

Graph Neural Networks (GNN)

Theory & Applications

*Andrea Agiollo**

*Dipartimento di Informatica – Scienza e Ingegneria (DISI)
ALMA MATER STUDIORUM – Università di Bologna
andrea.agiollo2@unibo.it

Internal Workshop on
Graph Neural Networks
April 30, 2021, Bologna



Next in Line...

- 1 Background
- 2 Graph Neural Networks (GNNs)
- 3 Applications



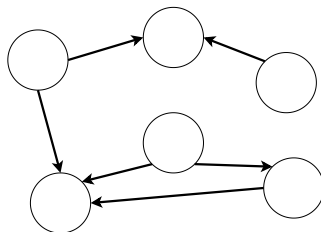
Focus on. . .

- 1 Background
 - **Graphs**
 - Neural Networks
- 2 Graph Neural Networks (GNNs)
 - Introduction
 - Spectral-based Convolutional GNN
 - Spatial Convolution
 - Spatial Convolution with Recurrent Operators
 - Spatial-based Convolutional GNN
 - Spectral-based vs Spatial-based CGNN
- 3 Applications
 - Nodes Classification
 - Graph Classification
 - Link Prediction
 - Graph Generation



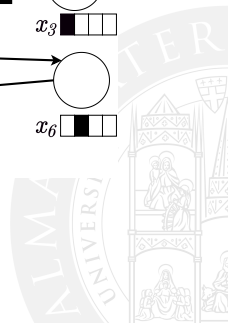
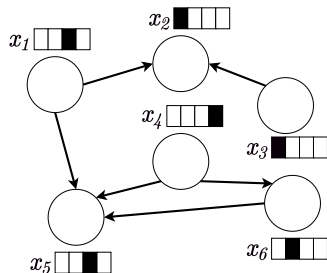
Graphs

- Nodes/Vertices
- Edges
- $G = (V, E)$



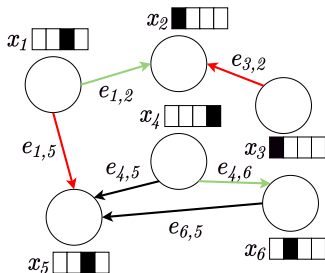
Graphs

- Nodes/Vertices
- Edges
- $G = (V, E)$
- Node feature vector $x_i \in \mathcal{R}^d$



Graphs

- Nodes/Vertices
- Edges
- $G = (V, E)$
- Node feature vector $x_i \in \mathcal{R}^d$
- Edge feature vector $e_{i,j} \in \mathcal{R}^l$



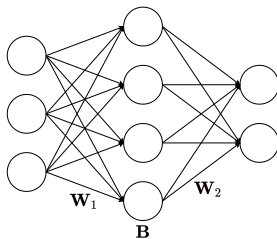
Focus on. . .

- 1 Background
 - Graphs
 - **Neural Networks**
- 2 Graph Neural Networks (GNNs)
 - Introduction
 - Spectral-based Convolutional GNN
 - Spatial Convolution
 - Spatial Convolution with Recurrent Operators
 - Spatial-based Convolutional GNN
 - Spectral-based vs Spatial-based CGNN
- 3 Applications
 - Nodes Classification
 - Graph Classification
 - Link Prediction
 - Graph Generation



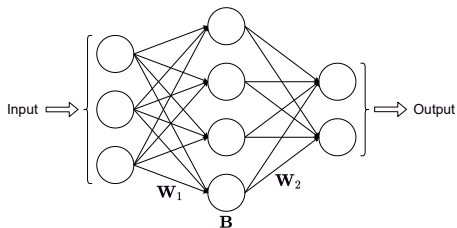
Neural Networks

- Feed forward models that maps input to proper output → Complex non linear function approximators.



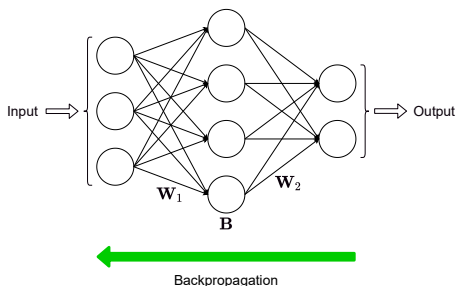
Neural Networks

- Feed forward models that maps input to proper output $\rightarrow$ Complex non linear function approximators.
- Each layer maps input to output with a function f depending on the layer type.



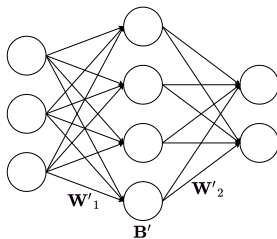
Neural Networks

- Feed forward models that maps input to proper output $\rightarrow$ Complex non linear function approximators.
- Each layer maps input to output with a function f depending on the layer type.
- Learnable parameters of the NN are optimized via backpropagation.



Neural Networks

- Feed forward models that maps input to proper output $\rightarrow$ Complex non linear function approximators.
- Each layer maps input to output with a function f depending on the layer type.
- Learnable parameters of the NN are optimized via backpropagation.



Next in Line...

- 1 Background
- 2 Graph Neural Networks (GNNs)
- 3 Applications



Focus on. . .

- 1 Background
 - Graphs
 - Neural Networks
- 2 Graph Neural Networks (GNNs)
 - **Introduction**
 - Spectral-based Convolutional GNN
 - Spatial Convolution
 - Spatial Convolution with Recurrent Operators
 - Spatial-based Convolutional GNN
 - Spectral-based vs Spatial-based CGNN
- 3 Applications
 - Nodes Classification
 - Graph Classification
 - Link Prediction
 - Graph Generation



Idea

Question: Can NN be applied to graph structured data?



Idea

Question: Can NN be applied to graph structured data?

- ✓ Data are always data



Idea

Question: Can NN be applied to graph structured data?

- ✓ Data are always data
- ✓ Nodes embeddings corresponds to pixel values in images



Idea

Question: Can NN be applied to graph structured data?

- ✓ Data are always data
- ✓ Nodes embeddings corresponds to pixel values in images
- ✗ Graphs may vary significantly in size and structure



Idea

Question: Can NN be applied to graph structured data?

- ✓ Data are always data
- ✓ Nodes embeddings corresponds to pixel values in images
- ✗ Graphs may vary significantly in size and structure
- ✗ Edges density variability makes everything more complex



Idea

Question: Can NN be applied to graph structured data?

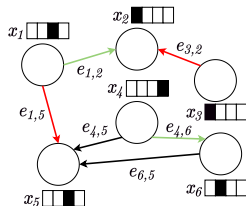
- ✓ Data are always data
- ✓ Nodes embeddings corresponds to pixel values in images
- ✗ Graphs may vary significantly in size and structure
- ✗ Edges density variability makes everything more complex

Answer: Yes, with some adjustments...



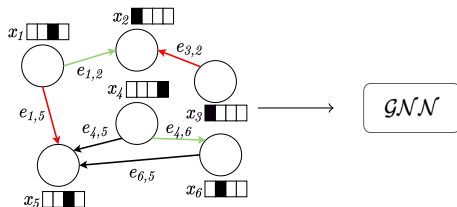
What are GNNs?

1. Start from graph structure.



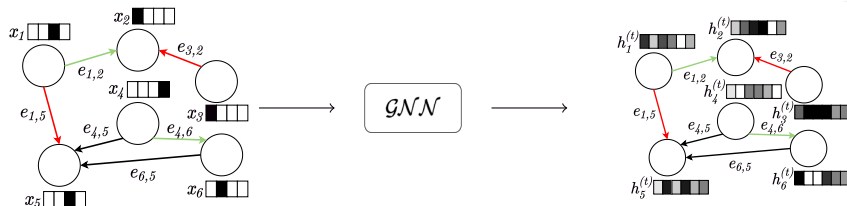
What are GNNs?

1. Start from graph structure.
2. Apply GNN to the graph.



What are GNNs?

1. Start from graph structure.
2. Apply GNN to the graph.
3. Obtain graph with meaningful representation (**hidden state h**) of nodes w.r.t. the whole graph and the task under consideration.



What are GNNs similar to?

- The target of GNN is to learn a state embedding $h_i \forall i \in [0, \dots, n]$ which contains information of the node and its role in the neighbourhood and in the graph.



What are GNNs similar to?

- The target of GNN is to learn a state embedding $h_i \forall i \in [0, \dots, n]$ which contains information of the node and its role in the neighbourhood and in the graph.
- Similar to Neural Networks embedding techniques in Natural Language Processing (Word2Vec [8], BERT [5], etc.).



What are GNNs similar to?

- The target of GNN is to learn a state embedding $h_i \forall i \in [0, \dots, n]$ which contains information of the node and its role in the neighbourhood and in the graph.
- Similar to Neural Networks embedding techniques in Natural Language Processing (Word2Vec [8], BERT [5], etc.).
- Embedding learning is only part of the whole pipeline. Learnt embedding is usually followed by simple classification/regression techniques.



Focus on. . .

- 1 Background
 - Graphs
 - Neural Networks
- 2 Graph Neural Networks (GNNs)
 - Introduction
 - **Spectral-based Convolutional GNN**
 - Spatial Convolution
 - Spatial Convolution with Recurrent Operators
 - Spatial-based Convolutional GNN
 - Spectral-based vs Spatial-based CGNN
- 3 Applications
 - Nodes Classification
 - Graph Classification
 - Link Prediction
 - Graph Generation



Intuition

Idea: Treat graph as a signal $x \in \mathcal{R}^n$. It is a feature vector of all nodes in a graph where x_i is the value of the i^{th} node.



Intuition

Idea: Treat graph as a signal $x \in \mathcal{R}^n$. It is a feature vector of all nodes in a graph where x_i is the value of the i^{th} node.

Laplacian matrix of graph [1] plays a similar role as the Fourier basis in signal processing.



Intuition

Idea: Treat graph as a signal $x \in \mathcal{R}^n$. It is a feature vector of all nodes in a graph where x_i is the value of the i^{th} node.

Laplacian matrix of graph [1] plays a similar role as the Fourier basis in signal processing.

$$\mathbf{L} = \mathbf{I}_n - \mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2}$$

where:

- $\mathbf{I}_n$ is the identity matrix in $\mathcal{R}^n$.
- $\mathbf{A}$ is the adjacency matrix of the graph.
- $\mathbf{D}$ is a diagonal matrix of node degrees such that $\mathbf{D}_{ii} = \sum_j \mathcal{A}_{i,j}$.



Intuition

Idea: Treat graph as a signal $x \in \mathcal{R}^n$. It is a feature vector of all nodes in a graph where x_i is the value of the i^{th} node.

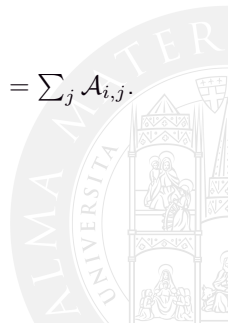
Laplacian matrix of graph [1] plays a similar role as the Fourier basis in signal processing.

$$\mathbf{L} = \mathbf{I}_n - \mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2}$$

where:

- $\mathbf{I}_n$ is the identity matrix in $\mathcal{R}^n$.
- $\mathbf{A}$ is the adjacency matrix of the graph.
- $\mathbf{D}$ is a diagonal matrix of node degrees such that $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{i,j}$.

Laplacian matrix can be factorized into:



Intuition

Idea: Treat graph as a signal $x \in \mathcal{R}^n$. It is a feature vector of all nodes in a graph where x_i is the value of the i^{th} node.

Laplacian matrix of graph [1] plays a similar role as the Fourier basis in signal processing.

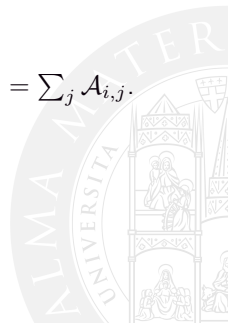
$$\mathbf{L} = \mathbf{I}_n - \mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2}$$

where:

- $\mathbf{I}_n$ is the identity matrix in $\mathcal{R}^n$.
- $\mathbf{A}$ is the adjacency matrix of the graph.
- $\mathbf{D}$ is a diagonal matrix of node degrees such that $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{i,j}$.

Laplacian matrix can be factorized into:

$$\mathbf{L} = \mathbf{U} \mathbf{\Lambda} \mathbf{U}^T$$



Intuition

Idea: Treat graph as a signal $x \in \mathcal{R}^n$. It is a feature vector of all nodes in a graph where x_i is the value of the i^{th} node.

Laplacian matrix of graph [1] plays a similar role as the Fourier basis in signal processing.

$$\mathbf{L} = \mathbf{I}_n - \mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2}$$

where:

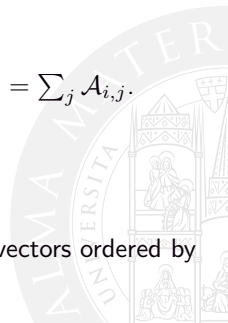
- $\mathbf{I}_n$ is the identity matrix in $\mathcal{R}^n$.
- $\mathbf{A}$ is the adjacency matrix of the graph.
- $\mathbf{D}$ is a diagonal matrix of node degrees such that $\mathbf{D}_{ii} = \sum_j \mathcal{A}_{i,j}$.

Laplacian matrix can be factorized into:

$$\mathbf{L} = \mathbf{U} \mathbf{\Lambda} \mathbf{U}^T$$

where:

- $\mathbf{U} = [u_0, u_1, \dots, u_{n-1}] \in \mathcal{R}^{n \times n}$ is the matrix of eigenvectors ordered by eigenvalues.
- $\mathbf{\Lambda}$ is the diagonal matrix of eigenvalues.



Intuition

It is possible to compute the graph Fourier transform of a signal (graph) x and its inverse:

$$\mathfrak{F}(x) = \hat{x} = \mathbf{U}^T x$$

$$\mathfrak{F}^{-1}(\hat{x}) = x = \mathbf{U} \hat{x}$$



Intuition

It is possible to compute the graph Fourier transform of a signal (graph) x and its inverse:

$$\mathfrak{F}(x) = \hat{x} = \mathbf{U}^T x$$

$$\mathfrak{F}^{-1}(\hat{x}) = x = \mathbf{U} \hat{x}$$

A filter $g \in \mathcal{R}^n$ can now be applied to the graph using convolution:

$$x * g = \mathfrak{F}^{-1}(\mathfrak{F}(x) \odot \mathfrak{F}(g)) = \mathbf{U}(\mathbf{U}^T x \odot \mathbf{U}^T g)$$

where:

- $*$ denotes the convolution operation
- $\odot$ denotes element-wise multiplication



Intuition

It is possible to compute the graph Fourier transform of a signal (graph) x and its inverse:

$$\mathfrak{F}(x) = \hat{x} = \mathbf{U}^T x$$

$$\mathfrak{F}^{-1}(\hat{x}) = x = \mathbf{U} \hat{x}$$

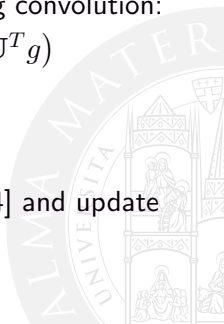
A filter $g \in \mathcal{R}^n$ can now be applied to the graph using convolution:

$$x * g = \mathfrak{F}^{-1}(\mathfrak{F}(x) \odot \mathfrak{F}(g)) = \mathbf{U}(\mathbf{U}^T x \odot \mathbf{U}^T g)$$

where:

- $*$ denotes the convolution operation
- $\odot$ denotes element-wise multiplication

Idea: consider g as a set of learnable parameters [2, 4] and update the graph signal $h = x * g$.



Issues

- $\mathbf{L}$ can be computed only for undirected graphs
→ information loss.



Issues

- $\mathbf{L}$ can be computed only for undirected graphs
→ information loss.
- To compute $\mathbf{L}$ the whole graph knowledge is required → does not scale well.



Issues

- $\mathbf{L}$ can be computed only for undirected graphs
→ information loss.
- To compute $\mathbf{L}$ the whole graph knowledge is required → does not scale well.
- Eigenvector computation is required → inefficient.



Issues

- $\mathbf{L}$ can be computed only for undirected graphs
→ information loss.
- To compute $\mathbf{L}$ the whole graph knowledge is required → does not scale well.
- Eigenvector computation is required → inefficient.
- Eigenvector may change rapidly between different graph structures → no generalization to multiple graphs.



Issues

- $\mathbf{L}$ can be computed only for undirected graphs
→ information loss.
- To compute $\mathbf{L}$ the whole graph knowledge is required → does not scale well.
- Eigenvector computation is required → inefficient.
- Eigenvector may change rapidly between different graph structures → no generalization to multiple graphs.
- Number of nodes in graph is required to be fixed → can not handle heterogeneous graphs.



Issues

- $\mathbf{L}$ can be computed only for undirected graphs
→ information loss.
- To compute $\mathbf{L}$ the whole graph knowledge is required → does not scale well.
- Eigenvector computation is required → inefficient.
- Eigenvector may change rapidly between different graph structures → no generalization to multiple graphs.
- Number of nodes in graph is required to be fixed → can not handle heterogeneous graphs.



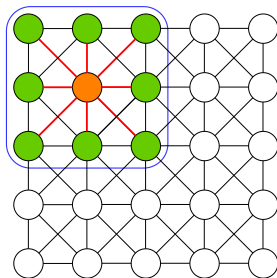
Focus on. . .

- 1 Background
 - Graphs
 - Neural Networks
- 2 Graph Neural Networks (GNNs)
 - Introduction
 - Spectral-based Convolutional GNN
 - **Spatial Convolution**
 - Spatial Convolution with Recurrent Operators
 - Spatial-based Convolutional GNN
 - Spectral-based vs Spatial-based CGNN
- 3 Applications
 - Nodes Classification
 - Graph Classification
 - Link Prediction
 - Graph Generation



Convolution in images

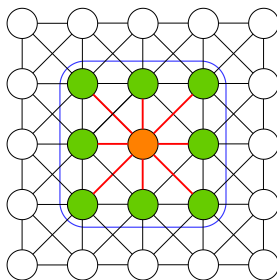
Convolutional filters applied to images aggregate information of pixels close to each other.



Convolution in images

Convolutional filters applied to images aggregate information of pixels close to each other.

Images are graphs where nodes (pixels) have a fixed set of neighbours.

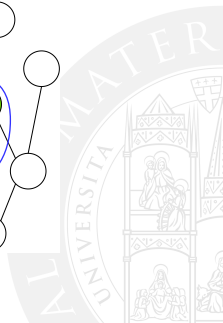
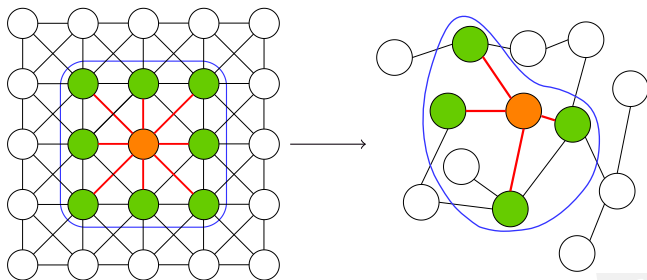


Convolution in images

Convolutional filters applied to images aggregate information of pixels close to each other.

Images are graphs where nodes (pixels) have a fixed set of neighbours.

Can we aggregate information over a variable set of neighbours?

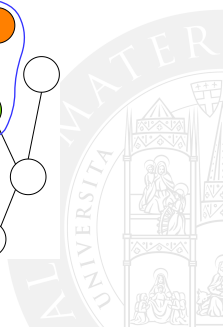
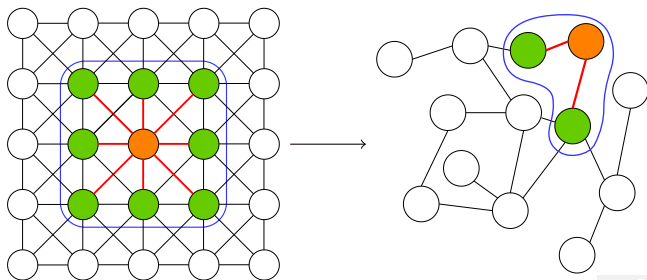


Convolution in images

Convolutional filters applied to images aggregate information of pixels close to each other.

Images are graphs where nodes (pixels) have a fixed set of neighbours.

Can we aggregate information over a variable set of neighbours?



Intuition

Idea 1: Nodes in a graph constantly exchange information with neighbouring nodes until a stable equilibrium is reached.



Intuition

Idea 1: Nodes in a graph constantly exchange information with neighbouring nodes until a stable equilibrium is reached.



Message passing

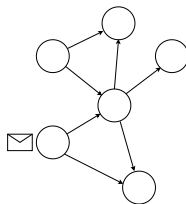


Intuition

Idea 1: Nodes in a graph constantly exchange information with neighbouring nodes until a stable equilibrium is reached.



Message passing

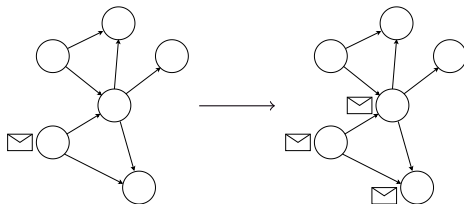


Intuition

Idea 1: Nodes in a graph constantly exchange information with neighbouring nodes until a stable equilibrium is reached.



Message passing

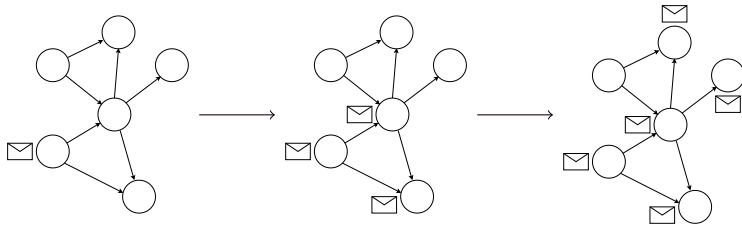


Intuition

Idea 1: Nodes in a graph constantly exchange information with neighbouring nodes until a stable equilibrium is reached.



Message passing



Intuition

Idea 2: Information received from neighbourhood is aggregated at node level with local information.



Intuition

Idea 2: Information received from neighbourhood is aggregated at node level with local information.



Information aggregation

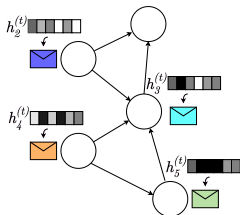


Intuition

Idea 2: Information received from neighbourhood is aggregated at node level with local information.



Information aggregation

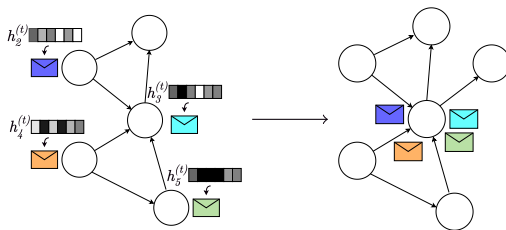


Intuition

Idea 2: Information received from neighbourhood is aggregated at node level with local information.



Information aggregation

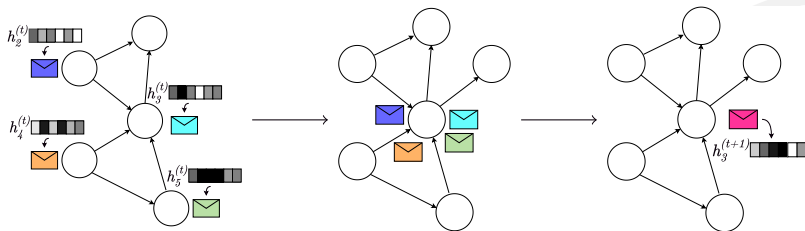


Intuition

Idea 2: Information received from neighbourhood is aggregated at node level with local information.



Information aggregation



Intuition $\rightarrow$ math

$$h_i^{(t)} = \gamma^{(t)} \left(h_i^{(t-1)}, \bigoplus_{j \in N(i)} \phi^{(t)} \left(h_i^{(t-1)}, h_j^{(t-1)}, e_{(j,i)} \right) \right)$$

where:

- h_i^t defines the node v hidden state at time t
- $N(i)$ defines the set of nodes that are neighbours of i
- $e_{(j,i)}$ is the feature vector of the edge between j and i .

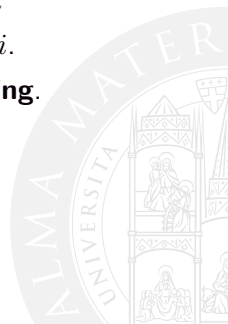


Intuition $\rightarrow$ math

$$h_i^{(t)} = \gamma^{(t)} \left(h_i^{(t-1)}, \bigoplus_{j \in N(i)} \phi^{(t)} \left(h_i^{(t-1)}, h_j^{(t-1)}, e_{(j,i)} \right) \right)$$

where:

- h_i^t defines the node v hidden state at time t
- $N(i)$ defines the set of nodes that are neighbours of i
- $e_{(j,i)}$ is the feature vector of the edge between j and i .
- ϕ denotes a differentiable function $\rightarrow$ **Message passing**.

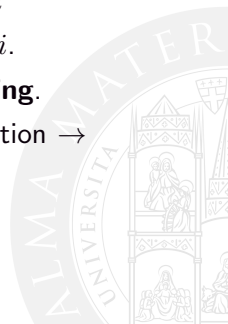


Intuition $\rightarrow$ math

$$h_i^{(t)} = \gamma^{(t)} \left(h_i^{(t-1)}, \bigoplus_{j \in N(i)} \phi^{(t)} \left(h_i^{(t-1)}, h_j^{(t-1)}, e_{(j,i)} \right) \right)$$

where:

- h_i^t defines the node v hidden state at time t
- $N(i)$ defines the set of nodes that are neighbours of i
- $e_{(j,i)}$ is the feature vector of the edge between j and i .
- ϕ denotes a differentiable function $\rightarrow$ **Message passing**.
- $\oplus$ denotes a differentiable, permutation invariant function $\rightarrow$ **Aggregation**.

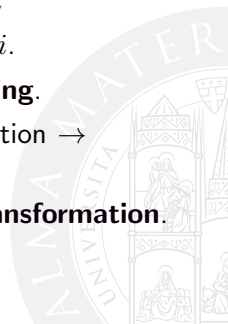


Intuition $\rightarrow$ math

$$h_i^{(t)} = \gamma^{(t)} \left(h_i^{(t-1)}, \bigoplus_{j \in N(i)} \phi^{(t)} \left(h_i^{(t-1)}, h_j^{(t-1)}, e_{(j,i)} \right) \right)$$

where:

- h_i^t defines the node v hidden state at time t
- $N(i)$ defines the set of nodes that are neighbours of i
- $e_{(j,i)}$ is the feature vector of the edge between j and i .
- ϕ denotes a differentiable function $\rightarrow$ **Message passing**.
- $\bigoplus$ denotes a differentiable, permutation invariant function $\rightarrow$ **Aggregation**.
- γ denote a differentiable function $\rightarrow$ **Information transformation**.



Focus on. . .

- 1 Background
 - Graphs
 - Neural Networks
- 2 Graph Neural Networks (GNNs)
 - Introduction
 - Spectral-based Convolutional GNN
 - Spatial Convolution
 - **Spatial Convolution with Recurrent Operators**
 - Spatial-based Convolutional GNN
 - Spectral-based vs Spatial-based CGNN
- 3 Applications
 - Nodes Classification
 - Graph Classification
 - Link Prediction
 - Graph Generation



Recurrent Operators

- Message passing & information aggregation translates well into recurrent operators [9].



Recurrent Operators

- Message passing & information aggregation translates well into recurrent operators [9].
- Node's hidden state $h_i^{(t)}$ is recurrently updated using a contraction function f until convergence.



Recurrent Operators

- Message passing & information aggregation translates well into recurrent operators [9].
- Node's hidden state $h_i^{(t)}$ is recurrently updated using a contraction function f until convergence.
- Aggregation function $\oplus$ is a simple summation.

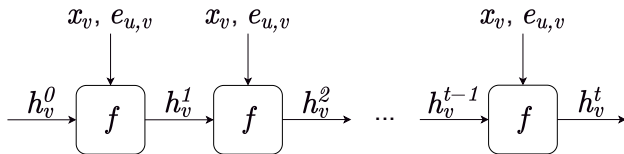
$$h_i^{(t)} = \sum_{j \in N(i)} f(x_i, e_{(j,i)}, x_j, h_i^{(t-1)})$$



Recurrent Operators

- Message passing & information aggregation translates well into recurrent operators [9].
- Node's hidden state $h_i^{(t)}$ is recurrently updated using a contraction function f until convergence.
- Aggregation function $\oplus$ is a simple summation.

$$h_i^{(t)} = \sum_{j \in N(i)} f(x_i, e_{(j,i)}, x_j, h_i^{(t-1)})$$



Considerations

Pros:



Considerations

Pros:

- ✓ Simplicity.



Considerations

Pros:

- ✓ Simplicity.
- ✓ Drop graph dependency.



Considerations

Pros:

- ✓ Simplicity.
- ✓ Drop graph dependency.
- ✓ High-level information of nodes and graphs.



Considerations

Pros:

- ✓ Simplicity.
- ✓ Drop graph dependency.
- ✓ High-level information of nodes and graphs.

Cons:



Considerations

Pros:

- ✓ Simplicity.
- ✓ Drop graph dependency.
- ✓ High-level information of nodes and graphs.

Cons:

- ✗ Do not consider variability in iterations of message passing.



Considerations

Pros:

- ✓ Simplicity.
- ✓ Drop graph dependency.
- ✓ High-level information of nodes and graphs.

Cons:

- ✗ Do not consider variability in iterations of message passing.
- ✗ Require states convergence to work properly.



Considerations

Pros:

- ✓ Simplicity.
- ✓ Drop graph dependency.
- ✓ High-level information of nodes and graphs.

Cons:

- ✗ Do not consider variability in iterations of message passing.
- ✗ Require states convergence to work properly.
- ✗ f usually manually tuned.
Complex implementation of learnable f [7].



Focus on. . .

- 1 Background
 - Graphs
 - Neural Networks
- 2 Graph Neural Networks (GNNs)
 - Introduction
 - Spectral-based Convolutional GNN
 - Spatial Convolution
 - Spatial Convolution with Recurrent Operators
 - **Spatial-based Convolutional GNN**
 - Spectral-based vs Spatial-based CGNN
- 3 Applications
 - Nodes Classification
 - Graph Classification
 - Link Prediction
 - Graph Generation



Surpass recurrent operators

- Select τ : the number of layers of the GNN.
- The final learned graph representation will be the set of $h_i^{(\tau)} \forall i \in [0, \dots, n]$.



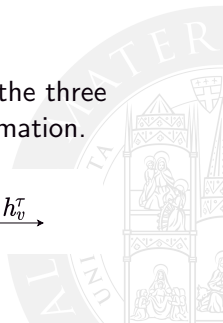
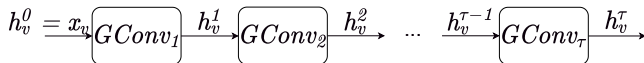
Surpass recurrent operators

- Select τ : the number of layers of the GNN.
- The final learned graph representation will be the set of $h_i^{(\tau)} \forall i \in [0, \dots, n]$.
- Each layer corresponds to one step of:
 1. Message passing
 2. Information aggregation
 3. Information transformation



Surpass recurrent operators

- Select τ : the number of layers of the GNN.
- The final learned graph representation will be the set of $h_i^{(\tau)} \forall i \in [0, \dots, n]$.
- Each layer corresponds to one step of:
 1. Message passing
 2. Information aggregation
 3. Information transformation
- **Important:** Depending on the particular layer any of the three operations can be learned. Most of the times transformation.



GCNConv [6]

- Standard message passing.



GCNConv [6]

- Standard message passing.
- Aggregate via summation.



GCNConv [6]

- Standard message passing.
- Aggregate via summation.
- Learnable transformation.



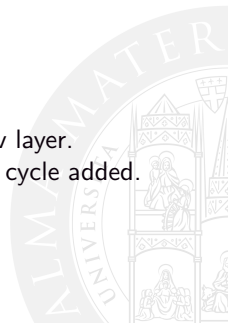
GCNConv [6]

- Standard message passing.
- Aggregate via summation.
- Learnable transformation.

$$h_i^{(t)} = \mathbf{W} \left(\sum_{j \in N(i) \cup i} \frac{e_{j,i}}{\sqrt{\hat{d}_i \hat{d}_j}} h_j^{(t-1)} \right)$$

where:

- $\mathbf{W}$ is the set of learnable parameters of the GCNConv layer.
- $\hat{d}_i = 1 + \sum_{j \in N(i)} e_{j,i}$ is the degree of node i with a self cycle added.



GINConv [10]

- Learnable weighted message passing.



GINConv [10]

- Learnable weighted message passing.
- Aggregate via summation.



GINConv [10]

- Learnable weighted message passing.
- Aggregate via summation.
- Learnable transformation.



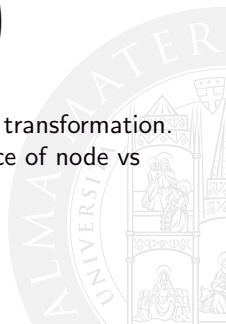
GINConv [10]

- Learnable weighted message passing.
- Aggregate via summation.
- Learnable transformation.

$$h_i^{(t)} = \mathbf{W} \left((1 + \epsilon) h_i^{(t-1)} + \sum_{j \in N(i)} h_j^{(t-1)} \right)$$

where:

- $\mathbf{W}$ is the set of learnable parameters of the GINConv transformation.
- ϵ is learnable parameter which balances the importance of node vs neighbourhood information.



PNACConv [3]

- Learnable weighted message passing.



PNAConv [3]

- Learnable weighted message passing.
- Parametrized (learnable) aggregation.



PNACConv [3]

- Learnable weighted message passing.
- Parametrized (learnable) aggregation.
- Learnable transformation.



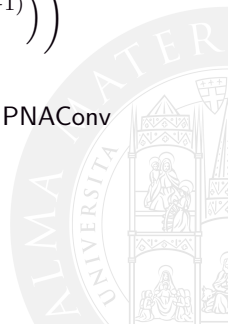
PNACConv [3]

- Learnable weighted message passing.
- Parametrized (learnable) aggregation.
- Learnable transformation.

$$h_i^{(t)} = \mathbf{W}_1 \left(h_i^{(t-1)}, \bigoplus_{j \in N(i)} \mathbf{W}_2 \left(h_j^{(t-1)}, h_i^{(t-1)} \right) \right)$$

where:

- $\mathbf{W}_1, \mathbf{W}_2$ are two sets of learnable parameters of the PNACConv transformation.
- $\bigoplus$ is a parametrized aggregation function.



Focus on. . .

- 1 Background
 - Graphs
 - Neural Networks
- 2 Graph Neural Networks (GNNs)
 - Introduction
 - Spectral-based Convolutional GNN
 - Spatial Convolution
 - Spatial Convolution with Recurrent Operators
 - Spatial-based Convolutional GNN
 - **Spectral-based vs Spatial-based CGNN**
- 3 Applications
 - Nodes Classification
 - Graph Classification
 - Link Prediction
 - Graph Generation



Spectral-based vs Spatial-based CGNN

Spectral-based:

✓ Theoretically sound.

Spatial-based:

✗ Experimentally successful.



Spectral-based vs Spatial-based CGNN

Spectral-based:

- ✓ Theoretically sound.
- ✗ Eigenvector $\rightarrow$ Inefficient.

Spatial-based:

- ✗ Experimentally successful.
- ✓ Message passing $\rightarrow$ Efficient.



Spectral-based vs Spatial-based CGNN

Spectral-based:

- ✓ Theoretically sound.
- ✗ Eigenvector $\rightarrow$ Inefficient.
- ✗ Eigenvector $\rightarrow$ Poor generalization.

Spatial-based:

- ✗ Experimentally successful.
- ✓ Message passing $\rightarrow$ Efficient.
- ✓ Local message passing $\rightarrow$ Better generalization.



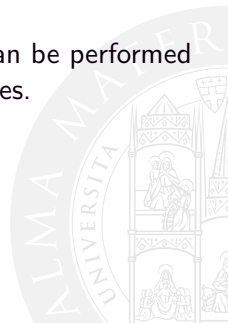
Spectral-based vs Spatial-based CGNN

Spectral-based:

- ✓ Theoretically sound.
- ✗ Eigenvector → Inefficient.
- ✗ Eigenvector → Poor generalization.
- ✗ Whole graph knowledge required.

Spatial-based:

- ✗ Experimentally successful.
- ✓ Message passing → Efficient.
- ✓ Local message passing → Better generalization.
- ✓ Computation can be performed in batch of nodes.



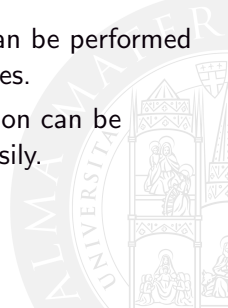
Spectral-based vs Spatial-based CGNN

Spectral-based:

- ✓ Theoretically sound.
- ✗ Eigenvector $\rightarrow$ Inefficient.
- ✗ Eigenvector $\rightarrow$ Poor generalization.
- ✗ Whole graph knowledge required.
- ✗ Undirected graphs are required.

Spatial-based:

- ✗ Experimentally successful.
- ✓ Message passing $\rightarrow$ Efficient.
- ✓ Local message passing $\rightarrow$ Better generalization.
- ✓ Computation can be performed in batch of nodes.
- ✓ Edges information can be incorporated easily.



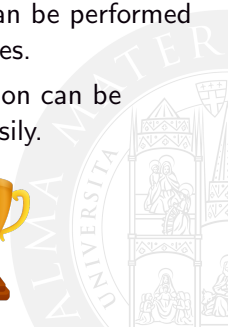
Spectral-based vs Spatial-based CGNN

Spectral-based:

- ✓ Theoretically sound.
- ✗ Eigenvector → Inefficient.
- ✗ Eigenvector → Poor generalization.
- ✗ Whole graph knowledge required.
- ✗ Undirected graphs are required.

Spatial-based:

- ✗ Experimentally successful.
- ✓ Message passing → Efficient.
- ✓ Local message passing → Better generalization.
- ✓ Computation can be performed in batch of nodes.
- ✓ Edges information can be incorporated easily.



Next in Line...

- 1 Background
- 2 Graph Neural Networks (GNNs)
- 3 Applications



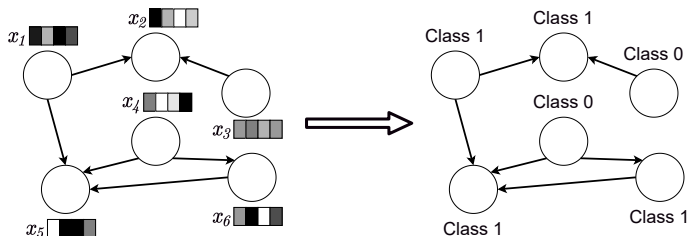
Focus on. . .

- 1 Background
 - Graphs
 - Neural Networks
- 2 Graph Neural Networks (GNNs)
 - Introduction
 - Spectral-based Convolutional GNN
 - Spatial Convolution
 - Spatial Convolution with Recurrent Operators
 - Spatial-based Convolutional GNN
 - Spectral-based vs Spatial-based CGNN
- 3 Applications
 - **Nodes Classification**
 - Graph Classification
 - Link Prediction
 - Graph Generation



Nodes classification

Idea: Each node in the graph must be assigned to a certain class, depending on the properties of the node.



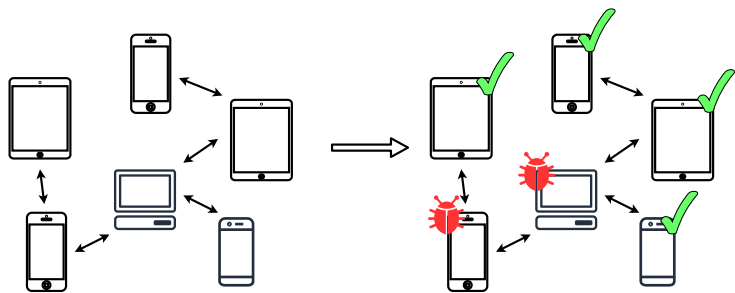
Examples

Fake account detection in social networks

- **Input:** Accounts in a social network.
- **Aim:** Identify fake accounts.

Intrusion Detection Systems

- **Input:** Network of interconnected devices.
- **Aim:** Identify malicious/infected devices.



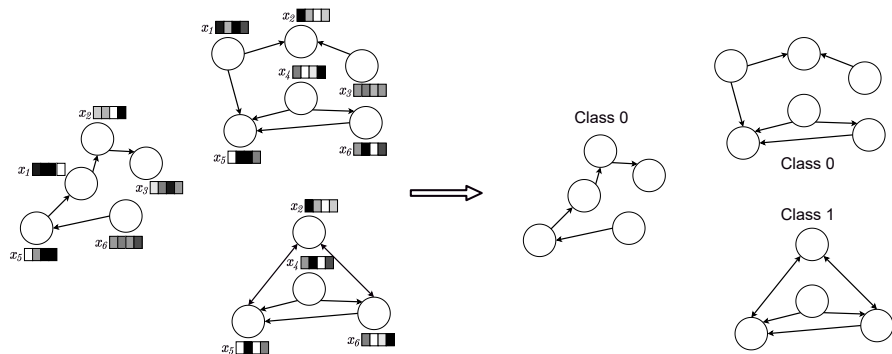
Focus on. . .

- 1 Background
 - Graphs
 - Neural Networks
- 2 Graph Neural Networks (GNNs)
 - Introduction
 - Spectral-based Convolutional GNN
 - Spatial Convolution
 - Spatial Convolution with Recurrent Operators
 - Spatial-based Convolutional GNN
 - Spectral-based vs Spatial-based CGNN
- 3 Applications
 - Nodes Classification
 - **Graph Classification**
 - Link Prediction
 - Graph Generation



Graph classification

Idea: Given a set of graphs, classify each graph into a class.



Examples

Molecule identification

- **Input:** Molecules.
- **Aim:** Classify molecules depending on natural properties.

Sentiment analysis

- **Input:** Graph representing text.
- **Aim:** Identify sentiment of author.

Image classification

- **Input:** Image.
- **Aim:** Assign label to image.



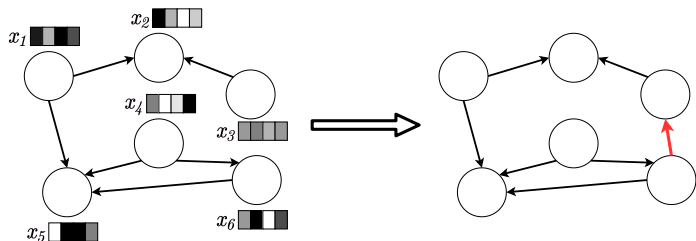
Focus on. . .

- 1 Background
 - Graphs
 - Neural Networks
- 2 Graph Neural Networks (GNNs)
 - Introduction
 - Spectral-based Convolutional GNN
 - Spatial Convolution
 - Spatial Convolution with Recurrent Operators
 - Spatial-based Convolutional GNN
 - Spectral-based vs Spatial-based CGNN
- 3 Applications
 - Nodes Classification
 - Graph Classification
 - **Link Prediction**
 - Graph Generation



Link prediction

Idea: Forecast pairs of nodes that will form a link in the future.



Examples

Friendship creation

- **Input:** Social network graph.
- **Aim:** Predict new friendships.

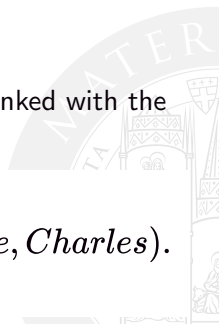
NN structure optimization

- **Input:** Graph representing structure of a Neural Network.
- **Aim:** Predict links that improve performance.

Predicate completion & Induction in LP

- **Input:** Logic programming predicates.
- **Aim:** Automatically produce new predicates logically linked with the ones given.

$\left. \begin{array}{l} \text{parent}(\text{Charles}, \text{William}). \\ \text{parent}(\text{Kate}, \text{William}). \end{array} \right\} \Rightarrow \text{wife}(\text{Kate}, \text{Charles}).$



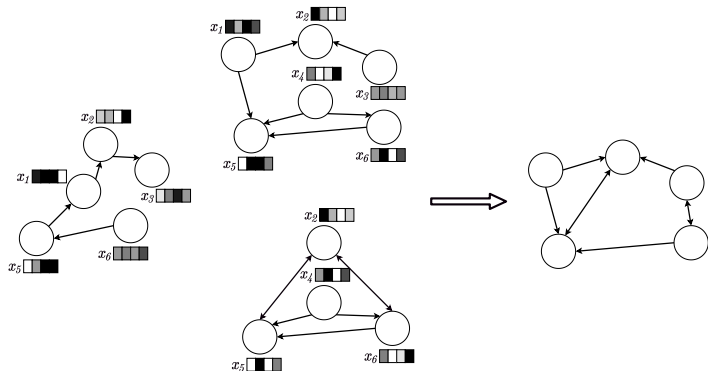
Focus on. . .

- 1 Background
 - Graphs
 - Neural Networks
- 2 Graph Neural Networks (GNNs)
 - Introduction
 - Spectral-based Convolutional GNN
 - Spatial Convolution
 - Spatial Convolution with Recurrent Operators
 - Spatial-based Convolutional GNN
 - Spectral-based vs Spatial-based CGNN
- 3 Applications
 - Nodes Classification
 - Graph Classification
 - Link Prediction
 - **Graph Generation**



Graph generation

Idea: Given a set of real graphs, generate unseen graphs with characteristics similar to given real graphs.



Examples

NN generation

- **Input:** Graphs representing structures of a Neural Networks.
- **Aim:** Structure of Neural Network with improved performance.

Molecule generation

- **Input:** Graphs representing structures of molecules used in pharmacology.
- **Aim:** Unseen molecules to be used in drugs.

Explainable image captioning

- **Input:** Image.
- **Aim:** Graph of relations between objects in the image.



Thanks for your attention!



References I

- [1] Mikhail Belkin and Partha Niyogi.

Laplacian eigenmaps and spectral techniques for embedding and clustering.

In Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani, editors, *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]*, pages 585–591. MIT Press, 2001.

- [2] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun.

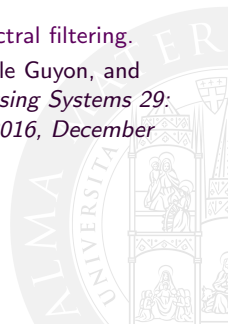
Spectral networks and locally connected networks on graphs.

In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.



References II

- [3] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Velickovic. Principal neighbourhood aggregation for graph nets. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.
- [4] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3837–3845, 2016.



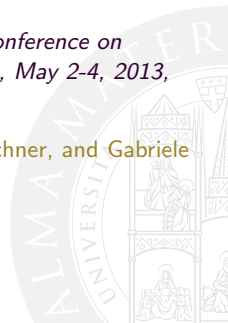
References III

- [5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova.
BERT: pre-training of deep bidirectional transformers for language understanding.
In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019.
- [6] Thomas N. Kipf and Max Welling.
Semi-supervised classification with graph convolutional networks.
In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.



References IV

- [7] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel.
Gated graph sequence neural networks.
In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.
- [8] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean.
Efficient estimation of word representations in vector space.
In Yoshua Bengio and Yann LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013.
- [9] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini.
The graph neural network model.
IEEE Trans. Neural Networks, 20(1):61–80, 2009.



References V

[10] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka.

How powerful are graph neural networks?

In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

