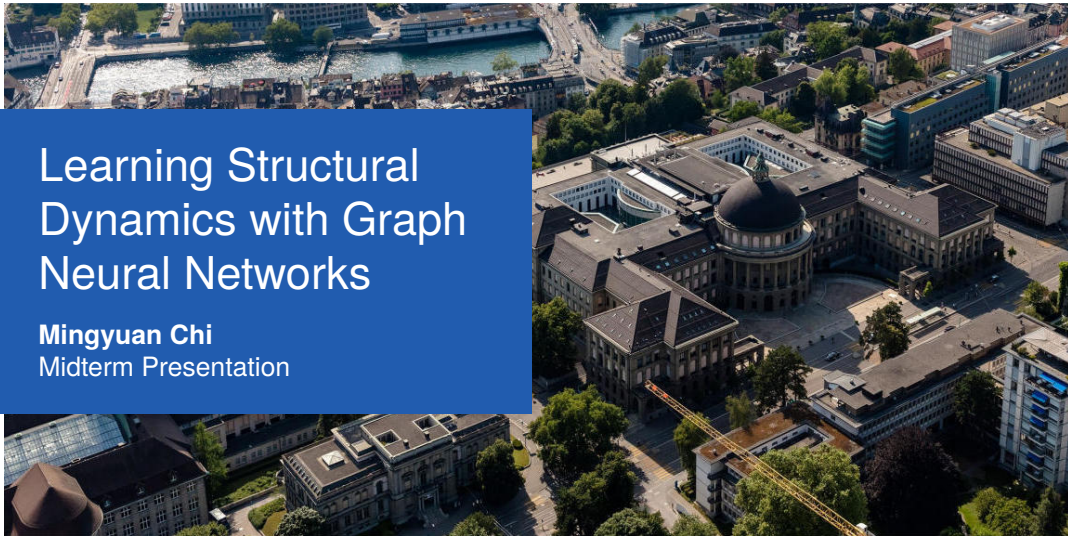


Learning Structural Dynamics with Graph Neural Networks

Mingyuan Chi
Midterm Presentation



Outline

1. Introduction
2. Background
3. Methodology
4. Experiment
5. Conclusion

Outline

1. Introduction

2. Background

3. Methodology

4. Experiment

5. Conclusion

Introduction

Goal

Description formalize the equivalence between message-passing neural networks and operations on structural matrices. The proposed research will contribute to the understanding of the underlying mechanisms of MPNNs and help develop more efficient and interpretable MPNNs as alternatives to structural simulators.

~~Review the existing methods and literature (Kick-off).~~

~~Become familiar with the PyTorch Geometric framework (already).~~

Implement a first prototype, based on existing work from the literature.

Develop a theoretical framework for formalizing the equivalence between MPNNs and structural operations.

Formulate methods for sub-structuring/condensation schemes which exploit this framework.

Demonstrate the method on a simple case study, exploring its advantages and limitations.

Outline

1. Introduction

2. Background

3. Methodology

4. Experiment

5. Conclusion

Background

Linear Elasticity

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

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

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

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

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

\hat{K}_{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}_{local} to \hat{K}_{global} , $\mathcal{P}_{\mathcal{E}} \in \mathbb{R}_{\text{sparse}}^{|\mathcal{E}| \times |\mathcal{C}| \times h \times h}$

\mathcal{C} : elements/cells

h : number of basis for each element/-cell

\mathcal{E} : connections for points

\mathcal{V} : points

Background

Project(Assemble) tensor

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

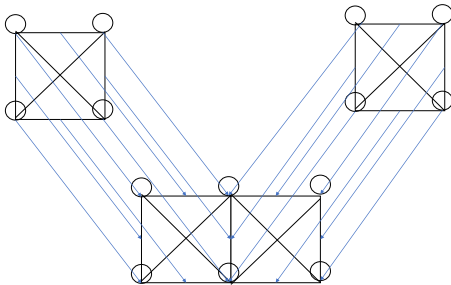


Figure: \mathcal{P} projection tensor

Background

Static Condensing

$$\begin{bmatrix} K_{II} & K_{IB} \\ K_{IB}^\top & K_{BB} \end{bmatrix} \begin{bmatrix} u_I \\ u_B \end{bmatrix} = \begin{bmatrix} f_I \\ f_B \end{bmatrix} \rightarrow K_{II}u_I = f_I - K_{IB}u_B \quad (5)$$

K is the stiffness matrix (rank-4 tensor for linear elasticity), $K \in \mathbb{R}(|\mathcal{V}| \times d) \times (|\mathcal{V}| \times d) \leftrightarrow \mathbb{R}_{\text{sparse}}^{|\mathcal{V}| \times |\mathcal{V}| \times d \times d}$

K_I is the stiffness matrix of the inner degree of freedom, $K_I \in \mathbb{R}(|\mathcal{V}_i| \times d) \times (|\mathcal{V}_i| \times d) \leftrightarrow \mathbb{R}^{|\mathcal{V}_i| \times |\mathcal{V}_i| \times d \times d}$

K_{IB} is the stiffness matrix between inner degree of freedom and boundary degree of freedom $K_{IB} \in \mathbb{R}(|\mathcal{V}_i| \times d) \times (|\mathcal{V}_b| \times d) \leftrightarrow \mathbb{R}^{|\mathcal{V}_i| \times |\mathcal{V}_b| \times d \times d}$

u is the displacement vector (rank-2 tensor), $u \in \mathbb{R}(|\mathcal{V}| \times d) \leftrightarrow \mathbb{R}^{|\mathcal{V}| \times d}$

u_B is the displacement at the boundary, $u_B \in \mathbb{R}(|\mathcal{V}_b| \times d) \leftrightarrow \mathbb{R}^{|\mathcal{V}_b| \times d}$

f is the applied force vector (rank-2 tensor), $f \in \mathbb{R}(|\mathcal{V}| \times d) \leftrightarrow \mathbb{R}^{|\mathcal{V}| \times d}$

f_I is the inner nodes applied force, $f_I \in \mathbb{R}(|\mathcal{V}_i| \times d) \leftrightarrow \mathbb{R}^{|\mathcal{V}_i| \times d}$

Outline

1. Introduction

2. Background

3. Methodology

4. Experiment

5. Conclusion

Methodology

Static Condense Equivelant Architecture(SCEA)

consider that boundary condition is homogenous(applied to all d dimension)

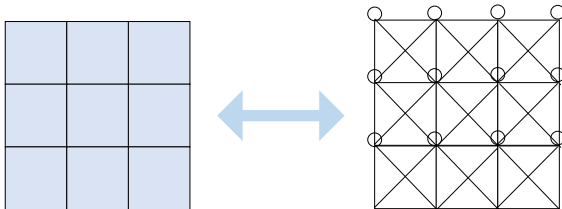


Figure: mesh2graph

Methodology

Static Condense Equivelant Architecture(SCEA)

$$\text{GNN}_{\theta_1}(A_{II}, f'_I, x) \approx K_{II}^{-1}(x_I)(f)$$

$$\text{B-GNN}_{\theta_2}(A_{IB}, u_B, x_B) \approx -K_{IB}u_B$$

$$u_I = K_{II}^{-1}(f_I - K_{IB}u_B)$$

↓

$$u_I = \text{GNN}_{\theta_1}(A_{II}, f_I + \text{B-GNN}_{\theta_2}(A_{BI}, u_B), x_I)$$

```
def b_gnn(x_src, edge_index, num_dst_nodes):  
    num_src_nodes = x_src.size(0)  
  
    x_src = self.mlp1(x_src)  
  
    adj = torch.sparse_coo_tensor(  
        edge_index,  
        torch.ones(edge_index.size(1)),  
        (num_dst_nodes, num_src_nodes),  
        dtype=x_src.dtype,  
        device=x_src.device  
    )  
  
    x_dst = adj @ x_src  
  
    x_dst = self.mlp2(x_dst)  
  
    return x_dst
```

Methodology

SCEA Architecture

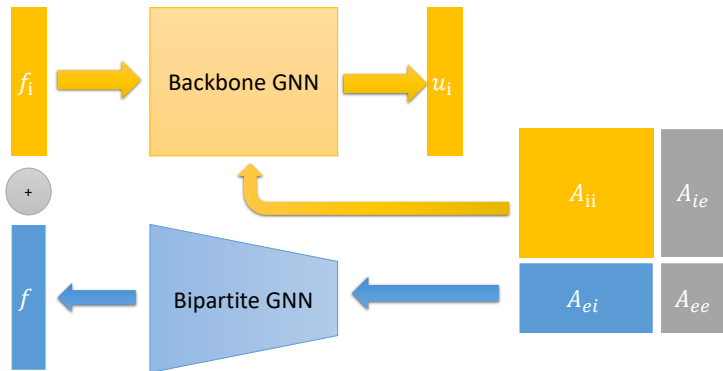


Figure: SCEA Architecture

Methodology

Galerkin Equivelant Architecture (GEA)

purpose : We want to develop a message passing neural network similar to the Finite Element Method forward process $K, u \rightarrow f$, therefore, we want to learn the forward process $\text{MPNN}_\theta(x, u) \approx Ku$

Training: $\mathcal{L} = \|K_\theta u - f\|$

with `torch.optim.lr_scheduler.CosineAnnealingLR`

Methodology

Galerkin Equivelant Architecture (GEA)

Local Pesudo Linear GEA

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

$$K = \text{bsr_matrix}(\mathcal{P}_\varepsilon \hat{K}_{\text{local}})$$

↓

$$K = \text{bsr_matrix}(\mathcal{P}_\varepsilon \text{MLP}_\theta(x))$$

Local Pesudo Bilinear GEA

$$\text{MLP}_{\theta_1}(x) \approx B$$

$$\theta_{2,ij} + \theta_{2,ji} - \theta_{2,ii} \approx D$$

$$K = \text{bsr_matrix}(\mathcal{P}_\varepsilon B^\top DB)$$

↓

$$K = \text{bsr_matrix}(\mathcal{P}_\varepsilon \text{MLP}_{\theta_1}(x)^\top (\theta_2 + \theta_2^\top - \text{diag}(\theta)) \text{MLP}_{\theta_1}(x))$$

Methodology

Galerkin Equivelant Architecture (GEA)

Global GEA

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

```
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
```

Outline

1. Introduction

2. Background

3. Methodology

4. Experiment

5. Conclusion

Experiment

Static Condense Equivelant Architecture (SCEA)

Dataset

The bottom are fixed in x and y direction.

The top are applied constant force downward.

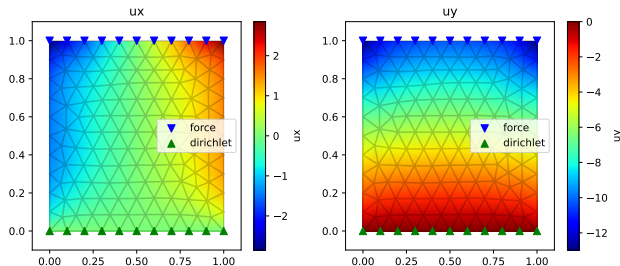


Figure: Illustration of the dataset setup.

Training

$$\mathcal{L} = \mathcal{L}_{\text{data}} + \lambda \mathcal{L}_{\text{physical}}$$

$$\mathcal{L}_{\text{data}} = \|u_{\theta} - u\|$$

$$\mathcal{L}_{\text{physical}} = \|Ku_{\theta} - f\|$$

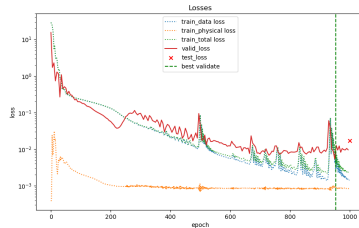
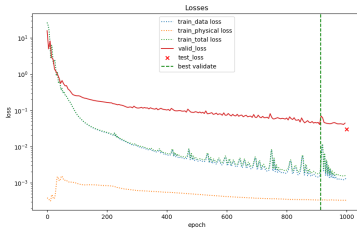
Experiment

SCEA:Loss

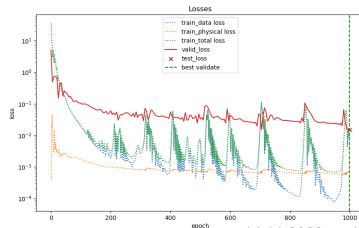
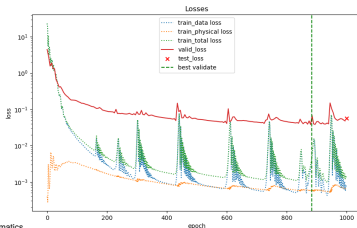
without SCEA

with SCEA

GAT



SIGN



Experiment

SCEA:Scale (In)variant Test

Scale-invariant test

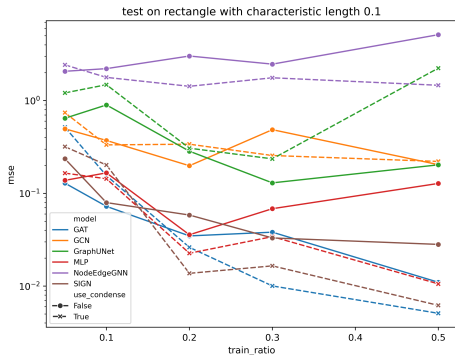


Figure: Characteristic length=0.1

Scale-variant test

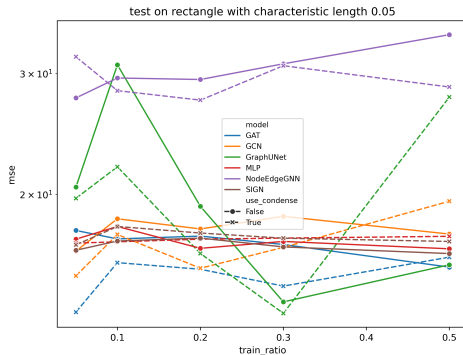


Figure: Characteristic length=0.05

Experiment

Galerkin Equivelant Architecture (GEA):Local

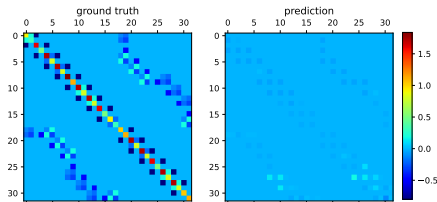
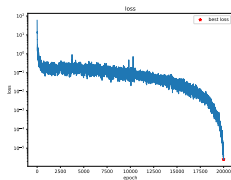
Training: $\mathcal{L} = \|K_{\theta}u - f\|$

with `torch.optim.lr_scheduler.CosineAnnealingLR`

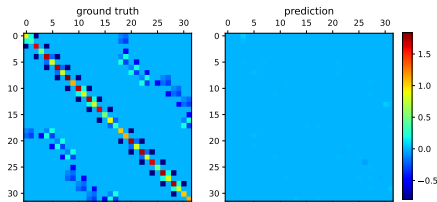
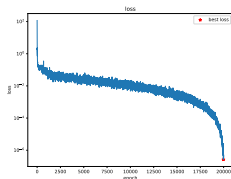
Loss

predicted galerkin

Pseudo Linear



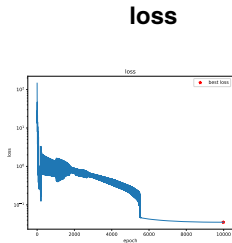
Pseudo Bilinear



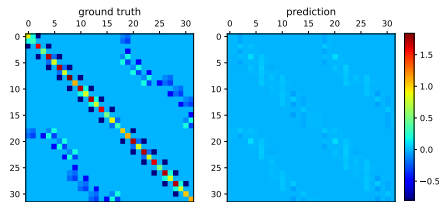
Experiment

Galerkin Equivelant Architecture (GEA):Global

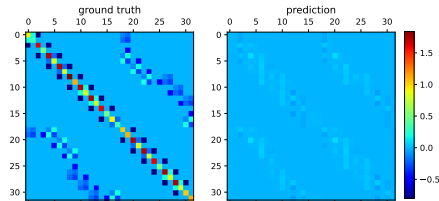
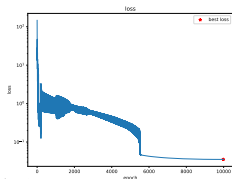
without PE



predicted galerkin



with PE



Outline

1. Introduction
2. Background
3. Methodology
4. Experiment
5. Conclusion

Conclusion

SCEA

with Static Condensing Equivelant Architecture, the model is much more robust in the learning process. The discrepancy between the $\mathcal{L}_{\text{valid}}$ an $\mathcal{L}_{\text{train}}$ is much smaller than the naive architecture.

The SCEA-GAT outperform other models when there is sufficient observation data.

When the observation data are insufficient ($< 10\%$), there is no extinguishable difference in performance for all models.

The scale-variant test shows that GraphUNet is incredibaly good when about 30% of data are available.

Conclusion

GEA

Galerkin Equivelant Architecture is hard to learn the matrix since it's linear combination. Low loss doesn't mean good accuracy $\operatorname{argmin}_{\theta} \|K_{\theta} u - f\| \neq \operatorname{argmin}_{\theta} \|K_{\theta} - K\|$

positional encoding doesn't help a lot in the training process.

Furtherwork

SCEA

- ✓ Review the existing methods and literature (Kick-off).
- ✓ Implement a first prototype, based on existing work from the literature.
- ✓ Develop a theoretical framework for formalizing the equivalence between MPNNs and structural operations.
- ✓ Formulate methods for sub-structuring/condensation schemes which exploit this framework.
- ✓ Demonstrate the method on a simple case study, exploring its advantages and limitations.
- ☐ Tune the training parameters to make the loss lower, currently the loss is too large.
- ☐ Test for different boundary condition and same mesh.
- ☐ Plot the best prediction result with the ground truth.

Furtherwork

GEA

- ☐ Review the existing methods and literature.
- ✓ Develop multi theoretical framework for formalizing the equivalence between MPNNs and forward Galerkin.
- ✓ Formulate methods for forward problem.
- ☐ Try to achieve a good result from these methods.

Thanks for your attention

D MATH

Mingyuan Chi
minchi@student.ethz.ch