ETH zürich

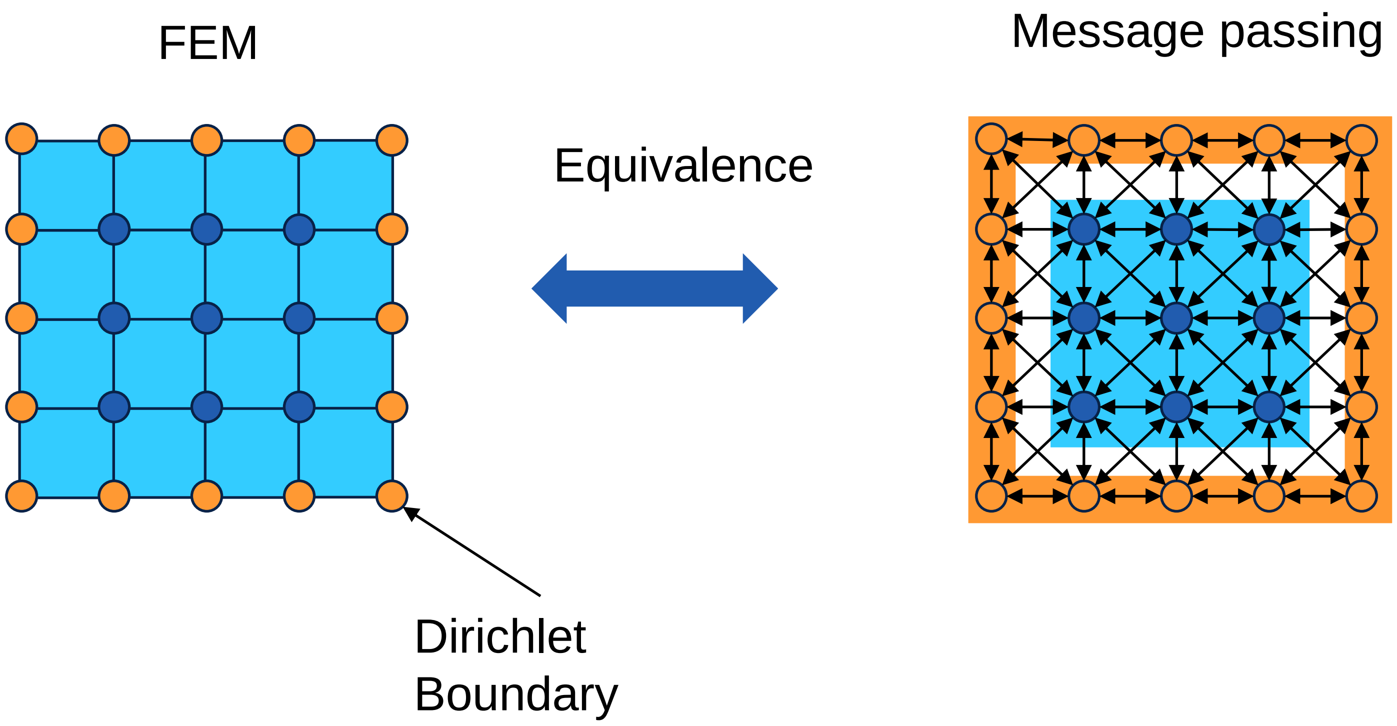
Department of Mathematics
Computational Science Engineering

# Learning Structural Dynamics with GNN

**Mingyuan Chi**
Semester Project
12. 01 2024, Zürich

D MATH

# Introduction



# Introduction

## Contribution

- Introduction Static Condensation Equivelant Architecture (SCEA) for modeling the Dirichlet boundary condition in the FEM problem.

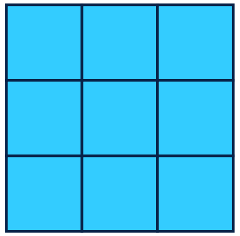
- Development of Galerkin Equivalent Architecture (GEA), which takes several forms including the local pseudo linear, local pseudo bilinear, and global version.

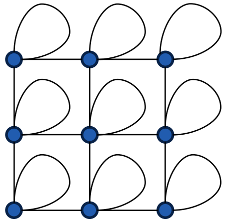
- Extensive experimentation to analyze the relation between the observation ratio and the precision of the model for various scenarios(invariant, boundary-variant, and force-variant), in which could we visualize the generalization competence for physical loss.

- Proposal of a fast and differentiable assemble method representing the assemble step in FEM as sparse-dense tensor multiplication.

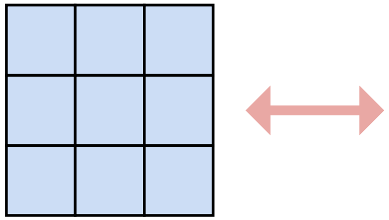
# Background

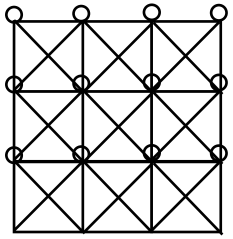
## Element Graph





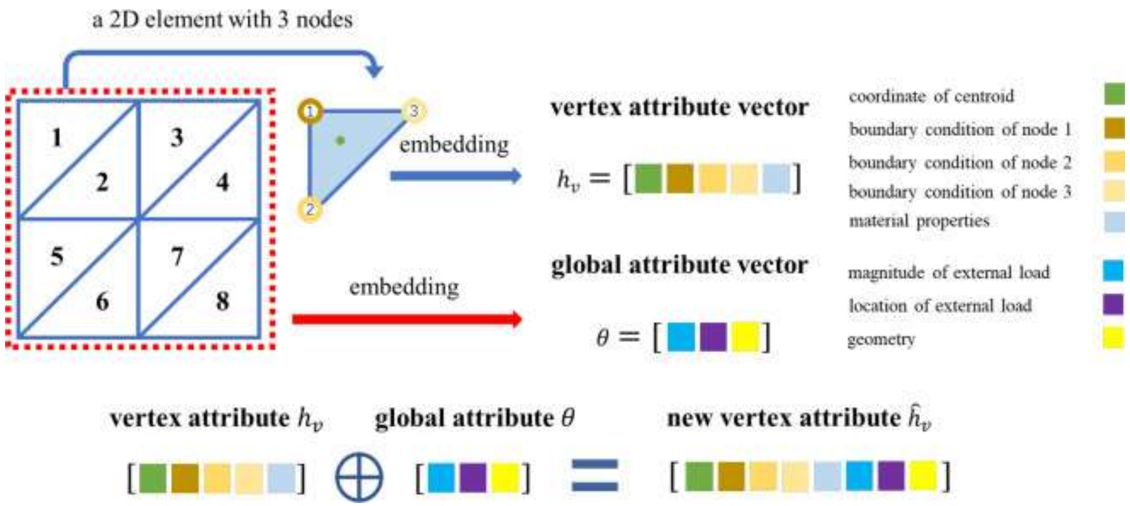
## Vertice Graph

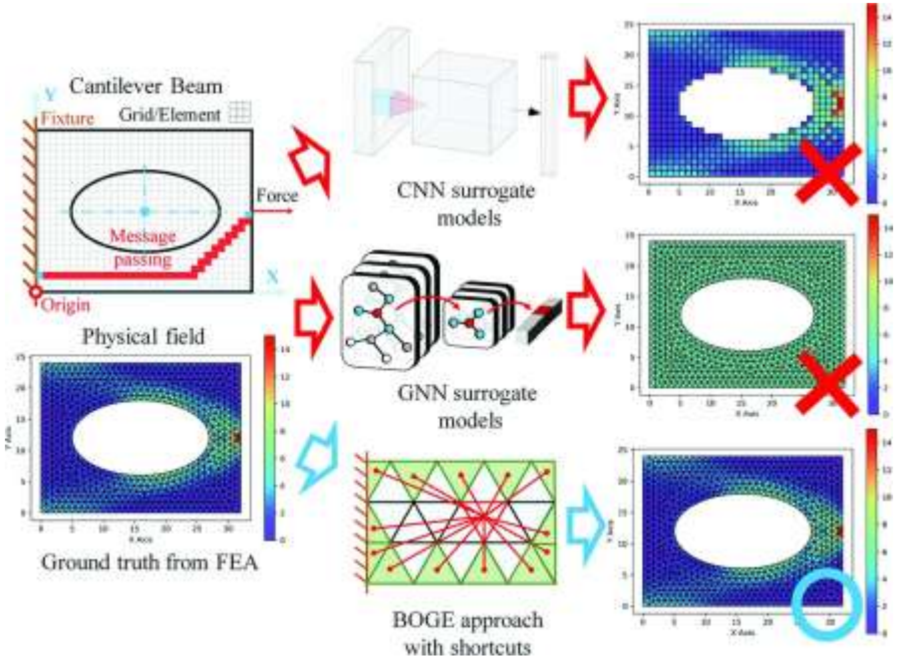




# Background

## Element Graph



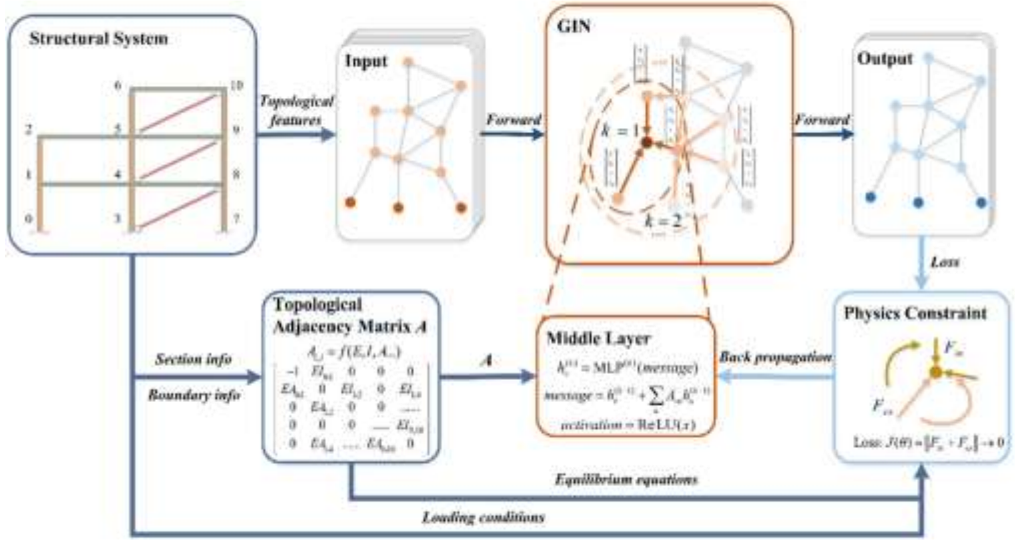




Jiang C, Chen N Z. Graph Neural Networks (GNNs) based accelerated numerical simulation[J]. Engineering Applications of Artificial Intelligence, 2023, 123: 106370.

Fu X, Zhou F, Peddireddy D, et al. An finite element analysis surrogate model with boundary oriented graph embedding approach for rapid design[J]. Journal of Computational Design and Engineering, 2023, 10(3): 1026-1046.

# Background

## Vertice Graph




---

**Algorithm 1** Solve forward PDE-governed problems via GCN

---

**Input**: PDE parameter $\bar{\mu}$, node coordinates $\chi$ and adjacency matrix $A$

**Output**: The solution $\hat{U}$

**1**.Pre-compute the matrix basis function $\Phi$ on the qudrature points to obtain $\Phi(\tilde{x}^v)$, $\Phi(\tilde{x}^s)$, $\nabla\Phi(\tilde{x}^v)$, $\nabla\Phi(\tilde{x}^s)$ ;

**2**.Formulate the residual function $R(\tilde{U};\mu)$ defined in (10);

**3**.Apply the static condensation to (10) to obtain (12);

**4**.Partition the degrees of freedom $\hat{U}(\Theta) = (\hat{U}_u(\Theta)^T, \hat{U}_c^T)^T$ and enforce the essential condition, $\hat{U}_c = U_e$, to formulate the physics-informed loss function (13);

**5**.Solve the optimization problem (14) to obtain $\hat{U} = (\hat{U}_u(\Theta^*)^T, U_e^T)^T$ ;

---

**Song L H, Wang C, Fan J S, et al. Elastic structural analysis based on graph neural network without labeled data[J]. Computer-Aided Civil and Infrastructure Engineering, 2023, 38(10): 1307-1323.**

**Gao H, Zahr M J, Wang J X. Physics-informed graph neural Galerkin networks: A unified framework for solving PDE-governed forward and inverse problems[J]. Computer Methods in Applied Mechanics and Engineering, 2022, 390: 114502.**

# Background

## Linear Elasticity

$$Ku = f \quad (1)$$

$$K \overset{\text{bsr matrix}}{\leftarrow} \hat{K}_{\text{global}} \quad (2)$$

$$\hat{K}^{nkl}_{\text{global}} = \mathcal{P}^{nhij}_{\mathcal{E}} \hat{K}^{hklij}_{\text{local}} \quad (3)$$

$$\hat{K}^{ij}_{\text{local}} = \sum_{m} \phi_m \mathbb{C}(\xi_m)_{ijkl} \nabla N^j(\xi_m)_l \nabla N^i(\xi_m) = B^\top D B \quad (4)$$

$\hat{K}_{\text{global}}$ : non zero value of the global galerkin matrix, $K_{\text{global}} \in \mathbb{R}^{|\mathcal{E}| \times d \times d}$

$\hat{K}_{\text{local}}$ : local galerkin matrix for each element , $K_{\text{local}} \in \mathbb{R}^{|\mathcal{C}| \times h \times h \times d \times d}$

$\mathcal{P}_{\mathcal{E}}$ : projection (assemble) tensor from $\hat{K}_{\text{local}}$ to $\hat{K}_{\text{global}}$, $\mathcal{P}_{\mathcal{E}} \in \mathbb{R}^{|\mathcal{E}| \times |\mathcal{C}| \times h \times h}_{\text{sparse}}$

$\mathcal{C}$ : elements/cells

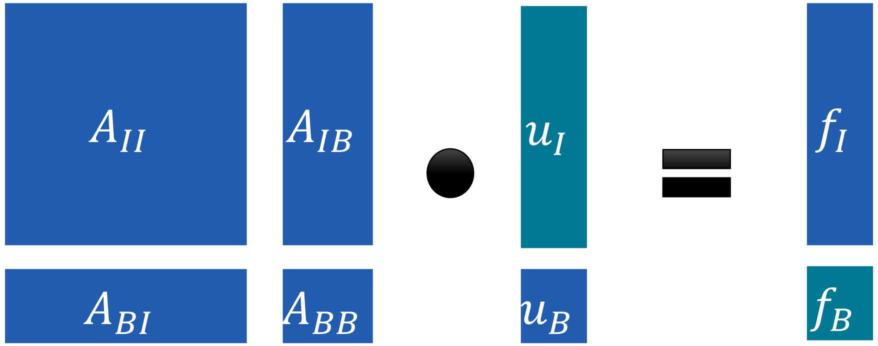
$h$ : number of basis for each element/-cell

$\mathcal{E}$ : connections for points

$\mathcal{V}$ : points

# Background

## Static Condensation




$$\begin{bmatrix} A_{II} & A_{IB} \\ A_{IB}^{\top} & A_{BB} \end{bmatrix} \begin{bmatrix} u_I \\ u_B \end{bmatrix} = \begin{bmatrix} f_I \\ f_B \end{bmatrix}$$

$$\begin{cases} A_{II} u_I = f_I - A_{IB} u_B \\ A_{IB}^{\top} u_I + A_{BB} u_B = f_B \end{cases}$$

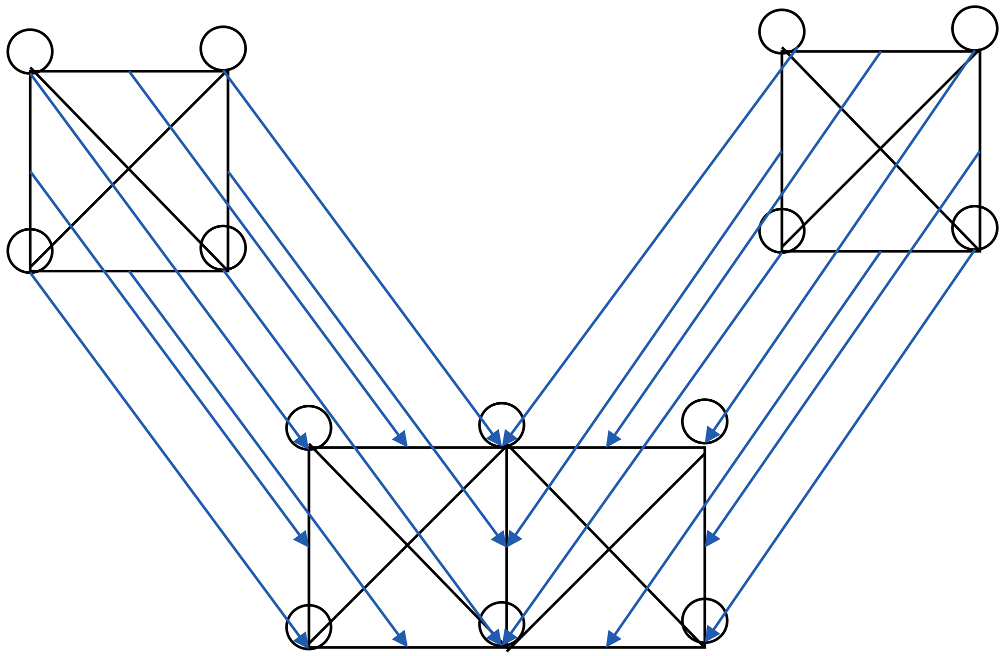
$\text{known} : A, u_B, f_I$

$\text{unknown} : u_I, f_B$

# Methodology

## Fast Assemble

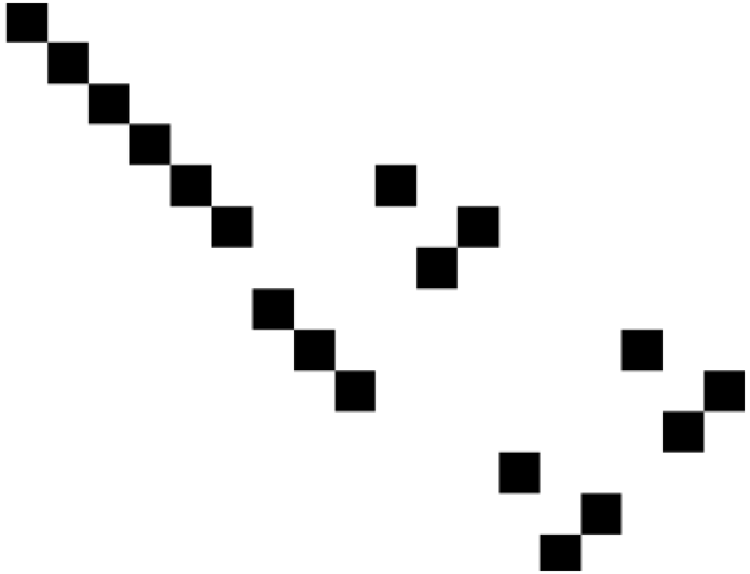
$\mathcal{P}_{\mathcal{E}}$ : projection (assemble) tensor from $\hat{K}_{\text{local}}$ to $\hat{K}_{\text{global}}$, $\mathcal{P}_{\mathcal{E}} \in \mathbb{R}_{\text{sparse}}^{|\mathcal{E}|\times|\mathcal{C}|\times h\times h}$



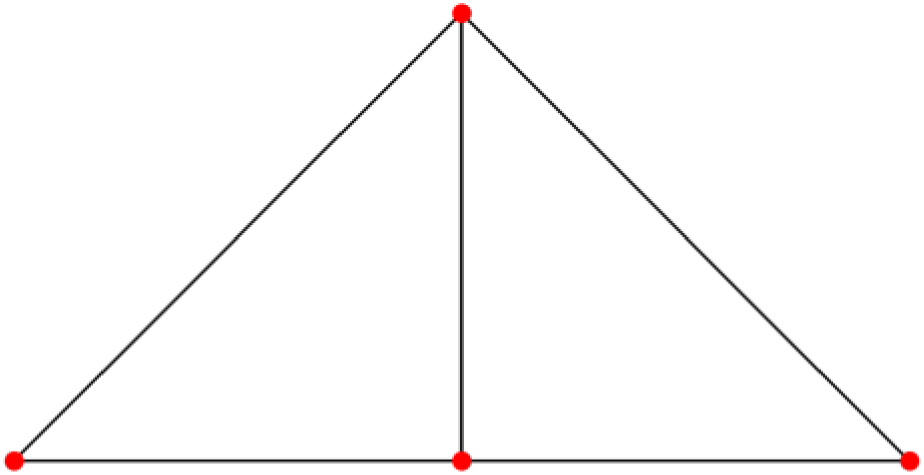
# Methodology

## Fast Assemble

$\mathcal{P}_{\mathcal{E}}$ : projection (assemble) tensor from $\hat{K}_{\text{local}}$ to $\hat{K}_{\text{global}}$, $\mathcal{P}_{\mathcal{E}} \in \mathbb{R}^{|\mathcal{E}|\times|\mathcal{C}|\times h\times h}_{\text{sparse}}$



$\mathcal{P}_{\mathcal{E}} \in \mathbb{R}^{14\times 18}$



# Methodology

**FEM** ⟷ **MPNN**

**Reverse Problem**

$$K, f \to u$$

**Static Condense Equivalent Architecture(SCEA)**

**Forward Problem**

$$f, u \to K$$

**Galerkin Equivelant Architecture (GEA)**

# Methodology

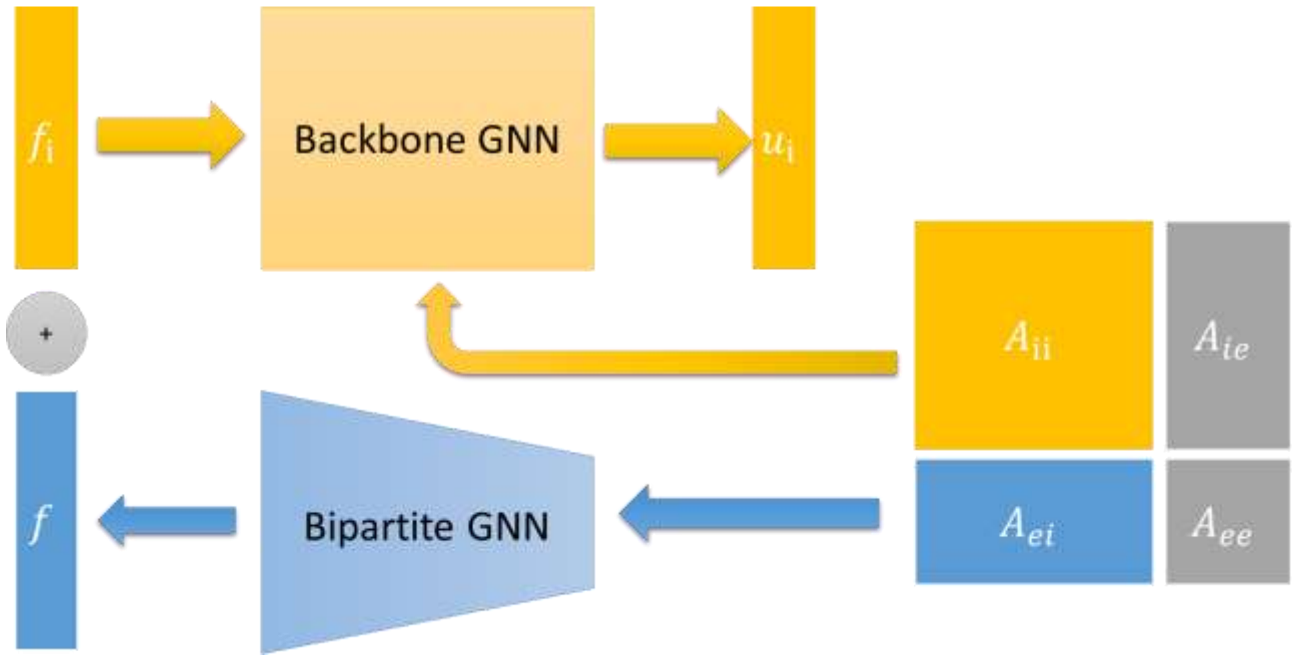
## Static Condense Equivalent Architecture(SCEA)

$$u_i = K_{ii}^{-1}(f_i - K_{ei}u_e)$$

$$u_i = \mathrm{GNN}_{\theta_1}(A_{ii}, f_i + \text{B-GNN}_{\theta_2}(A_{ei}, u_e), x_i)$$

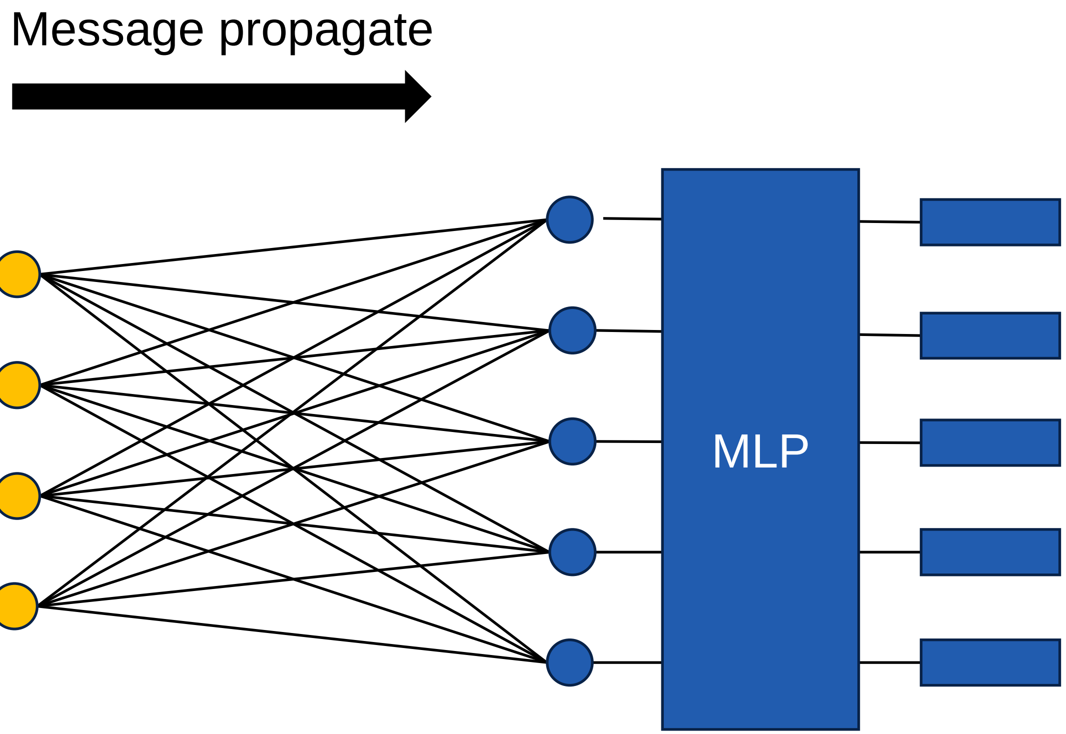
$$\mathrm{GNN}_{\theta_1}(A_{ii}, f_i', x_i) \approx K_{ii}^{-1}(x_i)(f_i')$$

$$\text{B-GNN}_{\theta_2}(A_{ei}, u_e, x_e) \approx -K_{ei}u_e$$



# Methodology

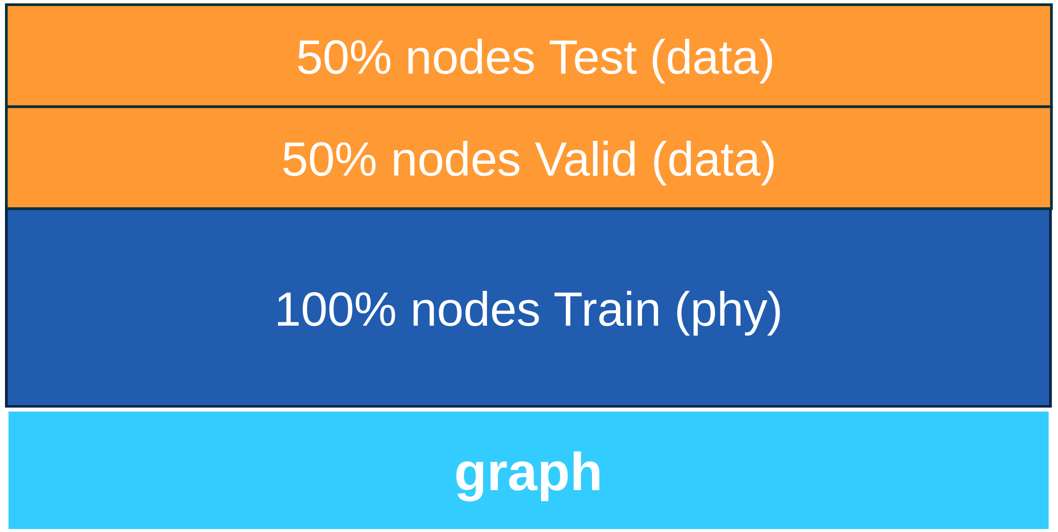
## Static Condense Equivalent Architecture(SCEA)

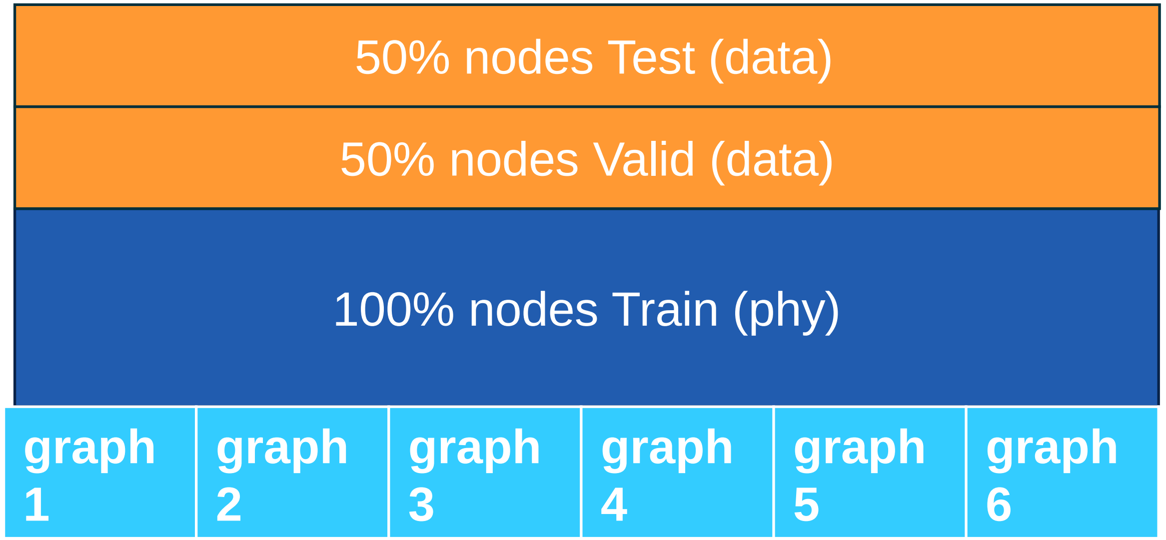



BiParite - GNN

# Methodology

## Static Condense Equivelant Architecture(SCEA)



# Methodology

## Static Condense Equivelant Architecture(SCEA)

when `train_ratio=0.0`

$$
\begin{cases}
\mathcal{L}_{\text{train}} = \mathcal{L}_{\text{phy}} = \begin{cases} \|Ku - f\|_2 & \text{strong pinn} \\ \left\| \mathcal{P}_{\mathcal{V}} \left( \int_\Omega \begin{bmatrix} \frac{\partial u}{\partial x} & 0 \\ 0 & \frac{\partial u}{\partial y} \\ \frac{\partial u}{\partial y} & \frac{\partial u}{\partial x} \end{bmatrix}_{eid} D_{eij} B_{ebjd} |J|_e - f_{ebd} N_b |J|_e \mathrm{d}v \right) \right\|_2 & \text{weak pinn} \end{cases} \\
\mathcal{L}_{\text{valid}} = \mathcal{L}_{\text{data}} = \|\mathrm{NN}(u_B, f_I) - u_I\|_2
\end{cases}
$$

# Methodology

## Static Condense Equivelant Architecture(SCEA)

when `train_ratio=0.0`

$$
\begin{cases}
\mathcal{L}_{\text{train}} = \mathcal{L}_{\text{phy}} = \begin{cases} \|Ku - f\|_2 & \text{strong pinn} \\ \left\| \mathcal{P}_{\mathcal{V}} \left( \int_\Omega \begin{bmatrix} \frac{\partial u}{\partial x} & 0 \\ 0 & \frac{\partial u}{\partial y} \\ \frac{\partial u}{\partial y} & \frac{\partial u}{\partial x} \end{bmatrix}_{eid} D_{eij} B_{ebjd} |J|_e - f_{ebd} N_b |J|_e \mathrm{d}v \right) \right\|_2 & \text{weak pinn} \end{cases} \\
\mathcal{L}_{\text{valid}} = \mathcal{L}_{\text{data}} = \|\mathrm{NN}(u_B, f_I) - u_I\|_2
\end{cases}
$$

# Experiments

## Static Condense Equivelant Architecture(SCEA) : Baselines

### Baselines

GCN: $$H^{l+1} = \sigma(\underbrace{D^{-\frac{1}{2}}(A+I)D^{-\frac{1}{2}}}_{\mathcal{L}}H^lW^l + b^l)$$

GAT:
$$H^{l+1} = \sigma\left(\sum_{j\in\mathcal{N}_i\cup\{i\}} \alpha_{ij}\mathbf{W}h_j\right)$$
$$\alpha_{ij} = \frac{\exp(\mathrm{LeakyReLU}(\mathbf{a}^\top[\mathbf{W}h_i\|\mathbf{W}h_j]))}{\sum_{k\in\mathcal{N}_i\cup\{i\}}\exp(\mathrm{LeakyReLU}(\mathbf{a}^\top[\mathbf{W}h_i\|\mathbf{W}h_k]))}$$

SIGN: $$H = \mathrm{MLP}(\|_{i=0}^{n}\mathrm{MLP}_i(L^iX))$$

Node-Edge GNN:
$$H_{\mathcal{E}}^{l+1} \leftarrow \sigma\left(W_{\mathcal{E}}\left([H_{\mathcal{V},u}^l\|H_{\mathcal{V},v}^l\|H_{\mathcal{E}}^l]\right) + b_{\mathcal{E}}\right)$$
$$H_{\mathcal{V},i}^{l+1} \leftarrow \sigma\left(W_{\mathcal{V}}\frac{1}{|\mathcal{N}_i\cup\{i\}|}\sum_{j\in\mathcal{N}_i} H_{\mathcal{E},ij}^{l+1} + b_{\mathcal{V}}\right)$$

### Condense Type

none: DBC added to training set (data loss)

static: $$u_i = \mathrm{GNN}_{\theta_1}(A_{ii}, f_i - K_{ei}u_e, x_i)$$

nn: $$u_i = \mathrm{GNN}_{\theta_1}(A_{ii}, f_i + \text{B-GNN}_{\theta_2}(A_{ei}, u_e), x_i)$$

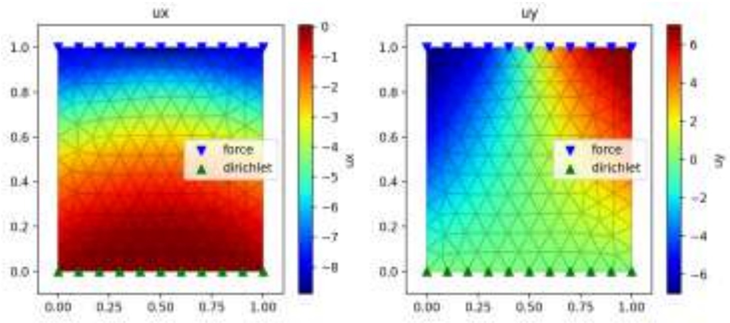
# Experiments

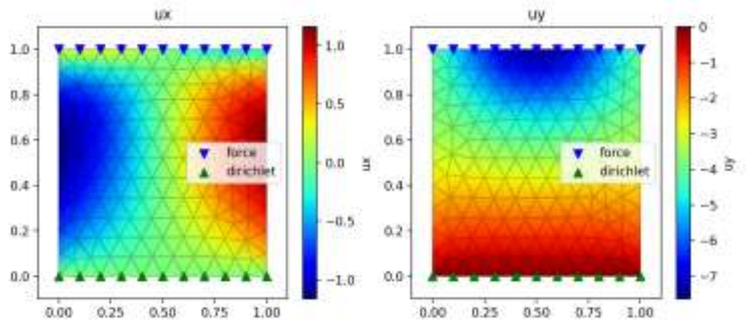
## Static Condense Equivelant Architecture(SCEA) : Dataset

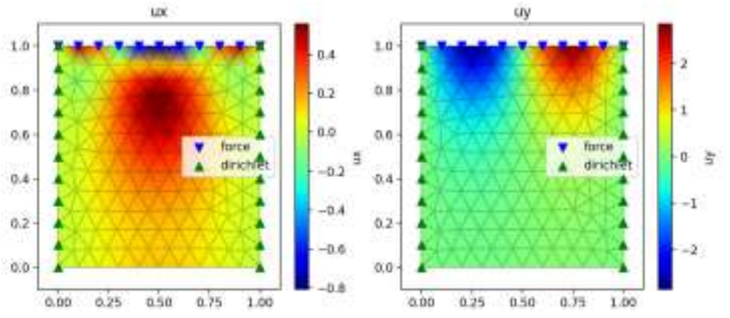
$p(x) = p_0 \sin(2\pi x / a) \quad x \in [0, a]$

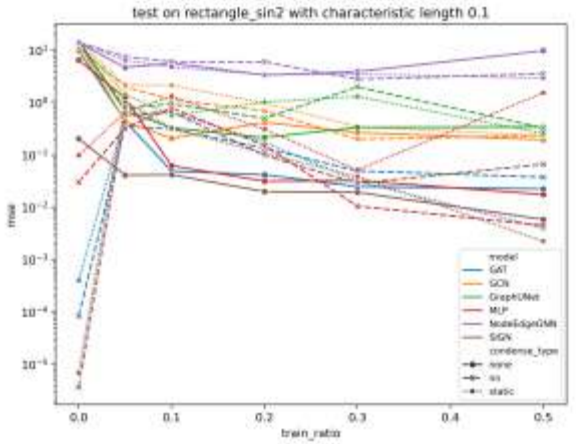
$p(x) = p_0 \sin(\pi x / a) \quad x \in [0, a]$

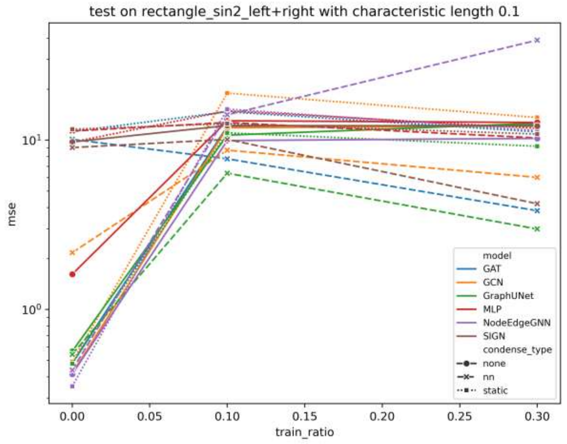
$p(x) = p_0 \sin(2\pi x / a) \quad x \in [0, a]$

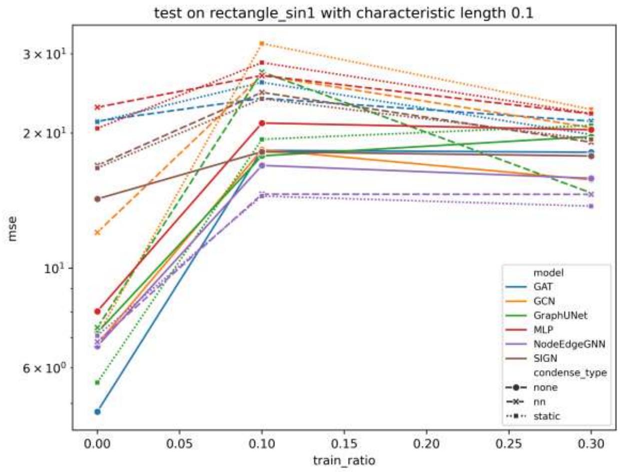


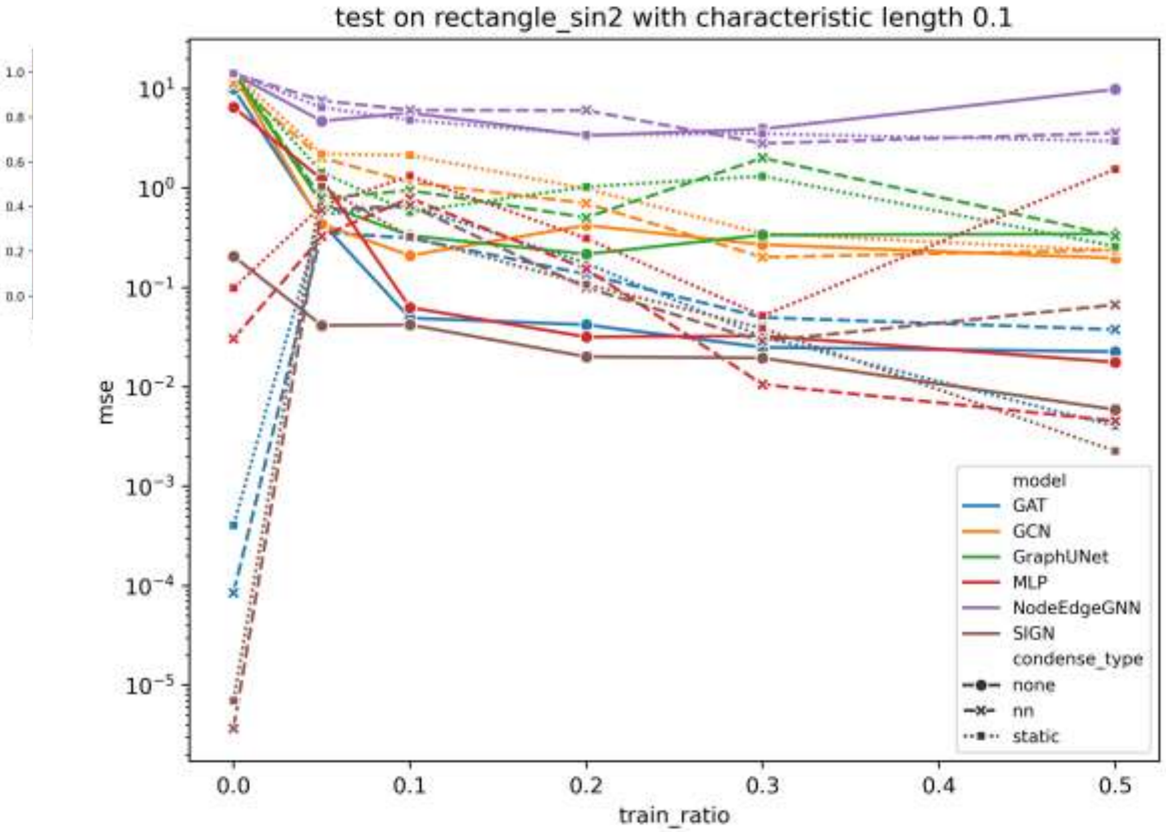
# Experiments

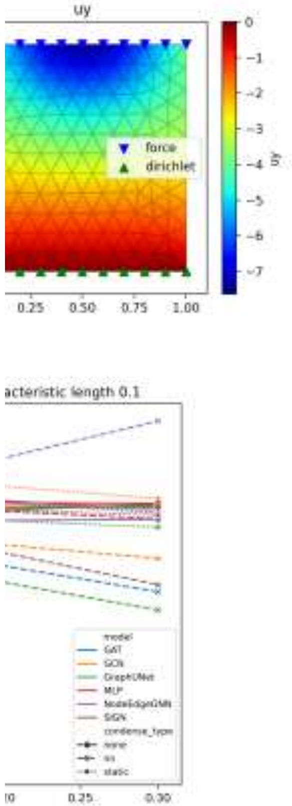
## Static Condense Equivelant Architecture(SCEA) : Dataset

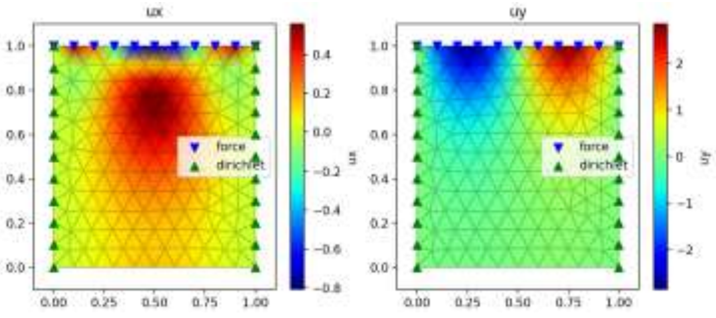
$p(x) = p_0 \sin(2\pi x/a) \quad x \in [0, a]$

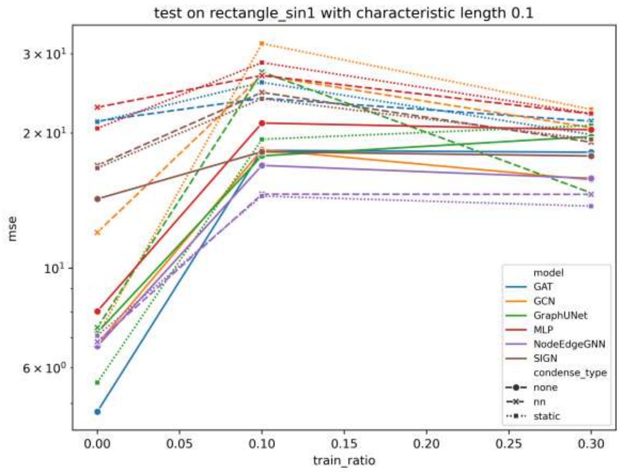
$p(x) = p_0 \sin(\pi x/a) \quad x \in [0, a]$

$p(x) = p_0 \sin(2\pi x/a) \quad x \in [0, a]$



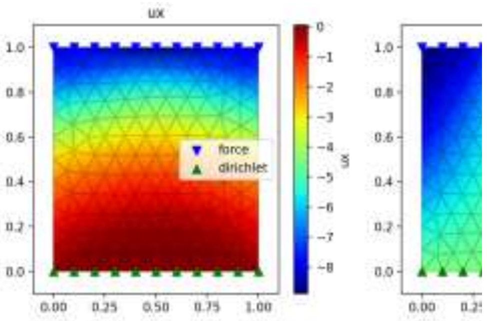



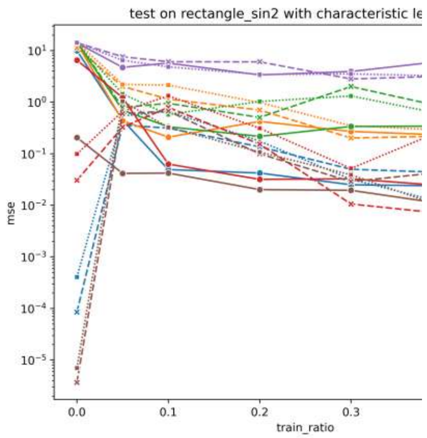



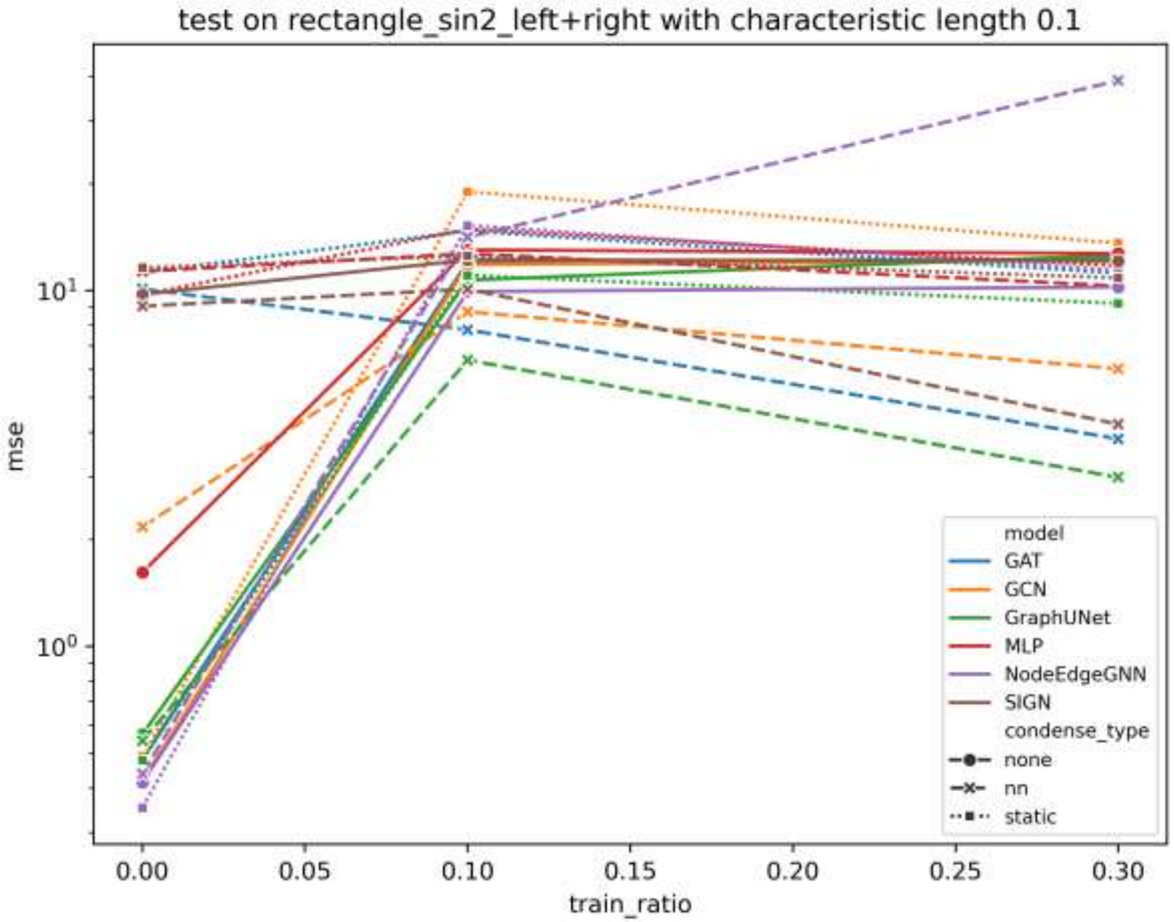
# Experiments

## Static Condense Equivelant Architecture(SCEA) : Dataset

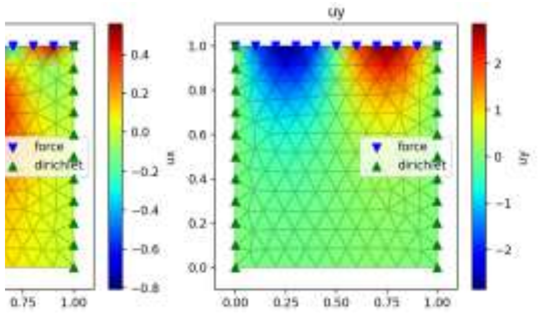
$$p(x) = p_0 \sin(2\pi x/a)$$

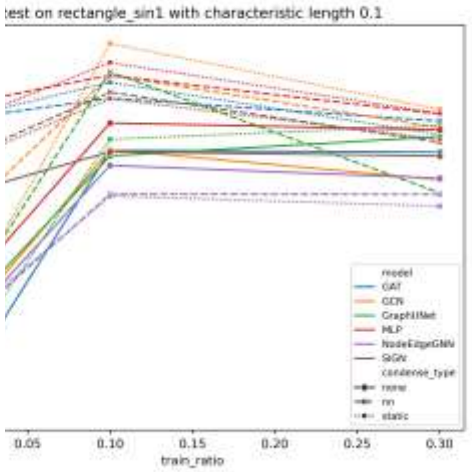



$$\ldots p_0 \sin(2\pi x/a) \quad x \in [0, a]$$

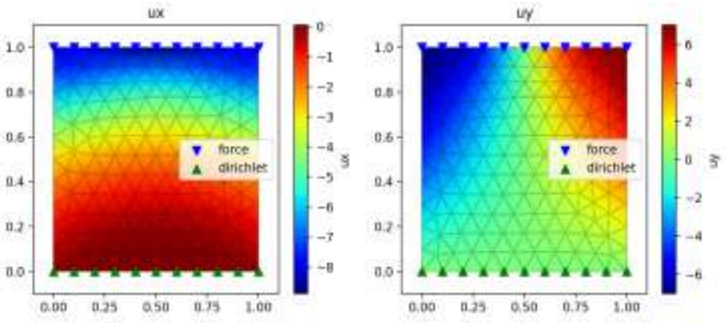


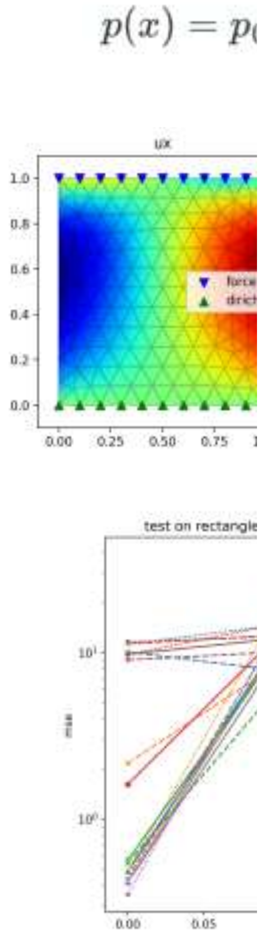



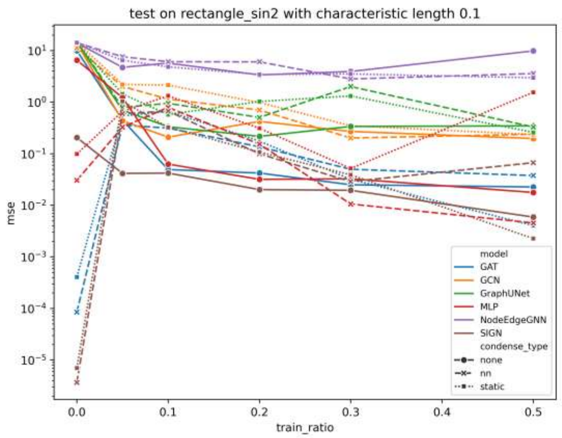
# Experiments

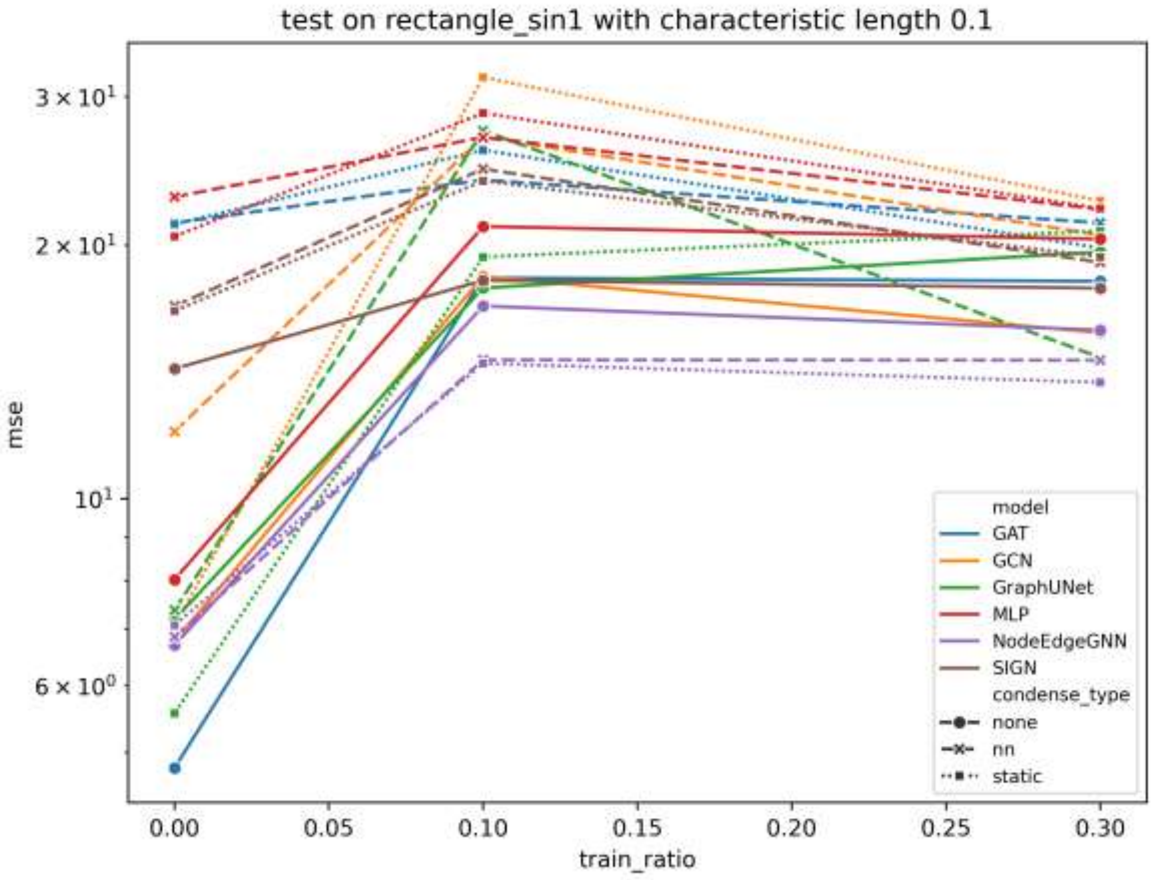
## Static Condense Equivelant Architecture(SCEA) : Dataset

$$p(x) = p_0 \sin(2\pi x/a) \quad x \in [0, a]$$



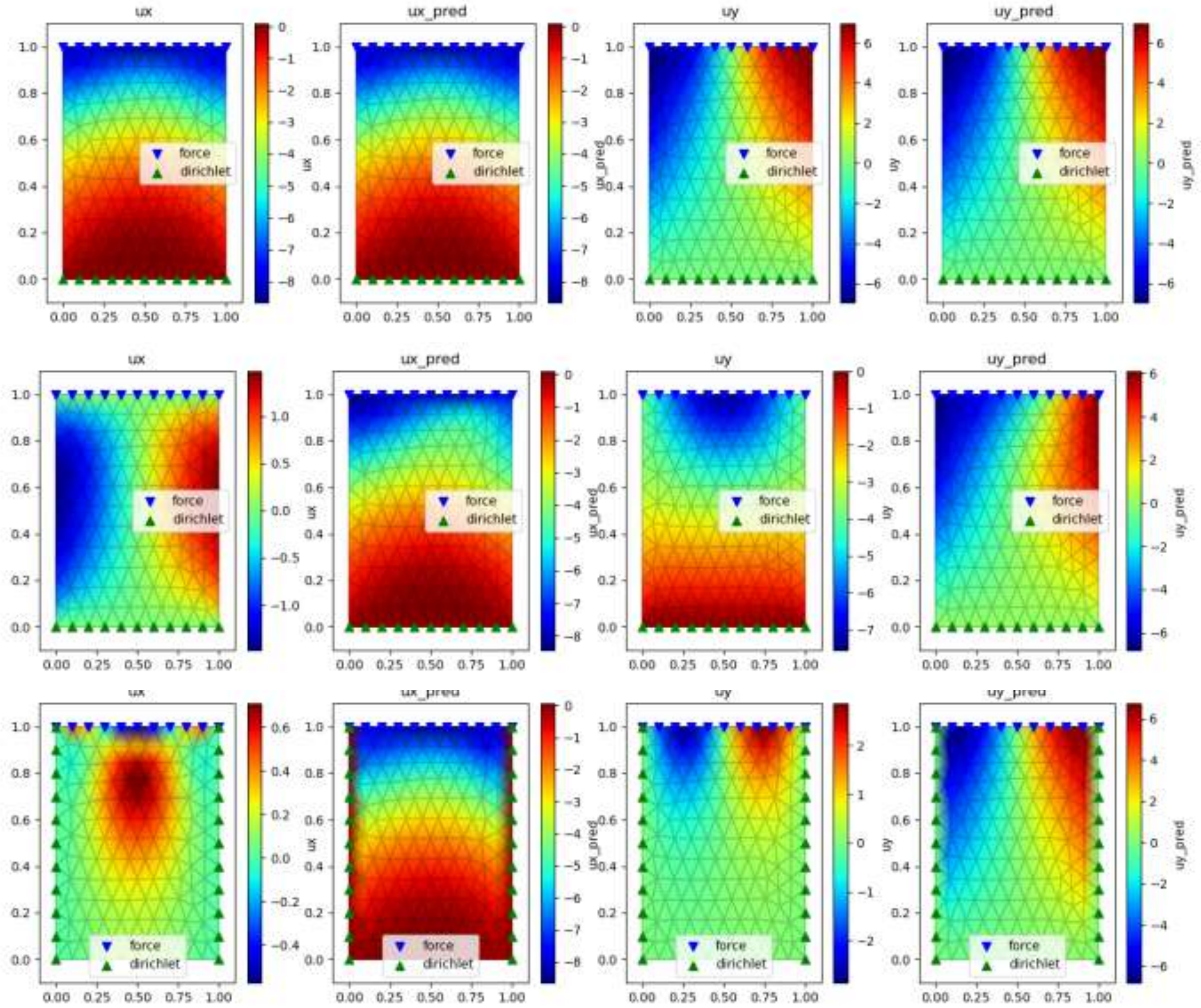





# Experiments

## Static Condense Equivelant Architecture(SCEA) : Case Study



0.0train ratio
+ SIGN(8 hops)
+ auto weight
+ strong form

- Training dataset is good, other dataset is not
- One-shot is not good at generalization

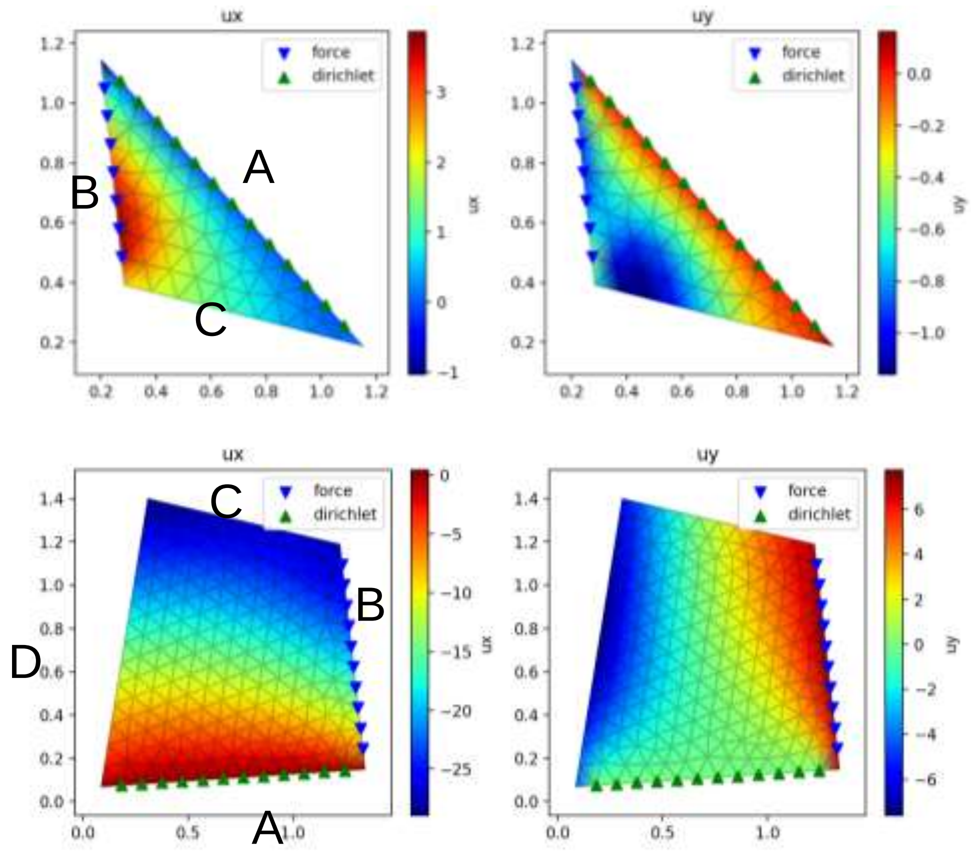
# Experiments

## Static Condense Equivelant Architecture(SCEA) : Summary

- SIGN/GAT + static/nn condense performs best on fully phy loss
- Phy loss tend to overfit, and remember the coordinate mapping to displacement
- Frequency Variant performance is better than Boundary Variant

# Experiments

## Static Condense Equivelant Architecture(SCEA) : Generalization Dataset




$$\mathcal{V} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} + \mathcal{N}(0, 0.4)$$

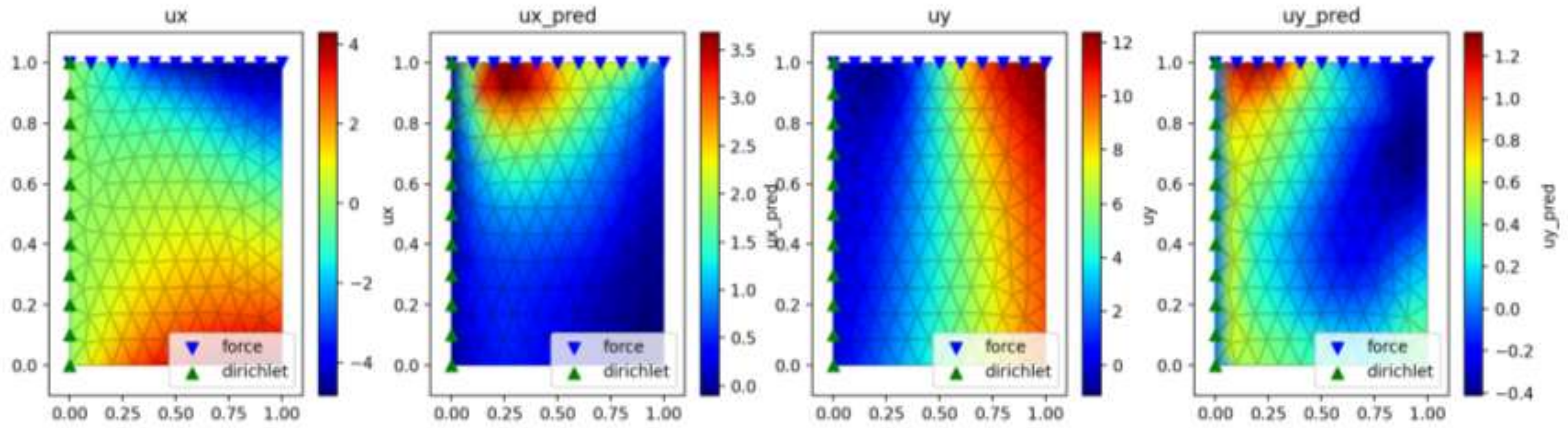
$$\mathcal{V} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} + \mathcal{N}(0, 0.4)$$

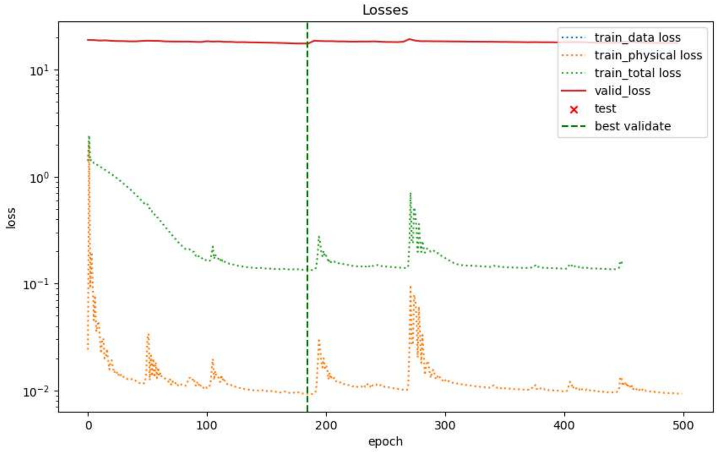


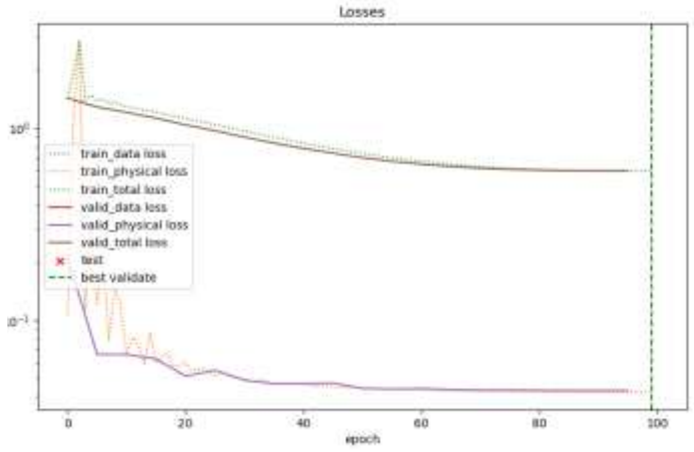
$$4 \times 7 \times (24 + 6) = 840$$

# Experiments

## Static Condense Equivelant Architecture(SCEA) : Generalization Result

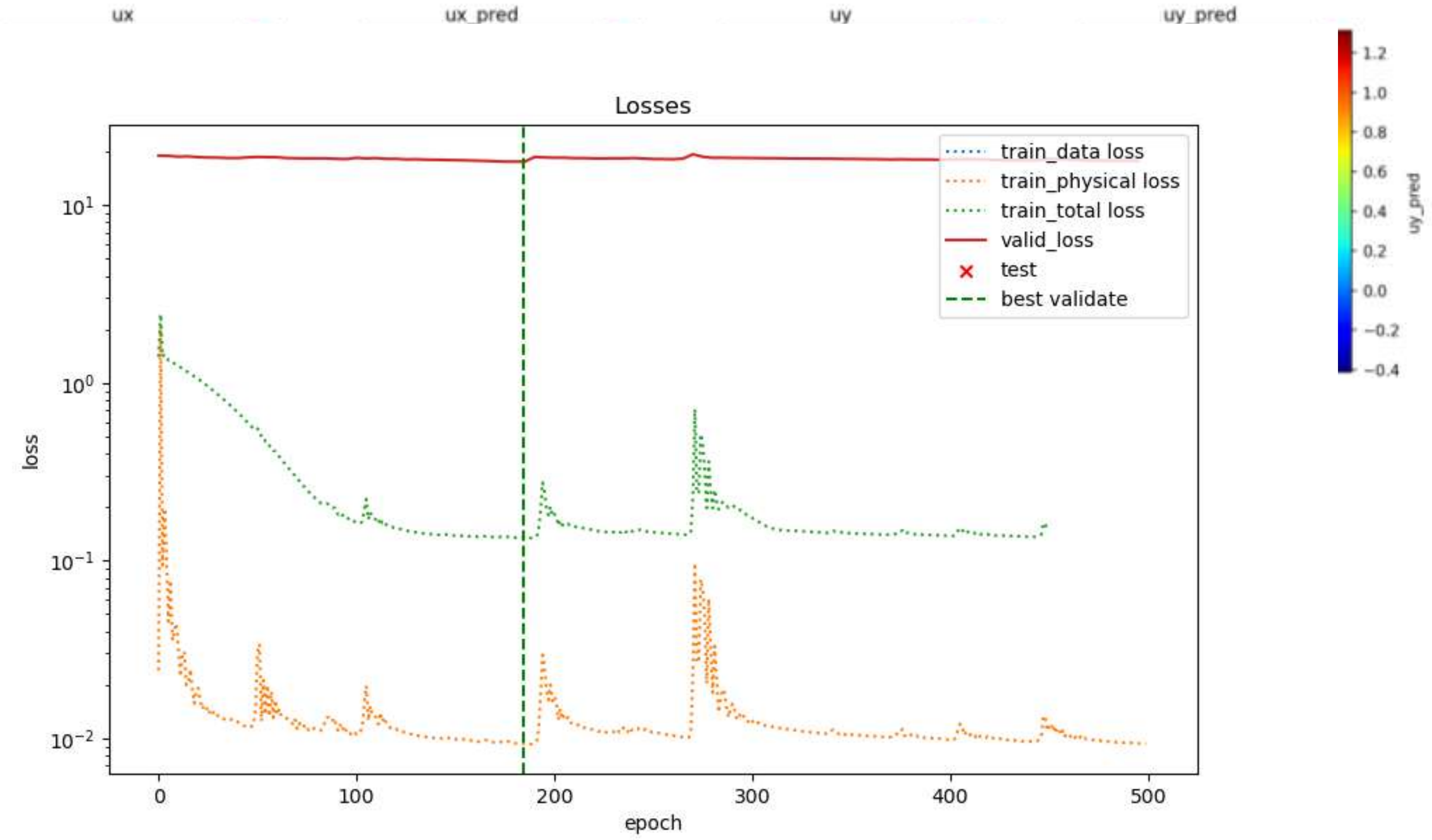


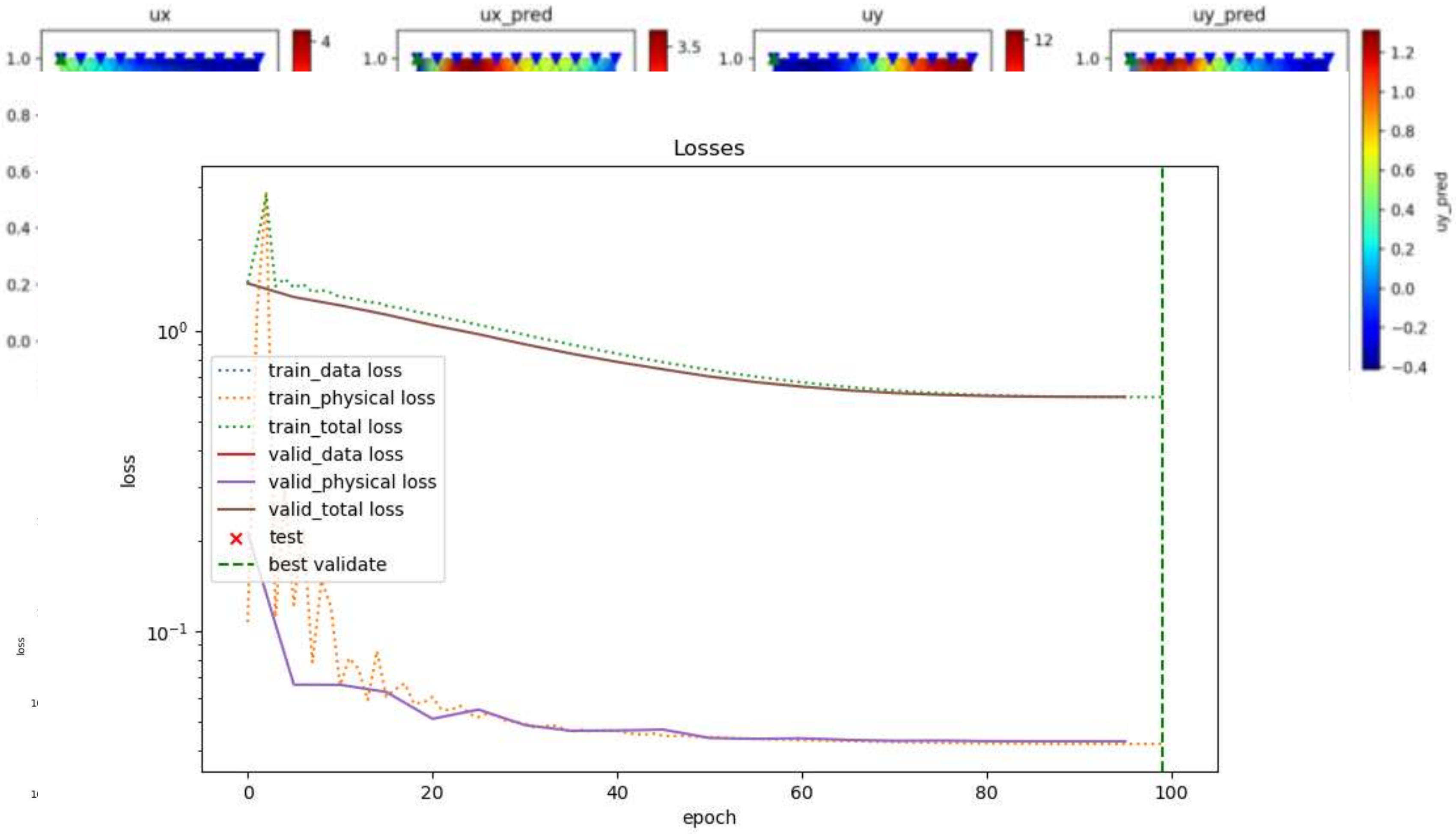
# Experiments

## Static Condense Equivelant Architecture(SCEA) : Generalization Result



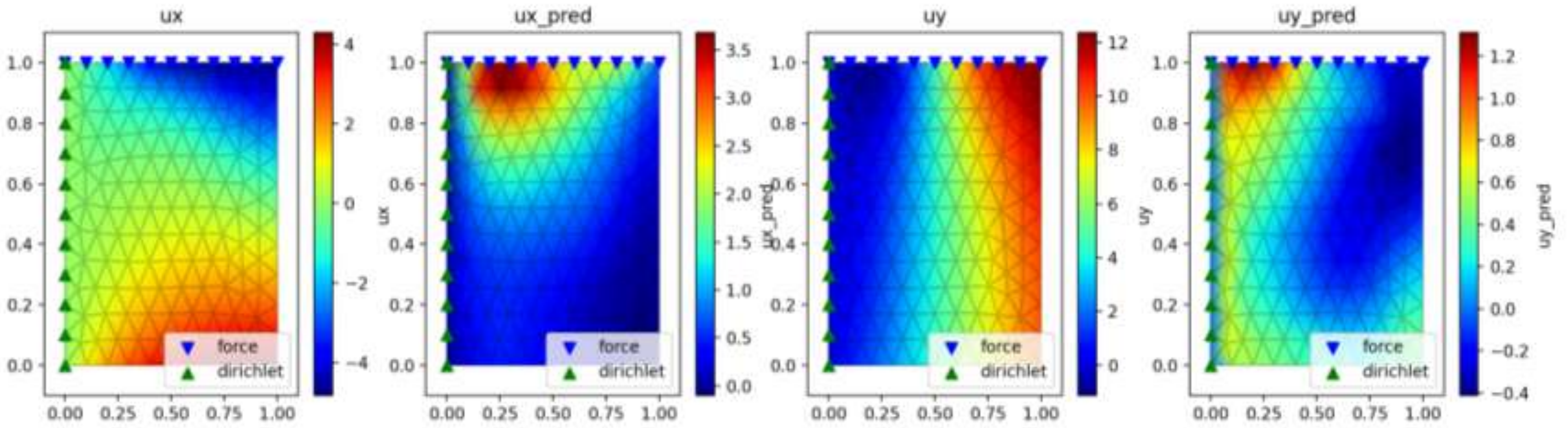
# Experiments

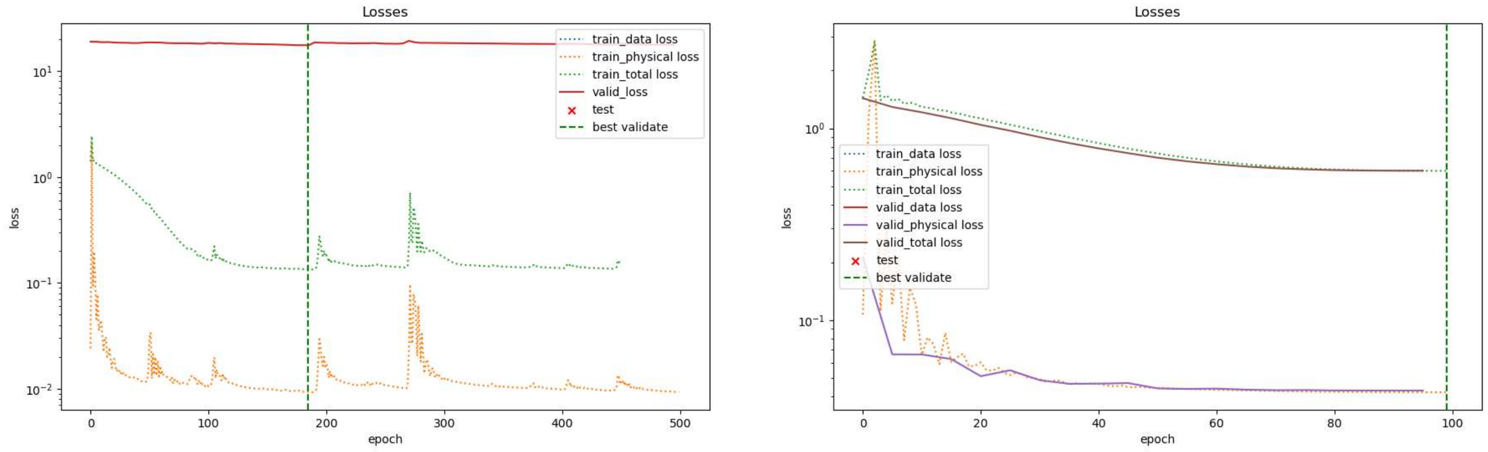
## Static Condense Equivelant Architecture(SCEA) : Generalization Result



# Experiments

**Static Condense Equivelant Architecture(SCEA) : Generalization Result**



# Experiments

**Static Condense Equivelant Architecture(SCEA) : Generalization Result**

- Multi Graph physical loss training still cannot perform well on generalization for data loss
- It may because of the disadvantage of PINN (physical loss is gradient instead of the function itself)

# Methodology

## Galerkin Equivelant Architecture (GEA) : Pseudo Bilinear

$$K^{ij} = \sum \phi_m \mathbb{C}(\xi_m)_{ijkl} \nabla N^j(\xi_m)_l \nabla N^i(\xi_m)$$

$$\mathcal{M}_\xi(\{x^i | x^i \in \mathcal{C}\}) \in \mathbb{R}^{b\times d \to \mathbb{R}^{|\phi|\times d}}$$

$$\mathcal{M}_{\nabla\phi}(\xi) \in \mathbb{R}^{|\phi|\times d} \to \mathbb{R}^{b\times d}$$

$$\mathcal{M}_{\mathbb{C}}(\xi) \in \mathbb{R}^{d} \to \mathbb{R}^{d\times d\times b\times b}$$

$$K^{ij}_{\text{local}} = \frac{\phi \mathcal{M}_{\mathbb{C}}(\xi_{\mathcal{M}_m})_{ijkl} \mathcal{M}_{\nabla N}(\xi_{\mathcal{M}})^j_l M_{\nabla N}(\xi_{\mathcal{M}})^i_k}{\sum \phi}$$

$$K^{nkl}_{\text{global}} = \mathcal{P}^{nhij}_{\mathcal{E}} K^{hklij}_{\text{local}}$$

$$K_{\text{global}} \overset{\text{bsr matrix}}{\rightarrow} \hat{K}_{\text{global}}$$

- $N$ : basis function
- $i, j$ : basis function notation
- $k, l$ : dimension notation
- $m$ : quadrature notation
- $n$ : edge notation
- $h$ : element notation
- $\phi$ : quadrature weight
- $\xi$ : quadrature point $\xi_m \in \mathbb{R}^d$
- $b$ : number of basis
- $d$ : number of dimension
- $c$ : number of cell/elements
- $\mathcal{C}$ : cell/element, which has $b$ basis
- $\mathcal{E}$ : edges in the graph representation
- $\mathcal{V}$ : number of points/vertex/basis of the graph representation
- $K_{\text{local}}$ : local stiffness/Galerkin tensor, $K_{\text{local}} \in \mathbb{R}^{c\times b\times b\times d\times d}$
- $K_{\text{global}}$ : global stiffness/Galerkin tensor, $K_{\text{global}} \in \mathbb{R}^{|\mathcal{E}|\times d\times d}$
- $\hat{K}_{\text{global}}$ : global stiffness/Galerkin matrix, $\hat{K}_{\text{global}} \in \mathbb{R}^{(|\mathcal{V}|\times d)\times(|\mathcal{V}|\times d)}_{\text{sparse}}$
- $\mathcal{P}_{\mathcal{E}}$ : projection tensor from $K_{\text{local}}$ to $K_{\text{global}}$, $\mathcal{P}_{\mathcal{E}} \in \mathbb{R}^{|\mathcal{E}|\times c\times b\times b}_{\text{sparse}}$

# Methodology

## Galerkin Equivelant Architecture (GEA)

**Local Pseudo Linear GEA**

$$\mathrm{MLP}_\theta(x) \approx \hat{K}_{\mathrm{local}}$$

$$K = \mathrm{bsr\_matrix}(\mathcal{P}_{\mathcal{E}} \hat{K}_{\mathrm{local}})$$
$$\downarrow$$
$$K = \mathrm{bsr\_matrix}(\mathcal{P}_{\mathcal{E}} \mathrm{MLP}_\theta(x))$$

**Local Pseudo Bilinear GEA**

$$\mathrm{MLP}_{\theta_1}(x) \approx B$$
$$\theta_{2,ij} + \theta_{2,ji} - \theta_{2,ii} \approx D$$

$$K = \mathrm{bsr\_matrix}(\mathcal{P}_{\mathcal{E}} B^\top D B)$$
$$\downarrow$$
$$K = \mathrm{bsr\_matrix}(\mathcal{P}_{\mathcal{E}} \mathrm{MLP}_{\theta_1}(x)^\top (\theta_2 + \theta_2^\top - \mathrm{diag}(\theta)) \mathrm{MLP}_{\theta_1}(x))$$

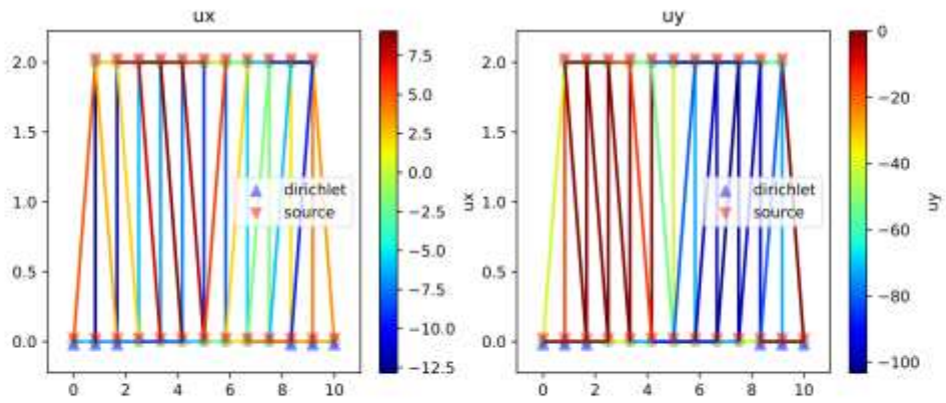
**Global GEA**

$$\text{Edge-GNN}(\tilde{x}) \approx K$$

```python
def edge_gnn_conv(x, edge_index):
    x_src = x[edge_index[0]]
    x_dst = x[edge_index[1]]
    x_edge = torch.cat([x_src, x_dst], -1)
    edge_weight = self.mlp(x_edge).squeeze() # [n_edge, 1]
    f = spmm(edge_index, edge_weight, x.shape[0], x.shape[0], u[:, None])[:, 0]
    return f
```

$$K_e = \mathrm{MLP}([x_u || x_v])$$

# Experiments



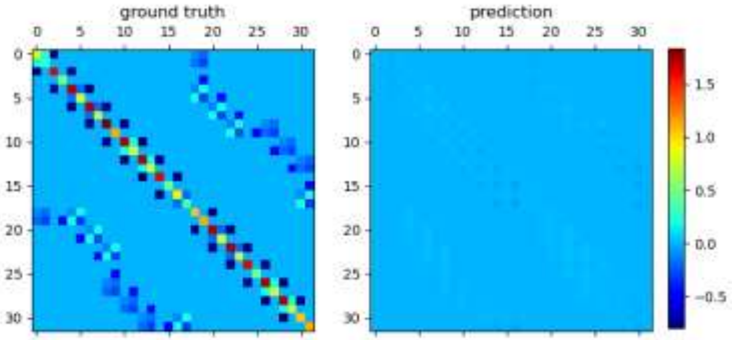

**Galerkin Equivelant Architecture (GEA)**

Local Pseudo Bilinear

$$K_{\text{local}} = \text{MLP}_{\theta_B}(x)^\top D_{\theta_D} \text{MLP}_{\theta_B}(x)$$

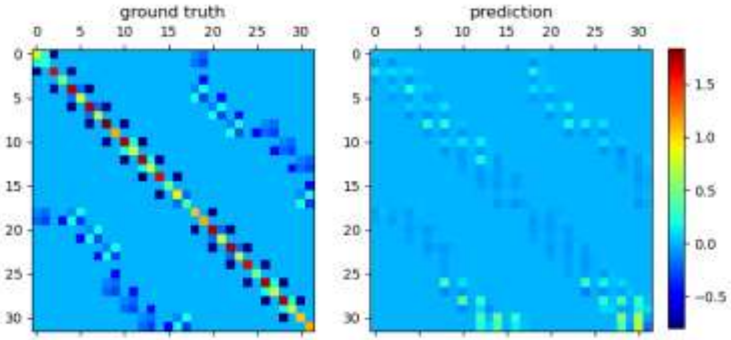
$$K_{\text{global}} = \mathcal{P} K_{\text{local}}$$

$$\ell = \|K_{\text{global}} u - f\|$$



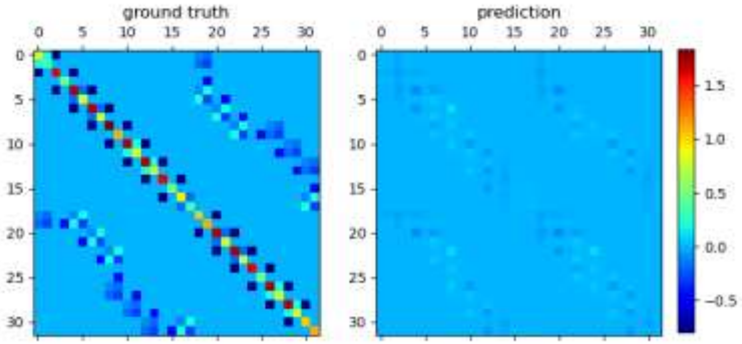

Local Pseudo Linear

$$K_{\text{local}} = \text{MLP}_{\theta}(x)$$




Global

$$K_e = \text{MLP}([x_u || x_v])$$



# Experiments

- The parameter space is too large and not smooth in matrix

- The local pseudo linear seems better

# Conclusion

- Propose Static Condensation Equivalent Architecture to find the equivalence in reverse problem
- Propose Galerkin Equivalent Architecture to find the  equivalence in forward problem
- Propose a differentiable fast assemble method
- Experiments to investigate the effectivness and generalization of the phy loss

- Physical Loss can do well on minimizing the data loss on single dataset
- Physical Loss can hardly minimizing the data loss for bunch of datasets(generalization)
- Galerkin prediction is hard to achieve, since large parameter space

# Thank you for your attention

Mingyuan Chi
Semester Project
minchi@student.ethz.ch

ETH Zürich
Department of Mathematics
Computational Science and Engineering

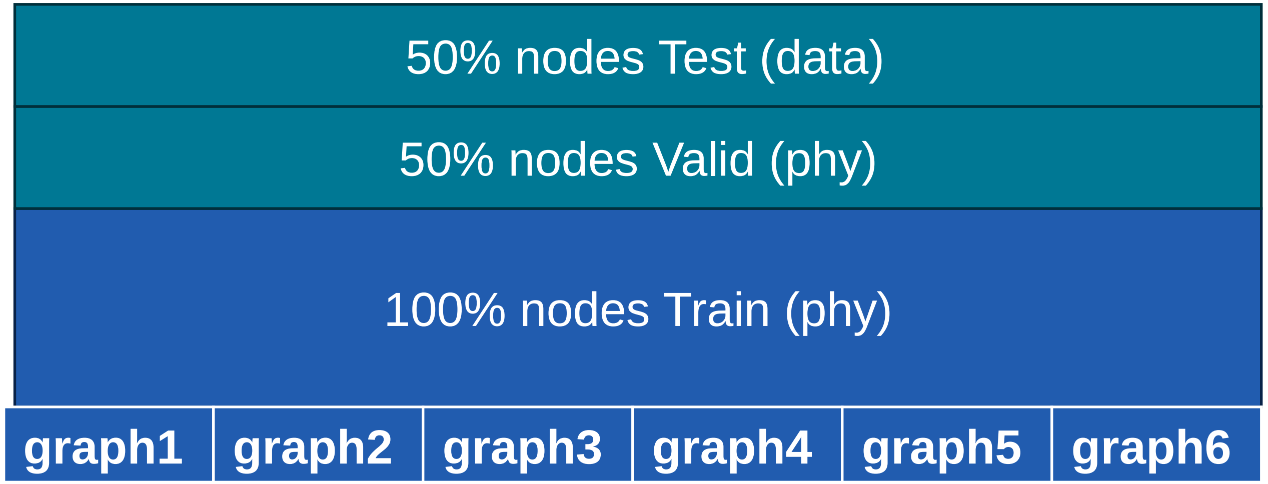
# Appendix

## Static Condense Equivelant Architecture(SCEA) : Generalization Dataset

```python
for _ in range(args.n_samples):
    for load in ["sin1","cos1", "sin2", "cos2", "sin4", "cos4"]:
        for source, boundary in [("A","B"),("A","C"),("A","D"),("A","B+C"),("A","B+D"),("A","C+D"),
                                 ("B","A"),("B","C"),("B","D"),("B","A+C"),("B","A+D"),("B","C+D"),
                                 ("C","A"),("C","B"),("C","D"),("C","A+B"),("C","A+D"),("C","B+D"),
                                 ("D","A"),("D","B"),("D","C"),("D","A+B"),("D","A+C"),("D","B+C")]:
```

```python
        for source, boundary  in [("A","B"),("A","C"),
                                  ("B","A"),("B","C"),
                                  ("C","A"),("C","B")]:
```

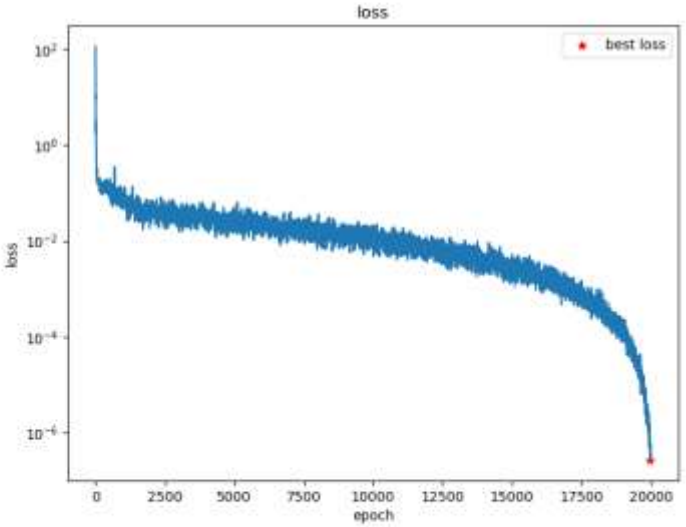
```python
parser.add_argument('--n_samples', type=int, default=4,  help="number of samples")
```



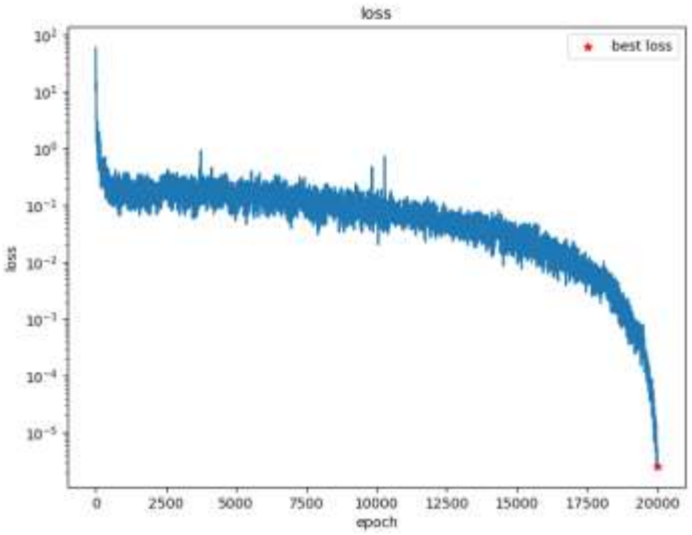
# Appendix

## Galerkin Equivelant Architecture (GEA)

Local Pseudo Bilinear




Local Pseudo Linear




Global