Graph attention networks

LECTURE 16

Attention fundamentals and learning on graph neighborhoods

General attention · GAT mechanics · Applications

How does a node decide which information to use?

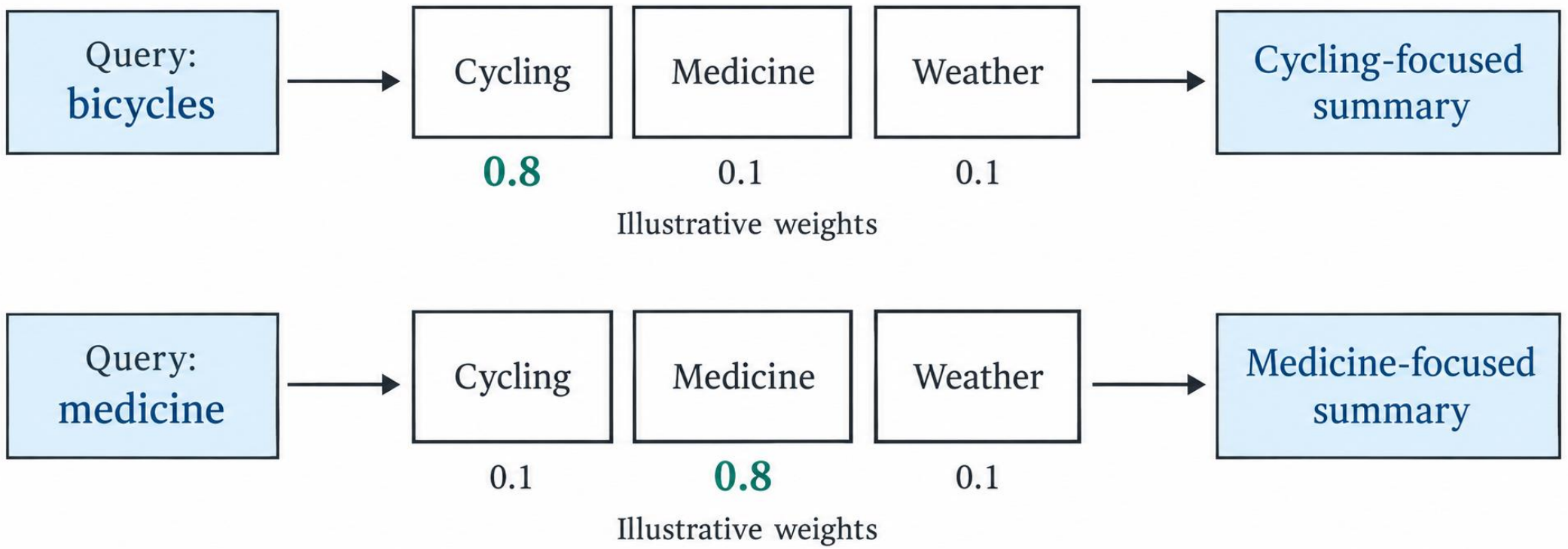
Veličković et al. (ICLR 2018). Graph Attention Networks.

What composes today’s lecture

	Topic	What you should be able to do
1	Attention fundamentals	Explain query, key, value and weighted averaging
2	Graph attention layers	Calculate a masked neighborhood update
3	Prediction tasks	Connect node embeddings to a task loss
4	Limits and discussion	Recognize ranking limits and implementation pitfalls

One recurring computation: score, normalize, mix

Attention as a query-dependent summary



Changing the query changes the weights, while the candidate documents stay the same

Bahdanau et al. (ICLR 2015). Neural Machine Translation by Jointly Learning to Align and Translate.

Query, key, and value

Where the columns come from

Inputs one feature vector per entry

x_1 repair manual, x_2 weather report,

x_3 medical article; x_q the question

Key

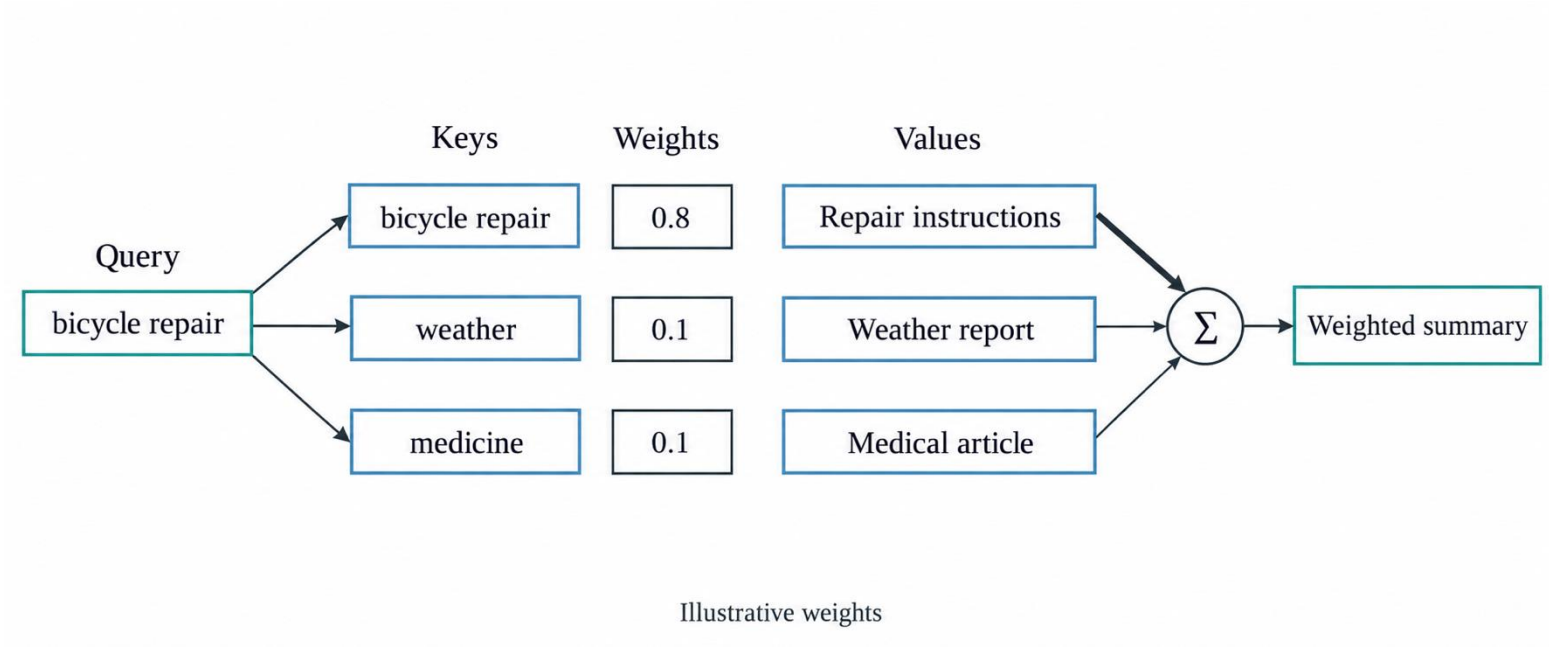
how entry is found

Value

what entry returns

Query

what is being looked for



Same input, two projections: W_K decides how an entry is found, W_V decides what it returns.

The general attention operation

1 SCORE

$$e_{ij} = s(q_i, k_j)$$

2 NORMALIZE

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{r \in C(i)} \exp(e_{ir})}$$

3 MIX VALUES

$$o_i = \sum_{j \in C(i)} \alpha_{ij} v_j$$

$C(i)$ is the candidate set for receiver i. The weights sum to one over this set.

The score function and candidate set define the attention variant

Bahdanau et al. (2015). Neural Machine Translation by Jointly Learning to Align and Translate.
Vaswani et al. (2017). Attention Is All You Need.

A three-candidate attention example

Constructed example: one query, three scalar values

Candidate	Score	Exponential	Weight	Value
A	ln 2	2	1/2	2
B	0	1	1/4	6
C	0	1	1/4	10

$$o = \frac{1}{2} \cdot 2 + \frac{1}{4} \cdot 6 + \frac{1}{4} \cdot 10 = 5$$

Equal averaging gives 6. Attention gives 5 by emphasizing candidate A.

Vaswani et al. (2017). Attention Is All You Need.

Self-attention and a common scoring choice

SELF-ATTENTION: ALL ROLES COME FROM THE SAME INPUT COLLECTION

$$q_i = W_Q x_i, \qquad k_j = W_K x_j, \qquad v_j = W_V x_j$$

SCALED DOT-PRODUCT SCORE

$$e_{ij} = \frac{q_i{}^T k_j}{\sqrt{d_k}}$$

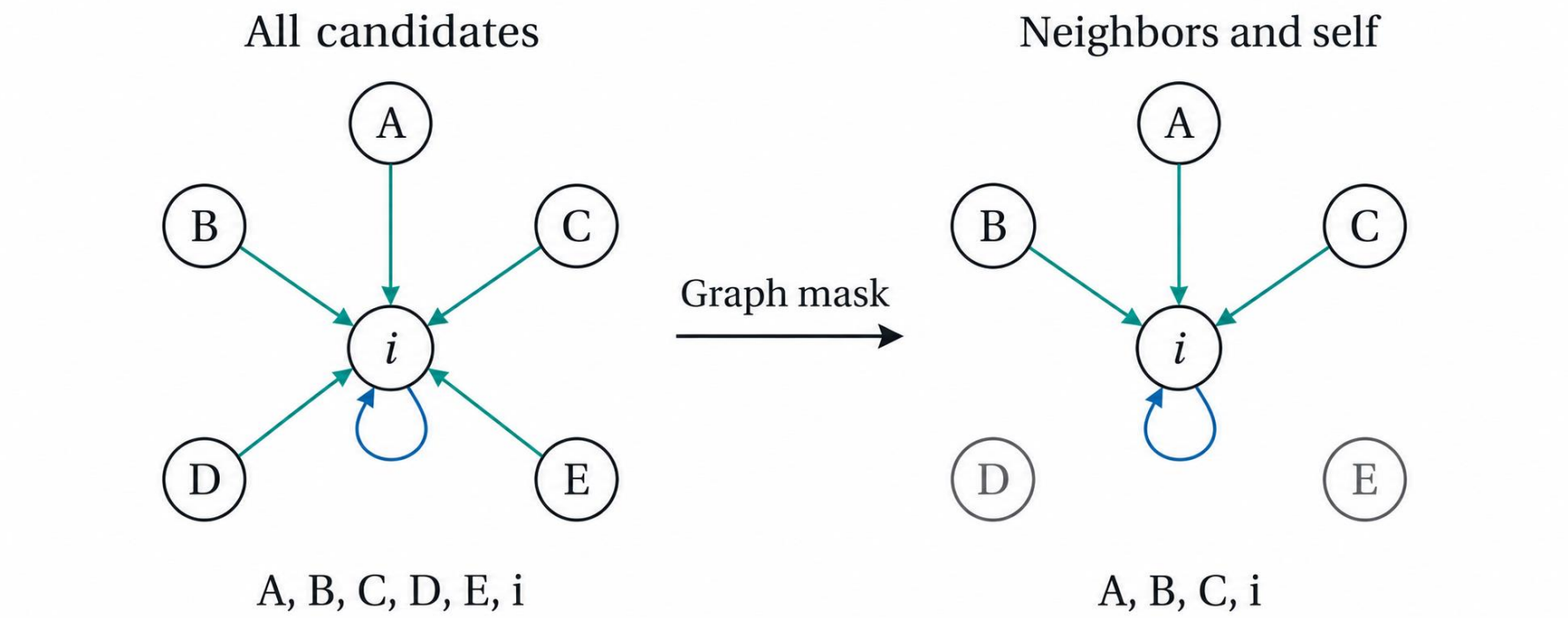
d_k = dimension of each query and key -- keeps the scores at unit scale so the softmax does not saturate

Cross-attention uses queries from one collection and keys / values from another.

Original GAT uses an additive scorer instead of this dot-product scorer

Vaswani et al. (2017). Attention Is All You Need.

General attention becomes graph-masked attention



$$C(i) = N(i) \cup \{i\}$$

GAT restricts candidates to neighbors and self, then scores and combines their features

Veličković et al. (ICLR 2018). Graph Attention Networks.

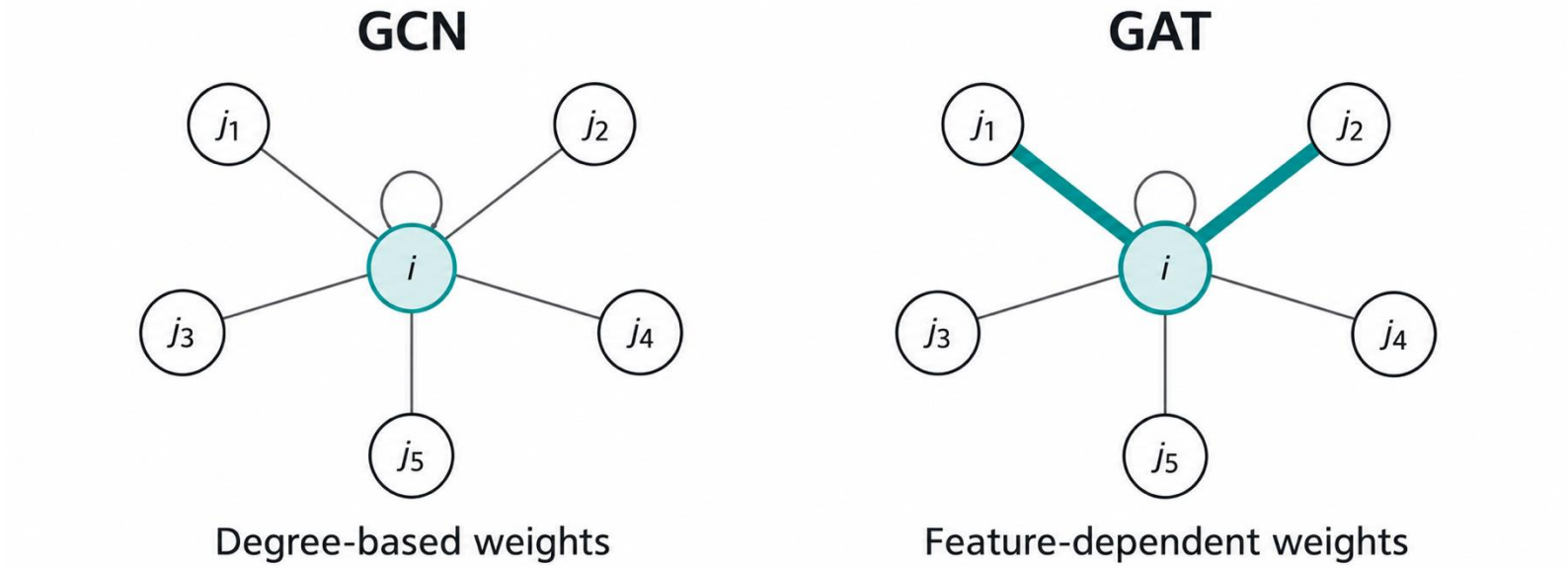
General attention and original GAT

General role	Original GAT realization
Query role	Receiving node’s transformed feature
Key role	Sending neighbor’s transformed feature
Value	Sending neighbor’s transformed feature
Candidate set	Incoming graph neighbors, including self
Score function	Shared additive scorer applied to the node pair

The query, key, and value roles share a transformed representation in original GAT

Veličković et al. (ICLR 2018). Graph Attention Networks.
Vaswani et al. (NeurIPS 2017). Attention Is All You Need.

GCN and GAT weight neighbors differently



Degree-based coefficients
for a fixed graph

Coefficients computed from features
using a learned scoring rule

Both learn feature transformations. GAT also learns how to calculate edge weights.

Kipf & Welling (ICLR 2017). Semi-Supervised Classification with Graph Convolutional Networks.
Veličković et al. (ICLR 2018). Graph Attention Networks.

Step 1: shared feature transformation

Node feature

$$h_i \in \mathbb{R}^F,$$

Weight matrix

$$W \in \mathbb{R}^{F' \times F},$$

Output feature

$$z_i = W h_i \in \mathbb{R}^{F'}$$

ROW-STACKED MATRIX NOTATION

$$H \in \mathbb{R}^{n \times F}, \qquad Z = HW^T \in \mathbb{R}^{n \times F'}$$

PARAMETER COUNT PER HEAD, WITHOUT BIAS

$$F'F + 2F'$$

One shared map changes the feature space at every node

Step 2: additive attention scores

$$e_{ij} = \text{LeakyReLU}\left(a^T \left[z_i \parallel z_j\right]\right)$$

Symbol	Meaning
z_i and z_j	Transformed receiver and sender features
$\parallel$	Concatenation of the two feature vectors
a	Shared learned vector with $2F'$ entries
e_{ij}	Scalar logit, before normalization

W and a are parameters. The logits are recomputed from the current input.

Step 3: normalize within each neighborhood

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{r \in C(i)} \exp(e_{ir})}$$

$$\sum_{j \in C(i)} \alpha_{ij} = 1, \qquad \alpha_{ij} \geq 0$$

Normalize separately for each receiver and each head.
Non-neighbors have zero weight through the mask.

The sum-to-one property holds before attention dropout

Veličković et al. (ICLR 2018). Graph Attention Networks.
PyTorch Geometric. GATConv documentation.

Step 4: one complete numerical node update

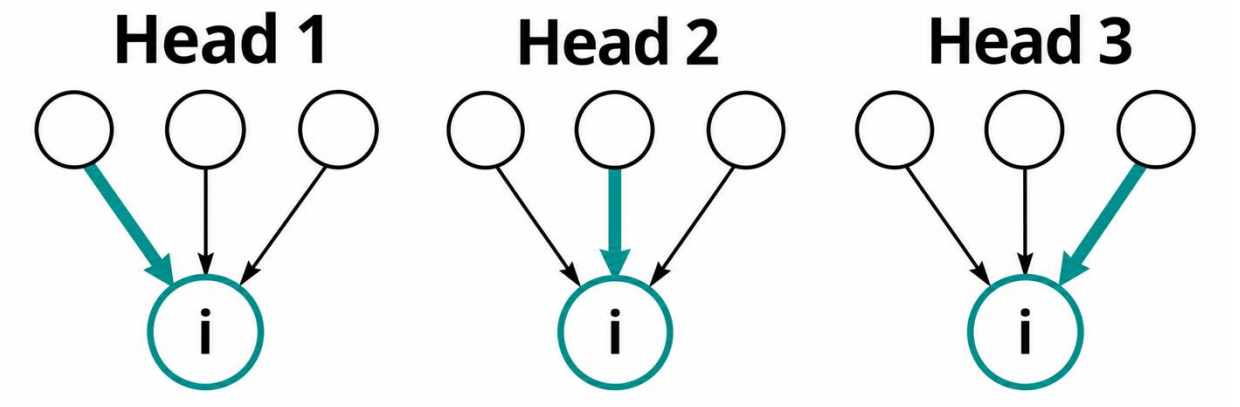
$$h'_i = \sigma\left(\sum_{j \in C(i)} \alpha_{ij} z_j\right)$$

Candidate	Logit	Weight ≈	Scalar value
Neighbor 1	1.4	0.480	2
Neighbor 2	0.8	0.263	0
Neighbor 3	−0.2	0.097	4
Self	0.3	0.160	1

$$h'_i \approx \mathrm{ReLU}(1.507) = 1.507$$

Multiply each message by its weight, then sum and activate

Multiple heads and output dimensions



$$u_i^{(k)} = \sigma\left(\sum_{j \in C(i)} \alpha_{ij}^{(k)} W^{(k)} h_j\right)$$

Each head has its own parameters and produces its own output.

$$h'_i = u_i^{(1)} \,\|\, \cdots \,\|\, u_i^{(K)} \in \mathbb{R}^{KF'}$$

Concatenation gives width KF'. Averaging head outputs keeps width F'.

Veličković et al. (ICLR 2018). Graph Attention Networks.

The complete single-head GAT layer

TRANSFORM

$$z_i = W h_i$$

SCORE

$$e_{ij} = \text{LeakyReLU}\left(a^T \left[z_i \parallel z_j\right]\right)$$

NORMALIZE

$$\alpha_{ij} = \frac{\exp\left(e_{ij}\right)}{\sum_{r \in C(i)} \exp\left(e_{ir}\right)}$$

AGGREGATE

$$h'_i = \sigma\left(\sum_{j \in C(i)} \alpha_{ij} z_j\right)$$

The task loss trains the shared transform and scorer through backpropagation

Node prediction: paper classification

INPUT

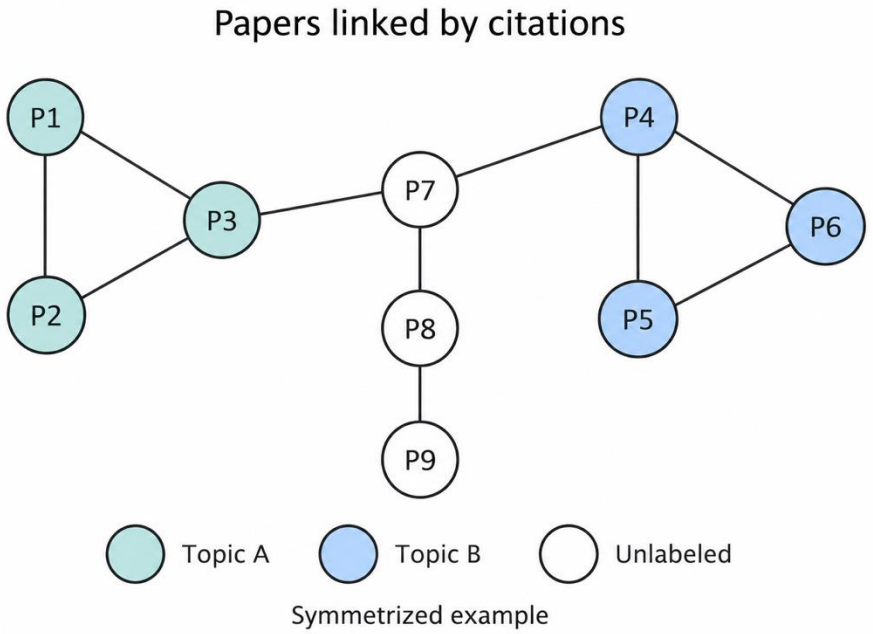
Text features for each paper

GRAPH

Citation connections
State whether edges are directed or symmetrized.

OUTPUT

One topic distribution per paper



Unlabeled nodes can participate in message passing without supplying training labels

Veličković et al. (2018). Graph Attention Networks.

Training labels and the classification loss

$$L = -\sum_{i \in V_{train}} \sum_{c=1}^{C} y_{ic} \log \hat{y}_{ic}$$

TWO DIFFERENT SOFTMAX OPERATIONS

Softmax over neighbors	Softmax over classes
Produces attention weights	Produces predicted label probabilities
Used inside every attention head	Used at the classification output

Only training labels enter the loss. Validation and test labels have separate roles.

Veličković et al. (ICLR 2018). Graph Attention Networks.

Node relabeling and use on new graphs

$$\Phi(PH, PAP^T) = P\Phi(H, A)$$

PERMUTATION EQUIVARIANCE

Relabeling the graph relabels the node outputs in the same way.

INDUCTIVE USE

The same parameters can run on another graph with compatible features.

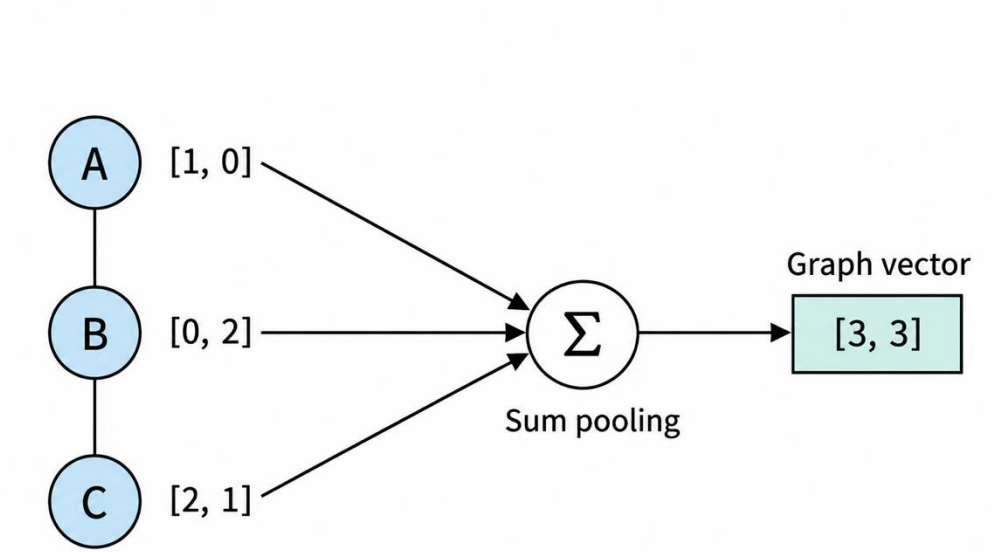
Architectural reuse across sizes does not by itself guarantee transfer accuracy

Veličković et al. (2018). Graph Attention Networks.

Graph prediction needs an invariant readout

$$g = \sum_{i=1}^{n} h_i^{(L)}$$

$$g = \frac{1}{n} \sum_{i=1}^{n} h_i^{(L)}$$



Sum retains count-related scale.
Mean removes that direct scale.

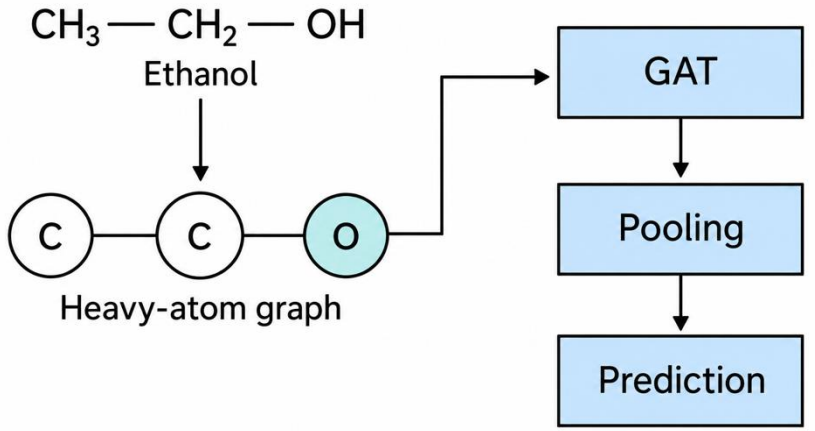
A graph readout must give the same result after node relabeling

Graph prediction: a molecular property

ILLUSTRATIVE BINARY CLASSIFICATION TASK

Atoms supply node features.
Bonds supply edges and optional attributes.

GAT layers and graph pooling produce one prediction for the molecule.



$$L = -y\log\hat{y} - (1-y)\log(1-\hat{y})$$

The local layer is reusable. The readout and loss determine the prediction task.

Original GAT has a restricted neighbor ranking

$$e_{ij} = \text{LeakyReLU}\left(a_L{}^T z_i + a_R{}^T z_j\right)$$

The receiver term adds the same scalar to every candidate.
LeakyReLU and softmax preserve the candidate ordering – static attention

For fixed candidate features, changing the receiver cannot reverse
the ranking of shared neighbors within one head.

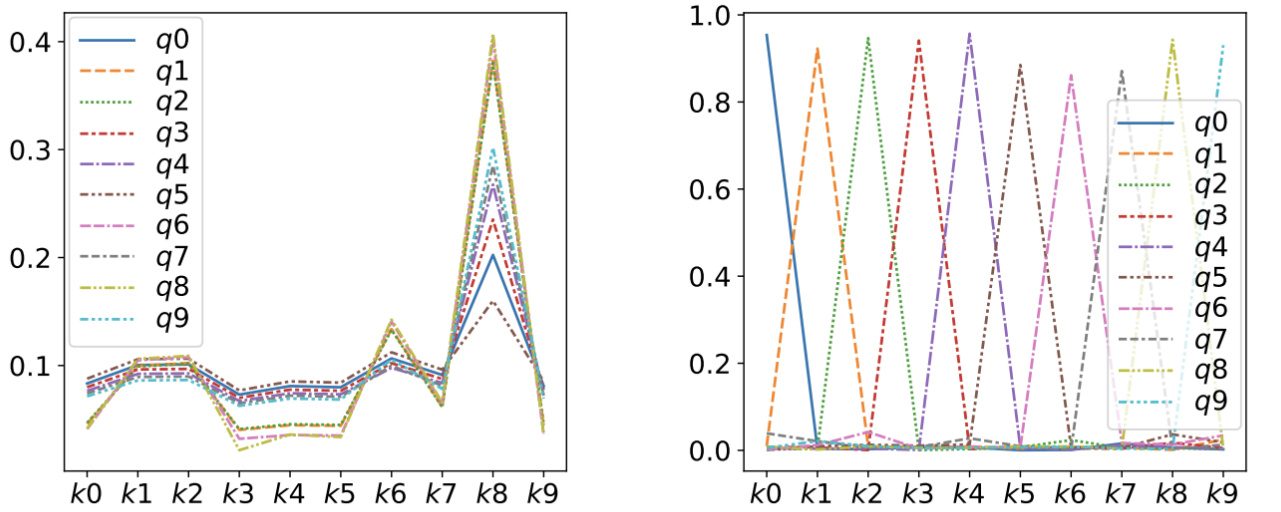
Coefficients can change even when the neighbor ranking cannot

Brody et al. (ICLR 2022). How Attentive Are Graph Attention Networks?

GATv2 allows query-dependent rankings

GATv2 SCORER

$$e_{ij} = a^T \text{LeakyReLU}\left(W\left[h_i \,\|\, h_j\right]\right)$$



(a) Attention in standard GAT (Veličković et al. (2018)) (b) Attention in GATv2, our fixed version of GAT

A one-hidden-layer MLP in (h_i, h_j) can approximate any continuous scoring function.

Vector nonlinearity before the final projection enables richer pair interactions

Brody et al. (2022). How Attentive Are Graph Attention Networks?

Interpreting attention weights

WHAT THE COEFFICIENT TELLS US

How a particular head weights a candidate value in this forward pass.

WHAT ELSE AFFECTS THE PREDICTION

The value magnitude, other messages, later layers, and the readout.

Use feature masking or edge removal to test model sensitivity.

A large coefficient alone does not establish an explanation of the prediction

Jain & Wallace (NAACL 2019). Attention is not Explanation.
Wiegreffe & Pinter (EMNLP 2019). Attention is not not Explanation.

Limits of local attention

OVERSMOOTHING

Repeated mixing can make node representations similar.

Attention does not automatically prevent this behavior.

OVERSQUASHING

Many distant signals must pass through limited connections.

Reweighting edges does not create a new graph path.

Depth, residual paths, graph connectivity, and task range still matter

Wu et al. (NeurIPS 2023). Demystifying Oversmoothing in Attention-Based Graph Neural Networks.
Alon & Yahav (ICLR 2021). On the Bottleneck of Graph Neural Networks and its Practical Implications.

Edge features in the attention score

$$e_{ij} = \text{LeakyReLU}\left(a_t{}^T W h_i + a_s{}^T W h_j + a_e{}^T W_e r_{ij}\right)$$

Quantity	Meaning
h_i	Receiver feature
h_j	Sender feature
r_{ij}	Edge attribute: bond type, distance, or direction
e_{ij}	Scalar attention logit

The edge attribute and the attention logit are different quantities

PyTorch Geometric. GATConv documentation.

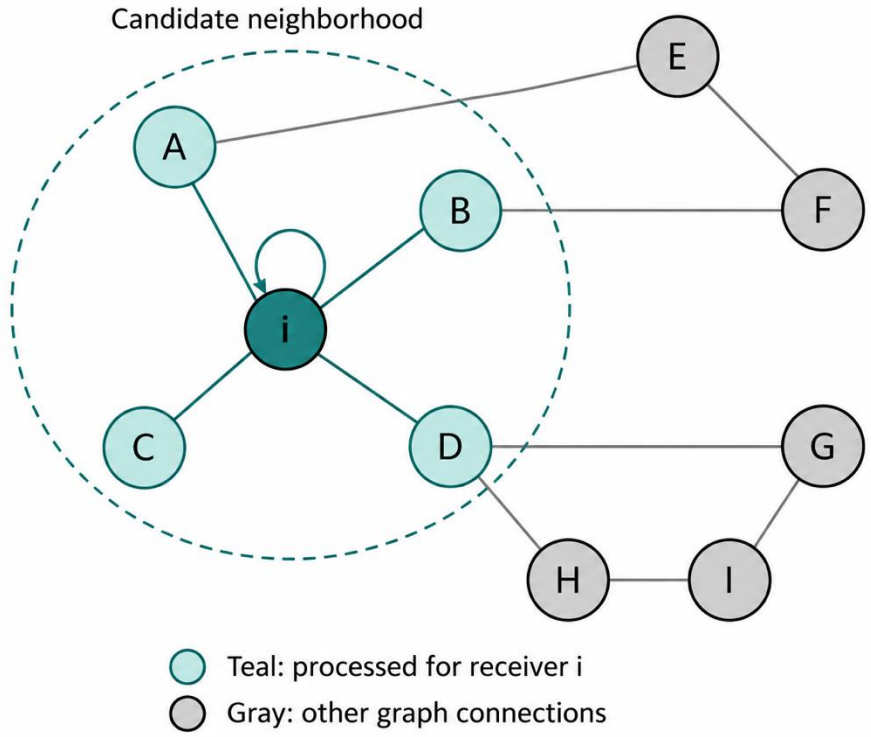
Sparse attention and large graphs

PER-LAYER TIME

$$O\big(K(nFF' + |E|F')\big)$$

K heads, input width F,
per-head output width F'

Count the edges actually processed,
including self-loops.



Sampling saves work but changes the neighborhood softmax denominator

Veličković et al. (ICLR 2018). Graph Attention Networks.
PyTorch Geometric. GATConv documentation.

Implementation and training checks

Check	Why it matters
Receiver / sender convention	Message direction must match the intended task
Self-loops and softmax grouping	Each receiver needs the correct candidate set
Heads and feature width	Concatenation changes the next layer’s input width
Dropout and residual paths	Validate regularization and information flow
Loss mask or graph readout	Match the prediction level and evaluation protocol

A hand-calculated graph is a useful test of indexing and normalization

PyTorch Geometric. GATConv documentation.

References and further reading

Attention foundations

Bahdanau et al., ICLR 2015. Neural Machine Translation by Jointly Learning to Align and Translate.
Vaswani et al., NeurIPS 2017. Attention Is All You Need.

Graph attention

Veličković et al., ICLR 2018. Graph Attention Networks.
Brody et al., ICLR 2022. How Attentive Are Graph Attention Networks?

Interpretation and depth

Jain & Wallace, NAACL 2019. Attention is not Explanation.
Wiegreffe & Pinter, EMNLP 2019. Attention is not not Explanation.
Wu et al., NeurIPS 2023. Demystifying Oversmoothing in Attention-Based Graph Neural Networks.
Alon & Yahav, ICLR 2021. On the Bottleneck of Graph Neural Networks and its Practical Implications.

Background and implementation

Kipf & Welling, ICLR 2017. Semi-Supervised Classification with Graph Convolutional Networks.
PyTorch Geometric. Official GATConv documentation.