Consolidation

Prof. O-Joun Lee

Dept. of Artificial Intelligence,
The Catholic University of Korea
ojlee@catholic.ac.kr

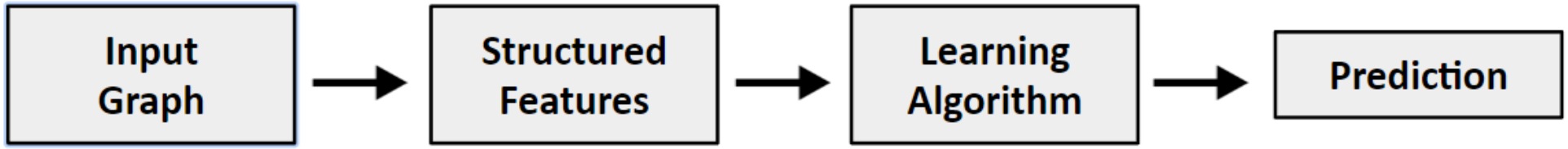
네트워크 과학 연구실
NETWORK SCIENCE LAB

가톨릭대학교
THE CATHOLIC UNIVERSITY OF KOREA

Contents

- Traditional Machine Learning for Graphs
- Shallow Embedding Models
- Graph Neural Networks (GNNs)
  - Scalability of GNNs
  - The variants of GNNs
  - Structure preservation of GNNs
- Graph Transformers
- Knowledge Graphs

- Given a graph, we can extract features (node-level, graph-level) from the graph, then directly put them to a shallow model to map the features to the ground truth.



Feature Engineering

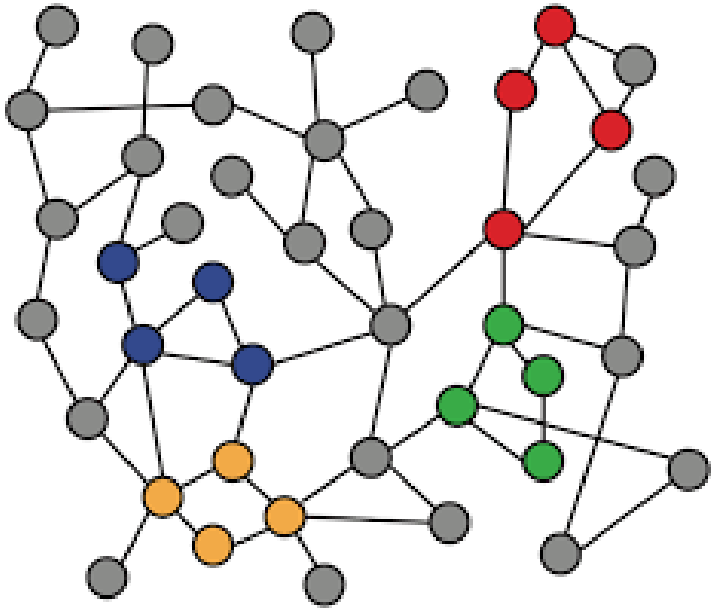
- Node feature
- Edge feature
- Graph feature

- SVM
- Random Forest
- DNN

- Node-level
- Edge-level
- Graph-level

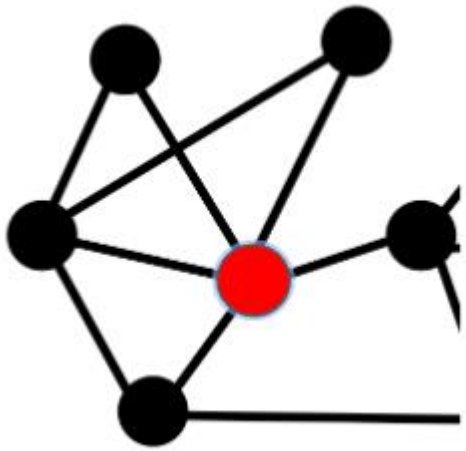
Node-level Features: Overview

- Goal: Characterize the structure and position of a node in the network
  - Importance-based features:
    - Node degree.
    - Node centrality.
  - Structure-based features:
    - Node degree.
    - Clustering coefficient.
    - Graphlets.

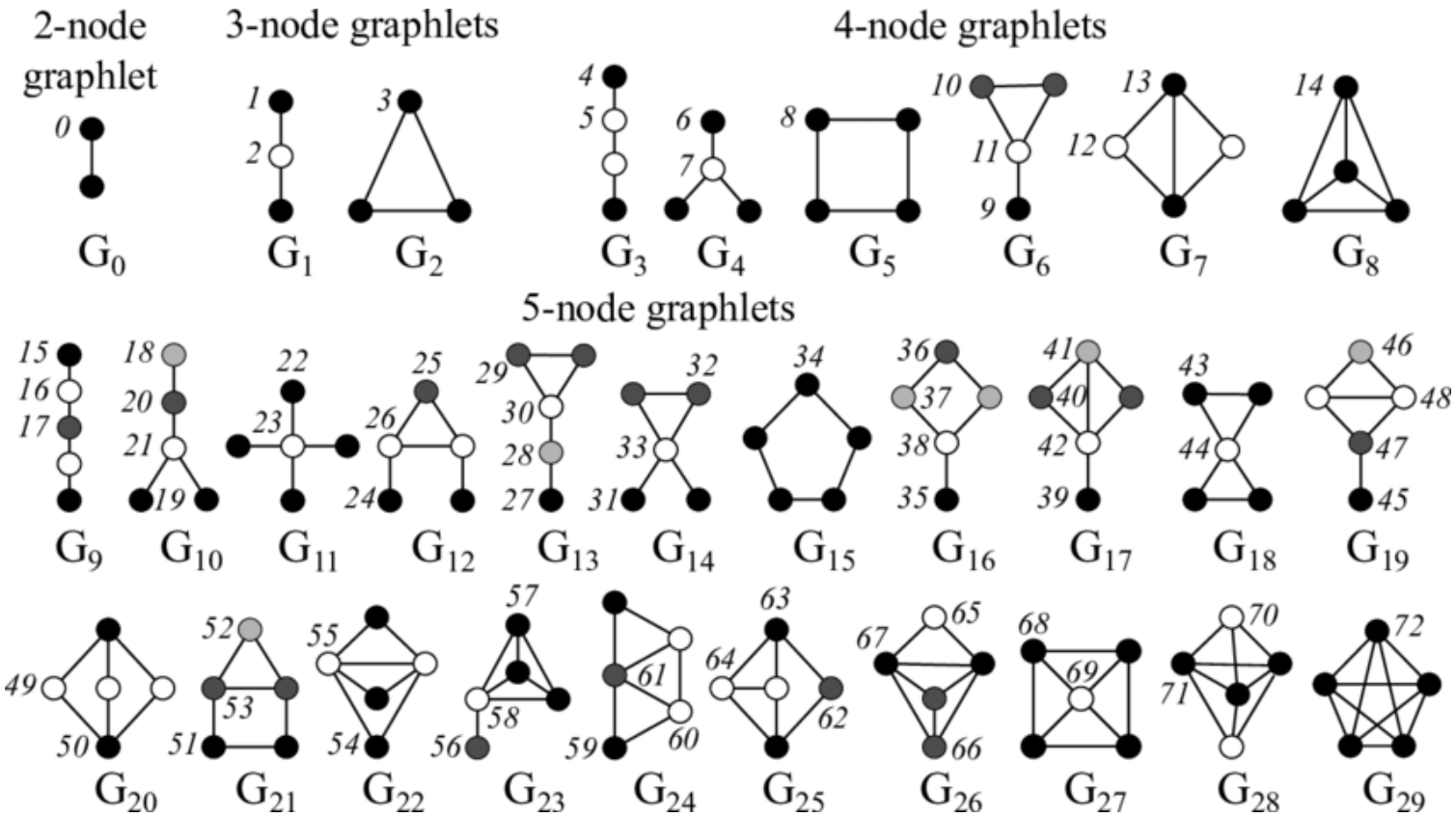


Node Features: Node Centrality

- Node degree counts the neighbouring nodes without capturing their importance.
- Node centrality cv takes the node importance in a graph into account.
- Different ways to evaluate importance:
  - Eigenvector centrality
  - Betweenness centrality
  - Closeness centrality
  - and many others….



- Graphlets are **induced, non-isomorphic** subgraphs that **describe the structure of node u's network neighborhood.**

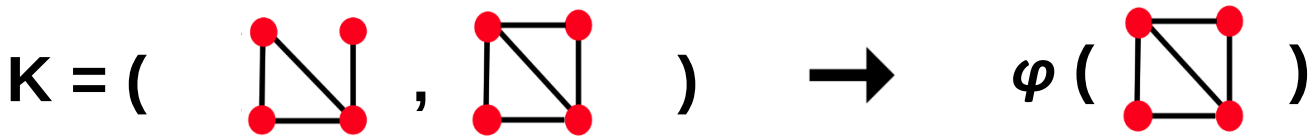


Graph Features: Kernel Methods

- A quick introduction to Kernels:
  - Kernel $K(G,G') \in R$ measures similarity between two graphs (data).
  - There exists a feature representation $\phi(\cdot)$ such that:

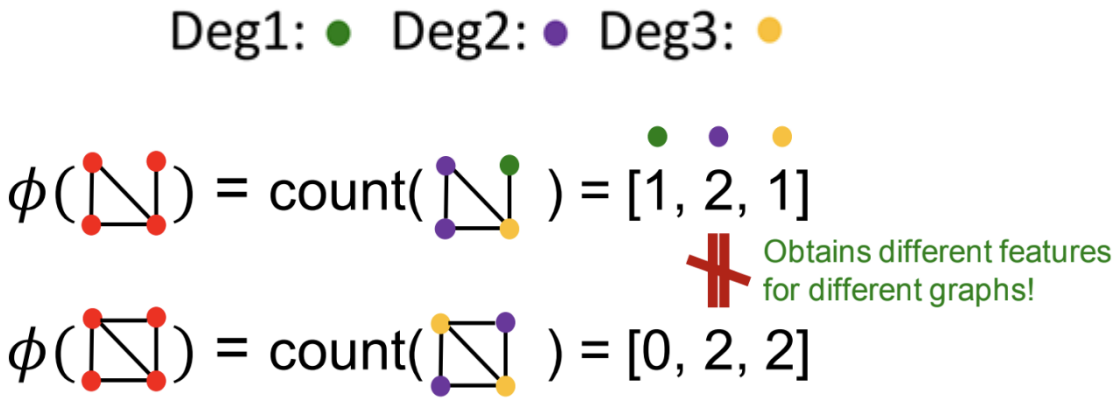
$$K(G, G') = \phi(G)^{T} \phi(G')$$

  - Kernel enables vectors to be operated in higher dimension
  - Once the kernel is defined, off-the-shelf ML model, such as kernel SVM, can be used to make predictions.



Graph Kernel: Key Idea

- Problem: Bag of node counts doesn't work well…
- What if we use Bag of Node Degrees?

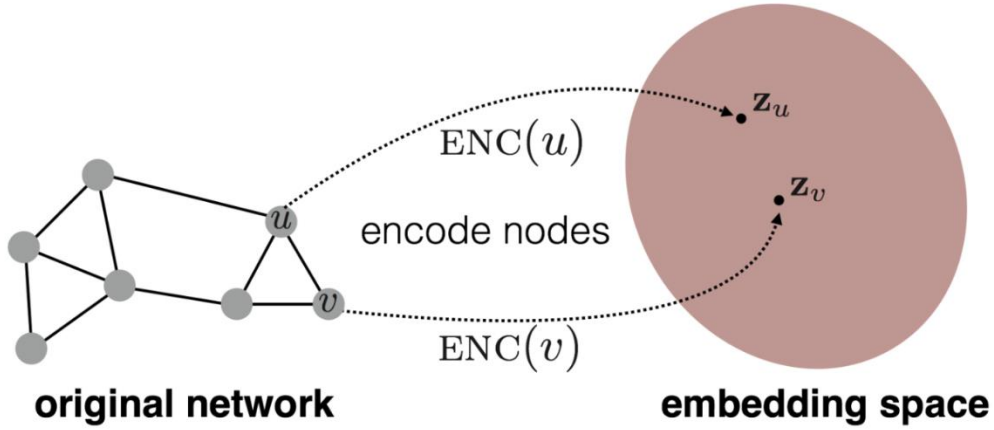


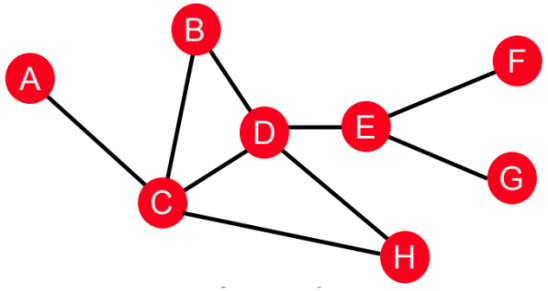
- Both Graph kernel and Weisfeiler-Lehman kernel use Bag-of-* representation of graph, but * is more sophisticated than node degrees!

- Each node is assigned a unique embedding vector.

→ We **directly optimize the embedding of each node.**

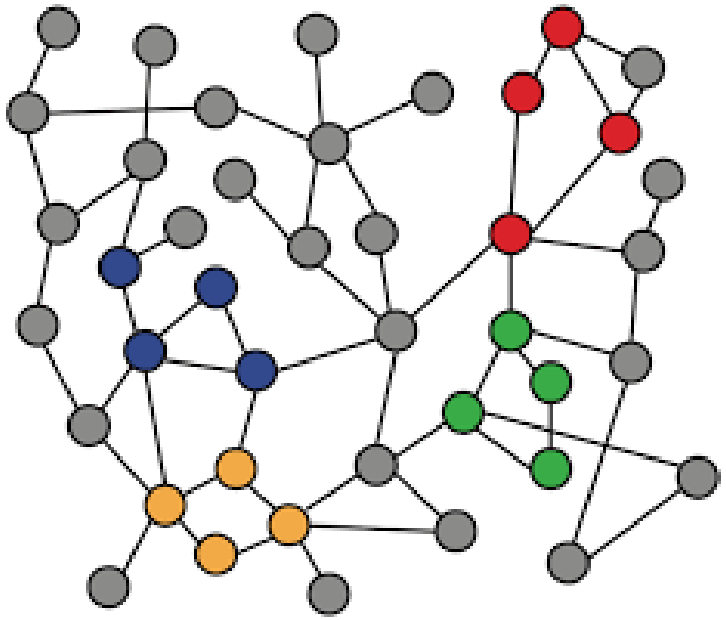
- Embedding is optimized to maximize $z_v^T z_u$ for each similar node pairs (u, v).
- Key choice: How to define node similarity? Should two nodes have similar embedding if:
  - They are linked?
  - They share neighbors?
  - They have similar structure roles?





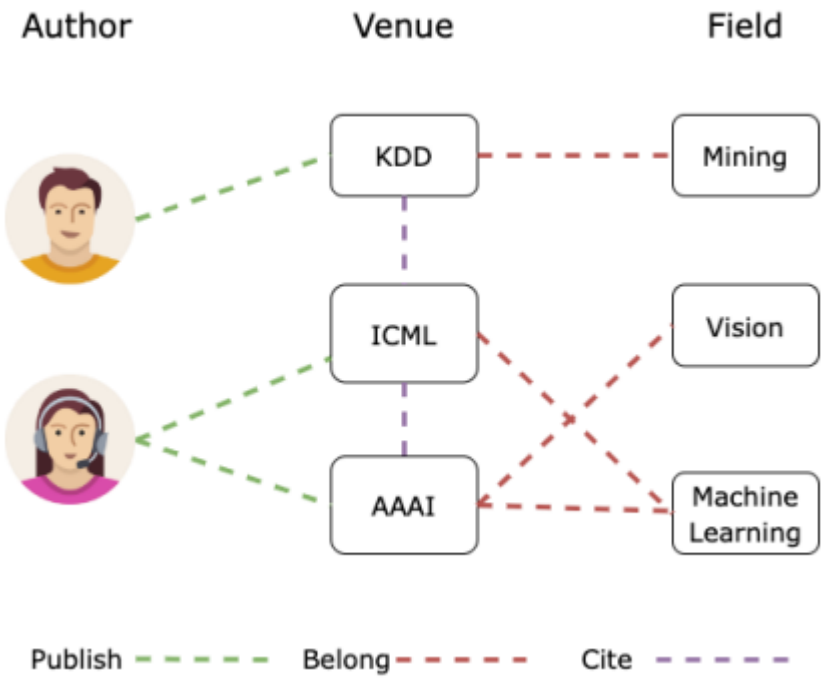
Shallow Methods for Node Embeddings

- Random Walk-based Methods:
  - DeepWalk, Node2vec, Div2Vec,Node2vec+, WalkLet.
- Proximity-based Methods:
  - LINE
- Role-based Methods:
  - Struct2vec, Rol2vec



Many Graphs are Heterogeneous Graphs

- Example: Academic Graph
- Node types: Author, Paper, Venue, Field, ...
- Edge types: Publish, Cite, …
- Benchmark dataset: Microsoft Academic Graph



Shallow Methods for Node Embeddings in Heterogeneous Graphs

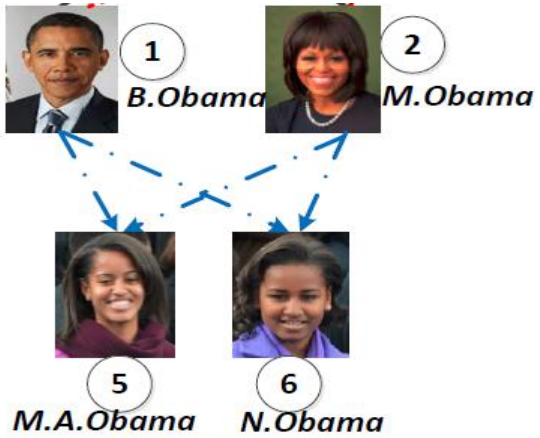
- Meta path and Meta path+
- PathSim

- Objective of Meta Paths:
  - Multi-hop relationships instead of direct links.
  - Combine multiple relationships.

$$m1: \text{USPresident} \xrightarrow{\text{hasChild}} \text{Person} \xrightarrow{\text{hasChild}^{-1}} \text{USFirstLady},$$

$$m2: \text{USPresident} \xrightarrow{\text{memberOf}} \text{USPoliticalParty} \xrightarrow{\text{memberOf}^{-1}} \text{USFirstLady},$$

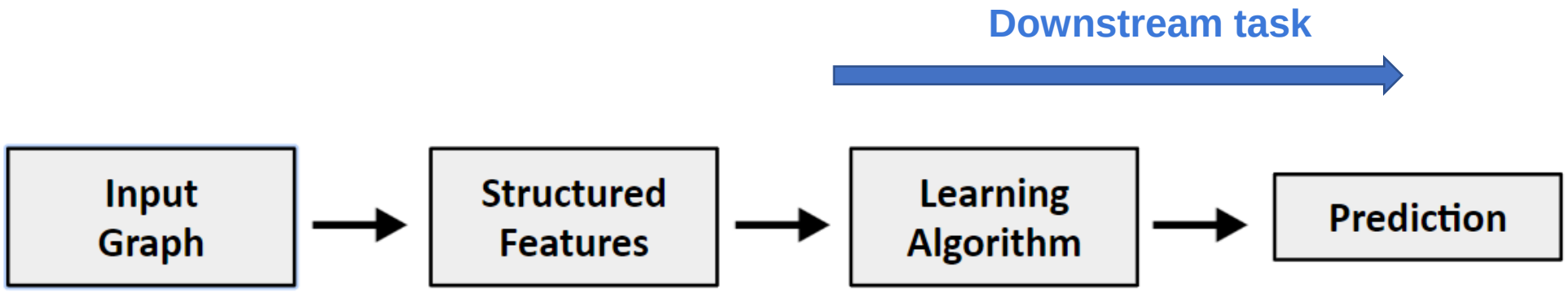
$$m3: \text{USPresident} \xrightarrow{\text{citizenOf}} \text{Country} \xrightarrow{\text{citizenOf}^{-1}} \text{USFirstLady}.$$



Yuxiao Dong et al:"metapath2vec: Scalable representation learning for heterogeneous networks." *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017.

What after Node Embeddings?

- Once we have node embeddings (independent to task), we can continue to the downstream prediction.



Shallow Encoding

- SVM
- Random Forest
- XGBoost
- DNN

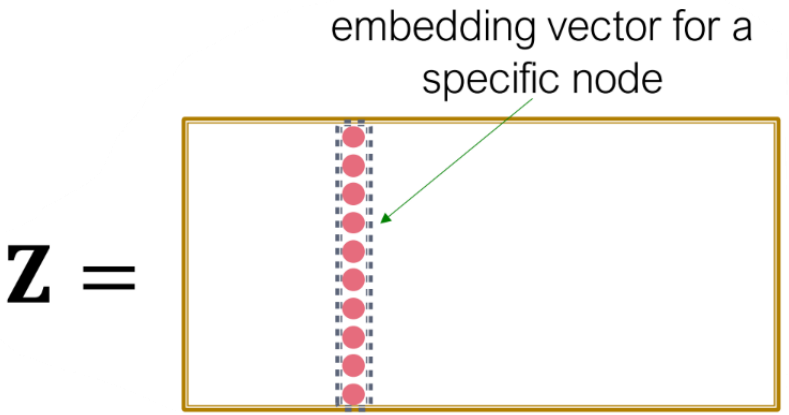
- Node-level
- Edge-level
- Graph-level

How to use these node Embeddings?

- **Given a node v_i in a graph, we have its embedding Z_i, we can do:**
  1. Clustering/community detection: Cluster $\boldsymbol{Z_i}$.
  2. Node classification: Predict label of node v_i based on $\boldsymbol{Z_i}$.
  3. Link prediction: Predict edge (v_i, v_j) based on $(\boldsymbol{Z_i}, \boldsymbol{Z_j})$.
     - Concatenate: f($\boldsymbol{Z_i}$, $\boldsymbol{Z_j}$) = g([$\boldsymbol{Z_i}$, $\boldsymbol{Z_j}$]).
     - Hadamard: f($\boldsymbol{Z_i}$, $\boldsymbol{Z_j}$) = g($\boldsymbol{Z_i} * \boldsymbol{Z_j}$) (per position product).
     - Sum/Average: f($\boldsymbol{Z_i}$, $\boldsymbol{Z_j}$) = g($\boldsymbol{Z_i} + \boldsymbol{Z_j}$).
     - Distance: f($\boldsymbol{Z_i}$, $\boldsymbol{Z_j}$) = g($\|\boldsymbol{Z_i} - \boldsymbol{Z_j}\|_2$).
  4. Graph classification: aggregate node embeddings to form graph embedding $\boldsymbol{Z_G}$. Predict graph label based on graph embedding $\boldsymbol{Z_G}$.

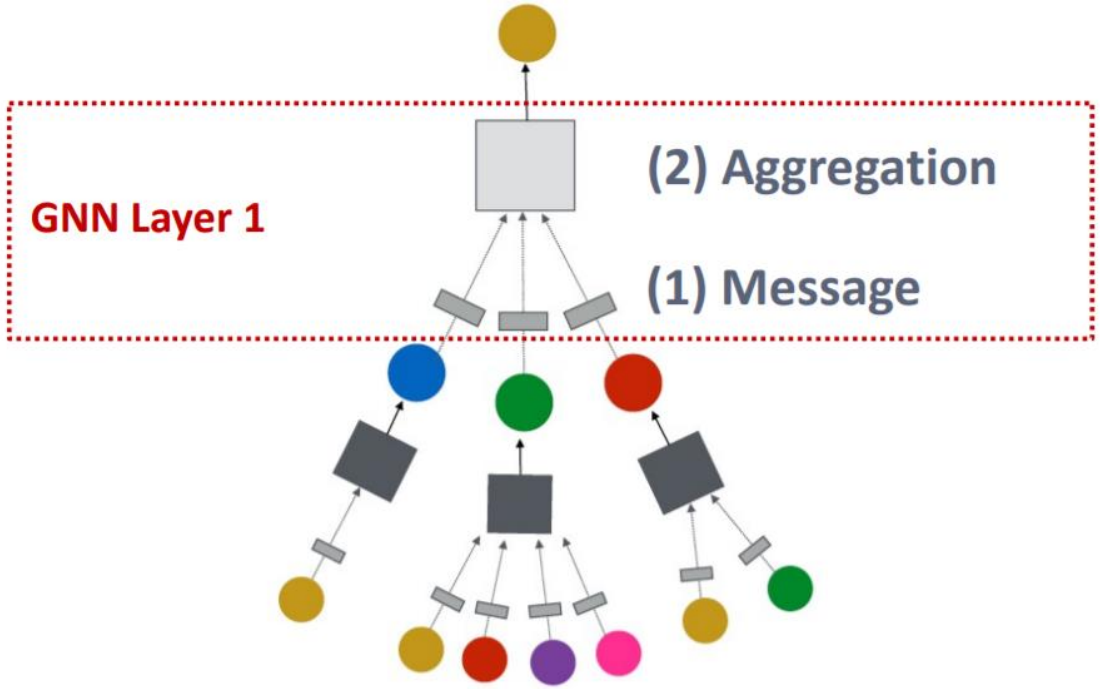
Shallow Encoding Summary

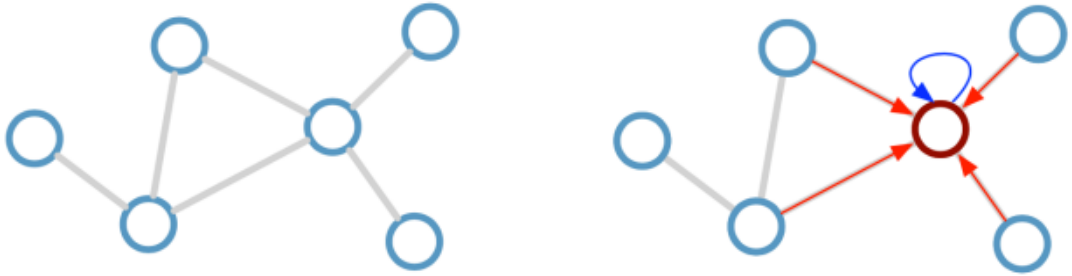
- **Goal**: Generate a lookup table for node embeddings
- Step-by-step:
  1. Run random walks for each node
  2. Collect the set of visited nodes for each node on the walks
  3. Optimize the embeddings Z_u using stochastic gradient descent



Graph Convolutional Networks (GCNs)

- GNN Layer = **Message** + **Aggregation**
  - Message COMPUTATION
    - How to make each neighborhood node as embedding?
  - Message AGGERGATION
    - How to combine those embeddings?





Update rule: $$\mathbf{h}_i^{(l+1)} = \sigma\left(\mathbf{h}_i^{(l)}\mathbf{W}_0^{(l)} + \sum_{j\in\mathcal{N}_i}\frac{1}{c_{ij}}\mathbf{h}_j^{(l)}\mathbf{W}_1^{(l)}\right)$$

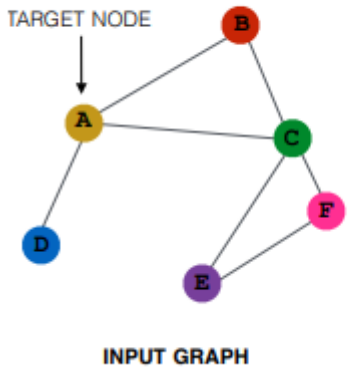
Kipf & Welling (ICLR 2017), related previous works by Duvenaud et al. (NIPS 2015) and Li et al. (ICLR 2016)

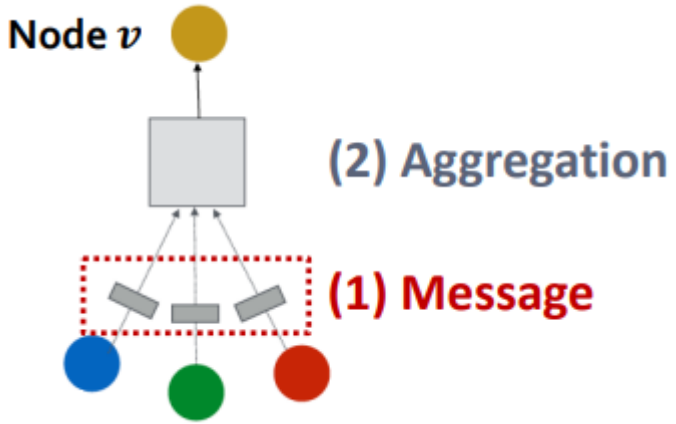
Message Computation

- Intuition: Each node will create a message, which will be sent to other nodes later

- Example: A Linear layer $\mathbf{m}_u^{(l)} = \mathbf{W}^{(l)}\mathbf{h}_u^{(l-1)}$
  - Multiply node features with weight matrix $\mathbf{W}^{(l)}$

Message function: $\mathbf{m}_u^{(l)} = \mathrm{MSG}^{(l)}\left(\mathbf{h}_u^{(l-1)}\right)$





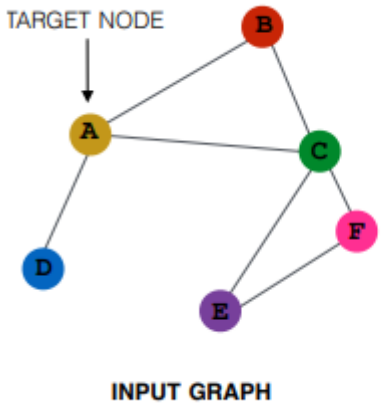
Message Aggregation

- **Intuition**: Each node will aggregate the messages from node v's neighbors

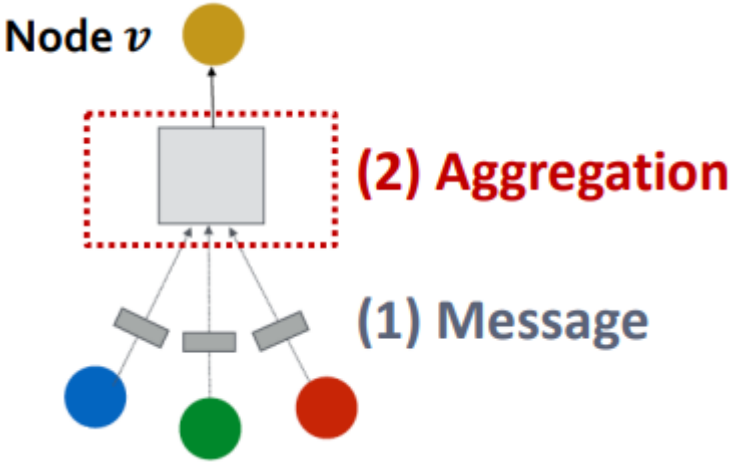
$$\mathbf{h}_v^{(l)} = \mathrm{AGG}^{(l)}\left(\left\{\mathbf{m}_u^{(l)}, u \in N(v)\right\}\right)$$

- **Example**: Sum(·), Mean(·) or Max(·) aggregator

$$\mathbf{h}_v^{(l)} = \mathrm{Sum}(\{\mathbf{m}_u^{(l)}, u \in N(v)\})$$



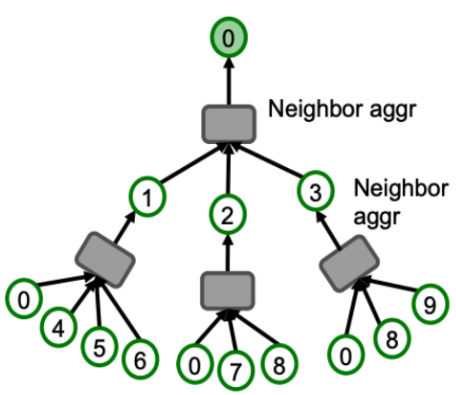
INPUT GRAPH

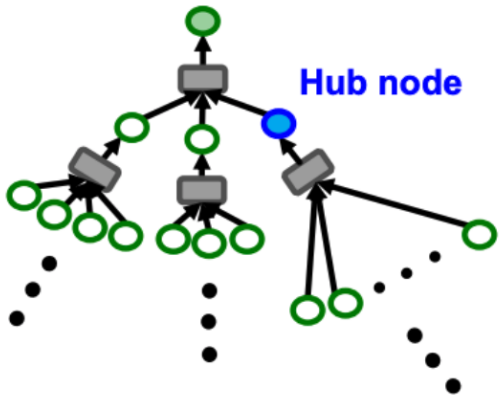


Graph Convolutional Network Limitations

- **Computationally Expensive:**
  - We need to generate the complete K-hop neighborhood computational graph and then need to aggregate plenty of information from its surroundings.
  - As we go deeper into the neighborhood computation graph becomes exponentially large.

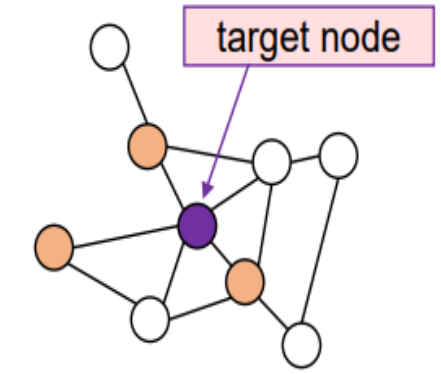
  → problem while fitting these computational graphs inside GPU memory.
- **The curse of Hub nodes or Celebrity nodes:**
  - Hub nodes are those nodes which are very high degree nodes in the graph



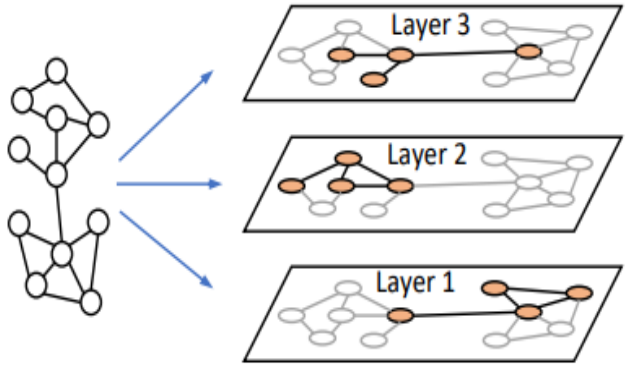


Scalability of Graph Neural Networks

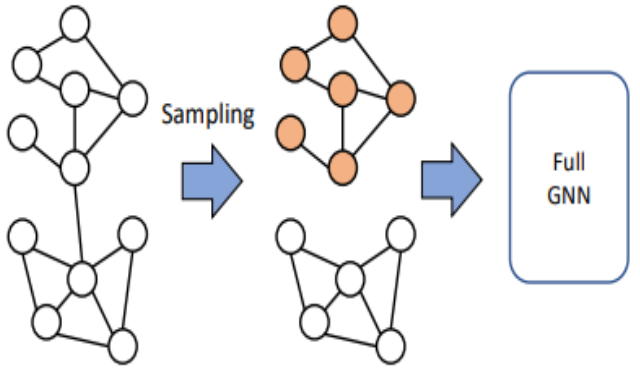
- **Why the original GNN fails on large graph?**
  - Large memory requirement.
  - Inefficient gradient update.
- **Three paradigms toward large-scale GNN:**
  - Node-wise sampling
  - Layer-wise sampling
  - Graph-wise sampling



Node-wise Sampling



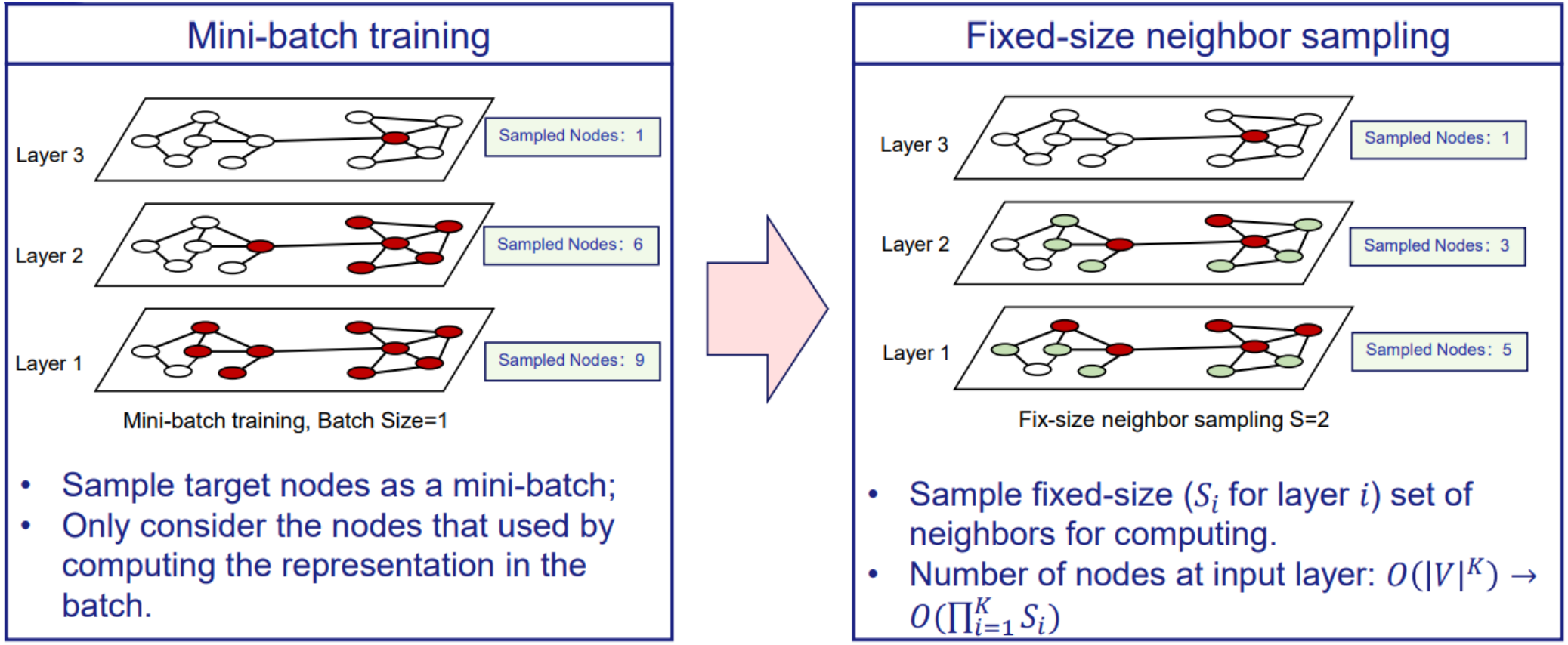
Layer-wise Sampling



Graph-wise Sampling

Scalability of GNNs: GraphSAGE

- Towards large-scale **GraphSAGE**:
  - Sampling mini-batch (sample target nodes as a mini-batch)
  - Sampling a fixed size set for each target nodes



Scalability of GNNs: GraphSAGE Differences

- Simple neighborhood aggregation:

$$\mathbf{h}_v^k = \sigma\left(\mathbf{W}_k \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|} + \mathbf{B}_k \mathbf{h}_v^{k-1}\right)$$

- GraphSAGE:

concatenate self embedding and neighbor embedding

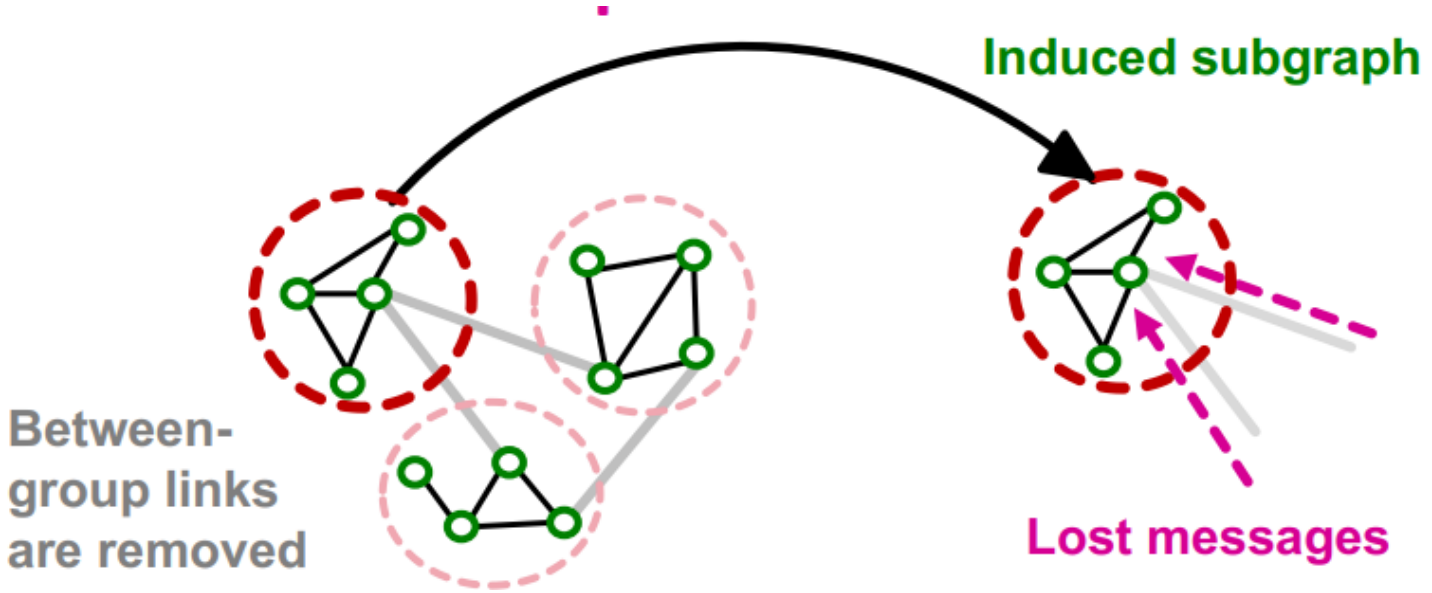
$$\mathbf{h}_v^k = \sigma\left(\left[\mathbf{A}_k \cdot \mathrm{AGG}\left(\{\mathbf{h}_u^{k-1}, \forall u \in N(v)\}\right), \mathbf{B}_k \mathbf{h}_v^{k-1}\right]\right)$$

generalized aggregation

Neighborhood sampling

Scalability of GNNs: Graph-wise Sampling - ClusterGCN

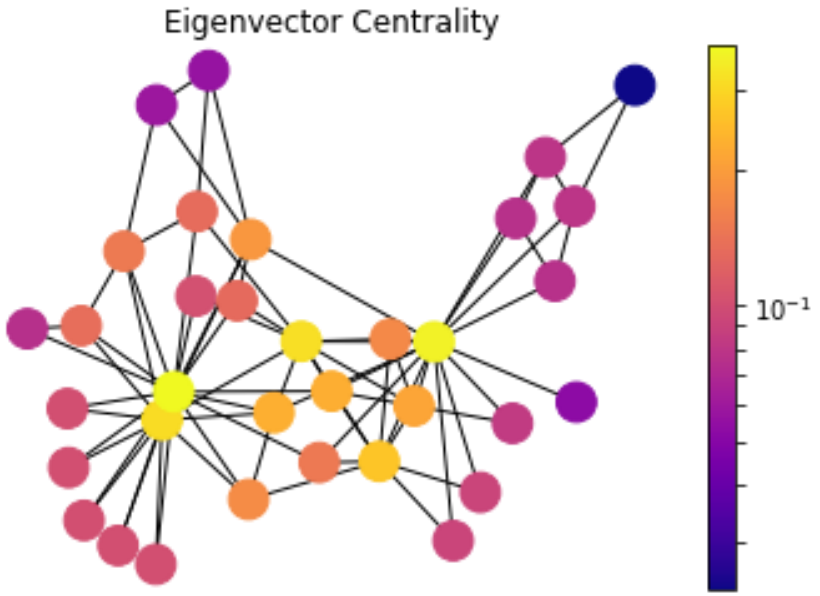
- **Problems**:
  - The induced subgraph removes between group links.
  - As a result, messages from other groups will be lost during message passing, which could hurt the GNN's performance.



Chiang, Wei-Lin, et al. "Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks."

Variants of GNNs: Why Attention in GNNs?

- GNN compute node representations from representations of neighbours.
- Nodes can have largely different neighbourhood sizes.
- Not all neighbours have relevant information for a certain node.
- Attention mechanism allow to adaptively weight the contribution of each neighbour when updating a node.



Analysis of Graph Attention: Computation Steps (1)

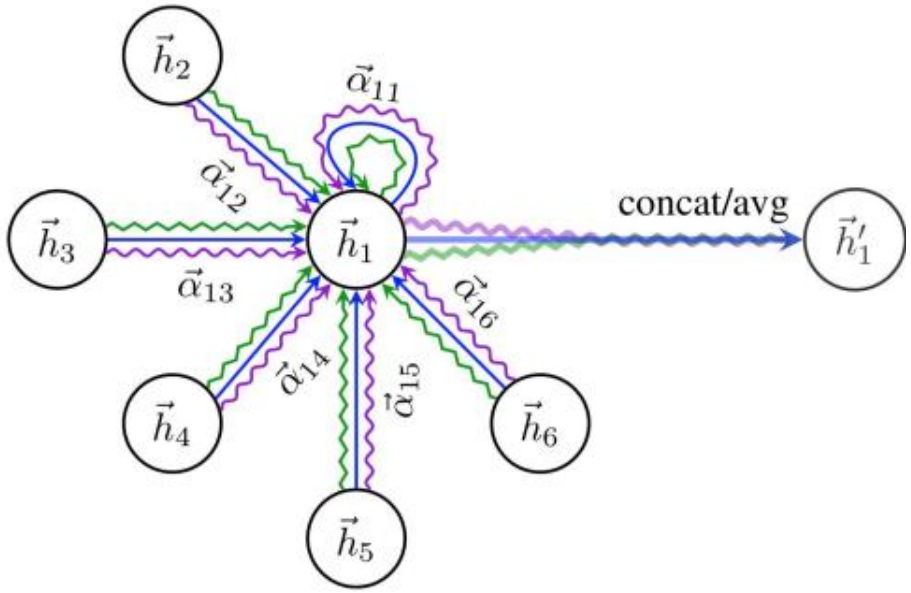
- Input node features: Each node in the graph has a feature vector.

$$\mathbf{h} = \{\vec{h}_1, \vec{h}_2, \ldots, \vec{h}_N\}, \vec{h}_i \in \mathbb{R}^F$$

- Calculate energy (co-efficient) between two nodes

$$e_{ij} = a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j)$$

a: attention function



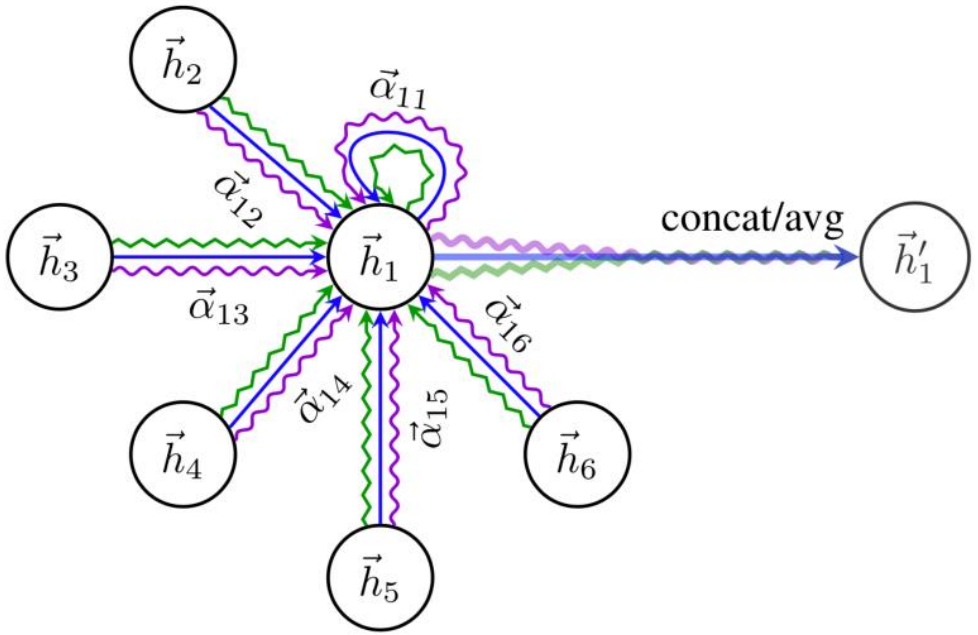
Analysis of Graph Attention: Computation Steps (2)

- Attention score (over the neighbors): Normalize over all the neighbors

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k]\right)\right)}$$

- Multi-head attention
  - Feature concatenation

$$\vec{h}'_i = \Big\Vert_{k=1}^{K} \sigma\left(\sum_{j\in\mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j\right)$$

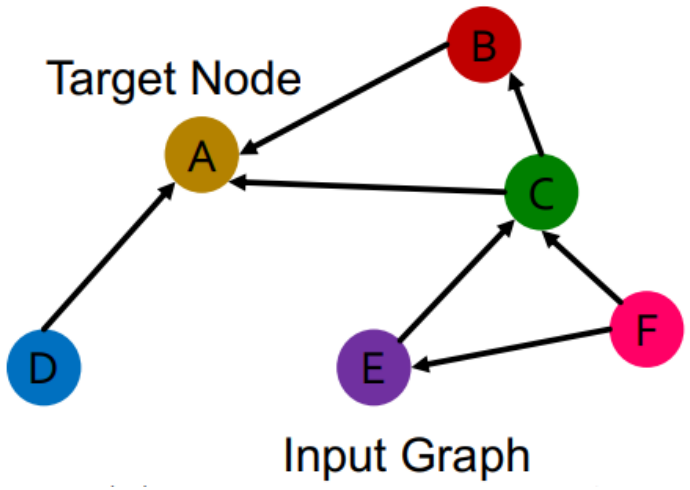


  - Feature averaging (for the final layer)

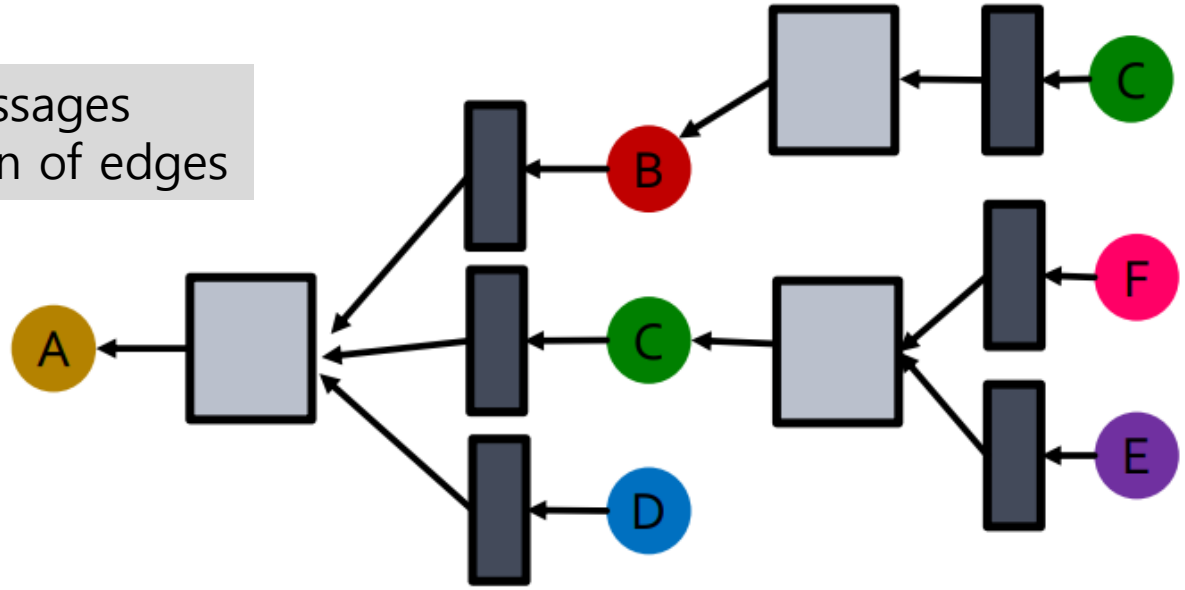
$$\vec{h}'_i = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j\right)$$

Variants of GNNs: Relational GCNs

- We will extend GCN to handle heterogeneous graphs with multiple edge/relation types
- We start with a directed graph with one relation
  - How do we run GCN and update the representation of the target node A on this graph?



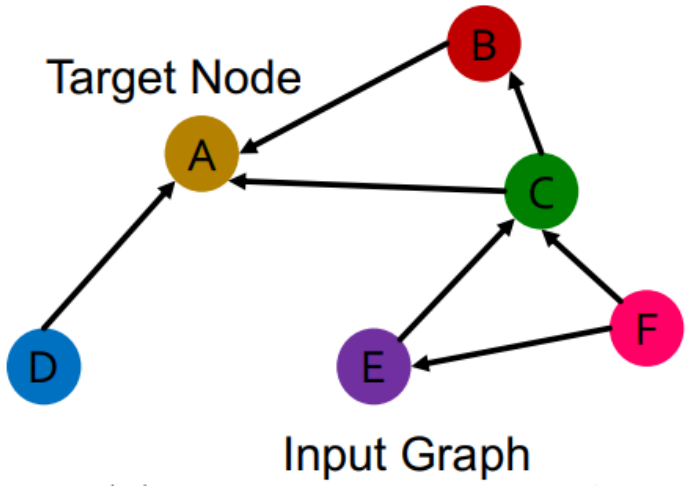
Only pass messages along direction of edges



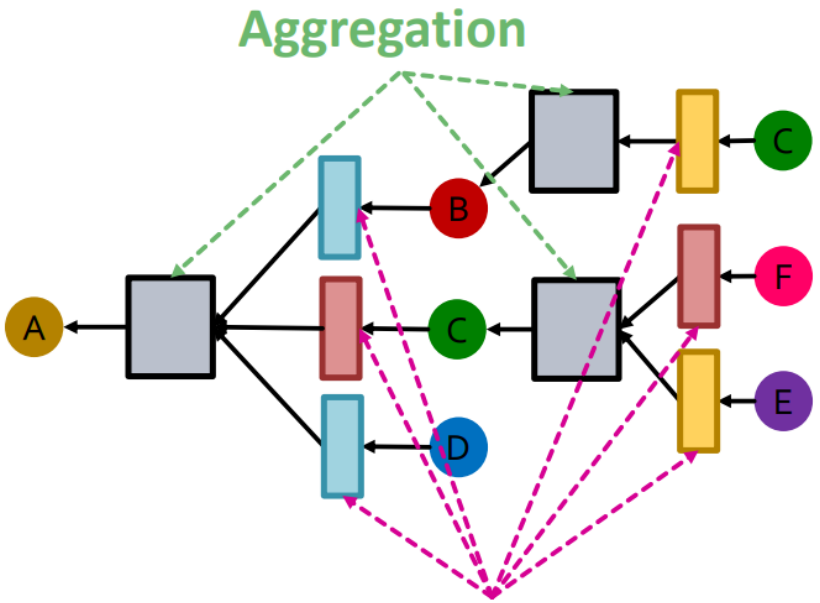
Schlichtkrull et al., Modeling Relational Data with Graph Convolutional Networks, ESWC 2018

Variants of GNNs: Relational GCNs

- What if the graph has multiple relation types?
- Use different neural network weights for different relation types.



Input Graph



Neural networks

Variants of GNNs: Relational GCNs

- Relational GCN (RGCN):

$$\mathbf{h}_v^{(l+1)} = \sigma\left(\sum_{r\in R}\sum_{u\in N_v^r}\frac{1}{c_{v,r}}\mathbf{W}_r^{(l)}\mathbf{h}_u^{(l)} + \mathbf{W}_0^{(l)}\mathbf{h}_v^{(l)}\right)$$

- How to write this as Message + Aggregation?
  - **Message**: Each neighbor of a given relation & Self-loop::

$$\mathbf{m}_{u,r}^{(l)} = \frac{1}{c_{v,r}}\mathbf{W}_r^{(l)}\mathbf{h}_u^{(l)} \qquad \mathbf{m}_v^{(l)} = \mathbf{W}_0^{(l)}\mathbf{h}_v^{(l)}$$

  - **Aggregation**: Sum over messages from neighbors and self-loop, then apply activation

$$\mathbf{h}_v^{(l+1)} = \sigma\left(\text{Sum}\left(\left\{\mathbf{m}_{u,r}^{(l)}, u \in N(v)\right\} \cup \left\{\mathbf{m}_v^{(l)}\right\}\right)\right)$$

Relational GCNs vs. GCNs

- How to define Message + Aggregation?

$$\mathbf{h}_v^{(l+1)} = \sigma\left(\sum_{r\in R}\sum_{u\in N_v^r}\frac{1}{c_{v,r}}\mathbf{W}_r^{(l)}\mathbf{h}_u^{(l)} + \mathbf{W}_0^{(l)}\mathbf{h}_v^{(l)}\right)$$

$$\mathbf{h}_v^{(l)} = \sigma\left(\mathbf{W}^{(l)}\sum_{u\in N(v)}\frac{\mathbf{h}_u^{(l-1)}}{|N(v)|}\right)$$

- Aggregation:

$$\mathbf{h}_v^{(l+1)} = \sigma\left(\text{Sum}\left(\left\{\mathbf{m}_{u,r}^{(l)}, u\in N(v)\right\}\cup\left\{\mathbf{m}_v^{(l)}\right\}\right)\right)$$

$$\mathbf{h}_v^{(l)} = \sigma\left(\sum_{u\in N(v)}\mathbf{W}^{(l)}\frac{\mathbf{h}_u^{(l-1)}}{|N(v)|}\right)$$

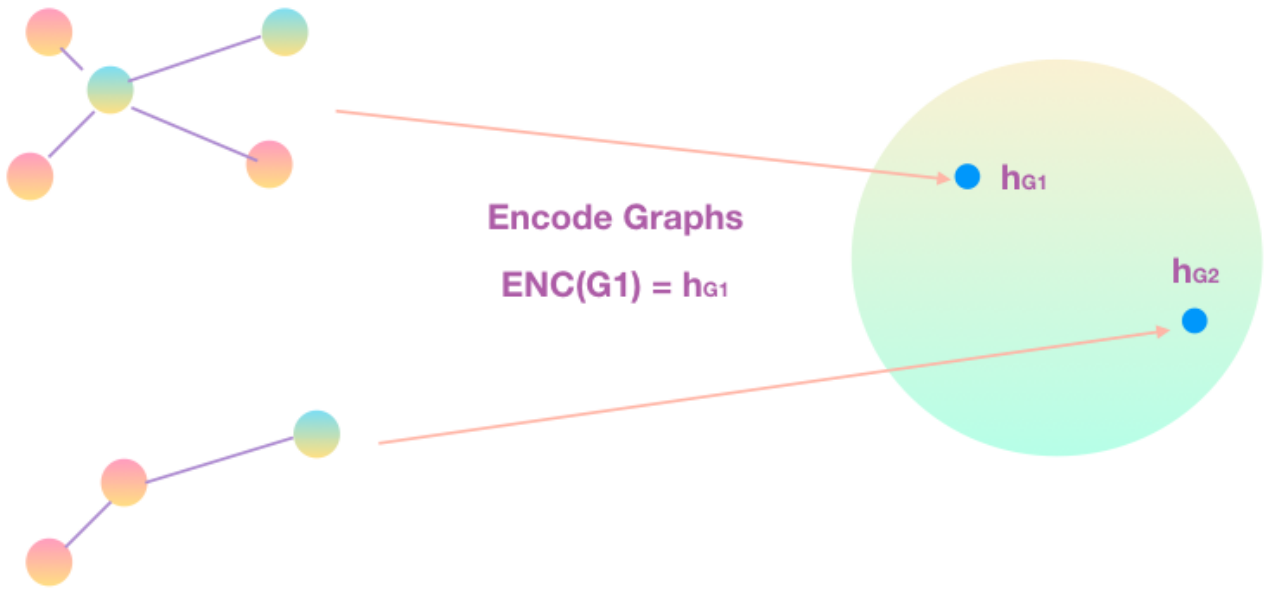
Message

Aggregation

Relational GCN

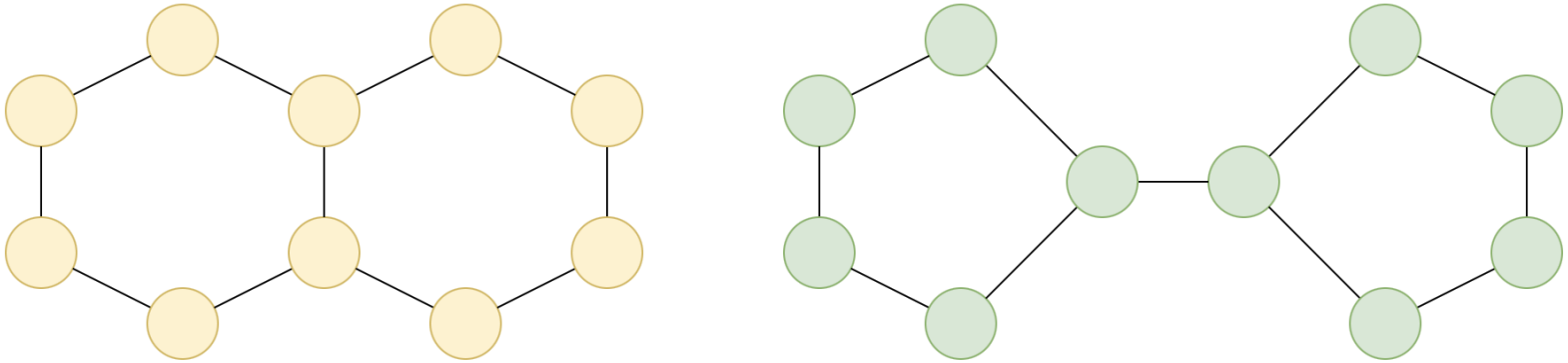
GCN

- **Do similar graphs really have the same label?**
  - The Graph Classification problem and the Graph Isomorphism problem are not the same.
  - Graph Isomorphism can be judged as isomorphic if the positions on the vector are close, but Graph Classification adds an element called "Label" to the graph, so that it is classified by the same Label, not by position.



Structure preservation: Understanding Expressiveness

- Most GNNs fail to distinguish the difference between two graphs
- Examples of two graphs indistinguishable by the WL test.
  - They produce similar distributions through the iterations of colour refinement.
  - It's catastrophic if datasets have graphs that exhibit similar properties and can't be told apart.



Structure preservation: What makes GIN powerful?

- Employing a sum aggregation function, which is injective.
- Model the $f^{(k+1)} \circ \phi^{(k)}$ with one MLP:

$$h_v^{(k)} = \mathrm{MLP}^{(k)}((1+\epsilon^{(k)}).h_v^{(k-1)} + \Sigma_{u \in \mathcal{N}(v)} h_u^{(k-1)})$$

ϵ can be a learnable parameter or a scalar

- **READOUT:**
  - More iterations gives better representational power
  - But less generalization
  - So GIN concatenates the embedding (information) from all layers

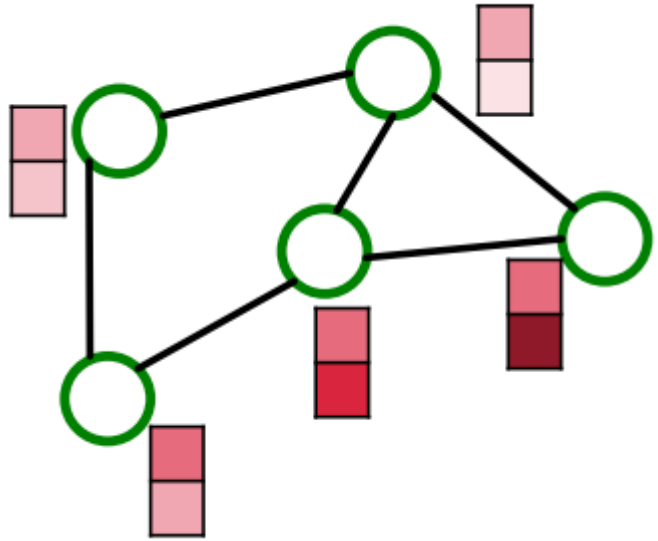
- A graph Transformer must take the following inputs:
  - (1) Node features?
  - (2) Adjacency information?
  - (3) Edge features (if any)

- Key components of Transformer:
  - (a) tokenizing
  - (b) positional encoding
  - (c) self-attention

SOLUTIONS:

There are many ways to do this:

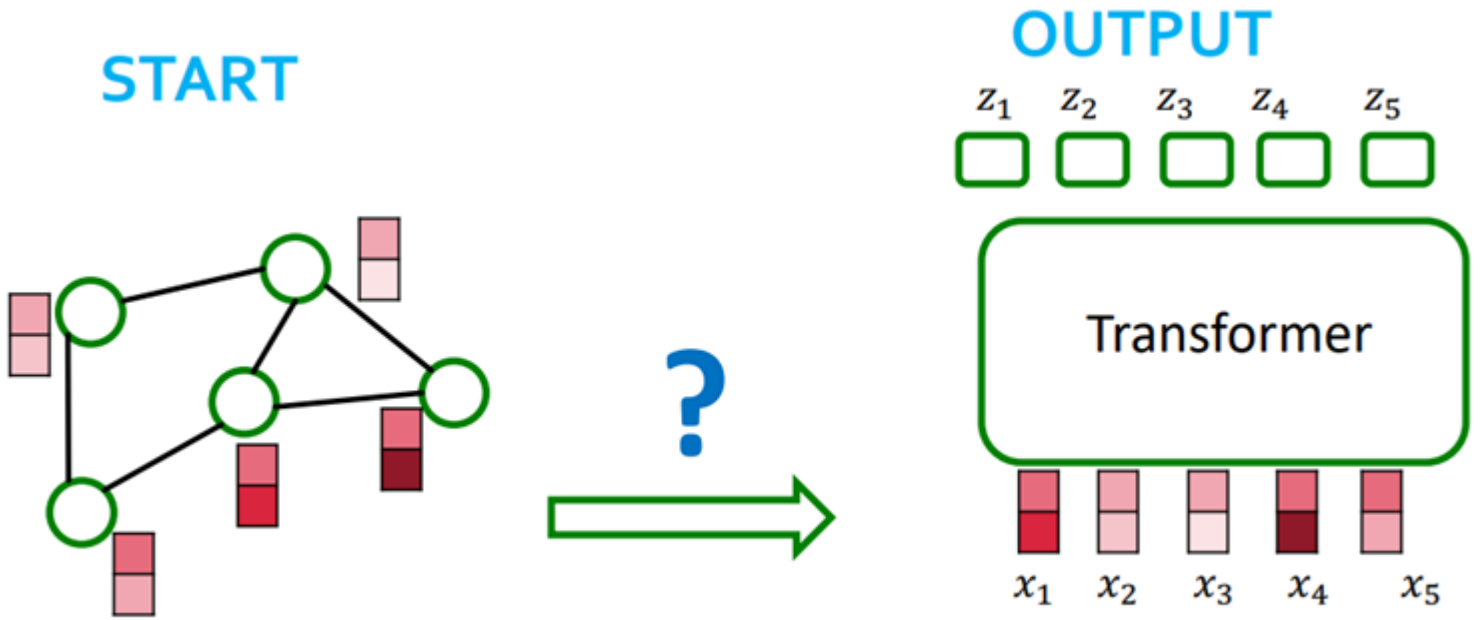
- Different approaches correspond to different “matchings” between graph inputs (1), (2), (3) transformer components (a), (b), (c)



Graphs Transformers: Nodes as Tokens

Q1: what should our tokens be?

- Sensible Idea: node features = input tokens
- This matches the setting for the “attention is message passing on the fully connected graph” observation
- **Problem**? We completely lose adjacency info!
- **How to also inject adjacency information?**

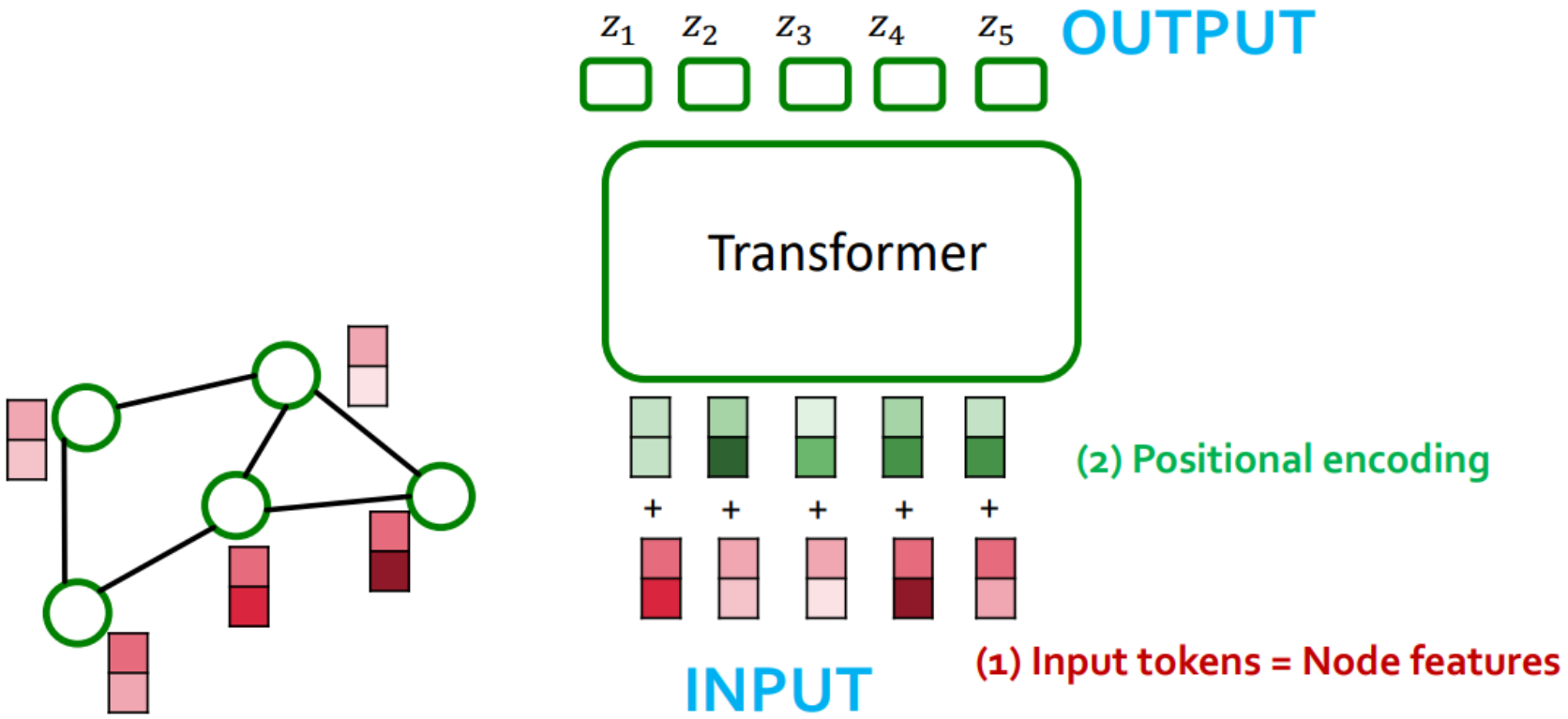


(1) Input tokens = Node features

Nodes as Tokens: How to add back Adjacency information

Problem: We completely lose adjacency info!

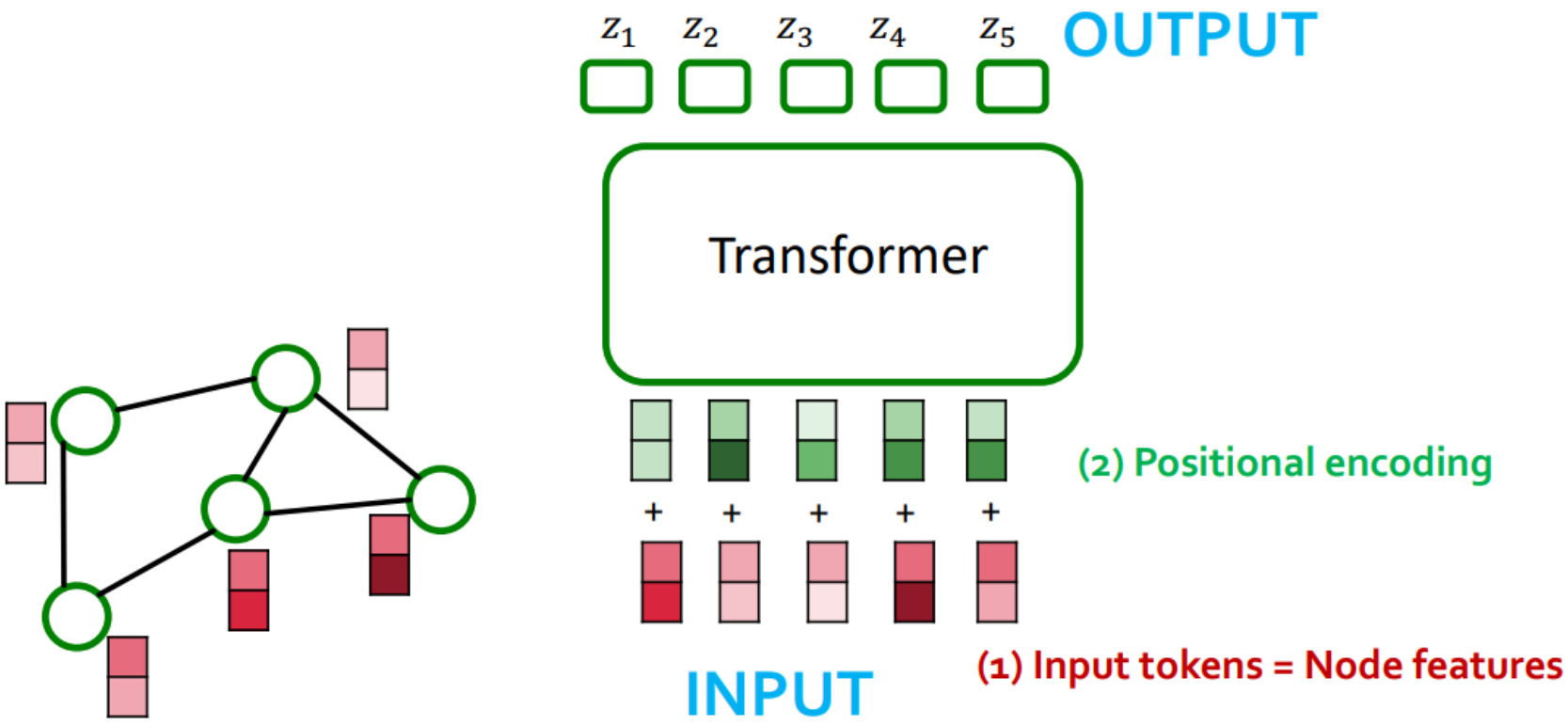
- **How to also inject adjacency information?**
- **Idea:** Encode adjacency info in the positional encoding for each node
- Positional encoding describes where a node is in the graph



Nodes as Tokens: How to add back Adjacency information

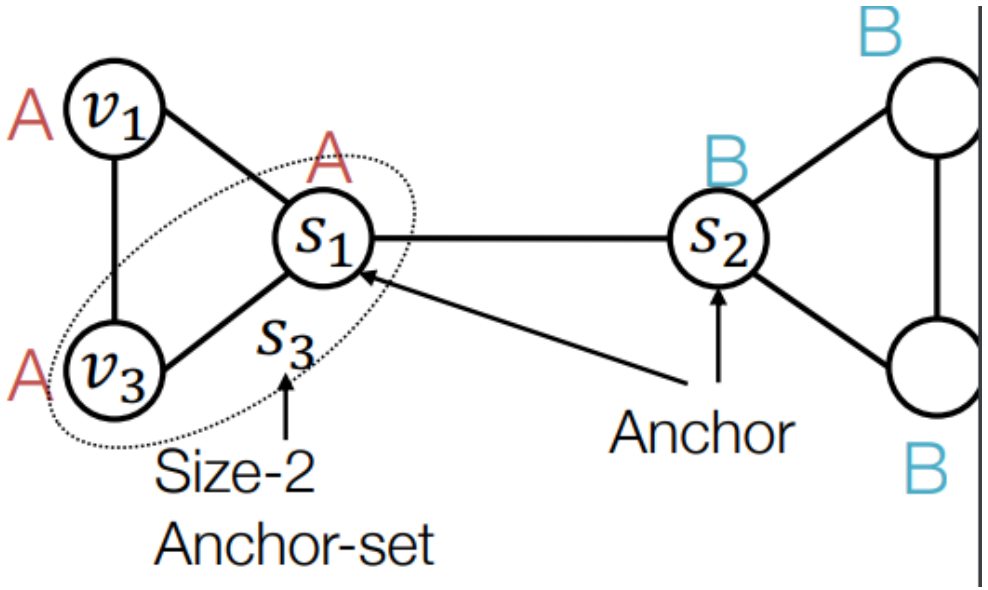
Q2: How to design a good positional encoding?

- Option 1: relative distance
- Option 2: Laplacian Eigenvector PE



Graphs Transformers: Nodes as Tokens- Relative distance PE

- Similar methods based on random walks
- This is a good idea. It works well in many cases
- Especially strong for tasks that require counting cycles



Positional encoding for node v_1 =

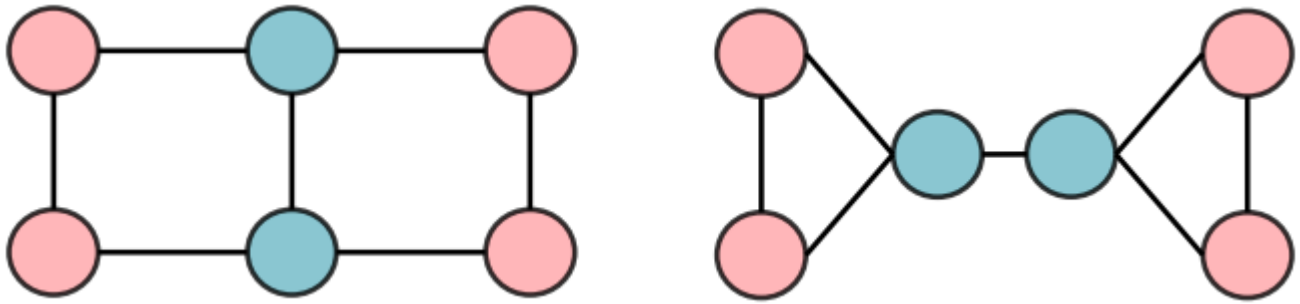
Relative Distances

	s_1	s_2	s_3
v_1	1	2	1
v_3	1	2	0

Anchor s_1, s_2 cannot differentiate node v_1, v_3, but anchor-set s_3 can

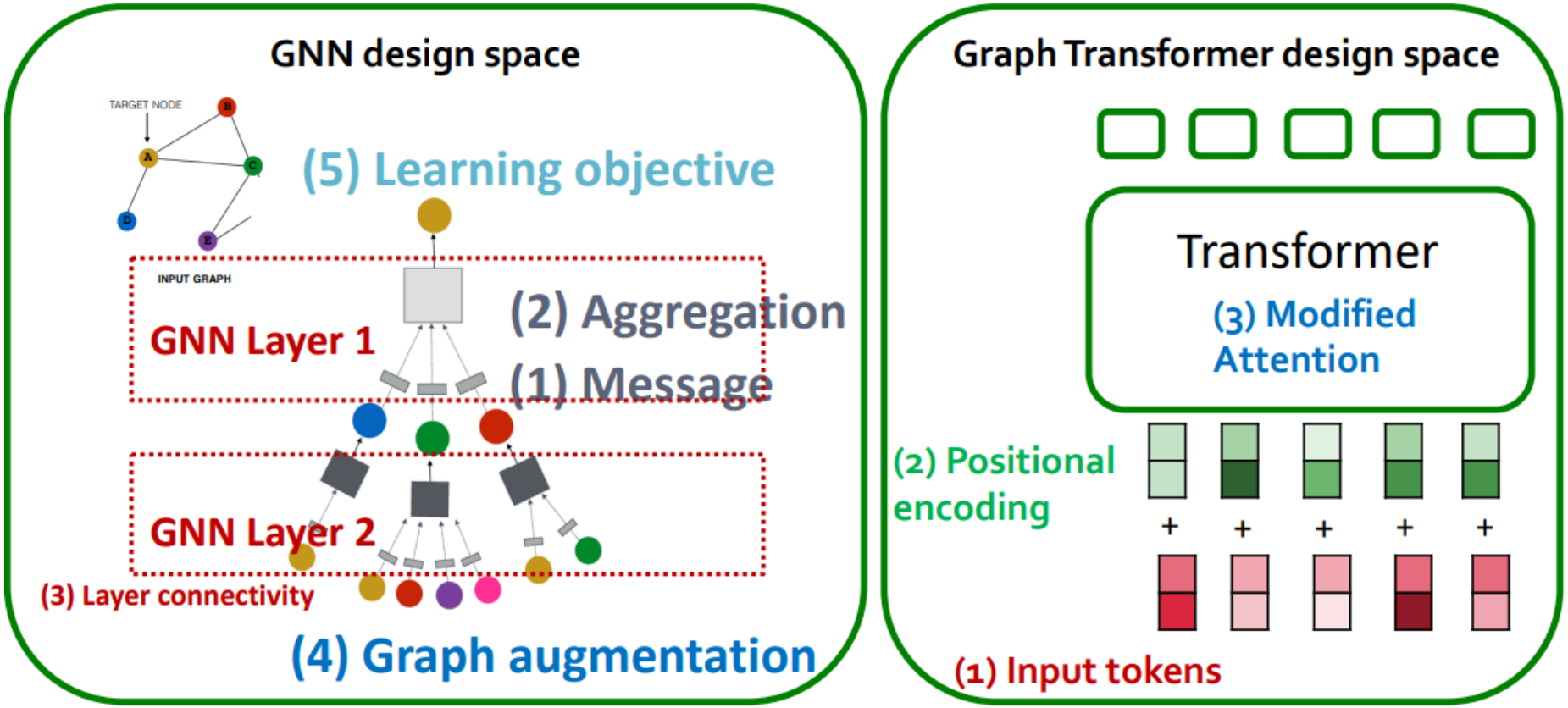
Graphs Transformers: Relative distance PE

- Relative distances useful for position-aware task
- SPD can be used to improve WL-Test:
  - These two graphs cannot be distinguished by 1-WL-test.
  - But the SPD sets, i.e., the SPD from each node to others, are different:
  - The two types of nodes in the left graph have SPD sets {0, 1, 1, 2, 2, 3} , {0, 1, 1, 1, 2, 2} while the nodes in the right graph have SPD sets {0, 1, 1, 2, 3, 3} , {0, 1, 1, 1, 2, 2}.



Summary: Comparing Transformer and GNNs

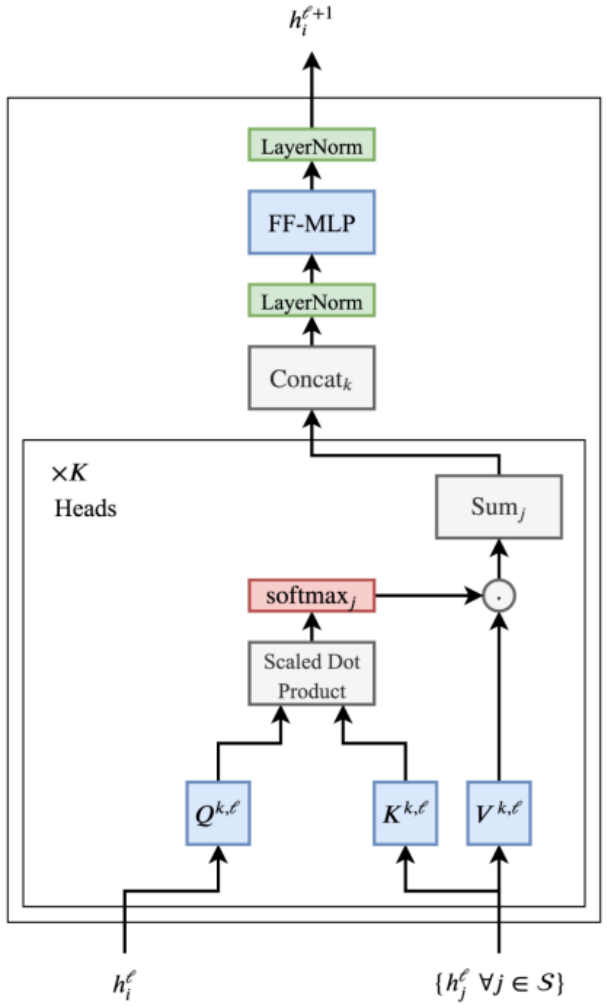
- Transformer are GNNs



https://web.stanford.edu/class/cs224w/slides/14-graph-transformer.pdf

Why Graph Transformers are GNNs?

- **Breaking down the Transformer**: Update each node's features through Multi-head Attention mechanism as a weighted sum of features of other words in the sentence.
  - Scaling dot product attention
  - Normalization layers
  - Residual links



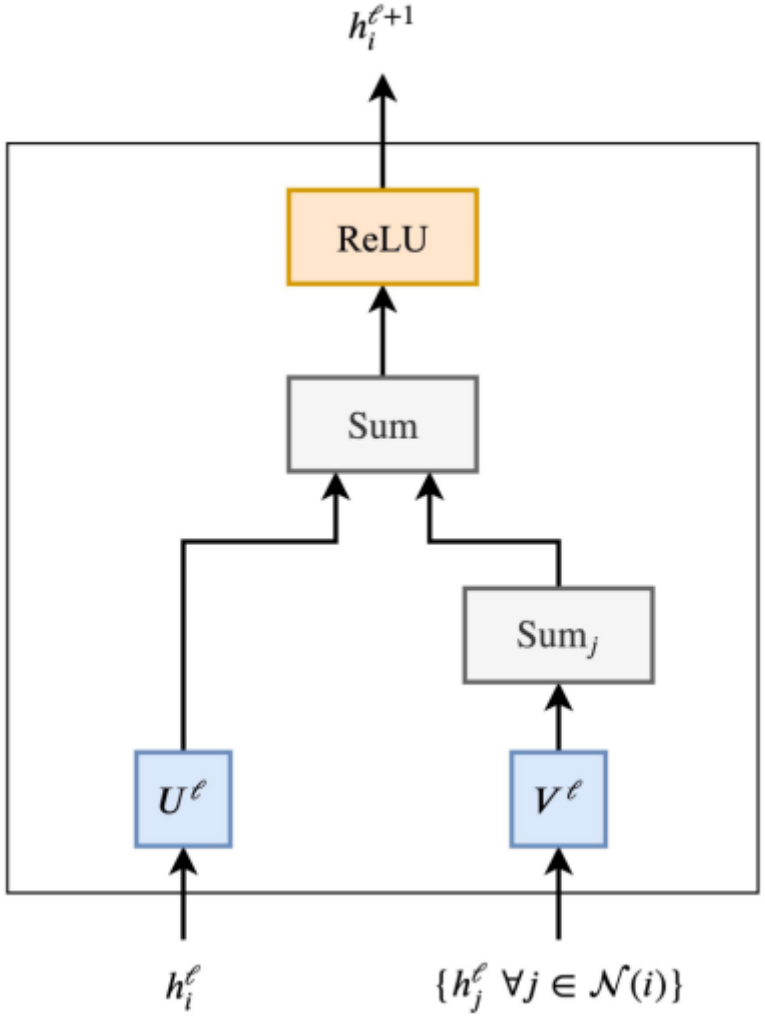
Transformer

Why Graph Transformers are GNNs?

- **Breaking down the GNNs**: GNNs update the hidden features h of node i at layer l via a non-linear transformation of the node's own features added to the aggregation of features from each neighbouring node $j \in N(i)$:

$$h_i^{\ell+1} = \sigma\Big(U^{\ell} h_i^{\ell} + \sum_{j \in \mathcal{N}(i)} \big(V^{\ell} h_j^{\ell} \big) \Big),$$

- where U, V are learnable weight matrices of the GNN layer and σ is a non-linearity.



GNNs

Why Graph Transformers are GNNs?

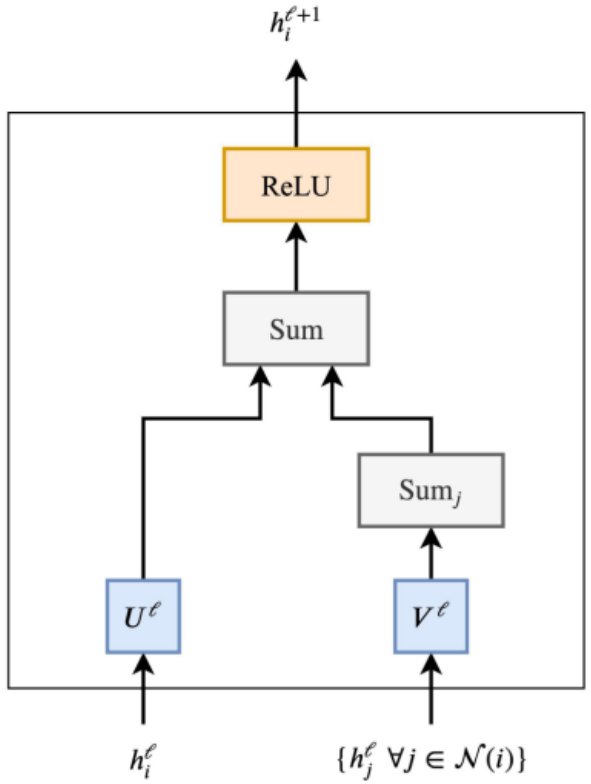
- **Breaking down the Transformer and GNNs**:
- **GNNs**:

$$h_i^{\ell+1} = \sigma\Big(U^\ell h_i^\ell + \sum_{j \in \mathcal{N}(i)} \big(V^\ell h_j^\ell \big) \Big),$$

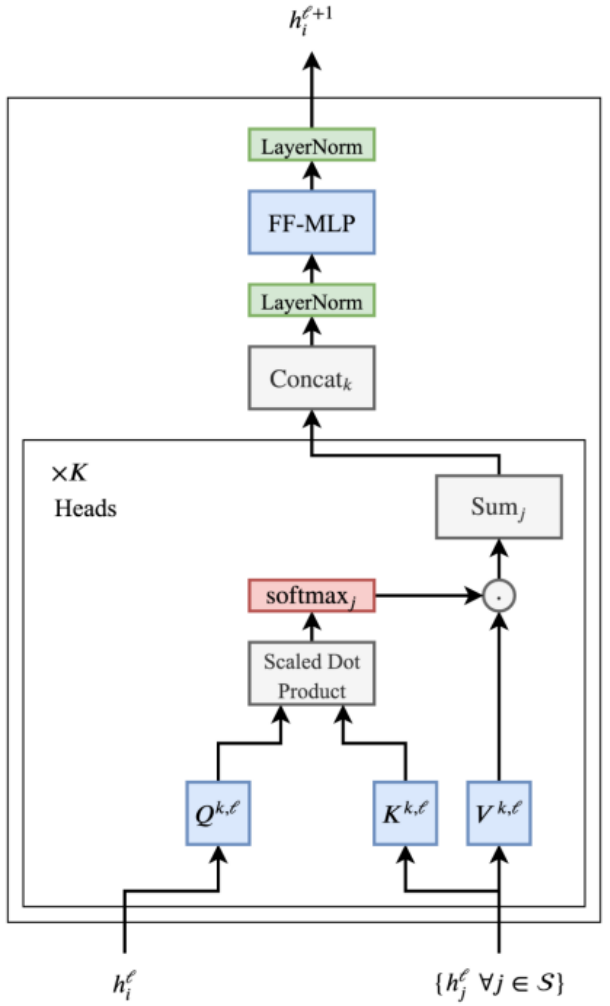
- **Transformers**:

$$i.e., \; h_i^{\ell+1} = \sum_{j \in \mathcal{S}} w_{ij} \big(V^\ell h_j^\ell \big),$$

$$\text{where } w_{ij} = \text{softmax}_j \big(Q^\ell h_i^\ell \cdot K^\ell h_j^\ell \big),$$



GNNs



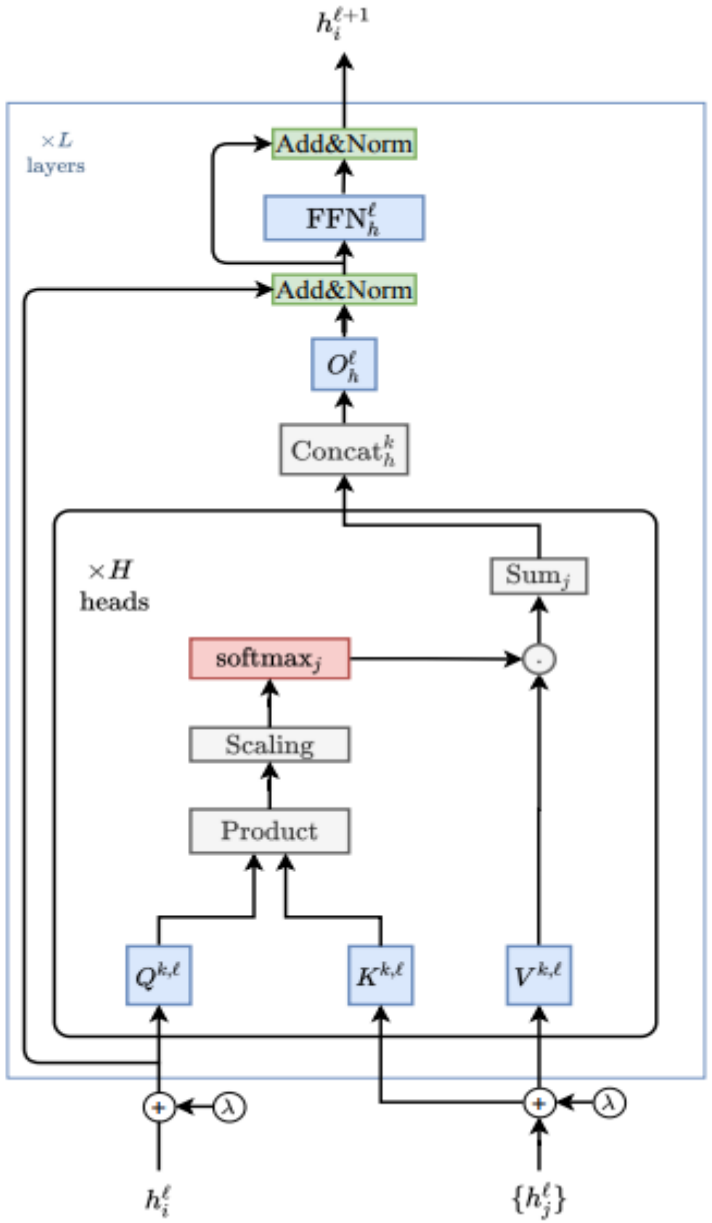
Transformer

GT (Graph Transformers *)

- Using Laplacian Eigvectors (λ) used as positional encoding (LapPE).
- Graph Transformer Layer:

$$\hat{h}_i^{\ell+1} = O_h^{\ell} \Big\|_{k=1}^{H} \Big(\sum_{j \in \mathcal{N}_i} w_{ij}^{k,\ell} V^{k,\ell} h_j^{\ell} \Big),$$

$$\text{where, } w_{ij}^{k,\ell} = \text{softmax}_j \Big(\frac{Q^{k,\ell} h_i^{\ell} \cdot K^{k,\ell} h_j^{\ell}}{\sqrt{d_k}} \Big)$$



* A Generalization of Transformer Networks to Graphs, AAAI 2021

Representative Transformer models: Graphormer

Graphormer (*)

- Centrality Encoding:

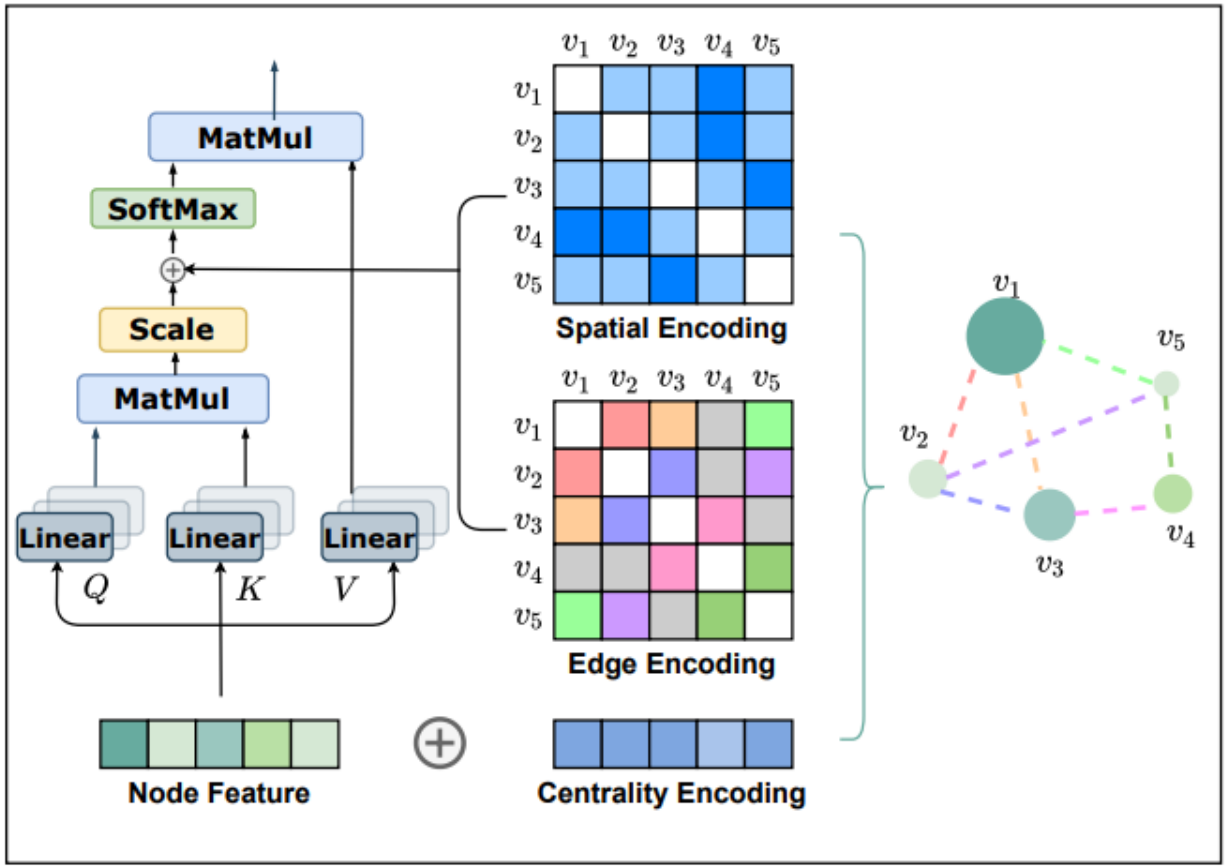
$$h_i^{(0)} = x_i + z^-_{\deg^-(v_i)} + z^+_{\deg^+(v_i)},$$

(learnable indegree z^-, and outdegree z^+)

- Self-attention bias:

$$A_{ij} = \frac{(h_i W_Q)(h_j W_K)^T}{\sqrt{d}} + b_{\phi(v_i,v_j)} + c_{ij}$$

$$\text{where } c_{ij} = \frac{1}{N}\sum_{n=1}^{N} x_{e_n}(w_n^E)^T$$

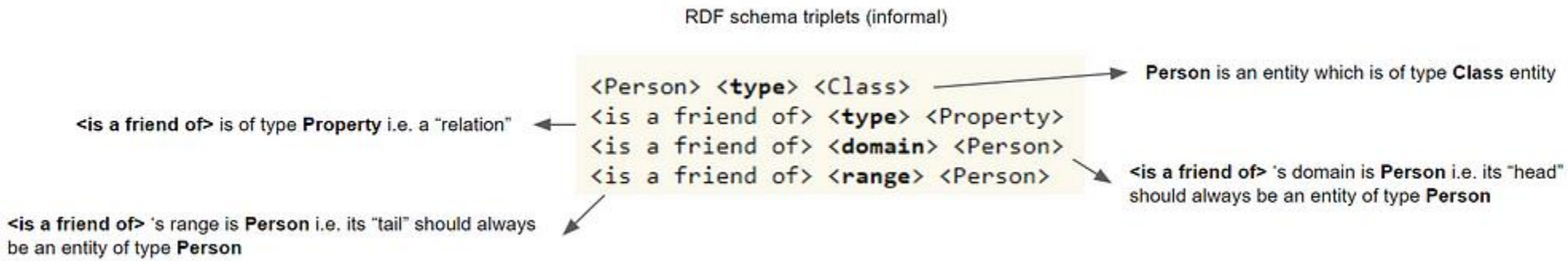


presents the path between two nodes i and j
via edge feature path: $\text{SP}_{ij} = (e_1, e_2, ..., e_N)$

$b_{\phi(v_i,v_j)}$: the distance of the shortest path (SPD) between two nodes i and j

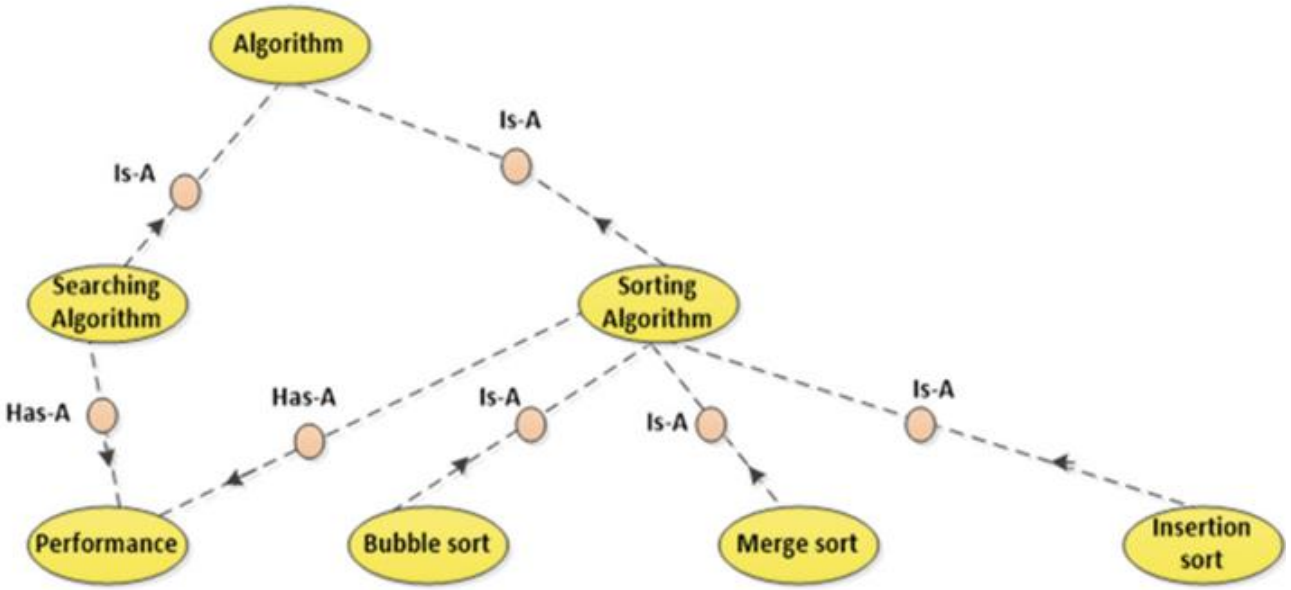
* A Do Transformers Really Perform Bad for Graph Representation? (NeurIPS 2021)

- **Knowledge graph Ontology**
  - An ontology is a model of the world (practically only a subset), listing the types of entities, the relationships that connect them, and constraints on the ways that entities and relationships can be combined.
  - Resource Description Framework (RDF) and Web Ontology Language (OWL) are some of the vocabulary frameworks used to model ontology.



https://towardsdatascience.com/a-guide-to-the-knowledge-graphs-bfb5c40272f1

- **Ontology as Foundation Layer for KG**
  - Ontology: extract taxonomic relations and attributes, plus some semantic relations.
  - Knowledge Graph focuses on extracting relationships in all forms with the same priority.



(**Domain**, Data structure)	// domain
(**Class**, Algorithm)	// c_1
(**SubClass**, Sorting algorithm, Algorithm)	//(r_1, c_2, c_1)
(**SubClass**, Searching algorithm, Algorithm)	//(r_1, c_3, c_1)
(**Has/Property**, Performance, Sorting algorithm)	//(r_2, c_4, c_2)
(**Has/Property**, Performance, Searching algorithm)	//(r_2, c_5, c_3)
(**SubClass**, Bubble sort, Sorting algorithm)	//(r_1, c_6, c_2)
(**SubClass**, Merge sort, Sorting algorithm)	//(r_1, c_7, c_2)
(**SubClass**, Insertion sort, Sorting algorithm)	//(r_1, c_8, c_2)

c_i: concept r_i: relation

Knowledge Graph Representation

- Edges in KG are represented as triples *(ℎ, r, t)*
  (head h has relation *r* with tail *t)*
- **Key Idea:**
  - Model entities and relations in the embedding/vector space R^d
  - Associate entities and relations with shallow embeddings
  - Given a true triple *(ℎ, r, t)*, the goal is that the embedding of (h, r) should be close to the embedding of t.
- **Main questions:**
  - How to embed (h, r) ?
  - How to define the similarity between them?

Translation-based Embedding Approaches

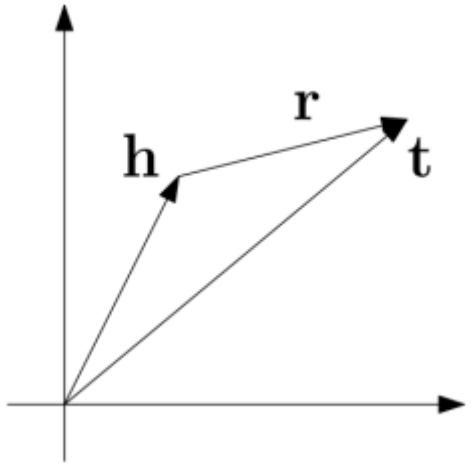
- Focused on embedding monolingual triples (h, r, t)
  - Exploit distance-based scoring functions
  - Measure the plausibility of a fact as the distance between the two entities

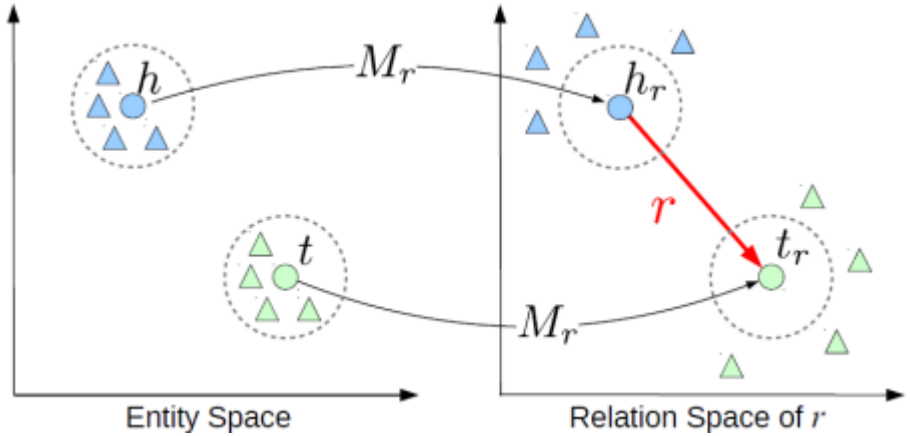
TransE: h+r ≈ t

Later approaches:

- TransH [Wang et al. 2014]
- TransR [Lin et al. 2015]
- TransD [Ji et al. 2015]
- HolE [Nickle et al. 2016]
- ComplEx [Trouillon et al. 2016]

Embedding of monolingual knowledge seems to be well-addressed.





네트워크 과학 연구실
NETWORK SCIENCE LAB

가톨릭대학교
THE CATHOLIC UNIVERSITY OF KOREA