DR. B. KHALDI
Assoc. Prof.
email: b.khaldi@esi-sba.dz

Part II

Basic GNNs & Applications

Lecture 06

Introduction to Graph Attention Networks (GATs)

- ❑ Learn GATs fundamentals and its significance in graph-based data analysis.
- ❑ Understand message passing using Self-attention in GATs.
- ❑ Apply GATs to various graph-based tasks for valuable insights.

DR. B. KHALDI
Assoc. Prof.
email: b.khaldi@esi-sba.dz

Challenges and Limitations of GCNs

V-GNN

Global Transition Function (Message Passing):

- The V-GNN employs a layer-wise propagation rule:

$$H^{(k+1)} = \Psi(\tilde{A} H^{(k)} W^{(k)})$$

Vanilla GNNs treat all neighbors equally.

No consideration for the difference in neighbor counts.

GCN

Global Transition Function (Message Passing):

- The GCN employs a specific layer-wise propagation rule:

$$H^{(k+1)} = \Psi(L_{norm} H^{(k)} W^{(k)})$$

$$L_{norm} = \tilde{D}^{-1/2} \tilde{A}\, \tilde{D}^{-1/2}$$

GCNs aim to tackle the issues of vanilla GNNs through **normalization**

Consideration for the difference in neighbor counts are made thanks to the normalization coefficient:

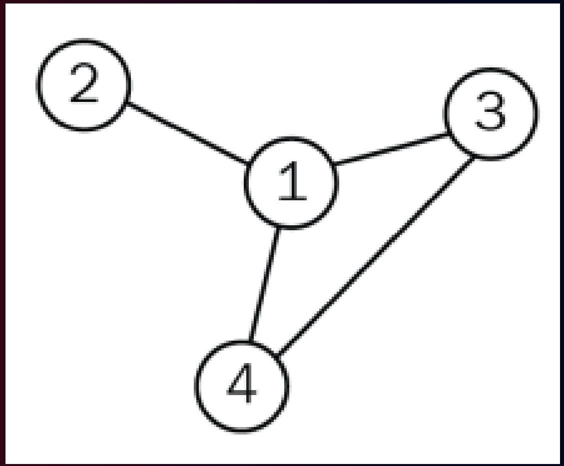
$$\frac{1}{\sqrt{Deg_{(i)}}\sqrt{Deg_{(j)}}}$$

DR. B. KHALDI
Assoc. Prof.
email: b.khaldi@esi-sba.dz

Challenges and Limitations of GCNs

Static Normalization Coefficients:	Relies on **fixed coefficients**, limiting adaptability.
Homophily:	Favors nodes with **similar degrees** due to **normalization**.
Fixed Aggregation:	Uses a **uniform aggregation** strategy for all nodes.
Limited Expressiveness:	Struggles with **diverse relationships** and **features**.
Lack of Feature Importance:	Does not account for **node feature significance**.

```
L_norm:
[[0.25 0.35 0.29 0.29]
 [0.35 0.5  0.   0.  ]
 [0.29 0.   0.33 0.33]
 [0.29 0.   0.33 0.33]]
```



In a Traffic Example, the **coefficients**:

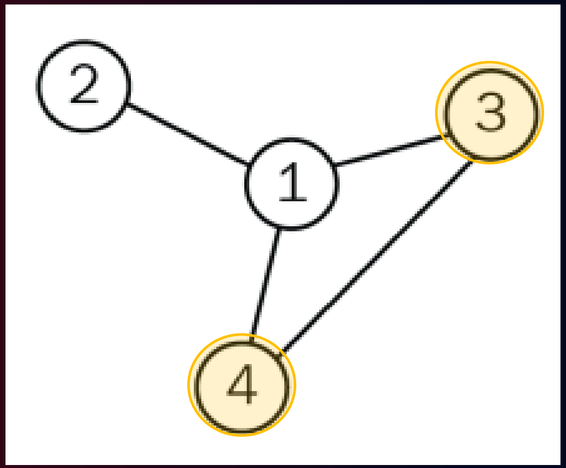
- Are **fixed** and do not change with traffic conditions.
- Do not account for the **variability** of **traffic flow** between different **cities** and **times**.
- May not be accurate or realistic for **modeling traffic flow**.

DR. B. KHALDI
Assoc. Prof.
email: b.khaldi@esi-sba.dz

Challenges and Limitations of GCNs

Static Normalization Coefficients:	Relies on **fixed coefficients**, limiting adaptability.
Homophily:	Favors nodes with **similar degrees** due to **normalization**.
Fixed Aggregation:	Uses a **uniform aggregation** strategy for all nodes.
Limited Expressiveness:	Struggles with **diverse relationships** and **features**.
Lack of Feature Importance:	Does not account for **node feature significance**.

```
L_norm:
[[0.25 0.35 0.29 0.29]
 [0.35 0.5  0.   0.  ]
 [0.29 0.   0.33 0.33]
 [0.29 0.   0.33 0.33]]
```



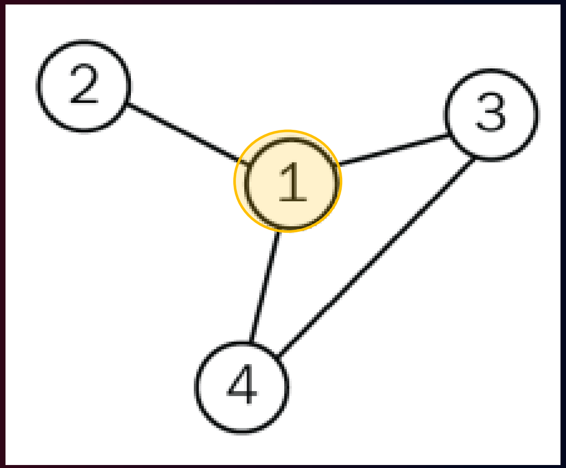
In a Traffic Example:
- Cities with **similar numbers of connections**, (like 4 and 3), are treated similarly by GCNs.
- In real word they have **different traffic patterns**.

DR. B. KHALDI
Assoc. Prof.
email: b.khaldi@esi-sba.dz


Challenges and Limitations of GCNs

Static Normalization Coefficients:	Relies on **fixed coefficients**, limiting adaptability.
Homophily:	Favors nodes with **similar degrees** due to **normalization**.
Fixed Aggregation:	Uses a **uniform aggregation** strategy for all nodes (Message Passing).
Limited Expressiveness:	Struggles with **diverse relationships** and **features**.
Lack of Feature Importance:	Does not account for **node feature significance**.

```
L_norm:
[[0.25 0.35 0.29 0.29]
 [0.35 0.5  0.   0.  ]
 [0.29 0.   0.33 0.33]
 [0.29 0.   0.33 0.33]]
```



In a Traffic Example, the aggregation strategy:

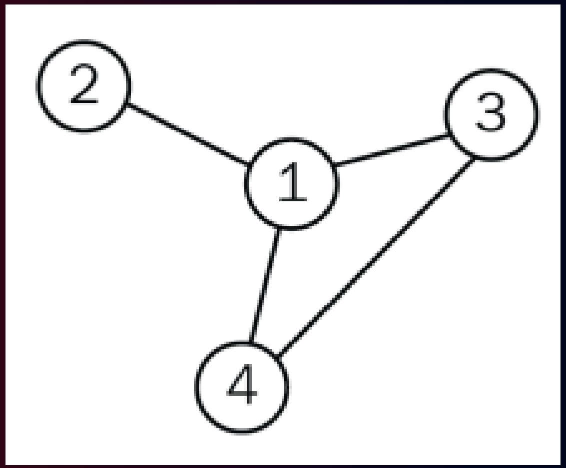
- **Equal Neighbors:** Cities may have diverse traffic pattern. But, yet GCNs assumes **equal neighbor importance** when performing the aggregation.
- **Ignores Dynamics:** Overlooks traffic variations.

DR. B. KHALDI
Assoc. Prof.
email: b.khaldi@esi-sba.dz

Challenges and Limitations of GCNs

Static Normalization Coefficients:	Relies on **fixed coefficients**, limiting adaptability.
Homophily:	Favors nodes with **similar degrees** due to **normalization**.
Fixed Aggregation:	Uses a **uniform aggregation** strategy for all nodes.
Limited Expressiveness:	Struggles with **diverse relationships** and **features**.
Lack of Feature Importance:	Does not account for **node feature significance**.

```
L_norm:
[[0.25 0.35 0.29 0.29]
 [0.35 0.5  0.   0.  ]
 [0.29 0.   0.33 0.33]
 [0.29 0.   0.33 0.33]]
```



In a Traffic Example:

- **Complex Traffic Factors:** Weather, road quality, and urban development affect traffic.
- **GCNs** may struggle to capture these complexities, reducing prediction accuracy.

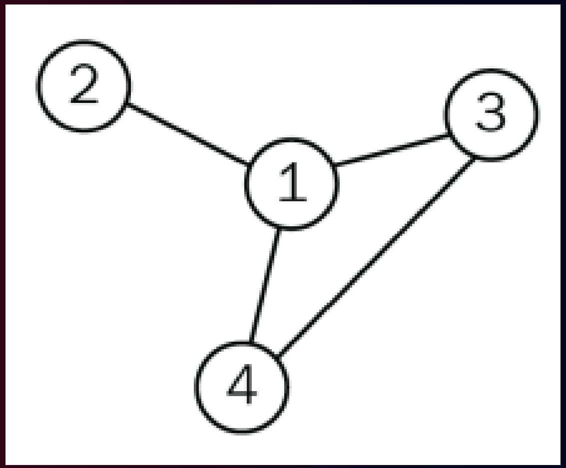
DR. B. KHALDI
Assoc. Prof.
email: b.khaldi@esi-sba.dz

Limitations of Vanilla GCN
GAT
Models Comparison

Challenges and Limitations of GCNs

Static Normalization Coefficients:	Relies on **fixed coefficients**, limiting adaptability.
Homophily:	Favors nodes with **similar degrees** due to **normalization**.
Fixed Aggregation:	Uses a **uniform aggregation** strategy for all nodes.
Limited Expressiveness:	Struggles with **diverse relationships** and **features**.
Lack of Feature Importance:	Does not account for **node feature significance**.

```
L_norm:
[[0.25 0.35 0.29 0.29]
 [0.35 0.5  0.   0.  ]
 [0.29 0.   0.33 0.33]
 [0.29 0.   0.33 0.33]]
```



In a Traffic Example:

- **Uniform Treatment of Features:** GCNs treat all city features equally, ignoring variations in the importance of data like population, geographical location, and infrastructure.
- This can lead to overlooking critical features for traffic prediction.

DR. B. KHALDI
Assoc. Prof.
email: b.khaldi@esi-sba.dz

Graph Attention Networks (GATs)

Dynamic Normalization:	Introduces **self-attention** for adaptive **coefficient** calculation.
Beyond Homophily:	Considers both **connections** and **features** for nuanced importance.
Multi-Head Attention:	Enables nodes to have **different importance levels** for **different neighbors**.
Improved Expressiveness:	Handles complex graphs with **varying relationships** and **features**.
Feature Significance:	Introduces **attention scores** for **adaptive feature** focus during **aggregation**.

What are GATs?

GATs are popular graph neural networks that are a **t**heoretical Improvement of GCNs.

Features of a GAN Layer

- **Dynamic Weights via Self-Attention:** GATs use dynamic weighting through **self-attention** instead of fixed **normalization coefficients**.
- **Shared Core with Transformers:** GATs share a core concept with the highly successful **transformer** architecture, linked to **BERT** and **GPT-3**.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2017). Graph attention networks. *arXiv preprint arXiv:1710.10903*.

DR. B. KHALDI
Assoc. Prof.
email: b.khaldi@esi-sba.dz

Introducing the Graph Attention Layer (GAL)

Lets Recall about GCN Layer

GCN

Global Transition Function (Message Passing):

- The GCN employs a specific layer-wise propagation rule:

$$H^{(k+1)} = \Psi(L_{norm} H^{(k)} W^{(k)})$$

$$L_{norm} = \tilde{D}^{-1/2} \tilde{A} \, \tilde{D}^{-1/2}$$

GCN

$$h_i^{(k+1)} = \Psi\left(\sum_{j \in \aleph_i} \alpha_{ij}^{(k)} W^{(k)} h_i^{(k)}\right)$$

$$\alpha_{ij}^{(k)} = \frac{1}{\sqrt{Deg_{(i)}^{(k)}}\sqrt{Deg_{(j)}^{(k)}}}$$

Weighting factor (importance) of node j's features to node i.

Attention Scores are Explicitly Fixed in GCNs

DR. B. KHALDI
Assoc. Prof.
email: b.khaldi@esi-sba.dz

Introducing the Graph Attention Layer (GAL)

Attention Mechanism In GAL – One Attention Head

GAL introduces the **attention mechanism** as a substitute for the **statically normalized convolution operation**.

$$\underbrace{\alpha_{ij}^{(k)}}_{\text{Self-Attention Scores}} = \underbrace{SoftMax}_{\text{SoftMax normalization}}\left(\underbrace{e_{ij}^{(k)}}_{\text{Unnormalised Coefficients}}\right) = \frac{Exp\left(e_{ij}^{(k)}\right)}{\sum_{k\in\aleph_i} Exp\left(e_{ik}^{(k)}\right)}$$

$$e_{ij}^{(k)} = \text{LeakyReLU}\left(\underbrace{\vec{a}^{(k)^T}}_{\text{A learnable weight vector}}\left(z_i^{(k)}\,||\,z_j^{(k)}\right)\right)$$

$$z_i^{(k)} = W^{(k)} h_i^{(k)}$$

$$z_j^{(k)} = W^{(k)} h_i^{(k)}$$

$$h_i^{(k+1)} = \Psi\left(\sum_{j\in\aleph_i} \alpha_{ij}^{(k)} z_j^{(k)}\right)$$

The result of this operation gives the final **attention scores** for one head attention.

DR. B. KHALDI
Assoc. Prof.
email: b.khaldi@esi-sba.dz

Introducing the Graph Attention Layer (GAL)

Attention Mechanism In GAL – Multi-head attention

- Analogous to multiple channels in **ConvNet**, **GAT** introduces **multi-head attention** to enrich the model capacity and to stabilize the learning process.
- Each attention head has its own parameters and their outputs can be merged in two ways:

Concatenation over N Heads

$$h_i^{(k+1)} = ||_{n=1}^{N} \Psi\left(\sum_{j \in \aleph_i} \left(\alpha_{ij}^{(k)}\right)_n \left(W^{(k)}\right)_n \left(h_i^{(k)}\right)_n\right)$$

Average over N Heads

$$h_i^{(k+1)} = \Psi\left(\frac{1}{N}\sum_{n=1}^{N}\sum_{j \in \aleph_i} \left(\alpha_{ij}^{(k)}\right)_n \left(W^{(k)}\right)_n \left(h_i^{(k)}\right)_n\right)$$

DR. B. KHALDI
Assoc. Prof.
email: b.khaldi@esi-sba.dz

Graph Attention Network Implementation

01 Class Overview

02 __init__()

03 forward()

Implementation the GAT using a built-in GAL:

```
from torch_geometric.nn import GATv2Conv
# Create a new class named GAT
class GAT(nn.Module):
    def __init__(self, dim_in, dim_h, dim_out, head=hd):
        # Initialize the GAT class with input, hidden,
          output layer dimensions, and number of heads
    def forward(self, x, edge_index):
        # Perform the forward pass of the GAT
    def accuracy(self, y_pred, y_true):
        # Calculate the accuracy of predictions
    def fit(self, data, epochs):
        # Train the model
    def test(self, data):
        # Evaluate the model
```

DR. B. KHALDI
Assoc. Prof.
email: b.khaldi@esi-sba.dz

Graph Attention Network Implementation

Implementation the GAT using a built-in GAL:

01 Class Overview

01 __init__()

02 forward()

```
from torch_geometric.nn import GATv2Conv
# class named GAT
class GAT(nn.Module):
  def __init__(self, dim_in, dim_h, dim_out, heads=heads):
      super().__init__()
      self.gat1 = GATv2Conv(dim_in, dim_h, heads)
      self.gat2 = GATv2Conv(dim_h*heads, dim_out, heads)
```

DR. B. KHALDI
Assoc. Prof.
email: b.khaldi@esi-sba.dz

Graph Attention Network Implementation

01 Class Overview

02 __init__()

03 **forward()**

Implementation the GAT using a built-in GAL:

```
# class named GAT
class GAT(nn.Module):
    def forward(self, x, edge_index):
        # Pass the input features to the 1st GAT layer (gat1)
        h = self.gat1(h, edge_index)
        # Apply Exponential Linear Unit (ELU) activation
          function
        h = F.elu(h)
        # Pass the output to the 2nd GAT layer (gat2)
        h = self.gat2(h, edge_index)
        # Apply log softmax to the final output and return the
        result
        return F.log_softmax(h, dim=1)
```

DR. B. KHALDI
Assoc. Prof.
email: b.khaldi@esi-sba.dz

Graph Attention Network Implementation

01 Class Overview

02 __init__()

03 forward()

04 Building, Training, and Testing the GCN

Implementation the GAT using a built-in GAL:

```
# Create a GAT instance with specified input, hidden, and output dimensions, and heads
gat = GAT(dataset.num_features, 32, dataset.num_classes)

# Print the model architecture
print(gat)
```

```
GAT(
  (gat1): GATv2Conv(1433, 32, heads=8)
  (gat2): GATv2Conv(256, 7, heads=1)
)
```

DR. B. KHALDI
Assoc. Prof.
email: b.khaldi@esi-sba.dz

Graph Attention Network Implementation

Implementation the GAT using a built-in GAL:

```
# Train the GCN model on the given data for a specified number of epochs (100 in this case) and the adjacency matrix.
gat.fit(data, epochs=100)
```

```
Epoch   0 | Train Loss: 1.969 | Train Acc: 15.00% | Val Loss: 1.96 | Val Acc: 11.80%
Epoch  20 | Train Loss: 0.259 | Train Acc: 96.43% | Val Loss: 1.10 | Val Acc: 67.60%
Epoch  40 | Train Loss: 0.163 | Train Acc: 98.57% | Val Loss: 0.90 | Val Acc: 70.80%
Epoch  60 | Train Loss: 0.205 | Train Acc: 98.57% | Val Loss: 0.96 | Val Acc: 69.00%
Epoch  80 | Train Loss: 0.130 | Train Acc: 100.00% | Val Loss: 0.91 | Val Acc: 70.80%
Epoch 100 | Train Loss: 0.148 | Train Acc: 99.29% | Val Loss: 0.90 | Val Acc: 73.00%
```

```
# Test the model and get accuracy
test_acc = gat.test(data)
print(f'\nGAT test accuracy: {test_acc*100:.2f}%')
```

```
GAT test accuracy: 82.00%
```

DR. B. KHALDI
Assoc. Prof.
email: b.khaldi@esi-sba.dz

Models Comparison

Dataset	MLP	Vanilla GNN	GCN	GAT
Cora	51.90%	72.50% +20.60%	79.70% +27.80%	82% +30.10%

DR. B. KHALDI
Assoc. Prof.
email: b.khaldi@esi-sba.dz

Models Comparison

GANs

Feature Importance:
- Uniform treatment of features.

Multi-Head Attention:
- No built-in support

Aggregation Strategy:
- Uniform aggregation for all nodes.

Normalization Coefficients:
- Static coefficients, limiting adaptability.

GATs

Feature Importance:
- Explicit feature importance with learned coefficients.

Multi-Head Attention:
- Multiple attention heads for diverse information.

Aggregation Strategy:
- Fine-grained aggregation with multiple attention heads.

Normalization Coefficients:
- Learnable attention coefficients, flexible and adaptive.

DR. B. KHALDI
Assoc. Prof.
email: b.khaldi@esi-sba.dz

THANK YOU