Fast Graph Representation Learning with PyTorch geometric

tu dortmund university

SFB 876 Providing Information by Resource-Constrained Data Analysis

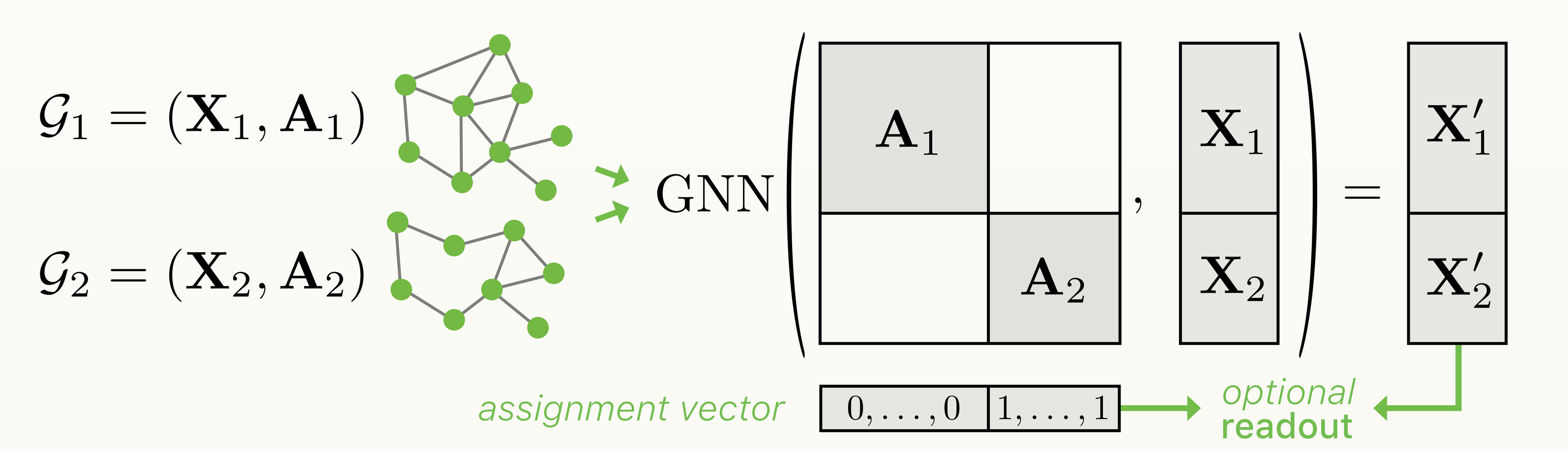
Matthias Fey
Jan Eric Lenssen

A **PyTorch** Extension Library for Deep Learning on **Graphs, Point Clouds** and **Manifolds**

- ✓ uniform implementations of over **25 GNN operators/models** →
- ✓ extendable via a simple **Message Passing interface**
- ✓ access to over **100 benchmark datasets**
- ✓ **dynamic** batch-wise **graph generation**
- ✓ deterministic and differentiable **pooling operators**
- ✓ basic as well as more sophisticated **readout functions**
- ✓ **automatic mini-batching for graphs with different sizes**
- ✓ useful transforms for **augmentation, point sampling**, ...
- ✓ leverages dedicated **CUDA kernels**
- ✓ supports **multi-GPUs**
- ✓ thoroughly **documented**

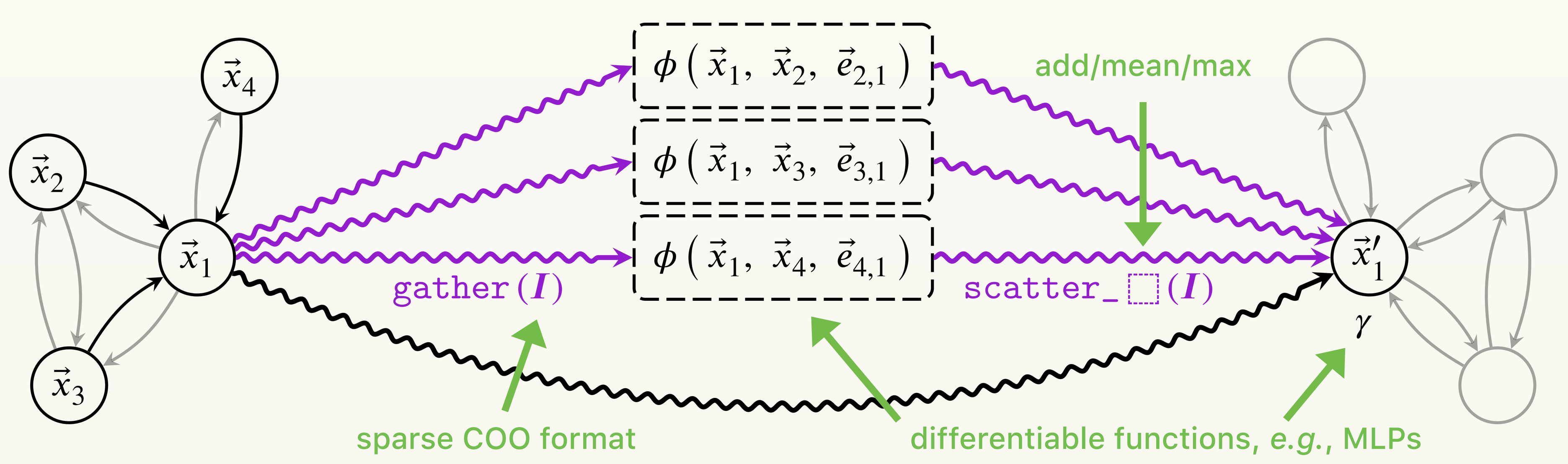
```python
class MyOwnNet(Module):
    def __init__(self, in_channels, out_channels):
        self.conv1 = GCNConv(in_channels, 16)
        self.conv2 = GCNConv(16, out_channels)

    def forward(self, x, edge_index):
        x = relu(self.conv1(x, edge_index))
        return softmax(self.conv2(x, edge_index))
```



An **Intuitive Message Passing Interface** based on **Gather and Scatter Operations**

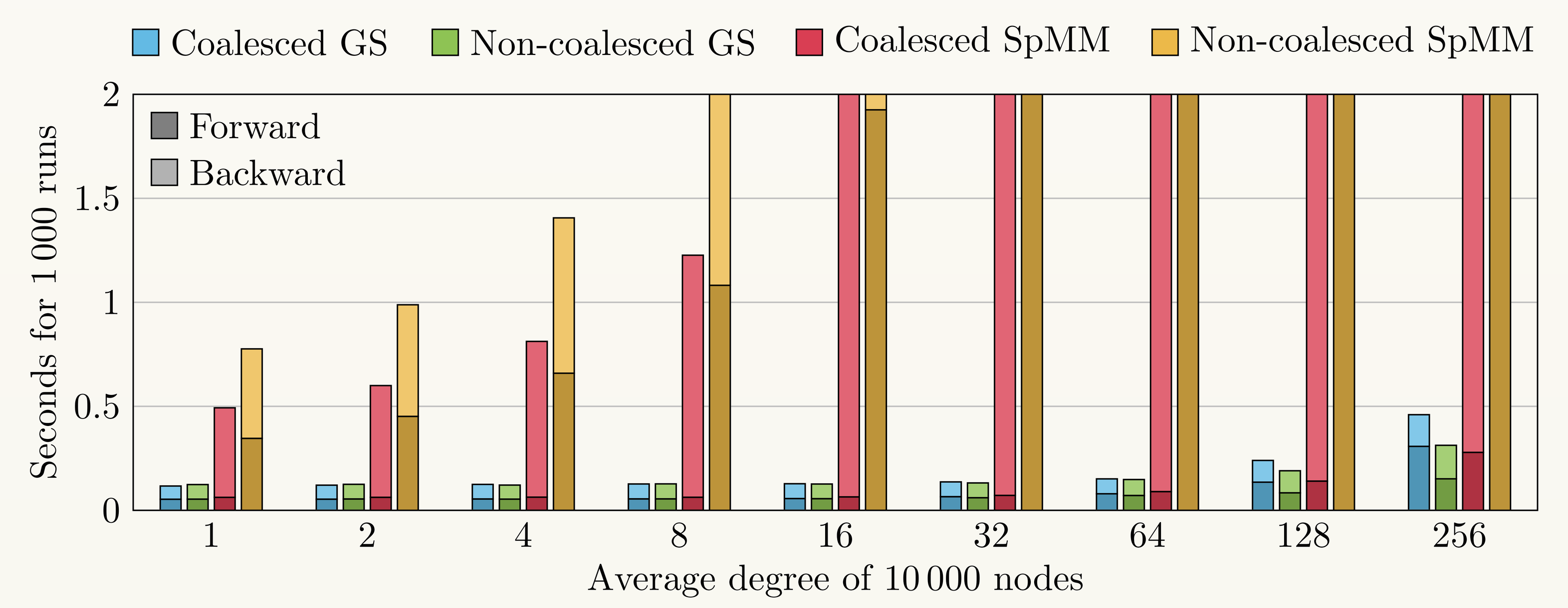
$$\vec{x}_i' = \gamma\left(\vec{x}_i, \underset{j \in \mathcal{N}(i)}{\square} \phi\left(\vec{x}_i, \vec{x}_j, \vec{e}_{j,i}\right)\right)$$




```python
class MyOwnConv(MessagePassing):
    def __init__(self, ...):
        super(MyOwnConv, self).__init__('add')

    def forward(self, x, edge_index, e=None):
        return self.propagate(edge_index, x=x, e=e)

    def message(self, x_i, x_j, e):
```



Sparse-Matrix Multiplication (SpMM) needs *coalesced* sparse tensors → backward pass is inherently slow

Gather/Scatter (GS):

- ✓ input does not need to be coalesced
- ✓ can integrate central node and edge information
- × non-deterministic by nature on GPU

An **easy-to-use Testbed** for **Evaluating new Research Ideas** with **Competitive Runtimes**

Method	Cora Fixed	Cora Random	CiteSeer Fixed	CiteSeer Random	PubMed Fixed	PubMed Random
GCN	81.5 ± 0.6	79.4 ± 1.9	71.1 ± 0.7	68.1 ± 1.7	79.0 ± 0.6	77.4 ± 2.4
GAT	83.1 ± 0.4	81.0 ± 1.4	70.8 ± 0.5	69.2 ± 1.9	78.5 ± 0.3	78.3 ± 2.3
SGC	81.7 ± 0.1	80.2 ± 1.6	71.3 ± 0.2	68.7 ± 1.6	78.9 ± 0.1	76.5 ± 2.4
ARMA	82.8 ± 0.6	80.7 ± 1.4	**72.3 ± 1.1**	68.9 ± 1.6	78.8 ± 0.3	77.7 ± 2.6
APPNP	**83.3 ± 0.5**	**82.2 ± 1.5**	71.8 ± 0.5	**70.0 ± 1.4**	**80.1 ± 0.2**	**79.4 ± 2.2**

	Method	MUTAG	PROTEINS	COLLAB	IMDB-BINARY	REDDIT-BINARY
Flat	GCN	74.6 ± 7.7	73.1 ± 3.8	**80.6 ± 2.1**	72.6 ± 4.5	89.3 ± 3.3
Flat	SAGE	74.9 ± 8.7	**73.8 ± 3.6**	79.7 ± 1.7	72.4 ± 3.6	89.1 ± 1.9
Flat	GIN-0	**85.7 ± 7.7**	72.1 ± 5.1	79.3 ± 2.7	**72.8 ± 4.5**	89.6 ± 2.6
Flat	GIN-ϵ	83.4 ± 7.5	72.6 ± 4.9	79.8 ± 2.4	72.1 ± 5.1	**90.3 ± 3.0**
Hier.	Graclus	77.1 ± 7.2	73.0 ± 4.1	79.6 ± 2.0	72.2 ± 4.2	88.8 ± 3.2
Hier.	top$_k$	76.3 ± 7.5	72.7 ± 4.1	**79.7 ± 2.2**	72.5 ± 4.6	87.6 ± 2.4
Hier.	DiffPool	**85.0 ± 10.3**	**75.1 ± 3.5**	78.9 ± 2.3	**72.6 ± 3.9**	**92.1 ± 2.6**

Runtimes of 200 epoch training procedures on a single GPU in comparison to the **DeepGraphLibrary**:

Dataset	Method	DGL DB	DGL GS	PyG
Cora	GCN	4.19s	0.32s	**0.25s**
Cora	GAT	6.31s	5.36s	**0.80s**
CiteSeer	GCN	3.78s	0.34s	**0.30s**
CiteSeer	GAT	5.61s	4.91s	**0.88s**
PubMed	GCN	12.91s	0.36s	**0.32s**
PubMed	GAT	18.69s	13.76s	**2.42s**
MUTAG	RGCN	18.81s	2.40s	**2.14s**

- ✓ much faster than the **inherently sequential Degree Bucketing (DB) approach**
- ✓ major GAT runtime improvements due to own **optimized sparse softmax kernels**

/rusty1s/pytorch_geometric

license MIT | PRs welcome

New features to come. Stay tuned! 😎