Graph Neural Network

Table of **contents**

01 Introduction

02 Application of Gragh neurol network

03 Gragh neurol network

04 Gragh convolutional network

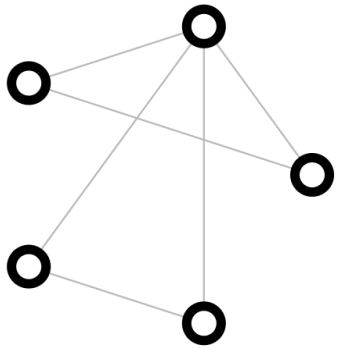
05 Graphsage

06 Bài tập

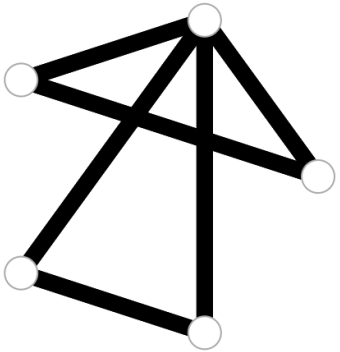
01

Introduction

Graph theory



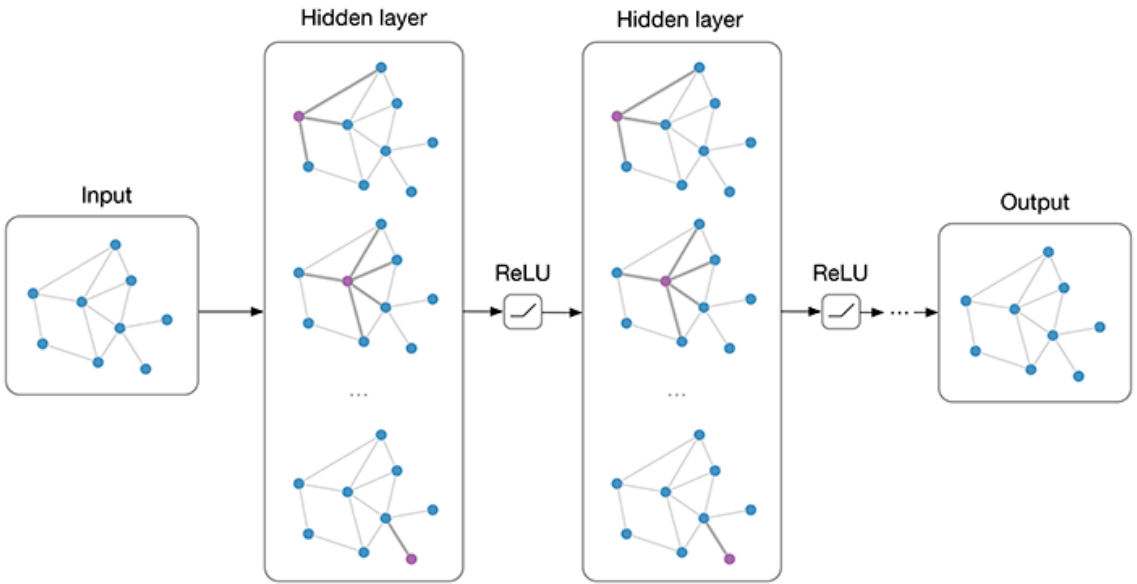
V là tập hợp các **nút** của đồ thì



E là tập hợp các **cạnh** kết nối các nút của đồ thị

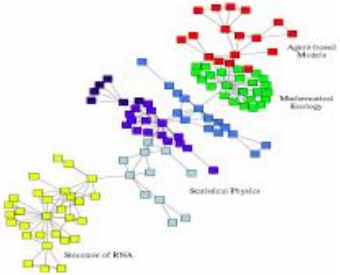
- Đồ thị được biểu diễn dưới dạng G = (V, E)
- G là đồ thị được cấu thành
- Kí hiệu N(v) = u ∈ V|(v,u) ∈ E là các nút kề u với v

Graph theory

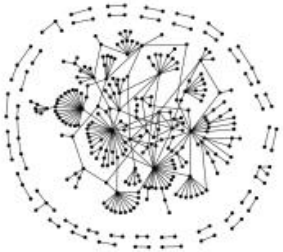


Graph theory

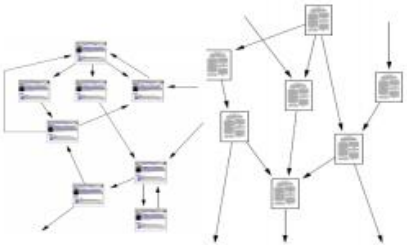
Social networks



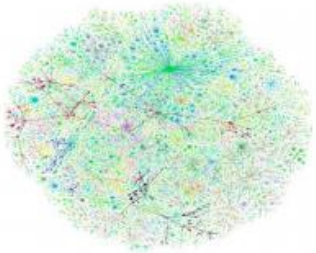
Economic networks



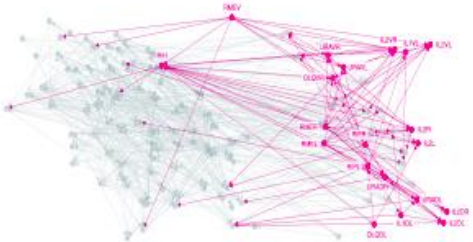
Communication networks



Information networks:
Web & citations



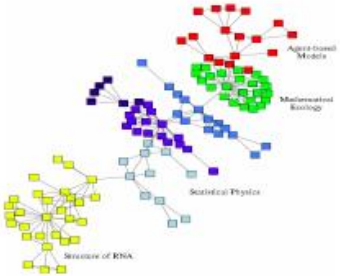
Internet



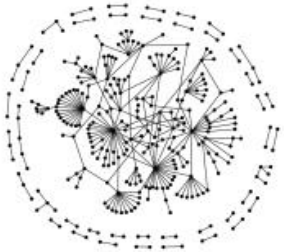
Networks of neurons

Graph theory

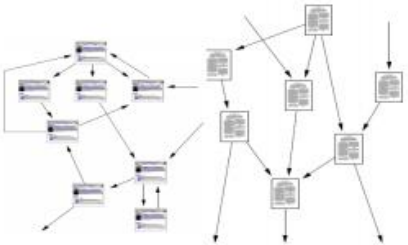
Social networks



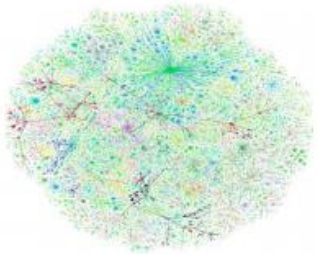
Economic networks



Communication networks



Information networks:
Web & citations



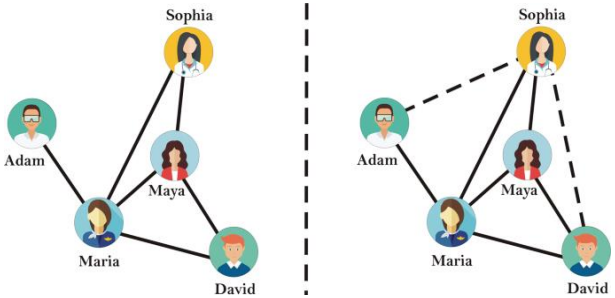
Internet

Networks of neurons

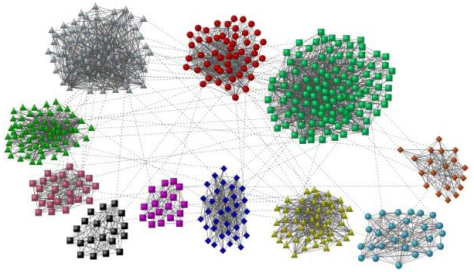
02

Application of GNNs

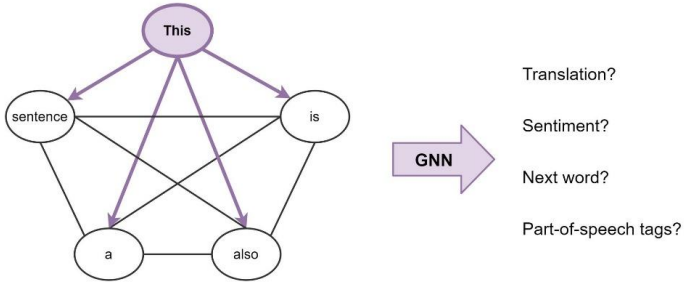
Application of **GNNs**



Link Prediction



Clustering & Community detection

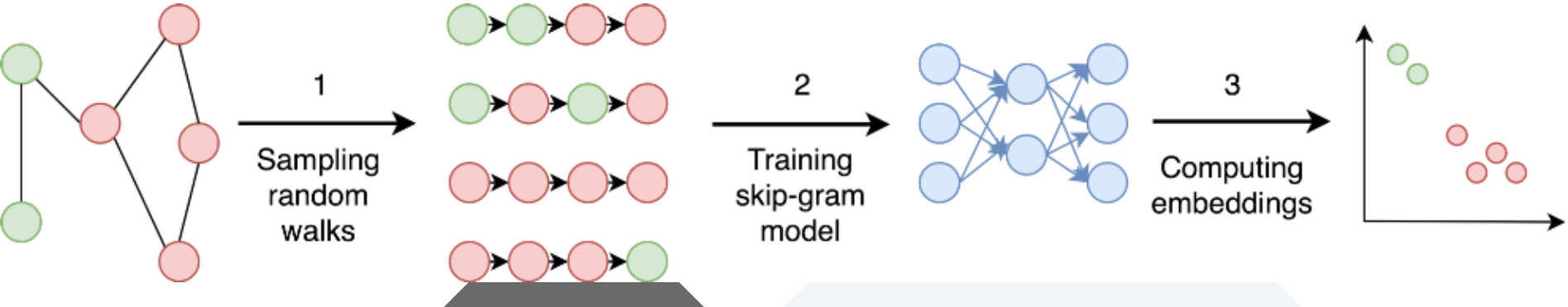


Natural Language Processing

Node embeding

DeepWalk là 1 mô hình **Node Embedding**, dựa trên ý tưởng chủ đạo từ **Word2Vec**, mà cụ thể là từ mô hình Skip-gram.

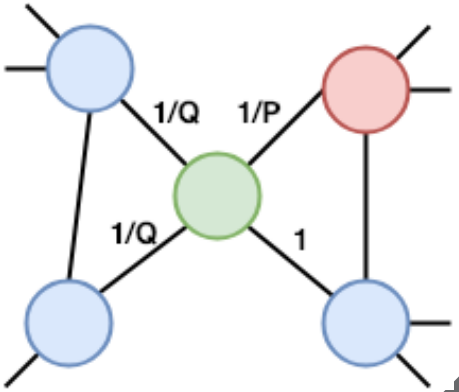
Bằng việc sampling sử dụng **random walk**, ta đã chuyển dữ liệu từ 1 dạng cấu trúc phức tạp là đồ thị sang dạng biểu diễn sequence 1D

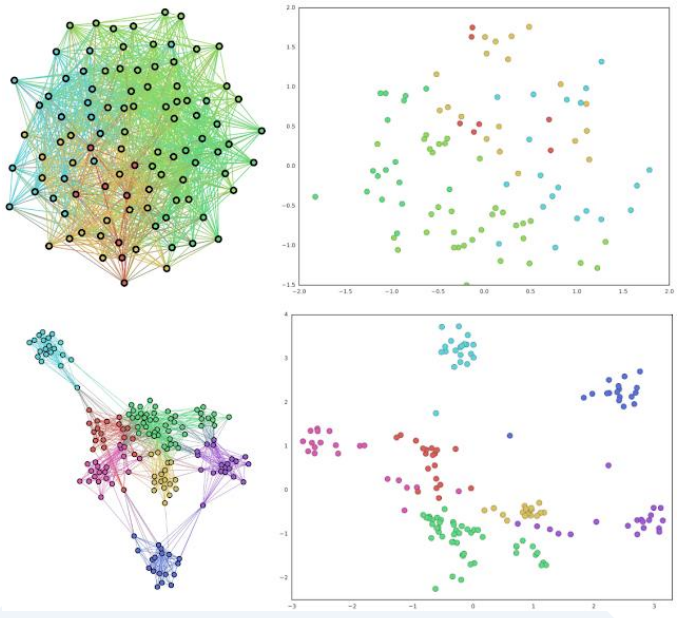


Node2Vec

Node2Vec cũng là 1 mô hình Node Embedding dựa trên ý tưởng của **DeepWalk và Word2Vec**

Điểm khác biệt của Node2Vec là ngoài việc sử dụng **random walk** như thông thường, mô hình giới thiệu thêm 2 thông số **P** và **Q** để điều chỉnh lại bước đi ngẫu nhiên trên đồ thị.



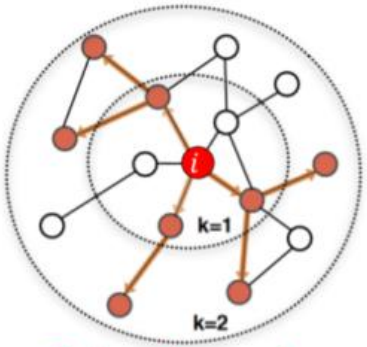


03

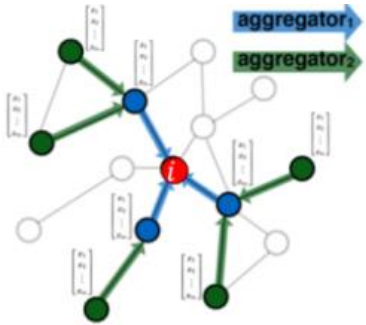
Gragh Neurol Network

Neighborhood Aggregation

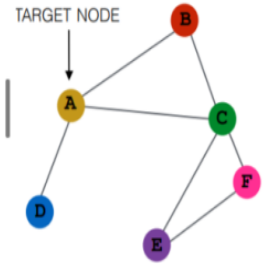
Key idea: Generate node embeddings based on local neighborhood



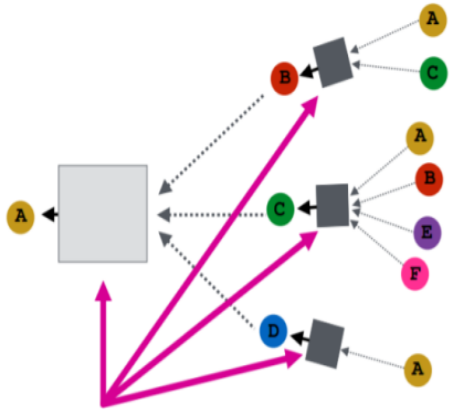
Determine node computation graph



Propagate and transform information



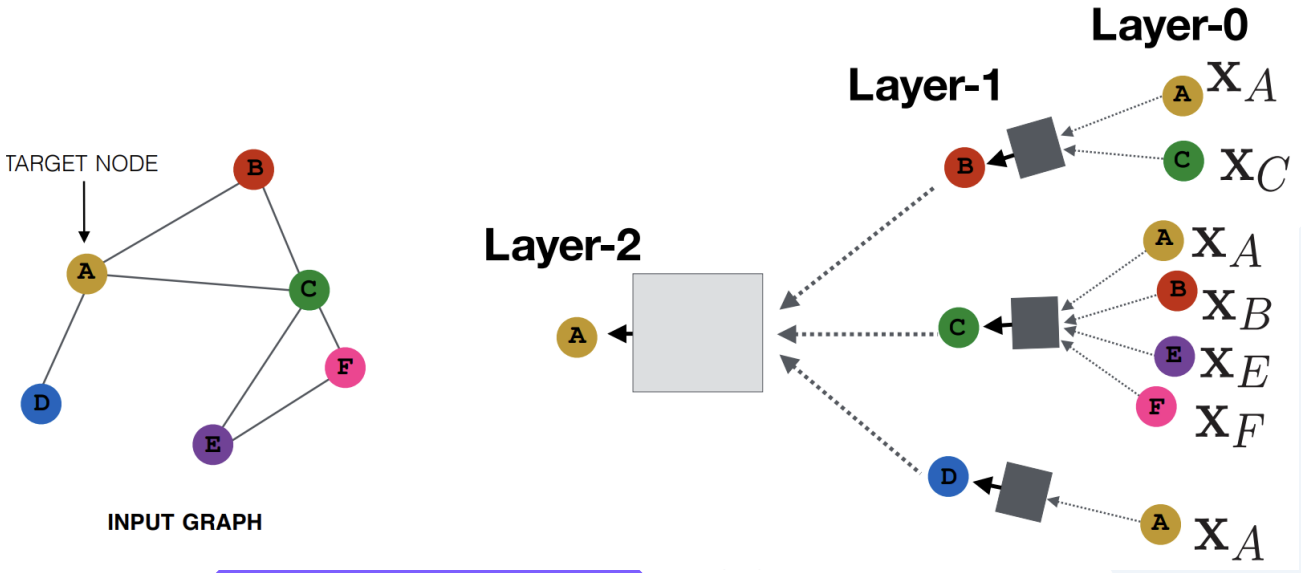
INPUT GRAPH



Neural networks

Neighborhood Aggregation

- Node have embeddings each layer
- Layer 0 embedding of node u is its input feature



Simple neighborhood aggregation

- **Basic approach**: Average neighbor messages and apply a neurol network

Initial layer 0 embeddings are equal to node feature

$$h_v^0 = X_v$$

$$h_v^k = \sigma\left(W_k \sum_{u \in N(v)} \frac{h_u^{k-1}}{|N(v)|} + B_k h_u^{k-1}\right), \forall k > 0$$

h_v^k: K^{th} layer embeddings of v

$\sum_{u \in N(v)} \frac{h_u^{k-1}}{|N(v)|}$: Average of neighbor previous layer embeddings

h_u^{k-1}: previous layer embeddings of v

04

Graph Convolutional Network

Graph Convolutional Network

- GCN are a slight variation on the neighborhood aggregation idea

$$h_v^k = \sigma\left(W_k \sum_{u \in N(v) \cup v} \frac{h_u^{k-1}}{\sqrt{N(u)||N(v)}}\right.$$

Use the same transformation matrix for self and neighbor embeddings

instead of simple average, normalization varies across neighbors

Batch Implementation

- 1 hidden layer có thể được biểu diễn như sau: $H^{(k+1)} = f(H^k, A)$

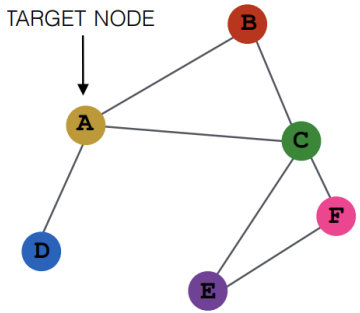
- Trong đó hàm f có thể biểu diễn đơn giản bằng công thức sau: $H^{(k+1)} = \sigma(D^{-\frac{1}{2}}\mathrm{A}\mathrm{I}D^{-\frac{1}{2}}H \quad W_k)$

05

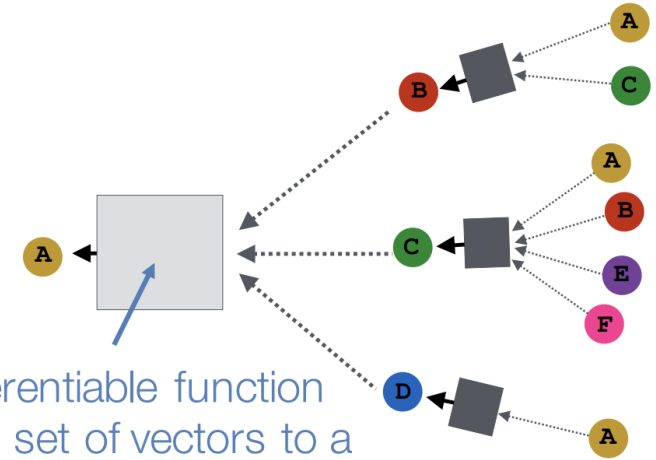
GraghSage

GraphSAGE

So far we have aggregated the neighbor messages by taking their (weighted) average, can we do better?



INPUT GRAPH



Any differentiable function that maps set of vectors to a single vector.

$$H_v^k = \sigma(A_k \cdot AGG(\{h_u^{k-1}, \forall u \in N(v)\}), B_k h_u^{k-1})$$

GraphSAGE Variants

Mean

$$\text{AGG} = \sum_{u \in N(v)} \frac{h_u^{k-1}}{|N(v)|}$$

Pool

Transfrom neighbor vectors and apply symmetric vectors function

$$\text{AGG} = \gamma(\{Qh_u^{k-1}, \forall u \in N(n)\})$$

LSTM

Apply LSTM to random permutation of neighbor

$$\text{AGG} = LTMS([h_u^{k-1}, \forall u \in \pi(N(n))])$$

GraphSAGE

- Input: đồ thị $G = (V, E)$, các note feature $x_v \in X, \forall v \in V, \forall k \in 1, ..., K$, với K là số aggregator function được áp dụng liên tiếp nhau
- Output: embedding vector $z_v \forall v \in V$

$\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$;
for $k = 1...K$ **do**
  for $v \in \mathcal{V}$ **do**
    $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$;
    $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$
  end
  $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$
end
$\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$

Disadvantages of Graph Neural Networks

Memory requirement

The model weights are still updated epoch-by-epoch, but for each epoch are updated according to **full-batch** gradient descent

Transductive setting

limited to **undirected graph**

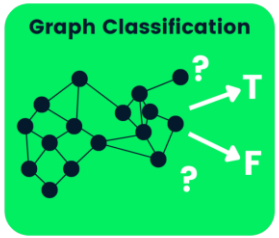
Directed edges and edge features

Very poor with those new buttons and requires **re-training** to update the model

Limiting assumption

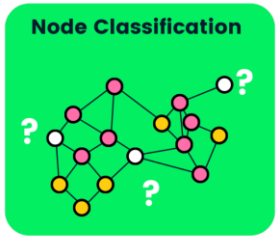
Types of Graph Neural Networks Tasks

1 Graph Classification

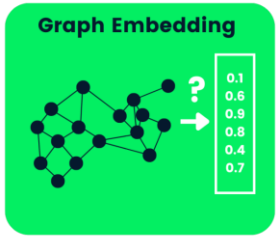


2 Node Classification

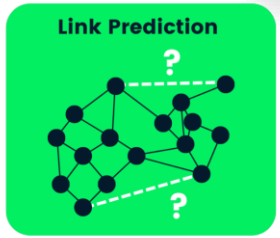
3 Link Prediction



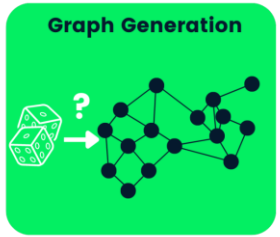
4 Community Detection



5 Graph Embedding



6 Graph Generation



06

Bài Tập

1

Hãy giải thích cách GNN áp dụng đặc trưng của các đỉnh và cạnh để tính toán đặc trưng của một đỉnh trong đồ thị.

Cách GNN áp dụng đặc trưng của các đỉnh và cạnh để tính toán đặc trưng của một đỉnh trong đồ thị là sử dụng một mạng nơ-ron để tính toán đặc trưng của mỗi đỉnh dựa trên đặc trưng của đỉnh đó và các đặc trưng của các đỉnh và cạnh kề. Quá trình tính toán đặc trưng này được thực hiện thông qua các lớp của GNN

2 Hãy nêu ví dụ về các loại lớp GNN khác nhau và giải thích sự khác nhau giữa chúng.

Graph Convolutional Network (GCN): Sử dụng tích chập đồ thị để lấy trung bình trọng số của các đặc trưng của các đỉnh và cạnh kề để tính toán đặc trưng của mỗi đỉnh.
GraphSAGE: Sử dụng phép tổng hợp cục bộ (local aggregation) để lấy thông tin đặc trưng từ các đỉnh và cạnh kề để tính toán đặc trưng của mỗi đỉnh.
Gated Graph Neural Network (GGNN): Sử dụng cổng đầu vào để xác định độ quan trọng của các đỉnh và cạnh kề để tính toán đặc trưng của mỗi đỉnh.

3

Hãy giải thích cách GNN giải quyết vấn đề về sự thay đổi kích thước đồ thị (graph size) trong quá trình huấn luyện.

GNN giải quyết vấn đề về sự thay đổi kích thước đồ thị trong quá trình huấn luyện bằng cách sử dụng các lớp chuyển đổi (pooling, unpooling) để điều chỉnh kích thước đồ thị.