AI in the Sciences and Engineering HS 2025: Lecture 9

Siddhartha Mishra

Computational and Applied Mathematics Laboratory (CamLab)
Seminar for Applied Mathematics (SAM), D-MATH (and),
ETH AI Center (and) Swiss National AI Institute (SNAI) ,
ETH Zürich, Switzerland.

What we have learnt so far ?

- AIM: Learn PDEs using Deep Neural Networks
- Operator Learning: Learn the PDE Solution Operator from data.
- Examples: FNO, CNO, VIT, scOT etc.
- Very successful on PDEs on 2D Cartesian Domains !!
- What are the caveats ?

What about Time-Dependent PDEs ?

Operator Learning for Time-Dependent PDEs

- PDEs of the Abstract form:

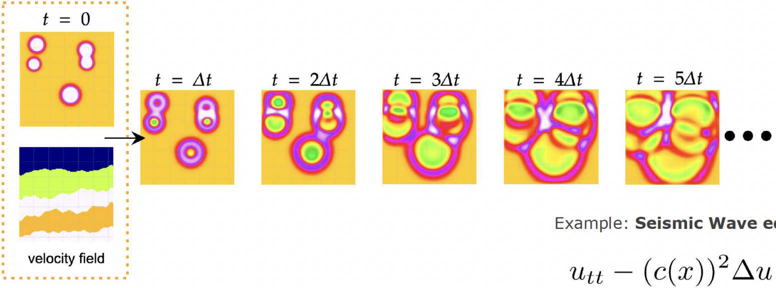
$$u_t + \mathcal{L}(t, x, u) = 0, \quad u(0) = \bar{u}.$$

- Solution operator: $\mathcal{S} : (0, T) \times \mathcal{X} \mapsto \mathcal{X}$; $\mathcal{S}(t, \bar{u}) = u(t)$
- Fo any time increment: $\mathcal{S}(\Delta t, u(t)) = u(t + \Delta t)$.
- Data is the form of Trajectories:

$$(u(0), u(t_1), u(t_2), \dots, u(T)) = (\bar{u}, \mathcal{S}(t_1, \bar{u}), \mathcal{S}(t_2, \bar{u}), \dots, u(T))$$
$$= (\bar{u}, \mathcal{S}(t_1, \bar{u}), \mathcal{S}(t_2 - t_1, u(t_1)), \dots, u(T))$$

- Operator Learning Task: Continuous-in-Time evaluations !!
- Given $\bar{u}$ + BC: generate the solution trajectory $u(t)$, for all $t \in (0, T]$

Operator Learning for Time-dependent PDEs



Example: **Seismic Wave equation**

$$u_{tt} - (c(x))^2 \Delta u = 0$$

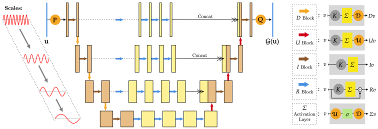
Autoregressive Evaluation

- Trajectory data on uniform timepoints: $u(t_k) = u(k\Delta t)$.
- Define $\mathrm{NO}_{\Delta t}(u(t_\ell)) \approx u(t_\ell + \Delta t)$: Next Time Prediction
- Then Autoregressive Rollout is

$$u(t_k) \approx \underbrace{\mathrm{NO}_{\Delta t} \circ \ldots \mathrm{NO}_{\Delta t} \circ \mathrm{NO}_{\Delta t}}_{k \text{ times}} \bar{u}.$$

- Issues:
  - Needs uniform spacing.
  - Long rollouts lead to training issues.
  - Error Accumulation
  - Only evaluation at discrete time levels

Time Conditioning



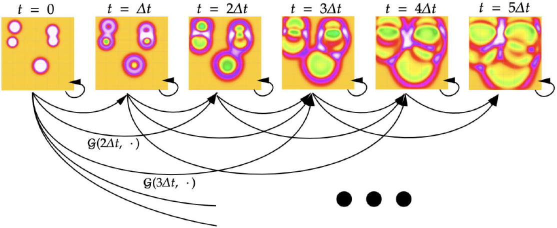
- Lead Time as an Input Channel
- $\mathrm{CNO}\ (\bar{t}, u(t)) \approx \mathcal{S}(\bar{t}, u(t)) = u(t + \bar{t})$.
- Add Conditional Normalizations after each layer !!

$$\mathcal{N}(w) = g_N(t) \odot \frac{w - \mathbb{E}(w)}{\sqrt{\mathrm{Var}\ (w) + \epsilon}} + h_N(t),$$

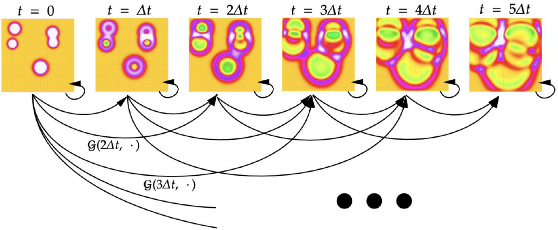
- g_N, h_N are MLPs in general but Linear suffices.
- Instance,Batch,Layer Normalizations.

Training Strategies I

- One at a Time training based on:
- Input-Target Pairs: $\bar{u}, \mathcal{S}(t_k, \bar{u}) = u(t_k)$
- For $t_K = T$, K training samples per trajectory.

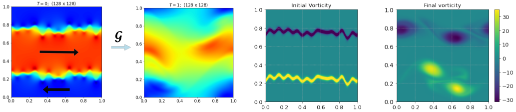


Training Strategies II

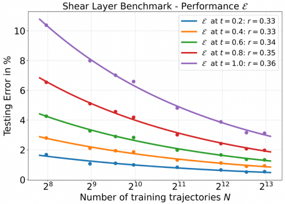


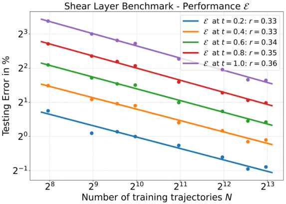
- all2all training based on:
- Input-Target Pairs: $u(t_i), \mathcal{S}(t_j - t_i, u(t_i)) = u(t_j), \forall i < j$
- Leverages Semi-group property of Solution Operator.
- $\frac{K^2+K}{2}$ training samples per trajectory !!
- Inference is Direct or Autoregressive
- Multiple possibilities for Autoregressive Rollouts
- Evaluation at any time $t > 0$ including Out-of-distribution times.

Results for Shear Layer

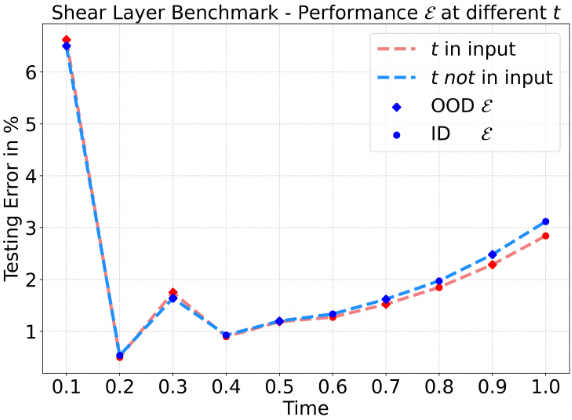


Error vs. Time

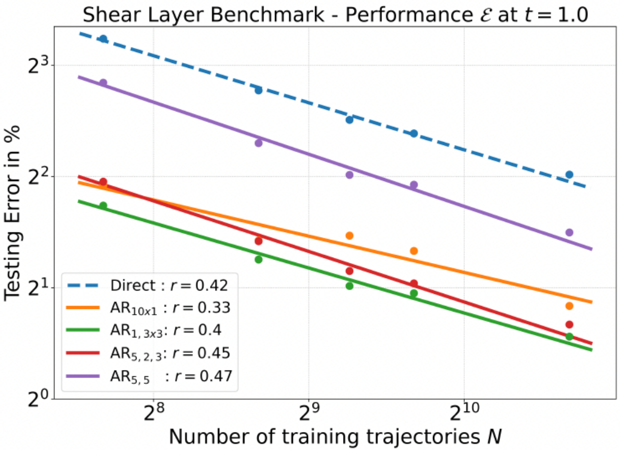




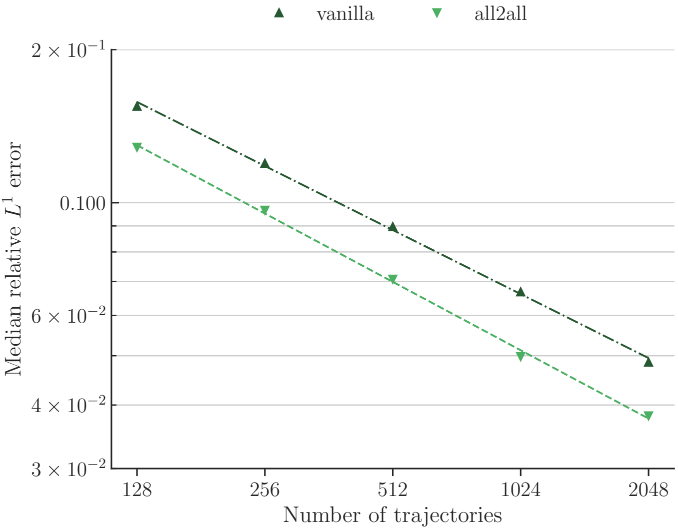
Results at OOD time levels.



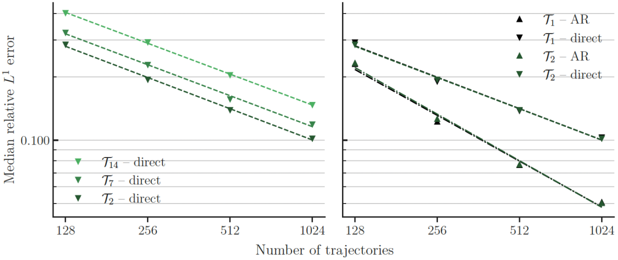
Results for Different Inference Strategies.



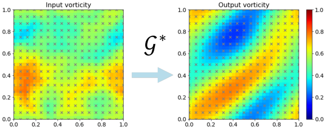
Results for Different Training Strategies.

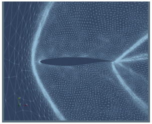
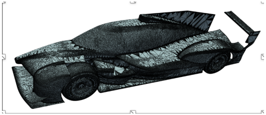


Further Results



Caveat II: PDEs on Arbitrary Domains



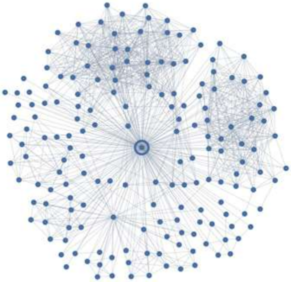



- Discussion so far has only focussed on Cartesian Domains
- Discretized with Uniform Grids.
- Most Real world PDEs are on Arbitrary Domains
- Discretized with Unstructured Grids or Point Clouds
- Need to handle such Data !!

Summary of Available Methods

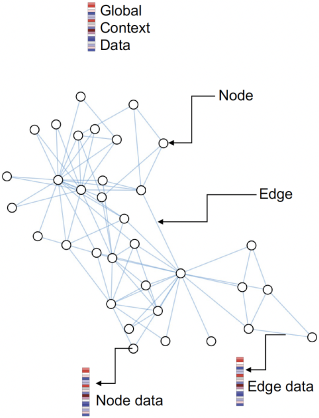
- Posing problem on Cartesian Domain through Masking.
- Direct Spectral Evaluations for FNO
  - DSE proposed in Lingsch, SM et. al., 2024
- Graph Neural Networks

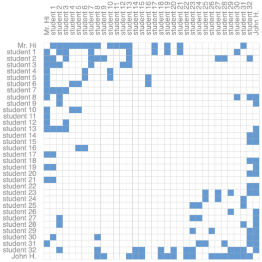
Learning Graph Structured data



- Input data on Graphs is ubiquitous:
  - Social networks, Recommender and Transport systems.
  - Chemistry and Biology (Molecules)
  - Mesh-based numerical simulations in Scientific computing

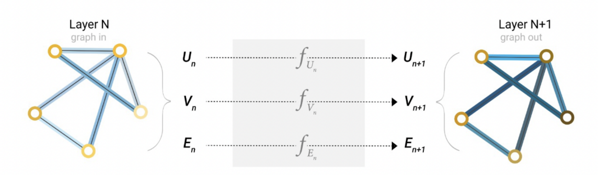
Basics on Graphs





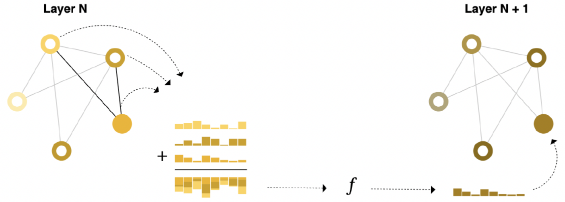
Adjacency matrix

What is a Graph Neural Network ?



- GNN is a composition of layers.
- Each layer updates Node + Edge features

Message Passing is the basis of all GNNs

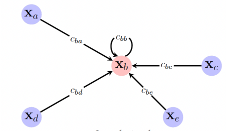


- Generic form of Message Passing:

$$h_i := f\left(v_i, \bigoplus_{j \in \mathcal{N}_i} \Psi(v_i, v_j) \right)$$

- f, Ψ are MLPs.
- $\bigoplus$ is an aggregation function

Ex I: Graph Convolutional Network (GCN)



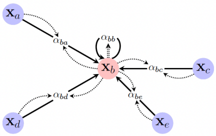
- Generic form of GCN:

$$\mathrm{h}_i := f\left(\mathrm{v}_i, \bigoplus_{j \in \mathcal{N}_i} c_{ij} \Psi(\mathrm{v}_j)\right)$$

- Specific example:

$$\mathrm{h}_i := g\left(\sum_{j \in \mathcal{N}_i \cup \{v_i\}} \frac{1}{\sqrt{\tilde{d}_i \tilde{d}_j}} \mathrm{W}\mathrm{v}_j\right), \quad \tilde{d}_i = 1 + \sum_j \mathrm{A}_{ij}$$

Ex II: Graph Attention Network (GAT)



- Generic form of GAT:

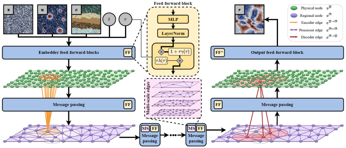
$$\mathrm{h}_i := f\left(\mathrm{v}_i, \bigoplus_{j \in \mathcal{N}_i} \alpha(v_i, v_j)\Psi(\mathrm{v}_j)\right)$$

- Specific example:

$$\mathrm{h}_i := g\left(\sum_{j \in \mathcal{N}_i \cup \{v_i\}} \alpha(v_i, v_j)\mathrm{W}\mathrm{v}_j\right),$$

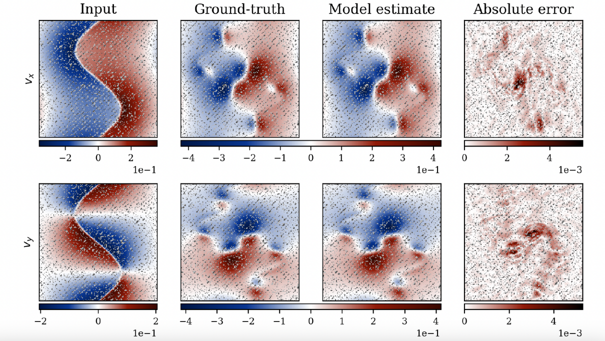
- α is a softmax based weight.

GNN for Operator Learning of PDEs ?



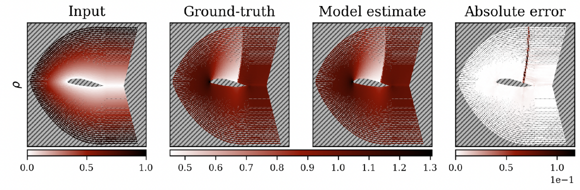
- RIGNO of Mousavi, SM, et. al., 2024
- Based on general MPNNs
- Multiple modifications to ensure:
  - Multiscale Information processing.
  - Temporal Continuity
  - Resolution Invariance

NS-SVS



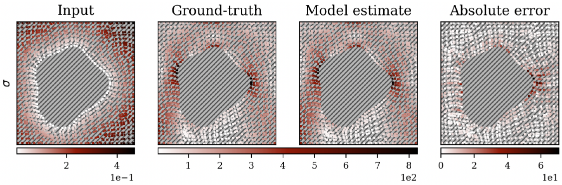
Model	FNO-DSE	GeoFNO	GINO	RIGNO
Errors (in %)	26.0	9.75	1.19	0.56

Flow Past Airfoills



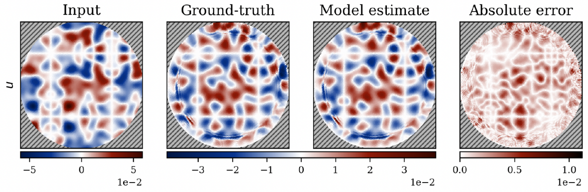
Model	FNO-DSE	GeoFNO	GINO	RIGNO
Errors (in %)	1.99	4.48	2.00	1.09

Plasticity



Model	FNO-DSE	GeoFNO	GINO	RIGNO
Errors (in %)	4.81	5.53	4.38	4.31

Waves in a Circle



Model	FNO-DSE	GeoFNO	GINO	RIGNO
Errors (in %)	5.52	13.1	5.82	5.35

RIGNO is very accurate

Dataset	Median relative L^1 error [%]				
(unstructured)	**RIGNO-18**	**RIGNO-12**	**GeoFNO**	**FNO DSE**	**GINO**
Heat-L-Sines	**0.04**	**0.05**	0.15	0.53	0.19
Wave-C-Sines	**5.35**	6.25	13.1	**5.52**	5.82
NS-Gauss	**2.29**	**3.80**	41.1	38.4	13.1
NS-PwC	**1.58**	**2.03**	26.0	56.7	5.85
NS-SL	**1.28**	**1.91**	24.3	29.6	4.48
NS-SVS	**0.56**	**0.73**	9.75	26.0	1.19
CE-Gauss	**6.90**	**7.44**	42.1	30.8	25.1
CE-RP	**3.98**	**4.92**	18.4	27.7	12.3
ACE	**0.01**	**0.01**	1.09	1.29	3.33
Wave-Layer	**6.77**	**9.01**	11.1	28.3	19.2
AF	**1.00**	**1.09**	4.48	1.99	2.00
Elasticity	**4.31**	4.63	5.53	4.81	**4.38**
(uniform grid)	**RIGNO-18**	**RIGNO-12**	**CNO**	**scOT**	**FNO**
NS-Gauss	**2.74**	3.78	10.9	**2.92**	14.24
NS-PwC	**1.12**	**1.82**	5.03	7.11	11.24
NS-SL	**1.13**	**1.82**	2.12	2.49	2.08
NS-SVS	**0.56**	0.75	**0.70**	0.99	6.21
CE-Gauss	**5.47**	**7.56**	22.0	9.44	28.69
CE-RP	**3.49**	**4.43**	18.4	9.74	31.19
ACE	**0.01**	**0.01**	0.28	0.21	0.60
Wave-Layer	**6.75**	8.97	**8.28**	13.44	28.01

- But not Efficient (more than 2d of training) !!
- Due to Repeated Sparse Memory Access