Neural Operators: Advancing Real-Time Structural Response

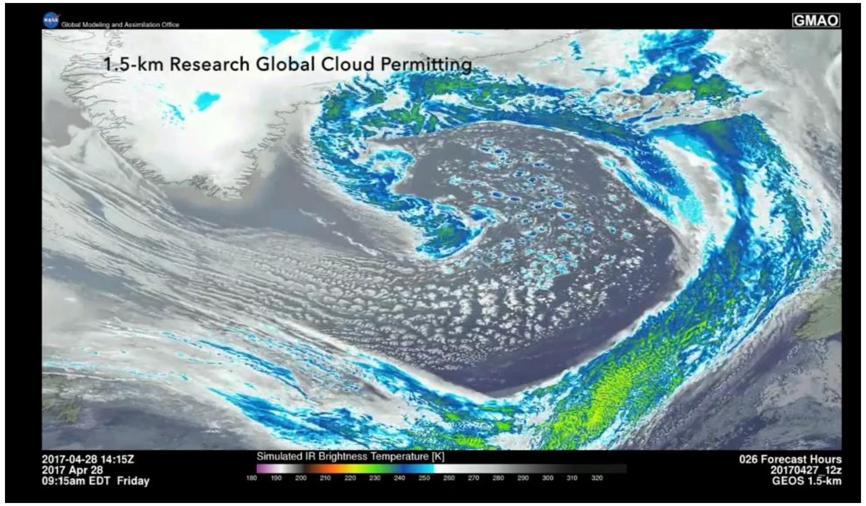
Somdatta Goswami

Assistant Professor, Civil and Systems Engineering

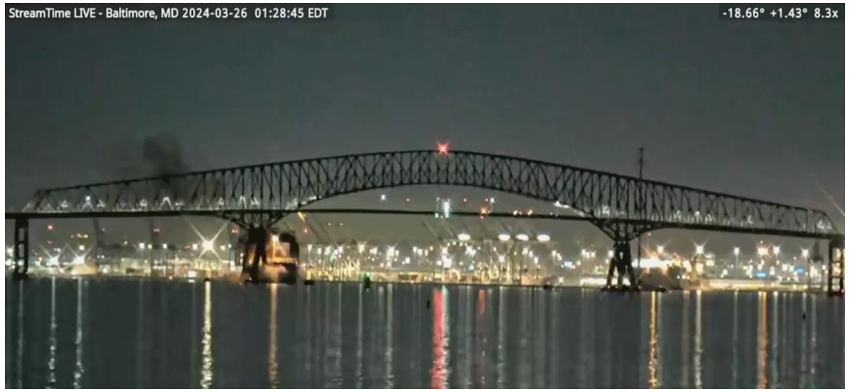
November 13, 2024

DesignSafe Training Webinar

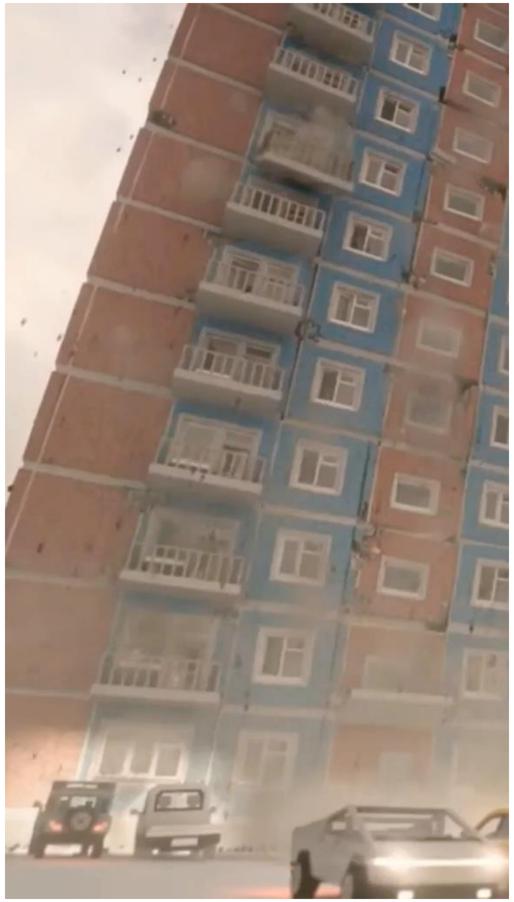
Processes of Nature



Evolution of an Icelandic Low in the North Atlantic Ocean over three days
Source: https://www.nas.nasa.gov/SC17/ (NASA)



Collapse of the Francis Scott Key Bridge in Baltimore on March 26, 2024
Source: https://www.youtube.com/watch?v=2kw1hH5XCYU



Bhuj Earthquake in India in 2001
https://www.youtube.com/shorts/SkNaxcxVz98

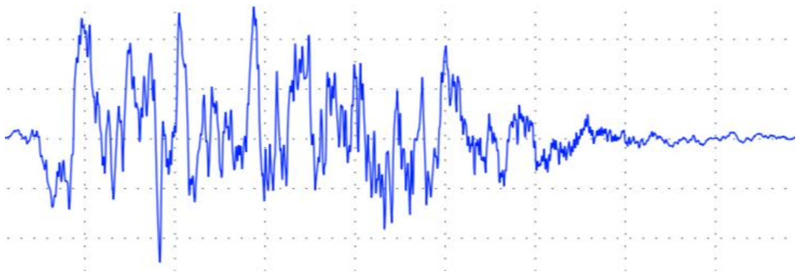
Physics-based Models

Can represent the **Processes of Nature**

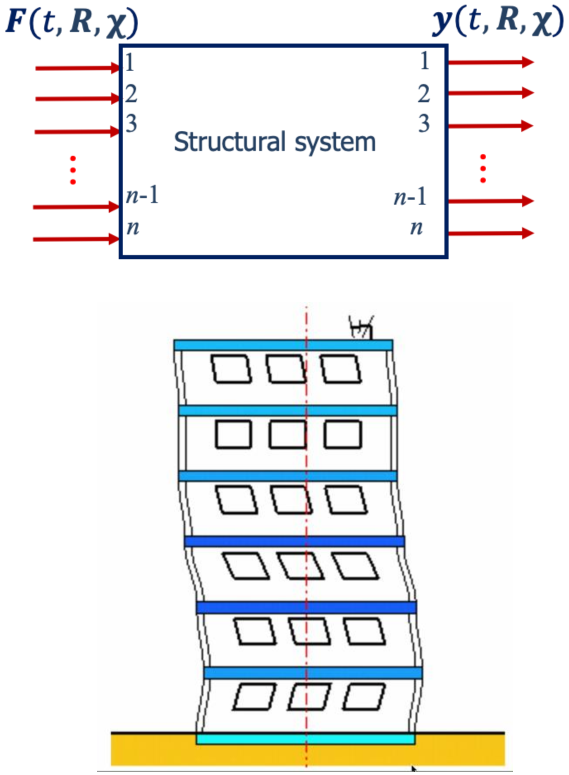
- Physics-based models are approximated via **ODEs/PDEs**

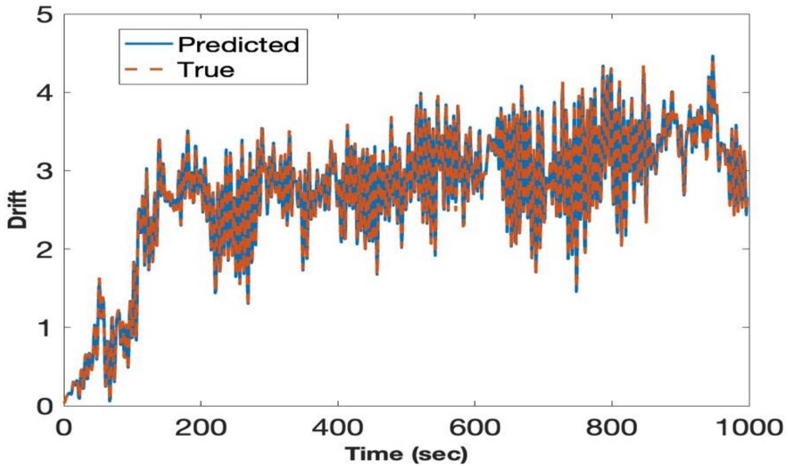
  To model earthquake: $m\frac{d^2u}{dt^2} + \mathrm{k}\frac{du}{dt} + F_0 = 0$

- Computational Mechanics helps us simulate these equations.



Input excitation





System response

Performance-based design

Extreme events

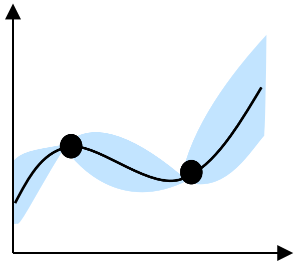
Non-linear Structural systems

Dynamic responses

Performance

State-of-the-art Performance assessment centered on **probabilistic** metrics

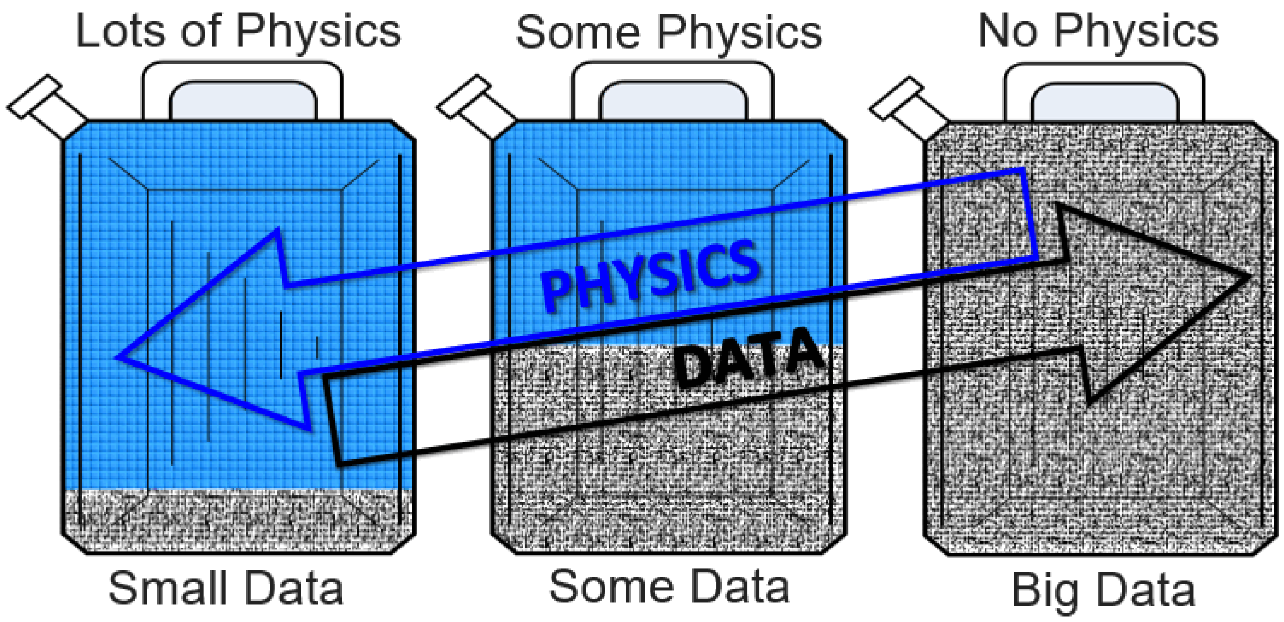
Surrogate Models



Computationally Demanding

Data + Law of physics

Three scenarios of
Physics-Informed Learning Machines



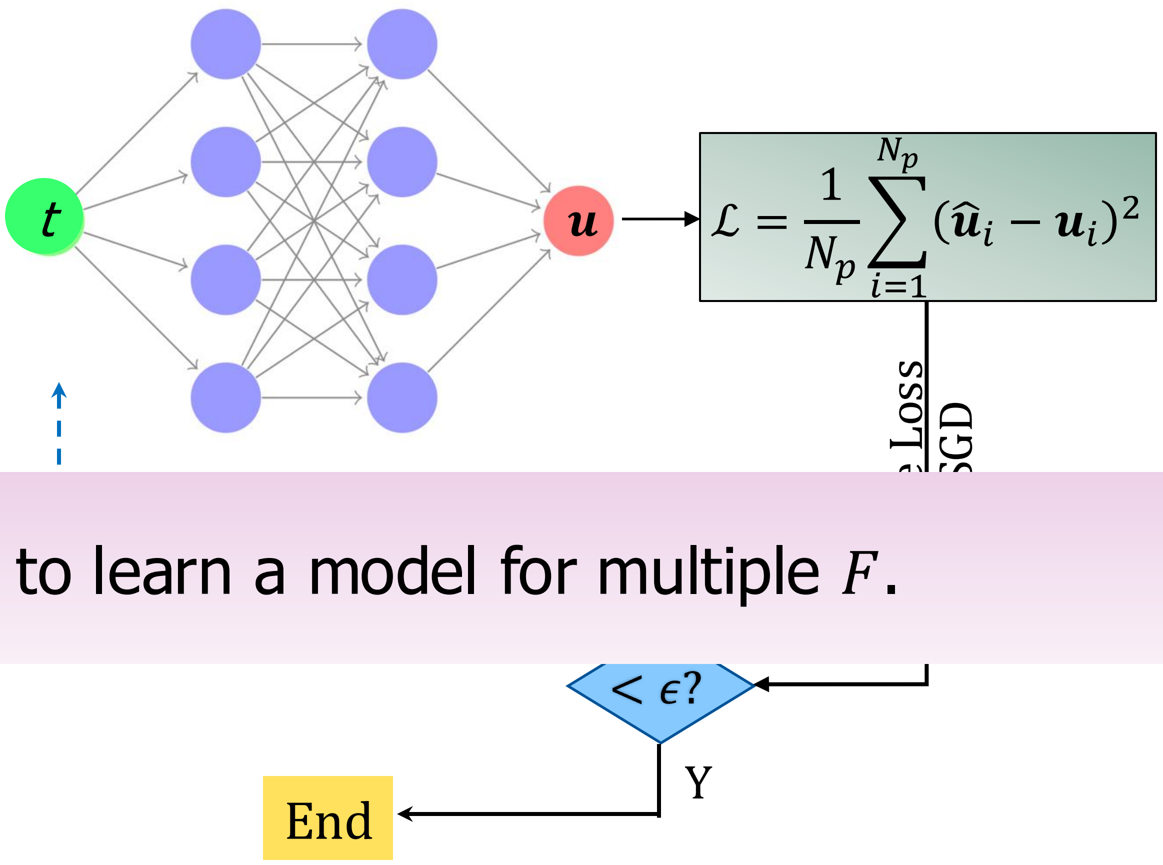
The 5-D Law: Dinky, Dirty, Dynamic, Deceptive Data

Training data-driven NNs

$$m\frac{d^2u(t)}{dt^2} + \mathrm{k}\frac{du(t)}{dt} + F = 0$$

$\mathcal{N}: t \rightarrow u(t)$ For a single F

$\hat{\boldsymbol{u}}$: Predicted solution
$\boldsymbol{\theta}$: Weights and biases
N_p: # of collocation points



Let's say we want to learn a model for multiple F.

Training data-driven NNs

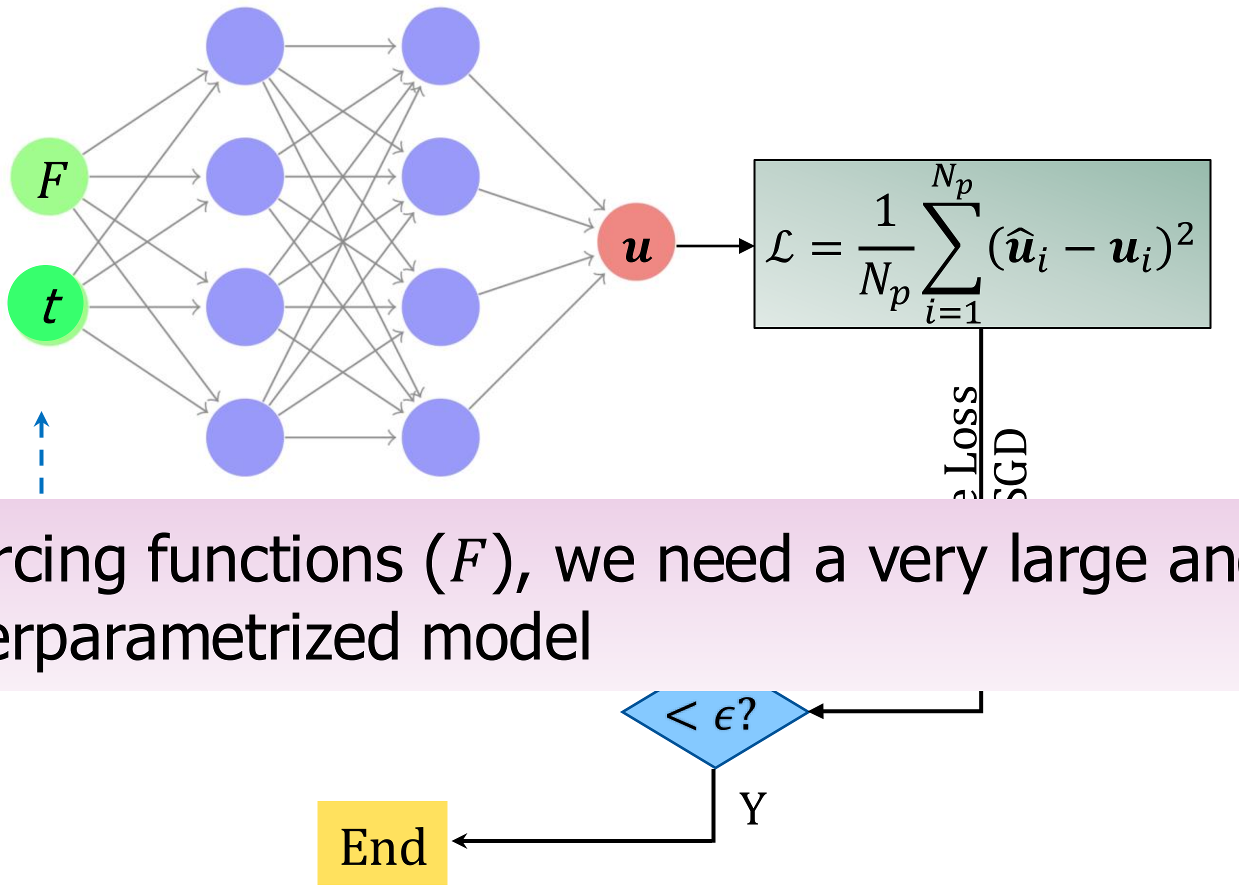
$$m\frac{d^2u}{dt^2} + \mathrm{k}\frac{du}{dt} + F = 0$$

$$\mathcal{N}: [t, F] \rightarrow u(t)$$

$\hat{\boldsymbol{u}}$: Predicted solution

$\boldsymbol{\theta}$: Weights and biases

N_p: # of collocation points



To learn a model for 1M Forcing functions (F), we need a very large and overparametrized model

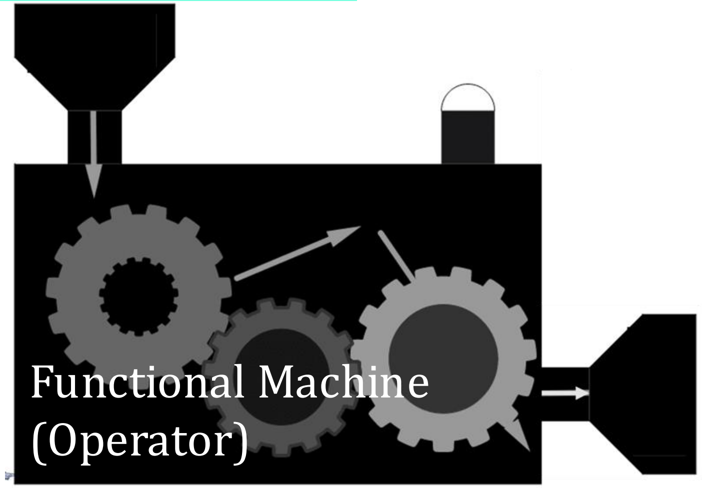
Learning a Neural Operator

Inputs

$\mathcal{F} = \{F_1, F_2, \ldots, F_n\}$

$\Phi : \mathcal{F} \to \mathcal{U}$

$\mathcal{F}, \mathcal{S}$ are infinite dimensional function space



Outputs

$\mathcal{U} = \{u_1(\boldsymbol{F_1}), u_2(\boldsymbol{F_2}), \ldots, u_n(\boldsymbol{F_n})\}$

What we know

$\{\mathcal{F}_n, \mathcal{U}_n\}_{n=1}^{N}$

$\mathcal{U}_n = \Phi(\mathcal{F}_n)\,, \mathcal{F}_n \sim \mu \; i.i.d$

Task

Operator, $\Psi : \mathcal{F} \times \Theta \to \mathcal{U}$ such that $\Psi(., \theta^*) \approx \Phi$

Training $\theta^* = \mathrm{argmin}_\theta l(\{\mathcal{F}_n, \Psi(\mathcal{U}_n, \theta)\})$

Surrogate Modeling Techniques

- Discretized Data
- Discretization dependent
- Queries on mesh
- Learning functions between vector spaces

- Functional Data
- Discretization Invariant
- Continuous quantities
- Learning operators between function spaces

Data-driven

PCA **Auto-encoders**

K-PCA **Diffusion maps**

f-PCA **DeepONet** **LNO**

FNO **WNO**

F-RKHS

Finite Dimensional

Infinite Dimensional

Physics Informed

PINNs*

PI-DeepONet

PINO

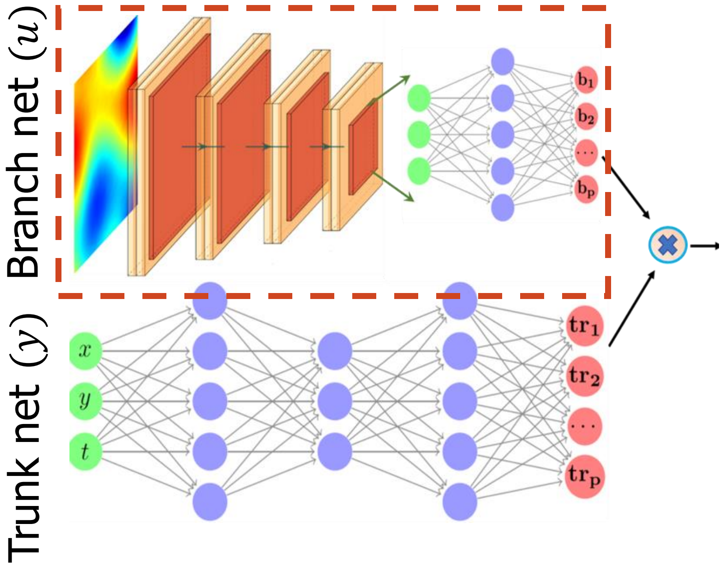
*continuous output

Deep Operator Networks (DeepONet)

- Generalized Universal Approximation Theorem for Operator [Chen '95, Lu et al. '19]
- **Branch net**: Input $\{u(x_i)\}_{i=1}^{m}$, output: $\left[b_1, b_2, .., b_p\right]^T \in \mathbb{R}^p$
- **Trunk net**: Input y, output: $\left[t_1, t_2, .., t_p\right]^T \in \mathbb{R}^p$
- Input u is evaluated at the fixed locations $\{y_i\}_{i=1}^{m}$

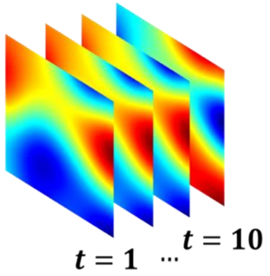
$$G_\theta(u)(y) = \sum_{i=1}^{p} \underbrace{b_i(u(x_1), u\,(x_2), \dots, u\,(x_m))}_{\text{branch net}} \cdot \underbrace{tr_i(y)}_{\text{trunk net}}$$

Can be FNN/CNN/U-Net/LSTM etc.



Minimize loss

$$G_\theta(u)(y) \downarrow \mathcal{L}_r(\theta) + \mathcal{L}_i(\theta) \downarrow \theta^*$$



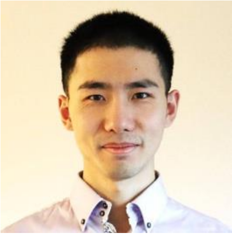
Other neural operators

- Fourier neural operator
- Wavelet neural operator
- Laplace neural operator

[1] Kovachki, N. et al. (2021), [2] Li, Z. et al. (2020), [3] Tripura, T. and Chakraborty, S. (2022), [4] Cao, Q. and Goswami S. (2023).

Manuscript in preparation:

Neural operators for stochastic modeling of system response to natural hazards

A collaboration between Johns Hopkins University, University of Texas Tech University and University of Michigan.

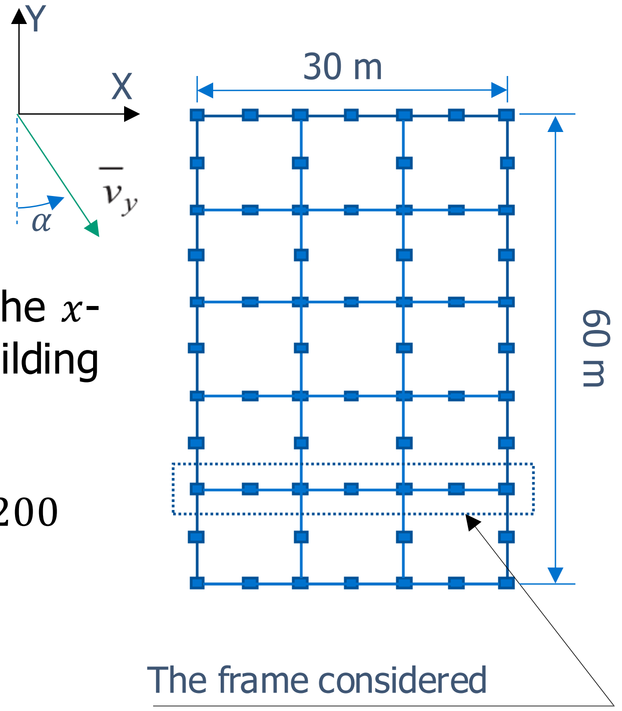
NO for wind response approximation

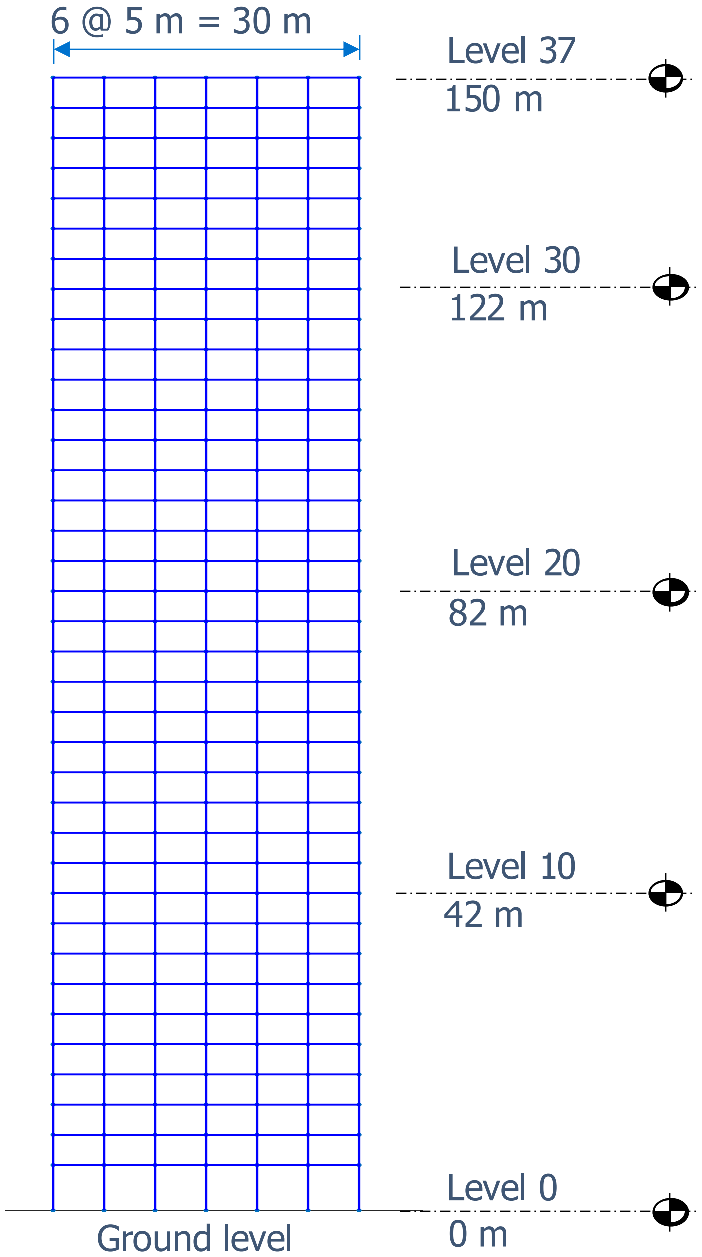
Structural system:
- 37-story 3D steel moment resisting frame;
- Elastic-perfectly plastic material model;
- Wind direction: α = 90°;
- Wind speed: $\bar{v}_y = 65$ m/s at the building top;
- Wind load motion: Wind tunnel data informed stochastic wind load model;

OBJECTIVE

Use DeepONet to map the wind excitation to the x-displacement of the right-most node of the building

Labelled dataset: 1000: $N_{train} = 800, N_{test} = 200$





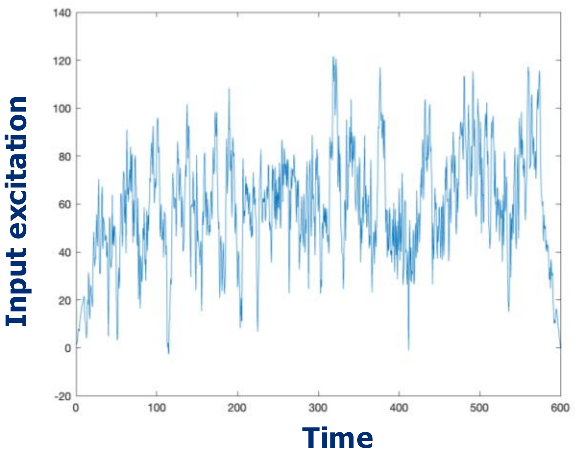
NO for wind response approximation

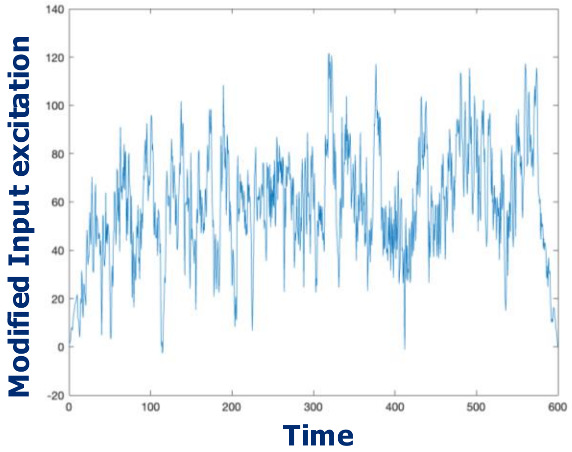
Data

- H_i : Time history of excitation at i^{th} floor
- Δt^H : 0.5s
- T : 600 s
- R_i : Displacement at i^{th} floor (R_i^x, R_i^y, R_i^z)
- Δt^R : 0.02s
- $\mathcal{G}: [H_1, H_2, \ldots, H_{37}] \rightarrow R_{37}$

Preprocessing

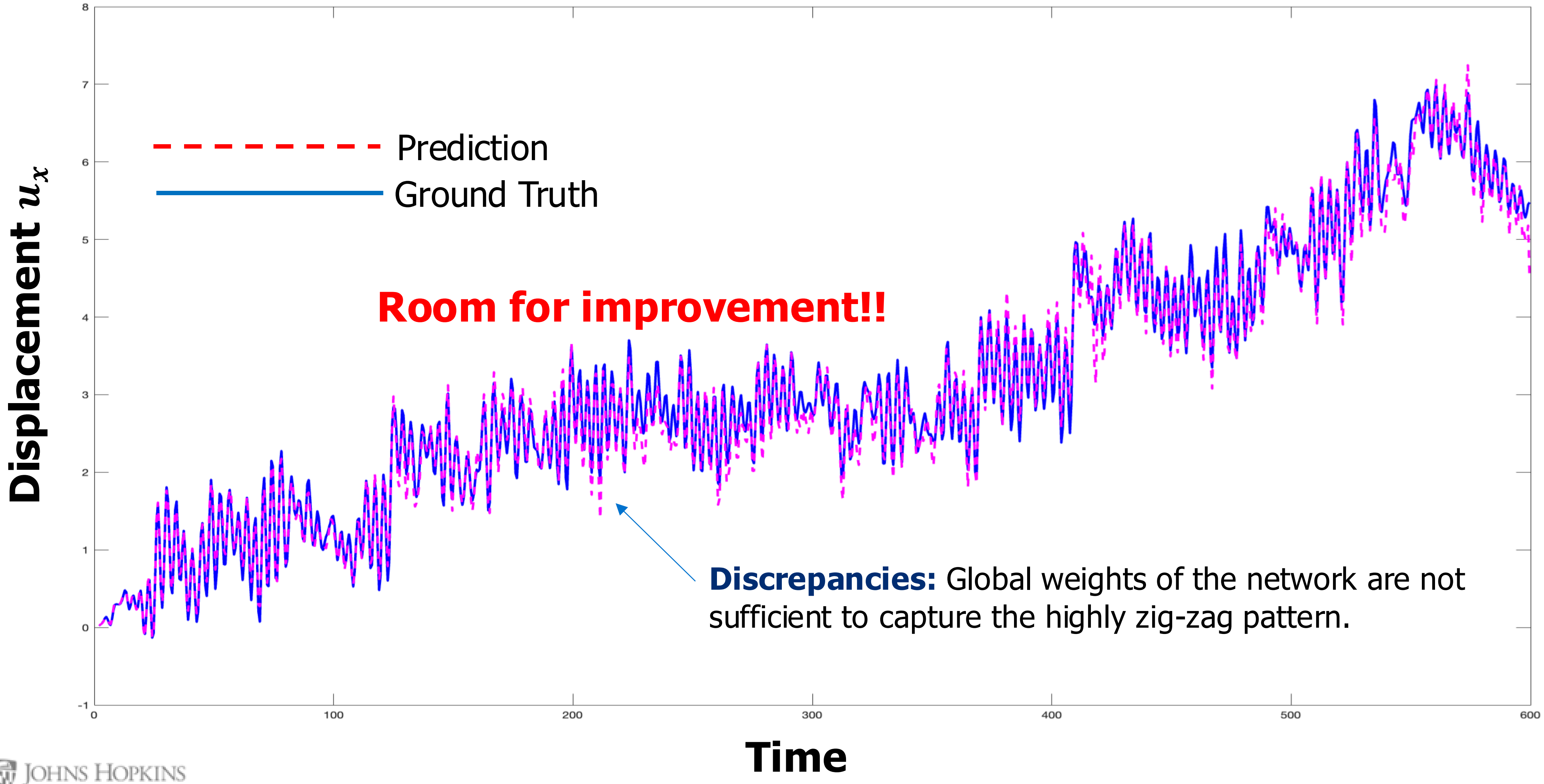
- Input excitation frequency $(\mathrm{H_{freq}})$ $\neq$ Response frequency $(\mathrm{R_{freq}})$
- **Option 1**: Filtering the response and reducing its frequency to $\mathrm{H_{freq}}$
  - Loss of information
- **Option 2:** Interpolation of $\mathrm{H_i}$ on the time discretization of $\mathrm{R_i}$
  - Taking time delay of 20s to remove the initial zeros





JOHNS HOPKINS
WHITING SCHOOL of ENGINEERING

NO for wind response approximation



NO for wind response approximation

Solution 1: DeepOnet with Self-adaptive Weights

- Introduce **a weight** at every time stamp.
- Such weights can be manually modulated. For the given problem size – impractical.
- Hence, adaptive weights to automatically update along with the network parameters can be used.

Original loss function:

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(u_i(\theta) - \mathcal{G}_\theta(\mathbf{v}_i)(\theta)\right)^2$$

Weighted loss function:

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N} w_j\left(u_i(\theta) - \mathcal{G}_\theta(\mathbf{v}_i)(\theta)\right)^2 \Rightarrow$$

$$\mathcal{L}(\theta, \boldsymbol{\lambda}) = \sum_{i=1}^{N} g(\boldsymbol{\lambda})\left(u_i(\theta) - \mathcal{G}_\theta(\mathbf{v}_i)(\theta)\right)^2$$

Masking function

NO for wind response approximation

DeepOnet with Self-adaptive Weights

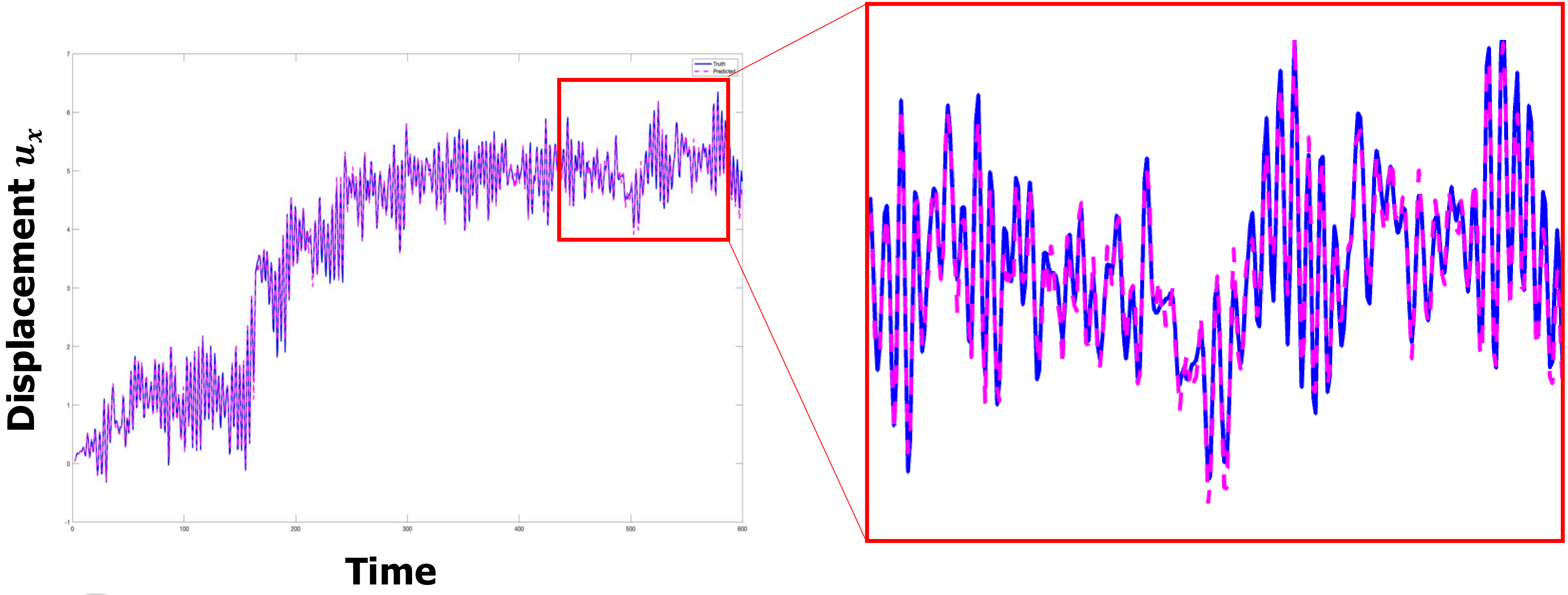
Optimization problem: $$\min_{\theta}\max_{\boldsymbol{\lambda}}\mathcal{L}(\theta,\boldsymbol{\lambda})$$

Updating the masking function: $$\boldsymbol{\lambda}^{\mathrm{k}+1}=\boldsymbol{\lambda}^{\mathrm{k}}+\eta_{\boldsymbol{\lambda}}\nabla_{\boldsymbol{\lambda}}\mathcal{L}(\theta,\boldsymbol{\lambda})$$ **Gradient ascent**

Constraints on the masking function: $$g(\boldsymbol{\lambda}_i)>\mathbf{0}$$

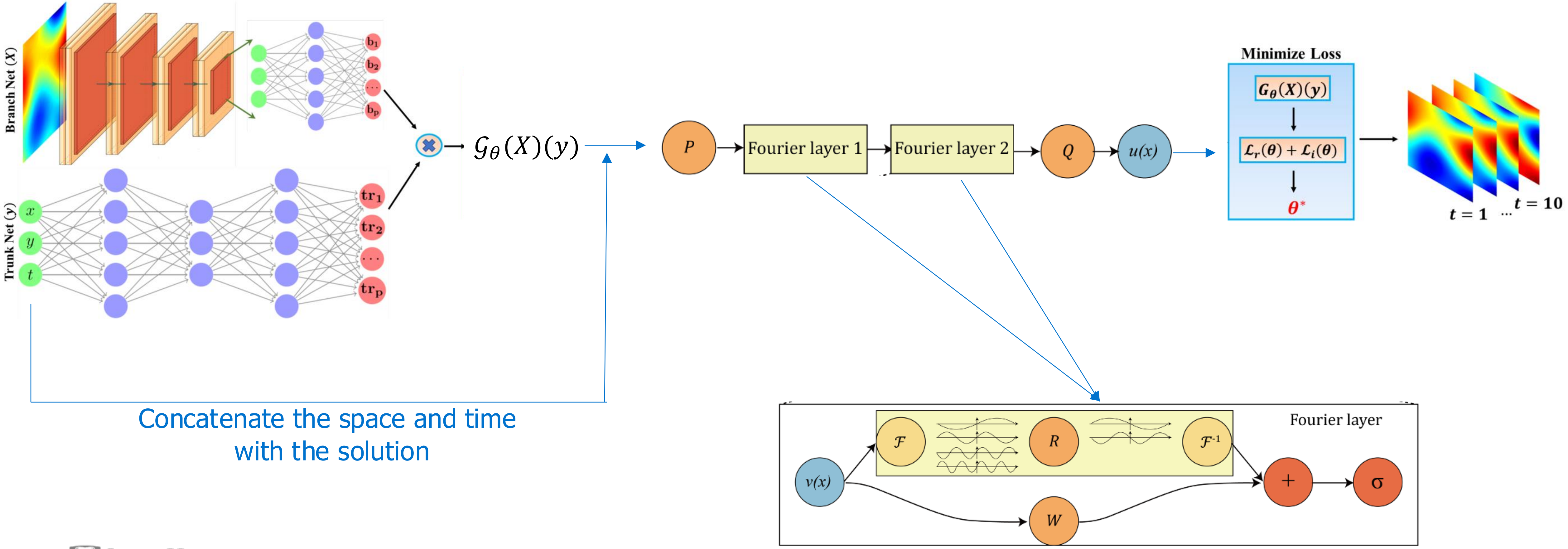
NO for wind response approximation

DeepOnet with Self-adaptive Weights



NO for wind response approximation

Solution 2: DeepOnet integrated with FNO



NO for wind response approximation

Relative L_2 error (%)

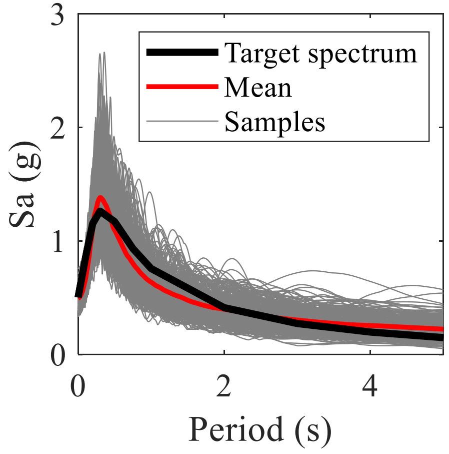
Method		u_x	u_y	u_z
DeepONet in time	Vanilla DeepONet	13.12	14.89	13.11
	Self-adaptive DeepONet	7.87	7.89	7.89
DeepONet in space and time	Vanilla DeepONet	15.92	15.66	15.12
	Self-adaptive DeepONet	7.91	7.56	7.62
	Integrated framework	1.12	1.34	1.76

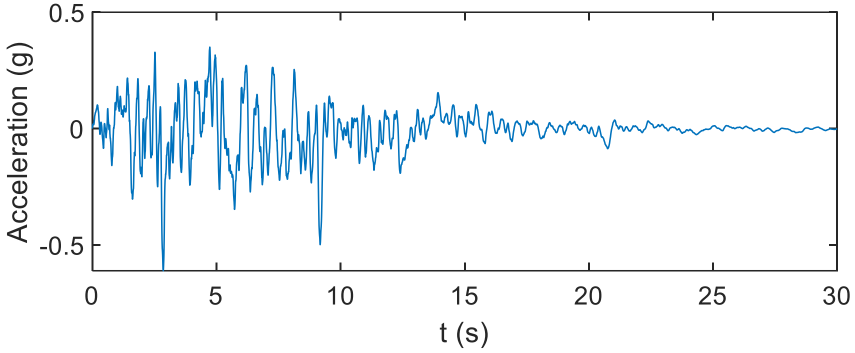
DeepONet in time: Mapping response to the location of the highest interstorey drift
DeepONet in space and time: Mapping response to the nodes on the right surface of the building for all 37 floors.

NO for earthquake response approximation

Structural system:

- 6-story nonlinear shear building
- Ground motion model: Filtered white noise;
- Target spectrum: Generated using USGS tool at Loma Prieta with 10% exceedance in 50 years.

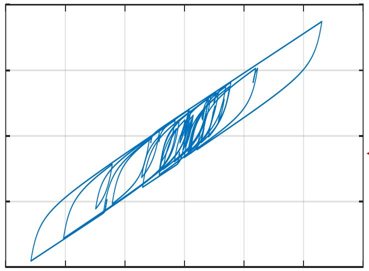




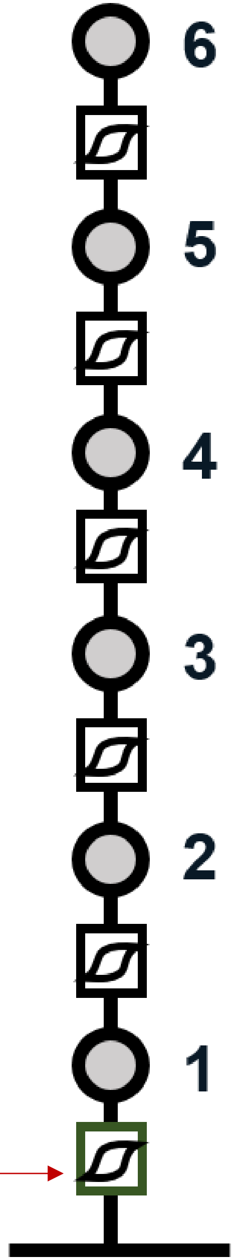
OBJECTIVE

Use FNO to map the ground motion to the acceleration at the 6 floors

Labelled dataset: 1000: $N_{train} = 800, N_{test} = 200$

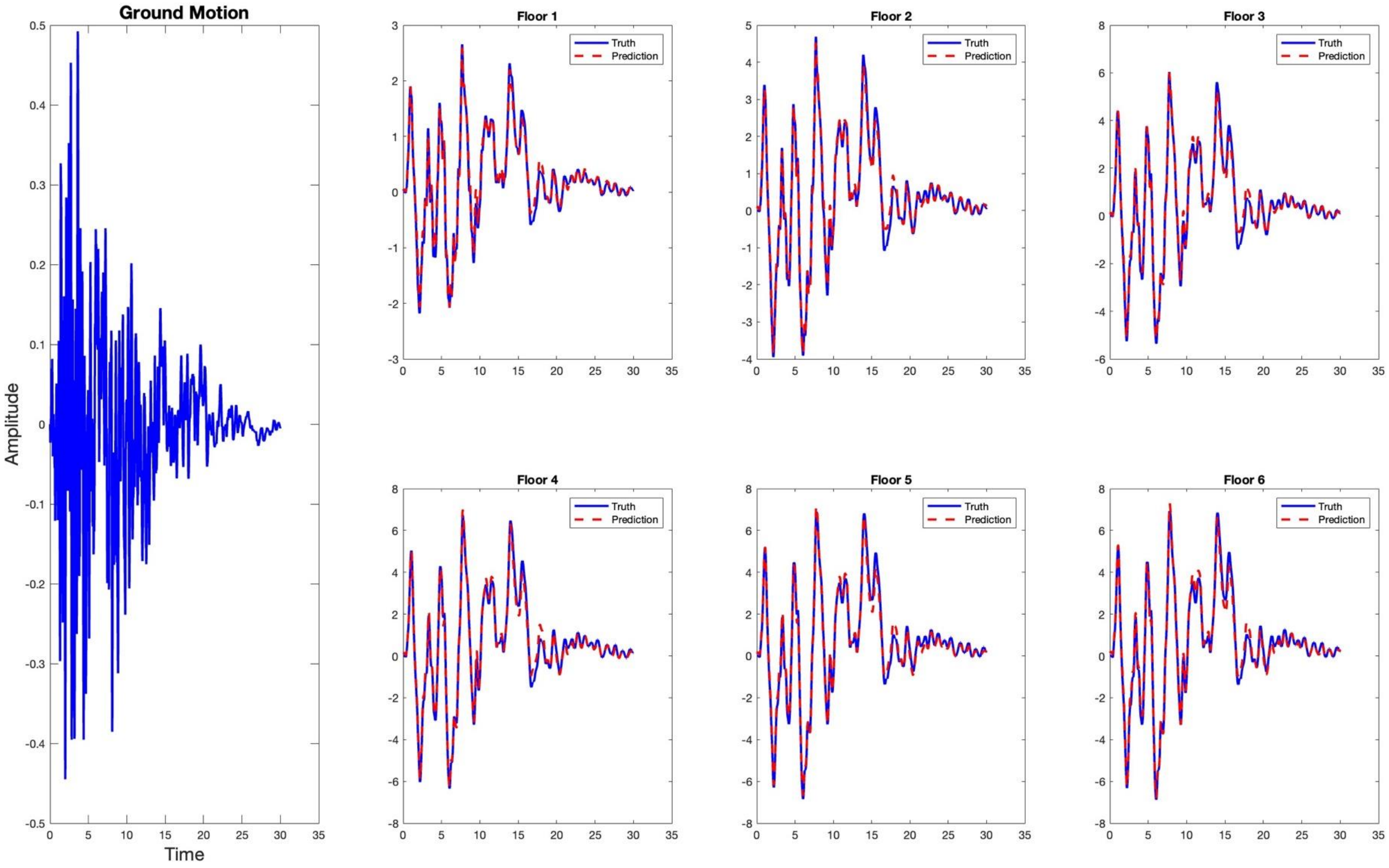


Steel 02 Material



Johns Hopkins Whiting School of Engineering

NO for earthquake response approximation



Relative L_2 error (%)

Method		Response
FNO 1D	Vanilla FNO	8.1
	SA - FNO	4.23

Hands-on training

Let us learn to construct a DeepONet for modeling the response of a cantilever beam under applied displacement

Training of DeepONet

$$m\frac{d^2u}{dt^2} + \mathrm{k}\frac{du}{dt} + F = 0$$

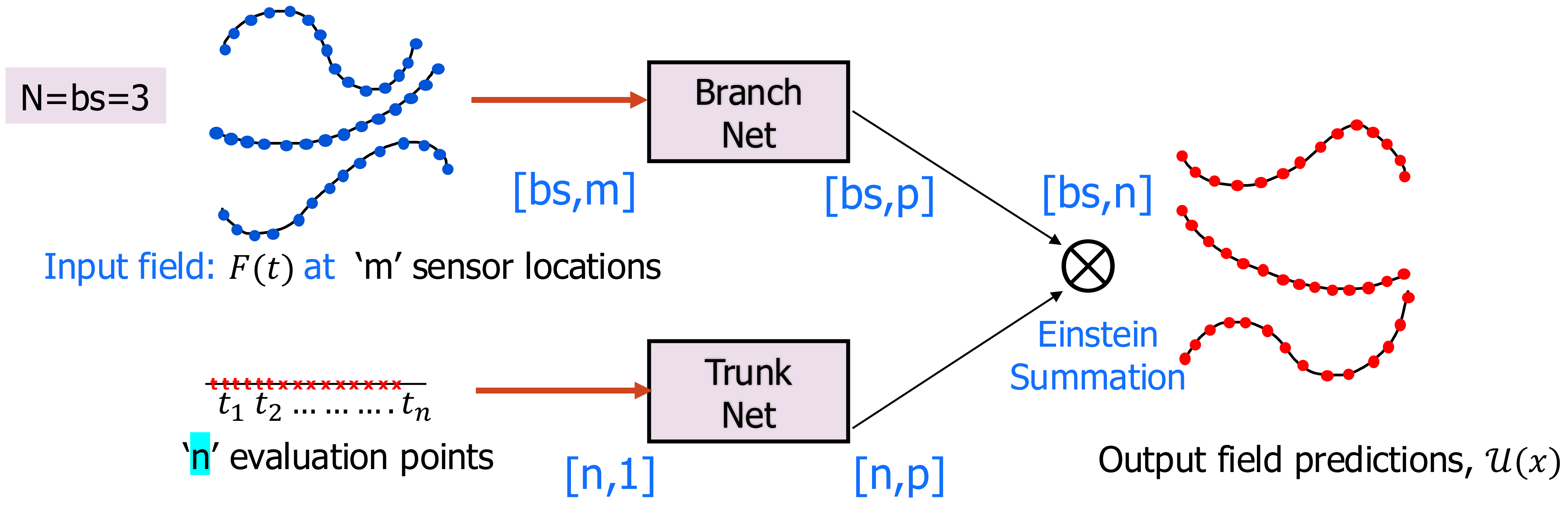
Nonlinear operator $\mathcal{G} : \mathcal{F} \to \mathcal{U}$

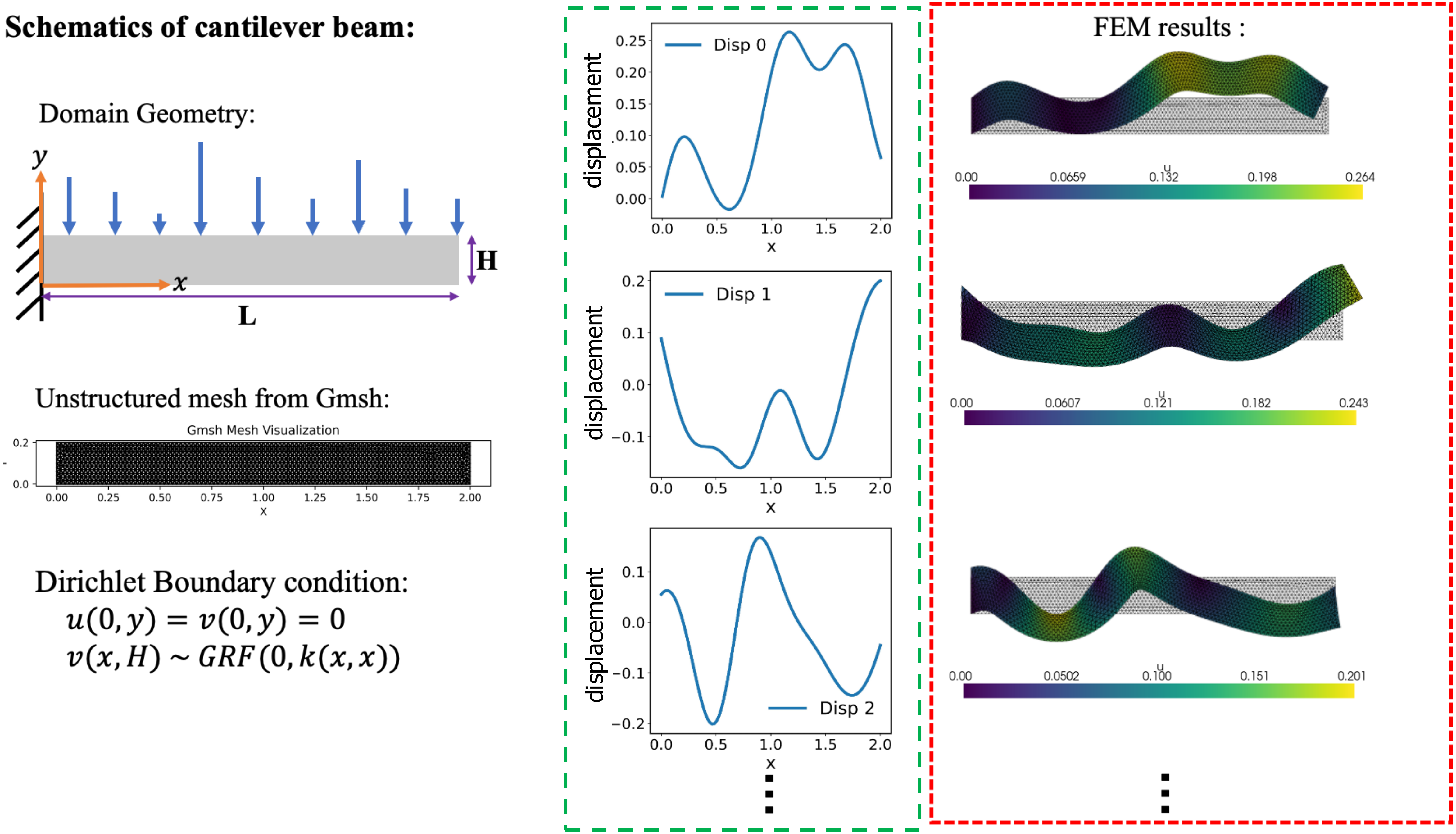
Neural operator $\mathcal{G}_\theta : \mathcal{F} \to \mathcal{U}, \quad \theta \in \Theta$

Training data $\{F^i(t_k), u^i(t_j)\}_{i=1,j=1,k=1}^{N,n,m}$

Training Dataset

S.No	Input field data	Output field data
1	$F^1(t_1), F^1(t_2), \cdots\cdots\cdots, F^1(t_m)$	$u^1(t_1), u^1(t_2), \cdots\cdots\cdots, u^1(t_n)$
2	$F^2(t_1), F^2(t_2), \cdots\cdots\cdots, F^2(t_m)$	$u^2(t_1), u^2(t_2), \cdots\cdots\cdots, u^2(t_n)$
.	.	.
N	$F^N(t_1), F^N(t_2), \cdots\cdots\cdots, F^N(t_m)$	$u^N(t_1), u^N(t_2), \cdots\cdots\cdots, u^N(t_n)$





Schematics of cantilever beam:
Domain Geometry:
y
x
H
L
Unstructured mesh from Gmsh:
Gmsh Mesh Visualization
Dirichlet Boundary condition:
$u(0, y) = v(0, y) = 0$
$v(x, H) \sim GRF(0, k(x, x))$
Disp 0
Disp 1
Disp 2
displacement
x
FEM results :
u
0.00 0.0659 0.132 0.198 0.264
0.00 0.0607 0.121 0.182 0.243
0.00 0.0502 0.100 0.151 0.201

The codes and the associated details are available on:
https://designsafe-training.github.io/deeponet/deeponet.html
and
https://github.com/DesignSafe-Training/deeponet

Centrum IntelliPhysics






Thank you!