2023 10th OpenFOAM Korea Users' Community Conference

물리정보 인공지능-OpenFOAM 결합 CFD 가속 연구

전준구

Assistant Professor, NINE Lab,

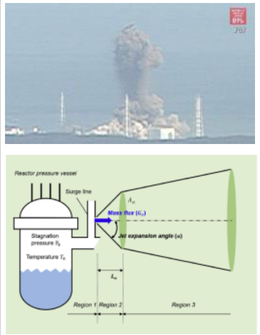
Graduate School of Integrated Energy-AI

Jeonbuk National University

Numerical Investigation for Nature & Energy Lab.

My research overview

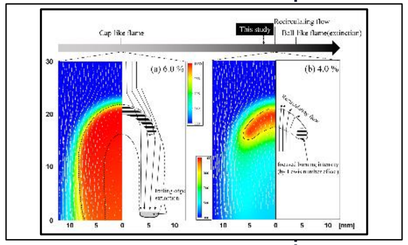
[1] J. Jeon et al., *Ann. Nucl. Energy*, 2018.
[2] J. Jeon et al., *Nucl. Eng. Technol.*, 2019.
[3] J. Jeon et al., *Energies*, 2020.
[4] J. Jeon et al., *Int. J. Heat Mass. Transf.*,2021
[5] J. Jeon et al., *Nucl. Eng. Technol.*, 2019.
[6] J. Jeon et al., *Nucl. Eng. Technol.*, 2021.
[7] J. Jeon et al., *Int. J. Energy Res.*, 2022.
[8] J. Jeon et al., arXiv preprint arXiv:2206.06817

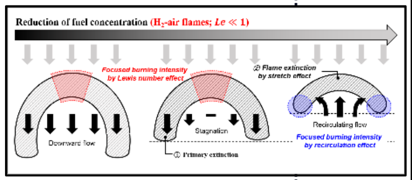


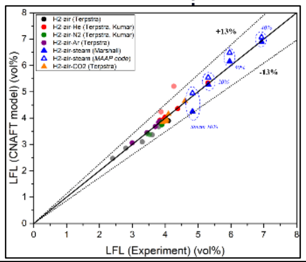
2016 — **2018** — **2020** — **2022** — **2023**

- *Numerical simulation of **severe accident** scenarios [1].*
- ***Turbulent gas behavior** at pipe rupture accident [2].*

"The need for hydrogen-LFL model"



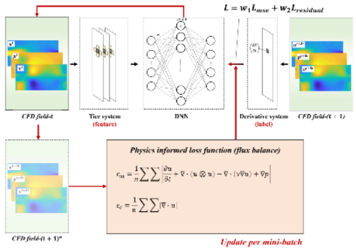


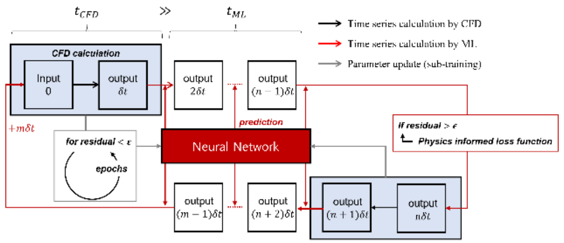


- *Identification of the **hydrogen flame extinction mechanism** with CFD simulation [3, 4].*
- *Analytical modeling and validation of **hydrogen flammability limit model** [5, 6].*
- *Development of **hydrogen combustion risk prediction code** for containment building (flammability, flame acceleration, DDT evaluation).*

"Need to accelerate CFD simulations"

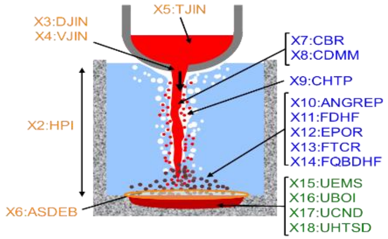
- *Development of a **new concept of network model** by introducing CFD principles [7].*
- *Validation of the model using **non-reacting and reacting flows**.*
- ***Physics-informed transfer learning strategy** to accelerate unsteady simulations [8].*





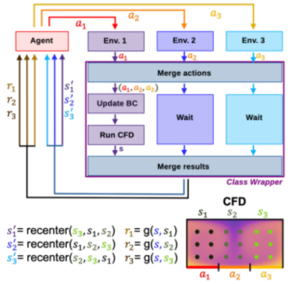
AI application for nuclear safety

- *RANS turbulent modeling with LES data*
- *Reinforcement learning approach to overcome scale differences*

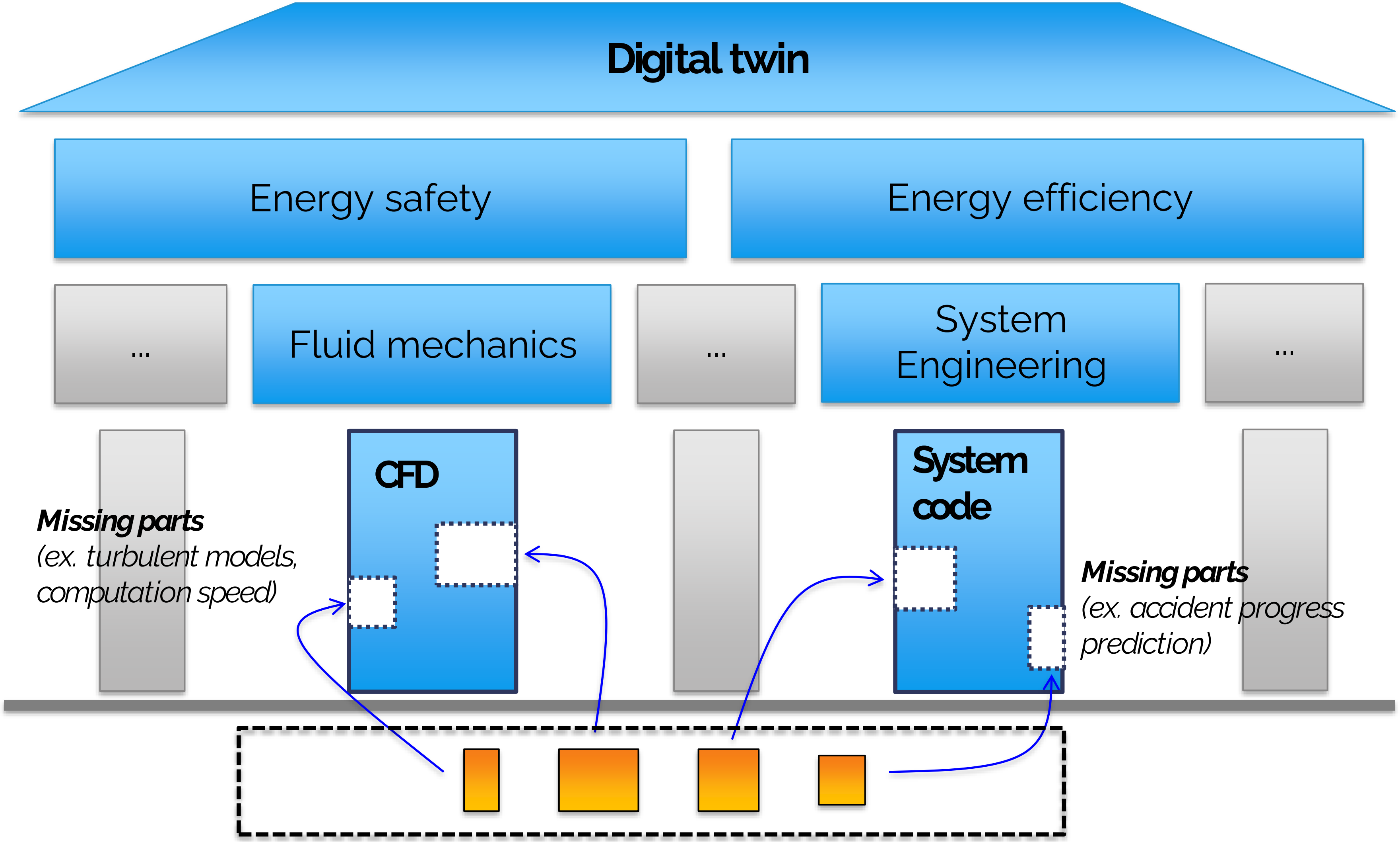


"Flow/heat control with DRL"

- *Control of 2D Rayleigh–Benard convection with multi-agent reinforcement learning*
- *Application of G-CNNs for symmetrical invariant*

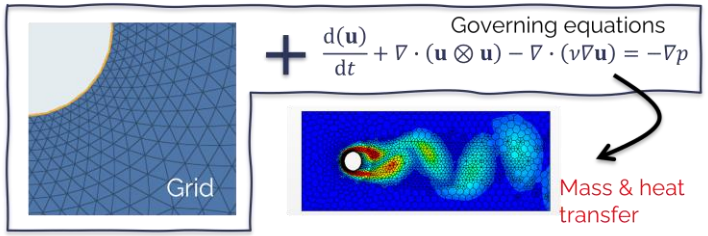


My research overview

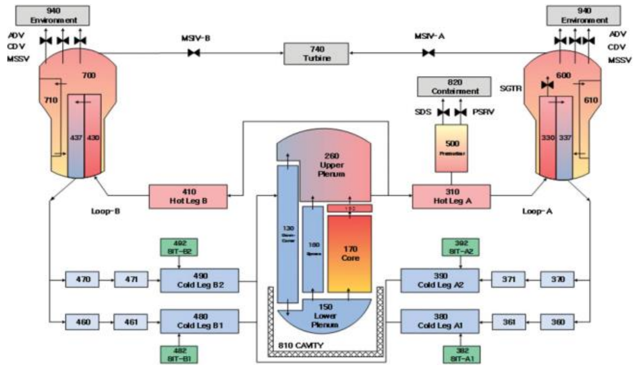


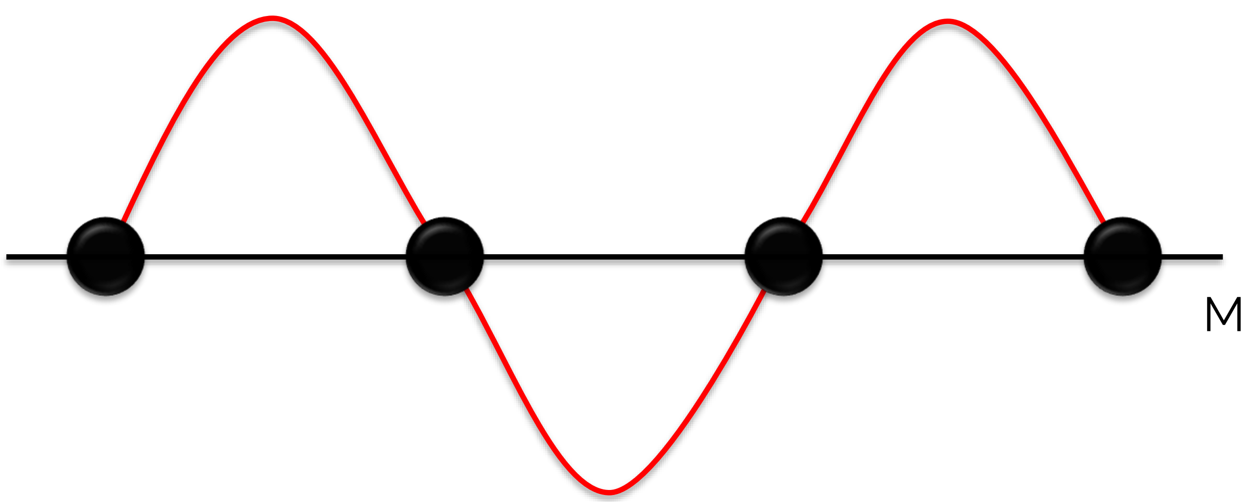
State-of-the art AI techniques

- Computational fluid dynamics (CFD)

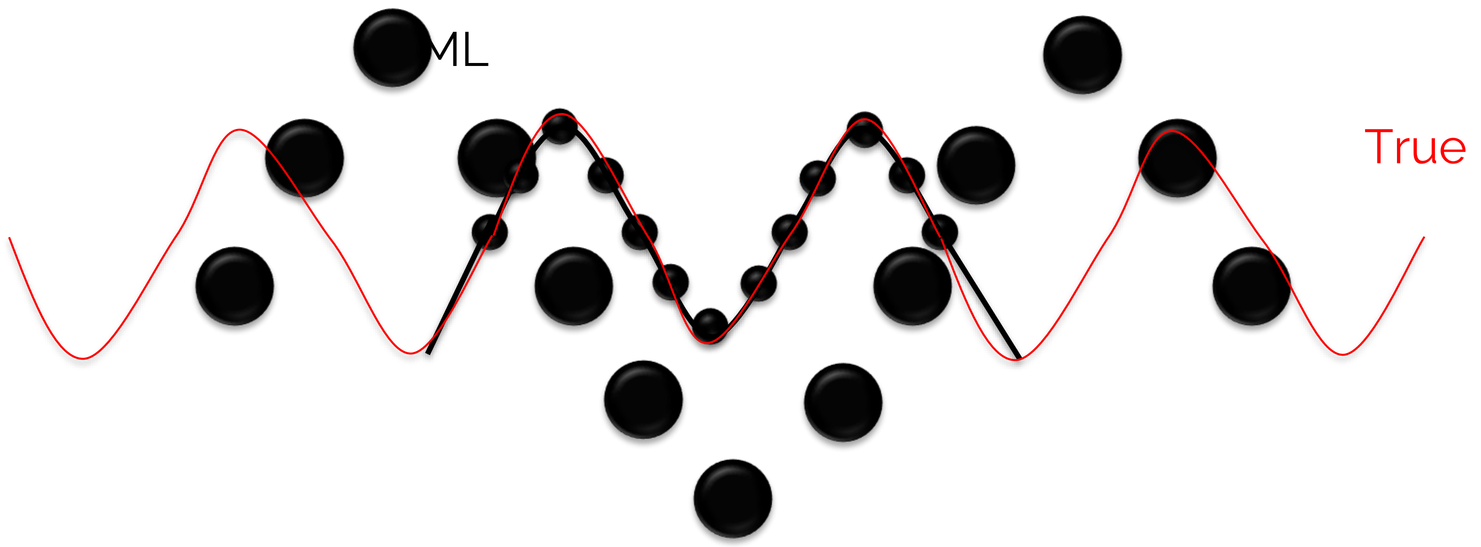


- System code (NPPs)



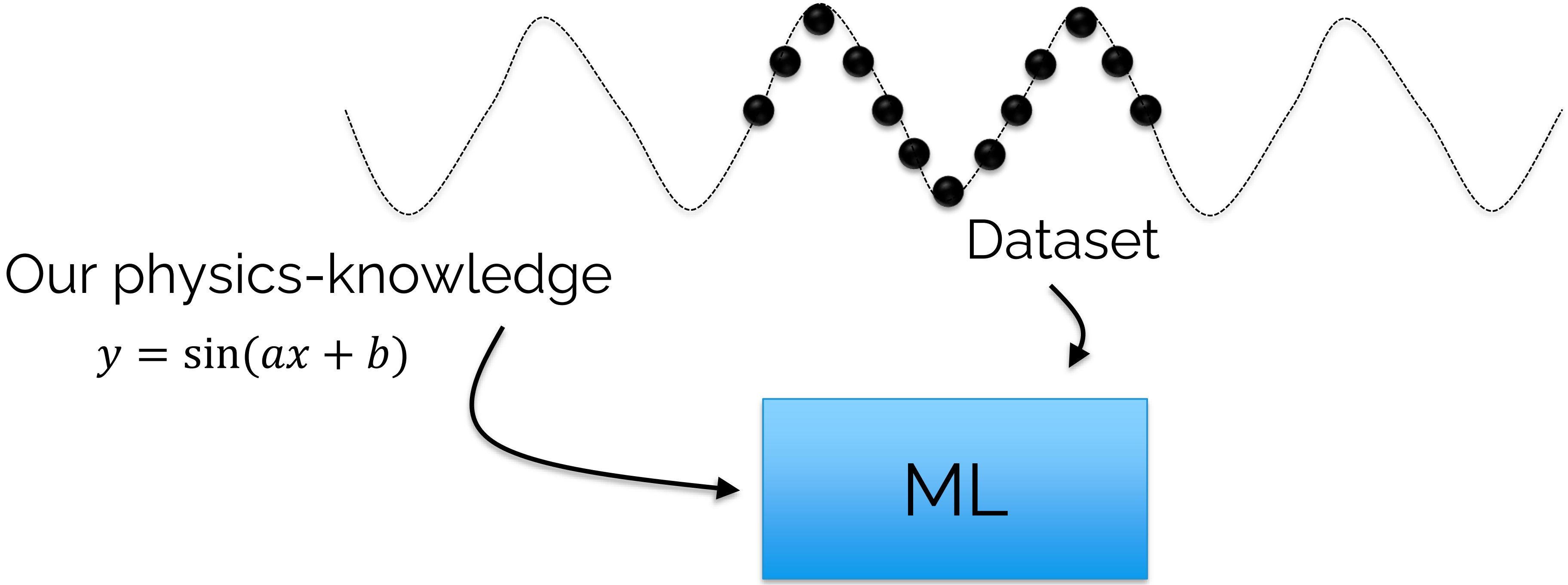


Problem 1: incorrect interpolation (insufficient data)



Problem 2: incorrect extrapolation (biased data)

"Almost all ML research faces these two problems"



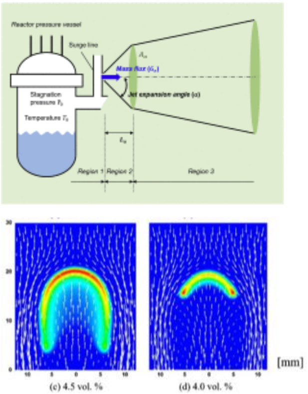
"Motivation for physics-informed machine learning"

Contents

Achilles heel of CFD

① **Unrealistic computation costs** (especially for turbulent, reacting, multiphase flows)

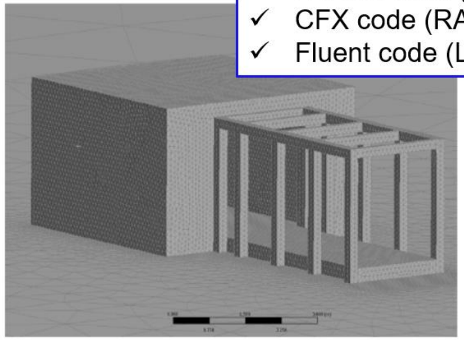
- my experiences…

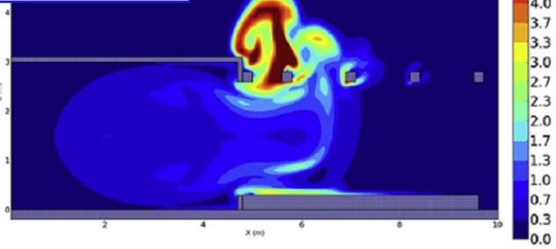


(Jeon et al., 2022)

Hydrogen explosion simulation (**~100h/1s**)

- ✓ FLACS code (RANS) ~ 6h/1s
- ✓ CFX code (RANS) ~ 80h/1s
- ✓ Fluent code (LES) ~ 177h/1s

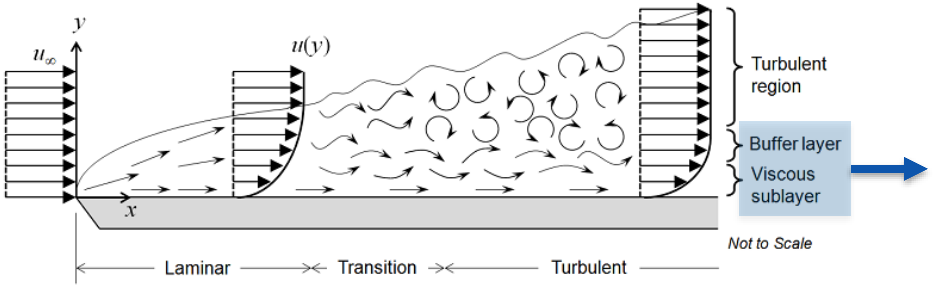




(Tolias et al., 2018)

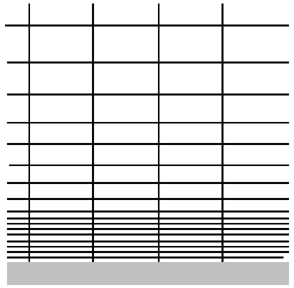
"Nuclear reactor severe accident simulation: 72 h"

- scaling studies for near-wall region

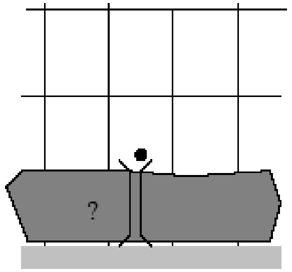


too expensive!

$\sim O(Re^{13/7})$



$\sim O(Re)$

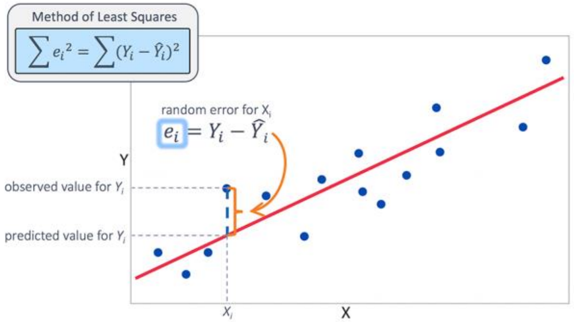


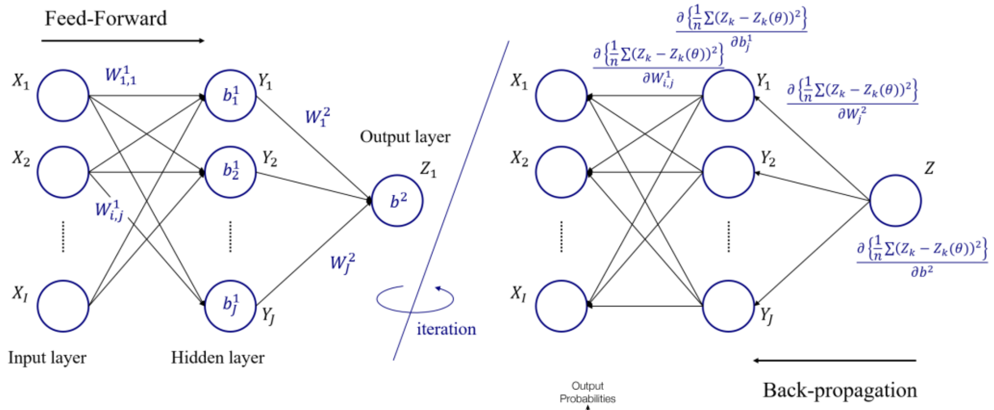
Based on LES

(S.T. Bose, 2019)

Neural networks: universal nonlinear function approximator

- The deep neural network algorithms were inspired by biological neural network .
- Below figure shows **the feed-forward algorithms in two-layer network model.**
  :I-dimensional input matrix and J unit number of a hidden layer
- The back-propagation allows to optimize parameter values.
  - Eq. (4) shows the representative loss function (mean square error)





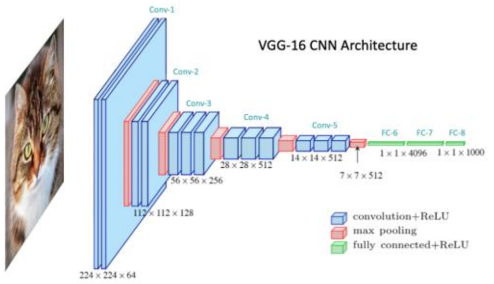
$$Y_j = \sum_i W^1_{i,j} X_i + b^1_j \tag{1}$$

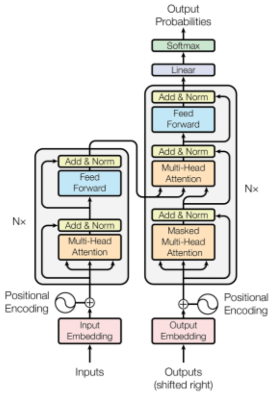
$$Z = \sum_j W^2_j Y_j + b^2 = \sum_j \left(W^2_j\left(\sum_i W^1_{i,j} X_i + b^1_j\right)\right) + b^2 \tag{2}$$

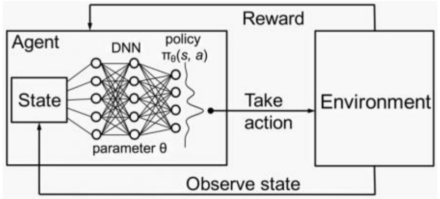
$$Z = \sum_j \left(W^2_j \cdot \mathrm{relu}(Y_j)\right) + b^2 = \sum_j \left(W^2_j \cdot \mathrm{relu}\left(\sum_i W^1_{i,j} X_i + b^1_j\right)\right) + b^2 \tag{3}$$

$$J(\theta) = \frac{1}{n}\sum_{k=1}^{n}\left(Z^k - Z^k(\theta)\right)^2 \tag{4}$$

$$\begin{aligned}\frac{\partial J}{\partial W^1} &= \left(\frac{\partial J}{\partial Y}\cdot\frac{\partial Y}{\partial W^1}\right)^{\mathsf{T}} = \left(\frac{\partial J}{\partial Y}\cdot X^{\mathsf{T}}\right)^{\mathsf{T}} \\ &= \left(W^2\cdot\left((W^2)^{\mathsf{T}}\cdot\mathrm{relu}\left((W^1)^{\mathsf{T}}X + b^1\right) + b^2 - Z\right)\cdot X^{\mathsf{T}}\right)^{\mathsf{T}}\end{aligned} \tag{5}$$



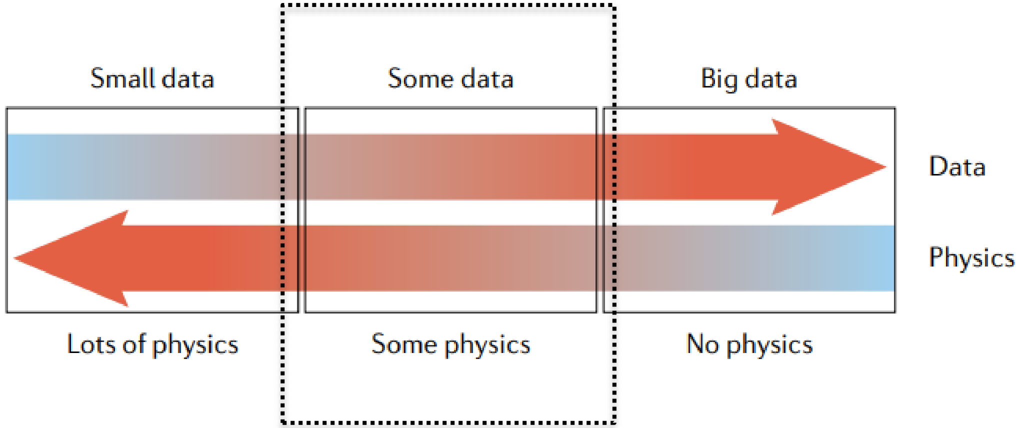




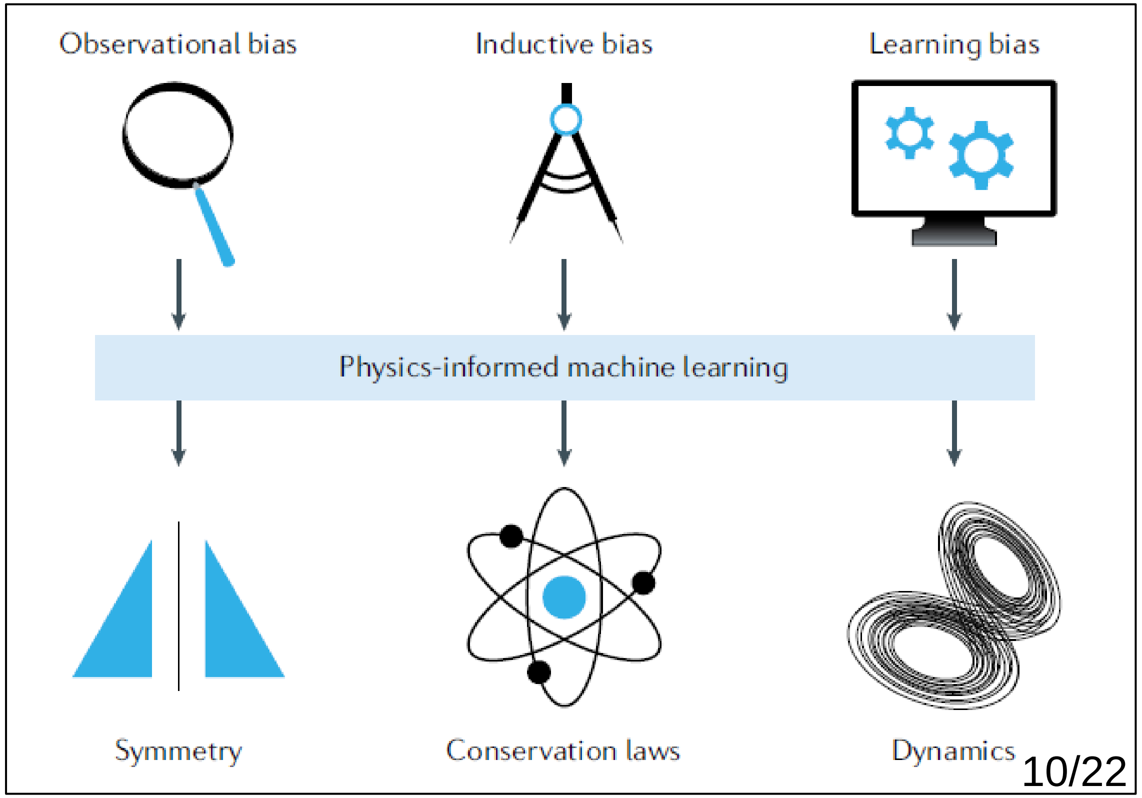
Recent advances in ML-PDEs

- Actually, it includes more broad ideas!!!
- We aims to enhance **accuracy & efficiency of NNs**.

Prof. George E. Karniadakis

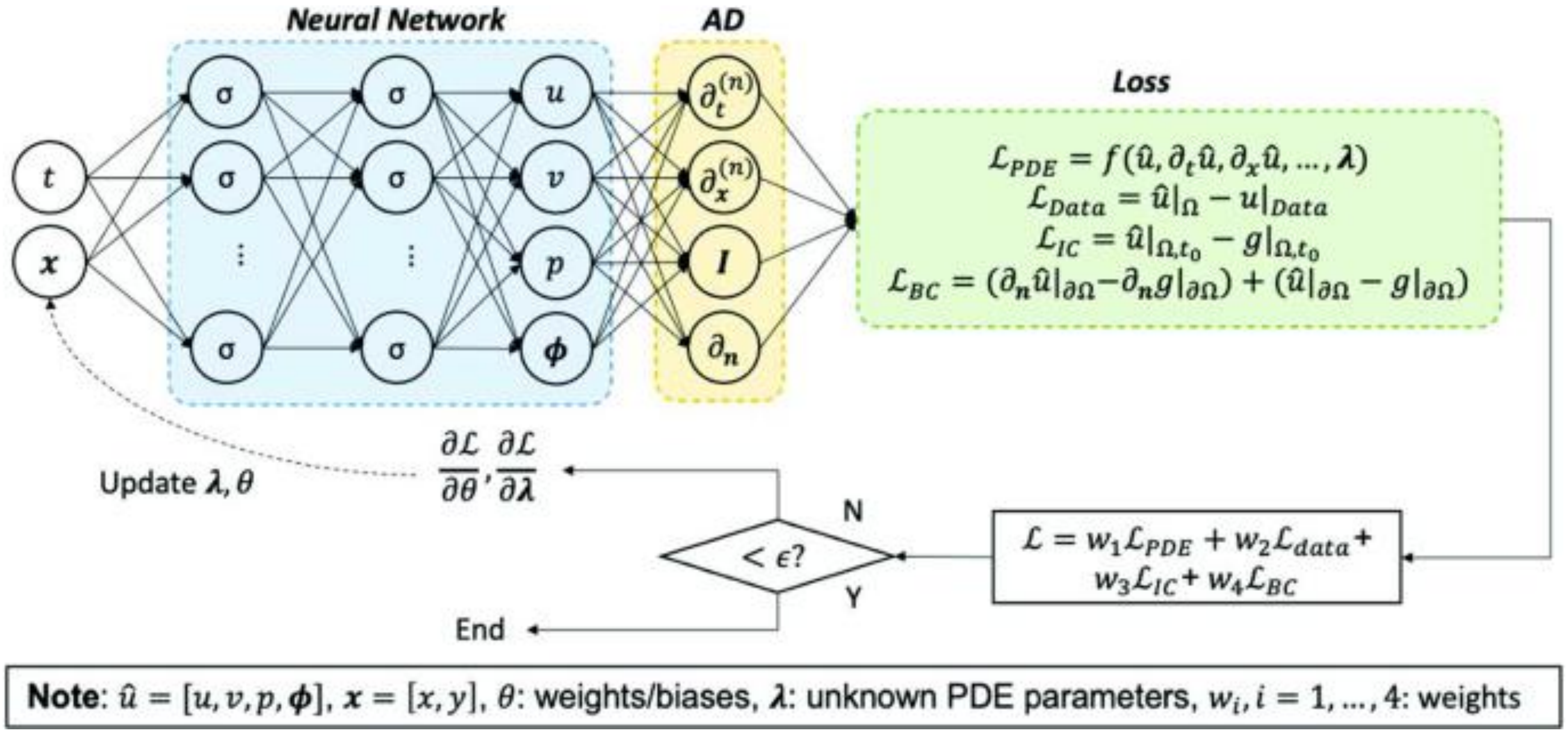


Nuclear engineering:
Multiphysics applications



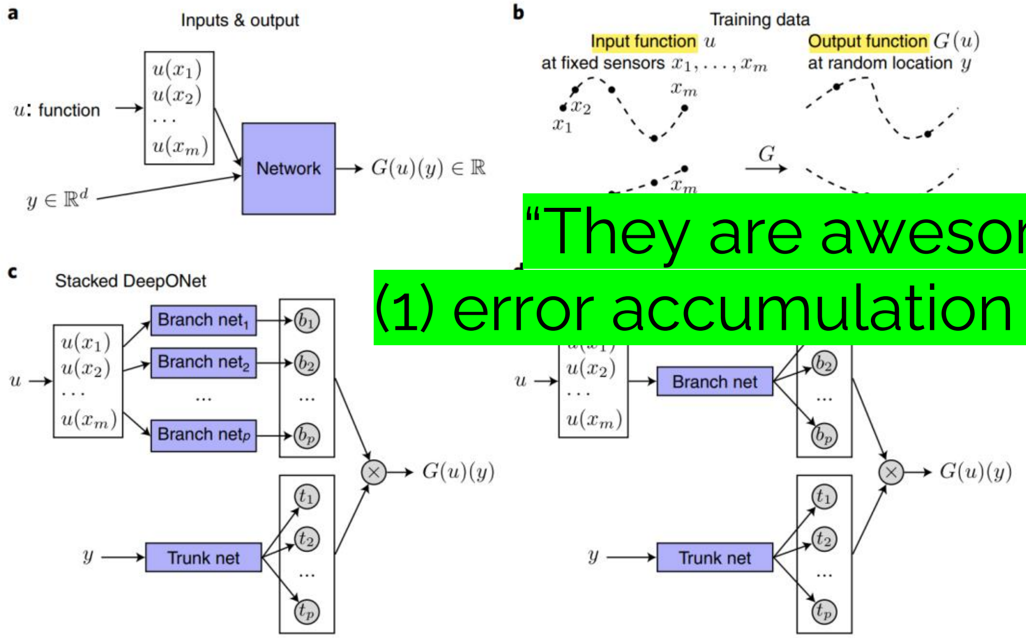
Physics-informed neural networks (PINNs) (M. Raissi, 2019)

- Network design



(Cai, 2022)

DeepONet (L. Lu, 2021)



"They are awesome, but have limitations
(1) error accumulation (2) changes in geometry/b.c".

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,* ... *> 0, there are positive integers* ... *, $w_k \in \mathbb{R}^d$, $x_j \in K_1$, $i = 1, \dots, n$,*

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

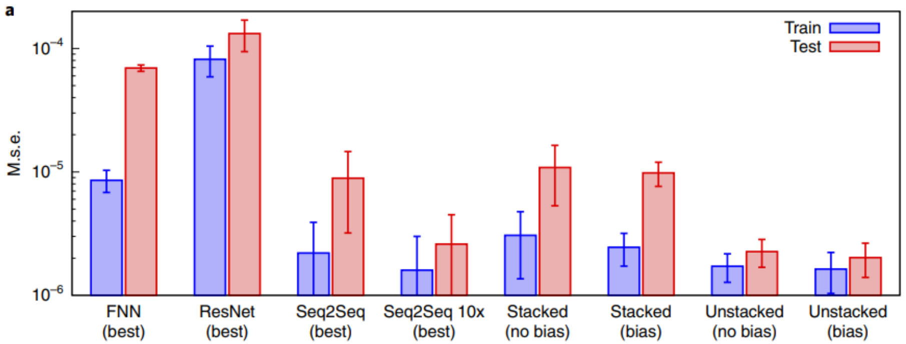
- **Example: antiderivative operator**

$$\frac{ds(x)}{dx} = u(x), s(x) = s_0 + \int_0^x u(\tau)\, d\tau,$$

$$\boldsymbol{G}: \boldsymbol{u}(\boldsymbol{x}) \to \boldsymbol{s}(\boldsymbol{x}),$$

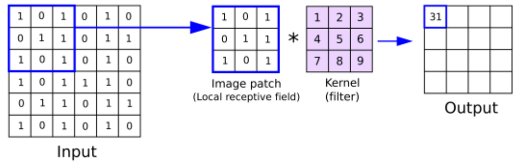
input: $\boldsymbol{u}(\boldsymbol{x}), \boldsymbol{y}$

output: $\boldsymbol{G}(\boldsymbol{u})(\boldsymbol{y})$



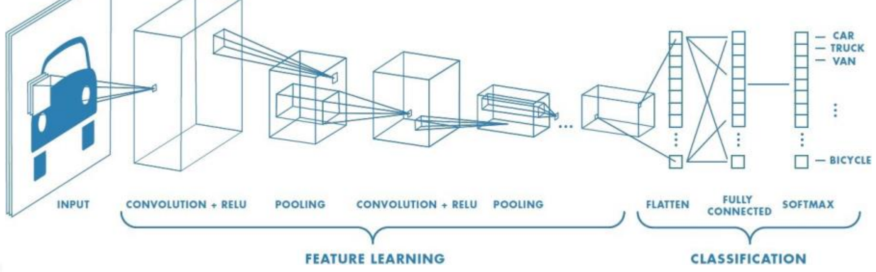
Finite volume method network (FVMN)

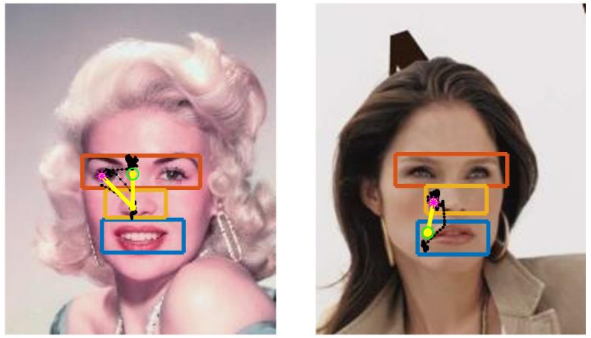
For best performance, we should develop a CFD fitted-network model!

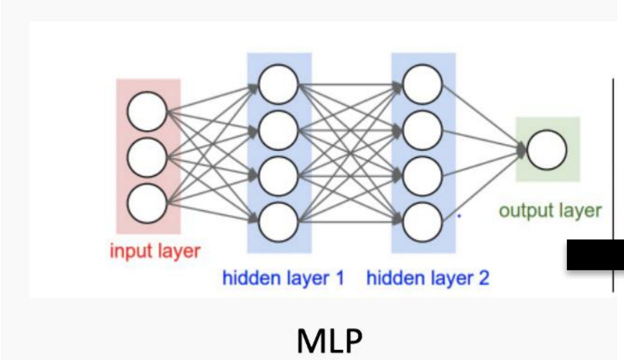


- Idea of CNN: image has the stationarity of statistic

Convolutional Neural Network





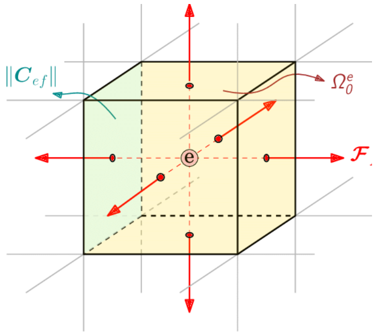


CNN: 1 image = 1 dataset
Our: 1 grid = 1 dataset

100,000 images
vs 10 images

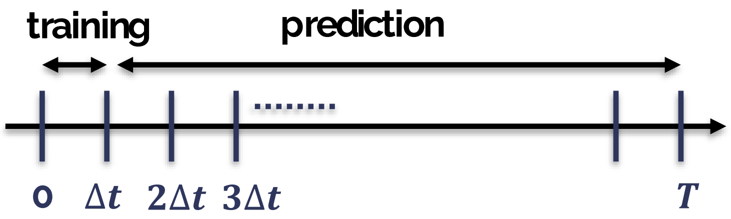
CFD acceleration!!

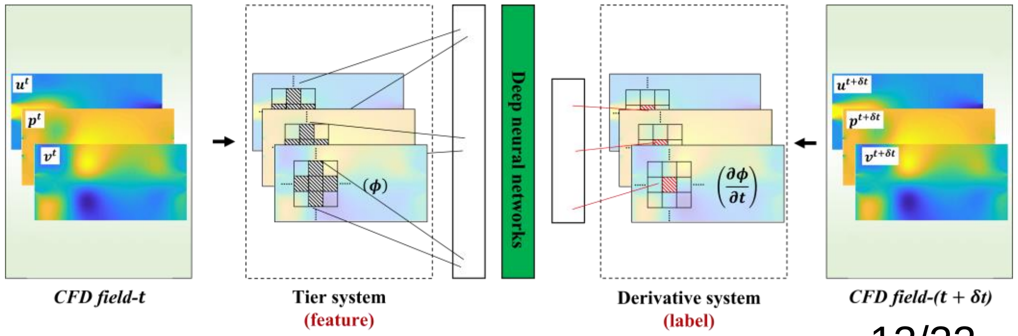
- Idea of our network model: all CFD nodes has the same rules



All nodes must be satisfied with near nodes:

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x}(\rho v_x) + \frac{\partial}{\partial y}(\rho v_y) = 0$$





(Jeon, 2022)

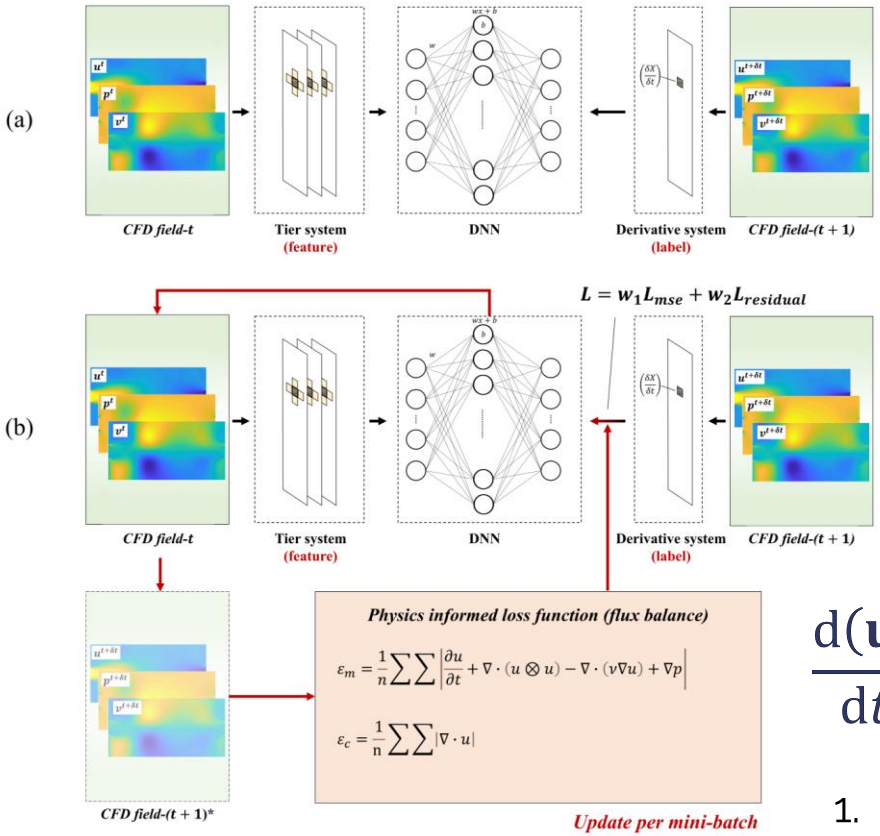
Finite volume method network (FVMN)

Physics-informed loss function

FVMN model

$$X_t^t = \left[x_{i,j}^t, x_{i-1,j}^t, x_{i+1,j}^t, x_{i,j-1}^t, x_{i,j+1}^t\right]^{\mathsf{T}} \text{ where } X_t^t \in R^5$$

$$Z_d^t = \left[\left(\frac{\delta x}{\delta t}\right)_{i,j}^{t+1}\right] \text{ where } Z_d^t \in R$$



(Jeon, 2022)

<Training loss>

(previous): $L_{mse} = \frac{1}{n}\sum_{k=1}^{n}\left(u^k - u^k(\theta)\right)^2$

(a): $L_{mse} = \frac{1}{n}\sum_{k=1}^{n}\left(\left(\frac{\delta u}{\delta t}\right)^k - \left(\frac{\delta u}{\delta t}(\theta)\right)^k\right)^2$

(b): $L_{mse} = w_1 \cdot \frac{1}{n}\sum_{k=1}^{n}\left(\left(\frac{\delta u}{\delta t}\right)^k - \left(\frac{\delta u}{\delta t}(\theta)\right)^k\right)^2 + w_2 \cdot \frac{1}{n}\sum_{j=1}^{2}\sum_{k=1}^{n}\left(\varepsilon_j^k\right)^2$

*ε_c: continuity residual, ε_m: Navier – Stokes residual

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

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

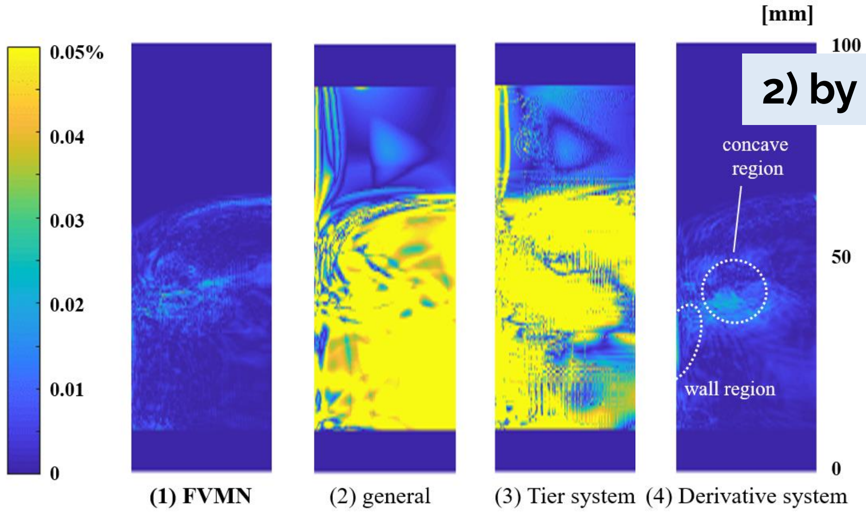
(Karniadakis, 2021)

$$\frac{d(\mathbf{u})}{dt} + \nabla\cdot(\mathbf{u}\otimes\mathbf{u}) - \nabla\cdot(\nu\nabla\mathbf{u}) + \nabla p = \varepsilon$$

1. Improve synchronization of FVM method and NNs
2. Prevention "non-physical overfitting"

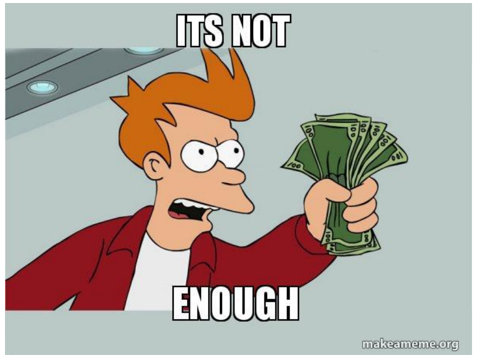
Finite volume method network (FVMN)

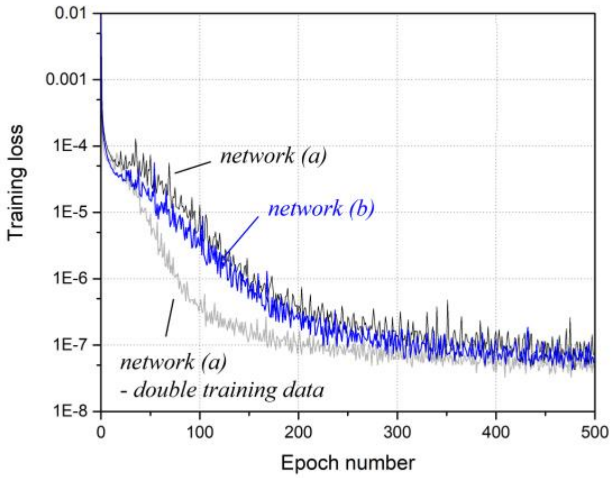
- Improved performance of FVMN

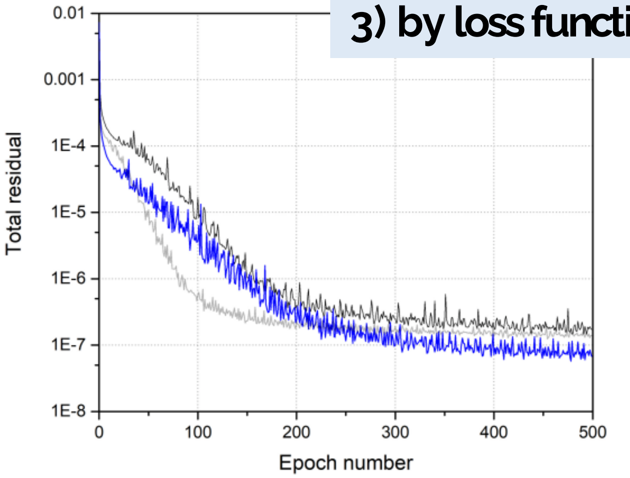


2) by FVM architecture

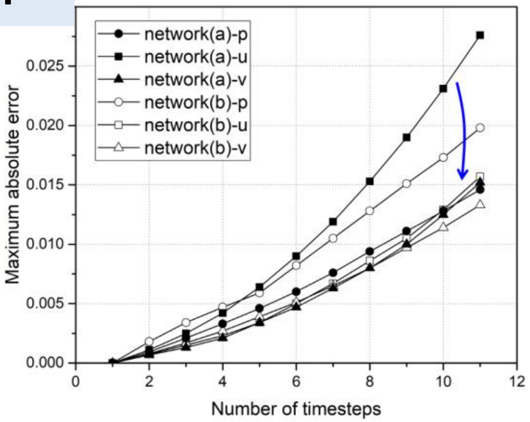
- Improved network performance
- **Reduced residuals** in prediction time series
- Still error growing…

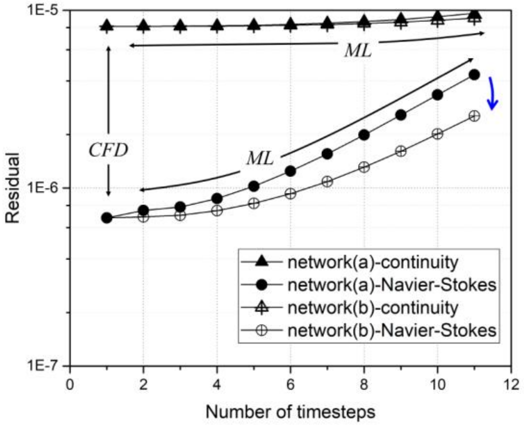






3) by loss function

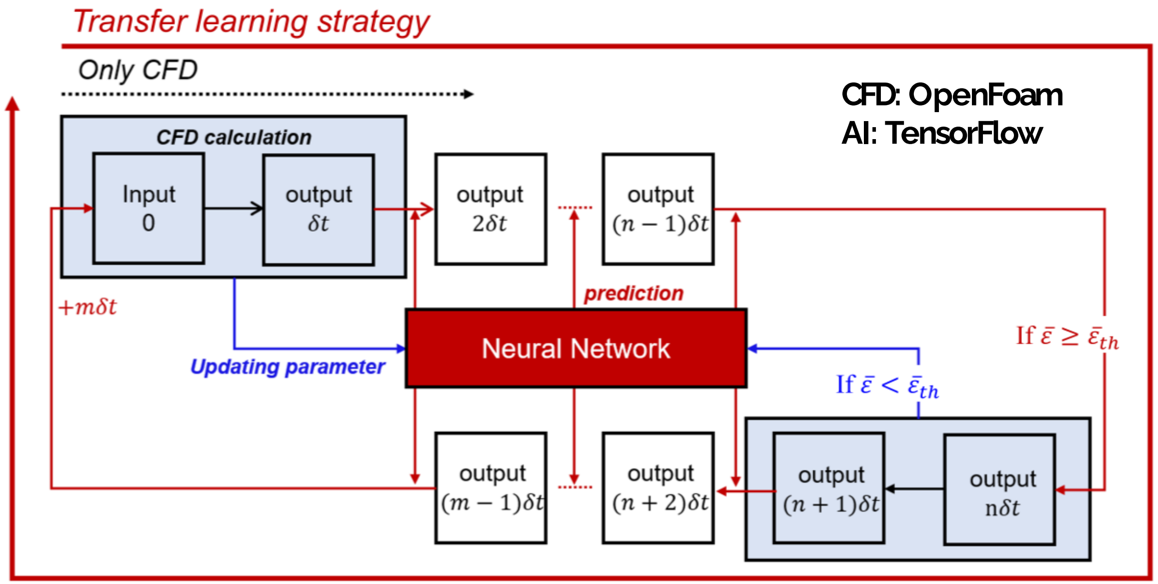




Our idea: hybrid approach

- **Re**sidual-based **p**hysics-**i**nformed **t**ransfer learning (RePIT) strategy

Car: AI
Repairman: CFD



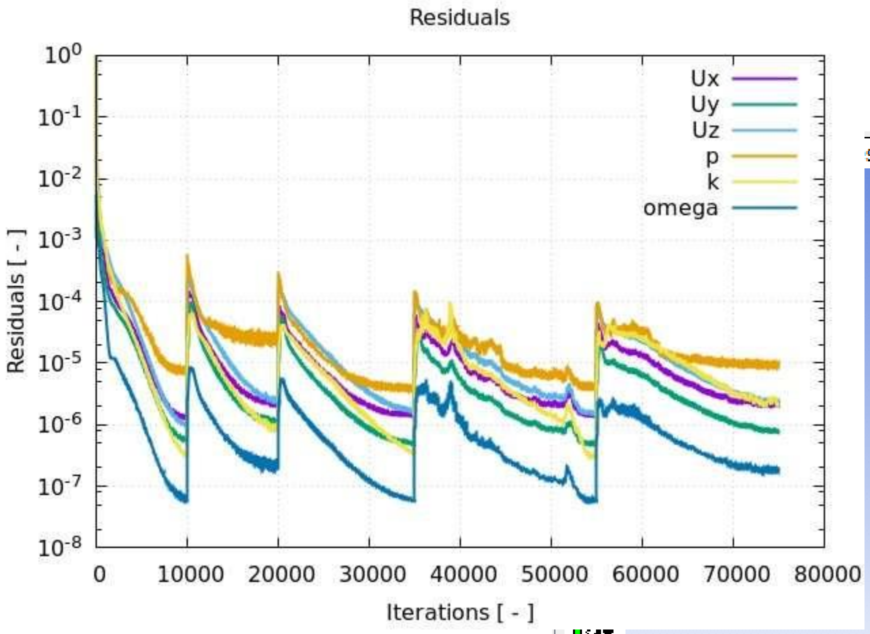
$$\frac{\mathrm{d}(\mathbf{u})}{\mathrm{d}t} + \nabla \cdot (\mathbf{u} \otimes \mathbf{u}) - \nabla \cdot (\nu \nabla \mathbf{u}) + \nabla p = \boldsymbol{\varepsilon}$$

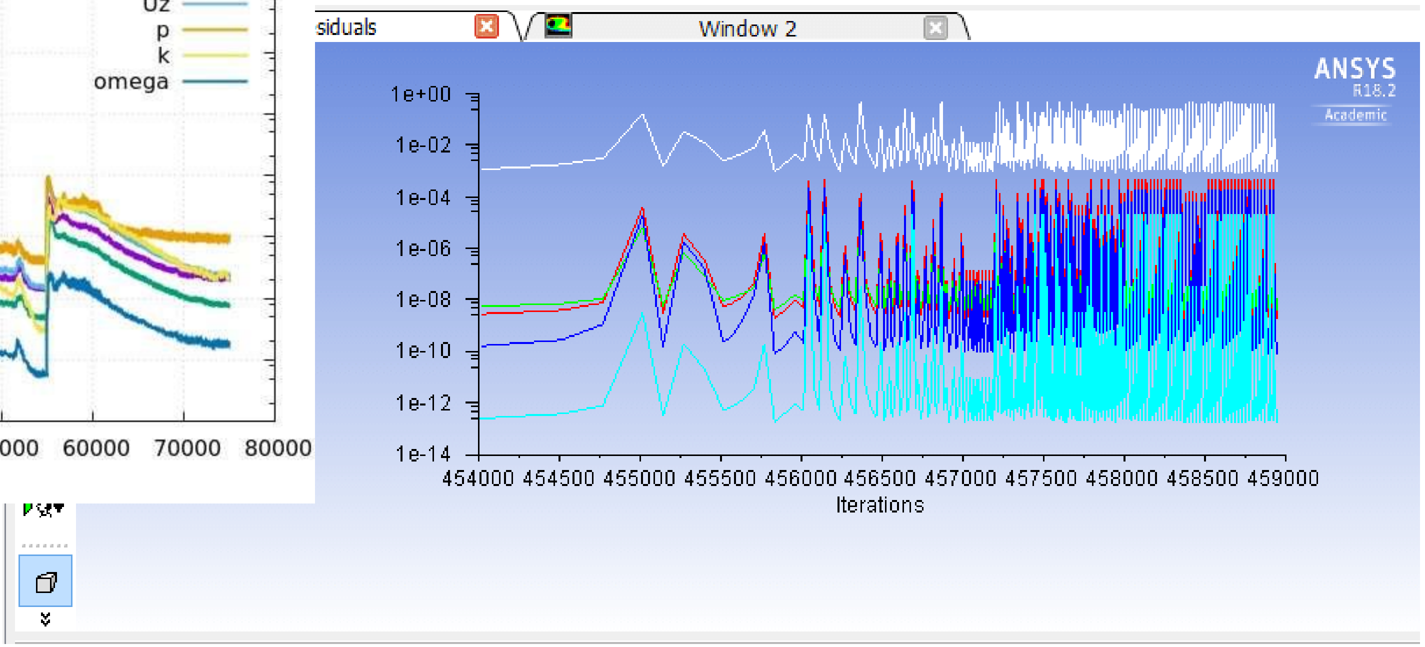
(Jeon, 2022)

Our idea: hybrid approach

- **Re**sidual-based **p**hysics-**i**nformed **t**ransfer learning (RePIT) strategy

  - Residual fluctuations also commonly occur in traditional CFD solvers.





Our idea: hybrid approach

- **Re**sidual-based **p**hysics-**i**nformed **t**ransfer learning (RePIT) strategy

  - Why OpenFOAM?

Commercial CFD $\xrightarrow{\text{Truncation error}}$ NNs $\xrightarrow{\text{Truncation error}}$ Commercial CFD

OpenFOAM $\rightarrow$ NNs $\rightarrow$ OpenFOAM

Unified framework

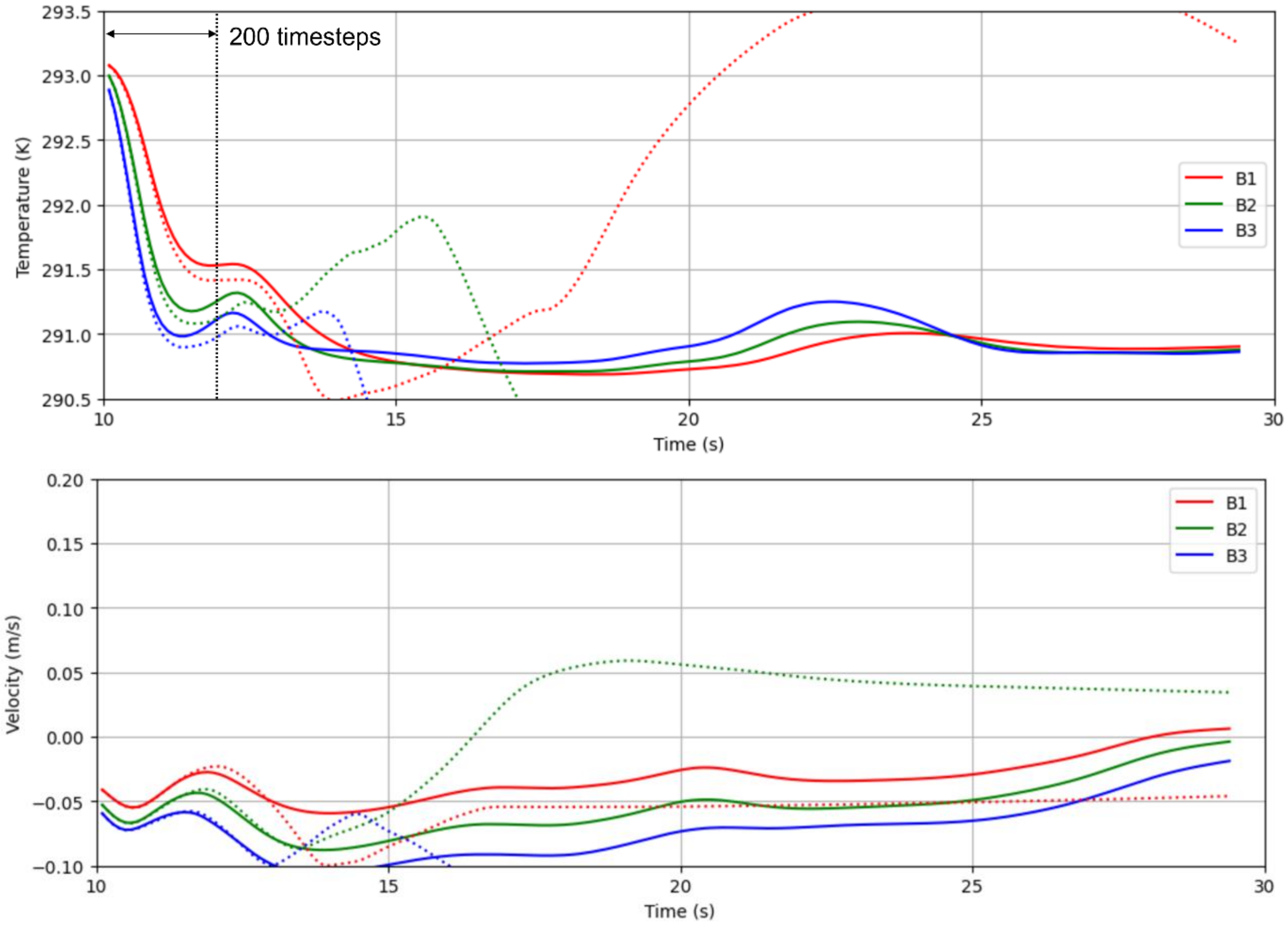
"OpenFOAM is very powerful to combine with ML algorithms"

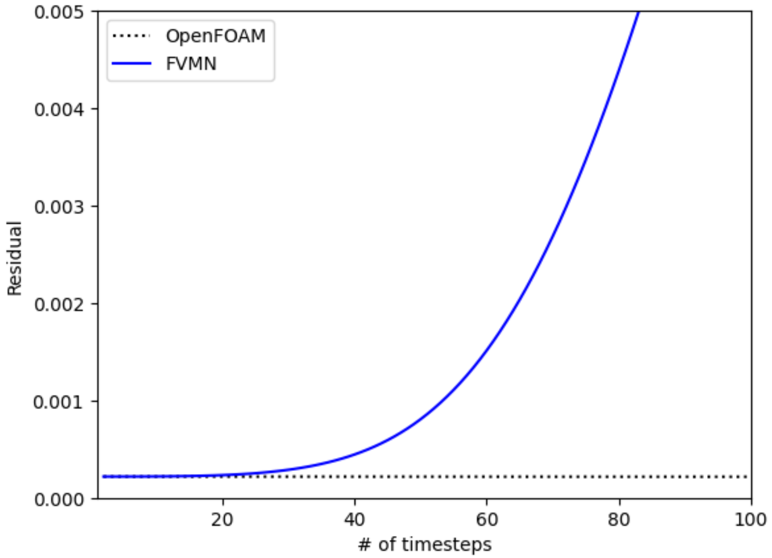
"Tremendous ML opensource codes"

PennState

Results and conclusion

- Single training approach (training data: initial 3 timesteps)





Residual divergence problem: residual of continuity equation

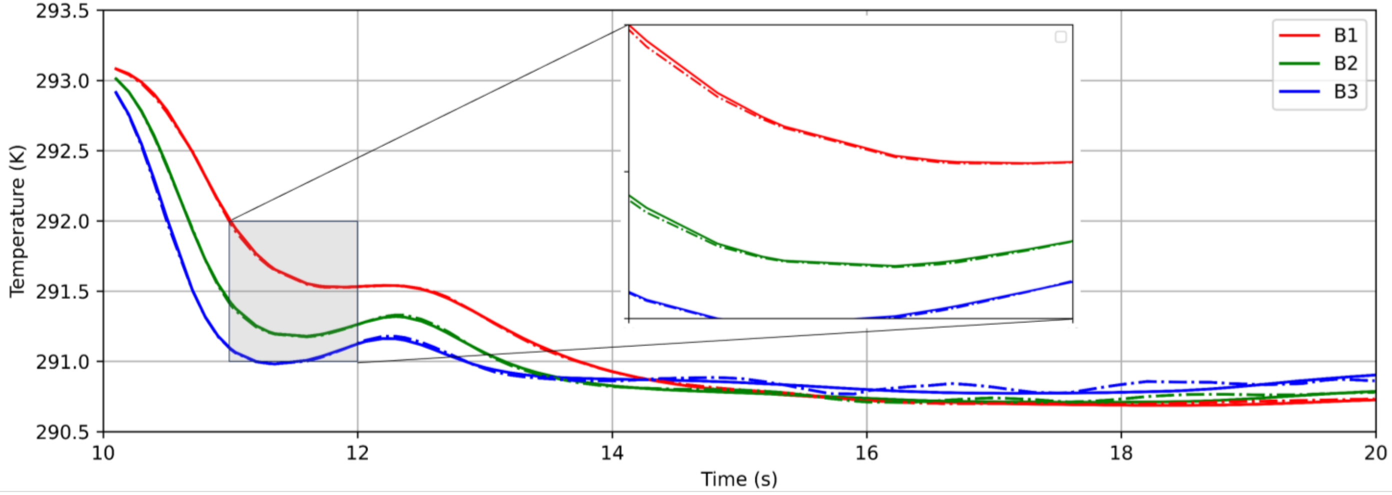
Results and conclusion

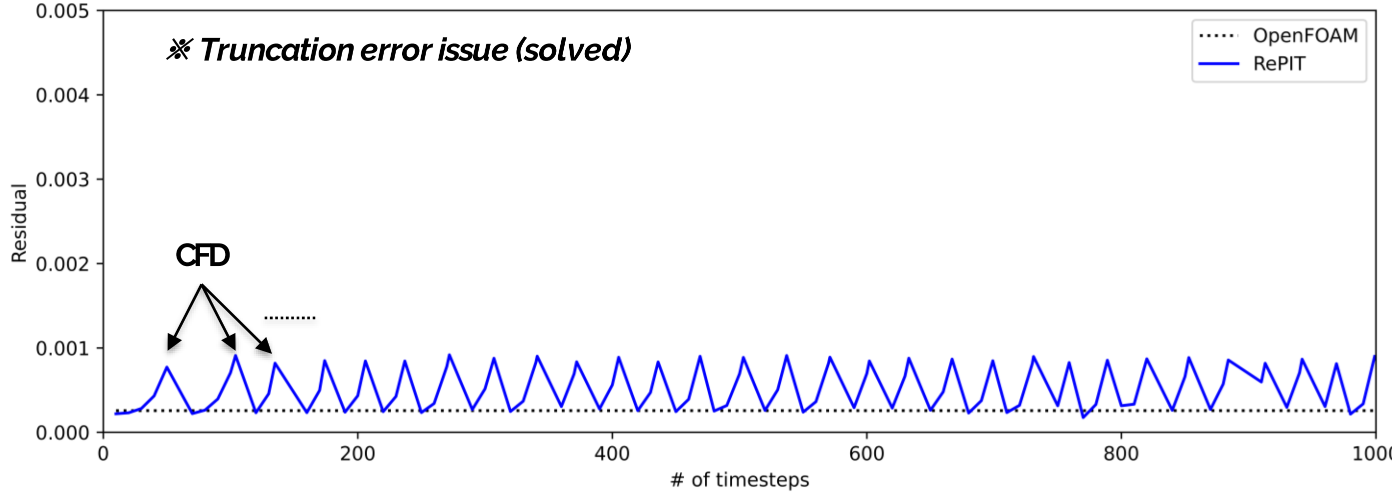
$$\frac{\rho^* - \rho_{ML}^t}{\delta t} + \nabla \cdot (\rho \boldsymbol{u})^* = \varepsilon$$

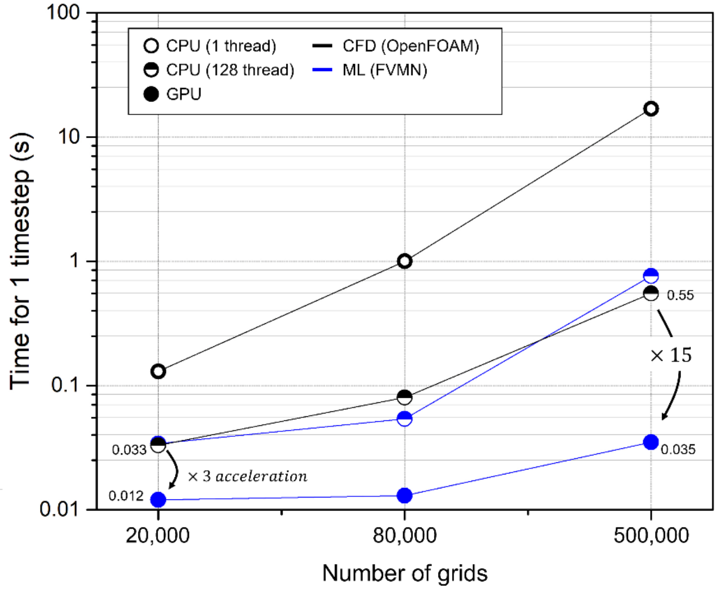
$$\frac{(\rho\mathbf{u})^* - (\rho\mathbf{u})_{ML}^t}{\delta t} + \nabla \cdot (\rho \boldsymbol{u}\boldsymbol{u})^* + \nabla p^* - \rho^* g - \nabla \cdot \left(\mu_{eff}(\nabla u + \nabla u^T)\right)^* + \nabla\left(\frac{2}{3}\mu_{eff}(\nabla \cdot \mathbf{u})\right)^* = \varepsilon$$

$$\frac{(\rho h)^* - (\rho h)_{ML}^t}{\delta t} + \nabla \cdot (\rho \boldsymbol{u} h)^* + \frac{(\rho K)^* - (\rho K)_{ML}^t}{\delta t} + \nabla \cdot (\rho \boldsymbol{u} K)^* - \frac{p^* - p_{ML}^t}{\delta t} - \nabla \cdot (\alpha_{eff} \nabla h)^* - \rho^* \boldsymbol{u}^* \cdot g = \varepsilon$$

- Feasibility study of RePIT strategy

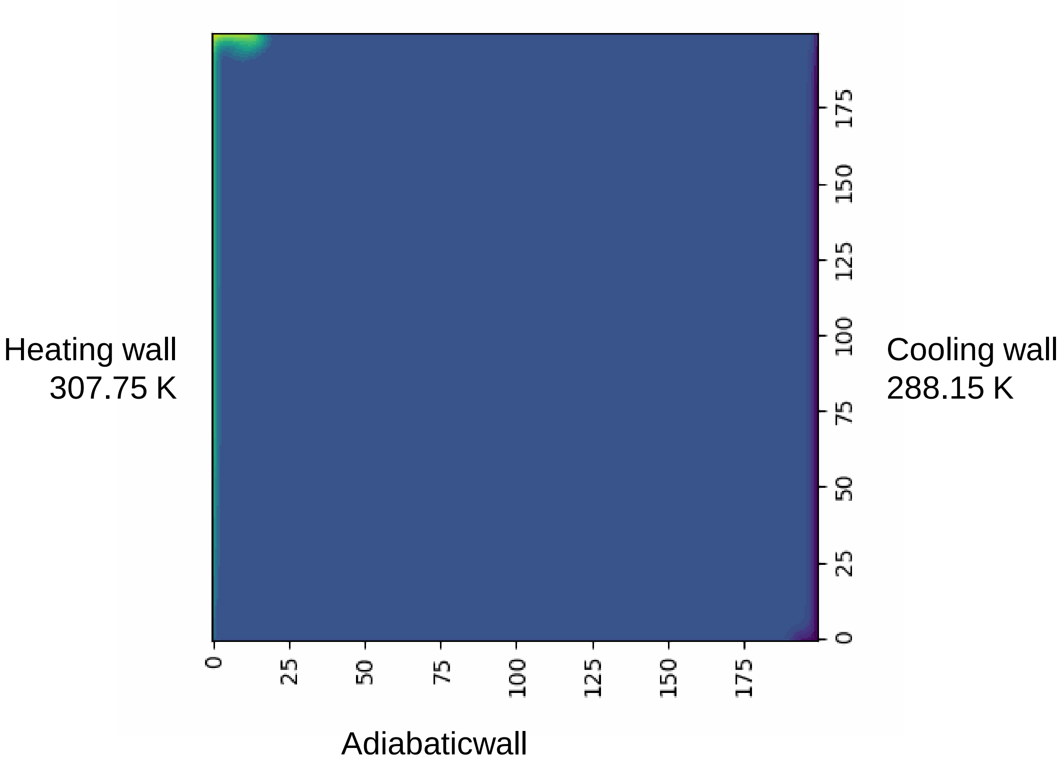


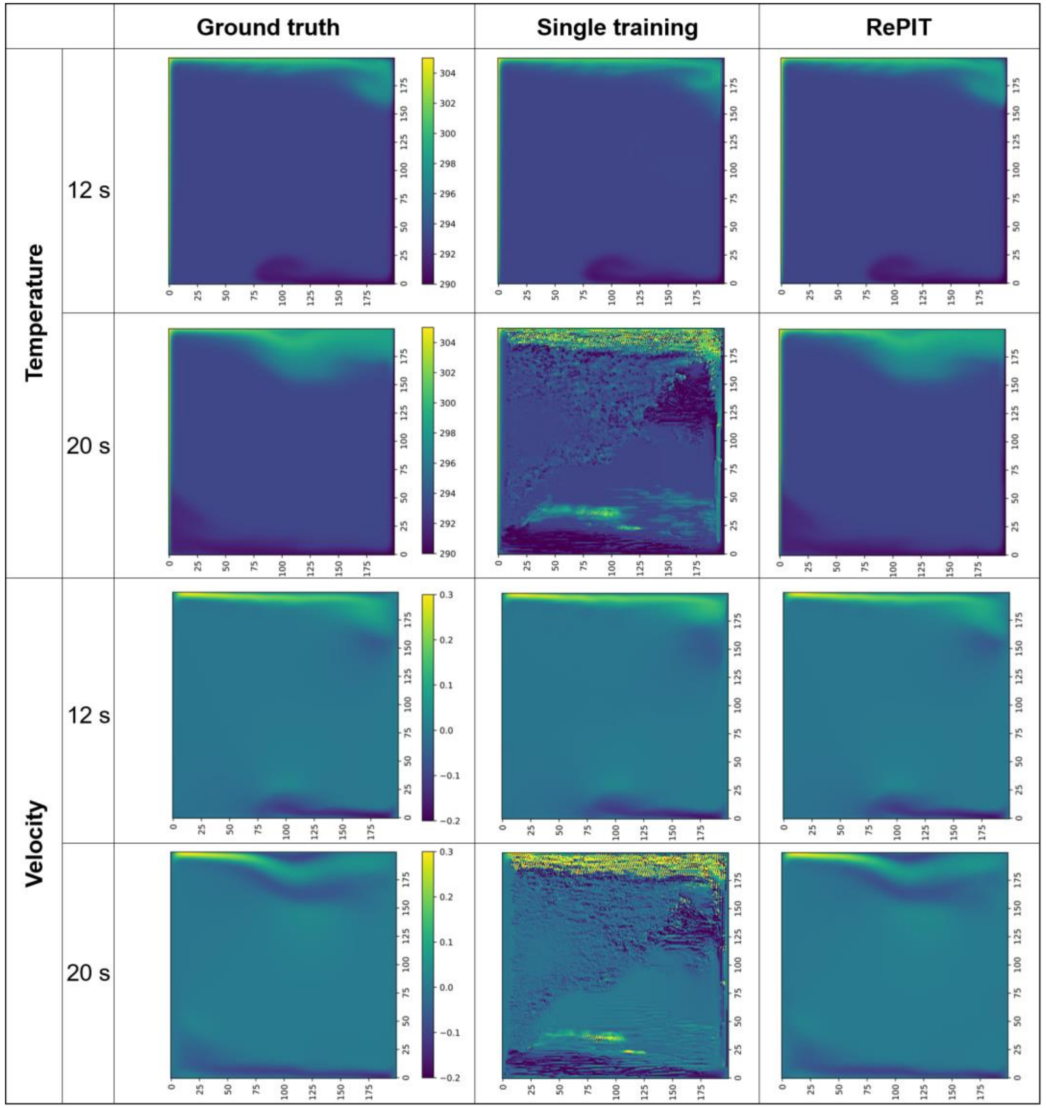




Results and conclusion

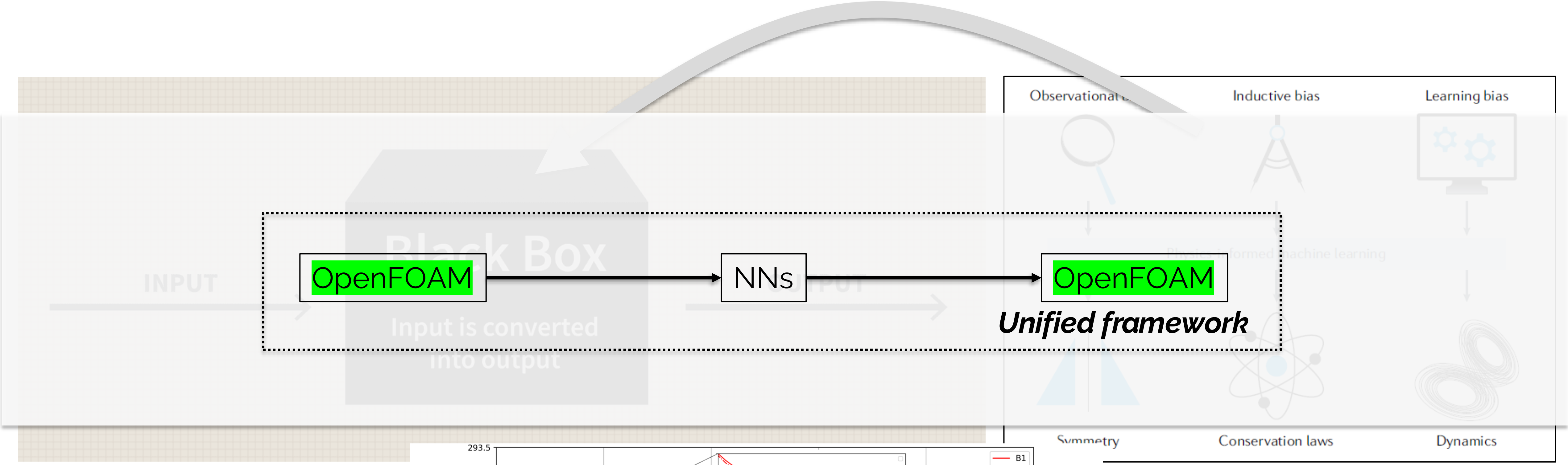
- Natural convection simulation by OpenFOAM
- laminar flow, buoyantPimpleFoam
- unified square grids (200x200).
- **x 11 acceleration** for 1 time series prediction

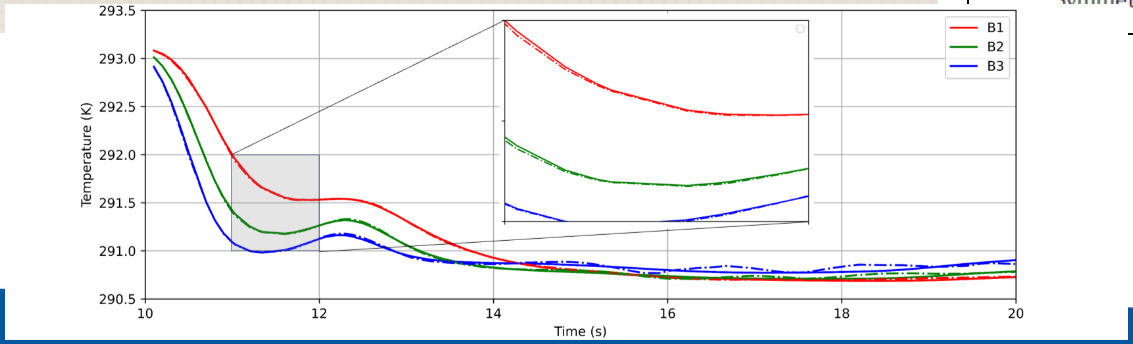




Summary and conclusions

“enlighten: to give knowledge or understanding”

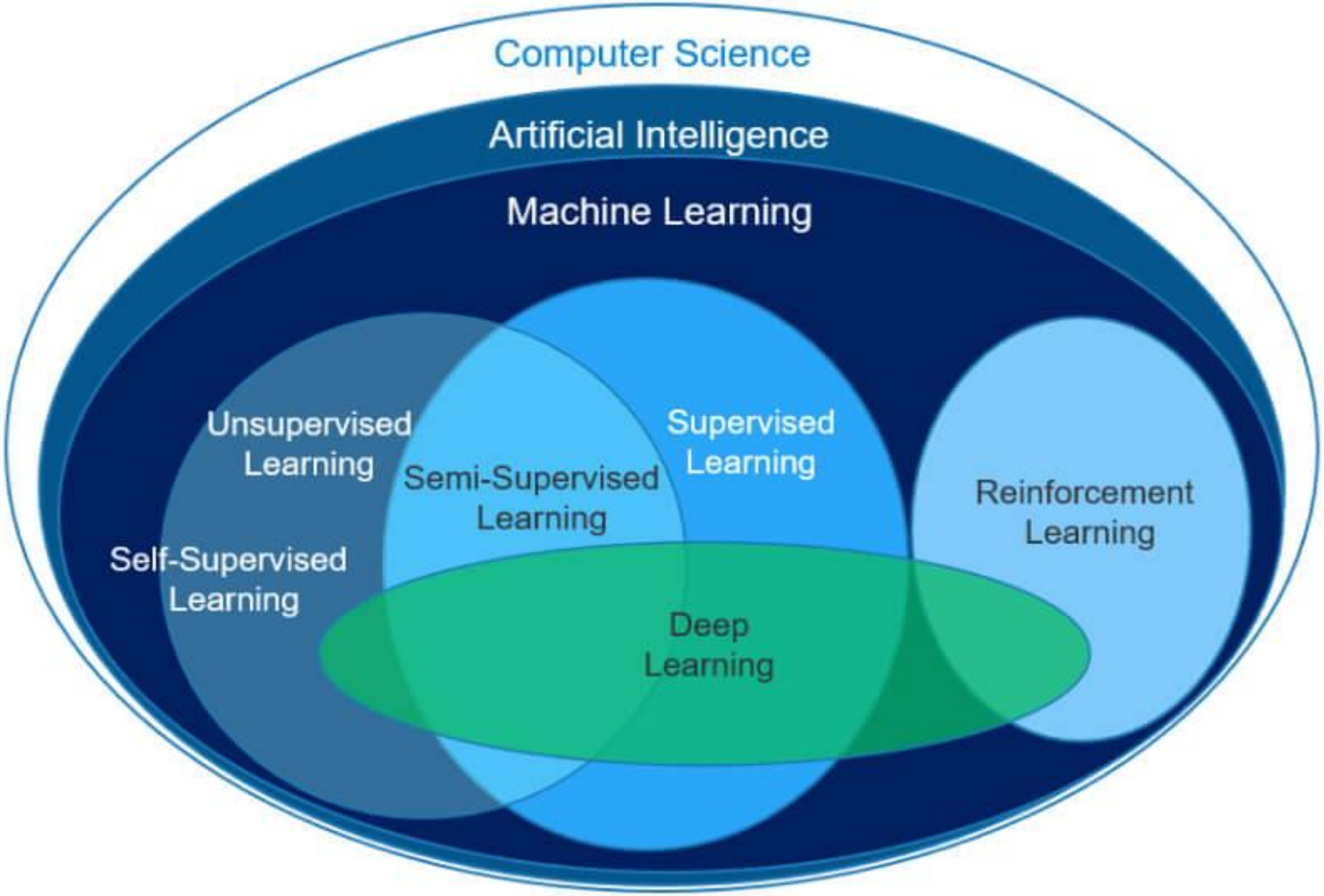




Thank you for listening!

jgjeon41@jbnu.ac.kr

What is artificial intelligence?



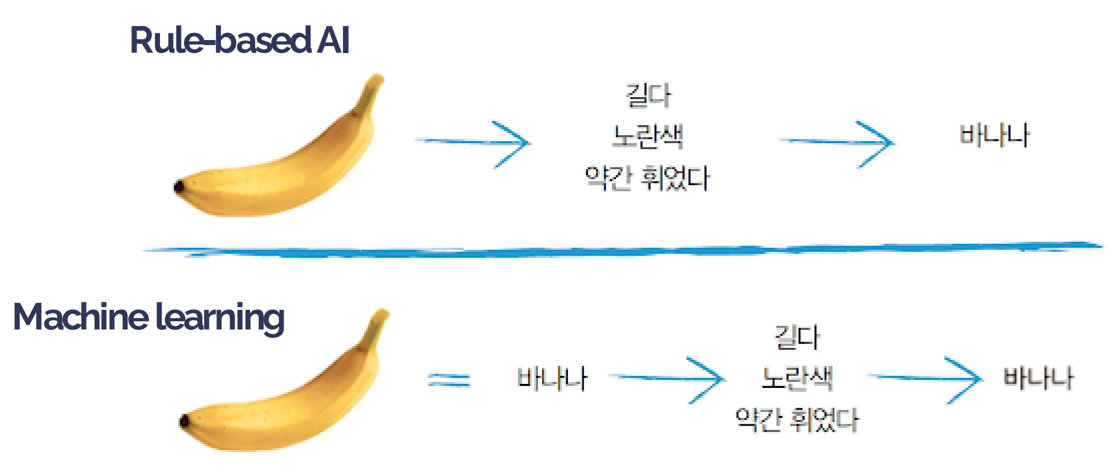
What is machine learning?

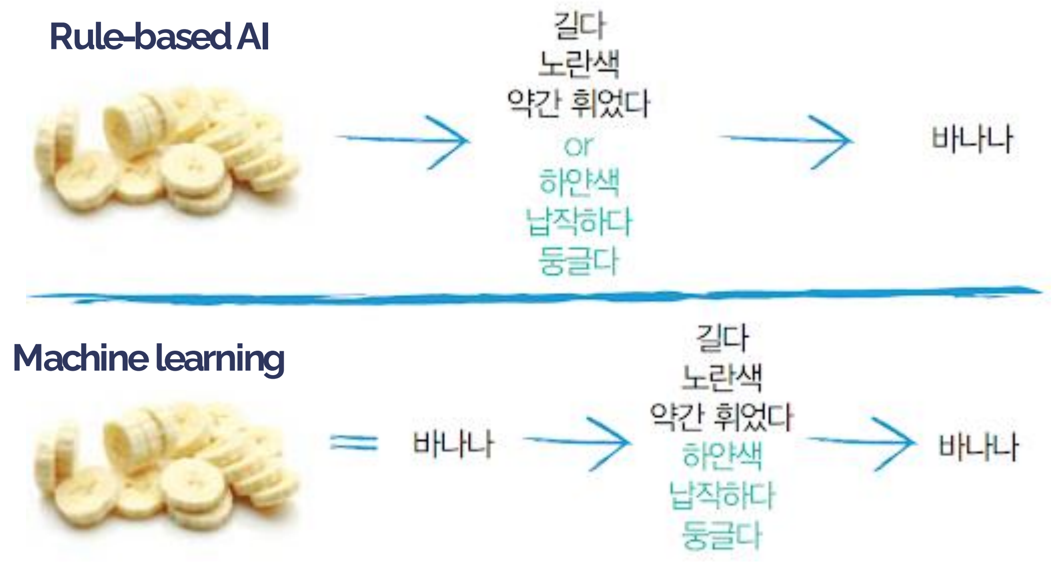
- artificial intelligence vs machine learning (neural networks)

  ✓ **'Artificial intelligence'**: the theory and development of **computer systems able to perform tasks** that normally require human intelligence, such as visual perception, speech recognition, decision-making, and translation between languages

  ✓ **'Machine learning':** the use and development of computer systems that **are able to learn and adapt without following explicit instructions**, by using algorithms and statistical models to analyze and draw inferences from patterns in data.

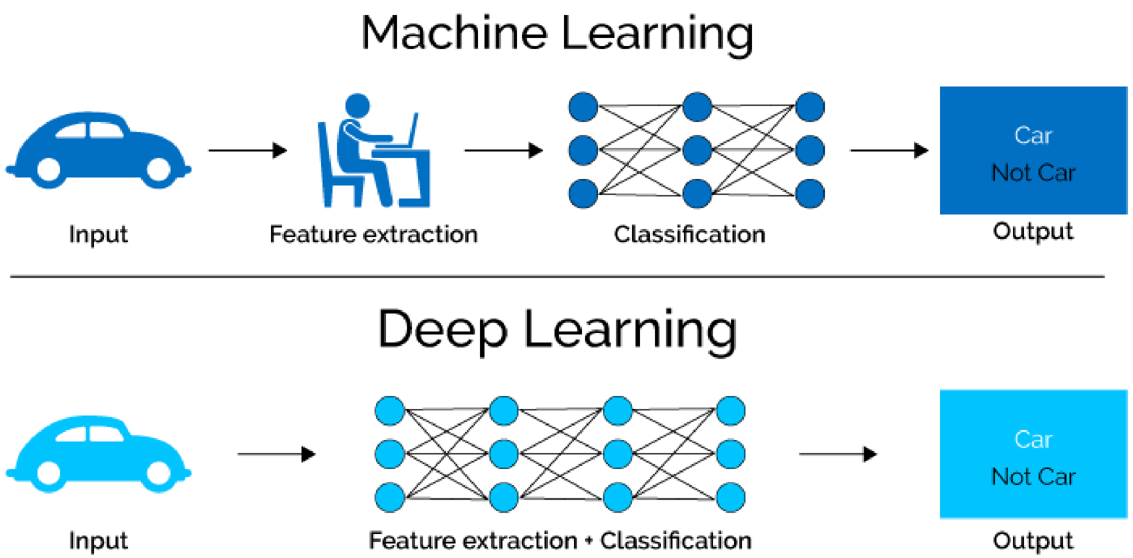
  ✓ **Simple example**

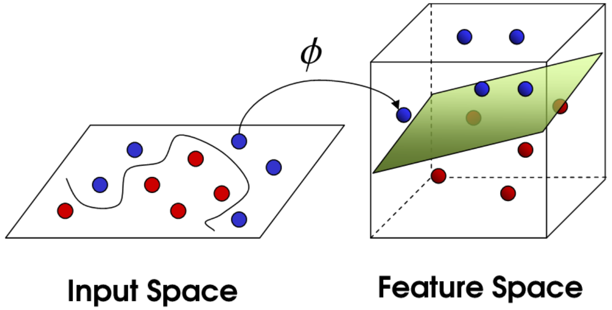


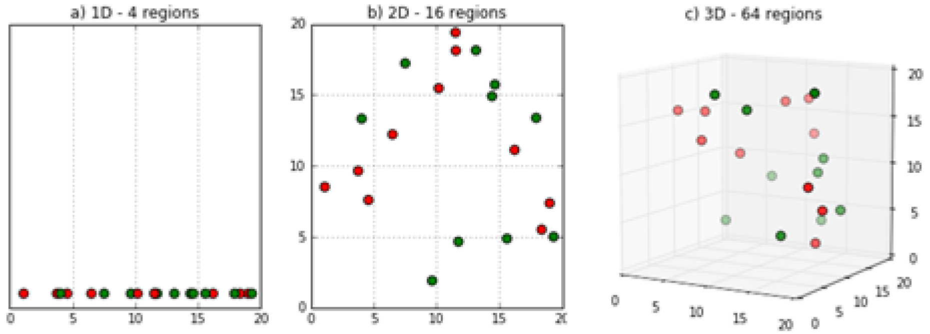


What is deep learning?

- machine learning vs deep learning







https://github.com/maziarraissi/PINNs/blob/master/main/continuous_time_inference%20(Schrodinger)/Schrodinger.py

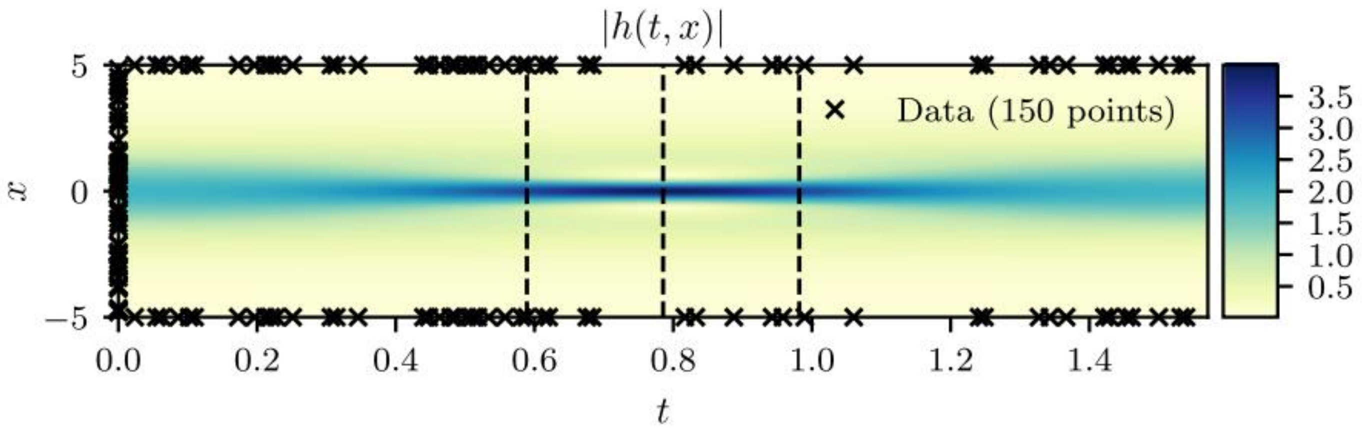
- Example: Shrodinger equation

$$ih_t + 0.5h_{xx} + |h|^2 h = 0, \quad x \in [-5, 5], \quad t \in [0, \pi/2],$$

$$h(0, x) = 2\ \mathrm{sech}(x),$$

$$h(t, -5) = h(t, 5),$$

$$h_x(t, -5) = h_x(t, 5),$$



$$MSE = MSE_0 + MSE_b + MSE_f,$$

where

$$MSE_0 = \frac{1}{N_0} \sum_{i=1}^{N_0} |h(0, x_0^i) - h_0^i|^2,$$

$$MSE_b = \frac{1}{N_b} \sum_{i=1}^{N_b} \left(|h^i(t_b^i, -5) - h^i(t_b^i, 5)|^2 + |h_x^i(t_b^i, -5) - h_x^i(t_b^i, 5)|^2 \right),$$

and

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

(Cai, 2022)

PINNs

- Issues in PINNs

  1. Long-time integration
  2. Complex problems
  3. Sampling methods
  4. Training dynamics

  and much more …

PINN vs FEM

	PINN	FEM
Basis function	NN (nonlinear)	Piecewise polynomial (linear)
Parameters	Weights and biases	Point values
Training points	Mesh-free	Mesh points
Governing equation	Loss function	Algebraic system
Parameter solver	Gradient-based optimization	Linear solver

(arXiv.1907.04502)

DeepONet

https://github.com/lululxvi/deeponet/blob/master/src/deeponet_pde.py

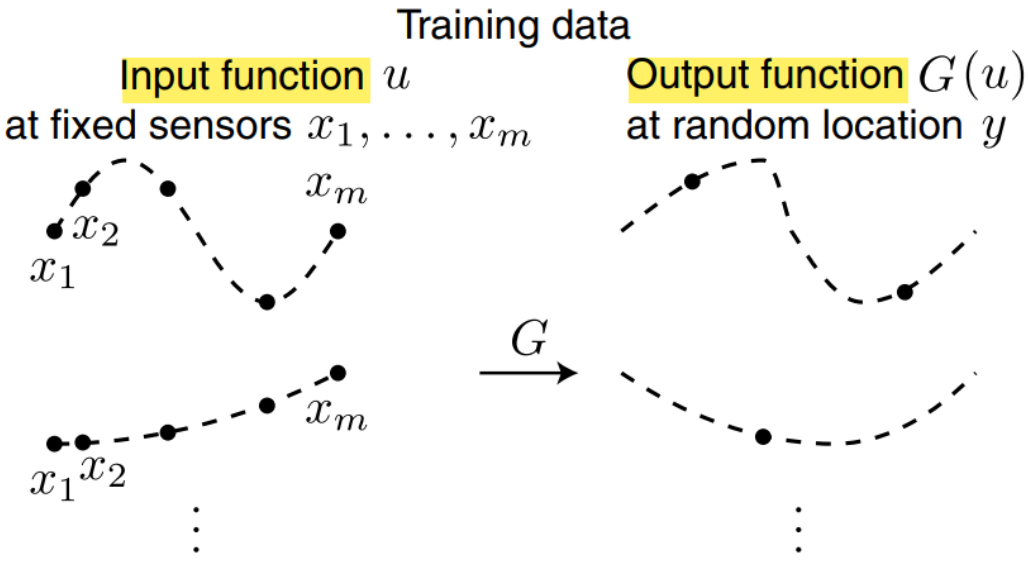
- Example: antiderivative operator and diffusion-reaction PDE

$$\frac{ds(x)}{dx} = u(x), s(x) = s_0 + \int_0^x u(\tau)\, d\tau,$$

$$\boldsymbol{G}: \boldsymbol{u}(\boldsymbol{x}) \to \boldsymbol{s}(\boldsymbol{x}),$$

$$\frac{\partial s}{\partial t} = D\frac{\partial^2 s}{\partial x^2} + ks^2 + u(x), x \in (0,1), t \in (0,1]$$

$$\boldsymbol{G}: \boldsymbol{u}(\boldsymbol{x}) \to \boldsymbol{s}(\boldsymbol{x}, \boldsymbol{t}),$$



input: $\boldsymbol{u}(\boldsymbol{x}), \boldsymbol{y}$

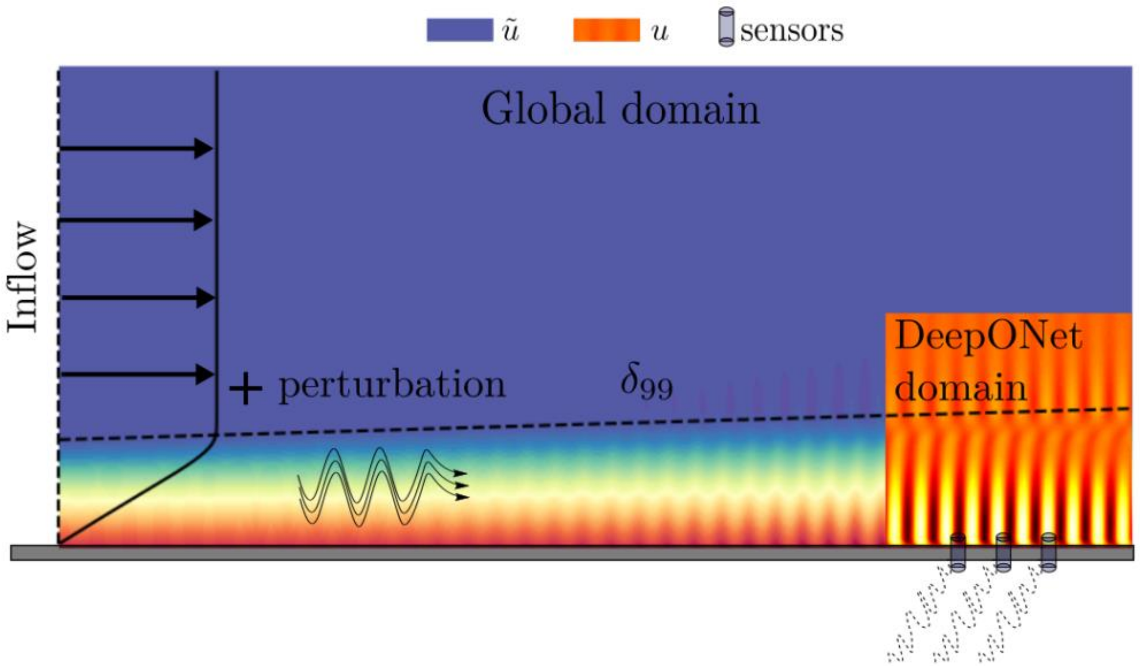
output: $\boldsymbol{G}(\boldsymbol{u})(\boldsymbol{y})$

DeepONet

https://github.com/lululxvi/deeponet/blob/master/src/deeponet_pde.py

- Example: Navier-stokes equation

  * For the ODEs and PDEs, the input function of the operators could be the boundary conditions, initial conditions or forcing terms (Lu, 2021).



Input: upstream disturbance
output: downstream perturbation field

PINN vs DeepONet

https://github.com/lululxvi/deeponet/blob/master/src/deeponet_pde.py

- **PINN**

  (+) easy to implement, applicable to various domains and equations

  (+) unsupervised learning

  (-) predict only a single PDE instance

  (-) hard to impose BCs

- **DeepONet**

  (+) predict multiple PDE instances

  (+) can use modern DNN architecture

  (-) supervised (in general), low accuracy on unseen data

  (-) hard to impose BCs

(Hong, 2023)