UNIVERSITY OF CAMBRIDGE

CVC Centre de Visió per Computador

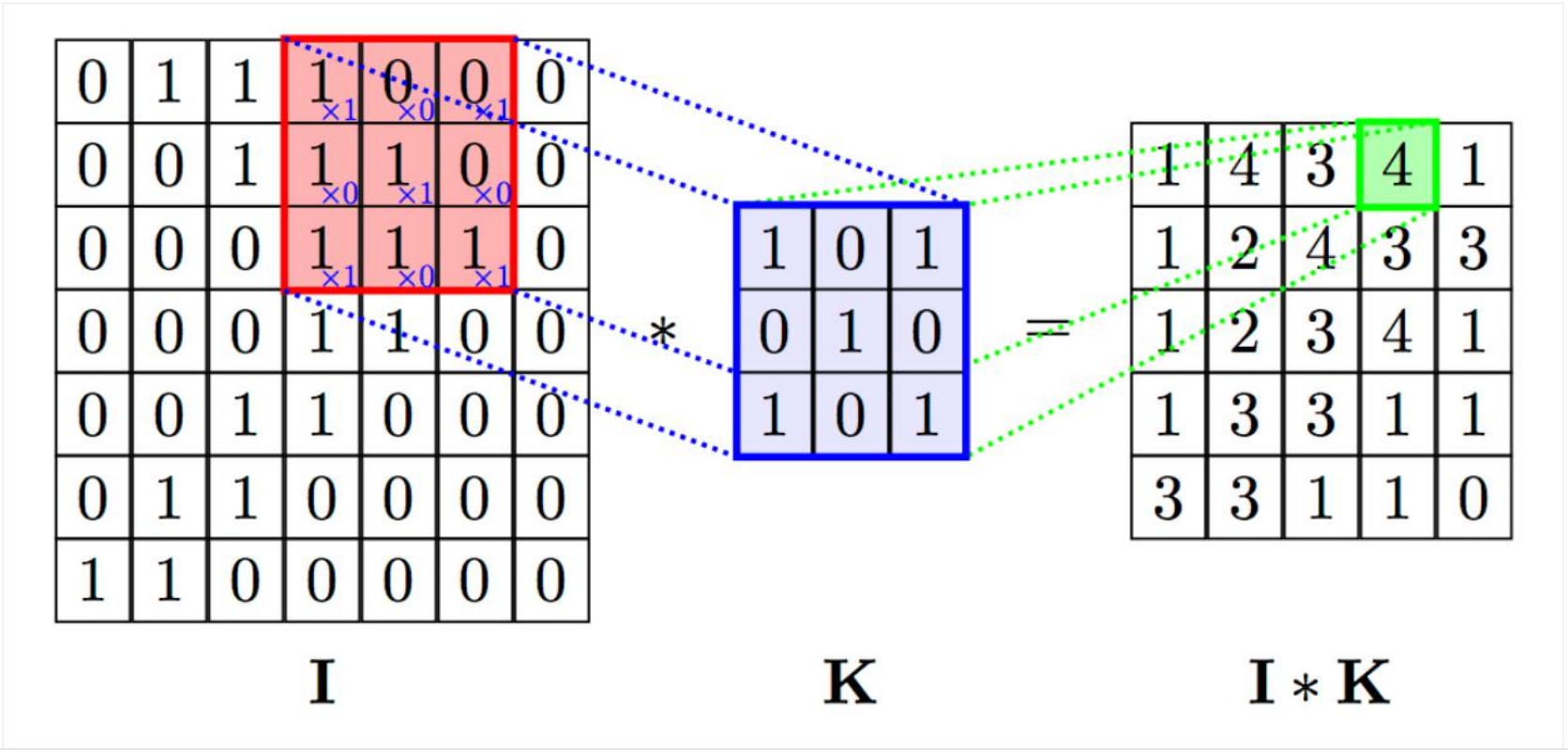
Mila Université de Montréal McGill

Graph Attention Networks

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò and Yoshua Bengio

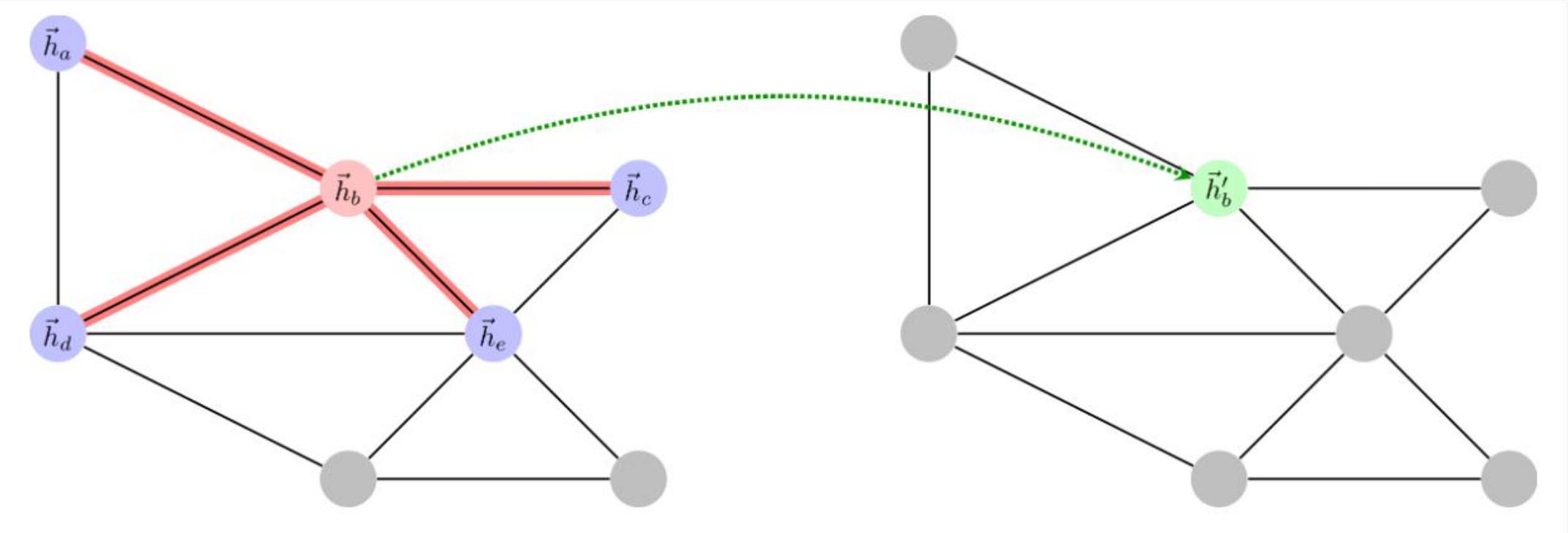
Motivation for graph convolutions

CNNs are a major workforce when it comes to working with image data. They exploit the fact that images have a highly rigid and regular connectivity pattern (each pixel “connected” to its eight neighbouring pixels), making such an operator trivial to deploy (as a small kernel matrix which is slid across the image).



Motivation for graph convolutions

Arbitrary graphs are a **much harder** challenge! Ideally, we would like to aggregate information across each of the nodes' neighbourhoods in a principled manner, but we are no longer guaranteed such rigidity of structure.



A desirable form of a graph convolutional operator.

Enumerating the desirable traits of image convolutions, we arrive at the following properties we would ideally like our graph convolutional layer to have:

- **Computational and storage efficiency** (requiring no more than $O(V + E)$ time and memory);
- **Fixed** number of parameters (independent of input graph size);
- **Localisation** (acting on a *local neighbourhood* of a node);
- Ability to specify **arbitrary importances** to different neighbours;
- Applicability to **inductive problems** (arbitrary, unseen graph structures).

Towards a viable graph convolution

Consider a graph of n nodes, specified as a set of node features, $(\vec{h}_1, \vec{h}_2, \ldots, \vec{h}_n)$, and an adjacency matrix $\mathbf{A}$, such that $\mathbf{A}_{ij} = 1$ if i and j are connected, and 0 otherwise[1]. A **graph convolutional layer** then computes a set of new node features, $(\vec{h}'_1, \vec{h}'_2, \ldots, \vec{h}'_n)$, based on the input features as well as the graph structure.

Every graph convolutional layer starts off with a shared node-wise feature transformation (in order to achieve a higher-level representation), specified by a weight matrix $\mathbf{W}$. This transforms the feature vectors into $\vec{g}_i = \mathbf{W}\vec{h}_i$. After this, the vectors $\vec{g}_i$ are typically recombined in some way at each node.

An attention coefficients αij over neighbourhoods of nodes Ni

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

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

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

where $\cdot^T$ represents transposition and $\|$ is the concatenation operation.

Once obtained, the normalized attention coefficients are used to compute a linear combination of the features corresponding to them, to serve as the final output features for every node

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

For stability, we employ *multi-head attention*—parallelising this process across K independent *attention heads*:

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

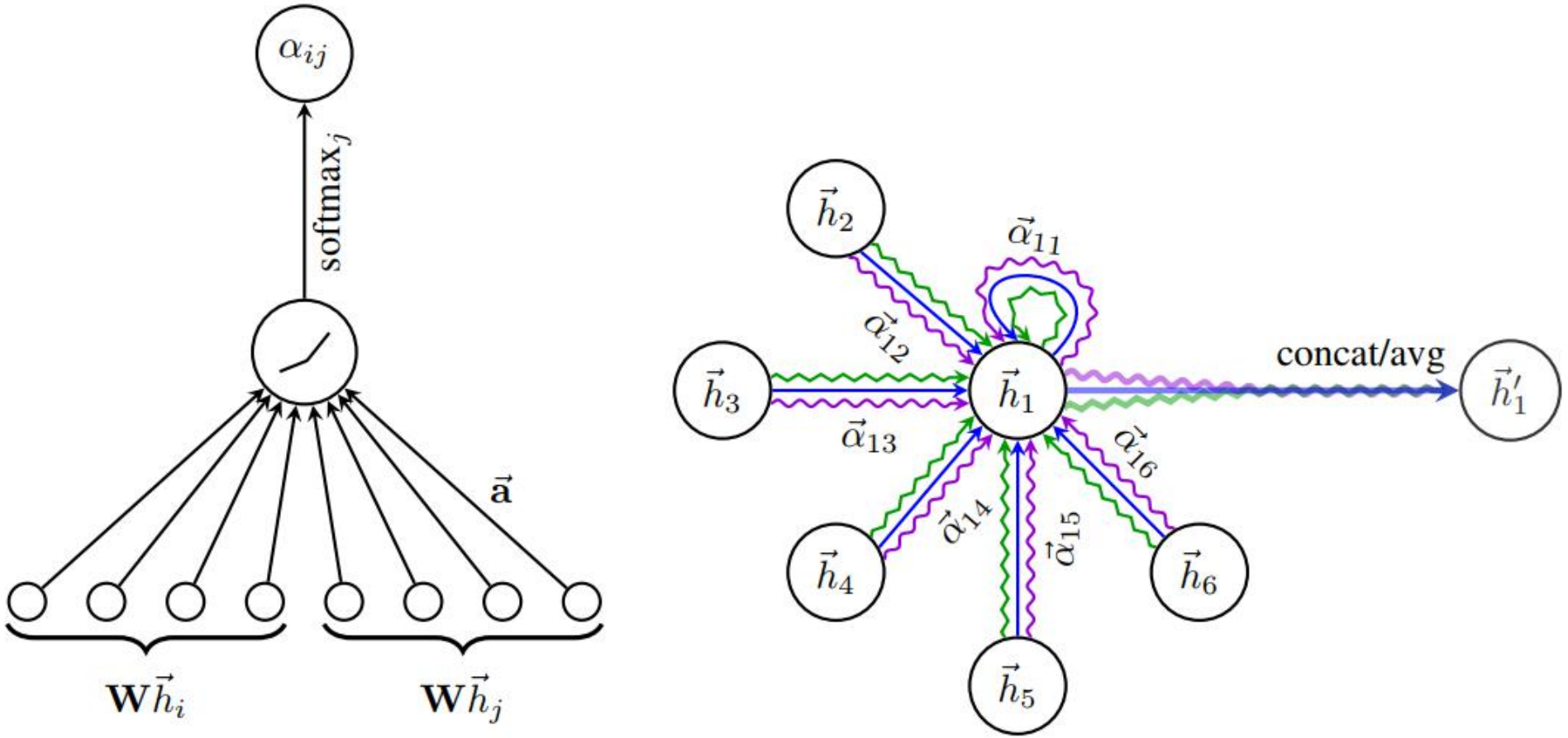


Figure 1: **Left:** The attention mechanism $a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j)$ employed by our model, parametrized by a weight vector $\vec{\mathbf{a}} \in \mathbb{R}^{2F'}$, applying a LeakyReLU activation. **Right:** An illustration of multi-head attention (with $K = 3$ heads) by node 1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

Properties

Computationally efficient: attention computation can be parallelised across all edges of the graph, and aggregation across all nodes!

Storage efficient—a sparse version does not require storing more than $O(V + E)$ entries anywhere;

Fixed number of parameters (dependent only on the desirable feature count, not on the node count);

Trivially **localised** (as we aggregate only over neighbourhoods);

Allows for (implicitly) specifying **different importances** to **different neighbours**, through its attentional mechanism;

Readily applicable to **inductive problems** (as it is a shared *edge-wise* mechanism that does not depend on the global graph structure)!

Satisfies all of the major requirements for a *graph convolutional layer* simultaneously.

Experiment

Table 2: Summary of results in terms of classification accuracies, for Cora, Citeseer and Pubmed. GCN-64* corresponds to the best GCN result computing 64 hidden features (using ReLU or ELU).

Transductive

Method	**Cora**	**Citeseer**	**Pubmed**
MLP	55.1%	46.5%	71.4%
ManiReg (Belkin et al., 2006)	59.5%	60.1%	70.7%
SemiEmb (Weston et al., 2012)	59.0%	59.6%	71.7%
LP (Zhu et al., 2003)	68.0%	45.3%	63.0%
DeepWalk (Perozzi et al., 2014)	67.2%	43.2%	65.3%
ICA (Lu & Getoor, 2003)	75.1%	69.1%	73.9%
Planetoid (Yang et al., 2016)	75.7%	64.7%	77.2%
Chebyshev (Defferrard et al., 2016)	81.2%	69.8%	74.4%
GCN (Kipf & Welling, 2017)	81.5%	70.3%	**79.0%**
MoNet (Monti et al., 2016)	81.7 ± 0.5%	—	78.8 ± 0.3%
GCN-64*	81.4 ± 0.5%	70.9 ± 0.5%	**79.0** ± 0.3%
GAT (ours)	**83.0** ± 0.7%	**72.5** ± 0.7%	**79.0** ± 0.3%

Table 3: Summary of results in terms of micro-averaged F_1 scores, for the PPI dataset. GraphSAGE* corresponds to the best GraphSAGE result we were able to obtain by just modifying its architecture. Const-GAT corresponds to a model with the same architecture as GAT, but with a constant attention mechanism (assigning same importance to each neighbor; GCN-like inductive operator).

Inductive

Method	**PPI**
Random	0.396
MLP	0.422
GraphSAGE-GCN (Hamilton et al., 2017)	0.500
GraphSAGE-mean (Hamilton et al., 2017)	0.598
GraphSAGE-LSTM (Hamilton et al., 2017)	0.612
GraphSAGE-pool (Hamilton et al., 2017)	0.600
GraphSAGE*	0.768
Const-GAT (ours)	0.934 ± 0.006
GAT (ours)	$\mathbf{0.973} \pm 0.002$

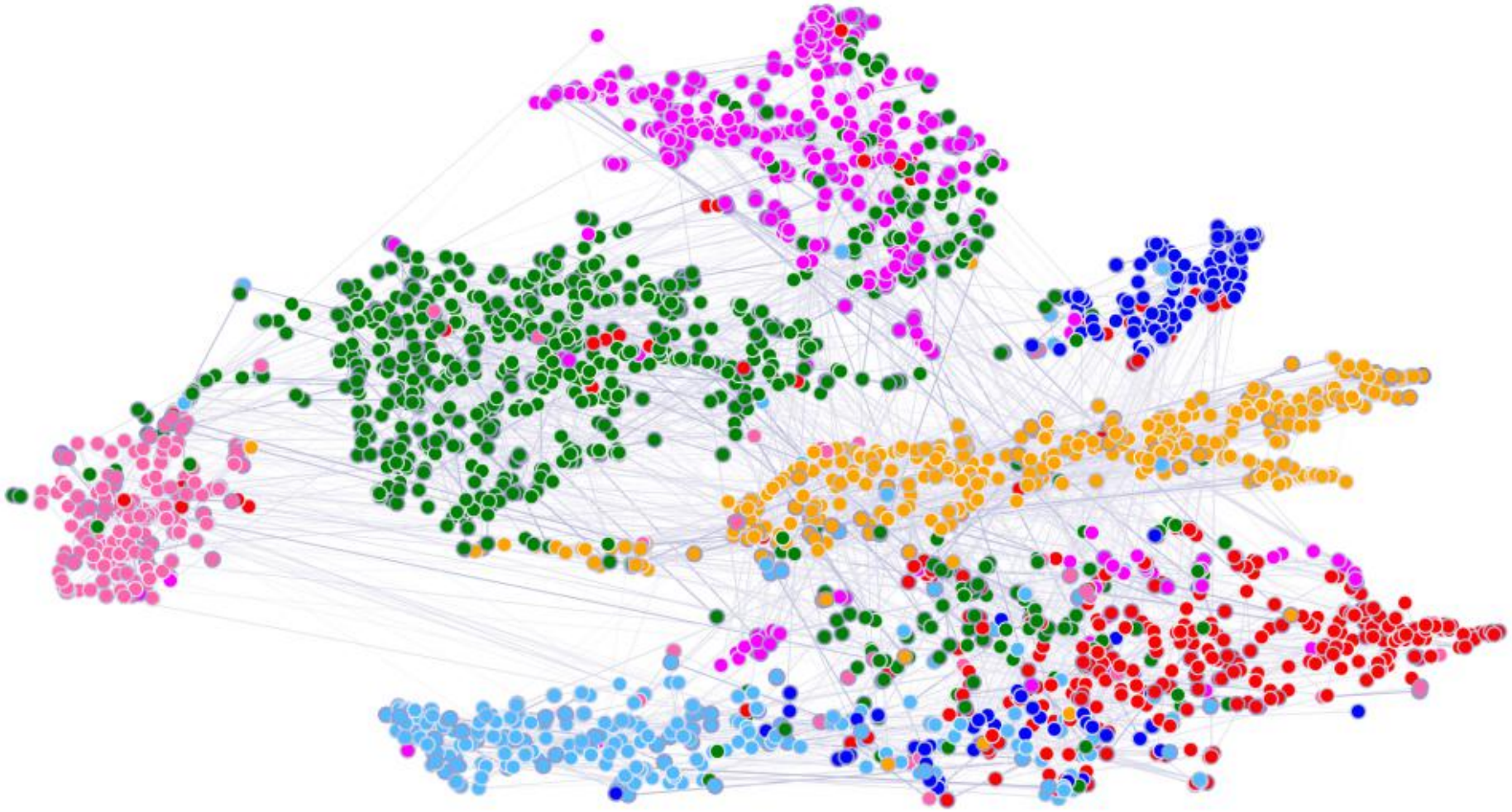


Figure 2: A t-SNE plot of the computed feature representations of a pre-trained GAT model's first hidden layer on the Cora dataset. Node colors denote classes. Edge thickness indicates aggregated normalized attention coefficients between nodes i and j, across all eight attention heads ($\sum_{k=1}^{K} \alpha_{ij}^{k} + \alpha_{ji}^{k}$).