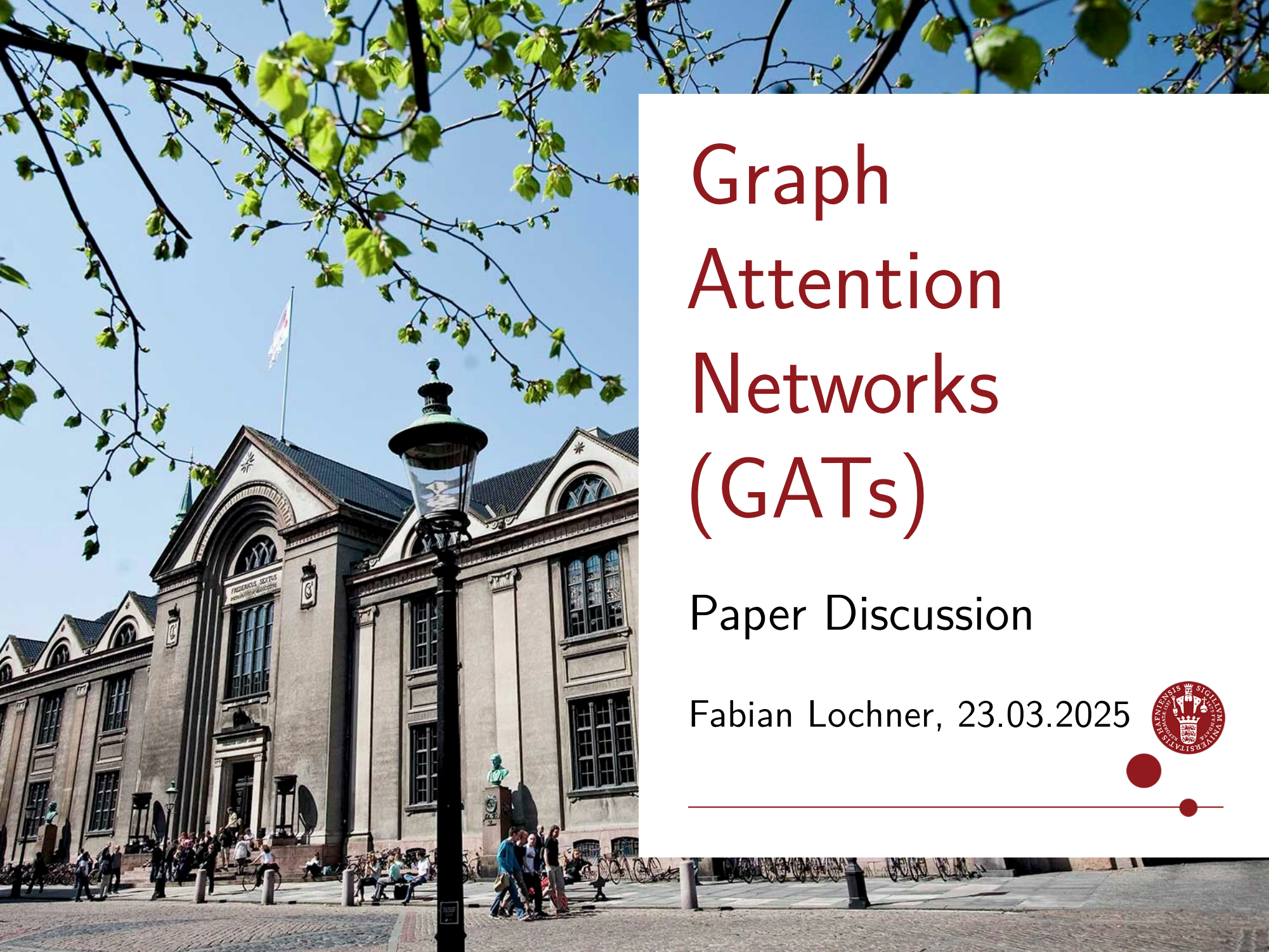


Graph Attention Networks (GATs)

Paper Discussion

Fabian Lochner, 23.03.2025

Structure of the presentation

1. Motivation for and Relevance of Graph Learning
2. Shallow vs. Deep Graph Learning
3. GNN: The MPGNN framework
4. Graph Attention Networks

1. Motivation for and Relevance of Graph Learning

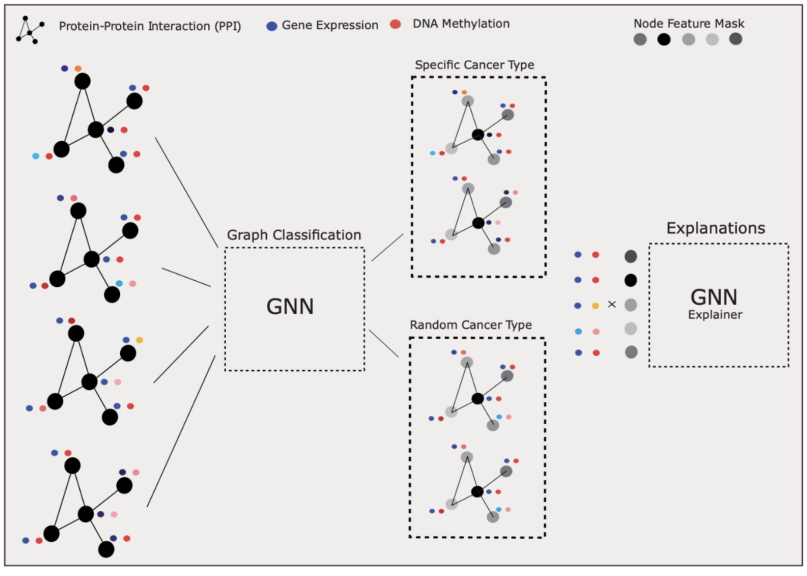


Fig. 1. Illustration of patient classification into a cancer-specific and randomized cancer group using explainable Graph Neural Networks. Each patient is represented by the topology of an protein-protein interaction network (PPI). Nodes are enriched by multi-omic features from gene expression and DNA Methylation (colored circles). The topology of each graph is the same for all patients, but the node feature values vary, reflecting the cancer-specific molecular patterns of each patient.

Cancer prediction [11]

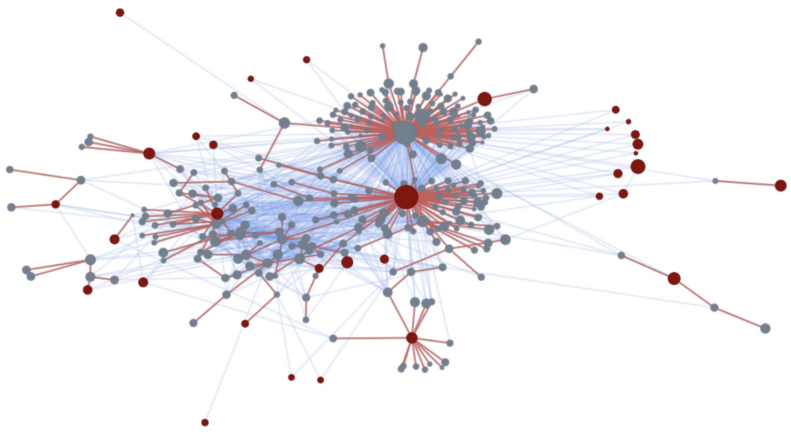


Figure 1: Example of a single news story spreading on a subset of the Twitter social network. Social connections between users are visualized as light-blue edges. A news URL is tweeted by multiple users (cascade roots denotes in red), each producing a cascade propagating over a subset of the social graph (red edges). Circle size represents the number of followers. Note that some cascades are small, containing only the root (the tweeting user) or just a few retweets.

Fake news detection on social media [10]

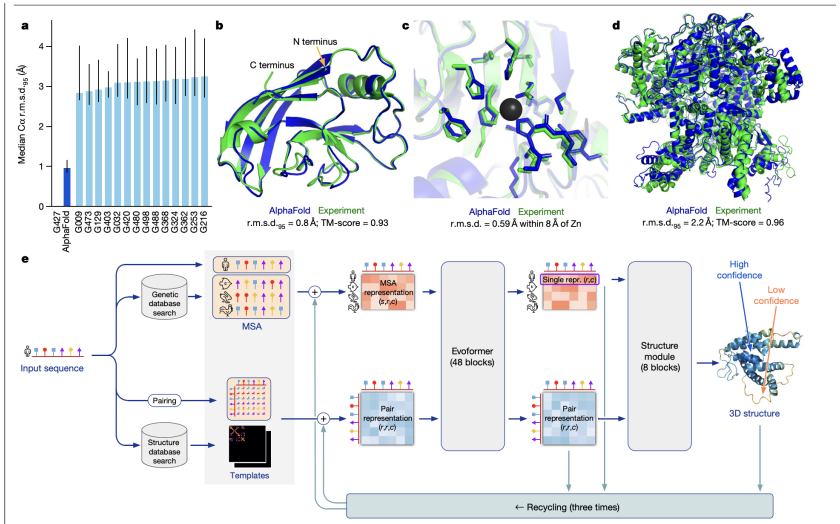
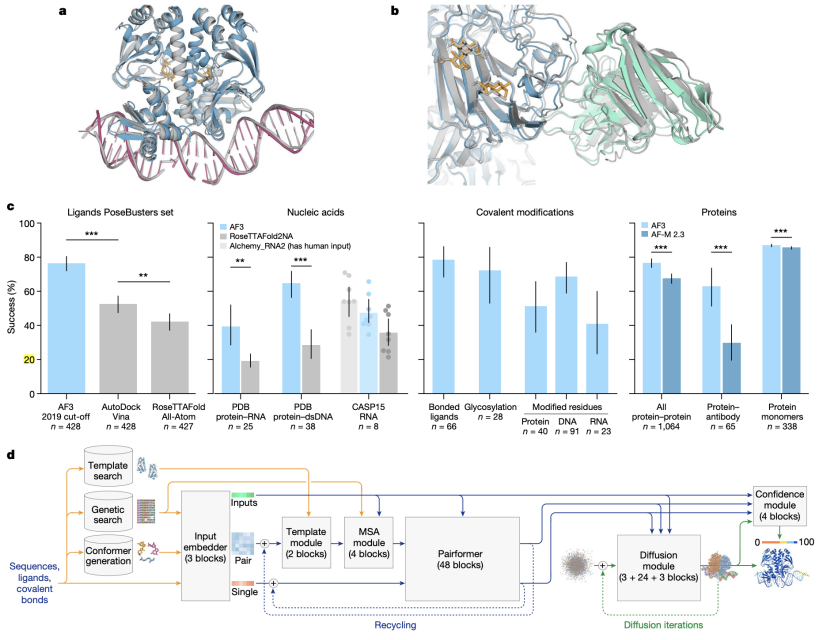
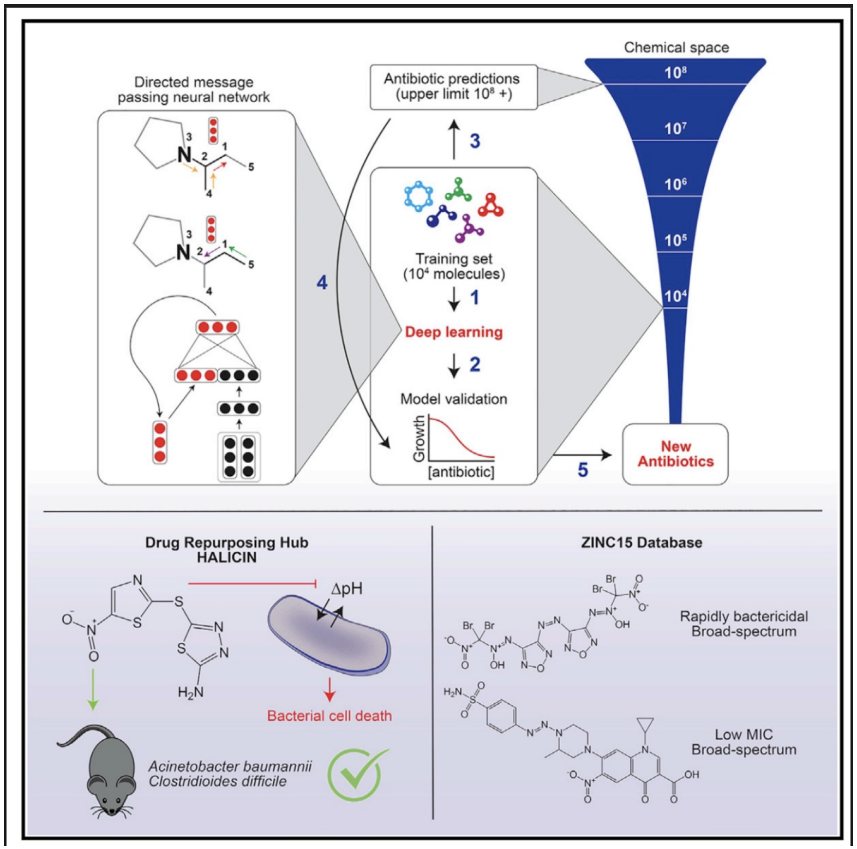


Fig. 1 | AlphaFold produces highly accurate structures. a, The performance of AlphaFold on the CASP14 dataset ($n = 87$ protein domains) relative to the top-15 entries (out of 146 entries), group numbers correspond to the numbers assigned to entrants by CASP. Data are median and the 95% confidence interval of the median, estimated from 10,000 bootstrap samples. **b**, Our prediction of CASP14 target T1049 (PDB 6Y4F, blue) compared with the true (experimental) structure (green). Four residues in the C terminus of the crystal structure are *B*-factor outliers and are not depicted. **c**, CASP14 target T1056 (PDB 6YJ1). An example of a well-predicted zinc-binding site (AlphaFold has accurate side chains even though it does not explicitly predict the zinc ion). **d**, CASP target T1044 (PDB 6VR4)—a 2,180-residue single chain—was predicted with correct domain packing (the prediction was made after CASP using AlphaFold without intervention). **e**, Model architecture. Arrows show the information flow among the various components described in this paper. Array shapes are shown in parentheses with *s*, number of sequences (N_{seq} in the main text); *r*, number of residues (N_{res} in the main text); *c*, number of channels.

Predicting 3-dimensional protein structures, Alphafold 2 [7]



Predicting 3-dimensional protein structures, Alphafold 3 [1]



Drug discovery, e.g., antibiotics [14]

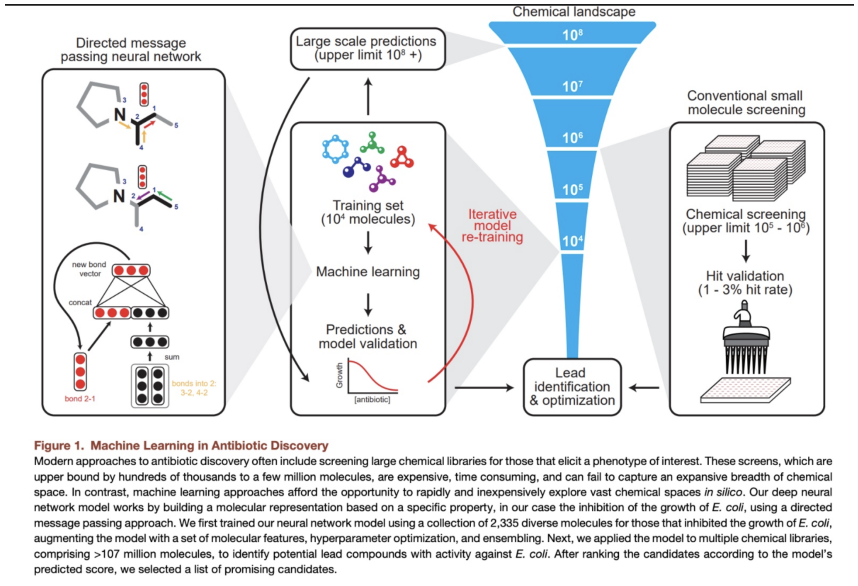


Figure 1. Machine Learning in Antibiotic Discovery
Modern approaches to antibiotic discovery often include screening large chemical libraries for those that elicit a phenotype of interest. These screens, which are upper bound by hundreds of thousands to a few million molecules, are expensive, time consuming, and can fail to capture an expansive breadth of chemical space. In contrast, machine learning approaches afford the opportunity to rapidly and inexpensively explore vast chemical spaces *in silico*. Our deep neural network model works by building a molecular representation based on a specific property, in our case the inhibition of the growth of *E. coli*, using a directed message passing approach. We first trained our neural network model using a collection of 2,335 diverse molecules for those that inhibited the growth of *E. coli*, augmenting the model with a set of molecular features, hyperparameter optimization, and ensembling. Next, we applied the model to multiple chemical libraries, comprising >107 million molecules, to identify potential lead compounds with activity against *E. coli*. After ranking the candidates according to the model's predicted score, we selected a list of promising candidates.

Drug discovery, e.g., antibiotics [14]

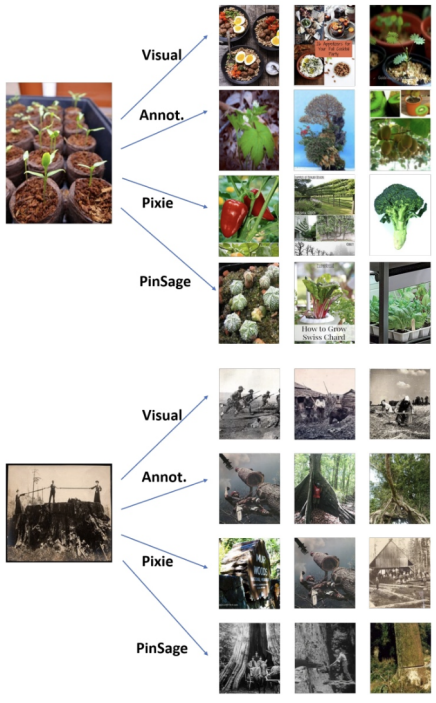


Figure 5: Examples of Pinterest pins recommended by different algorithms. The image to the left is the query pin. Recommended items to the right are computed using Visual embeddings, Annotation embeddings, graph-based Pixie, and PinSage.

Web recommendation systems, e.g., Pinterest [22]

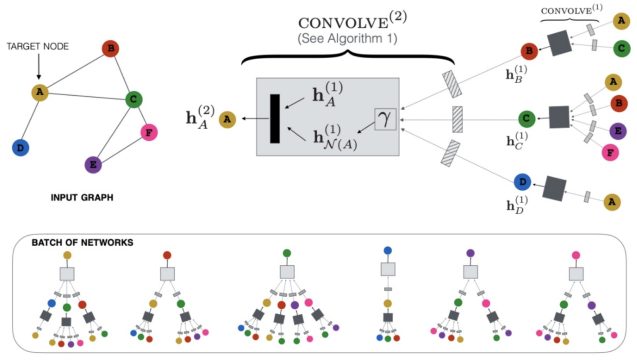


Figure 1: Overview of our model architecture using depth-2 convolutions (best viewed in color). Left: A small example input graph. Right: The 2-layer neural network that computes the embedding $\mathbf{h}_A^{(2)}$ of node A using the previous-layer representation, $\mathbf{h}_A^{(1)}$, of node A and that of its neighborhood $\mathcal{N}(A)$ (nodes B, C, D). (However, the notion of neighborhood is general and not all neighbors need to be included (Section 3.2).) Bottom: The neural networks that compute embeddings of each node of the input graph. While neural networks differ from node to node they all share the same set of parameters (*i.e.*, the parameters of the CONVOLVE$^{(1)}$ and CONVOLVE$^{(2)}$ functions; Algorithm 1). Boxes with the same shading patterns share parameters; γ denotes an importance pooling function; and thin rectangular boxes denote densely-connected multi-layer neural networks.

Web recommendation systems, e.g., Pinterest [22]

2. Shallow vs Deep Graph Learning

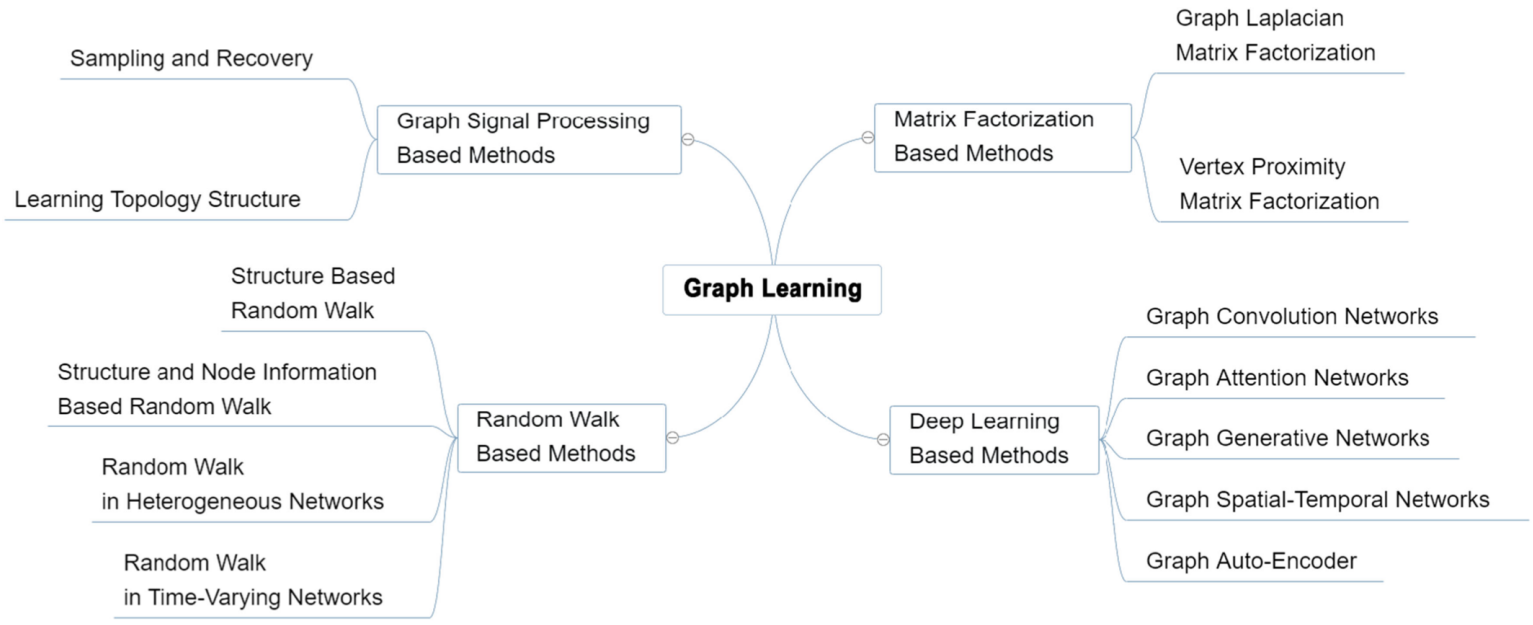


Fig. 1. Categorization of graph learning.

Overview of different graph learning methods [20]

What makes a good embedding?

Method Category	Approach	Low Dimensionality	Dense	Permutation Invariant	Faithful to Topology	Transductive/Inductive	Comments
Spectral	Laplacian + SVD	✓	✓	✓	Partial	Transductive	Requires full matrix recomputation for new nodes
Random Walk	DeepWalk	✓	✓	✓	✓	Transductive	Cannot handle unseen nodes
Random Walk	Node2Vec	✓	✓	✓	✓✓	Transductive	Must retrain for new nodes
Random Walk	LINE	✓	✓	✓	✓	Transductive	Limited to training graph
Random Walk	Struct2Vec	✓	✓	✓	✓✓	Transductive	Structural roles but still transductive
Random Walk	Metapath2Vec	✓	✓	✓	✓	Transductive	Semantic paths but requires retraining
MPGNN	GCN	✓	✓	✓	✓	Inductive*	*If using node features
MPGNN	GraphSAGE	✓	✓	✓	✓	Inductive	Designed for inductive learning
MPGNN	GAT	✓	✓	✓	✓✓	Inductive*	*If using node features
Advanced GNN	Higher-Order GNN	✓	✓	✓	✓✓	Inductive	Preserves higher-order structures
Advanced GNN	Distance Encoding	✓	✓	✓	✓✓	Inductive	Uses distance-based features
Advanced GNN	Graph Wavelet	✓	✓	✓	✓✓	Inductive	Multi-scale analysis
Deep Generative	VAE	✓	✓	✓	Partial	Both**	**Depends on implementation
Deep Generative	GAN	✓	✓	✓	Partial	Both**	**Depends on implementation
Deep Generative	Autoregressive	✓	✓	Partial	Partial	Both**	**Depends on generation process

Note:
✓ = Satisfies property
✓✓ = Particularly strong in this property
Partial = Partially satisfies or depends on implementation

- = Requires additional conditions
  ** = Can be implemented either way

Shallow Graph Learning: Node2vec algorithm [5]

- 2^{nd} Order: Transition probability: current (v) and previous node (t):

$$P(c_i = x | c_{i-1} = v) = \begin{cases} \frac{\pi_{vx}}{Z} & \text{if } (v, x) \in E \\ 0 & \text{otherwise} \end{cases}$$

- π_{vx} = unnormalized transition probability:

$$\pi_{vx} = \alpha_{pq}(t, x) \cdot w_{vx}$$

where w = edge weight between v and x

- Search bias α:

$$\alpha_{pq}(t, x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{cases}$$

where d = shortest path between t and x

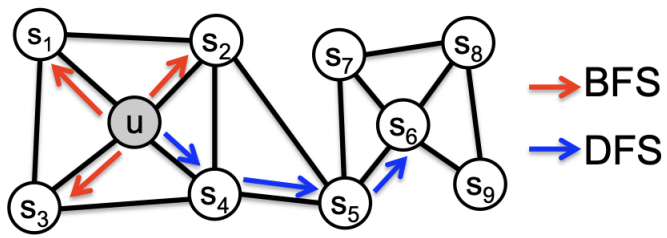


Figure 1: BFS and DFS search strategies from node u ($k = 3$).

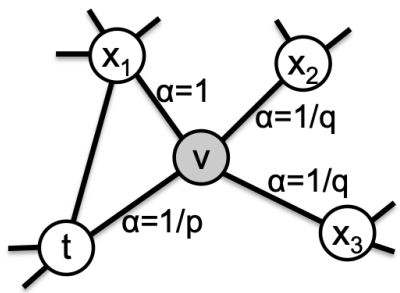


Figure 2: Illustration of the random walk procedure in *node2vec*. The walk just transitioned from t to v and is now evaluating its next step out of node v. Edge labels indicate search biases α.

Limitations of Node2vec

- Transductivity of node2vec ($\leftrightarrow$ inductive methods, e.g., **MPGNNs**)
  - Problem for e.g., *dynamic graphs*
- No inclusion of node/edge attributes for representation learning ($\leftrightarrow$ **MPGNNs**)
  - Only *structural* information is used
- Focus on *local* structure for node similarity ($\leftrightarrow$ **Struc2Vec**)
  - Far apart nodes can be similar (e.g., roles)
- No application to heterogeneous networks ($\leftrightarrow$ **Metapath2Vec**)
  - Multiple types of nodes/edges in a network

Transductive vs. inductive learning [9]

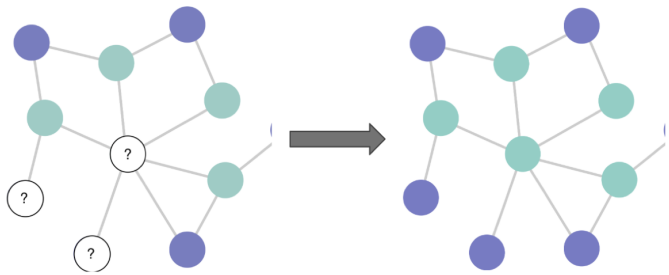
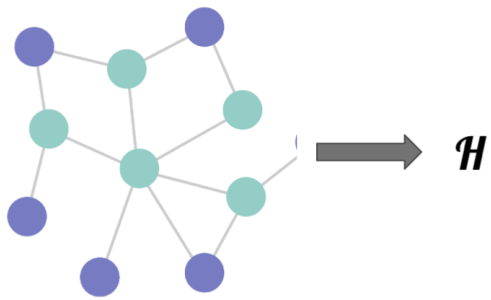
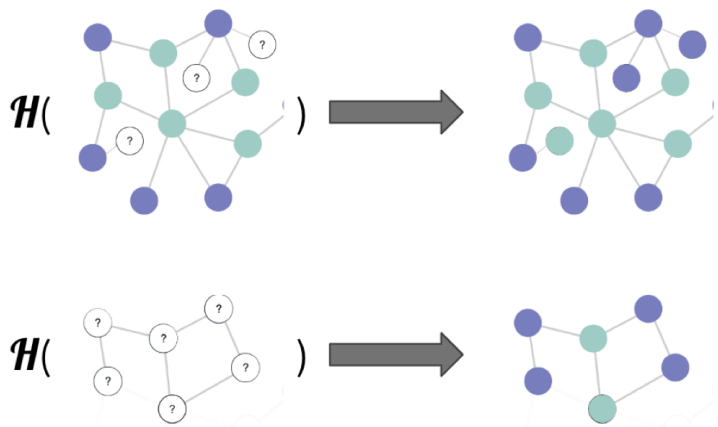


Figure 1: Node classification in transductive settings. At the training time, the learning algorithm has access to all the nodes and edges, including those nodes for which labels are to be predicted (denoted by question marks).



(a) A model $\mathcal{H}$ is learned over some graph



(b) The model is then by applied to new nodes and edges

Figure 2: Node classification in inductive settings. Once learned, the model can be applied to new unseen nodes (denoted by question marks). There may or may not exist edges between such new nodes and the nodes used for training.

3. GNN: The MPGNN framework

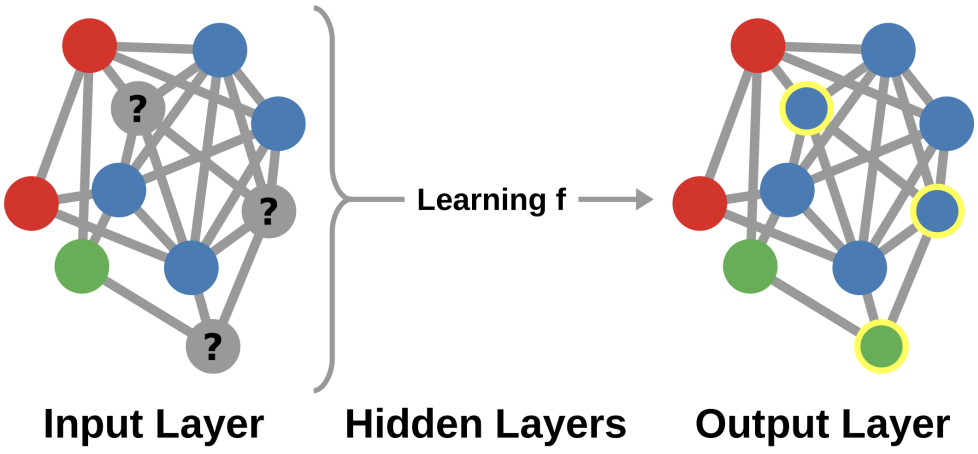
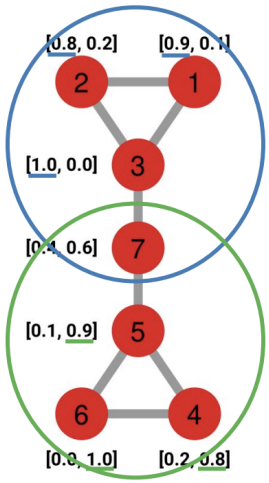


Figure 15.10: A general schema for graph convolutional learning. The gray nodes in the input network are unclassified. By learning the function f behind the classification of nodes, we can classify the rest of the network (yellow outline in the output layer). [3]

Basic Idea

- Relation between structure & node attributes
- Nearby nodes reinforce signals
- We can use this to learn

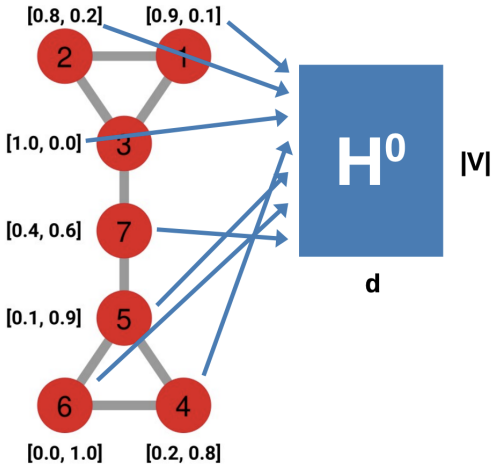


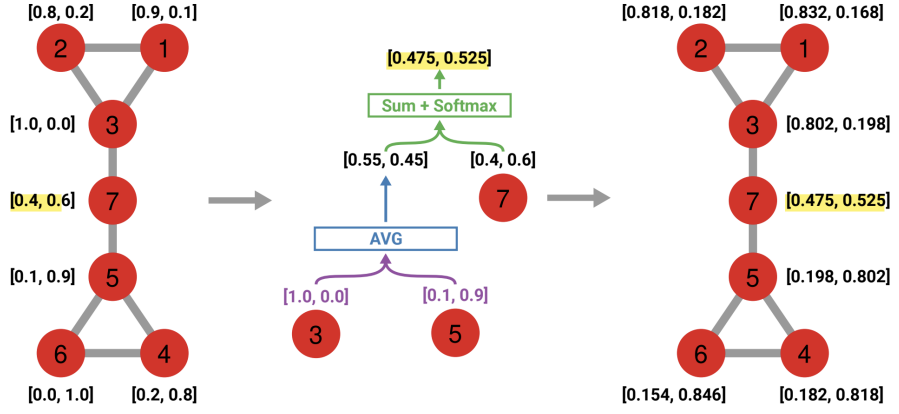
Source: Lecture Slide

The Main Process of MPGNN: Formulate, Aggregate, Update

Key Ingredient

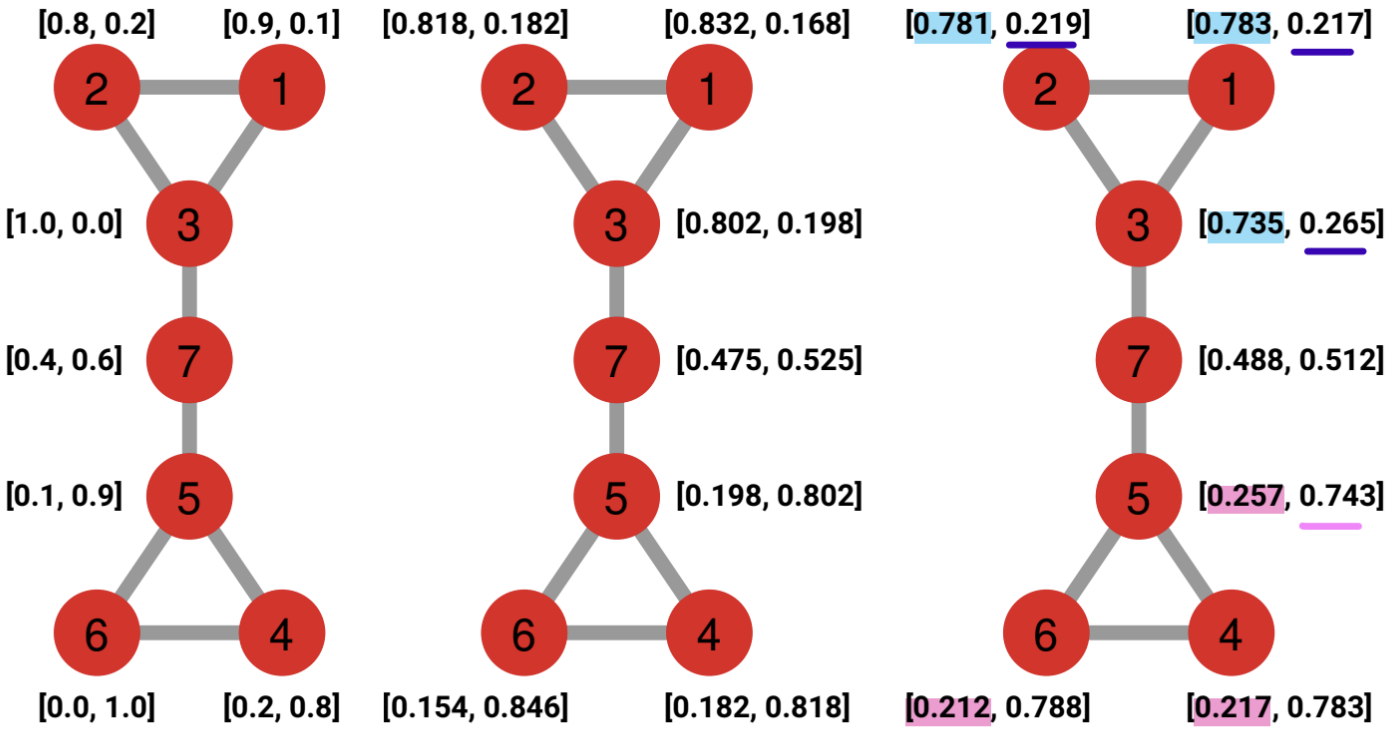
- H^0: the node feature matrix
- |V| x d matrix
- Basically, the set of node attributes





1. Formulate message
2. Aggregate messages from neighbors
3. Update own attributes/representation

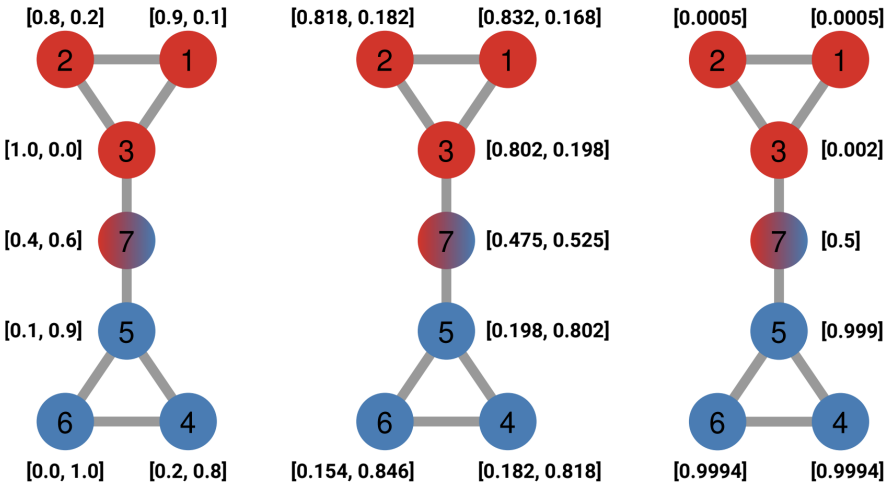
Most commonly used functions	
Formulate message	• Identity Function (no transformation) • Linear Transformations (e.g., GraphSAGE, GCN) → learnable weight matrix • Multi-Layer Perceptrons (e.g., Graph Isomorphism Networks)
Aggregate messages from neighbors	• (Row-wise) Average Aggregation • Max/Min Aggregation • Sum Aggregation • Median Aggregation • Learnable Aggregation Function (e.g., GAT)
Update own attributes/representation	• Sum + Softmax • Outer Product • GRU/LSTM Updates • Residual Connections • Skip Connections



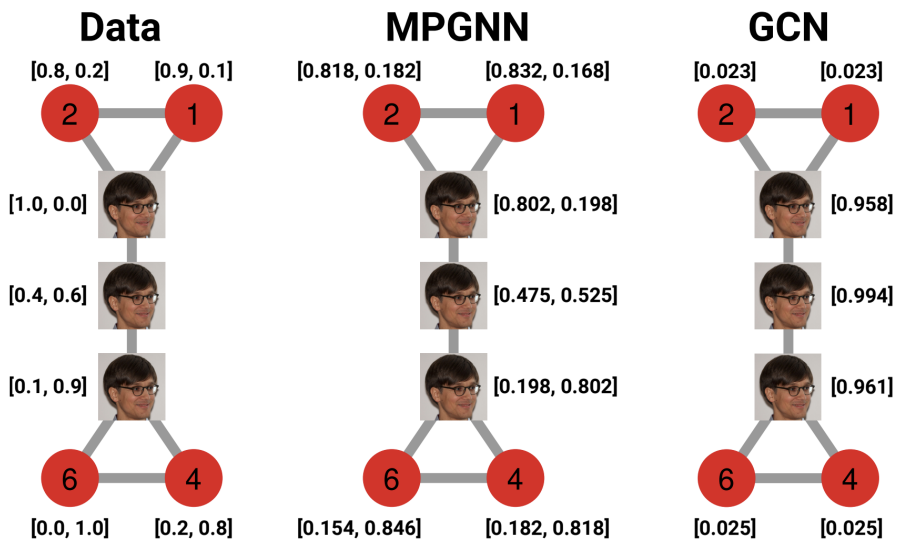
(a) H^0 (b) H^1 (c) H^2

"You can see how this can be powerful: in H2 we're starting to delineate very clearly not only the clustered structure of the network, but also which nodes are in which position in this clustered structure, whether on the border or deeply embedded in the community." [3]

MPGNN Limitation I: No learning (i.e., no weight matrix, backpropagation)

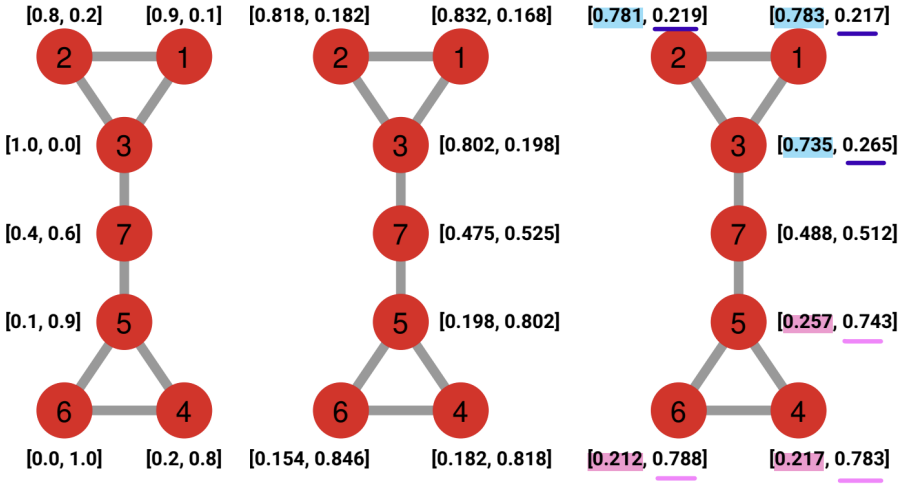


Nodes are papers connected by undirected edges if the papers cite one another. The node's color tells us the field of the paper it represents. The numbers next to the node represent the node's embeddings obtained from the last layer of the MPGNN and GCN. [3]

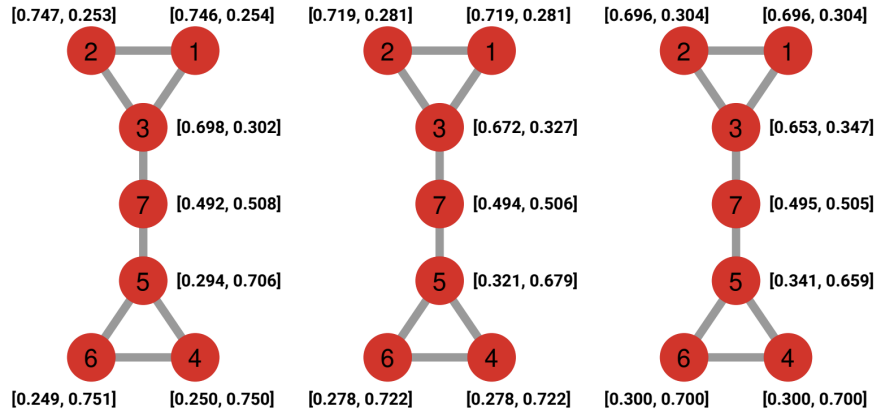


The same network as Figure 44.11, however this time we want to predict the node's picture rather than its color. The embeddings produced by the MPGNN and the GCN. [3]

MPGNN Limitation II: Oversmoothing



(a) H^0 (b) H^1 (c) H^2



(a) H^3 (b) H^4 (c) H^5

Problem: What is the optimal number of hidden layers in a GNN? This is also a problem for other spatial GNNs, e.g., GCNs and GATs.

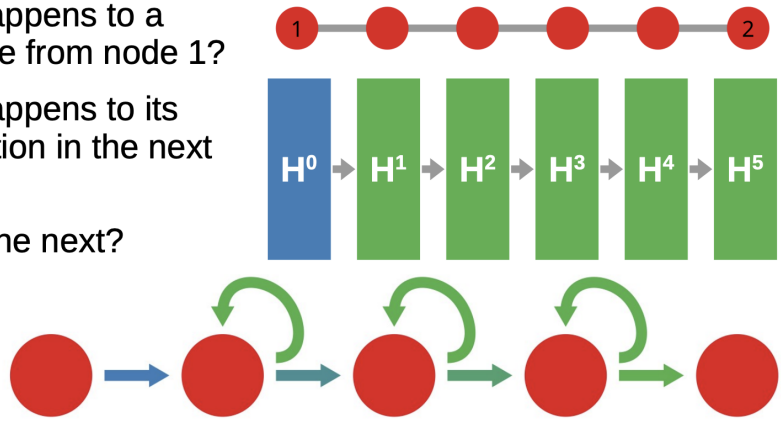
Possible solutions:

- Skip Connection Approach [12, 17] →slowing down information propagation
- Jumping Knowledge Network [21]
- Designing layers without message passing [23]

MPGNN Limitation III: Over-Squashing [15]: hubs vs. peripheral nodes

Over-Squashing

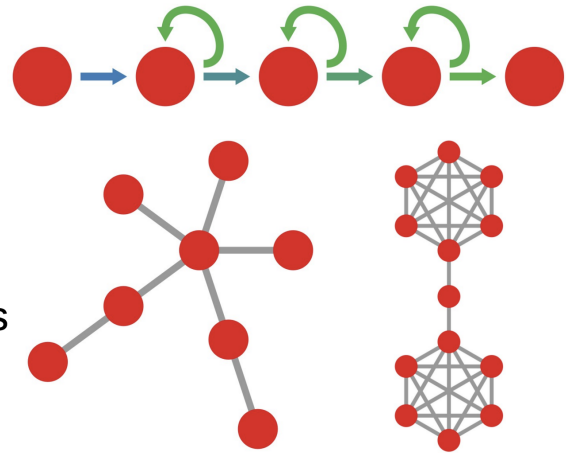
- What happens to a message from node 1?
- What happens to its information in the next layer?
- And in the next?
- And…?



Source: Lecture Slide

Over-Squashing

- Each node “overwrites” some information
- Some nodes overwrite more
- Hubs & bottlenecks



Source: Lecture Slide

4. Graph Attention Networks [18] - *Attention is all we need* [16]

As the notorious Morten Holst once said: *"Everything before 2017 is stone age."*

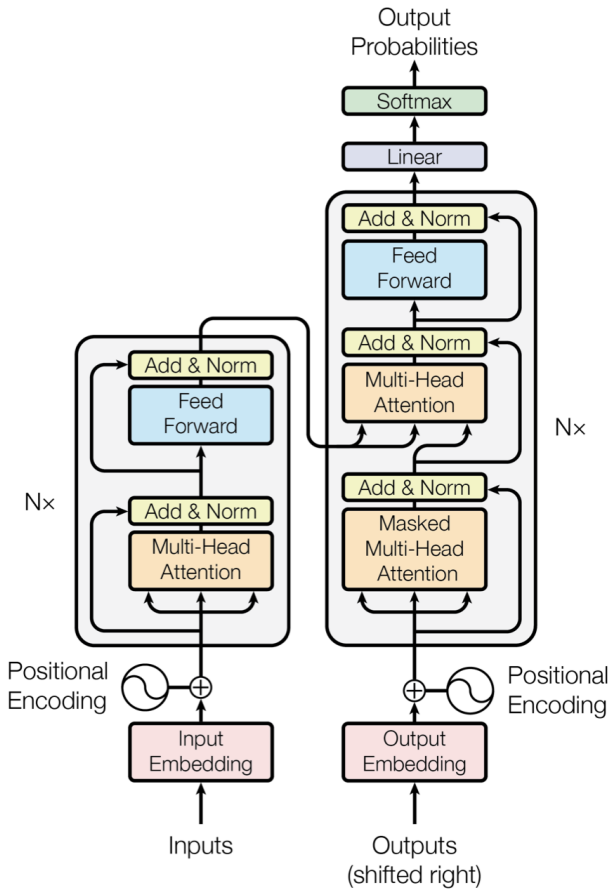
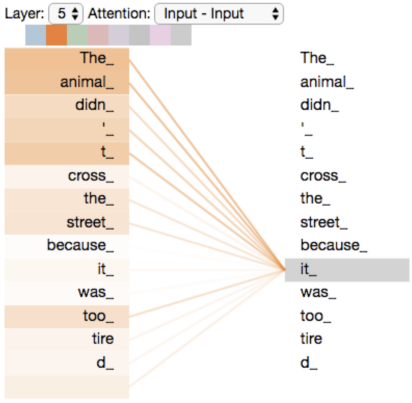


Figure 1: The Transformer - model architecture.

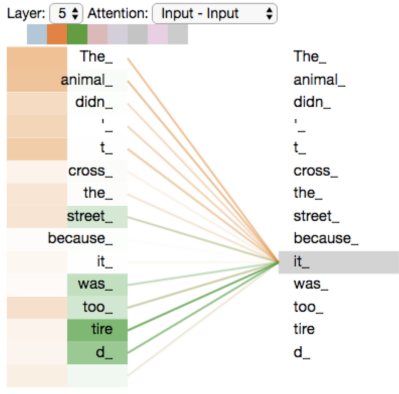
What was (self-)attention again?



As we are encoding the word "it" in encoder #5 (the top encoder in the stack), part of the attention mechanism was focusing on "The Animal", and baked a part of its representation into the encoding of "it".

Visualization of a single attention head [2]

Now that we have touched upon attention heads, let's revisit our example from before to see where the different attention heads are focusing as we encode the word "it" in our example sentence:



As we encode the word "it", one attention head is focusing most on "the animal", while another is focusing on "tired" -- in a sense, the model's representation of the word "it" bakes in some of the representation of both "animal" and "tired".

Visualization of multiple attention heads [2]

Scaled Dot-Product Attention

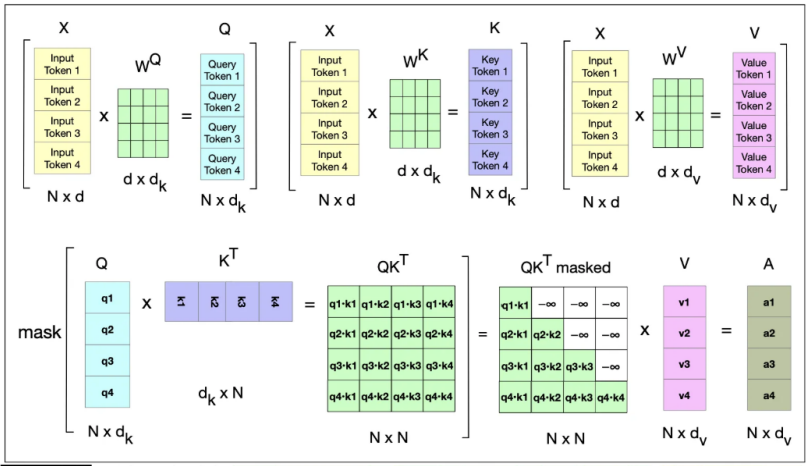


Figure 9.10 Schematic of the attention computation for a single attention head in parallel. The first row shows the computation of the **Q**, **K**, and **V** matrices. The second row shows the computation of $\mathbf{QK}^T$, the masking (the softmax computation and the normalizing by dimensionality are not shown) and then the weighted sum of the value vectors to get the final attention vectors.

Attention computation for a single attention head in parallel [8]

Here's a final set of equations for computing self-attention for a single self-attention output vector $\mathbf{a}_i$ from a single input vector $\mathbf{x}_i$. This version of attention computes $\mathbf{a}_i$ by summing the *values* of the prior elements, each weighted by the similarity of its *key* to the *query* from the current element:

$$\mathbf{q}_i = \mathbf{x}_i\mathbf{W}^{\mathbf{Q}}; \quad \mathbf{k}_j = \mathbf{x}_j\mathbf{W}^{\mathbf{K}}; \quad \mathbf{v}_j = \mathbf{x}_j\mathbf{W}^{\mathbf{V}} \tag{9.10}$$

$$\text{score}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{q}_i \cdot \mathbf{k}_j}{\sqrt{d_k}} \tag{9.11}$$

$$\alpha_{ij} = \text{softmax}(\text{score}(\mathbf{x}_i, \mathbf{x}_j)) \;\; \forall j \leq i \tag{9.12}$$

$$\mathbf{a}_i = \sum_{j \leq i} \alpha_{ij}\mathbf{v}_j \tag{9.13}$$

Computation of self-attention for a single self-attention output vector a_i from a single input vector x_i [8]

Multi-Head Attention [8]

Actual Attention: slightly more complicated

- Instead of one attention head, we'll have lots of them!
- Intuition: each head might be attending to the context for different purposes
  - Different linguistic relationships or patterns in the context

$$\mathbf{q}_i^c = \mathbf{x}_i \mathbf{W}^{\mathbf{Qc}}; \quad \mathbf{k}_j^c = \mathbf{x}_j \mathbf{W}^{\mathbf{Kc}}; \quad \mathbf{v}_j^c = \mathbf{x}_j \mathbf{W}^{\mathbf{Vc}}; \quad \forall c \quad 1 \leq c \leq h$$

$$\text{score}^c(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{q}_i^c \cdot \mathbf{k}_j^c}{\sqrt{d_k}}$$

$$\alpha_{ij}^c = \text{softmax}(\text{score}^c(\mathbf{x}_i, \mathbf{x}_j)) \quad \forall j \leq i$$

$$\mathbf{head}_i^c = \sum_{j \leq i} \alpha_{ij}^c \mathbf{v}_j^c$$

$$\mathbf{a}_i = (\mathbf{head}^1 \oplus \mathbf{head}^2 \ldots \oplus \mathbf{head}^h)\mathbf{W}^O$$

$$\text{MultiHeadAttention}(\mathbf{x}_i, [\mathbf{x}_1, \cdots, \mathbf{x}_N]) = \mathbf{a}_i$$

Benefit of GAT over Graph Convolutional Network (GCN)

GCN

$$H^{l} = \sigma\left(H^{l-1}\,\hat{D}^{-\frac{1}{2}}\,(I+A)\,\hat{D}^{-\frac{1}{2}}\,W^{l}\right)$$

Problem: Fixed attention weights where $\hat{D}^{-\frac{1}{2}}(I+A)\hat{D}^{-\frac{1}{2}} = \frac{1}{\sqrt{k_v k_u}}$

GAT

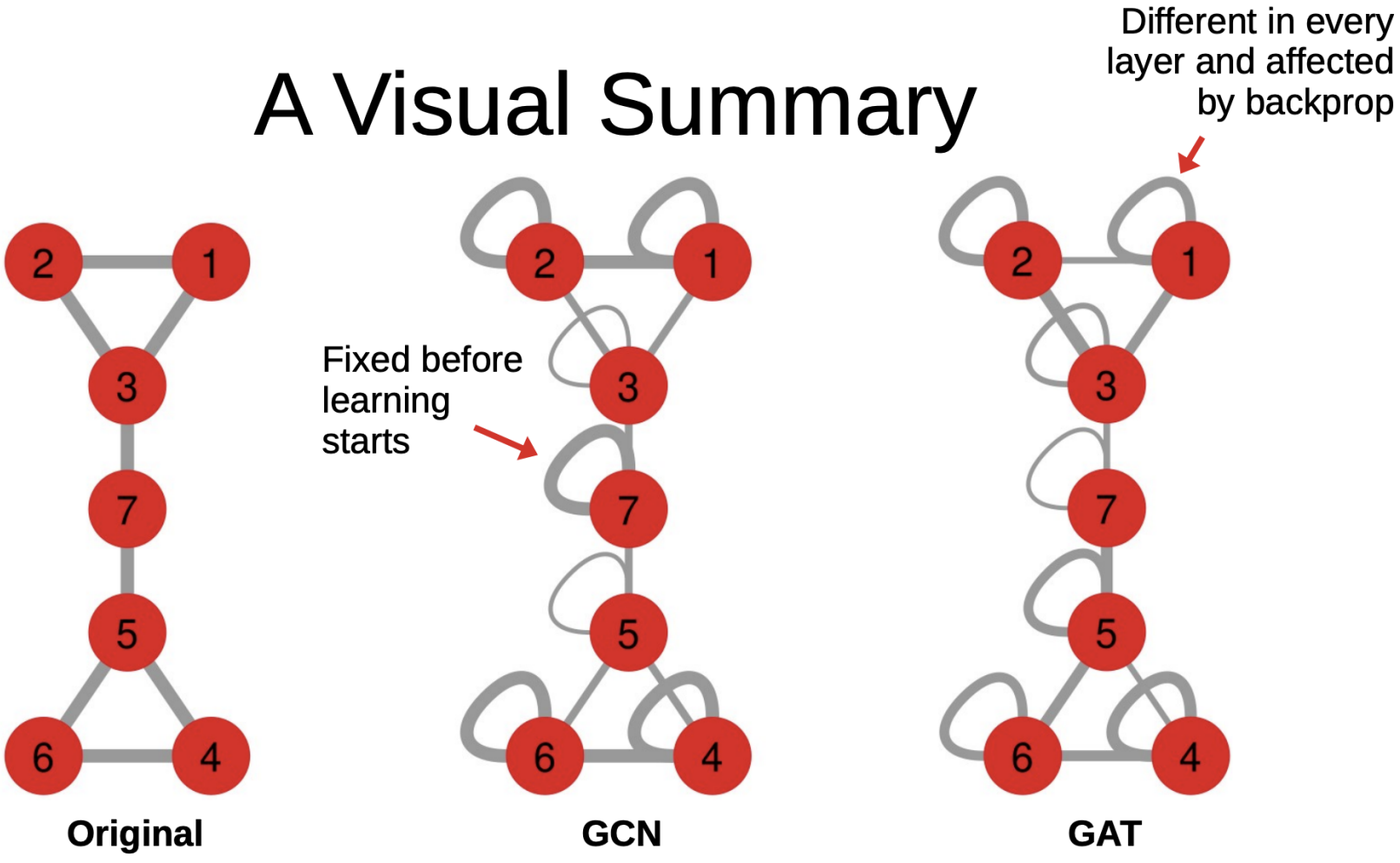
$$H^{l} = \sigma\left(H^{l-1}\,\alpha^{l}\,W^{l}\right)$$

Solution: GATs try to learn a different attention value for each of v's neighbors with backpropagation.

Where:

- σ is the activation function.
- $\hat{D}^{-\frac{1}{2}}(I+A)\hat{D}^{-\frac{1}{2}}$ is the symmetric normalization.
- W^{l} is the learnable weight matrix at layer l.
- α^{l} is a learnable function at layer l.

Benefit of GAT over GCN



Graph Attention - Visually I [18]

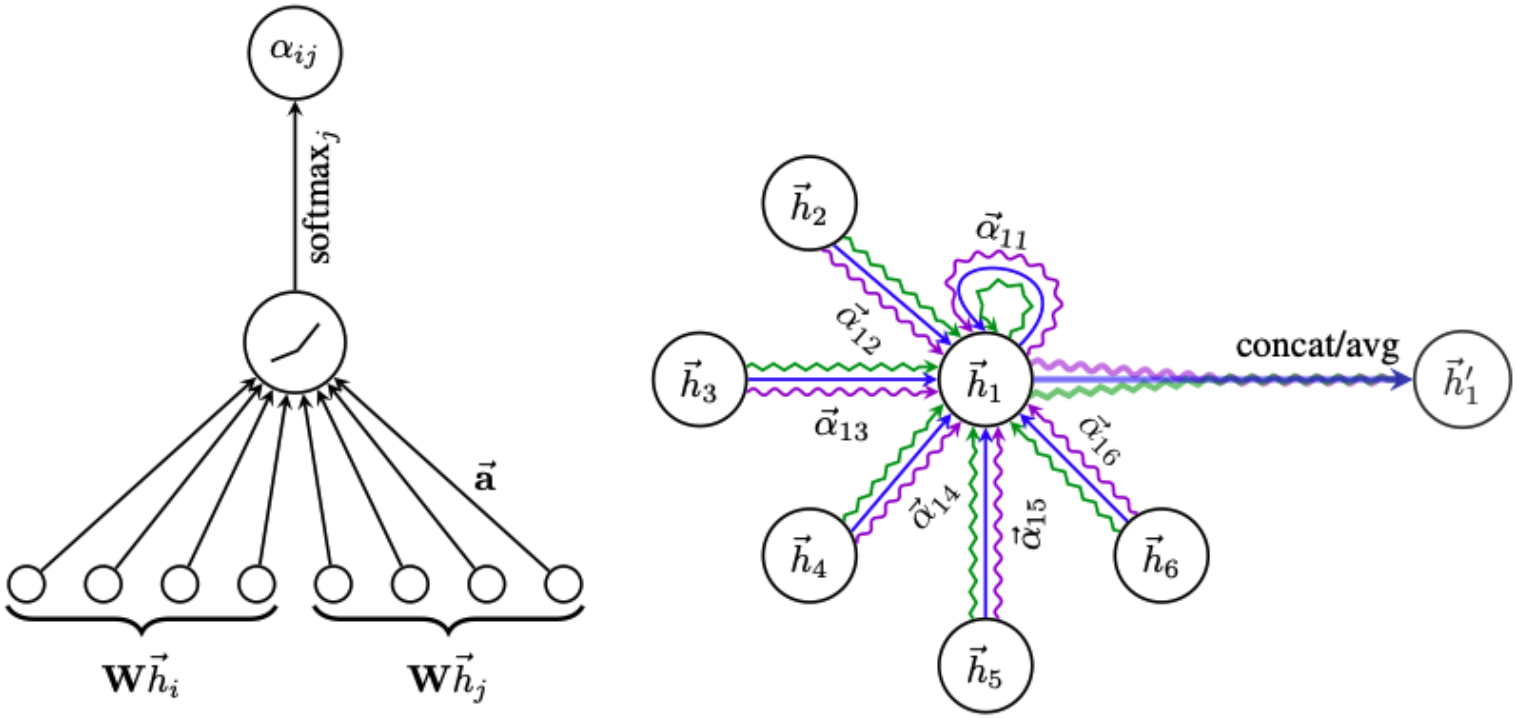
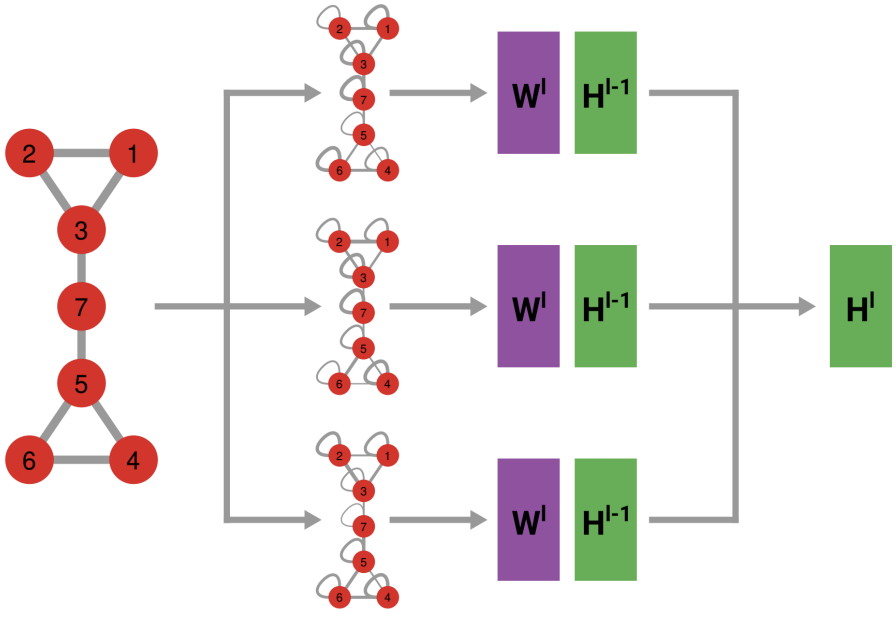


Figure 1: **Left:** The attention mechanism $a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j)$ employed by our model, parametrized by a weight vector $\vec{\mathbf{a}} \in \mathbb{R}^{2F'}$, applying a LeakyReLU activation. **Right:** An illustration of multi-head attention (with $K = 3$ heads) by node 1 on its neighborhood. Different arrow styles and colors denote independent attention computations. The aggregated features from each head are concatenated or averaged to obtain $\vec{h}'_1$.

Graph Attention - Visually II [3]



- What happens in the layer l of a GAT/Graph Transformer Network
- See the different edge thickness in each layer?
- The benefit of multiple attention heads: Different heads can learn about different network patterns/features: e.g., communities, roles, hierarchies.
- Similar to capturing different linguistic patterns in NLP.

A single graph attention layer - Mathematically [18]

Input

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

Output

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

Where:
N = number of nodes
F = number of features per node

Step 1: Applying learnable linear transformation with *weight matrix* $\mathbf{W} \in \mathbb{R}^{F' \times F}$ to every node

GAT: Attention Mechanism - Step 2

Step 2: Performing self-attention by computing the attention coefficients

$$e_{ij} = a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j) \tag{1}$$

where:

- $a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \rightarrow \mathbb{R}$
- e_{ij} indicates the *importance* of node j's features to node i
- *masked attention:* only computing self-attention for the first-order neighbors of i

GAT: Normalizing Attention - Step 3 & 4

Step 3: Normalize attention coefficients with softmax

$$\alpha_{ij} = \text{softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})} \tag{2}$$

Step 4: Applying activation function

LeakyReLU nonlinearity (with negative input slope $\alpha = 0.2$)

GAT: Full Attention Computation - Step 5

Step 5: Full computation of the attention coefficients

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{a}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\text{LeakyReLU}\left(\vec{a}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k]\right)\right)} \tag{3}$$

where:

- $\cdot^T$ represents transposition and $\|$ is the concatenation operation
- the attention mechanism a is a single-layer feedforward neural network

GAT: Computing Final Node Features - Step 6

Step 6: Using normalized attention coefficients to compute final output features

$$\vec{h}'_i = \sigma\left(\sum_{j\in\mathcal{N}_i} \alpha_{ij}\mathbf{W}\vec{h}_j\right) \tag{4}$$

where:

- σ is a nonlinearity (activation function)
- The output combines neighbors' features weighted by attention coefficients

GAT: Multi-Head Attention - Step 7

Step 7: Applying multi-head attention with K independent attention mechanisms

$$\vec{h}_i' = \Big\Vert_{k=1}^{K} \sigma\left(\sum_{j\in\mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j\right) \tag{5}$$

where:

- $\|$ represents concatenation
- α_{ij}^k are normalized attention coefficients computed by the k-th attention mechanism (a^k)
- $\mathbf{W}^k$ is the corresponding input linear transformation's weight matrix

GAT: Final Layer Configuration - Step 8

Step 8: Using *averaging* instead of *concatenation* for the final layer

$$\vec{h}'_i = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}_i}\alpha_{ij}^{k}\mathbf{W}^{k}\vec{h}_j\right) \tag{6}$$

- Final nonlinearity (softmax or sigmoid) is delayed until this step
- Useful for classification problems
- Averages the outputs from all attention heads

Expanding the Applicability of GATs

- GATs on heterogenous graphs (several types of nodes and edges) $\rightarrow$*Heterogeneous Graph Attention Networks* [19]
- GATs on signed graphs (positive and negative edges) $\rightarrow$*Signed Graph Attention Networks* [6]
- *Spatial-Temporal Graph Attention Networks* for traffic predictions [25]

Limitations of GATs

1. Only local, first-order attention $\leftrightarrow$ global attention
2. Rather simple attention mechanism $\leftrightarrow$ *QKV* attention mechanism
3. No positional encoding of nodes

Solution: *Graph Transformer Networks*

Finally: Graph Transformer Networks

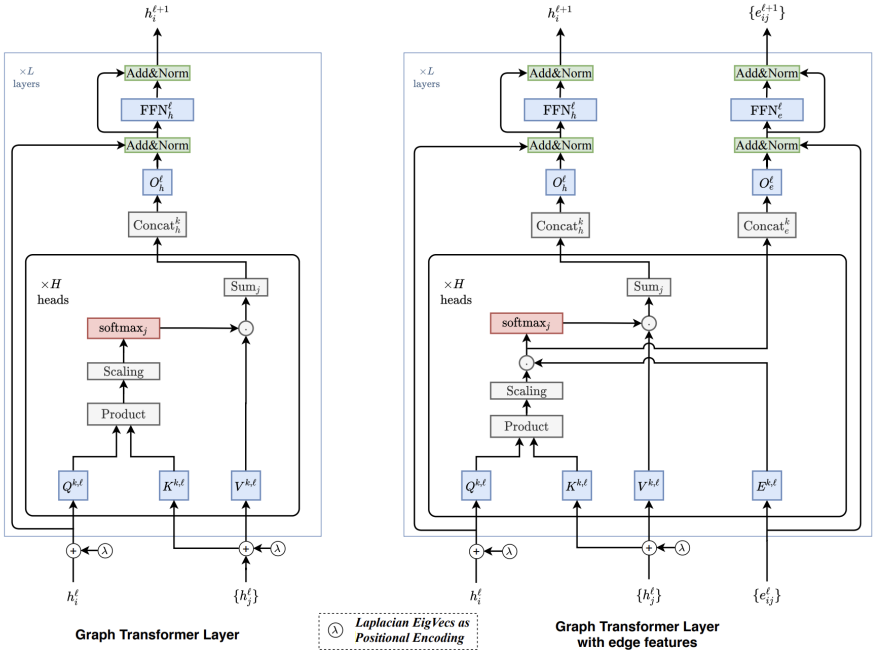


Figure 1: Block Diagram of Graph Transformer with Laplacian Eigvectors (λ) used as positional encoding (LapPE). LapPE is added to input node embeddings before passing the features to the first layer. ***Left***: Graph Transformer operating on node embeddings only to compute attention scores; ***Right***: Graph Transformer with edge features with designated feature pipeline to maintain layer wise edge representations. In this extension, the available edge attributes in a graph is used to explicitly modify the corresponding pairwise attention scores.

A Generalization of Transformer Networks to Graphs [4]

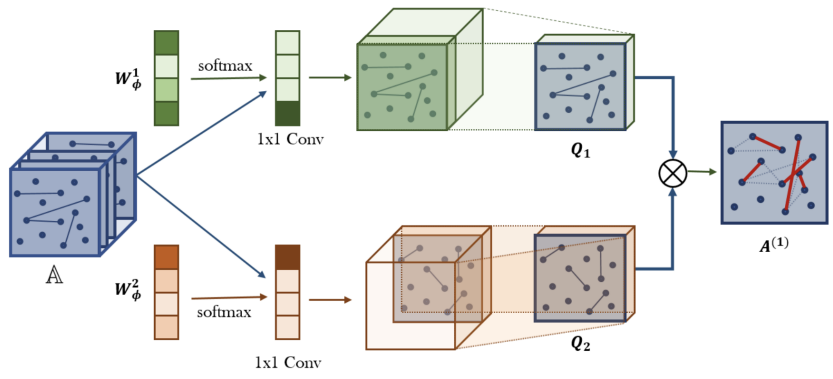


Figure 1: **Graph Transformer Layer** softly selects adjacency matrices (edge types) from the set of adjacency matrices $\mathbb{A}$ of a heterogeneous graph G and learns a new meta-path graph represented by $A^{(1)}$ via the matrix multiplication of two selected adjacency matrices Q_1 and Q_2. The soft adjacency matrix selection is a weighted sum of candidate adjacency matrices obtained by 1×1 convolution with non-negative weights from softmax(W_ϕ^1).

The original Graph Transformer Networks paper applied on heterogeneous graphs [24]

BILLIONS and BILLIONS of Graph Transformer models... [13]: Shallow and Deep Graph Transformers

TABLE II
SHALLOW GRAPH TRANSFORMERS

Model	Graph inductive bias			Graph attention mechanisms		Application	Dataset	Code availability
	Node Positional encoding	Edge Structural Encoding	Message-Passing Bias	Global attention	Local Attention			
Graph Transformer [16]	Yes (local)	Yes (local)	Yes (pre-processing)	No	Yes (message passing)	Graph representation learning	ZINC, PATTERN, CLUSTER	Yes
GAT [42]	No	No	No	Yes (linear)	Yes (message passing)	Node classification of graph-structured data	Cora, Citeseer, Pubmed, PPI	Yes
GATv2 [95]	No	No	No	Yes (linear)	Yes (message passing)	Graph representation learning	OGB, QM9, VARMISUSE	Yes
GTN [36]	No	No	No	No	Yes (message passing)	Node classification on heterogeneous graphs	DBLP, ACM, IMDB	Yes
GATE [63]	No	No	No	Yes (linear)	No	Cross-lingual relation and event extraction	ACE 2005 corpus	Yes
PiT [96]	No	No	No	Yes (quadratic)	No	Image classification, object detection	ImageNet, COCO, ImageNet-A	Yes
WDPruning [97]	No	No	No	Yes (quadratic)	No	Image classification and image retrieval	CIFAR-10 and ILSVRC-12	No
SVT [98]	No	No	No	Yes (quadratic)	No	Image classification	Synthetic data and CIFAR-10	No
Gubelmann et al. [99]	No	No	No	Yes	No	Natural language inference (NLI)	SNLI and MNLI	No
TADDY [100]	Yes (global and local)	No	No	Yes (linear)	No	Anomaly detection in dynamic graphs	UCI Messages, Enron, DBLP, Facebook, Twitter and Reddit.	No
Tree Transformer [38]	Yes (local)	No	No	Yes (quadratic)	No	Program translation and semantic parsing	For2Lam, CoffeeScript-JavaScript, JOBS, GEO, ATIS	Yes
GAT-POS [43]	Yes (local)	No	No	Yes (linear)	Yes (message passing)	Node classification	Chameleon, Squirrel, Actor, Cora, Citeseer, Pubmed	No
DGT [45]	Yes	No	No	Yes (linear)	No	Node classification on graphs	Actor, Squirrel, Chameleon, Cora, Citeseer, twitch-gamers, ogbn-arxiv, Reddit	No
EDTER [54]	No	No	No	Yes (linear)	Yes (message passing)	Edge detection	BSDS500, NYUDv2, Multicue	Yes
BGT-Net [55]	No	No	No	Yes	No	Scene Graph Generation for Images	Visual Genome (VG), OpenImages (OI), Visual Relationship Detection (VRD)	No
PMPNet [61]	No	No	Yes (pre-processing)	No	Yes (message passing)	3D object detection	nuScenes	Yes
GAMLP [74]	No	No	No	Yes (linear)	Yes (spectral)	Node classification	14 real-world graph datasets	Yes
SA-GAT [101]	No	No	No	Yes (linear)	Yes (spectral)	Graph classification	MUTAG, PTC, NCI1, NCI109, PROTEINS, ENZYMES, DD and COLLAB	No
GKAT [76]	No	No	No	Yes (linear)	No	Graph classification and graph pattern detection	Bioinformatics and social networks datasets, ImageNet dataset	No
EcoFormer [78]	No	No	No	Yes (linear)	No	Image classification and long-range tasks	CIFAR-100, ImageNet-1K, Long Range Arena	Yes
Reformer [84]	No	No	No	Yes (linear)	Yes (message passing)	NLP and generative tasks	Synthetic task, enwik8, imagenet64, WMT 2014 English-German	Yes
LISA [85]	No	No	No	Yes (linear)	No	Sequential recommendation	Alibaba, Amazon Video Games, MovieLens 1M, MovieLens 25M	Yes

TABLE III
DEEP GRAPH TRANSFORMERS

Model	Graph inductive bias			Graph attention mechanisms		Application	Datasets	Code availability
	Node Positional encoding	Edge Structural Encoding	Message-Passing Bias	Global attention	Local Attention			
HG-Transformer [102]	No	No	No	Yes (linear)	No	Multi-label text classification	RCV1, RCV1-2K, AmazonCat-13K	No
Unfolded Transformer [103]	No	No	No	Yes (quadratic)	No	Graph neural networks and NLP	IMDB and SST2	Yes
DeepGraph [94]	Yes (relative)	No	No	Yes (linear)	Yes (message passing)	Graph representation learning	PCQM4M-LSC, ZINC, CLUSTER, PATTERN	Yes
GraphWriter [105]	No	Yes	No	Yes (linear)	No	Text generation from knowledge graphs	AGENDA dataset	Yes
GraphLoG [106]	No	No	Yes (pre-processing)	No	Yes (message passing)	Graph representation learning	ZINC15, MoleculeNet, Protein ego-networks	Yes
DIET [40]	Yes	No	No	Yes (quadratic)	No	cross-lingual generalization, machine translation	GLUE, XTREME, WMT 2018	No
GRPE [41]	Yes (local and global)	Yes (local and global)	No	Yes (linear)	No	Graph representation learning	ZINC, MolPCBA, MolHIV, PATTERN, CLUSTER, PCQM4M, PCQM4Mv2	No
Graphormer [11]	No	Yes	No	Yes (quadratic)	No	Graph representation learning	PCQM4M-LSC, OGB, ZINC	Yes
Relation-aware Self-Attention [26]	No	Yes (local)	No	Yes (quadratic)	No	Machine translation	WMT 2014 English-German and English-French	Yes
GraphiT [46]	Yes (local and global)	No	No	Yes (quadratic)	Yes (message passing)	Graph representation learning for classification and regression tasks	MUTAG, PROTEINS, PTC, NCI1, ZINC	Yes
Tree Transformer [48]	Yes (global and local)	No	No	Yes (quadratic)	Yes (message passing)	Code representation learning	Python150k and JavaScript150k	Yes
MagLapNet [49]	Yes (local and global)	No	No	Yes (quadratic)	Yes (spectral)	Graph learning tasks such as correctness testing of sorting networks and source code understanding	Synthetic graphs, OGB Code2	Yes
SAT [52]	Yes (local and global)	No	No	Yes (quadratic)	Yes (message passing)	Graph representation learning for graph and node property prediction	ZINC, CLUSTER, PATTERN, OGBG-PPA, OGBG-CODE2	Yes
Edge Transformer [107]	Yes	Yes	No	Yes (quadratic)	No	Improve systematic generalization in natural language understanding and relational reasoning tasks	CLUTRR, CFQ and COGS	Yes
Cai et al. [62]	Yes	Yes	No	Yes (quadratic)	No	Graph-to-sequence learning	LDC2015E86, LDC2017T10, WMT16 English-German, WMT16 English-Czech	Yes
SAN [66]	Yes (local and global)	No	No	Yes (quadratic)	Yes (spectral)	Graph representation learning	ZINC, PATTERN, CLUSTER, MolHIV, MolPCBA	Yes
KMCA [79]	No	No	No	Yes (quadratic)	Yes (message passing)	Time series analysis	Synthetic, FordA, uWave, IMDB, ECG	No
ABT-MPNN [87]	No	No	Yes (pre-processing)	Yes (quadratic and linear)	No	Molecular property prediction and biological activity prediction	Johnson et al. [108], Tox21, ClinTox, ToxCast, HIV, QM8, ESOL, FreeSolv, Lipophilicity and QM8.	Yes
UGformer [93]	No	No	No	Yes (quadratic)	No	Graph classification and text classification	COLLAB, IMDB-B, IMDB-M, DD, PROTEINS, MUTAG, PTC, MR, R8, R52, Ohsumed	Yes
DTI-GTN [109]	No	Yes	Yes (interleaving)	Yes (quadratic)	No	Drug-target interaction prediction	Peng et al. [110] dataset	No
Dual-GCN+Transformer [111]	No	No	No	Yes (linear)	No	Image Captioning	MS COCO and Visual Genome	Yes
SGTransformer [112]	Yes	Yes	No	Yes (quadratic)	Yes (message passing)	Scene graph to image generation	Visual Genome, COCO-Stuff and CLEVR-Dialog	Yes,
TPT [113]	Yes	No	No	Yes	No	Video scene graph generation	VidHOI and Action Genome	No

Scalable and Pre-Trained Graph Transformers

TABLE IV
SCALABLE GRAPH TRANSFORMERS

Model	Graph inductive bias			Graph attention mechanisms		Application	Datasets	Code availability
	Node Positional encoding	Edge Structural Encoding	Message-Passing Bias	Global attention	Local Attention			
GPS [39]	Yes (local and global)	Yes (local and relative)	Yes (inter-leaving)	Yes (quadratic and linear)	Yes (message passing)	Graph representation learning	ZINC, PCQM4Mv2, CIFAR10, MalNet-Tiny, OGB benchmarks	Yes
NodeFormer [53]	No	No	Yes (relational bias)	Yes (linear)	No	Node classification	Citation networks, OGB-Proteins, Amazon2M, Mini-ImageNet, 20News-Groups	Yes
DIFFORMER [115]	No	No	No	Yes (linear)	Yes (spectral)	Node classification, image/text classification, spatial-temporal dynamics prediction	Cora, Citeseer, Pubmed, ogbn-Proteins, Pokec, CIFAR, STL, 20News	Yes
DyFormer [116]	Yes (global)	Yes (global)	Yes (pre-processing)	Yes (quadratic)	No	Dynamic graph representation learning	RDS, UCI, Yelp, ML-10M, SNAP-Wikipedia, SNAP-Reddit	No
Graph ViT/MLP-Mixer [117]	Yes	No	No	No	Yes (message passing)	Graph representation learning	ZINC, MNIST, CIFAR10, MolTOX21, MolHIV, Peptide-func, Peptide-struct, CSL, EXP, SR25, LRGB, TreeNeighbourMatch	Yes
DiGress [118]	No	Yes	No	Yes	Yes	Graph generation	QM9, MOSES, GuacaMol, SBM, planar graphs	Yes
EXPHORMER [119]	Yes (local and global)	Yes (local and global)	Yes (inter-leaving)	Yes (linear)	Yes (spectral)	Graph learning and representation tasks	CIFAR10, MalNet-Tiny, MNIST, CLUSTER, PATTERN, PascalVOC-SP, COCO-SP, PCQM-Contact, ogbn-arxiv,	Yes
EGT [120]	Yes (global)	Yes (global)	No	Yes (quadratic)	No	Node and edge classification, graph classification and regression, transfer learning.	PATTERN, CLUSTER, TSP, MNIST, CIFAR10, ZINC, PCQM4M, PCQM4Mv2, MolPCBA and MolHIV.	Yes
GRIT [121]	Yes	No	No	Yes (quadratic)	No	Graph regression and classification	ZINC, MNIST, CIFAR10, PATTERN, CLUSTER, Peptides-func, Peptides-struct, ZINC-full, PCQM4Mv2	Yes
HSGT [122]	No	Yes (local, based on shortest path distance)	No	Yes (quadratic)	Yes (message passing, based on neighborhood sampling)	Node classification	Cora, CiteSeer, PubMed, Amazon-Photo, ogbn-arxiv, ogbn-proteins, ogbn-products, Reddit, Flickr, Yelp	No
Higher-Order Transformers [123]	No	No	No	Yes (quadratic and linear)	Yes (message passing)	Graph and hypergraph modeling	Synthetic chains, PCQM4M-LSC, Jets, Delaunay, Hyperedge prediction	Yes
EGAT [73]	No	No	Yes (inter-leaving)	Yes (linear)	No	Graph representation learning and molecular property prediction	AMLSim, Cora, Citeseer, PubMed	No
L-CAT [90]	No	No	No	Yes (linear)	No	Node classification	CSBM, Amazon, Facebook, GitHub, TwitchEN, Coauthor Physics, OGB (arxiv, products, mag, proteins)	Yes
Point Transformer [88]	Yes (relative)	No	No	No	Yes (message passing)	3D point cloud processing	S3DIS, ModelNet40, ShapeNetPart	Yes
Specformer [92]	No	No	No	Yes (linear)	Yes (spectral)	Graph representation learning	Synthetic, node-level and graph-level datasets	Yes
HierGAT [124]	Yes (local)	No	No	Yes (linear)	No	Entity Resolution	Magellan and WDC product matching datasets	Yes
Geo-ER [125]	No	No	No	Yes (linear)	Yes (message passing)	Geospatial Entity Resolution	OSM, FSQ, Yelp	Yes
GTA [126]	No	No	No	Yes (linear)	Yes (message passing)	Multivariate time series anomaly detection in IoT	SWaT, WADI, SMAP, MSL	Yes
GraTransDRP [127]	Yes	No	No	Yes (linear)	No	Drug response prediction	GDSC	Yes
LGT [128]	Yes (local and global)	No	No	Yes (linear)	Yes (message passing)	Molecular properties prediction	ZINC, QM9, ESOL, FreeSolv	No,
GTAGC [129]	Yes (global)	No	No	Yes (linear)	Yes (spectral)	Graph clustering	Citeseer, Cora, Pubmed	No
ANS-GT [130]	Yes (local and global)	No	No	Yes (linear)	Yes (message passing)	Node classification	Cora, Citeseer, Pubmed, Chameleon, Actor, Squirrel, Cornell, Texas, Wisconsin	No
RelTR [131]	No	No	No	Yes (linear)	No	Scene graph generation	Visual Genome, Open Images V6, VRD	Yes
BSTG-Trans [132]	Yes (local and global)	Yes (local and global)	No	Yes (linear)	Yes (message Passing)	Long-term Pose Forecasting	Human3.6M, HumanEva-I, Human360K	Yes
GraphSum [133]	Yes (local and global)	No	No	Yes (linear)	Yes (message passing)	Abstractive MDS	WikiSum and MultiNews	Yes
KG-Transformer [134]	No	No	Yes (pre-processing)	Yes (linear)	No	Text generation from knowledge graphs	UMLS, PubMed, 2010 i2b2/VA, JNLPBA, BC5CDR, GAD, EU-ADR	No

TABLE V
PRE-TRAINED GRAPH TRANSFORMERS

Model	Graph inductive bias			Graph attention mechanisms		Application	Datasets	Code availability
	Node Positional encoding	Edge Structural Encoding	Message-Passing Bias	Global attention	Local Attention			
GRAPHIX-T5 [136]	No	Yes	Yes (pre-processing)	Yes (linear)	No	Text-to-SQL	SPIDER, SYN, DK, REALISTIC, SPIDER-SSP	Yes
GROVER [137]	No	No	Yes (inter-leaving)	Yes (linear)	Yes (message passing)	Molecular representation learning	MoleculeNet	No
G-BERT [135]	No	No	No	Yes (linear)	No	Medication recommendation	MIMIC-III	Yes
ChemBERTa [138]	No	No	No	Yes (quadratic)	No	Molecular property prediction	PubChem 77M, MoleculeNet	Yes
HEAT [139]	Yes (local)	No	No	Yes (linear)	Yes (message passing)	Structured reconstruction	SpaceNet Challenge and Structured3D	Yes
FormNetV2 [140]	No	No	No	Yes (linear)	Yes (message passing)	Form document information extraction	FUNSD, CORD, SROIE	No
MPG [143]	No	No	No	Yes (linear)	Yes (spectral)	Drug discovery	MoleculeNet, DDI, DTI	Yes
LiGhT [144]	No	Yes (path encoding and distance encoding)	No	Yes (linear)	No	Molecular property prediction	ChEMBL29, BACE, BBBP, ClinTox, SIDER, Estrogen, MetStab, Tox21, ToxCast, FreeSolv, Lipophilicity	Yes
LightGT [146]	Yes (global)	No	No	Yes (linear)	No	Multimedia recommendation	Movielens, Tiktok, Kwai	Yes
KGTransformer [145]	No	No	No	Yes (linear)	Yes (message passing)	Knowledge graph transfer	WFC, WN18RR, AwA-KG, CommonsenQA	Yes
CoVGT [147]	No	No	No	Yes (linear)	Yes (message-passing)	Video Question Answering	NExT-QA, TGIF-QA, TGIF-QA-R, STAR-QA, Causal-VidQA, MSRVTT-QA	Yes
G-Adapter [148]	No	Yes (local)	No	Yes (quadratic)	No	Molecular graph tasks	MolHIV, MolPCBA, FreeSolv, ESOL, BBBP, Estrogen-α, Estrogen-β, MetStablow, MetStabhigh	Yes
kgTransformer [149]	No	No	Yes (inter-leaving)	Yes (quadratic)	No	Knowledge graph reasoning for complex logical queries	FB15k-237, NELL995	Yes
Li et al. [149]	Yes (local and global)	No	No	Yes (linear)	Yes (message passing)	Movement synchrony estimation in autism therapy interventions	PT13 and Human3.6MS	No
Pellegrini et al. [150]	No	Yes	No	Yes (linear)	No	Disease prediction using graph convolutional networks	TADPOLE and MIMIC-III	Yes
Video Graph Transformer (VGT) [151]	No	No	No	Yes (linear)	Yes (message passing)	Video Question Answering (VideoQA)	NExT-QA, TGIF-QA, MSRVTT-QA	Yes
Tan et al. [152]	No	No	No	Yes (linear)	No	Few-shot node classification	CoraFull, ogbn-arxiv, CiteSeer, Cora	No
GPT-GNN [60]	No	Yes	Yes	No	Yes	Graph representation learning and graph mining	OAG, Amazon, Reddit, OAG (citation)	Yes

Some corny jokes to wrap it up.. Cudos to Claude

Why did the Graph Neural Network refuse to attend the party?

Because it heard the Graph Attention Network was coming, and it knew it would never get any attention! Meanwhile, the Graph Transformer Network could see and interact with everyone in the room at once, but still couldn't decide which conversations were actually important

Why did the Graph Neural Network become a successful therapist?

Because it was great at node counseling and maintaining healthy relationships! The Graph Attention Network tried to follow suit but couldn't stop focusing on just the popular vertices. As for the Graph Transformer Network, it kept transforming everyone's problems into entirely different ones!

Couldn't leave that one out..

References I

[1] Josh Abramson, Jonas Adler, Jack Dunger, Richard Evans, Tim Green, Alexander Pritzel, Olaf Ronneberger, Lindsay Willmore, Andrew J Ballard, Joshua Bambrick, et al. Accurate structure prediction of biomolecular interactions with alphafold 3. *Nature*, 630(8016):493–500, 2024.

[2] Jay Alammar. The illustrated transformer, 2018. URL `https://jalammar.github.io/illustrated-transformer/`. Accessed: 2025-03-22.

[3] Michele Coscia. The atlas for the aspiring network scientist. *arXiv preprint arXiv:2101.00863*, 2021.

[4] Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. arxiv. *arXiv preprint arXiv:2012.09699*, 2020.

References II

[5] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[6] Junjie Huang, Huawei Shen, Liang Hou, and Xueqi Cheng. Signed graph attention networks. In *Artificial Neural Networks and Machine Learning–ICANN 2019: Workshop and Special Sessions: 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17–19, 2019, Proceedings 28*, pages 566–577. Springer, 2019.

[7] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *nature*, 596 (7873):583–589, 2021.

References III

[8] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*. 3rd edition, 2025. URL `https://web.stanford.edu/~jurafsky/slp3/`. Online manuscript released January 12, 2025.

[9] Pushkar Mishra, Aleksandra Piktus, Gerard Goossen, and Fabrizio Silvestri. Node masking: Making graph neural networks generalize and scale better. *arXiv preprint arXiv:2001.07524*, 2020.

[10] Federico Monti, Fabrizio Frasca, Davide Eynard, Damon Mannion, and Michael M Bronstein. Fake news detection on social media using geometric deep learning. *arXiv preprint arXiv:1902.06673*, 2019.

References IV

[11] Bastian Pfeifer, Anna Saranti, and Andreas Holzinger. Gnn-subnet: disease subnetwork detection with explainable graph neural networks. *Bioinformatics*, 38(Supplement_2):ii120–ii126, 2022.

[12] Trang Pham, Truyen Tran, Dinh Phung, and Svetha Venkatesh. Column networks for collective classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

[13] Ahsan Shehzad, Feng Xia, Shagufta Abid, Ciyuan Peng, Shuo Yu, Dongyu Zhang, and Karin Verspoor. Graph transformers: A survey. *arXiv preprint arXiv:2407.09777*, 2024.

[14] Jonathan M Stokes, Kevin Yang, Kyle Swanson, Wengong Jin, Andres Cubillos-Ruiz, Nina M Donghia, Craig R MacNair, Shawn French, Lindsey A Carfrae, Zohar Bloom-Ackermann, et al. A deep learning approach to antibiotic discovery. *Cell*, 180(4): 688–702, 2020.

References V

[15] Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. *arXiv preprint arXiv:2111.14522*, 2021.

[16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[17] Andreas Veit, Michael J Wilber, and Serge Belongie. Residual networks behave like ensembles of relatively shallow networks. *Advances in neural information processing systems*, 29, 2016.

[18] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

References VI

[19] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. Heterogeneous graph attention network. In *The world wide web conference*, pages 2022–2032, 2019.

[20] Feng Xia, Ke Sun, Shuo Yu, Abdul Aziz, Liangtian Wan, Shirui Pan, and Huan Liu. Graph learning: A survey. *IEEE Transactions on Artificial Intelligence*, 2(2):109–127, 2021.

[21] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *International conference on machine learning*, pages 5453–5462. PMLR, 2018.

References VII

[22] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 974–983, 2018.

[23] Jiaxuan You, Zhitao Ying, and Jure Leskovec. Design space for graph neural networks. *Advances in Neural Information Processing Systems*, 33:17009–17021, 2020.

[24] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. Graph transformer networks. *Advances in neural information processing systems*, 32, 2019.

[25] Chenhan Zhang, JQ James, and Yi Liu. Spatial-temporal graph attention networks: A deep learning approach for traffic forecasting. *Ieee Access*, 7:166246–166256, 2019.