Graph Neural Networks in TensorFlow and Keras with

graphneural.network

Built for TensorFlow

- Training loop.
- Distributed training.
- GPU/TPU.
- Industry standard.

Message passing layers

GraphConv *Kipf & Welling*	**ChebConv** *Defferrard et al.*	**GraphSageConv** *Hamilton et al.*	**ARMAConv** *Bianchi et al.*
ECConv *Simonovsky & Komodakis*	**GraphAttention** *Velickovic et al.*	**GraphConvSkip** *Bianchi et al.*	**APPNP** *Klicpera et al.*
GINConv *Xu et al.*	**DiffusionConv** *Li et al.*	**GatedGraphConv** *Li et al.*	**AGNNConv** *Thekumparampil et al.*
TAGConv *Du et al.*	**CrystalConv** *Xie & Grossman*	**EdgeConv** *Wang et al.*	**MessagePassing** *Gilmer et al.*

Pooling layers

DiffPool
Ying et al.

MinCutPool
Bianchi et al.

TopKPool
Gao & Ji

SAGPool
Lee et al.

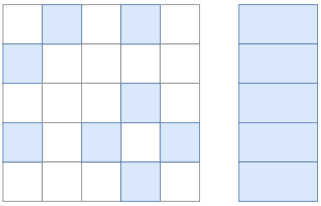
Plus 6 global pooling / readout layers.

Other features

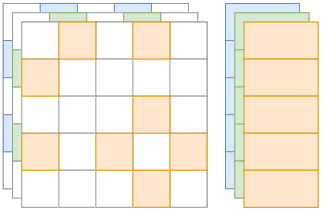
- Benchmark datasets (citation nets, TUD, QM9, OGB, ...).
- Utils to create new layers.
- Transparent support for **data modes**.

Data Modes

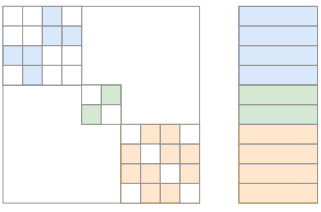
Single



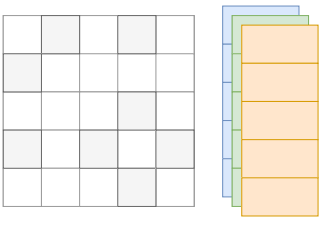
Batch



Disjoint



Mixed



Execution times

Model	Dataset	PyG	Spektral	Change
GCN	Cora	0.332s ±0.002	**0.183s** ±0.002	**- 44.9%**
	Citeseer	0.488s ±0.009	**0.396s** ±0.011	**- 18.8%**
	Pubmed	0.834s ±0.004	**0.683s** ±0.001	**- 18.1%**
ChebNet	Cora	4.690s ±0.007	**2.059s** ±0.008	**- 56.0%**
	Citeseer	11.441s ±0.165	**5.470s** ±0.006	**- 52.2%**
	Pubmed	12.517s ±0.148	**6.221s** ±0.004	**- 50.2%**
GAT	Cora	**1.527s** ±0.002	2.042s ±0.074	+ 33.7%
	Citeseer	**2.032s** ±0.003	3.427s ±0.085	+ 68.6%
	Pubmed	**7.427s** ±0.014	10.63s ±0.132	+ 43.1%

Code example

Declarative API

```
X_in = Input(shape=(F_in, ))
A_in = Input((N, ), sparse=True)

X_1 = GraphConv(16, 'relu')([X_in, A_in])
X_1 = Dropout(0.5)(X_1)
X_2 = GraphConv(F_out, 'softmax')([X_1, A_in])

net = Model(inputs=[X_in, A_in], outputs=X_2)
```

Imperative API

```
class Net(Model):
    def __init__(self, F_out, **kwargs):
        super().__init__(**kwargs)
        self.conv1 = GraphConv(16, 'relu')
        self.conv2 = GraphConv(F_out, 'softmax')
        self.dropout = Dropout(0.5)

    def call(self, inputs):
        X, A = inputs
        X_1 = self.conv1([X, A])
        X_1 = self.dropout(X_1)
        X_2 = self.conv2([X_1, A])

        return X_2
```

Training: `net.fit(x, y)`

Conclusion

- TensorFlow + Keras.
- Lots of features.
- Data modes.
- Fast.
- Easy.

Docs: `graphneural.network`
Code: `github.com/danielegrattarola/spektral`
Paper: `arxiv.org/abs/2006.12138`