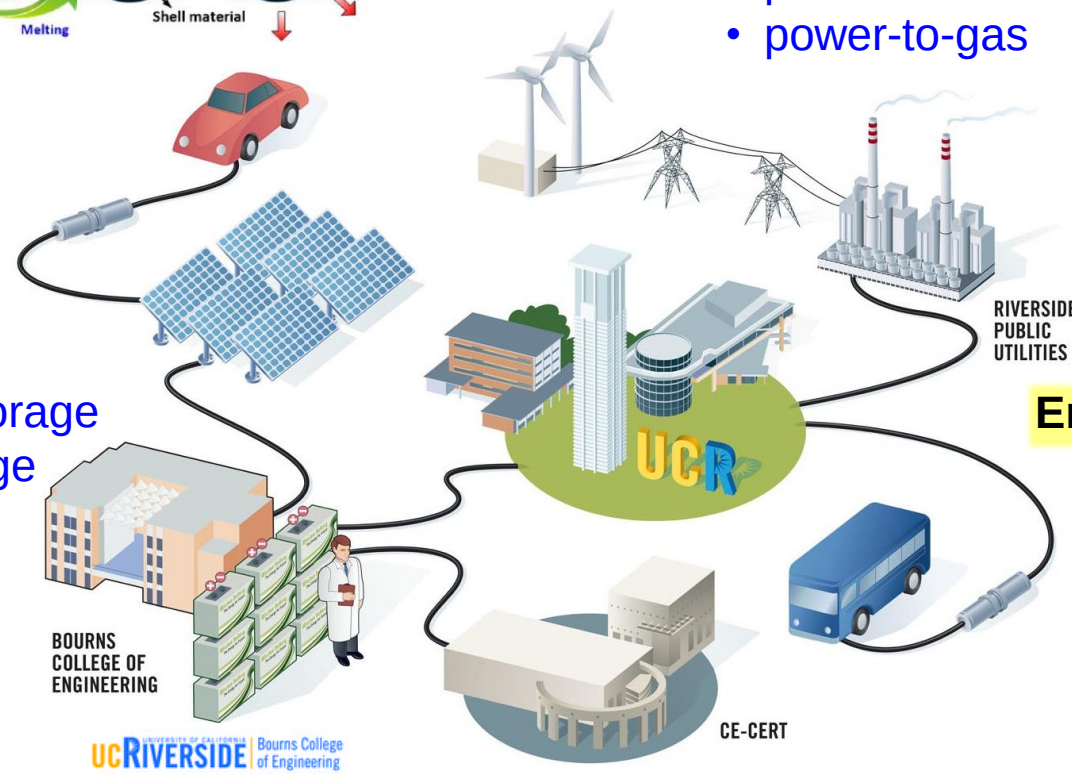


Energy Storage

- solar energy storage
- H_2 & CH_4 storage

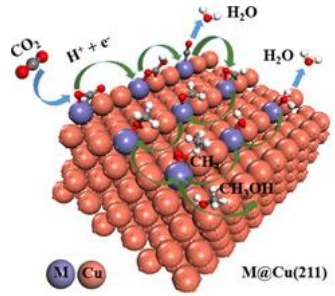


MOF, COF, etc.



Energy Conversion

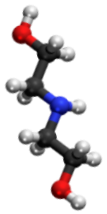
- power-to-chemicals
- power-to-heat
- power-to-gas



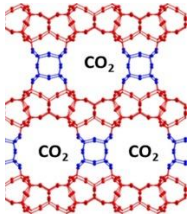
electrochemical CO_2 reduction

Emission Reduction

- solvents
- adsorbents



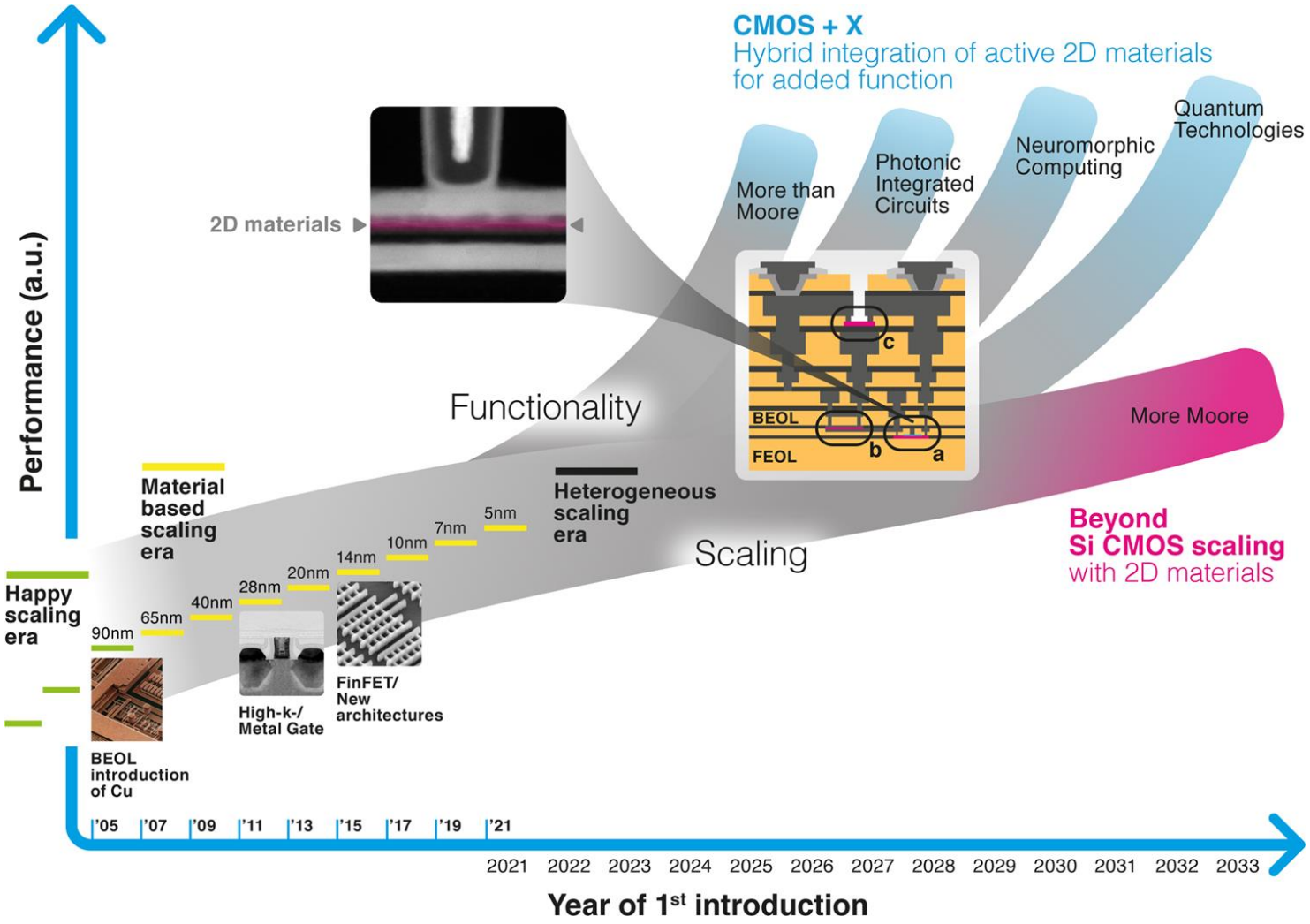
Amines



Zeolites

Advanced Materials are Key to Sustainable Energy & Environment

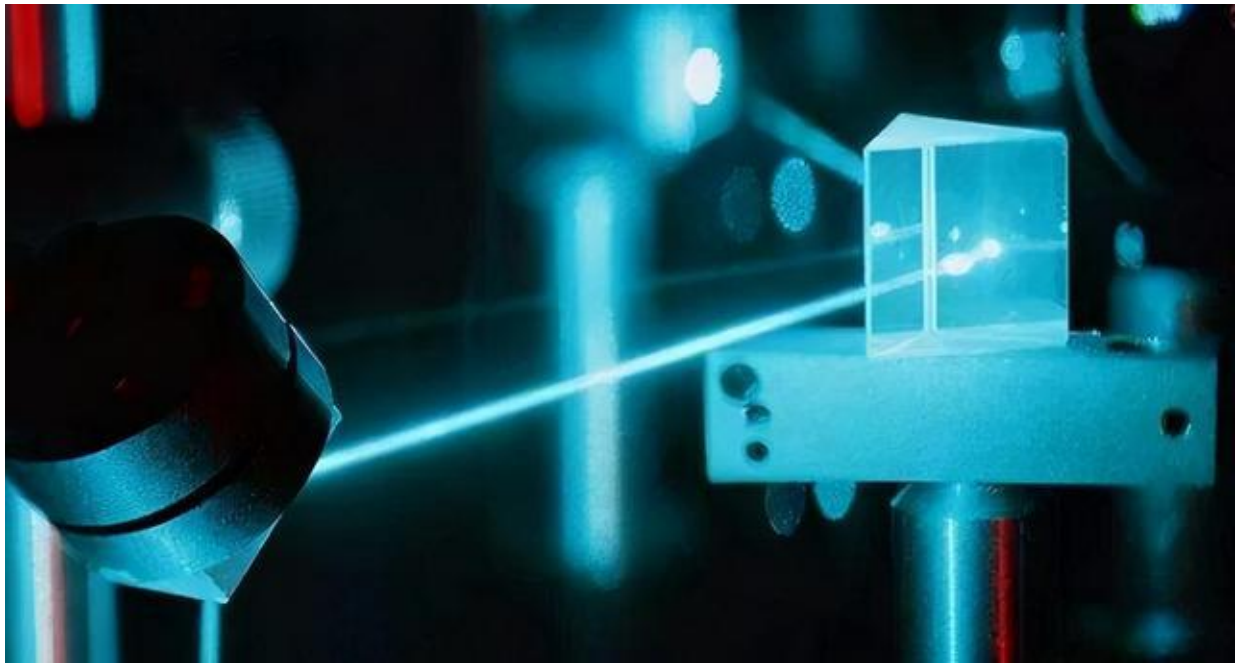
Advanced Materials for Microelectronics



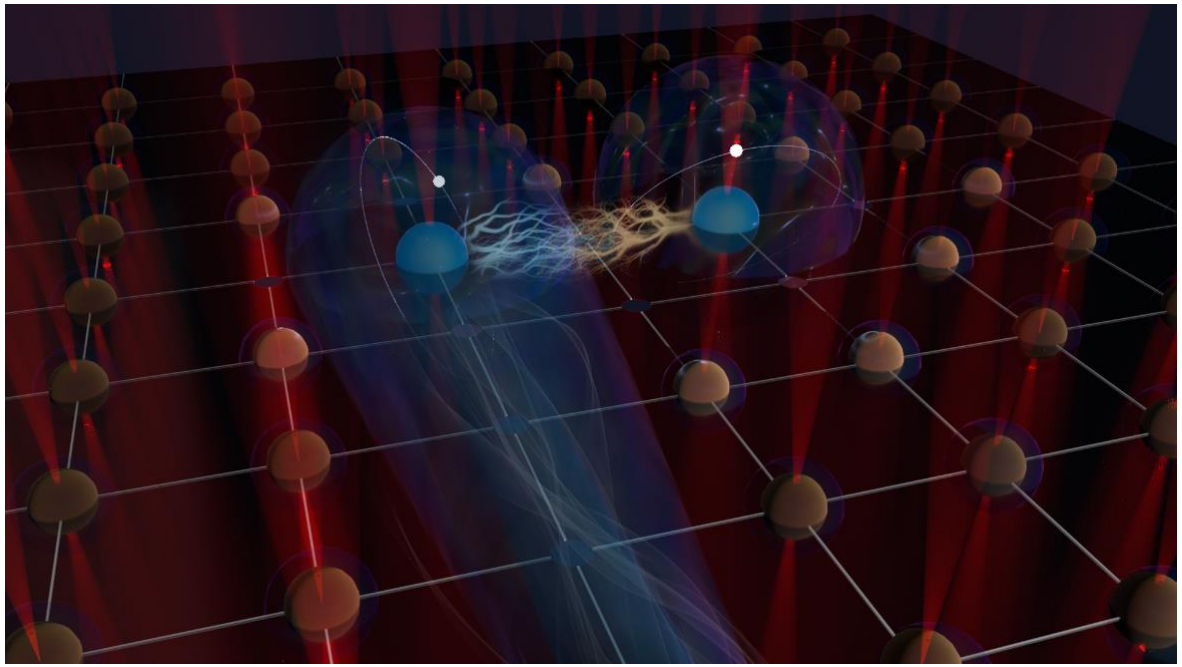
- Graphene and 2D materials for future electronics

Advanced Materials are Key to Microelectronics

Advanced Materials for Quantum Devices



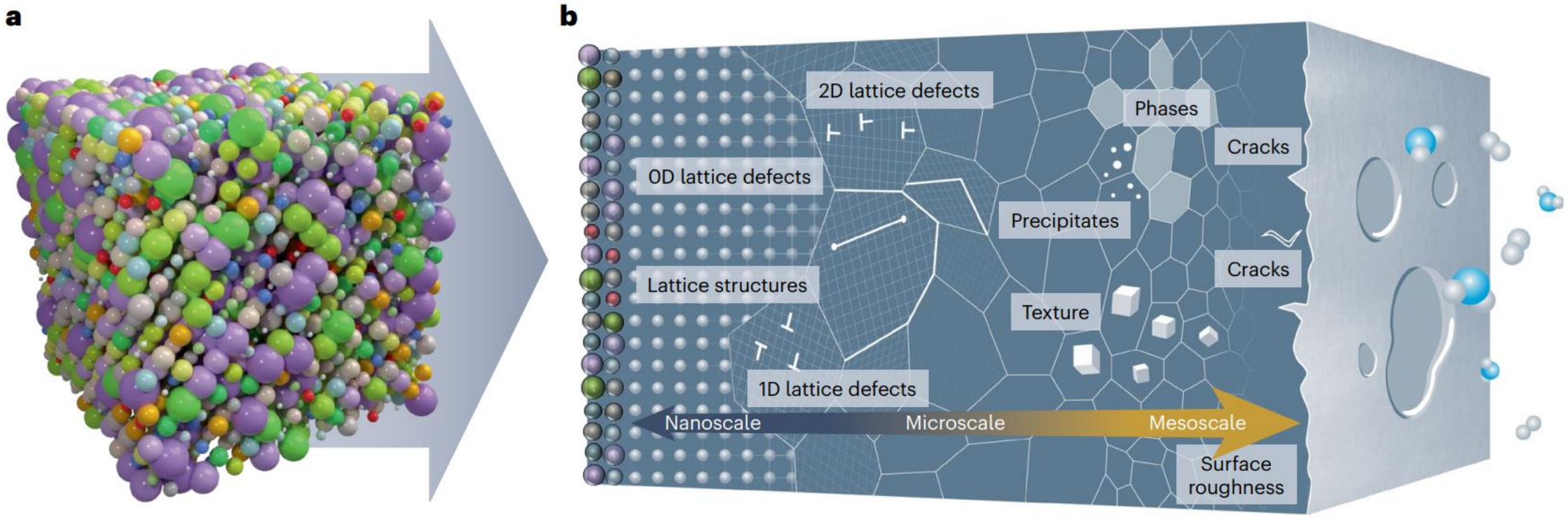
LSBO crystal for non-linear optics



Ryderberg atom arrays for quantum computing

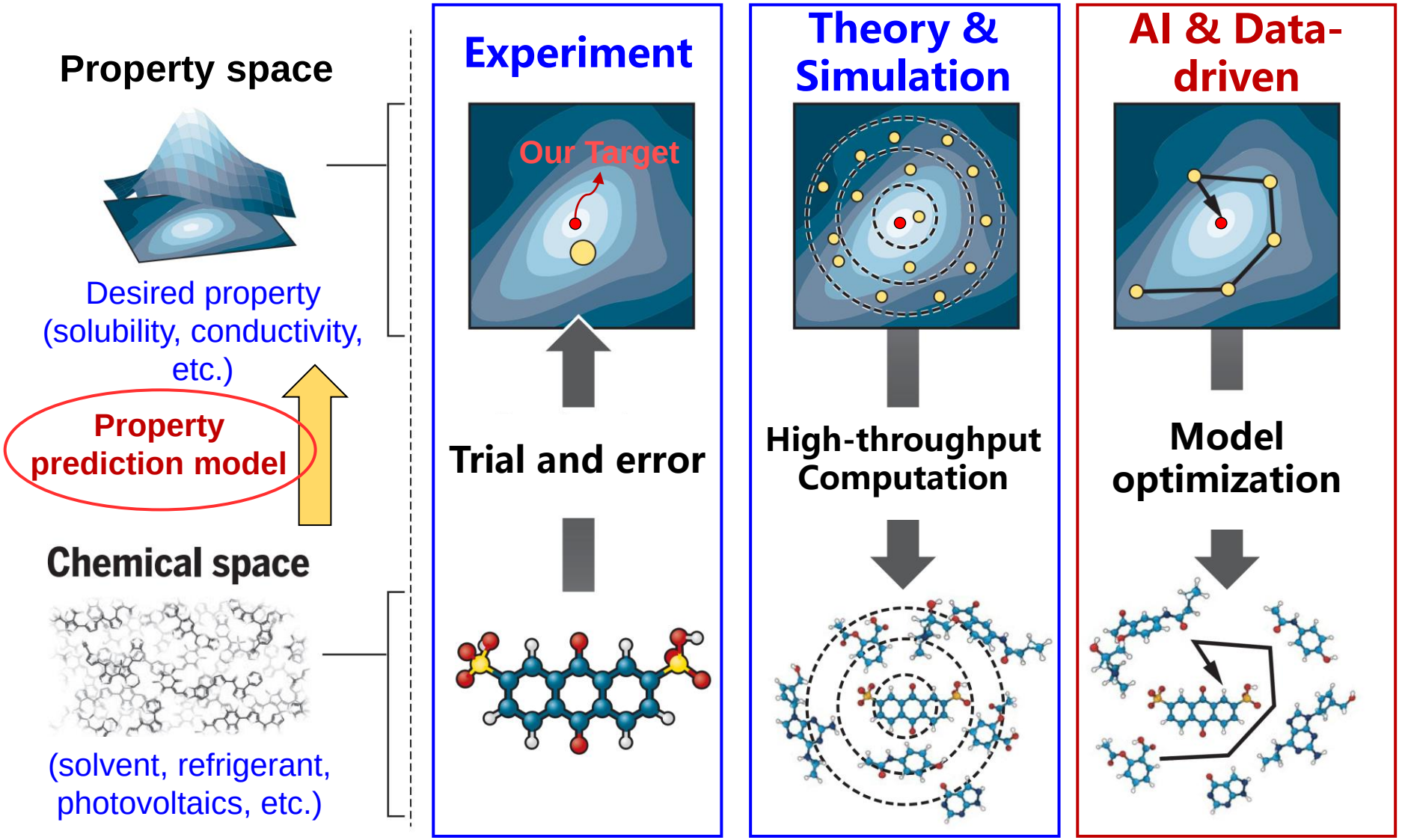
Advanced Materials are Key to Quantum Devices

The Challenge in Advanced Materials



- Chemical complexity influences structure complexity at multi-scales
- Understanding and quantifying the link between chemical and structural complexity is one of the grand challenges

Research Methods for Advanced Materials

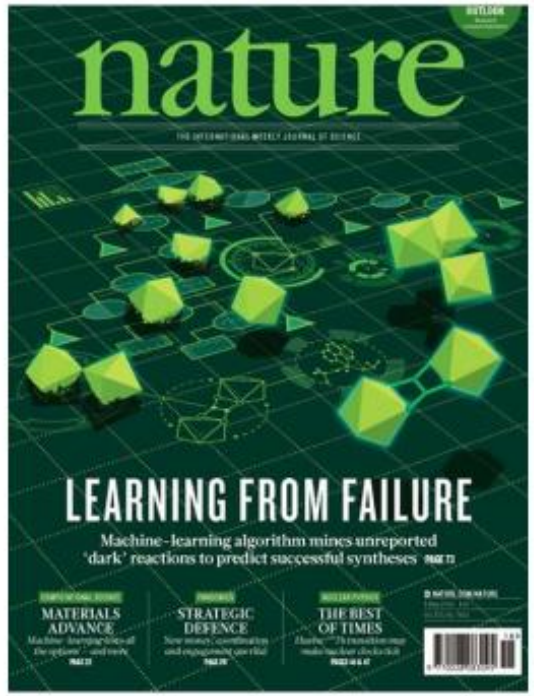


Sanchez-Lengeling and Aspuru-Guzik, *Science* (2018)

Fast Development of AI

doi:10.1038/nature.2016.19553

doi:10.1038/nature21056



doi:10.1038/nature17439

ARTICLE

doi:10.1038/nature24270

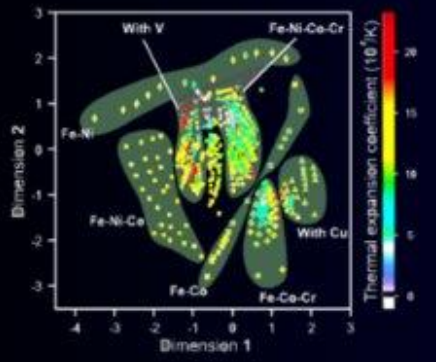
Mastering the game of Go without human knowledge

David Silver[1]*, Julian Schrittwieser[1]*, Karen Simonyan[1]*, Ioannis Antonoglou[1], Aja Huang[1], Arthur Guez[1], Thomas Hubert[1], Lucas Baker[1], Matthew Lai[1], Adrian Bolton[1], Yutian Chen[1], Timothy Lillicrap[1], Fan Hui[1], Laurent Sifre[1], George van den Driessche[1], Thore Graepel[1] & Demis Hassabis[1]

Alpha-go zero

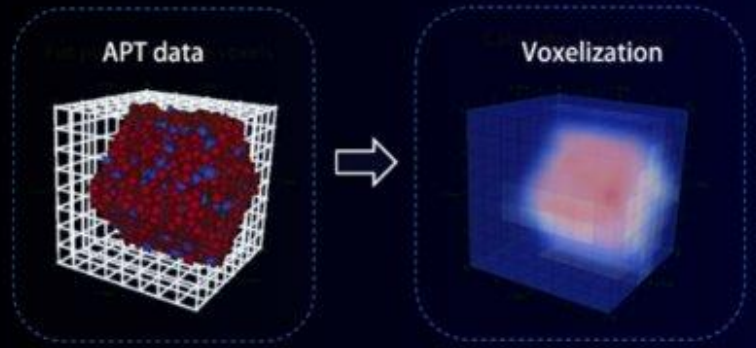
AI for Advanced Materials

Advanced materials design



Text and data mining

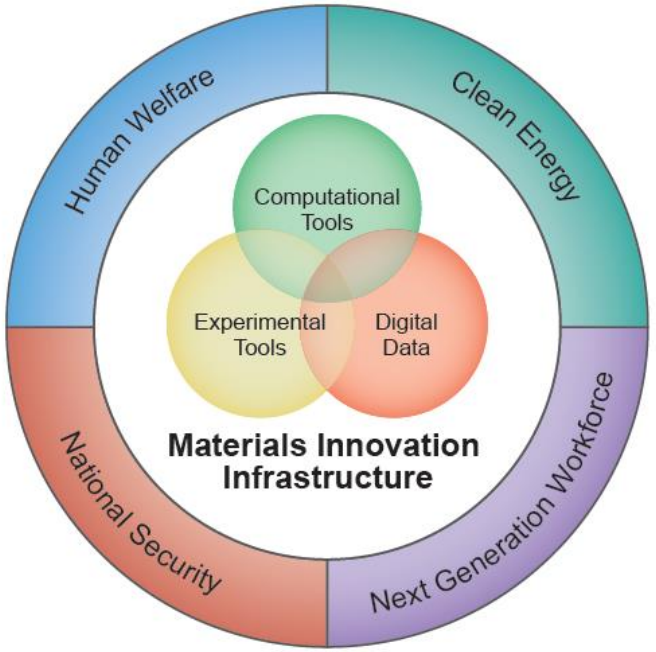
Experimental data analysis



Explainable artificial intelligence

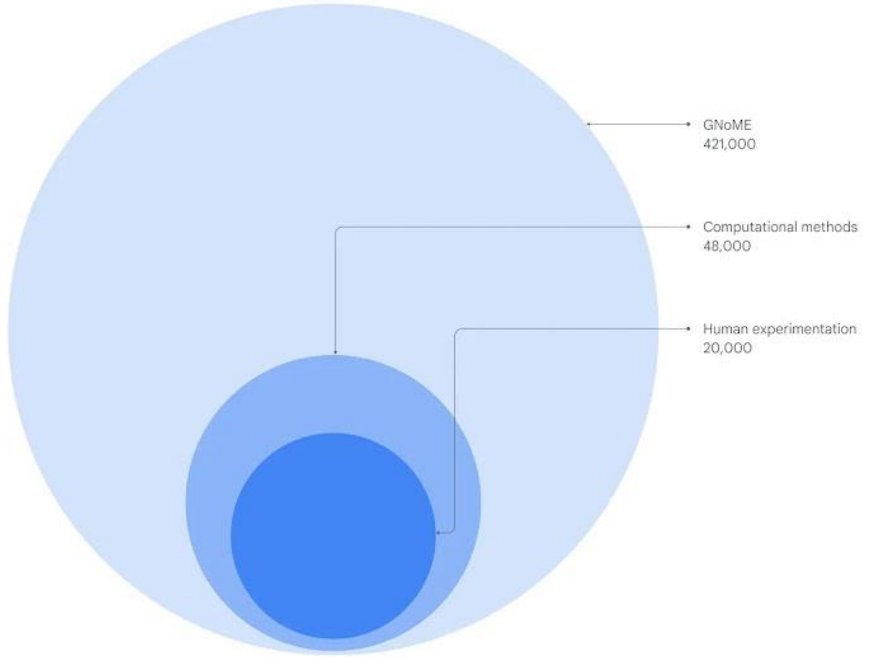
Artificial Intelligence

AI for Advanced Materials



In June 2011, the US President Obama announced AMP(Advanced Manufacturing Partnership), in which "**Materials Genome Initiative**"(**MGI**) is a key component. The core is to integrate high-throughput computation, experimentation and databases, "to discover, develop, manufacture, and deploy advanced materials at twice the speed than is possible today".

GNoME for materials discovery



nature

Explore content ⌄ About the journal ⌄ Publish with us ⌄

nature › articles › article

Article | Open access | Published: 29 November 2023

Scaling deep learning for materials discovery

Amil Merchant ✉, Simon Batzner, Samuel S. Schoenholz, Muratahan Aykol, Gowoon Cheon & Ekin Dogus Cubuk ✉

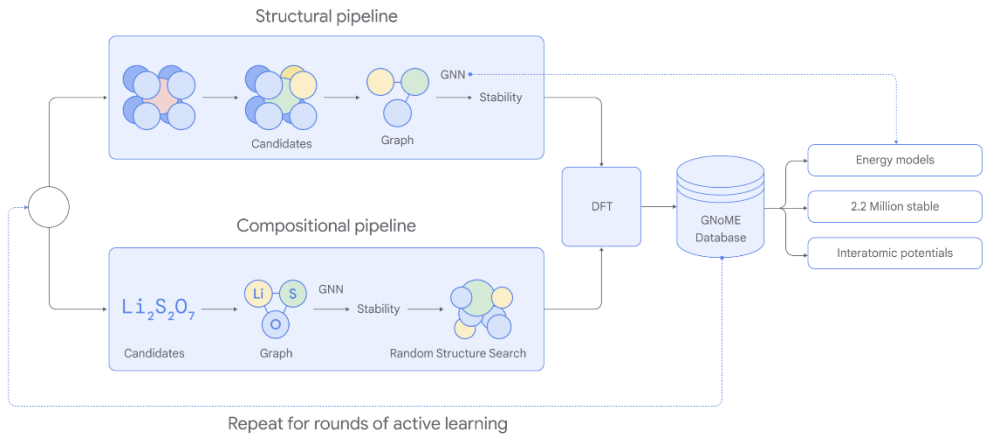
Nature **624**, 80–85 (2023) | Cite this article

RESEARCH

Millions of new materials discovered with deep learning

29 NOVEMBER 2023

Amil Merchant and Ekin Dogus Cubuk



Explainable AI: model discovery

SCIENCE ADVANCES | RESEARCH ARTICLE

APPLIED MATHEMATICS

Data-driven discovery of partial differential equations

Samuel H. Rudy,[1]* Steven L. Brunton,[2] Joshua L. Proctor,[3] J. Nathan Kutz[1]

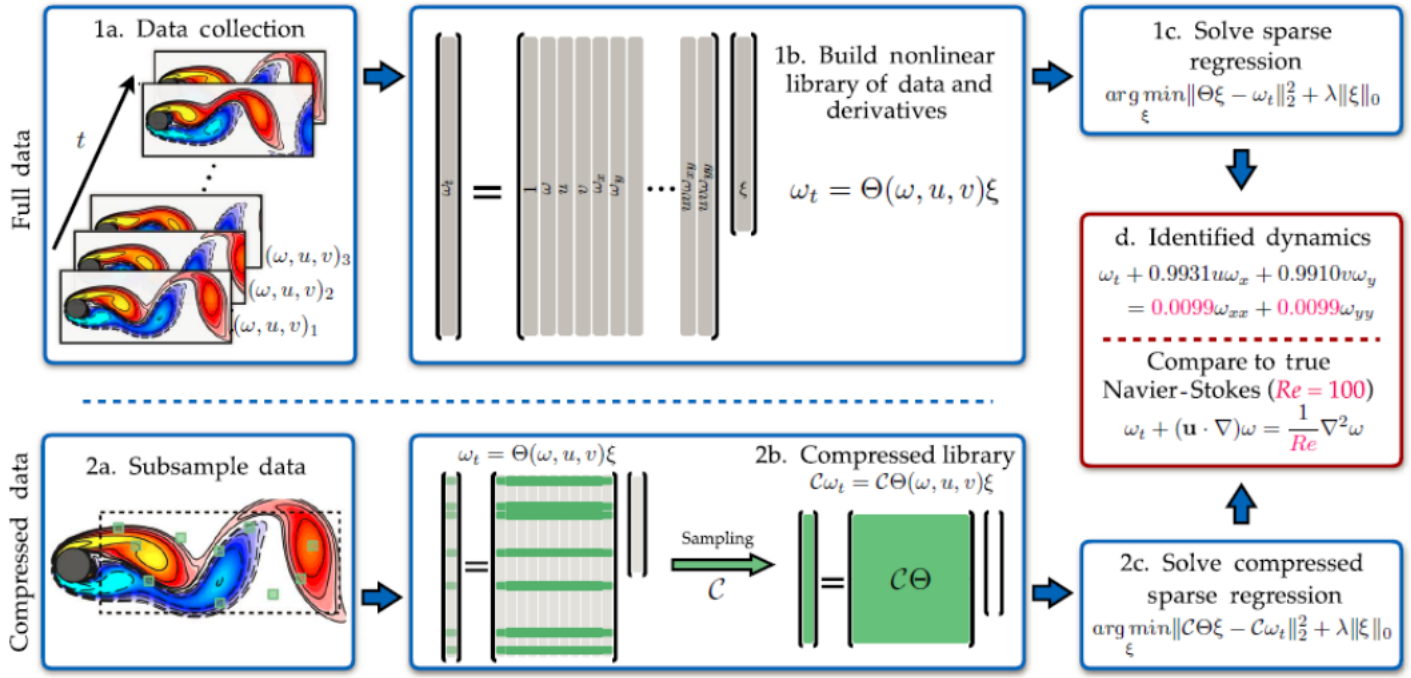
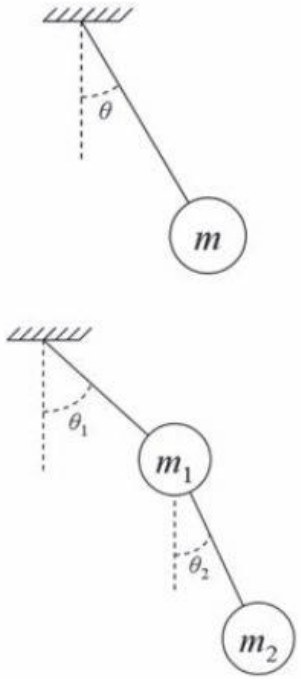
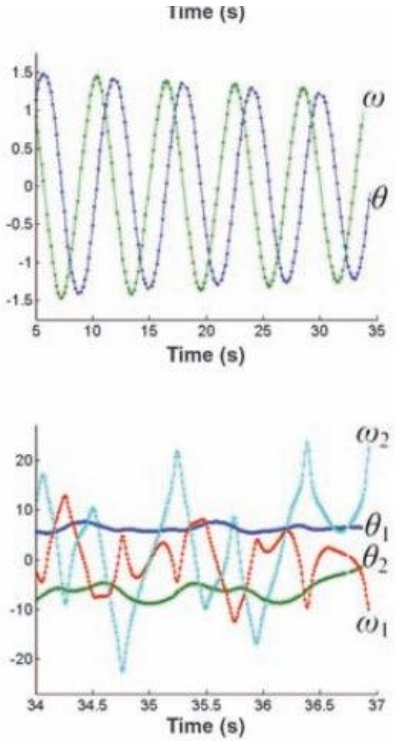


Fig. 1. Steps in the PDE functional identification of nonlinear dynamics (PDE-FIND) algorithm, applied to infer the Navier-Stokes equations from data. (1a) Data are collected as snapshots of a solution to a PDE. (1b) Numerical derivatives are taken, and data are compiled into a large matrix $\boldsymbol{\Theta}$, incorporating candidate terms for the PDE. (1c) Sparse regressions are used to identify active terms in the PDE. (2a) For large data sets, sparse sampling may be used to reduce the size of the problem. (2b) Subsampling the data set is equivalent to taking a subset of rows from the linear system in Eq. 2. (2c) An identical sparse regression problem is formed but with fewer rows. (d) Active terms in ξ are synthesized into a PDE.

Importance of Experimental Data

Distilling Free-Form Natural Laws from Experimental Data

Michael Schmidt[1] and Hod Lipson[2,3]*

Physical System	Schematic	Experimental Data	Inferred Laws
			$1.37 \cdot \omega^2 + 3.29 \cdot \cos(\theta)$ **Lagrangian** $2.71\alpha + 0.054\omega - 3.54\sin(\theta)$ **Equation of motion** $(x - 77.72)^2 + (y - 106.48)^2$ **Circular manifold** $\omega_1^2 + 0.32\omega_2^2 - 124.13\cos(\theta_1) - 46.82\cos(\theta_2) + 0.82\omega_1\omega_2\cos(\theta_1 - \theta_2)$ **Hamiltonian**

Material Big Data

Public Material Databases

AFLOW
American Mineralogist Crystal Structure Database
Computer Coupling of Phase Diagrams and Thermochemistry (CALPHAD)
Cambridge Structural Database
CatApp
ChEMBL
ChemSpider
Citrination
Computational Materials Repository
CoRE MOF
Crystallography Open Database
Dark Reactions Project
GDB Database
Harvard Clean Energy Project
The Inorganic Crystal Structure Database (ICSD)
Materials Project
MatNavi
MatWeb
Mindat.org
NanoHUB
Nanomaterials Registry
Nanoporous Materials Explorer
National Institute of Standards and Technology (NIST) Chemistry WebBook
NIST Materials Data Repository
NIST Interatomic Potentials Repository
NIST Standard Reference Data
The Novel Materials Discovery (NOMAD) Laboratory
National Renewable Energy Laboratory (NREL) Materials Database
Open Quantum Materials Database
PubChem
The Thermoelectrics Design Laboratory (TEDesignLab)
University of California, Santa Barbara (UCSB) thermoelectric database
ZINC

Different Types of Materials

- ✓ Inorganic compounds
- ✓ Organic compounds
- ✓ Metal-organic compounds
- ✓ Nanoporous materials
- ✓ Thermoelectric materials
- ✓ Catalytic materials
- ✓ Drug-like molecules
- ✓ Absorber
- ✓ Superconductor
- ✓ Thermoplastics
- ✓ Minerals, rocks, etc
- ✓ Nanomaterials
- ✓ Semi-conductors
- ✓ Fibers

…

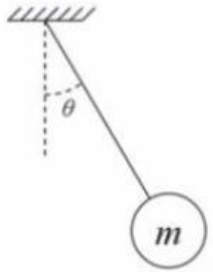
Different Types of Properties

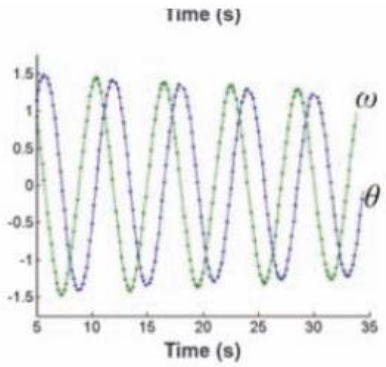
- ✓ Thermodynamic property
- ✓ Kinetic property
- ✓ Surface property
- ✓ Catalytic property
- ✓ Topological property
- ✓ Geometrical property
- ✓ Melting and boiling points
- ✓ Thermochemistry property
- ✓ Spectroscopic property
- ✓ Interatomic potential
- ✓ Force field
- ✓ Biological activity
- ✓ Toxicity
- ✓ Thermal conductivity
- ✓ Heat capacity
- ✓ Density

…

Importance of Physical Picture

Science 324:81, 2009





$1.37 \cdot \omega^2 + 3.29 \cdot \cos(\theta)$

Lagrangian

$2.71\alpha + 0.054\omega - 3.54\sin(\theta)$

Equation of motion

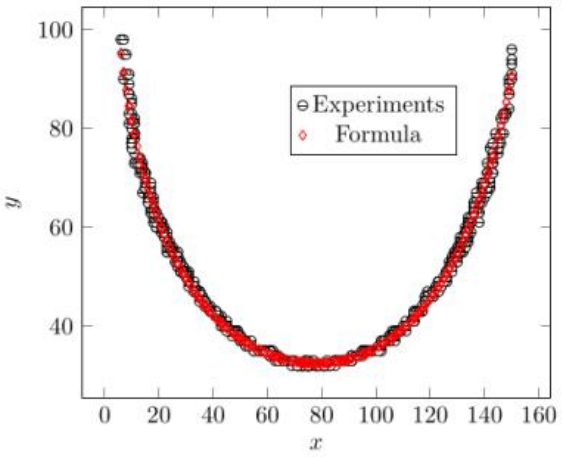
$(x - 77.72)^2 + (y - 106.48)^2$

Circular manifold

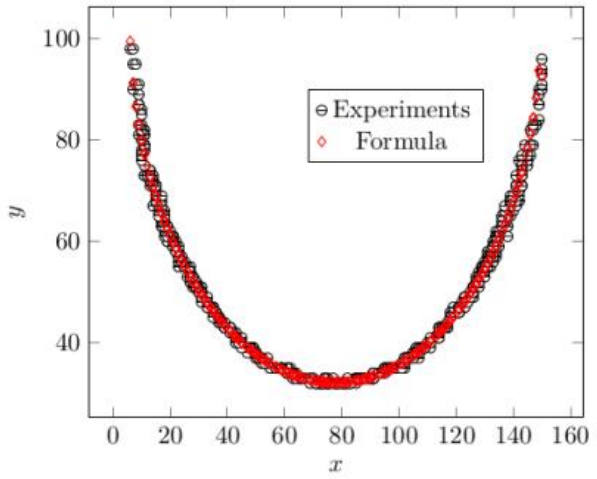
$Q_{ic} = 0.05$
$R_c = 0.1$
Operator set: sqrt, log, sin, exp

1. $2.3869 \times 10^{-5}x + 0.0003x\sqrt{\frac{1}{y}} - 0.0800\sqrt{\frac{1}{x}} - 0.0924\sqrt{\frac{1}{y}} + \sqrt{\frac{1}{xy}} = 0$
2. $-208.442y - 155.60x + 11717.797 + 1.002y^2 + x^2 = 0$

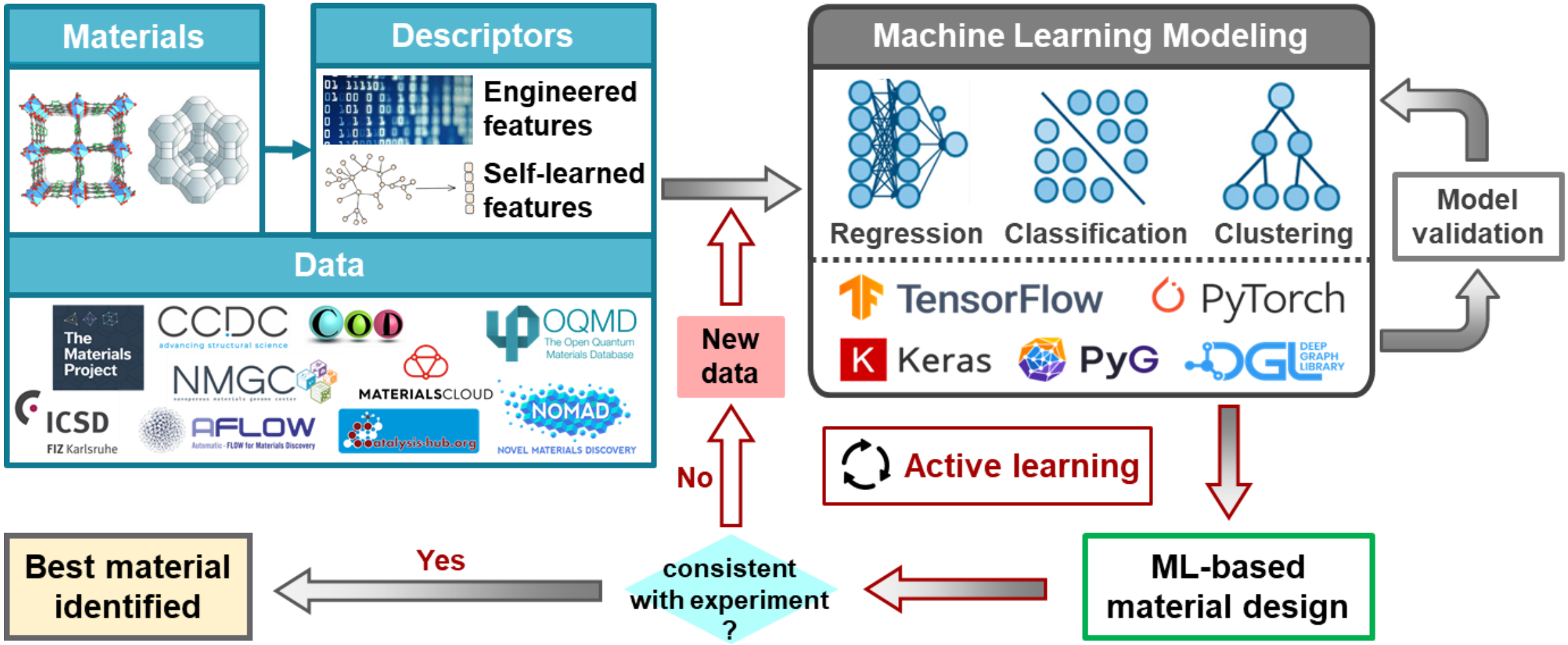
$2.3869 \times 10^{-5}x + 0.0003x\sqrt{(1/y)} - 0.0800\sqrt{(1/x)} - 0.0924\sqrt{1/y} + \sqrt{1/(xy)} = 0$



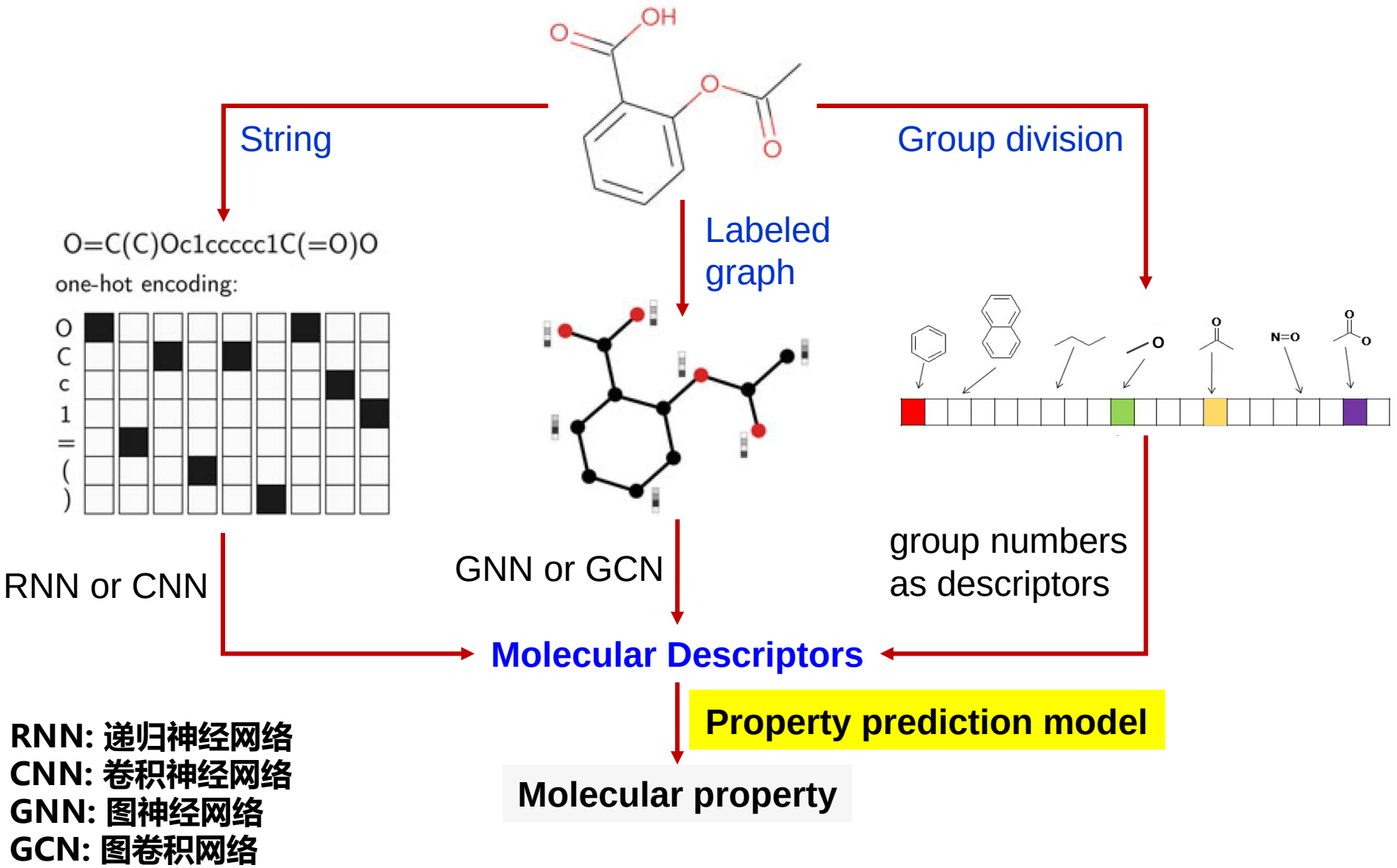
$-208.442y - 155.60x + 11717.797 + 1.002y^2 + x^2 = 0$



Workflow for AI-aided material design



Structure → Descriptor



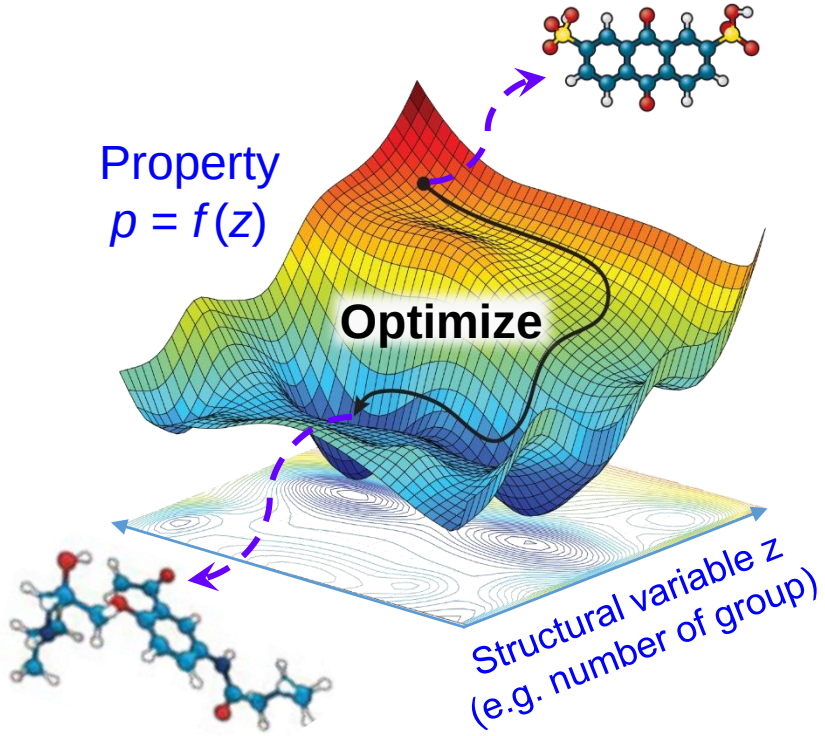
RNN: 递归神经网络
CNN: 卷积神经网络
GNN: 图神经网络
GCN: 图卷积网络

Optimization-Based Molecular Design

Computer Aided Molecular Design (CAMD)

Molecular structure (***Variable z***) → Property model $f(z)$ → Molecular property p (***Obj. function***)

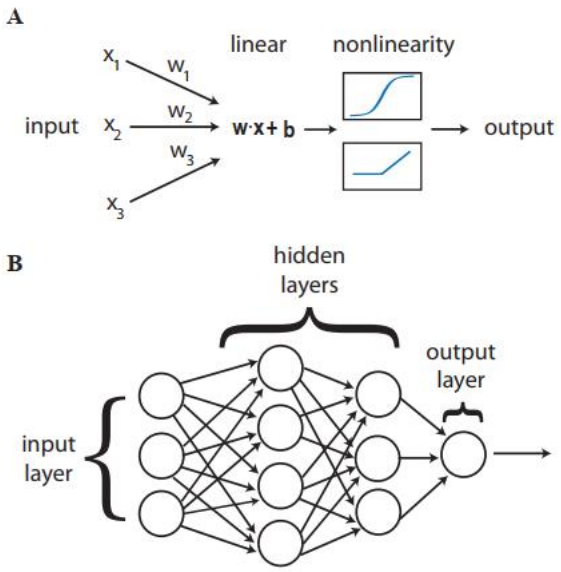
Optimization-based Molecular Design



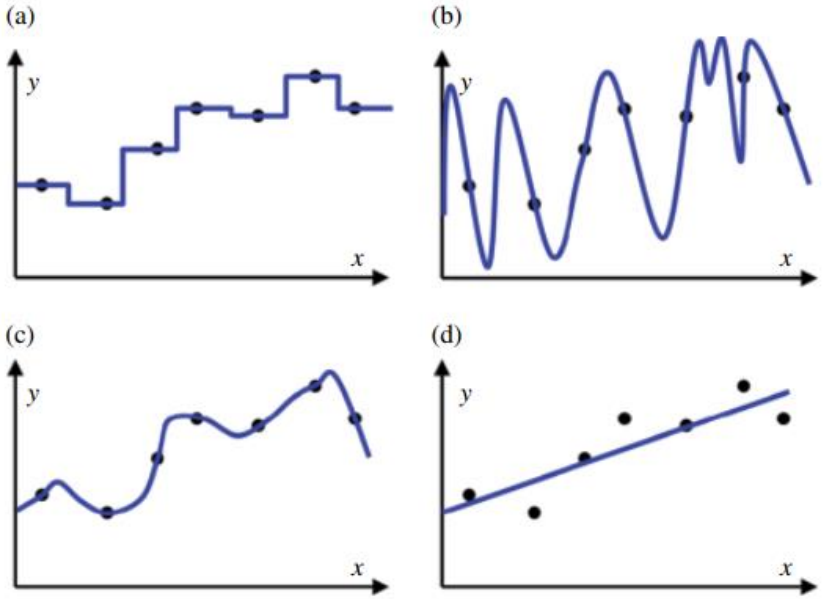
Variable *z*: e.g. **number of group in the molecule**
Objective function: e.g. minimal toxicity
Constraints: structural feasibility rules, etc.

AI Models for AMAT: Regression Model

A basic Neural Network

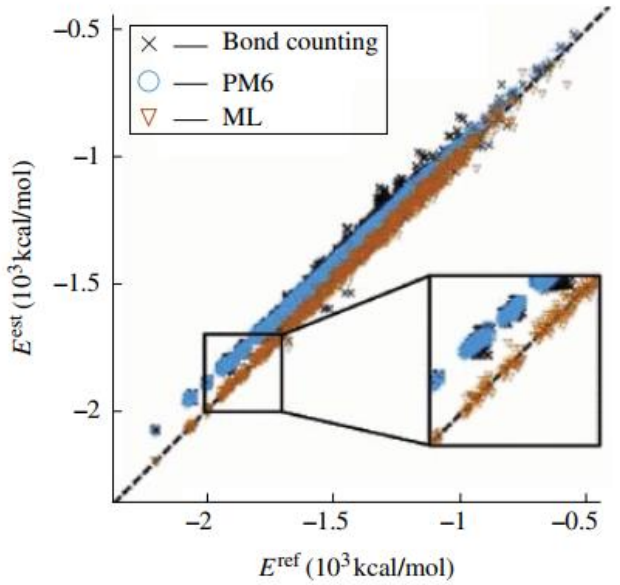


Regression task



Applications in AMAT

- Predicting structure-property relationships
- Develop model Hamiltonians
- Predict crystal structures
- …



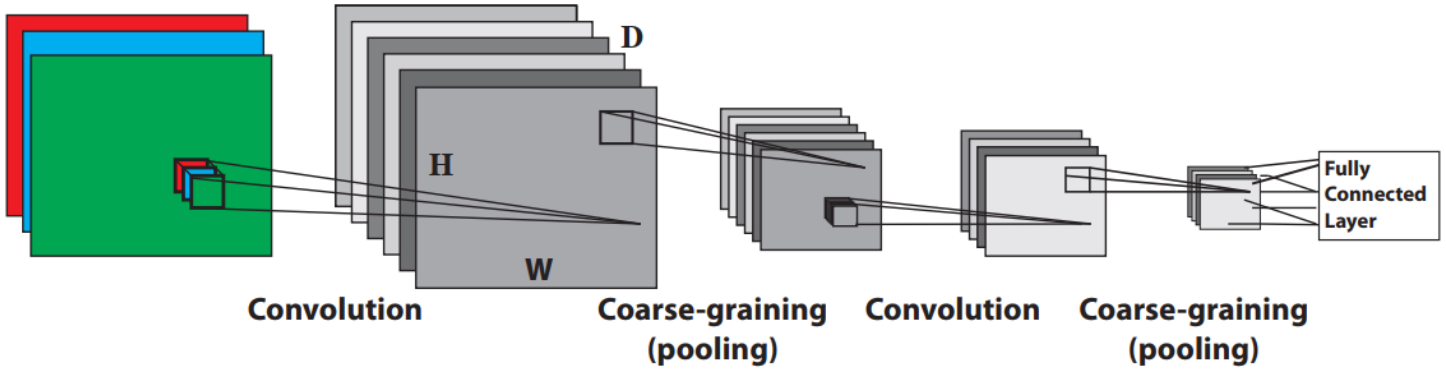
AI-aided first principal computation

- **Support Vector Machines**
- **Kernel Ridge Regression**
- **Neural Networks**
- **Decision Tree**
- **Genetic Programming**

AI Models for AMAT: Classification Model

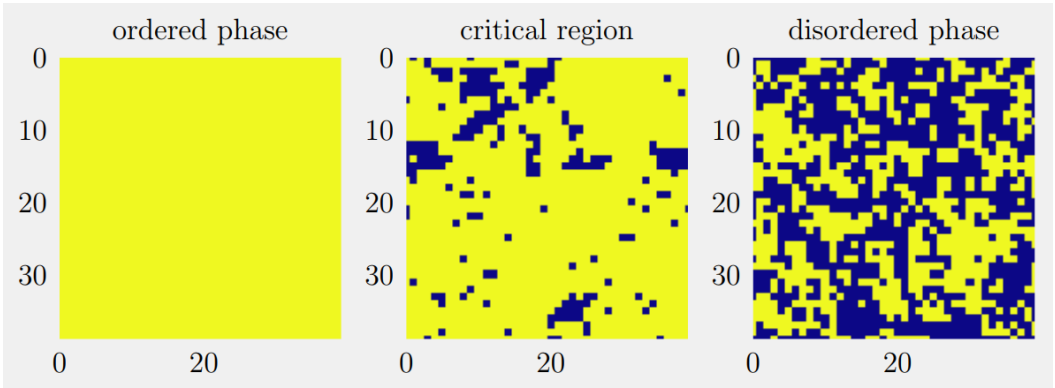
A Convolutional Neural Network

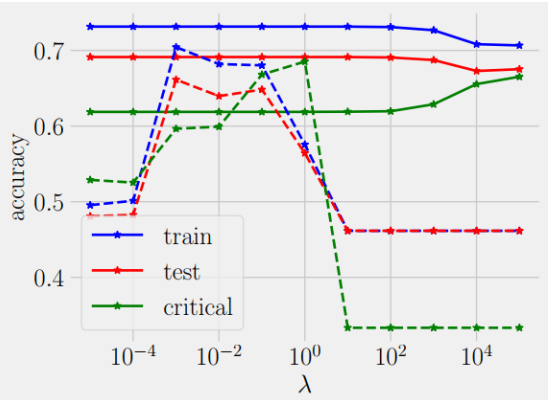
- Convolutional Neural Networks
- Support Vector Machine
- …



Applications in AMAT

- Identifying the phase of materials
- Classify crystal structures
- Experimental image classification
- …



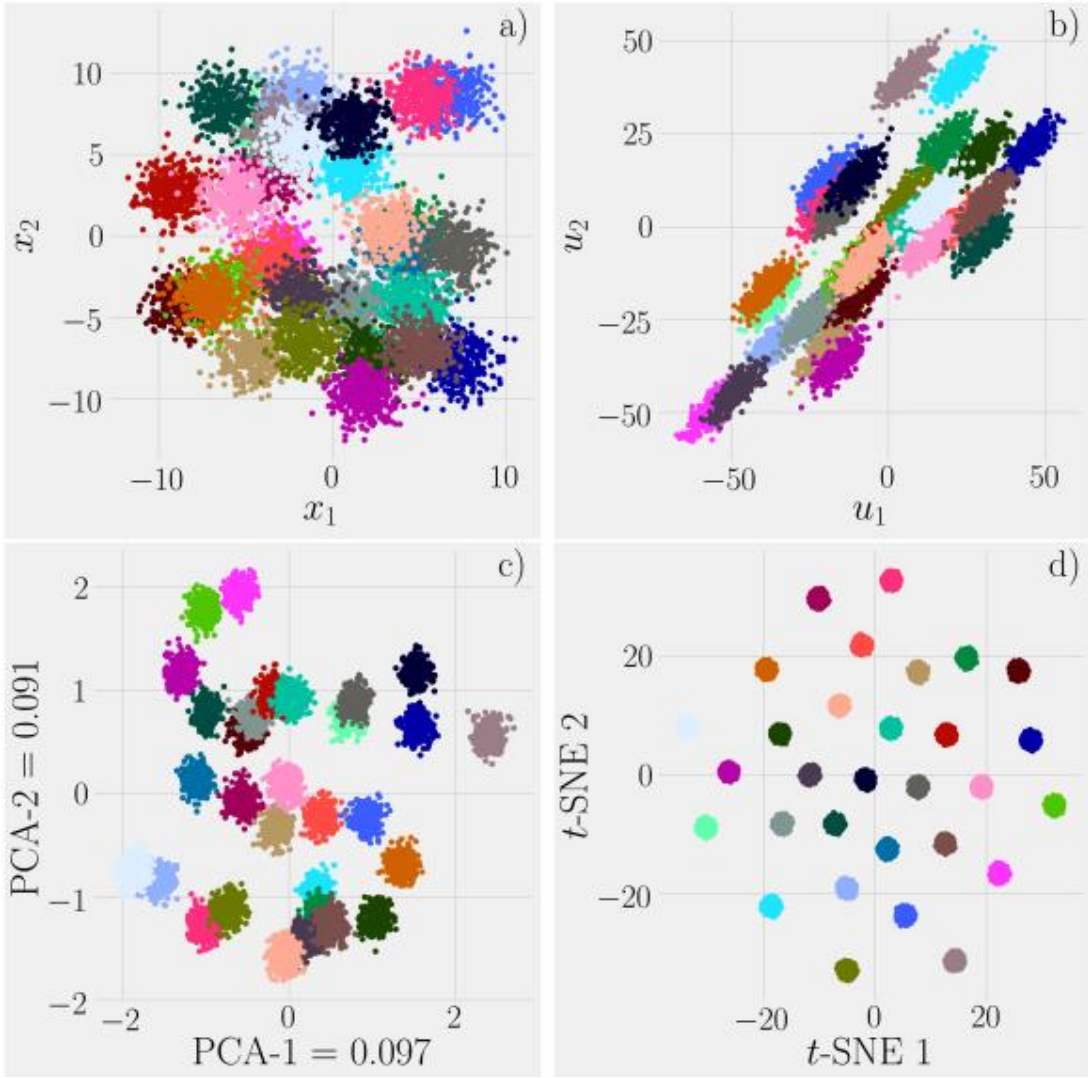


AI-aided phase identification for the 2D Ising Model

AI Models for AMAT: Clustering Model

Clustering for unlabeled data

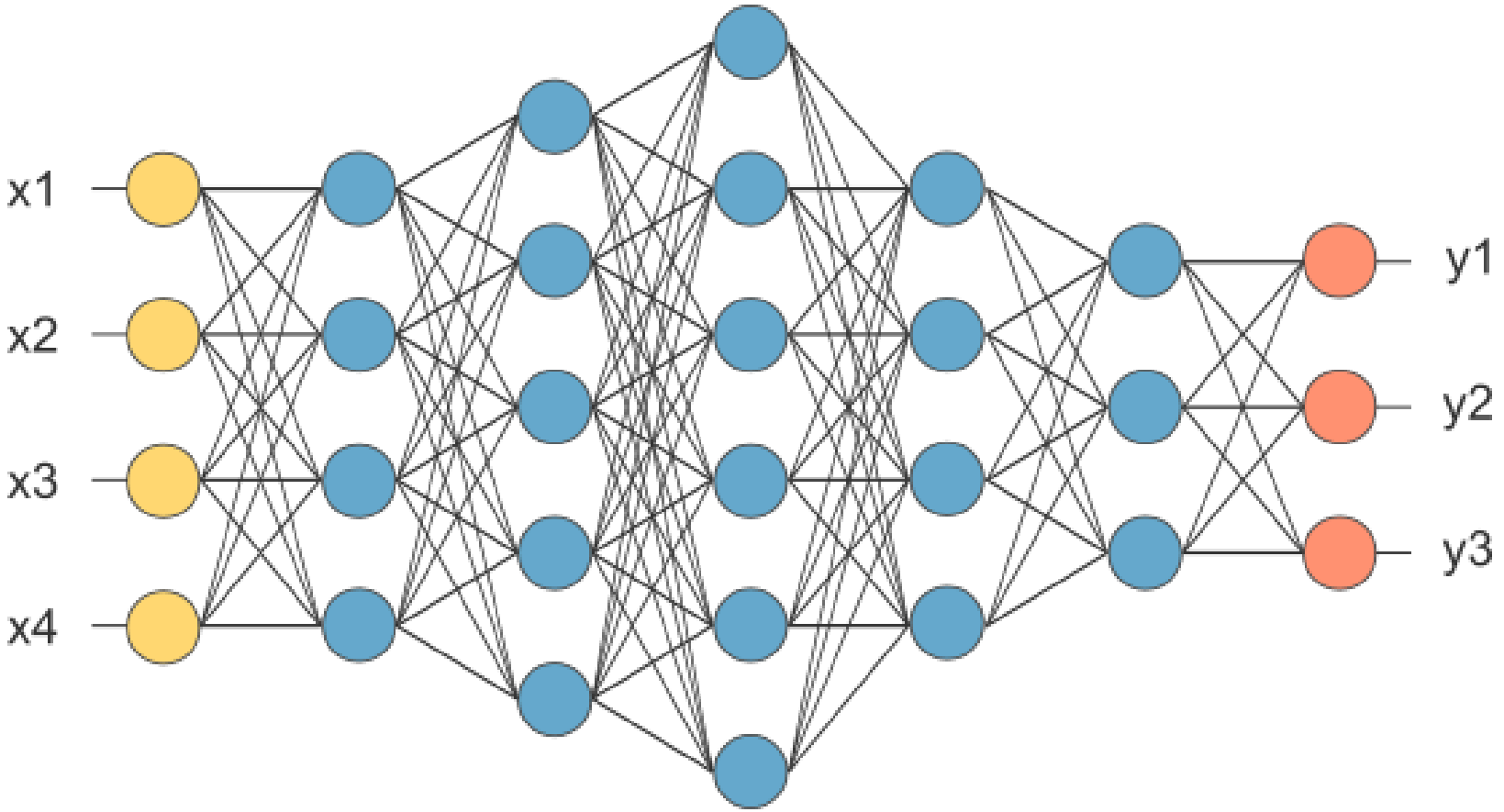
- **K-mean**
- **Agglomerative method**
- **Density clustering**
- **Principle Component Analysis**
- **…**



Applications in AMAT

- Analyze micrographs
- Analyze composition spreads from combinatorial experiments
- Noise reduction in datasets
- …

A Quick Review on Feedforward Neural Networks



Feedforward Neural Nets (FNN)

$\mathcal{N} : \mathbb{R}^{d_{in}} \to \mathbb{R}^{d_{out}}$ with L layers and N^l neurons in the layer l

$$\mathbf{x}^l = \sigma(\mathbf{h}^l) \qquad \mathbf{h}^l = \mathbf{W}^l \mathbf{x}^{l-1} + \mathbf{b}^l \quad \text{for } l = 1, \ldots, L-1$$

$$\mathbf{y} \equiv \mathbf{x}^L = \mathbf{h}^L = \mathbf{W}^L \mathbf{x}^{L-1} + \mathbf{b}^L$$

- $\mathbf{x}^0 \in \mathbb{R}^{d_{in}}$: input
- $\mathbf{W}^l$: weight matrix $(N^l \times N^{l-1})$ in the layer l
- $\mathbf{b}^l \in \mathbb{R}^{N^l}$: biases in the layer l
- σ: $\mathbb{R} \to \mathbb{R}$, component-wise activation function (nonlinear)
- $\mathbf{x}^l \in \mathbb{R}^{N^l}$: neural activity in the layer l

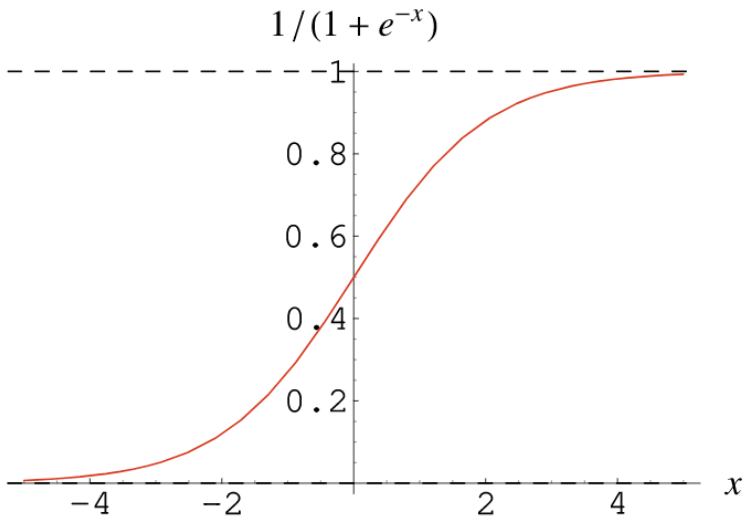
$$\mathbf{y} \equiv \mathbf{x}^L = \mathbf{W}^L \sigma(\mathbf{W}^{L-1} \sigma(\ldots \sigma(\mathbf{W}^2 \sigma(\mathbf{W}^1 \mathbf{x}^0 + \mathbf{b}^1) + \mathbf{b}^2)) + \mathbf{b}^{L-1}) + \mathbf{b}^L$$

What activations CAN we use IN THEORY?

- **Any continuous, slowly increasing, non-linear (and non-polynomial) function can be used as an activation.**

- **The Universal Approximation Theorem: Every measurable function f can be approximated arbitrarily well by a single-hidden-layer FNN with sigmoidal function as activation function.**

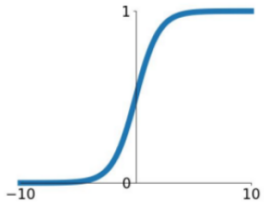
Sigmoid function:



What activations SHOULD we use IN REALITY?[3]

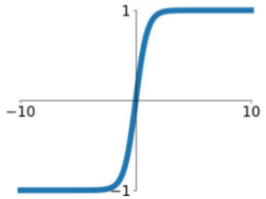
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



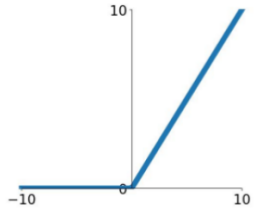
tanh

$\tanh(x)$



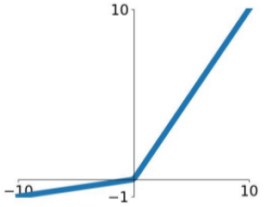
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

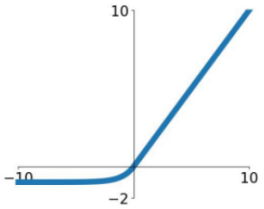


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Suggestions:

- Never use Sigmoid
- For very deep NN (>10 layers), try ReLU-based activations first

[3] LeCun et al., Neural Netw, 1998; Glorot & Bengio, AISTATS, 2010; Glorot et al., AISTATS, 2011.

What is TensorFlow?

- Open sourced by Google in November 2015
- Library for **numerical computation**
- **NOT** provide out-of-the-box machine learning solutions
- Tensor: ~~geometric objects that describe linear relations between geometric vectors, scalars, and other tensors~~
  n-dimensional matrix

Other tools includes PyTorch, Keras, FLUX, Caffe, etc...

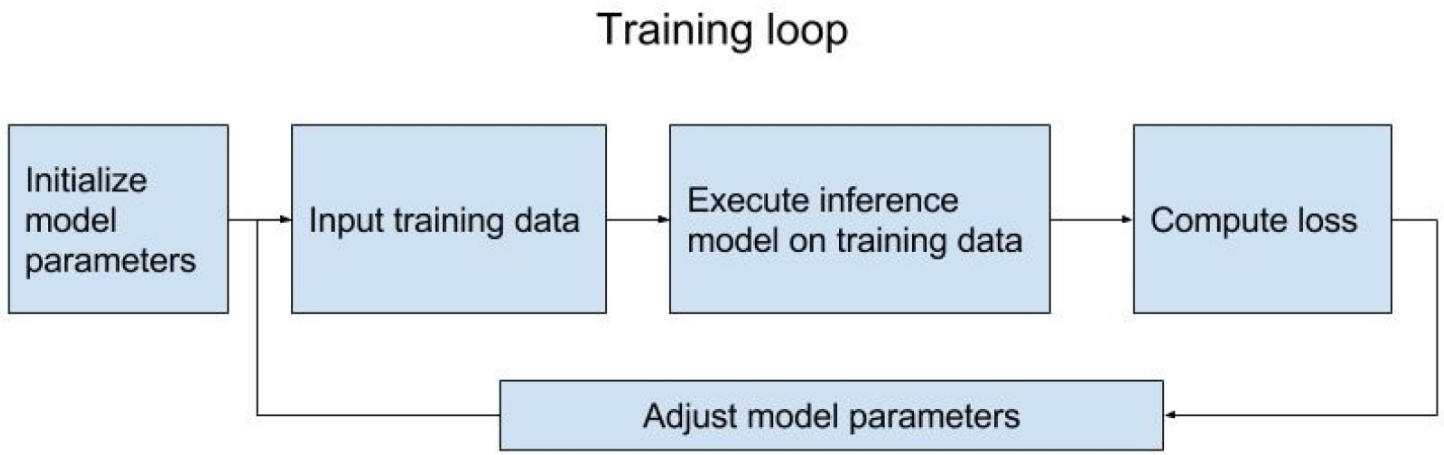
TensorFlow

```
import tensorflow as tf
```

- data type: `tf.float32`, `tf.float64`
- Inputs: `tf.placeholder`
- Variables to be optimized: `tf.Variable`
- Math operations
  - Multiplies matrix a by matrix b: `tf.matmul(a, b)`
  - `tf.nn.tanh`, `tf.nn.relu`
  - mean: `tf.reduce_mean`
- Session

```
# Build a NN
...
# Launch, init, run
sess = tf.Session()
sess.run(tf.global_variables_initializer())
```

What is next?



After building a neural network, train, i.e., optimize **W** and **b** using data.

- Define a loss to be minimized
- Initialize the net
- Choose an optimizer

Loss

Traing data set: $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$

- Mean absolute error (MAE) or L1

$$\frac{1}{n}\sum_{i=1}^{n}|\mathcal{N}(x_i) - y_i|$$

- Mean squared error (MSE) or L2

$$\frac{1}{n}\sum_{i=1}^{n}(\mathcal{N}(x_i) - y_i)^2$$

```
tf.reduce_mean((y_pred - y_true)**2)
```

- ...

How to initialize the weights?

Weights are randomly sampled (zero mean). $\mathbf{b} = 0$.

Initializer	Var[w]	Activation
Glorot uniform/normal[4]	$2/(fan_{in} + fan_{out})$	tanh
He normal[5]	$2/fan_{in}$	ReLU
LeCun normal[6]	$1/fan_{in}$	SeLU
Orthogonal[7]	-	all
LSUV[8]	-	all

- fan_{in}: the number of input units of the layer
- fan_{out}: the number of output units of the layer

[4] Glorot & Bengio, AISTATS, 2010.
[5] He et al., ICCV, 2015.
[6] LeCun et al., Neural Netw, 1998; Klambauer et al., NIPS, 2017.
[7] Saxe et al., ICLR, 2014.
[8] Mishkin & Matas, ICLR, 2015.

How to optimize?

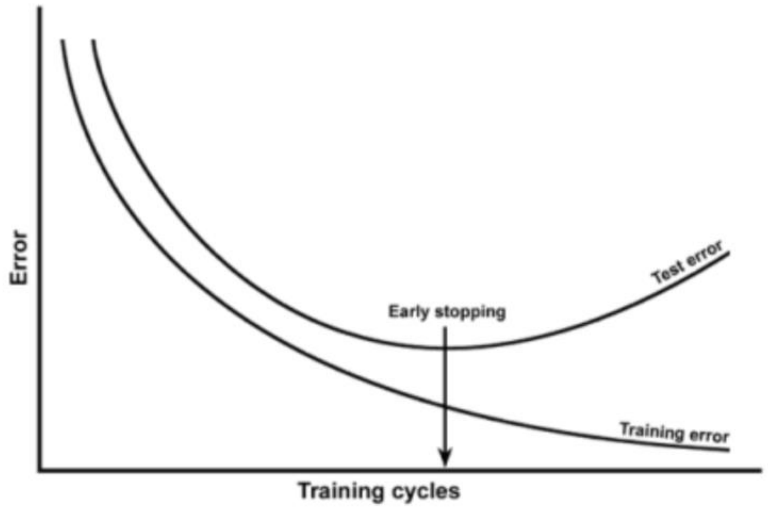
Optimizers

- SGD: $w_{t+1} = w_t - \eta \nabla_w loss(w)$
- SGDNesterov[9]: momentum
- AdaGrad[10]: adaptive per-parameter learning rates
- AdaDelta[11], RMSProp[12]: extensions of AdaGrad
- Adam[13]: adaptive & momentum
- ...

```
learning_rate = ...
loss = ...
opt = tf.train.AdamOptimizer(learning_rate)
train = opt.minimize(loss)
```

[9] Sutskever et al., ICML, 2013.
[10] Duchi et al., JMLR, 2011.
[11] Zeiler, arXiv, 2012.
[12] Hinton, csc321, 2014.
[13] Kingma & Ba, ICLR, 2015.

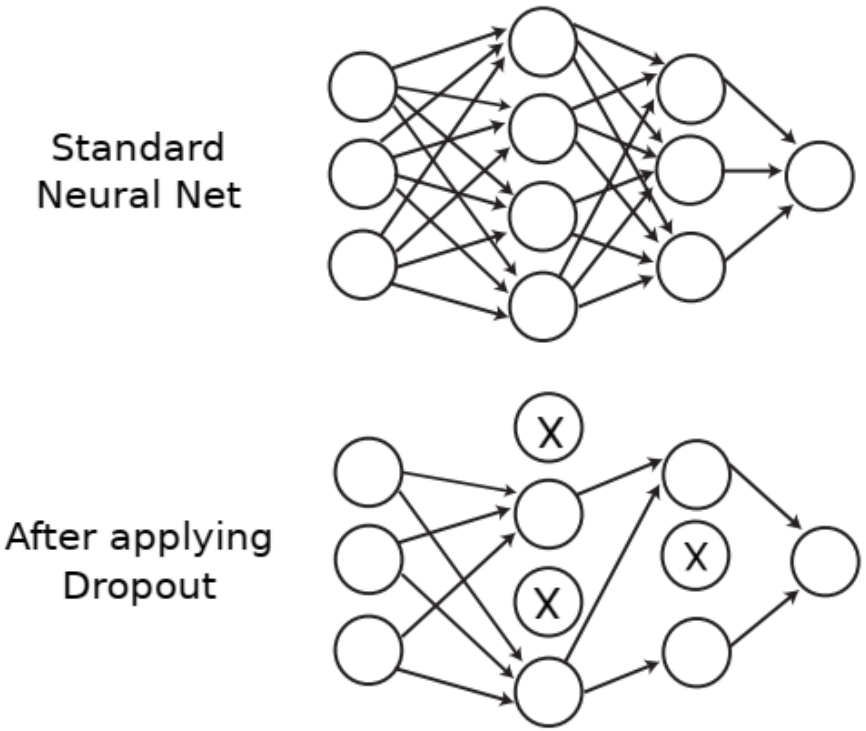
What else?



- Overfitting
  - Early stopping: Beautiful FREE LUNCH[14]
  - L1/L2 regularization: $\lambda \sum_w |w|^2$
  - Dropout[15]
- Normalization
- ...

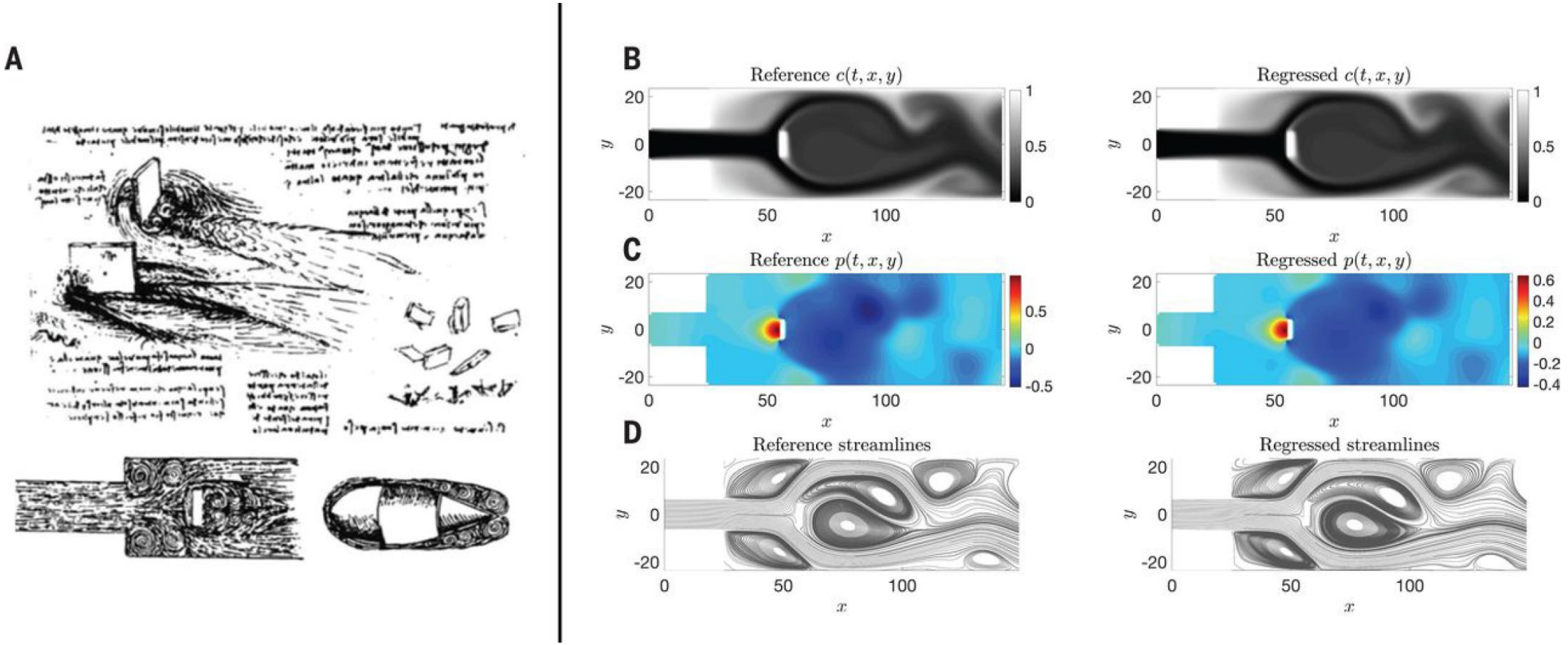
[14] Hinton, NIPS, 2015.
[15] Srivastava et al., JMLR, 2014.

Other NNs:
GNN, CNN, RNN, GAN…



Domain-knowledge guided AI: the physics informed approach

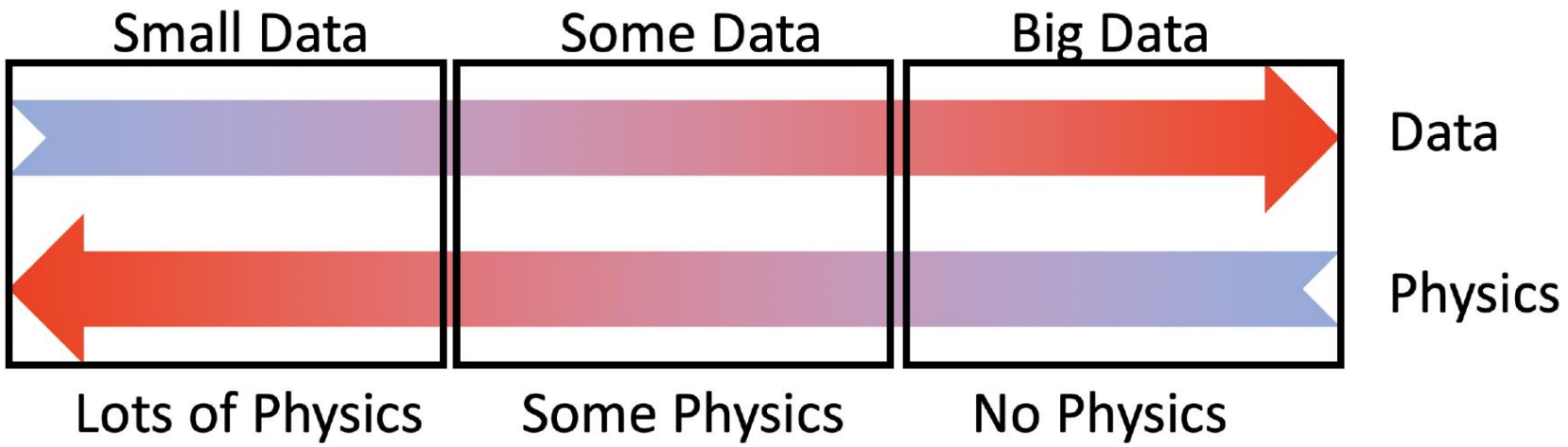
Infer pressure p and velocity fields (u, v) from concentration c



Raissi, Yazdani, & Karniadakis, *Science*, 2020

Data & Physics relationship

Three scenarios:



1. Lots of physics – Forward problems (finite element/finite difference methods)
2. Some physics – Inverse problems (Physics-informed neutral network)
3. No physics – system identification/discovery (Operator learning)

Karniadakis*, Kevrekidis*, **Lu***, et al., *Nature Rev Phys*, 2021

Physics-Informed Deep Learning
Learning from Small Data

Review Article | Published: 24 May 2021

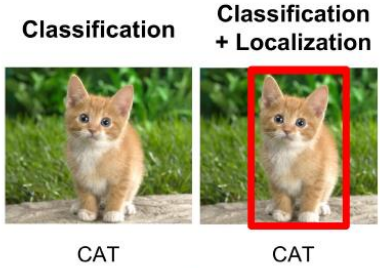
Physics-informed machine learning

George Em Karniadakis, Ioannis G. Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang & Liu Yang

Nature Reviews Physics **3**, 422–440 (2021) | Cite this article

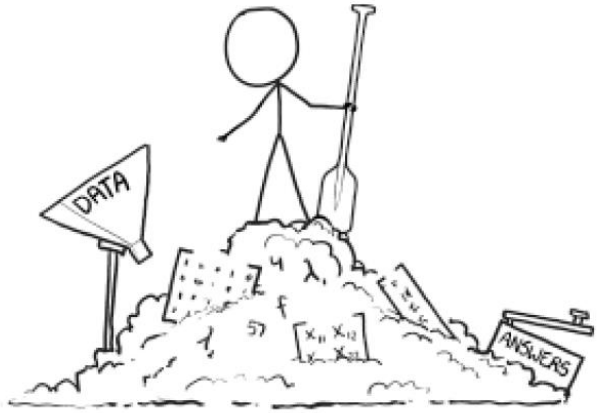
Data & Physics relationship

- Computer vision



- Machine translation

- Black-box & Trial-and-error
- Big data



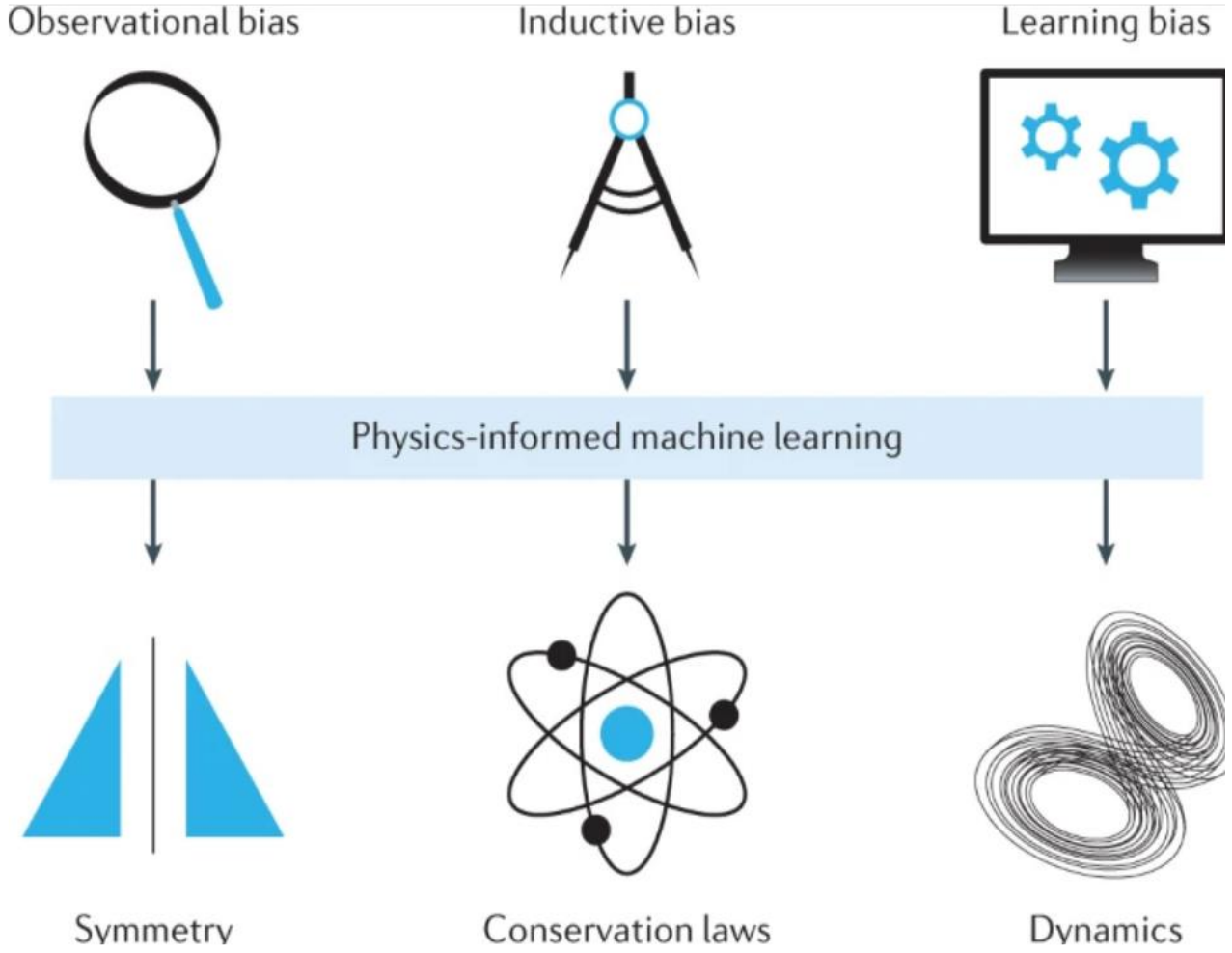
Scientific Machine Learning: Learning from Small Data

- *Dinky, Dirty, Dynamic, Deceptive* Data
- Scientific domain knowledge
  e.g., physical principles, constraints, symmetries, computational simulations
- Accurate, Robust, Reliable, Interpretable, Explainable

Karniadakis*, Kevrekidis*, **Lu***, et al., *Nature Rev Phys*, 2021

How to embed physics in Machine Learning?

Principles of physics-informed learning:



Karniadakis*, Kevrekidis*, **Lu***, et al., *Nature Rev Phys*, 2021

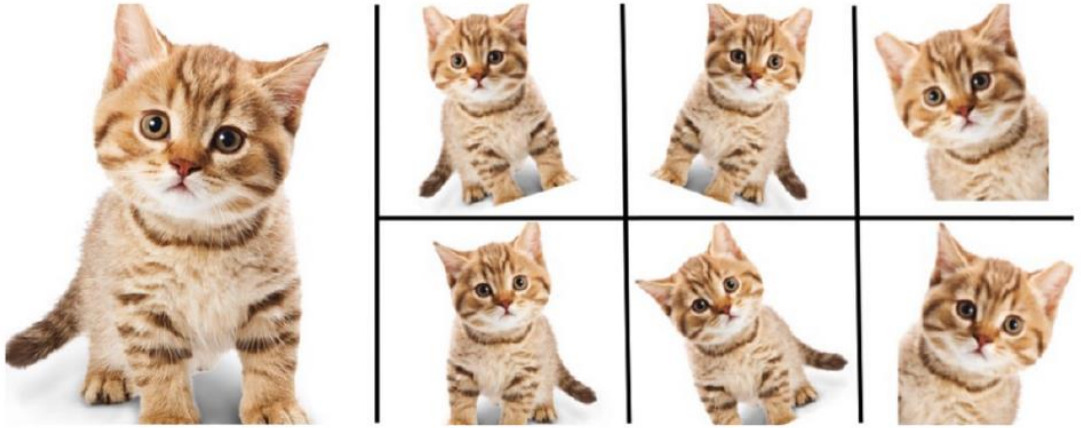
Observation bias

Directly from **data**

- e.g., $f \mapsto u$ is symmetry invariant

$$\mathrm{sym}(f) \mapsto u$$

- Data augmentation



- Brute-force: Let the machine learn.
- May need a large dataset

Inductive bias (embed domain knowledge)

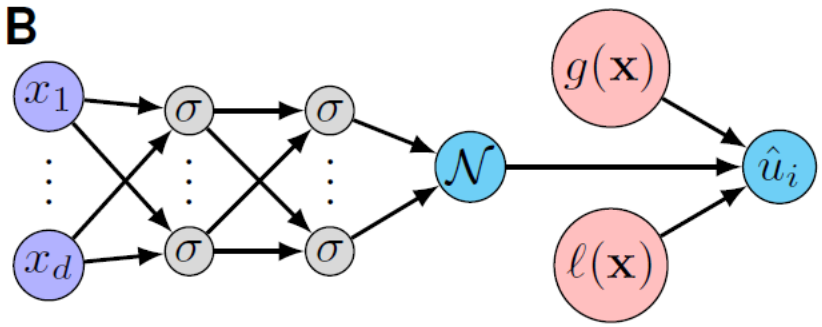
ML model architecture: Embed any prior knowledge

- CNN: translation invariance
- Partially fixed-value network (Dirichlet BC):
  $u(0) = 0, u(1) = 1$: $g(x) = x$, $\ell(x) = x(1-x)$

$$\mathcal{N}(x) = g(x), \quad x \in \Gamma_D \subset \Omega$$

$$\hat{u}(x) = g(x) + \ell(x)\mathcal{N}(x)$$

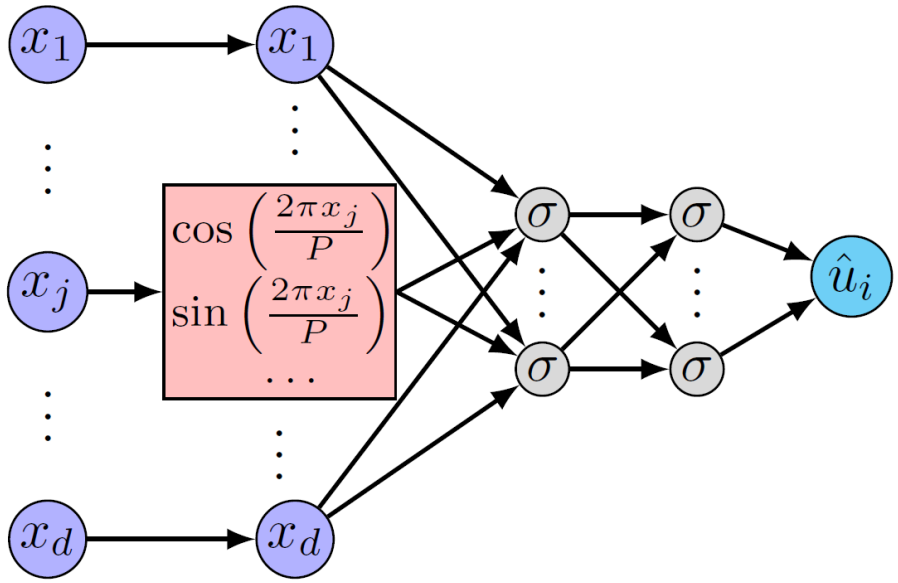
$$\begin{cases} \ell(\mathbf{x}) = 0, & \mathbf{x} \in \Gamma_D, \\ \ell(\mathbf{x}) > 0, & \mathbf{x} \in \Omega - \Gamma_D. \end{cases}$$



Inductive bias (embed domain knowledge)

Periodic network (Periodic BC) via Fourier series:

$$\{1, \cos(\frac{2\pi x_j}{P}), \sin(\frac{2\pi x_j}{P}), \cos(\frac{4\pi x_j}{P}), \sin(\frac{4\pi x_j}{P}), \cdots\}$$



Can use as few as two terms $\{\cos(\frac{2\pi x_j}{P}), \sin(\frac{2\pi x_j}{P})\}$

Lu† et al., *SIAM J Sci Comput*, 2021

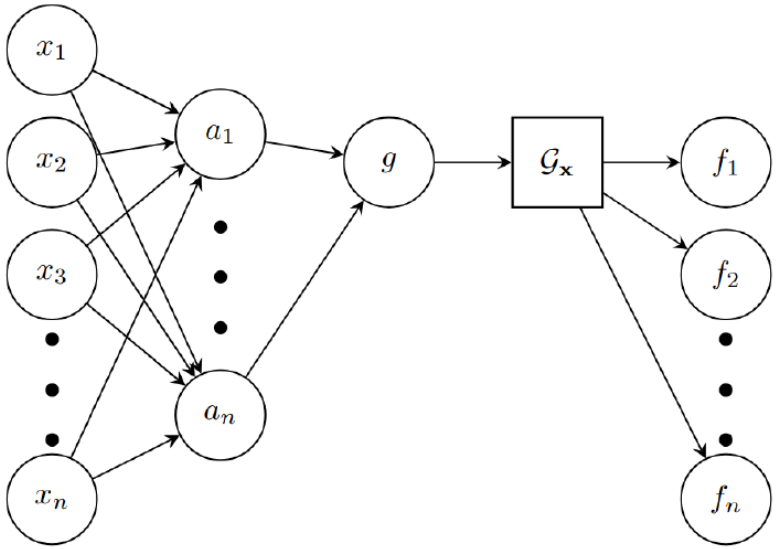
Inductive bias (embed domain knowledge)

- Linearly constrained neural networks, e.g., divergence-free

$$\nabla \cdot \mathbf{f} = \frac{\partial f_1}{\partial x} + \frac{\partial f_2}{\partial y} = 0$$

Divergence of the curl is 0: $\nabla \cdot (\nabla \times g) \equiv 0$, so choose

$$\mathbf{f} = \nabla \times g$$



Hendriks et al., arXiv:2002.01600

Inductive bias (embed domain knowledge)

Free lunch: The network satisfies the physical constraints **automatically** and **exactly**.

- Do it if possible
- Designed case by case
- Currently limited to relatively simple and well-defined physics

Learning bias

Idea:

- loss functions
- modulate the training phase

e.g., Physics-informed neural network (PINN)

Physics-informed neutral network (PINN): Forward problem

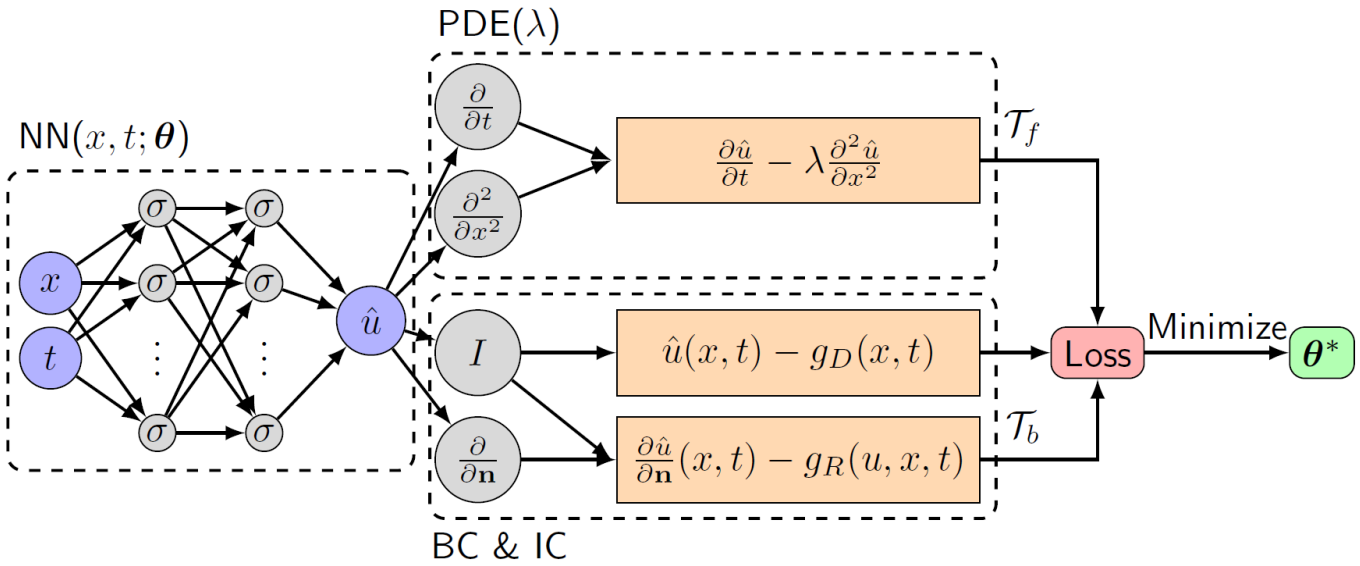
Diffusion equation

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

BCs: $u(x,t) = g_D(x,t)$ on $\Gamma_D \subset \partial\Omega$

$\frac{\partial u}{\partial \mathbf{n}}(x,t) = g_R(u,x,t)$ on $\Gamma_R \subset \partial\Omega$

IC is treated as a Dirichlet BC on the spatio-temporal domain



PDE embedded into the loss via automatic differentiation (AD)

Lu et al., *SIAM Rev*, 2021

PINN

Accepted Manuscript

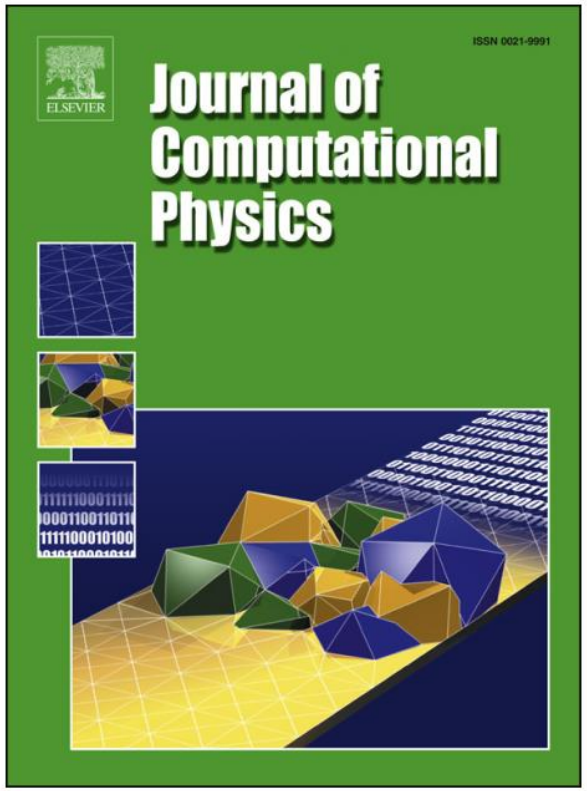
Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations

M. Raissi, P. Perdikaris, G.E. Karniadakis

PII: S0021-9991(18)30712-5
DOI: https://doi.org/10.1016/j.jcp.2018.10.045
Reference: YJCPH 8347

To appear in: *Journal of Computational Physics*

Received date: 13 June 2018
Revised date: 26 October 2018
Accepted date: 28 October 2018



Physics-informed neural networks: A **deep** learning framework for solving forward and inverse problems involving nonlinear partial differential equations

[PDF] sciencedirect.com

M Raissi, P Perdikaris, GE Karniadakis - Journal of Computational **physics**, 2019 - Elsevier

… We introduce **physics-informed neural networks** – **neural networks** that are trained to solve supervised learning tasks while respecting any given laws of **physics** described by general …

☆ Save 99 Cite Cited by 6625 Related articles All 7 versions

3.1. Continuous time models

We define $f(t,x)$ to be given by the left-hand-side of equation (2); i.e.,

$$f := u_t + \mathcal{N}[u], \tag{3}$$

and proceed by approximating $u(t,x)$ by a deep neural network. This assumption along with equation (3) result in a *physics-informed neural network* $f(t,x)$. This network can be derived by applying the chain rule for differentiating compositions of functions using automatic differentiation [12], and has the same parameters as the network representing $u(t,x)$, albeit with different activation functions due to the action of the differential operator $\mathcal{N}$. The shared parameters between the neural networks $u(t,x)$ and $f(t,x)$ can be learned by minimizing the mean squared error loss

$$MSE = MSE_u + MSE_f, \tag{4}$$

where

$$MSE_u = \frac{1}{N_u}\sum_{i=1}^{N_u} |u(t_u^i, x_u^i) - u^i|^2,$$

and

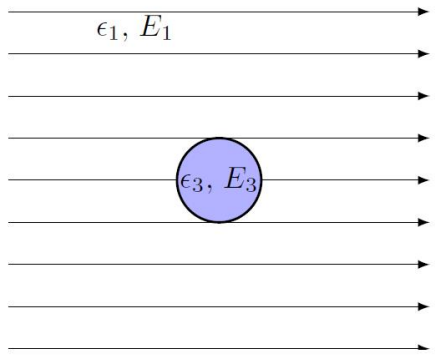
$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2.$$

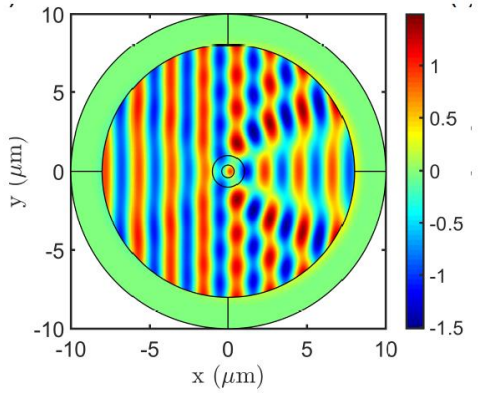
Physics-informed neutral network (PINN): Inverse Problem

Example: invisible cloaking

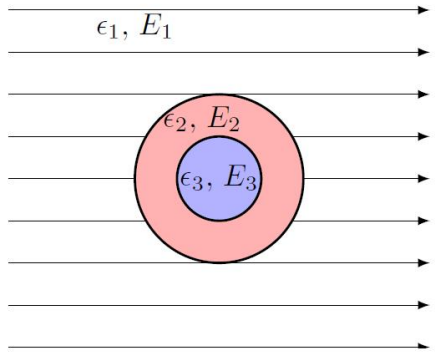
Challenge: small data + incomplete physical law

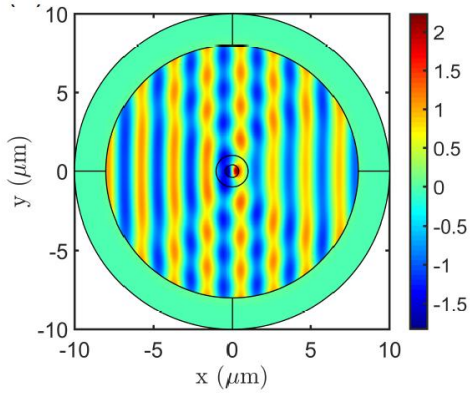
Electric field E without coating





Electric field E with coating



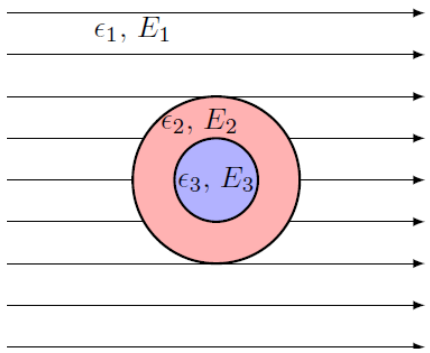


Chen, **Lu**, et al., *Opt Express*, 2020

Physics-informed neutral network (PINN): Inverse Problem

Example: invisible cloaking

Goal: Given ϵ_1 and ϵ_3, find $\epsilon_2(x, y)$ s.t. $E_1 \approx E_{1,target}$



Helmholtz equation $(k_0 = \frac{2\pi}{\lambda_0})$

$$\nabla^2 E_i + \epsilon_i k_0^2 E_i = 0, \qquad i = 1, 2, 3$$

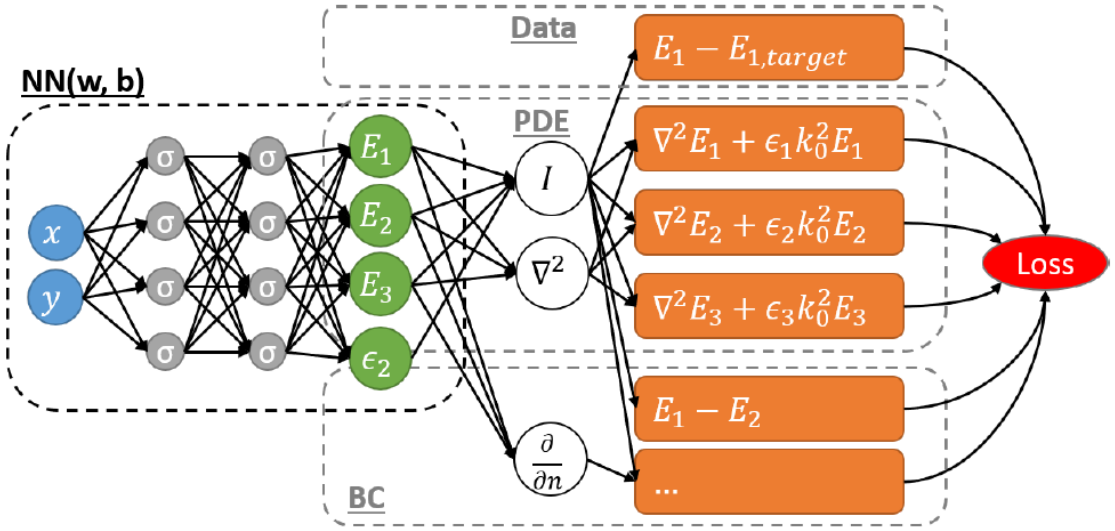
Boundary conditions:

- Outer circle: $E_1 = E_2,\ \frac{1}{\mu_1}\frac{\partial E_1}{\partial \mathbf{n}} = \frac{1}{\mu_2}\frac{\partial E_2}{\partial \mathbf{n}}$
- Inner circle: $E_2 = E_3,\ \frac{1}{\mu_2}\frac{\partial E_2}{\partial \mathbf{n}} = \frac{1}{\mu_3}\frac{\partial E_3}{\partial \mathbf{n}}$

Chen, **Lu**, et al., *Opt Express*, 2020

Physics-informed neutral network (PINN): Inverse Problem

Embed both PDE and data into the loss function:



- mesh-free & particle-free
- inverse problems: seamlessly integrate data and physics
- black-box or noisy IC/BC/forcing terms (Pang*, **Lu***, et al., *SIAM J Sci Comput*, 2019)
- a unified framework: PDE, integro-differential equations (**Lu** et al., *SIAM Rev*, 2021), fractional PDE (Pang*, **Lu***, et al., *SIAM J Sci Comput*, 2019), stochastic PDE (Zhang, **Lu**, et al., *J Comput Phys*, 2019)

Physics-informed neutral network (PINN): Inverse Problem

Embed both PDE and data into the loss function:

$$f\left(\mathbf{x}; \frac{\partial u}{\partial x_1}, \ldots, \frac{\partial u}{\partial x_d}; \frac{\partial^2 u}{\partial x_1 \partial x_1}, \ldots, \frac{\partial^2 u}{\partial x_1 \partial x_d}; \ldots; \boldsymbol{\lambda}\right) = 0, \quad \mathbf{x} \in \Omega$$

- Initial/boundary conditions $\mathcal{B}(u, \mathbf{x}) = 0 \quad \text{on} \quad \partial\Omega$
- Extra information $\mathcal{I}(u, \mathbf{x}) = 0 \quad \text{for} \quad \mathbf{x} \in \mathcal{T}_i$

$$\min_{\boldsymbol{\theta}, \boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda}; \mathcal{T}) = w_f \mathcal{L}_f(\boldsymbol{\theta}, \boldsymbol{\lambda}; \mathcal{T}_f) + w_b \mathcal{L}_b(\boldsymbol{\theta}, \boldsymbol{\lambda}; \mathcal{T}_b) + w_i \mathcal{L}_i(\boldsymbol{\theta}, \boldsymbol{\lambda}; \mathcal{T}_i)$$

where

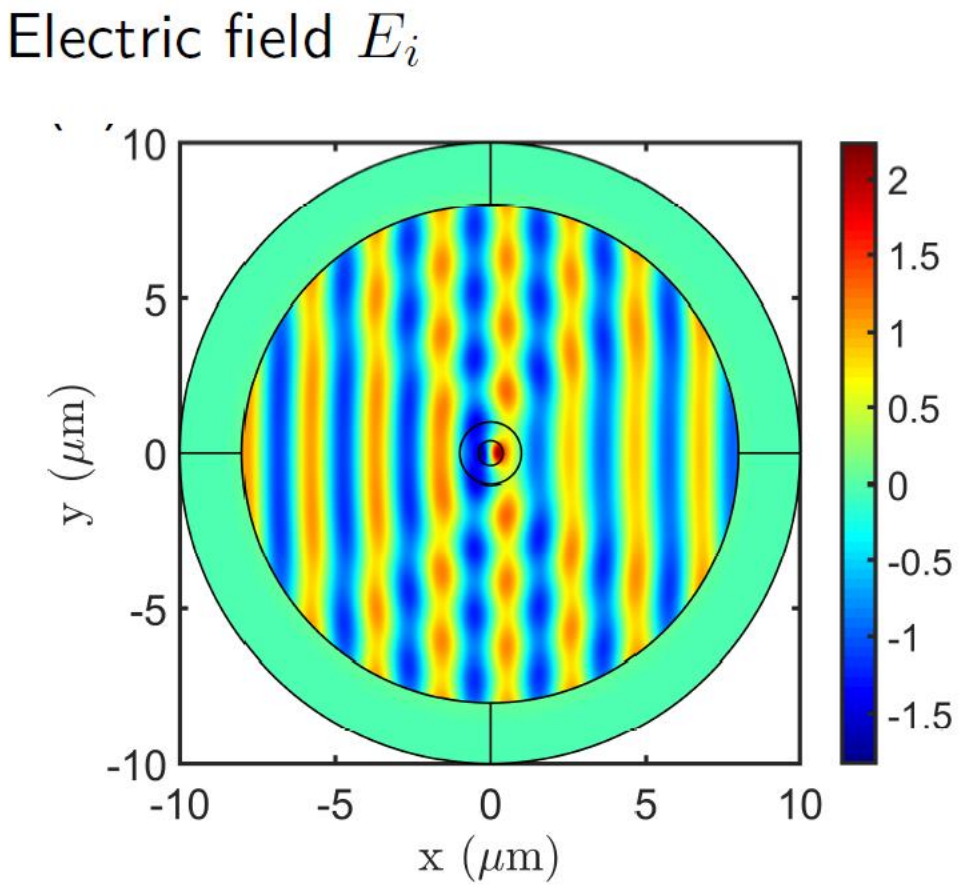
$$\mathcal{L}_f = \frac{1}{|\mathcal{T}_f|} \sum_{\mathbf{x} \in \mathcal{T}_f} \left\| f\left(\mathbf{x}; \frac{\partial \hat{u}}{\partial x_1}, \ldots; \frac{\partial^2 \hat{u}}{\partial x_1 \partial x_1}, \ldots; \boldsymbol{\lambda}\right) \right\|_2^2$$

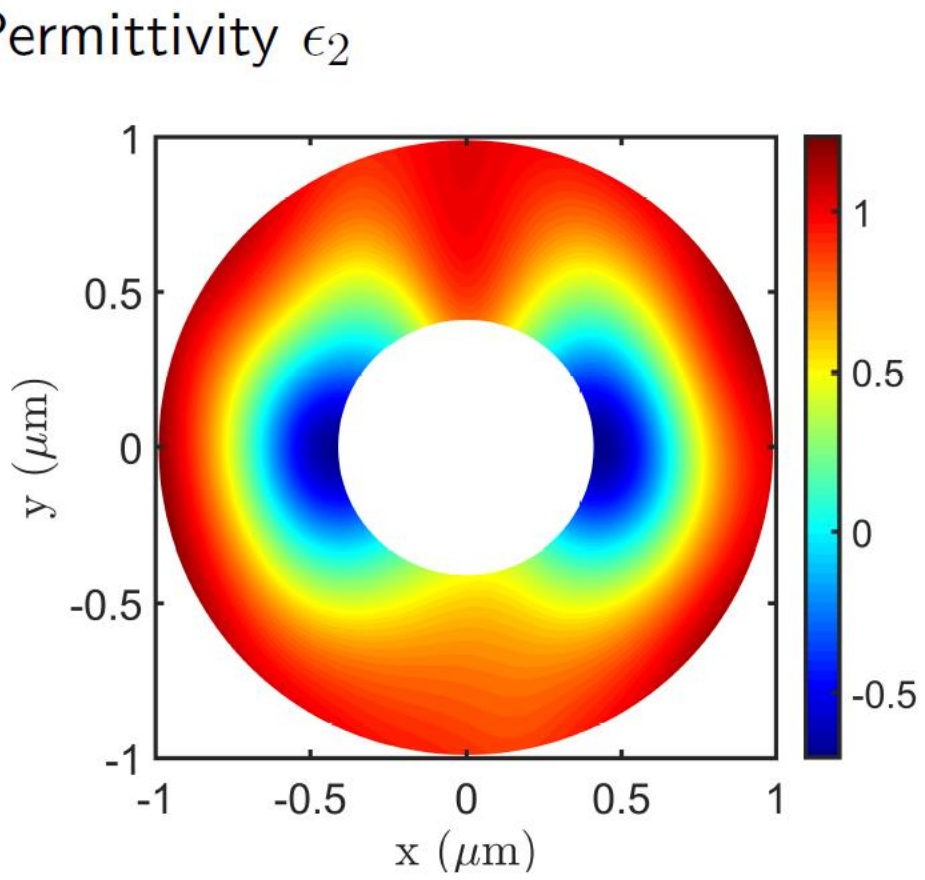
$$\mathcal{L}_b = \frac{1}{|\mathcal{T}_b|} \sum_{\mathbf{x} \in \mathcal{T}_b} \|\mathcal{B}(\hat{u}, \mathbf{x})\|_2^2$$

$$\mathcal{L}_i = \frac{1}{|\mathcal{T}_i|} \sum_{\mathbf{x} \in \mathcal{T}_i} \|\mathcal{I}(\hat{u}, \mathbf{x})\|_2^2$$

Physics-informed neutral network (PINN): Inverse Problem

Optimized design for Invisible cloaking by PINN:





Inverse Problem: application in biophysics

Ultradian model for the glucose-insulin interaction (Sturis et al., 1991)

Goal: Infer 21 unknown parameters

$$\frac{dI_p}{dt} = f_1(G) - E\left(\frac{I_p}{V_p} - \frac{I_i}{V_i}\right) - \frac{I_p}{t_p}$$

$$\frac{dI_i}{dt} = E\left(\frac{I_p}{V_p} - \frac{I_i}{V_i}\right) - \frac{I_i}{t_i}$$

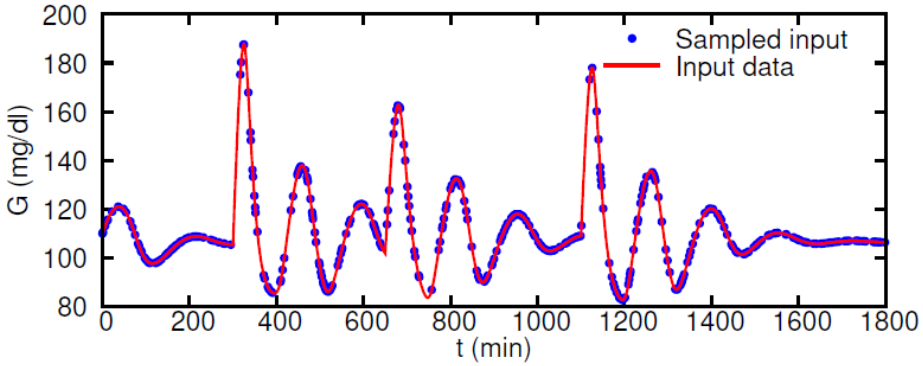
$$\frac{dG}{dt} = f_4(h_3) + I_G(t) - f_2(G) - f_3(I_i)G$$

$$\frac{dh_1}{dt} = \frac{1}{t_d}\left(I_p - h_1\right)$$

$$\frac{dh_2}{dt} = \frac{1}{t_d}\left(h_1 - h_2\right)$$

$$\frac{dh_3}{dt} = \frac{1}{t_d}\left(h_2 - h_3\right)$$

- I_p: plasma insulin concentration
- I_i: interstitial insulin concentration
- G: glucose concentration (Data)
- I_G: glucose from nutritional intake

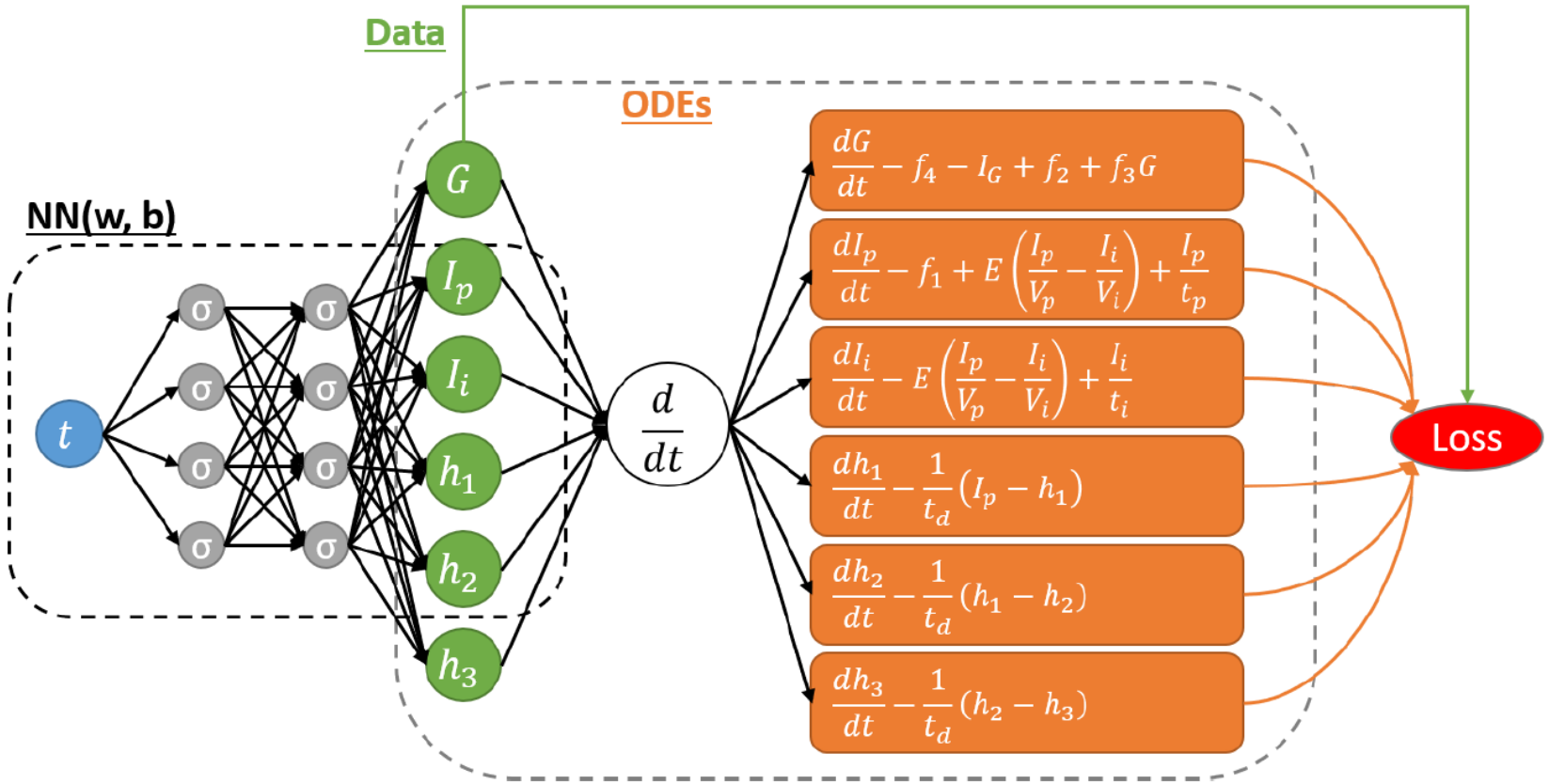


$$f_1(G) = \frac{R_m}{1 + \exp(\frac{-G}{V_g C_1} + a_1)}, f_2(G) = U_b\left(1 - \exp(\frac{-G}{C_2 V_g})\right), f_3(I_i) = \frac{1}{C_3 V_g}\left(U_0 + \frac{U_m}{1 + (\kappa I_i)^{-\beta}}\right)$$

$$f_4(h_3) = \frac{R_g}{1 + \exp(\alpha(\frac{h_3}{C_5 V_p} - 1))}, \kappa = \frac{1}{C_4}\left(\frac{1}{V_i} + \frac{1}{E t_i}\right), I_G(t) = \sum_{j=1}^{N} m_j k \exp(k(t_j - t))$$

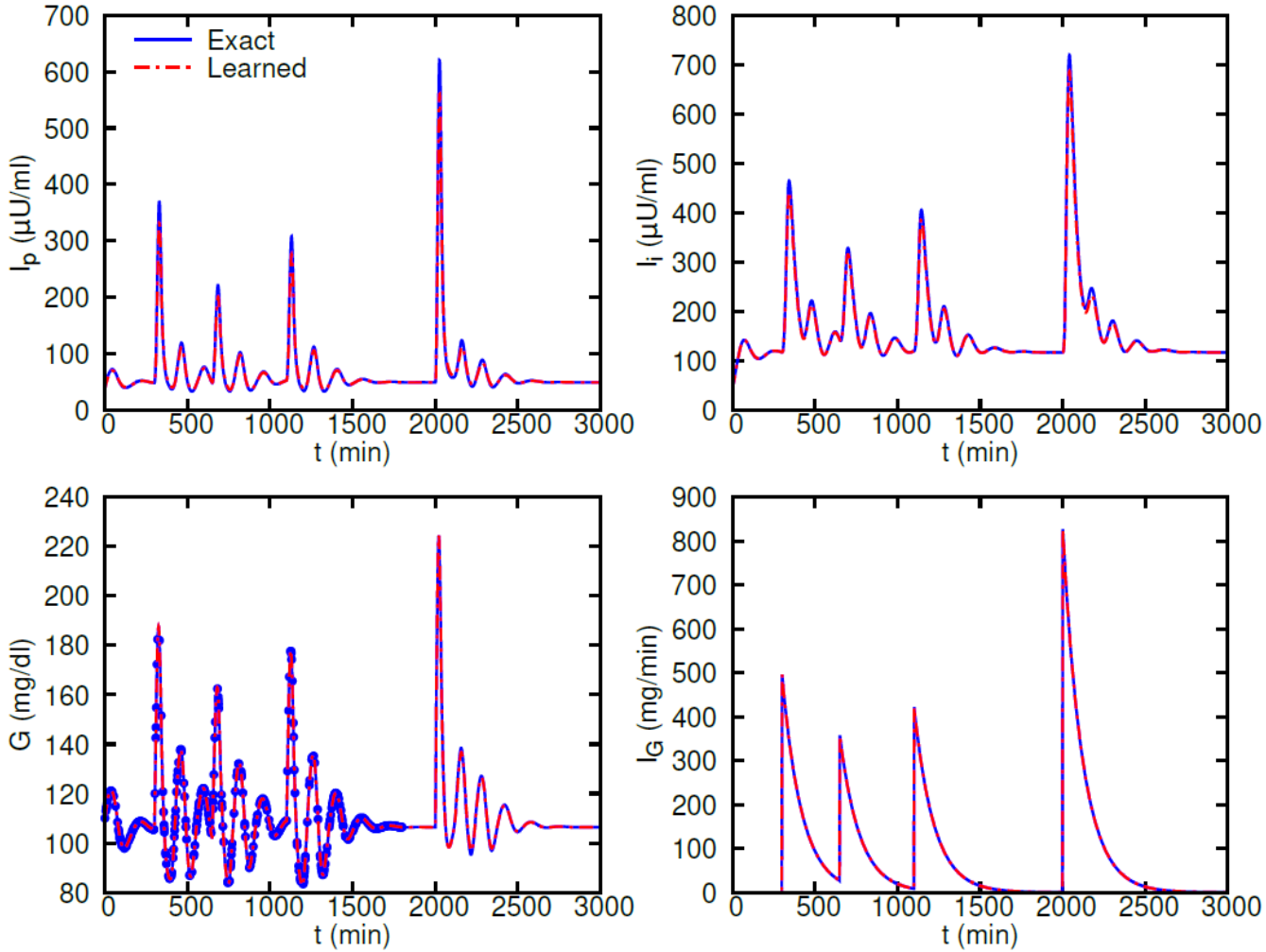
Inverse Problem: application in biophysics

Idea: Embed ODEs with unknown parameters into the loss via automatic differentiation (backpropagation)



- Seamlessly integrate *sparse* data and *incomplete* physics

Inferred dynamics and forecasting



Research Highlight in

Nature Computational Science

SYSTEMS BIOLOGY

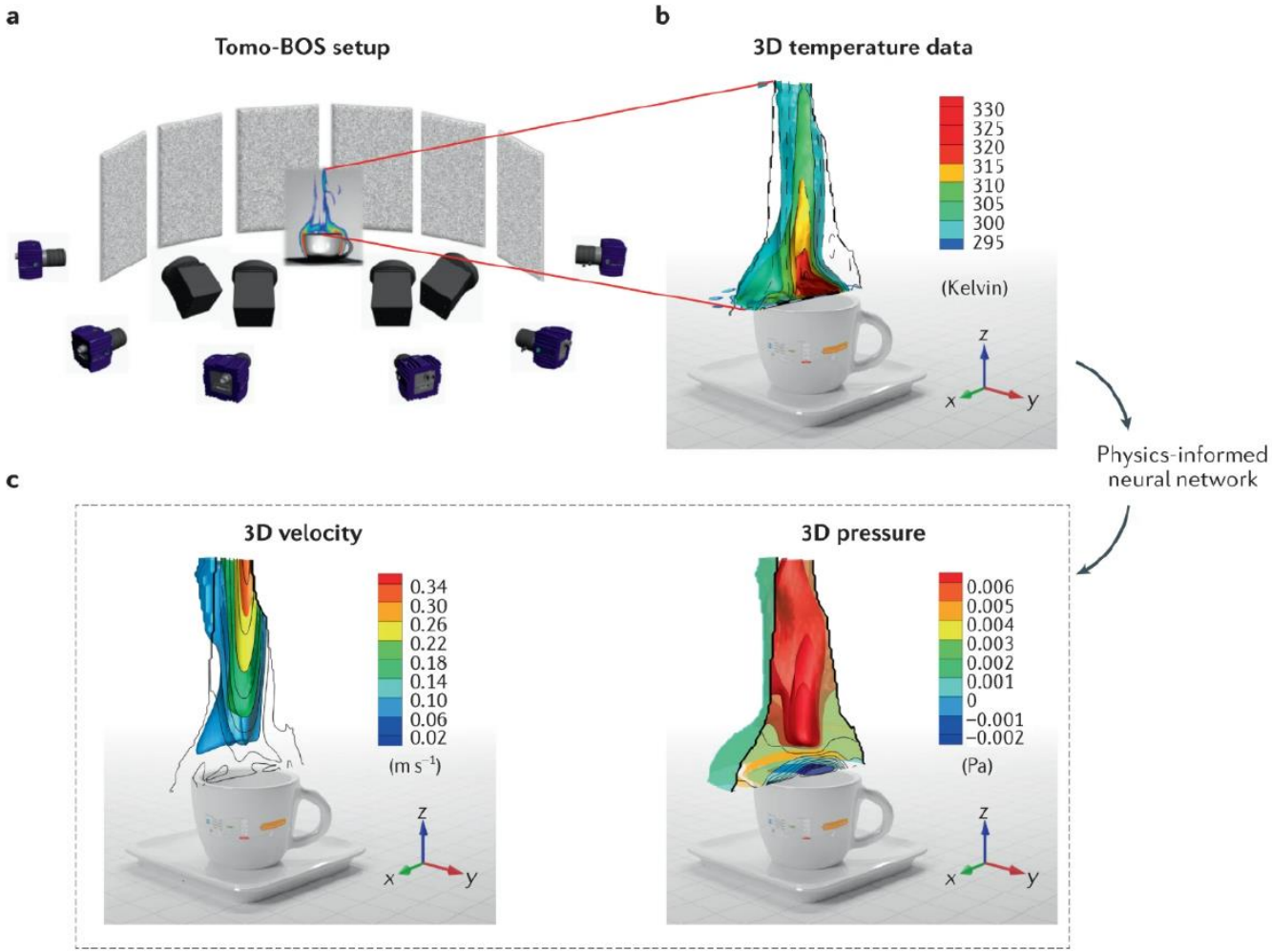
Parameter inference for systems biology models

Ananya Rastogi

Nature Computational Science **1**, 16(2021) | Cite this article

More applications: Inferring the flow over an espresso cup

Data: A video of temperature field.

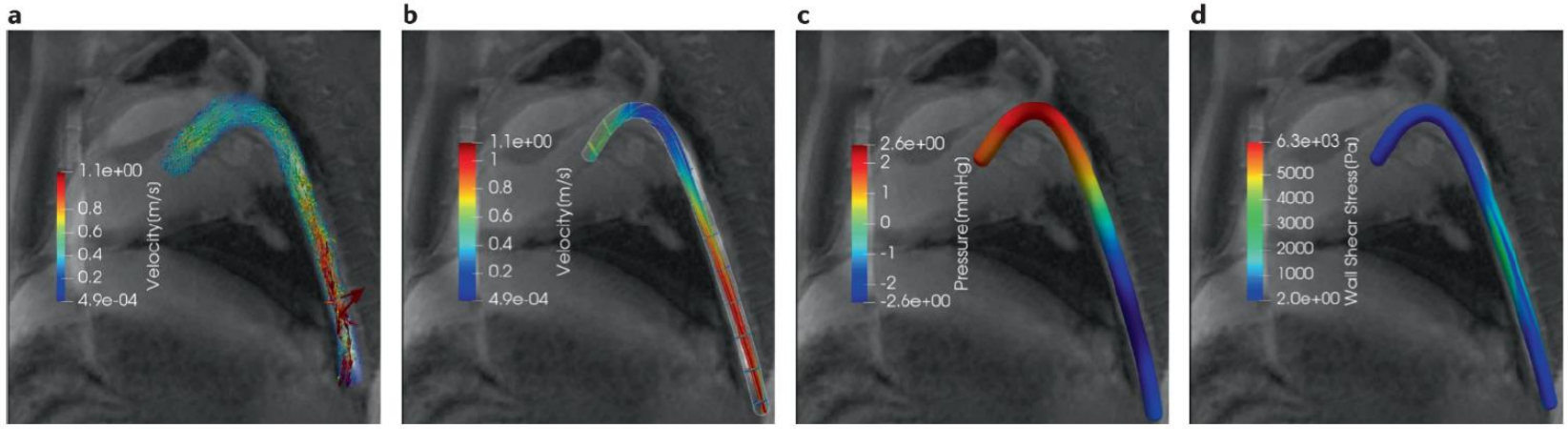


Cai et al., *J Fluid Mech*, 2021

More applications:

Filtering of *in-vivo* 4D-flow magnetic resonance imaging (MRI) data of blood flow in a porcine descending aorta

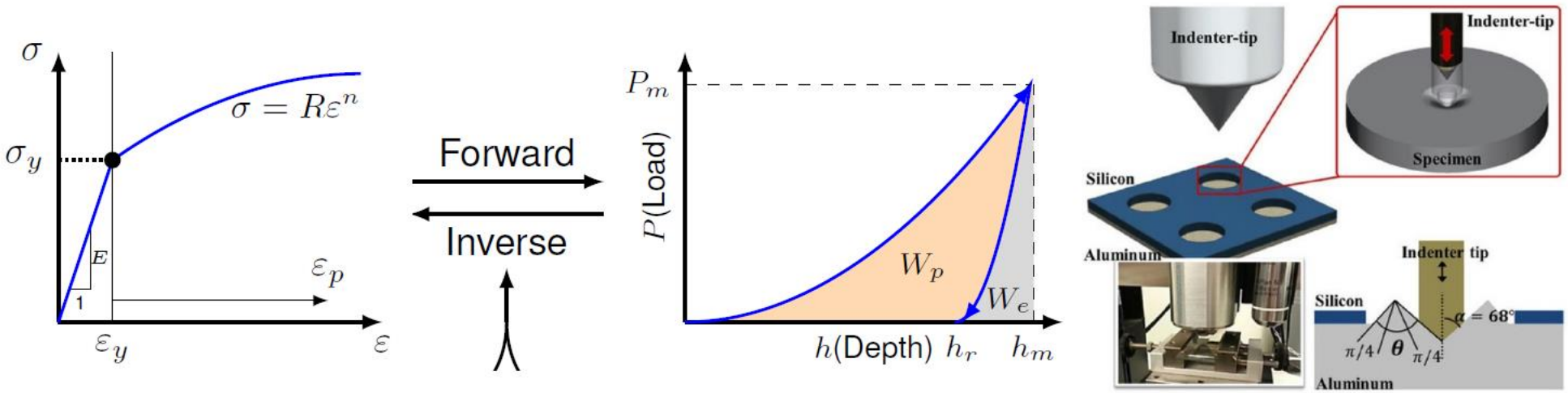
- MRI data: Very coarse resolution & heavily corrupted by noise



- Reconstructions of the velocity and pressure fields
- Identify regions of no-slip flow, from which one can reconstruct the location and motion of the arterial wall

Wang, Kissas, & Perdikaris

More applications: Inverse Indentation



Goal: Stress-strain response $\Leftarrow$ Penetration depth vs Loading force

Challenge: small data

- Physical laws: $E, \sigma_y, n \Leftrightarrow h_m, P_m, C, \frac{dP}{dh}|_{h_m}, \frac{W_p}{W_p+W_e}$
- Multi-fidelity: small experimental data + computational modeling

Lu et al., *PNAS*, 2020

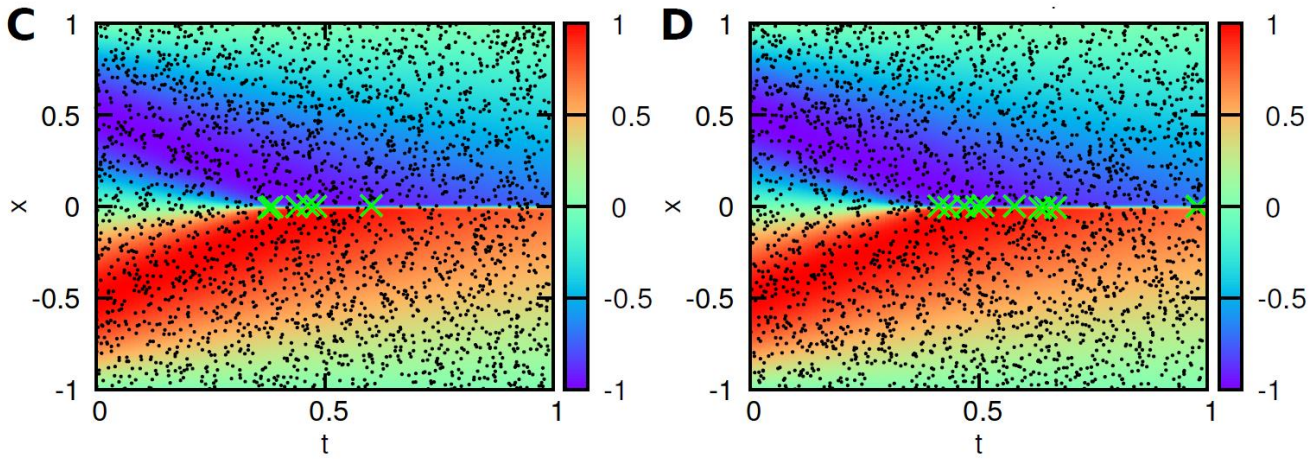
Other Useful techniques: Residual-based Adaptive Refinement (RAR)

Challenge: Uniform residual points sometimes are not efficient for steep solutions

e.g., Burgers equation $(x \in [-1, 1],\ t \in [0, 1])$:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}, \quad u(x, 0) = -\sin(\pi x), \quad u(-1, t) = u(1, t) = 0.$$

- **Idea**: adaptively add more points in locations with large PDE residual $\left\| f\left(\mathbf{x}; \frac{\partial \hat{u}}{\partial x_1}, \ldots, \frac{\partial \hat{u}}{\partial x_d}; \frac{\partial^2 \hat{u}}{\partial x_1 \partial x_1}, \ldots, \frac{\partial^2 \hat{u}}{\partial x_1 \partial x_d}; \ldots; \boldsymbol{\lambda}\right)\right\|$



Solution is discontinuous at *x=0*

Raissi et al., *J Comput Phys*, 2019

Other Useful techniques: Gradient-enhanced PINN (gPINN)

Idea: The derivatives of the PDE residual f are also zero.

$$\nabla f(\mathbf{x}) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \cdots, \frac{\partial f}{\partial x_d} \right) = \mathbf{0}, \quad \mathbf{x} \in \Omega$$

Embed the gradient of the PDE residual into the loss

$$\mathcal{L} = w_f \mathcal{L}_f + w_b \mathcal{L}_b + w_i \mathcal{L}_i + \sum_{i=1}^{d} w_{g_i} \mathcal{L}_{g_i} \left(\boldsymbol{\theta}; \mathcal{T}_{g_i} \right)$$

$$\mathcal{L}_{g_i} \left(\boldsymbol{\theta}; \mathcal{T}_{g_i} \right) = \frac{1}{|\mathcal{T}_{g_i}|} \sum_{\mathbf{x} \in \mathcal{T}_{g_i}} \left| \frac{\partial f}{\partial x_i} \right|^2$$

Yu, **Lu**†, et al., arXiv:2111.02801

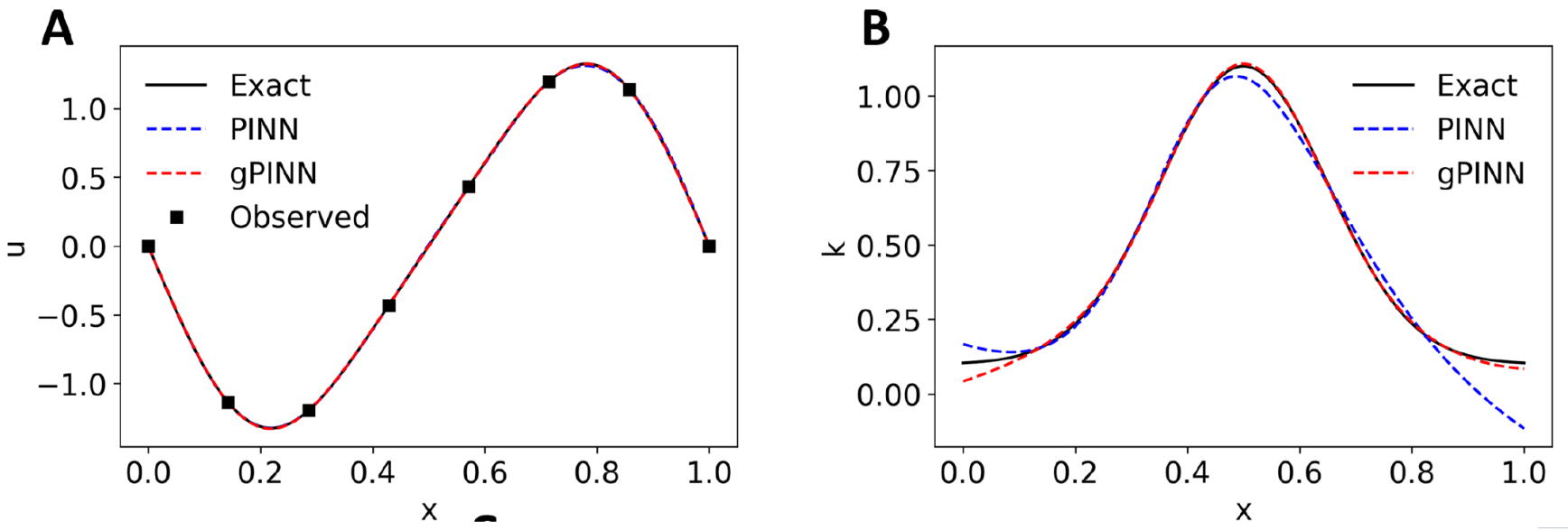
Inferring the reaction rate in a diffusion-reaction system (gPINN)

A diffusion-reaction system with the diffusion coefficient $\lambda = 0.01$:

$$\lambda \frac{\partial^2 u}{\partial x^2} - k(x)u = \sin(2\pi x), \quad x \in [0, 1]$$

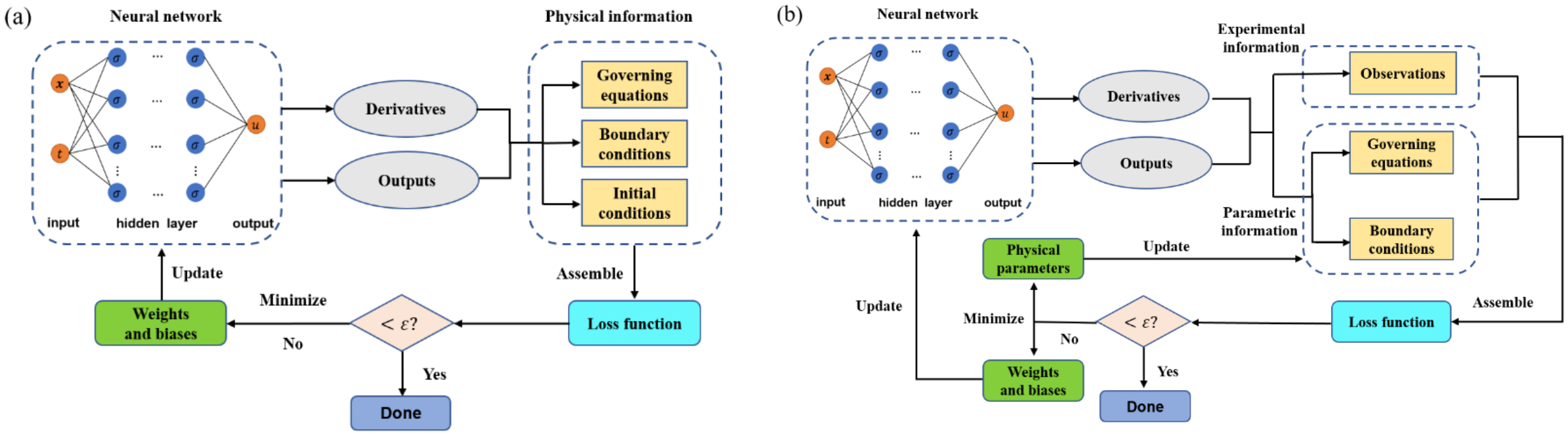
with zero boundary condition

- **Goal**: Infer space-dependent reaction rate $k(x)$
- **Data**: 8 observations of u & 10 PDE residual points



Yu, **Lu**†, et al., arXiv:2111.02801

Summary of Workflow for PINN



The schematic PINN working flows (a) for forward problems and (b) for inverse problems.

FOURIER NEURAL OPERATOR (Z. Li et. al. 2021)

FOURIER NEURAL OPERATOR FOR PARAMETRIC PARTIAL DIFFERENTIAL EQUATIONS

Zongyi Li
zongyili@caltech.edu

Nikola Kovachki
nkovachki@caltech.edu

Kamyar Azizzadenesheli
kamyar@purdue.edu

Burigede Liu
bgl@caltech.edu

Kaushik Bhattacharya
bhatta@caltech.edu

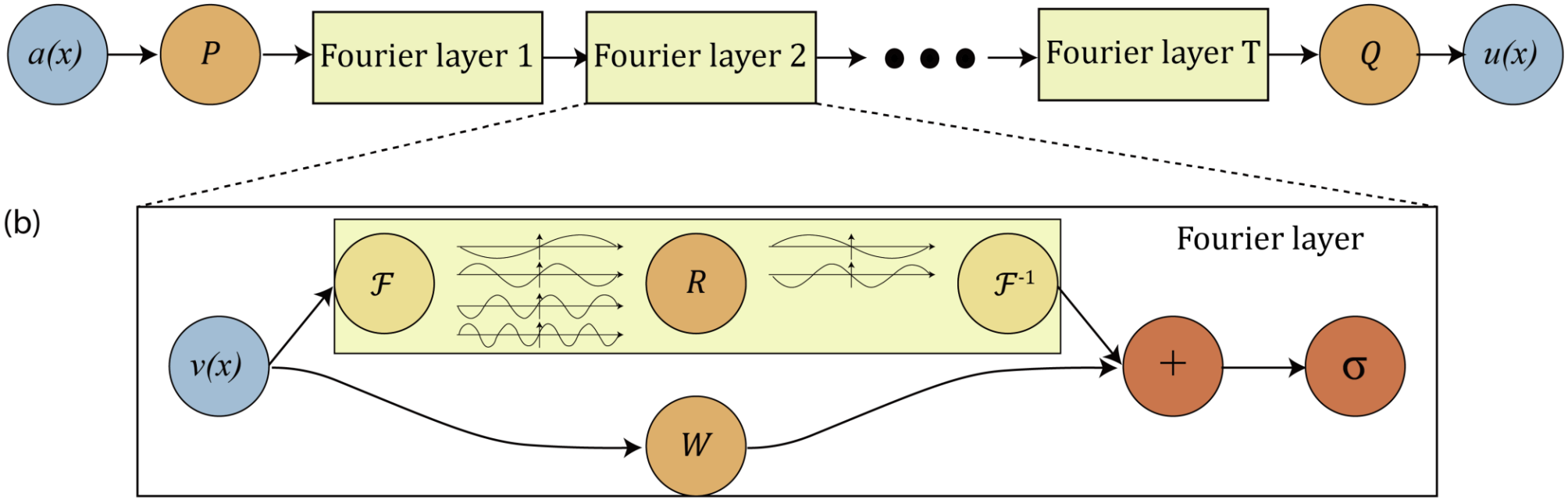
Andrew Stuart
astuart@caltech.edu

Anima Anandkumar
anima@caltech.edu

ABSTRACT

The classical development of neural networks has primarily focused on learning mappings between finite-dimensional Euclidean spaces. Recently, this has been generalized to neural operators that learn mappings between function spaces. For partial differential equations (PDEs), neural operators directly learn the mapping from any functional parametric dependence to the solution. Thus, they learn an entire family of PDEs, in contrast to classical methods which solve one instance of the equation. In this work, we formulate a new neural operator by parameterizing the integral kernel directly in Fourier space, allowing for an expressive and efficient architecture. We perform experiments on Burgers' equation, Darcy flow, and Navier-Stokes equation. The Fourier neural operator is the first ML-based method to successfully model turbulent flows with zero-shot super-resolution. It is up to three orders of magnitude faster compared to traditional PDE solvers. Additionally, it achieves superior accuracy compared to previous learning-based solvers under fixed resolution.

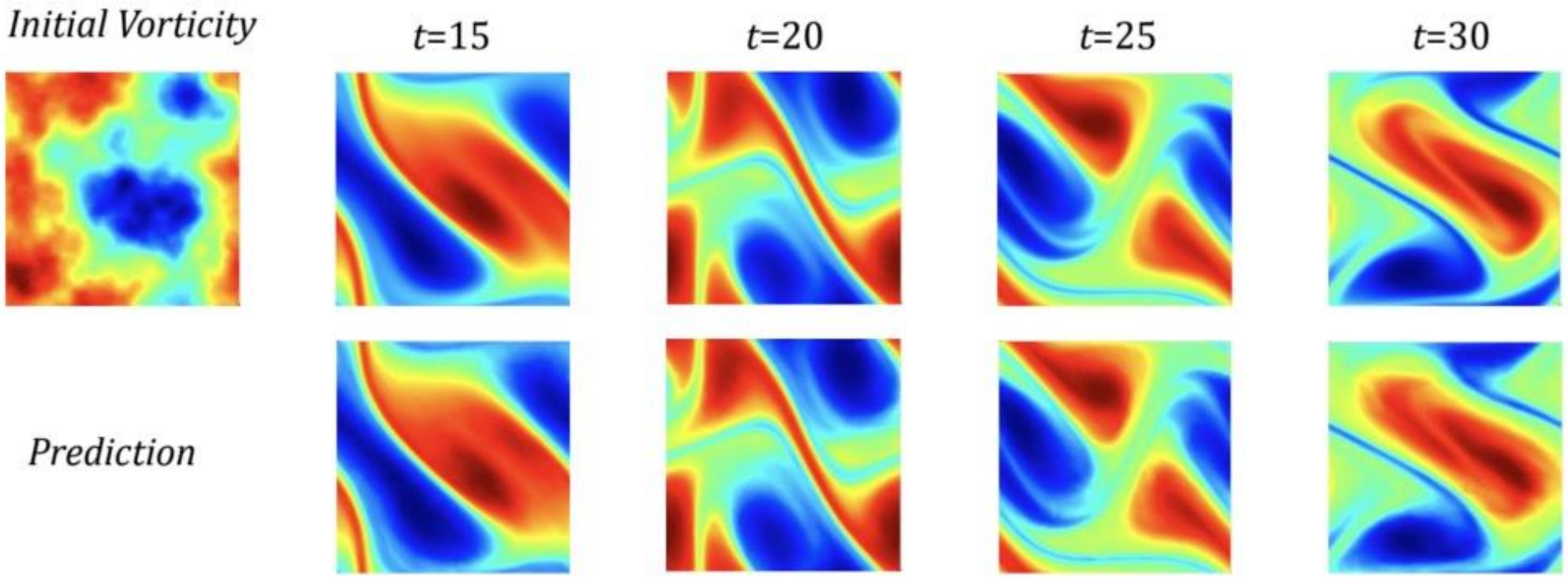
FOURIER NEURAL OPERATOR (Z. Li et. al. 2021)



(a) The full architecture of neural operator: start from input a. 1. Lift to a higher dimension channel space by a neural network P. 2. Apply four layers of integral operators and activation functions. 3. Project back to the target dimension by a neural network Q. Output u. **(b) Fourier layers**: Start from input v. On top: apply the Fourier transform $\mathcal{F}$; a linear transform R on the lower Fourier modes and filters out the higher modes; then apply the inverse Fourier transform $\mathcal{F}^{-1}$. On the bottom: apply a local linear transform W.

Figure 2: **top:** The architecture of the neural operators; **bottom:** Fourier layer.

FOURIER NEURAL OPERATOR (Z. Li et. al. 2021)



Zero-shot super-resolution: Navier-Stokes Equation with viscosity $\nu = 1e{-}4$; Ground truth on top and prediction on bottom; trained on $64 \times 64 \times 20$ dataset; evaluated on $256 \times 256 \times 80$ (see Section 5.4).

Figure 1: **top:** The architecture of the Fourier layer; **bottom:** Example flow from Navier-Stokes.

香港科技大学(广州)
THE HONG KONG
UNIVERSITY OF SCIENCE AND
TECHNOLOGY (GUANGZHOU)

Thanks for your attention!