Appendix

Table of Contents

A Theoretical results

A.1 A hard example for GCN

In this subsection, we present a dataset and classification task for which GCN performs poorly. Note that we follow the similar techniques and notation as [14], as described in the main paper.

We recall our data model. Fix $n, d \in \mathbb{N}$ and let $\varepsilon_1, \ldots, \varepsilon_n$ be i.i.d uniformly sampled from $\{-1, 0, 1\}$. Let $C_k = \{j \in [n] \mid \varepsilon_j = k\}$ for $k \in \{-1, 0, 1\}$. For each index $i \in [n]$, we set the feature vector $\mathbf{X}_i \in \mathbb{R}^d$ as $\mathbf{X}_i \sim \mathcal{N}(\varepsilon_i \cdot \boldsymbol{\mu}, \mathbf{I} \cdot \sigma^2)$, where $\boldsymbol{\mu} \in \mathbb{R}^d$, $\sigma \in \mathbb{R}$ and $\mathbf{I} \in \{0, 1\}^{d \times d}$ is the identity matrix. For a given pair $p, q \in [0, 1]$ we consider the stochastic adjacency matrix $\mathbf{A} \in \{0, 1\}^{n \times n}$ defined as follows. For $i, j \in [n]$ in the same class, we set $a_{ij} \sim \mathrm{Ber}(p)$, and if i, j are in different classes, we set $a_{ij} \sim \mathrm{Ber}(q)$. We let $\mathbf{D} \in \mathbb{R}^{n \times n}$ be a diagonal matrix containing the degrees of the vertices. We denote by $(\mathbf{X}, \mathbf{A}) \sim \mathsf{CSBM}(n, p, q, \boldsymbol{\mu}, \sigma^2)$ a sample obtained according to the above random process.

The task we wish to solve is classifying C_0 vs $C_{-1} \cup C_1$. Namely, we want our model φ to satisfy $\varphi(\mathbf{X}_i) < 0$ if and only if $i \in C_0$. Moreover, note that the posed problem *is not linearly classifiable*.

To this end, we start by stating an assumption on the choice of parameters. This assumption is necessary to achieve degree concentration in the graph.

Assumption 1. $p, q = \Omega(\log^2 n/n)$.

We now show the distribution of the convolved features. The following lemma can be easily obtained using the techniques in [3].

Lemma 3. Fix p, q satisfying Assumption 1. With probability at least $1 - o(1)$ over $\mathbf{A}$ and $\{\varepsilon_i\}_i$,

$$(\mathbf{D}^{-1}\mathbf{A}\mathbf{X})_i \sim \mathcal{N}\left(\varepsilon_i \cdot \frac{p-q}{p+2q}\boldsymbol{\mu}, \frac{\sigma^2}{n(p+2q)}\right), \qquad \forall i \in [n].$$

To prove the above lemma, we need the following definition of our high probability event.

Definition 1. We define the even $\boldsymbol{\mathcal{E}}$ as the intersection of the following events over $\mathbf{A}$ and $\{\varepsilon_i\}_i$:

1. $\boldsymbol{\mathcal{E}}_1$ is the event that $|C_0| = \frac{n}{3} \pm O(\sqrt{n \log n})$, $|C_1| = \frac{n}{3} \pm O(\sqrt{n \log n})$ and $|C_{-1}| = \frac{n}{3} \pm O(\sqrt{n \log n})$.
2. $\boldsymbol{\mathcal{E}}_2$ is the event that for each $i \in [n]$, $\mathbf{D}_{ii} = \frac{n(p+2q)}{3}\left(1 \pm \frac{10}{\sqrt{\log n}}\right)$.
3. $\boldsymbol{\mathcal{E}}_3$ is the event that for each $i \in [n]$ and $k \in \{-1, 0, 1\}$,
$$|N_i \cap C_k| = \begin{cases} \mathbf{D}_{ii} \cdot \frac{p}{p+2q} \cdot \left(1 \pm \frac{10}{\sqrt{\log n}}\right) & \text{if } \ i \in C_k \\ \mathbf{D}_{ii} \cdot \frac{q}{p+2q} \cdot \left(1 \pm \frac{10}{\sqrt{\log n}}\right) & \text{if } \ i \notin C_k \end{cases}.$$

The following lemma is a direct application of Chernoff bound and a union bound.

Lemma 4. With probability at least $1 - 1/\mathrm{poly}(n)$ the event $\boldsymbol{\mathcal{E}}$ holds.

Proof of Lemma 3. By applying Lemma 4, and conditioned on $\boldsymbol{\mathcal{E}}$, for any $i \in [n]$

$$(\mathbf{D}^{-1}\mathbf{A}\mathbf{X})_i = \frac{1}{\mathbf{D}_{ii}} \sum_{j \in N_i} \mathbf{X}_j = \frac{1}{\mathbf{D}_{ii}} \left(\sum_{j \in N_i \cap C_{-1}} \mathbf{X}_j + \sum_{j \in N_i \cap C_0} \mathbf{X}_j + \sum_{j \in N_i \cap C_1} \mathbf{X}_j \right).$$

Using the definition of $\boldsymbol{\mathcal{E}}$ and properties of Gaussian distributions the lemma follows. □

Lemma 3 shows that essentially, the convolution reduced the variance and moved the means closer, but the structure of the problem stayed exactly the same. Therefore, one layer of GCN cannot separate C_0 from $C_{-1} \cup C_1$ with high probability.

A.2 A solution for GAT and CAT

In what follows, we show that GAT is able to handle the above classification task easily when the distance between the means is large enough. Then, we show how the additional convolution on the inputs to the score function improves the regime of perfect classification when the graph is not too noisy. Our main technical lemma considers a specific attention architecture and characterize the attention scores for our data model.

Lemma 5. Suppose that p, q satisfy Assumption 1, $\|\boldsymbol{\mu}\| \geq \omega\left(\sigma\sqrt{\log n}\right)$, fix the $\mathrm{LeakyRelu}$ constant $\beta \in (0,1)$ and $R \in \mathbb{R}$. Then, there exists a choice of attention architecture Ψ such that with probability at least $1 - o_n(1)$ over the data $(\mathbf{X}, \mathbf{A}) \sim \mathsf{CSBM}(n, p, q, \boldsymbol{\mu}, \sigma^2)$ the following holds.

$$\Psi(\mathbf{X}_i, \mathbf{X}_j) = \begin{cases} 10R\beta\|\boldsymbol{\mu}\|(1 \pm o(1)) & \text{if } i, j \in C_1^2 \\ -2R\|\boldsymbol{\mu}\|(1+2\beta)(1 \pm o(1)) & \text{if } i, j \in C_{-1}^2 \\ -2R\|\boldsymbol{\mu}\|(1+5\beta)(1 \pm o(1)) & \text{if } i \in C_1,\ j \in C_{-1} \\ 10R\beta\|\boldsymbol{\mu}\|(1 \pm o(1)) & \text{if } i \in C_{-1},\ j \in C_1 \\ -\frac{R}{2}\|\boldsymbol{\mu}\|(1-21\beta)(1 \pm o(1)) & \text{if } i \in C_0,\ j \in C_1 \\ -\frac{R}{2}\|\boldsymbol{\mu}\|(1-11\beta)(1 \pm o(1)) & \text{if } i \in C_0,\ j \in C_{-1} \\ -\frac{R}{2}\|\boldsymbol{\mu}\|(1-5\beta)(1 \pm o(1)) & \text{if } i \in C_1,\ j \in C_0 \\ -\frac{R}{2}\|\boldsymbol{\mu}\|(1-5\beta)(1 \pm o(1)) & \text{if } i \in C_{-1},\ j \in C_0 \\ 2R\beta\|\boldsymbol{\mu}\|(1 \pm o(1)) & \text{if } i, j \in C_0^2 \end{cases}.$$

Proof. We consider as an ansatz the following two layer architecture Ψ.

$$\tilde{\boldsymbol{w}} \stackrel{\text{def}}{=} \frac{\boldsymbol{\mu}}{\|\boldsymbol{\mu}\|}, \qquad \mathbf{S} \stackrel{\text{def}}{=} \begin{bmatrix} 1 & 1 \\ -1 & -1 \\ 1 & -1 \\ -1 & 1 \\ 0 & 1 \\ 1 & 0 \\ 0 & -1 \\ -1 & 0 \end{bmatrix}, \qquad \boldsymbol{b} \stackrel{\text{def}}{=} \begin{bmatrix} -3/2 \\ -3/2 \\ -3/2 \\ -3/2 \\ -1/2 \\ -1/2 \\ -1/2 \\ -1/2 \end{bmatrix} \cdot \|\boldsymbol{\mu}\|, \qquad \boldsymbol{r} \stackrel{\text{def}}{=} R \cdot \begin{bmatrix} 2 \\ -2 \\ -2 \\ 2 \\ -1 \\ -1 \\ -1 \\ -1 \end{bmatrix},$$

where $R > 0$ is an arbitrary scaling parameter. The output of the attention model is defined as

$$\Psi(\mathbf{X}_i, \mathbf{X}_j) \stackrel{\text{def}}{=} \boldsymbol{r}^T \cdot \mathrm{LeakyRelu}\left(\mathbf{S} \cdot \begin{bmatrix} \tilde{\boldsymbol{w}}^T \mathbf{X}_i \\ \tilde{\boldsymbol{w}}^T \mathbf{X}_j \end{bmatrix} + \boldsymbol{b}\right).$$

Let $\boldsymbol{\Delta}_{ij} \stackrel{\text{def}}{=} \mathbf{S} \cdot \begin{bmatrix} \tilde{\boldsymbol{w}}^T \mathbf{X}_i \\ \tilde{\boldsymbol{w}}^T \mathbf{X}_j \end{bmatrix} + \boldsymbol{b} \in \mathbb{R}^8$, and note that for each element $t \in [8]$ of $\boldsymbol{\Delta}_{ij}$, we have that $(\boldsymbol{\Delta}_{ij})_t = \mathbf{S}_{t,1}\tilde{\boldsymbol{w}}^T\mathbf{X}_i + \mathbf{S}_{t,2}\tilde{\boldsymbol{w}}^T\mathbf{X}_j + \boldsymbol{b}_t$. Note that the random variable $(\boldsymbol{\Delta}_{ij})_t$ is distributed as follows:

$$(\boldsymbol{\Delta}_{ij})_t \sim \begin{cases} \mathcal{N}\left((\mathbf{S}_{t,1} + \mathbf{S}_{t,2})\tilde{\boldsymbol{w}}^T\boldsymbol{\mu} + \boldsymbol{b}_t,\ \|\mathbf{S}_{t,*}\|^2\sigma^2\right) & \text{if } i, j \in C_1^2 \\ \mathcal{N}\left(-(\mathbf{S}_{t,1} + \mathbf{S}_{t,2})\tilde{\boldsymbol{w}}^T\boldsymbol{\mu} + \boldsymbol{b}_t,\ \|\mathbf{S}_{t,*}\|^2\sigma^2\right) & \text{if } i, j \in C_{-1}^2 \\ \mathcal{N}\left((\mathbf{S}_{t,1} - \mathbf{S}_{t,2})\tilde{\boldsymbol{w}}^T\boldsymbol{\mu} + \boldsymbol{b}_t,\ \|\mathbf{S}_{t,*}\|^2\sigma^2\right) & \text{if } i \in C_1,\ j \in C_{-1} \\ \mathcal{N}\left(-(\mathbf{S}_{t,1} - \mathbf{S}_{t,2})\tilde{\boldsymbol{w}}^T\boldsymbol{\mu} + \boldsymbol{b}_t,\ \|\mathbf{S}_{t,*}\|^2\sigma^2\right) & \text{if } i \in C_{-1},\ j \in C_1 \\ \mathcal{N}\left(\mathbf{S}_{t,2}\tilde{\boldsymbol{w}}^T\boldsymbol{\mu} + \boldsymbol{b}_t,\ \|\mathbf{S}_{t,*}\|^2\sigma^2\right) & \text{if } i \in C_0,\ j \in C_1 \\ \mathcal{N}\left(-\mathbf{S}_{t,2}\tilde{\boldsymbol{w}}^T\boldsymbol{\mu} + \boldsymbol{b}_t,\ \|\mathbf{S}_{t,*}\|^2\sigma^2\right) & \text{if } i \in C_0,\ j \in C_{-1} \\ \mathcal{N}\left(\mathbf{S}_{t,1}\tilde{\boldsymbol{w}}^T\boldsymbol{\mu} + \boldsymbol{b}_t,\ \|\mathbf{S}_{t,*}\|^2\sigma^2\right) & \text{if } i \in C_1,\ j \in C_0 \\ \mathcal{N}\left(-\mathbf{S}_{t,1}\tilde{\boldsymbol{w}}^T\boldsymbol{\mu} + \boldsymbol{b}_t,\ \|\mathbf{S}_{t,*}\|^2\sigma^2\right) & \text{if } i \in C_{-1},\ j \in C_0 \\ \mathcal{N}\left(\boldsymbol{b}_t,\ \|\mathbf{S}_{t,*}\|^2\sigma^2\right) & \text{if } i, j \in C_0^2 \end{cases}.$$

Therefore, for a fixed $i, j \in [n]^2$ we have that the entries of $\boldsymbol{\Delta}_{ij}$ are distributed as follows (where we use $\mathcal{N}_x^y$ as abbreviation for the Gaussian $\mathcal{N}(x, y)$)

$$\begin{bmatrix} \mathcal{N}^{4\sigma^2}_{\frac{\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{4\sigma^2}_{\frac{-7\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{4\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{4\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{\sigma^2}_{\frac{\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{\sigma^2}_{\frac{\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}} \end{bmatrix} \qquad \text{for } i, j \in C_1^2,$$

$$
\begin{array}{ll}
\begin{bmatrix} \mathcal{N}^{4\sigma^2}_{-\frac{7\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{4\sigma^2}_{\frac{\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{4\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{4\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{\sigma^2}_{\frac{\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{\sigma^2}_{\frac{\|\boldsymbol{\mu}\|}{2}} \end{bmatrix} & \text{for } i,j \in C_{-1}^2, \\
\begin{bmatrix} \mathcal{N}^{4\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{4\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{4\sigma^2}_{\frac{\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{4\sigma^2}_{-\frac{7\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{\sigma^2}_{\frac{\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{\sigma^2}_{\frac{\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}} \end{bmatrix} & \text{for } i,j \in C_1 \times C_{-1}, \\
\begin{bmatrix} \mathcal{N}^{4\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{4\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{4\sigma^2}_{-\frac{7\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{4\sigma^2}_{\frac{\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{\sigma^2}_{\frac{\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{\sigma^2}_{\frac{\|\boldsymbol{\mu}\|}{2}} \end{bmatrix} & \text{for } i,j \in C_{-1} \times C_1, \\
\begin{bmatrix} \mathcal{N}^{4\sigma^2}_{-\frac{\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{4\sigma^2}_{-\frac{5\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{4\sigma^2}_{-\frac{5\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{4\sigma^2}_{-\frac{\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{\sigma^2}_{\frac{\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{\sigma^2}_{-\frac{\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{\sigma^2}_{-\frac{\|\boldsymbol{\mu}\|}{2}} \end{bmatrix} & \text{for } i,j \in C_0 \times C_1, \\
\begin{bmatrix} \mathcal{N}^{4\sigma^2}_{-\frac{5\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{4\sigma^2}_{-\frac{\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{4\sigma^2}_{-\frac{\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{4\sigma^2}_{-\frac{5\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{\sigma^2}_{-\frac{\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{\sigma^2}_{\frac{\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{\sigma^2}_{-\frac{\|\boldsymbol{\mu}\|}{2}} \end{bmatrix} & \text{for } i,j \in C_0 \times C_{-1}, \\
\begin{bmatrix} \mathcal{N}^{4\sigma^2}_{-\frac{\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{4\sigma^2}_{-\frac{5\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{4\sigma^2}_{-\frac{\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{4\sigma^2}_{-\frac{5\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{\sigma^2}_{-\frac{\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{\sigma^2}_{\frac{\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{\sigma^2}_{-\frac{\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}} \end{bmatrix} & \text{for } i,j \in C_1 \times C_0, \\
\begin{bmatrix} \mathcal{N}^{4\sigma^2}_{-\frac{5\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{4\sigma^2}_{-\frac{\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{4\sigma^2}_{-\frac{5\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{4\sigma^2}_{-\frac{\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{\sigma^2}_{-\frac{\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{\sigma^2}_{-\frac{\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{\sigma^2}_{\frac{\|\boldsymbol{\mu}\|}{2}} \end{bmatrix} & \text{for } i,j \in C_{-1} \times C_0, \\
\begin{bmatrix} \mathcal{N}^{4\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{4\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{4\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{4\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{\sigma^2}_{-\frac{\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{\sigma^2}_{-\frac{\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{\sigma^2}_{-\frac{\|\boldsymbol{\mu}\|}{2}} & \mathcal{N}^{\sigma^2}_{-\frac{\|\boldsymbol{\mu}\|}{2}} \end{bmatrix} & \text{for } i,j \in C_0^2,
\end{array}
$$

Next, we will use the following lemma regarding $\mathrm{LeakyRelu}$ concentration.

Lemma 6 (Lemma A.6 in [14])**.** Fix $s \in \mathbb{N}$, and let $z_1, \dots, z_s$ be jointly Gaussian random variables with marginals $\boldsymbol{z}_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$. There exists an absolute constant $C > 0$ such that with probability at least $1 - o_s(1)$, we have

$$
\mathrm{LeakyRelu}(z_i) = \mathrm{LeakyRelu}\left(\mu_i\right) \pm C\sigma_i\sqrt{\log s}, \quad \text{for all } i \in [s].
$$

Using Lemma 6 with the assumption on $\|\boldsymbol{\mu}\|$ and a union bound, we have that with probability at least $1 - o_n(1)$, $\mathrm{LeakyRelu}(\boldsymbol{\Delta}_{ij})$ is (up to $1 \pm o(1)$)

$$
\begin{array}{ll}
\begin{bmatrix} \frac{\|\boldsymbol{\mu}\|}{2} & \frac{-7\beta\|\boldsymbol{\mu}\|}{2} & -\frac{3\beta\|\boldsymbol{\mu}\|}{2} & -\frac{3\beta\|\boldsymbol{\mu}\|}{2} & \frac{\|\boldsymbol{\mu}\|}{2} & \frac{\|\boldsymbol{\mu}\|}{2} & -\frac{3\beta\|\boldsymbol{\mu}\|}{2} & -\frac{3\beta\|\boldsymbol{\mu}\|}{2} \end{bmatrix} & \text{for } i,j \in C_1^2, \\
\begin{bmatrix} -\frac{7\beta\|\boldsymbol{\mu}\|}{2} & \frac{\|\boldsymbol{\mu}\|}{2} & -\frac{3\beta\|\boldsymbol{\mu}\|}{2} & -\frac{3\beta\|\boldsymbol{\mu}\|}{2} & -\frac{3\beta\|\boldsymbol{\mu}\|}{2} & -\frac{3\beta\|\boldsymbol{\mu}\|}{2} & \frac{\|\boldsymbol{\mu}\|}{2} & \frac{\|\boldsymbol{\mu}\|}{2} \end{bmatrix} & \text{for } i,j \in C_{-1}^2, \\
\begin{bmatrix} -\frac{3\beta\|\boldsymbol{\mu}\|}{2} & -\frac{3\beta\|\boldsymbol{\mu}\|}{2} & \frac{\|\boldsymbol{\mu}\|}{2} & -\frac{7\beta\|\boldsymbol{\mu}\|}{2} & -\frac{3\beta\|\boldsymbol{\mu}\|}{2} & \frac{\|\boldsymbol{\mu}\|}{2} & \frac{\|\boldsymbol{\mu}\|}{2} & -\frac{3\beta\|\boldsymbol{\mu}\|}{2} \end{bmatrix} & \text{for } i,j \in C_1 \times C_{-1}, \\
\begin{bmatrix} -\frac{3\beta\|\boldsymbol{\mu}\|}{2} & -\frac{3\beta\|\boldsymbol{\mu}\|}{2} & -\frac{7\beta\|\boldsymbol{\mu}\|}{2} & \frac{\|\boldsymbol{\mu}\|}{2} & \frac{\|\boldsymbol{\mu}\|}{2} & -\frac{3\beta\|\boldsymbol{\mu}\|}{2} & -\frac{3\beta\|\boldsymbol{\mu}\|}{2} & \frac{\|\boldsymbol{\mu}\|}{2} \end{bmatrix} & \text{for } i,j \in C_{-1} \times C_1, \\
\begin{bmatrix} -\frac{\beta\|\boldsymbol{\mu}\|}{2} & -\frac{5\beta\|\boldsymbol{\mu}\|}{2} & -\frac{5\beta\|\boldsymbol{\mu}\|}{2} & -\frac{\beta\|\boldsymbol{\mu}\|}{2} & \frac{\|\boldsymbol{\mu}\|}{2} & -\frac{\|\boldsymbol{\mu}\|}{2} & -\frac{3\beta\|\boldsymbol{\mu}\|}{2} & -\frac{\|\beta\boldsymbol{\mu}\|}{2} \end{bmatrix} & \text{for } i,j \in C_0 \times C_1, \\
\begin{bmatrix} -\frac{5\beta\|\boldsymbol{\mu}\|}{2} & -\frac{\beta\|\boldsymbol{\mu}\|}{2} & -\frac{\beta\|\boldsymbol{\mu}\|}{2} & -\frac{5\beta\|\boldsymbol{\mu}\|}{2} & -\frac{3\beta\|\boldsymbol{\mu}\|}{2} & -\frac{\beta\|\boldsymbol{\mu}\|}{2} & \frac{\|\boldsymbol{\mu}\|}{2} & -\frac{\beta\|\boldsymbol{\mu}\|}{2} \end{bmatrix} & \text{for } i,j \in C_0 \times C_{-1}, \\
\begin{bmatrix} -\frac{\beta\|\boldsymbol{\mu}\|}{2} & -\frac{5\beta\|\boldsymbol{\mu}\|}{2} & -\frac{\beta\|\boldsymbol{\mu}\|}{2} & -\frac{5\beta\|\boldsymbol{\mu}\|}{2} & -\frac{\beta\|\boldsymbol{\mu}\|}{2} & \frac{\|\boldsymbol{\mu}\|}{2} & -\frac{\beta\|\boldsymbol{\mu}\|}{2} & -\frac{3\beta\|\boldsymbol{\mu}\|}{2} \end{bmatrix} & \text{for } i,j \in C_1 \times C_0, \\
\begin{bmatrix} -\frac{5\beta\|\boldsymbol{\mu}\|}{2} & -\frac{\beta\|\boldsymbol{\mu}\|}{2} & -\frac{5\beta\|\boldsymbol{\mu}\|}{2} & -\frac{\beta\|\boldsymbol{\mu}\|}{2} & -\frac{\beta\|\boldsymbol{\mu}\|}{2} & -\frac{3\beta\|\boldsymbol{\mu}\|}{2} & -\frac{\|\beta\boldsymbol{\mu}\|}{2} & \frac{\|\boldsymbol{\mu}\|}{2} \end{bmatrix} & \text{for } i,j \in C_{-1} \times C_0, \\
\begin{bmatrix} -\frac{3\beta\|\boldsymbol{\mu}\|}{2} & -\frac{3\beta\|\boldsymbol{\mu}\|}{2} & -\frac{3\beta\|\boldsymbol{\mu}\|}{2} & -\frac{3\beta\|\boldsymbol{\mu}\|}{2} & -\frac{\beta\|\boldsymbol{\mu}\|}{2} & -\frac{\beta\|\boldsymbol{\mu}\|}{2} & -\frac{\beta\|\boldsymbol{\mu}\|}{2} & -\frac{\beta\|\boldsymbol{\mu}\|}{2} \end{bmatrix} & \text{for } i,j \in C_0^2.
\end{array}
$$

Then,

$$
\boldsymbol{r}^T \cdot \mathrm{LeakyRelu}(\boldsymbol{\Delta}_{ij}) = \begin{cases}
10R\beta\|\boldsymbol{\mu}\|(1 \pm o(1)) & \text{if } i,j \in C_1^2 \\
-2R\|\boldsymbol{\mu}\|(1+2\beta)(1 \pm o(1)) & \text{if } i,j \in C_{-1}^2 \\
-2R\|\boldsymbol{\mu}\|(1+5\beta)(1 \pm o(1)) & \text{if } i \in C_1,\ j \in C_{-1} \\
10R\beta\|\boldsymbol{\mu}\|(1 \pm o(1)) & \text{if } i \in C_{-1},\ j \in C_1 \\
-\frac{R}{2}\|\boldsymbol{\mu}\|(1-21\beta)(1 \pm o(1)) & \text{if } i \in C_0,\ j \in C_1 \\
-\frac{R}{2}\|\boldsymbol{\mu}\|(1-11\beta)(1 \pm o(1)) & \text{if } i \in C_0,\ j \in C_{-1} \\
-\frac{R}{2}\|\boldsymbol{\mu}\|(1-5\beta)(1 \pm o(1)) & \text{if } i \in C_1,\ j \in C_0 \\
-\frac{R}{2}\|\boldsymbol{\mu}\|(1-5\beta)(1 \pm o(1)) & \text{if } i \in C_{-1},\ j \in C_0 \\
2R\beta\|\boldsymbol{\mu}\|(1 \pm o(1)) & \text{if } i,j \in C_0^2
\end{cases},
$$

and the proof is complete. $\square$

Next we will define our high probability event.

Definition 2. $\boldsymbol{\mathcal{E}}' \stackrel{\text{def}}{=} \boldsymbol{\mathcal{E}} \cap \boldsymbol{\mathcal{E}}^*$, where $\boldsymbol{\mathcal{E}}^*$ is the event that for a fixed $\boldsymbol{w} \in \mathbb{R}^d$, all $i \in [n]$ satisfy $|\boldsymbol{w}^T \mathbf{X}_i - \mathbf{E}[\boldsymbol{w}^T \mathbf{X}_i]| \leq 10\sigma \|\boldsymbol{w}\|_2 \sqrt{\log n}$.

The following lemma is obtained by using Lemma 4 with standard Gaussian concentration and a union bound.

Lemma 7. With probability at least $1 - 1/\text{poly}(n)$ event $\boldsymbol{\mathcal{E}}'$ holds.

Corollary 8. Suppose that p, q satisfy Assumption 1, $\|\boldsymbol{\mu}\| = \omega(\sigma\sqrt{\log n})$ and fix $R \in \mathbb{R}$. Then, there exists a choice of attention architecture Ψ such that with probability $1 - o_n(1)$ over $(\mathbf{A}, \mathbf{X}) \sim \mathsf{CSBM}(n, p, q, \boldsymbol{\mu}, \sigma^2)$ it holds that

$$\gamma_{ij} = \begin{cases} \frac{3}{np}(1 \pm o(1)) & \text{if } i, j \in C_0^2 \cup C_1^2 \\ \frac{3}{nq}(1 \pm o(1)) & \text{if } i, j \in C_{-1} \times C_1 \\ \frac{3}{nq}\exp(-\Theta(R\|\boldsymbol{\mu}\|)) & \text{if } i, j \in C_{-1} \times C_{-1} \cup C_0 \\ \frac{3}{np}\exp(-\Theta(R\|\boldsymbol{\mu}\|)) & \text{otherwise} \end{cases},$$

where R is a parameter of the architecture.

Proof. The proof is immediate. First applying the ansatz from Lemma 5 with $\beta < 1/25$, Lemma 7 and a union bound. Using the definition of γ_{ij} concludes the proof. $\square$

Next, we prove Theorem 1 that the model distinguish nodes from C_0 for any choice of p, q satisfying Assumption 1. We restate the theorem for convince.

Theorem 9 (Formal restatement of Theorem 1)**.** Suppose that p, q satisfy Assumption 1 and $\|\boldsymbol{\mu}\|_2 = \omega(\sigma\sqrt{\log n})$. Then, there exists a choice of attention architecture Ψ such that with probability at least $1 - o_n(1)$ over the data $(\mathbf{X}, \mathbf{A}) \sim \mathsf{CSBM}(n, p, q, \boldsymbol{\mu}, \sigma^2)$, the estimator

$$\hat{x}_i \stackrel{\text{def}}{=} \sum_{j \in N_i} \gamma_{ij} \tilde{\boldsymbol{w}}^T \mathbf{X}_j + b \ \text{ where } \tilde{\boldsymbol{w}} = \boldsymbol{\mu}/\|\boldsymbol{\mu}\|, \ b = -\|\boldsymbol{\mu}\|/2$$

satisfies $\hat{x}_i < 0$ if and only if $i \in C_0$.

Proof. Let Ψ be the architecture from Corollary 8 and let R satisfy $R\|\boldsymbol{\mu}\|_2 = \omega(1)$. We will compute the mean and variance of the estimator $\hat{x}_i$ conditioned on $\boldsymbol{\mathcal{E}}'$. Suppose that $i \in C_0$. By using Corollary 8, Definition 2 and our assumption on $\|\boldsymbol{\mu}\|$ and R, we have

$$\max\left\{\frac{3}{np}\exp(-\Theta(R\|\boldsymbol{\mu}\|)), \frac{3}{nq}\exp(-\Theta(R\|\boldsymbol{\mu}\|))\right\} = o\left(\frac{1}{n(p+2q)}\right),$$

and therefore

$$\begin{aligned}
\mathbf{E}\left[\hat{x}_i \mid \boldsymbol{\mathcal{E}}'\right] &= \mathbf{E}\left[\sum_{k \in \{-1,0,1\}} \sum_{j \in N_i \cap C_k} \gamma_{ij} \tilde{\boldsymbol{w}}^T \mathbf{X}_j \ \mid \boldsymbol{\mathcal{E}}'\right] - \frac{\|\boldsymbol{\mu}\|}{2} \\
&= \mathbf{E}[|C_0 \cap N_i| \mid \boldsymbol{\mathcal{E}}'] \left(\pm \frac{3}{np}(1 \pm o(1)) \cdot 10\sigma\sqrt{\log n}\right) \\
&\quad + \mathbf{E}[|C_1 \cap N_i| \mid \boldsymbol{\mathcal{E}}'] \left(o\left(\frac{1}{n(p+2q)}\right) \cdot (\|\boldsymbol{\mu}\| \pm 10\sigma\sqrt{\log n})\right) \\
&\quad + \mathbf{E}[|C_{-1} \cap N_i| \mid \boldsymbol{\mathcal{E}}'] \left(o\left(\frac{1}{n(p+2q)}\right) \cdot (-\|\boldsymbol{\mu}\| \pm 10\sigma\sqrt{\log n})\right) - \frac{\|\boldsymbol{\mu}\|}{2} \\
&= -\frac{\|\boldsymbol{\mu}\|}{2}(1 \pm o(1)).
\end{aligned}$$

By similar reasoning we have that for $i \in C_{-1} \cup C_1$, $\mathbf{E}\left[\hat{x}_i \mid \boldsymbol{\mathcal{E}}'\right] = \frac{\|\boldsymbol{\mu}\|}{2}(1 \pm o(1))$.

Next, we claim that for each $i \in [n]$ the random variable $\hat{x}_i$ given $\boldsymbol{\mathcal{E}}'$ is sub-Gaussian with a small sub-Gaussian constant compared to the above expectation. The following lemma is a straightforward adaptation of Lemma A.11 in [14], and we provide its proof for completeness.

Lemma 10. Conditioned on $\boldsymbol{\mathcal{E}}'$, the random variables $\{\hat{\boldsymbol{x}}_i\}_i$ are sub-Gaussian with parameter $\tilde{\sigma}_i^2 = O\left(\frac{\sigma^2}{np}\right)$ if $i \in C_0 \cup C_1$ and $\tilde{\sigma}_i^2 = O\left(\frac{\sigma^2}{nq}\right)$ otherwise.

Proof. Fix $i \in [n]$, and write $\mathbf{X}_i = \varepsilon_i \boldsymbol{\mu} + \sigma \boldsymbol{g}_i$ where $\boldsymbol{g}_i \sim \mathcal{N}(0, \mathbf{I}_d)$, and ε_i denotes the class membership. Consider $\hat{x}_i$ as a function of $\boldsymbol{g} = [\boldsymbol{g}_1 \circ \boldsymbol{g}_2 \circ \cdots \circ \boldsymbol{g}_n] \in \mathbb{R}^{nd}$, where $\circ$ denotes vertical concatenation. Namely, consider the function

$$\hat{x}_i = f_i(\boldsymbol{g}) \stackrel{\text{def}}{=} \sum_{j \in N_i} \gamma_{ij}(\boldsymbol{g})\, \tilde{\boldsymbol{w}}^T(\varepsilon_j \boldsymbol{\mu} + \sigma \boldsymbol{g}_j) - \|\boldsymbol{\mu}\|/2, \quad i \in [n].$$

Since $\boldsymbol{g} \sim \mathcal{N}(0, \mathbf{I}_{nd})$, proving that $\hat{x}_i$ given $\boldsymbol{\mathcal{E}}'$ is sub-Gaussian for each $i \in [n]$, reduces to showing that the function $f_i : \mathbb{R}^{nd} \to \mathbb{R}$ is Lipschitz over $E \subseteq \mathbb{R}^{nd}$ defined by $\boldsymbol{\mathcal{E}}'$ and the relation $\mathbf{X}_i = \varepsilon_i \boldsymbol{\mu} + \sigma \boldsymbol{g}_i$. That is, $E \stackrel{\text{def}}{=} \left\{\boldsymbol{g} \in \mathbb{R}^{nd} \;\middle|\; |\tilde{\boldsymbol{w}}^T \boldsymbol{g}_i| \le 10\sqrt{\log n}, \forall i \in [n]\right\}$. Specifically, we show that conditioning on the event $\boldsymbol{\mathcal{E}}'$ (which restricts $\boldsymbol{g} \in E$), the Lipschitz constant L_{f_i} of f_i satisfies $L_{f_i} = O\left(\frac{\sigma}{\sqrt{np}}\right)$ for $i \in C_0 \cup C_1$ and $L_{f_i} = O\left(\frac{\sigma}{nq}\right)$ otherwise, and hence proving the claim.

To compute the Lipschitz constant of $f_i(\boldsymbol{g})$ for $i \in [n]$, let us denote $\mathbf{X} = [\mathbf{X}_1 \circ \mathbf{X}_2 \circ \cdots \circ \mathbf{X}_n]$ and consider the function

$$\tilde{f}_i(\mathbf{X}) \stackrel{\text{def}}{=} \sum_{j \in N_i} \gamma_{ij}(\mathbf{X})\, \tilde{\boldsymbol{w}}^T \mathbf{X}_j, \quad i \in [n]$$

Let us assume without loss of generality that $i \in C_0$ (the cases for $i \in C_1$ and $i \in C_{-1}$ are obtained identically). Conditioning on the event $\boldsymbol{\mathcal{E}}'$, which imposes the restriction that $\mathbf{X} \in \tilde{E}$ where

$$\tilde{E} \stackrel{\text{def}}{=} \left\{\mathbf{X} \in \mathbb{R}^{nd} \;\middle|\; |\mathbf{X}_i - \varepsilon_i \boldsymbol{\mu}| \le 10\sigma\sqrt{\log n}, \forall i \in [n]\right\}.$$

Conditioning on $\boldsymbol{\mathcal{E}}'$ (which restricts $\mathbf{X}, \mathbf{X}' \in \tilde{E}$), using Corollary 8 and recalling that R satisfies $R\|\boldsymbol{\mu}\|_2 = \omega(1)$, we get[5]

$$\begin{aligned}
&\left|\tilde{f}_i(\mathbf{X}) - \tilde{f}_i(\mathbf{X}')\right| \\
&\simeq \left|\sum_{j \in N_i \cap C_0} \frac{3}{np} \tilde{\boldsymbol{w}}^T(\mathbf{X}_j - \mathbf{X}_j') + \sum_{j \in N_i \cap C_1} \frac{3}{np} \cdot e^{-\Theta(R\|\boldsymbol{\mu}\|_2)} \tilde{\boldsymbol{w}}^T(\mathbf{X}_j - \mathbf{X}_j') + \sum_{j \in N_i \cap C_{-1}} \frac{3}{np} \cdot e^{-\Theta(R\|\boldsymbol{\mu}\|_2)} \tilde{\boldsymbol{w}}^T(\mathbf{X}_j - \mathbf{X}_j')\right| \\
&= \left|\begin{bmatrix} \frac{3}{np}(1 \pm o(1))\tilde{\boldsymbol{w}} & \text{if } j \in N_i \cap C_0 \\ \frac{3}{np}\exp(-\Theta(R\|\boldsymbol{\mu}\|_2))(1 \pm o(1))\tilde{\boldsymbol{w}} & \text{if } j \in N_i \cap C_1 \\ \frac{3}{np}\exp(-\Theta(R\|\boldsymbol{\mu}\|_2))(1 \pm o(1))\tilde{\boldsymbol{w}} & \text{if } j \in N_i \cap C_{-1} \\ 0 & \text{if } j \notin N_i \end{bmatrix}_{j \in [n]}^T (\mathbf{X} - \mathbf{X}')\right| \\
&\le \left\|\begin{bmatrix} \frac{3}{np}(1 \pm o(1))\tilde{\boldsymbol{w}} & \text{if } j \in N_i \cap C_0 \\ \frac{3}{np}\exp(-\Theta(R\|\boldsymbol{\mu}\|_2))(1 \pm o(1))\tilde{\boldsymbol{w}} & \text{if } j \in N_i \cap C_1 \\ \frac{3}{np}\exp(-\Theta(R\|\boldsymbol{\mu}\|_2))(1 \pm o(1))\tilde{\boldsymbol{w}} & \text{if } j \in N_i \cap C_{-1} \\ 0 & \text{if } j \notin N_i \end{bmatrix}_{j \in [n]}\right\|_2 \|\mathbf{X} - \mathbf{X}'\|_2 \\
&\le \sqrt{\frac{3}{np}}(1 + o(1))\|\tilde{\boldsymbol{w}}\|_2 \|\mathbf{X} - \mathbf{X}'\|_2 \\
&= \sqrt{\frac{3}{np}}(1 + o(1))\, \|\mathbf{X} - \mathbf{X}'\|_2\,.
\end{aligned}$$

[5]We drop the $(1 \pm o(1))$ in the first line of the computation for compactness and use $\simeq$ as notation.

This shows the Lipschitz constant of $\tilde{f}_i(\mathbf{X})$ over $\tilde{E}$ satisfies $L_{\tilde{f}_i} = O\left(\frac{1}{\sqrt{np}}\right)$. On the other hand, by viewing $\mathbf{X}$ as a function of $\boldsymbol{g}$, it is straightforward to see that the function $h(\boldsymbol{g}) : \mathbb{R}^{nd} \to \mathbb{R}^{nd}$ defined by $h(\boldsymbol{g}) \stackrel{\text{def}}{=} \mathbf{X}(\boldsymbol{g})$ has Lipschitz constant $L_h = \sigma$, as

$$\|h(\boldsymbol{g}) - h(\boldsymbol{g}')\|_2 = \|\boldsymbol{\varepsilon}\boldsymbol{\mu} + \sigma\boldsymbol{g} - (\boldsymbol{\varepsilon}\boldsymbol{\mu} + \sigma\boldsymbol{g}')\|_2 = \sigma\|\boldsymbol{g} - \boldsymbol{g}'\|_2.$$

Therefore, since $f_i(\boldsymbol{g}) = \tilde{f}_i(h(\boldsymbol{g}))$ and $\boldsymbol{g} \in E$ if and only if $\mathbf{X} \in \tilde{E}$, we have that, conditioning on $\boldsymbol{\mathcal{E}}'$, the function $\hat{x}_i = f_i(\boldsymbol{g})$ is Lipschitz continuous with Lipschitz constant $L_{f_i} = L_{\tilde{f}_i} L_h = O\left(\frac{\sigma}{\sqrt{np}}\right)$. Since $\boldsymbol{g} \sim \mathcal{N}(0, \mathbf{I}_{nd})$, we know that $\hat{x}_i$ is sub-Gaussian with sub-Gaussian constant $\tilde{\sigma}^2 = L_{f_i}^2 = O\left(\frac{\sigma^2}{np}\right)$. □

The following lemma will be used for bounding the misclassification probability.

Lemma 11 ([30]). Let $x_1, \ldots, x_n$ be sub-Gaussian random variables with the same mean and sub-Gaussian parameter $\tilde{\sigma}^2$. Then,

$$\mathbf{E}\left[\max_{i\in[n]} (x_i - \mathbf{E}[x_i])\right] \le \tilde{\sigma}\sqrt{2\log n}.$$

Moreover, for any $t > 0$

$$\mathbf{Pr}\left[\max_{i\in[n]} (x_i - \mathbf{E}[x_i]) > t\right] \le 2n \exp\left(-\frac{t^2}{2\tilde{\sigma}^2}\right).$$

We bound the probability of misclassification

$$\mathbf{Pr}\left[\max_{i\in C_0} \hat{x}_i \ge 0\right] \le \mathbf{Pr}\left[\max_{i\in C_0} \hat{x}_i > t + \mathbf{E}[\hat{x}_i]\right],$$

for $t < |\mathbf{E}[\hat{x}_i]| = \frac{\|\boldsymbol{\mu}\|_2}{2}(1 \pm o(1))$. By Lemma 10, picking $t = \Theta\left(\sigma\sqrt{\log|C_0|}\right)$ and applying Lemma 11 implies that the above probability is $1/\mathrm{poly}(n)$.

Similarly for class $C_1 \cup C_{-1}$ we have that the misclassification probability is

$$\begin{aligned}\mathbf{Pr}\left[\min_{i\in C_1\cup C_{-1}} \hat{x}_i \le 0\right] &= \mathbf{Pr}\left[-\max_{i\in C_1\cup C_{-1}} (-\hat{x}_i) \le 0\right] = \mathbf{Pr}\left[\max_{i\in C_1\cup C_{-1}} (-\hat{x}_i) \ge 0\right] \\ &\le \mathbf{Pr}\left[\max_{i\in C_1\cup C_{-1}} -\hat{x}_i > t - \mathbf{E}[\hat{x}_i]\right],\end{aligned}$$

for $t < \mathbf{E}[\hat{x}_i]$. Picking $t = \Theta\left(\sigma\sqrt{\log|C_1 \cup C_{-1}|}\right)$ and applying Lemma 11 and a union bound over the misclassification probabilities of both classes conclude the proof of the corollary. □

Combining Theorem 9 with Lemma 3, we immediately get Corollary 2 which we restate below.

Corollary 12. Suppose $p, q = \Omega(\log^2 n/n)$ and $\|\boldsymbol{\mu}\| \ge \omega\left(\sigma\sqrt{\frac{(p+2q)\log n}{n(p-q)^2}}\right)$. Then, there is a choice of attention architecture Ψ such that, with probability at least $1 - o(1)$ over the data $(\mathbf{X}, \mathbf{A}) \sim \mathsf{CSBM}(n, p, q, \boldsymbol{\mu}, \sigma^2)$, CAT separates nodes C_0 from $C_1 \cup C_{-1}$.

B Synthetic experiments

In this section, we present the complete results for the synthetic data experiments of §6.1. First, we describe the parameterization we use for the 1-layer GCN, GAT, and CAT models; then, we verify the behavior of the normalized score function (γ_{ij}) matches that of the theory presented in Corollary 8. In particular, we visualize the average of the following three groups of gammas (Fig. 4):

- Gammas γ_{ij} included in $i, j \in C_0^2 \cup C_1^2$. Solid lines.
- Gammas γ_{ij} included in $i, j \in C_{-1} \times C_1$. Dashed lines.
- The rest of gammas. Dotted lines.

For completeness, we also include the empirical results that validate Theorem 1 and Corollary 2, which were discussed already in §6.1.

Experimental setup. We assume the following parametrization for the 1-layer GCN, GAT, and CAT:

$$\boldsymbol{h}'_i = \left(\sum_{j \in N_i} \gamma_{ij} \tilde{\boldsymbol{w}}^T \mathbf{X}_j \right) - C \cdot \|\boldsymbol{\mu}\|/2 \,, \tag{8}$$

where N_i are the set of neighbors of node i, $\mathbf{X}_j$ are the features of node j—obtained from the CSBM described in §4, and $\boldsymbol{h}'_i$ are the logits of the prediction of node i. Note that for GCN we have $\gamma_{ij} = \frac{1}{|N_i^*|}$. Otherwise, we consider the following parameterization of the score function Ψ:

$$\gamma_{ij} = \frac{\exp\left(\Psi(\boldsymbol{h}_i, \boldsymbol{h}_j)\right)}{\sum_{k \in N_i^*} \exp(\Psi(\boldsymbol{h}_i, \boldsymbol{h}_k))} \quad \text{where} \tag{9}$$

$$\Psi(\boldsymbol{h}_i, \boldsymbol{h}_j) \stackrel{\text{def}}{=} \boldsymbol{r}^T \cdot \text{LeakyRelu}\left(\mathbf{S} \cdot \begin{bmatrix} \tilde{\boldsymbol{w}}^T \boldsymbol{h}_i \\ \tilde{\boldsymbol{w}}^T \boldsymbol{h}_j \end{bmatrix} + \boldsymbol{b} \right) \,. \tag{10}$$

For these experiments, we define the parameters $\tilde{\boldsymbol{w}}$, $\mathbf{S}$, $\boldsymbol{b}$ and $\boldsymbol{r}$ as in the proofs in Appendix A:

$$\tilde{\boldsymbol{w}} \stackrel{\text{def}}{=} \frac{\boldsymbol{\mu}}{\|\boldsymbol{\mu}\|}, \qquad \mathbf{S} \stackrel{\text{def}}{=} \begin{bmatrix} 1 & 1 \\ -1 & -1 \\ 1 & -1 \\ -1 & 1 \\ 0 & 1 \\ 1 & 0 \\ 0 & -1 \\ -1 & 0 \end{bmatrix}, \qquad \boldsymbol{b} \stackrel{\text{def}}{=} \begin{bmatrix} -3/2 \\ -3/2 \\ -3/2 \\ -3/2 \\ -1/2 \\ -1/2 \\ -1/2 \\ -1/2 \end{bmatrix} \cdot \|\boldsymbol{\mu}\| \cdot C, \qquad \boldsymbol{r} \stackrel{\text{def}}{=} R \cdot \begin{bmatrix} 2 \\ -2 \\ -2 \\ 2 \\ -1 \\ -1 \\ -1 \\ -1 \end{bmatrix}, \tag{11}$$

where $R > 0$ and $C > 0$ are arbitrary scaling parameters. Both C and R and input to the score function are set different for each of the models, as indicated in Table 4. In particular, we set $R = \frac{7}{\|\boldsymbol{\mu}\|}$ for both GAT and CAT such that: i) all γ_{ij} are distinguishable as we decrease $\|\boldsymbol{\mu}\|$; and ii) we avoid numerical instabilities in the implementation when computing the exponential of $R \times \|\boldsymbol{\mu}\|$ in order to obtain γ_{ij} (see Corollary 8), as the exponential of small or large values leads to under/overflow issues. As for C, we set $C = 1$ for GAT and $C = (p-q)/(p+2q)$ for CAT such that we counteract the fact that the distance between classes shrink as we increase q, see Lemma 3:

Table 4: Parameters for the synthetic experiments.

Model	C	R	$\boldsymbol{h}_i$
GCN	0	–	–
GAT	1	$\frac{7}{\|\boldsymbol{\mu}\|}$	$\mathbf{X}_i$
CAT	$\frac{p-q}{p+2q}$	$\frac{7}{\|\boldsymbol{\mu}\|}$	$\frac{1}{\|N_i^*\|} \sum_{k \in N_i^*} \mathbf{X}_k$

Regarding the data model, we set (as described in §6.1) $n = 10000$, $p = 0.5$, $\sigma = 0.1$, and $d = n/\left(5 \log^2(n)\right)$. We set the slope of the LeakyReLU activation to $\beta = 1/5$ for the GAT and

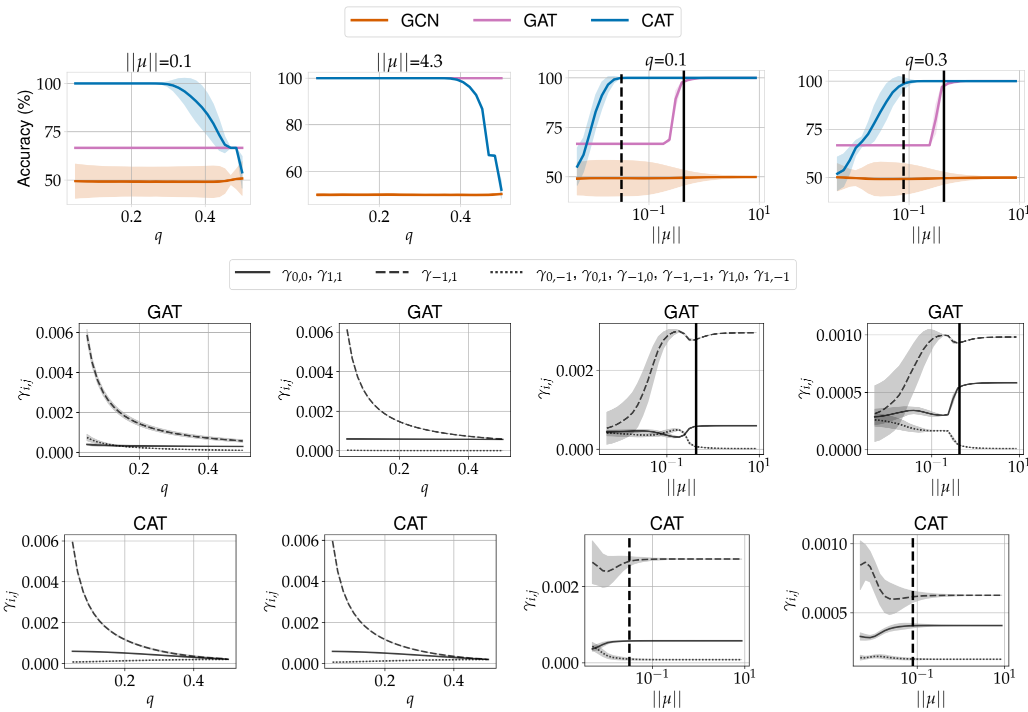

Figure 4: Synthetic data results. On the top row, we show the node classification, and in the following two rows we show the γ_{ij} values for GAT and CAT respectively. In the two left-most figures, we show how the results vary with the noise level q for $\|\boldsymbol{\mu}\| = 0.1$ and $\|\boldsymbol{\mu}\| = 4.3$. In the two right-most figures, we show how the results vary with the norm of the means $\|\boldsymbol{\mu}\|$ for $q = 0.1$ and $q = 0.3$. We use two vertical lines to present the classification threshold stated in Theorem 1 (solid line) and Corollary 2 (dashed line).

$\beta = 0.01$ for CAT, such that the proof of Corollary 8 is valid. As described in the main paper, to assess the sensitivity to structural noise, we present the complete results for two sets of experiments. First, we vary the noise level q between 0 and 0.5, fixing the mean vector $\boldsymbol{\mu}$. We test two values of $\|\boldsymbol{\mu}\|$: the first corresponds to the *easy* regime ($\|\boldsymbol{\mu}\| = 10\sigma\sqrt{2\log n} \approx 4.3$) where classes are far apart; and the second correspond to the *hard* regime ($\|\boldsymbol{\mu}\| = \sigma = 0.1$) where the distance between the clusters is small. In the second experiment we modify instead the distance between the means, sweeping $\|\boldsymbol{\mu}\|$ in the range $\left[\sigma/20, 20\sigma\sqrt{2\log n}\right]$ which corresponds to $[0.005, 8.58]$, and thus covering the transition from the hard setting (small $\|\boldsymbol{\mu}\|$) to the easy one (large $\|\boldsymbol{\mu}\|$). In these experiments, we fix q to 0.1 (low noise) and 0.3 (high noise).

Results are summarized in Fig. 4. The top row contains the node classification performance for each of the models (i.e., Fig. 1), the next two rows contain the γ_{ij} values for GAT and CAT respectively. The two left-most columns of Fig. 4 show the results for the hard and easy regimes, respectively, as we vary the noise level q. In the hard regime, we observe that GAT is unable to achieve separation for any value of q, whereas CAT achieves perfect classification when q is small enough. The gamma plots help shed some light on this question. For GAT, we observe that the gammas represented with the dotted and solid lines collapse for any value of q (see middle plot), while this does not happen for CAT when the noise level is low (see bottom plot). This exemplifies the advantage of CAT over GAT as stated in Corollary 2. When the distance between the means is large enough, we see that GAT achieves perfect results independently of q, as stated in Theorem 1. We also observe that, in this case, the gammas represented with the dotted and solid lines do not collapse for any value of q. In contrast, as we increase q, CAT fails to satisfy the condition in Corollary 2, and therefore achieves inferior performance. We note that the low performance is due to the fact that all gammas collapse to the same value for large noise levels.

For the second set of experiments (two right-most columns of Fig. 1), where we fix q and sweep $\|\boldsymbol{\mu}\|$, we observe that, for both values of q, there exists a transition in the accuracy of both GAT

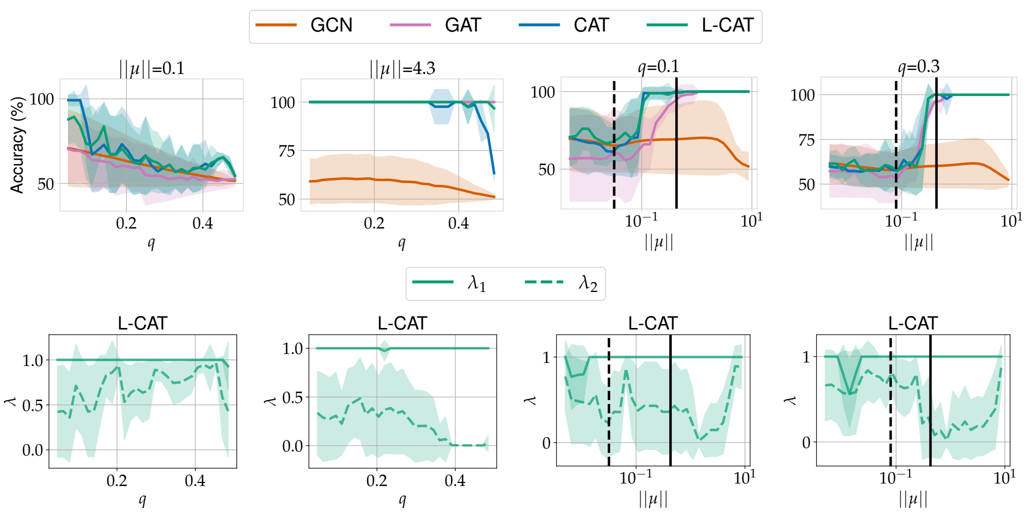

Figure 5: Synthetic data results learning C, λ_1 and λ_2. On the top row, we show the node classification accuracy, and in the bottom row we show the learned values of λ_1 and λ_2 for L-CAT. In the two left-most figures, we show how the results vary with the noise level q for $\|\boldsymbol{\mu}\| = 0.1$ and $\|\boldsymbol{\mu}\| = 4.3$. In the two right-most figures, we show how the results vary with the norm of the means $\|\boldsymbol{\mu}\|$ for $q = 0.1$ and $q = 0.3$. We use two vertical lines to present the classification threshold stated in Theorem 1 (solid line) and Corollary 2 (dashed line).

and CAT as a function of $\|\boldsymbol{\mu}\|$. As shown in the main manuscript, GAT achieves perfect accuracy when the distance between the means satisfies the condition in Theorem 1 (solid vertical line in Fig. 1). Moreover, we can see the improvement CAT obtains over GAT. Indeed, when $\|\boldsymbol{\mu}\|$ satisfies the conditions of Corollary 2 (dashed vertical line in Fig. 1), the classification threshold is improved. As we increase q, we see that the gap between the two vertical lines decreases, which means that the improvement decreases as q increments, exactly as stated in Corollary 2. This transition from the hard regime to the easy regime is also observed in the gamma plots: we observe the largest difference in value between the different groups of lambdas for values of $\|\boldsymbol{\mu}\|$ that satisfy the condition in Theorem 1 (that is to the right of the vertical lines).

B.1 Other experiments

In the following, we extend the results for the synthetic data presented above. In particular, we aim to evaluate if L-CAT is able to achieve top performance regardless of the scenario. That is, we want to evaluate if L-CAT consistently performs at least as good as the best-performing model. We change the fixed-parameter setting of the previous section and, instead, we evaluate the performance of GCN, GAT, CAT and L-CAT when we learn the model-dependent parameters.

Experimental setup. We assume the same parametrization for the 1-layer GCN, GAT and CAT described in Eq. 8 and Eq. 10. For L-CAT, we add the parameters λ_1 and λ_2, as indicated in Eq. 7. We fix the parameters shared among the models, that is, $\tilde{\boldsymbol{w}}$, $\mathbf{S}$, $\boldsymbol{b}$, $\boldsymbol{r}$, and R, with the values indicated in Eq. 11. Different from previous experiments, we now learn C and, for L-CAT, we also learn λ_1 and λ_2. We choose to fix part of the parameters (instead of learning them all) to keep the problem as similar as possible to the theoretical analysis we provided in §4 and Appendix A. If we instead learn all the parameters, it takes a single dimension of the features to be (close to) linearly separable to find a solution that achieves a similar performance regardless of the model, which hinders the analysis. This is a consequence of the probabilistic nature of the features. One way of solving this issue would be to make n big enough. Instead, we opt to have a fixed n and reduce the degrees of freedom of the models by fixing the parameters shared across all models. The rest of the experimental setup matches the one from Appendix B. Additionally, we use the Adam optimizer [24] with a learning rate of 0.05, and we train for 100.

Results are summarized in Fig. 5. The top row contains the node classification performance for every model, while the bottom row contains the learned values of λ_1 (solid line) and λ_2 (dashed line) with L-CAT. The two left-most columns of Fig. 5 show the results for the hard and easy regimes, respectively, as we vary the noise level q. In the hard regime, we see rather noisy results. Still, the behaviour is similar to that of Fig. 4: the performance of CAT degrades as we increase q. We also observe that, on average, CAT outperforms GAT. In this case, we observe that L-CAT achieves similar performance as CAT, which can be explained by inspecting the learned values of lambda in the bottom row. We observe that $\lambda_1 = 1$ and $\lambda_2 \geq 0.5$ on average for all values of q. This indicates that L-CAT is closer to CAT than to GAT. When the distance between the means is large enough (i.e., $\|\boldsymbol{\mu}\| = 4.3$), we see that GAT achieves perfect results independently of q while the performance of CAT deteriorates with large values of q, the same trend as in Fig. 4. Remarkably, we observe that L-CAT also achieves perfect results independently of q. If we inspect the lambda values, we first see that $\lambda = 1$ for all q, thus the interpolation happens between CAT and GAT. Looking at the values of λ_2, we observe that, for small values of q, λ_2 is pretty noisy, which is expected since any solution achieves perfect performance. Interestingly, we have that $\lambda_2 = 0$ for large values of q, with negligible variance. This indicates that L-CAT learns that it must behave like GAT in order to perform well.

For the second set of experiments (two right-most columns of Fig. 5), we fix q and sweep $\|\boldsymbol{\mu}\|$ like we did in Fig. 4. Here, we observe a similar trend: for both values of q, there exists a transition in the accuracy of both GAT and CAT as a function of $\|\boldsymbol{\mu}\|$. Yet once again, we observe that L-CAT consistently achieves a similar performance to the best-performing model in every situation.

C Dataset description

We present further details about the datasets used in our experiments, summarized in Table 5. All datasets are undirected (or transformed to undirected otherwise) and transductive.

The upper rows of the table refer to datasets used in §6.2 taken from the PyTorch Geometric framework.[6] The following paragraphs present a short description of such datasets.

Amazon Computers & Photos are datasets taken from [34], in which nodes represent products, and edges indicate that the products are usually bought together. The node features are a Bag of Words (BoW) representation of the product reviews. The task is to predict the category of the products.

GitHub is a dataset introduced in [31], in which nodes correspond to developers, and edges indicate mutual follow relationship. Node features are embeddings extracted from the developer's starred repositories and profile information (e.g., location or employer). The task is to infer whether a node relates to web or machine learning development.

FacebookPagePage is a dataset introduced in [31], where nodes are Facebook pages, and edges imply mutual likes between the pages. Nodes features are text embeddings extracted from the pages' description. The task consist on identifying the page's category.

TwitchEN is a dataset introduced in [31]. Here, nodes correspond to Twitch gamers, and links reveal mutual friendship. Node features are an embedding of games liked, location, and streaming habits. The task is to infer if a gamer uses explicit content.

Coauthor Physics & CS are datasets introduced in [34]. In this case, nodes represent authors which are connected with an edge if they have co-authored a paper. Node features are BoW representations of the keywords of the author's papers. The task consist on mapping each author to their corresponding field of study.

DBLP is a dataset introduced in [7] that represents a citation network. In this dataset, nodes represent papers and edges correspond to citations. Node features are BoW representations of the keywords of the papers. The task is to predict the research area of the papers.

PubMed, Cora & CiteSeer are citation networks introduced in [46]. Nodes represent documents, and edges refer to citations between the documents. Node features are BoW representations of the documents. The task is to infer the topic of the documents.

The bottom rows of Table 5 refer to the datasets from Open Graph Benchmark (OGB) [20] [7] used in §6.3. We include a short description of them in the paragraphs below.

[6] https://pytorch-geometric.readthedocs.io/en/latest/modules/datasets.html
[7] https://ogb.stanford.edu/docs/nodeprop

ogbn-arxiv is a citation network of computer science papers in arXiv [40]. Nodes represent papers, and directed edges refer to citations among them. Node features are embeddings of the title and abstract of the papers. The task is to predict the research area of the nodes.

ogbn-products contains a co-purchasing network [6]. Nodes represent products, and links are present whenever two products are bought together. Node features are embeddings of a BoW representation of the product description. The task is to infer the category of the products.

ogbn-mag is a heterogeneous network formed from a subgraph of the Microsoft Academic Graph (MAG) [40]. Nodes can belong to one of these four types: authors, papers, institutions and fields of study. Moreover, directed edges belong to one of the following categories: "author is affiliated with an institution," "author has written a paper," "paper cites a paper," and "paper belongs to a research area." Only nodes that are papers contain node features, which are a text embedding of the document content. The task is to predict the venue of the nodes that are papers.

ogbn-proteins is a network whose nodes represent proteins and edges indicate different types of associations among them. This dataset does not contain node features. The tasks are to predict multiple protein functions, each of them being a binary classification problem.

Table 5: Dataset statistics. On the top part of the table, we show the datasets used in §6.2. On the bottom part of the table, we show the datasets used in §6.3.

Name	#Nodes	#Edges	Avg. degree	#Node feats.	#Edge feats.	#Tasks	Task Type
AmazonComp.	13,752	491,722	35.76	767	-	1	10-class clf.
AmazonPhoto	7,650	238,162	31.13	745	-	1	8-class clf.
GitHub	37,700	578,006	15.33	128	-	1	Binary clf.
FacebookP.	22,470	342,004	15.22	128	-	1	4-class clf.
CoauthorPh.	34,493	495,924	14.38	8415	-	1	5-class clf.
TwitchEN	7,126	77,774	10.91	128	-	1	Binary clf.
CoauthorCS	18,333	163,788	8.93	6805	-	1	15-class clf.
DBLP	17,716	105,734	5.97	1639	-	1	4-class clf.
PubMed	19,717	88,648	4.50	500	-	1	3-class clf.
Cora	2,708	10,556	3.90	1433	-	1	7-class clf.
CiteSeer	3,327	9,104	2.74	3703	-	1	6-class clf.
ogbn-arxiv	169,343	1,166,243	6.89	128	-	1	40-class clf.
ogbn-products	2,449,029	123,718,280	50.52	100	-	1	47-class clf.
ogbn-mag	1,939,743	21,111,007	18.61	128	4	1	349-class clf.
ogbn-proteins	132,534	79,122,504	597.00	-	8	112	Multi-task

D Real data experiments

D.1 Experimental details

Computational resources. We used CPU cores to run this set of experiments. In particular, for each trial, we used 2 CPU cores and up to 16 GB of memory. We ran the experiments in parallel using a shared cluster with 10000 CPU cores approximately.

General experimental setup. As mentioned in §6.2, we repeat all experiments 10 times, which correspond to 10 different random initialization of the parameters of the GNNs. In all cases, we choose the model parameters with the best validation performance during training. In order to run the experiments and collect the results, we used the GraphGym framework [47], which includes the data processing and loading of the datasets, as well as the evaluation and collection of the results. We split the datasets in 70 % training, 15 % validation, and 15 % test.

We cross-validate the number of message-passing layers in the network $(2, 3, 4)$, as well as the learning rate $([0.01, 0.005])$. Then, we report the results of the best validation error among the 4 possible combinations. However, in practice we found the best performance always to use 4 message-passing layers, and thus the only difference in configuration lies in the learning rate.

We use residual connections between the GNN layers, 4 heads in the attention models, and the Parametric ReLU (PReLU) [19] as the nonlinear activation function. We do not use batch normalization [22], nor dropout [35]. We use the Adam optimizer [24] with $\beta = (0.9, 0.999)$, and an exponential learning-rate scheduler with $\gamma = 0.998$. We train all the models for 2500 epochs. Importantly, we do not use weight decay, since this will bias the solution towards $\lambda_1 = 0$ and $\lambda_2 = 1$.

We use the Pytorch Geometric [13] implementation of L-CAT for all experiments, switching between models by properly by setting λ_1 and λ_2. We parametrize λ_1 and λ_2 as free-parameters in log-space that pass through a sigmoid function—i.e., $\texttt{sigmoid}(10^x)$—such that they are constrained to the unit interval, and they are learned quickly.

D.2 Additional results

Table 6 shows the results presented in the main paper (with the addition of a dense feed-forward network), while Table 7 presents the results for the remaining datasets, with smaller average degree.

If we focus on Table 7, we observe that all models perform equally well, yet in a few cases CAT and L-CAT are significantly better than the baselines—e.g., L-CATv2 in *CoauthorCS*, or L-CAT in *Cora*. Following a similar discussion as the one presented in the main paper, these results indicates that L-CAT achieves similar or better performance than baseline models and thus, should be the preferred architecture.

Competitive performance without the graph. We also include in Tables 6 and 7 the performance of a feed-forward network, referred to as Dense (first row). Note that the only data available to this model are the node features, and thus no graph information is provided. Therefore, we should expect a significant drop in performance, which indeed happens for some datasets such as *Amazon Computers* ($\approx 7\%$ drop), *FacebookPagePage* ($\approx 20\%$ drop), *DBLP* ($\approx 9\%$ drop) and *Cora* ($\approx 14\%$ drop). Still, we found that for other commonly used datasets the performance is similar, e.g., *Coauthor Physics* and *PubMed*; or *it is even better CoauthorCS*. These results manifest the importance of a proper benchmarking, and of carefully considering the datasets used to evaluate GNN models.

Table 6: Test accuracy (%) of the considered convolution and attention models for different datasets (sorted by their average node degree), and averaged over ten runs. Bold numbers are statistically different to their baseline model ($\alpha = 0.05$). Best average performance is underlined.

Dataset	*Amazon Computers*	*Amazon Photo*	*GitHub*	*Facebook PagePage*	*Coauthor Physics*	*TwitchEN*
Avg. Deg.	35.76	31.13	15.33	15.22	14.38	10.91
Dense	83.73 ± 0.34	91.74 ± 0.46	81.21 ± 0.30	75.89 ± 0.66	95.41 ± 0.14	56.26 ± 1.74
GCN	90.59 ± 0.36	95.13 ± 0.57	84.13 ± 0.44	94.76 ± 0.19	96.36 ± 0.10	57.83 ± 1.13
GAT	89.59 ± 0.61	94.02 ± 0.66	83.31 ± 0.18	94.16 ± 0.48	96.36 ± 0.10	57.59 ± 1.20
CAT	**90.58 ± 0.40**	**94.77 ± 0.47**	**84.11 ± 0.66**	**94.71 ± 0.30**	96.40 ± 0.10	58.09 ± 1.61
L-CAT	**90.34 ± 0.47**	**94.93 ± 0.37**	**84.05 ± 0.70**	**94.81 ± 0.25**	96.35 ± 0.10	57.88 ± 2.07
GATv2	89.49 ± 0.53	93.47 ± 0.62	82.92 ± 0.45	93.44 ± 0.30	96.24 ± 0.19	57.70 ± 1.17
CATv2	**90.44 ± 0.46**	**94.81 ± 0.55**	**84.10 ± 0.88**	**94.27 ± 0.31**	96.34 ± 0.12	57.99 ± 2.02
L-CATv2	**90.33 ± 0.44**	**94.79 ± 0.61**	**84.31 ± 0.59**	**94.44 ± 0.39**	96.29 ± 0.13	57.89 ± 1.53

E Open Graph Benchmark experiments

E.1 Experimental details

Computational resources. For this set of experiments, we had at our disposal a set of 16 Tesla V100-SXM GPUs with 160 CPU cores, shared among the rest of the department.

Statistical significance. For each CAT and L-CAT model, we highlight significant improvements according to a two-sided paired t-test ($\alpha = 0.05$), with respect to its corresponding baseline model. For example, for L-CATv2 with 8 heads we perform the test with respect to GATv2 with 8 heads.

General experimental setup. As mentioned in §6.3, we repeat all experiments with OGB datasets 5 times. In all cases, we choose the model parameters with the best validation performance during

Table 7: Test accuracy (%) of the considered convolution and attention models for different datasets (sorted by their average node degree), and averaged over ten runs. Bold numbers are statistically different to their baseline model ($\alpha = 0.05$). Best average performance is underlined.

Model Avg. Deg.	*CoauthorCS* 8.93	*DBLP* 5.97	*PubMed* 4.5	*Cora* 3.9	*CiteSeer* 2.74
Dense	<u>94.88 ± 0.21</u>	75.46 ± 0.27	88.13 ± 0.33	72.75 ± 1.72	73.02 ± 1.01
GCN	93.85 ± 0.23	84.18 ± 0.40	88.50 ± 0.18	<u>86.68 ± 0.78</u>	<u>75.76 ± 1.09</u>
GAT	93.80 ± 0.38	84.15 ± 0.39	<u>88.62 ± 0.18</u>	85.95 ± 0.95	75.40 ± 1.43
CAT	93.70 ± 0.31	84.10 ± 0.29	<u>88.58 ± 0.25</u>	85.85 ± 0.79	75.64 ± 0.91
L-CAT	93.65 ± 0.23	84.13 ± 0.26	88.45 ± 0.32	**86.66 ± 0.87**	75.04 ± 1.12
GATv2	93.19 ± 0.64	<u>84.33 ± 0.18</u>	88.52 ± 0.27	85.65 ± 1.01	75.14 ± 1.20
CATv2	93.51 ± 0.34	**84.15 ± 0.41**	88.54 ± 0.29	85.50 ± 0.94	74.68 ± 1.30
L-CATv2	**93.65 ± 0.20**	84.31 ± 0.31	88.48 ± 0.24	85.75 ± 0.72	75.04 ± 1.30

training. Moreover, when we show the results without specifying the number of heads, we take the model with the best validation error among the two models with 1 and 8 heads.

We use the same implementation of L-CAT for all experiments, switching between models by properly setting λ_1 and λ_2. Experiments on *arxiv*, *mag*, *products* use a version of L-CAT implemented in Pytorch Geometric [13]. Experiments on *proteins* use a version of L-CAT implemented in DGL [41]. We parametrize λ_1 and λ_2 as free-parameters in log-space that pass through a sigmoid function—i.e., $\texttt{sigmoid}(10^x)$—such that they are constrained to the unit interval, and they are learned quickly.

ArXiv. As described in §6.3, we use the example code from the OGB framework [20]. The network is composed of 3 GNN layers with a hidden size of 128. We use batch normalization [22] and a dropout [35] of 0.5 between the GNN layers, and Adam [24] with a learning rate of 0.01. We use the ReLU as activation function. For the initial experiments, we train for 1500 epochs, while we train for 500 epochs for the noise experiments in §6.3.1. This is justified given the convergence plots in Fig. 2.

MAG. We adapted the official code from [8]. The network is composed of 2 layers with 128 hidden channels. This time, we use layer normalization [2] and a dropout of 0.5 between the layers. Again, we use ReLU as the activation function, and add residual connections to the network. As with *arxiv*, we use Adam [24] with learning rate 0.01. We set a batch size of 20000 and train for 100 epochs.

Products. We use the same setup as [8], with a network of 3 GNN layers and 128 hidden dimensions. We apply residual connections once again, with a dropout [35] of 0.5 between layers. This time, we use ELU as the activation function. The batch size is set to 256. Adam [24] is again the optimizer in use, this time with a learning rate of 0.001. We train for 100 epochs, although we apply early stopping whenever the validation accuracy stops increasing for more than 10 epochs. Note the training split of this dataset only contains 8 % of the data.

Proteins. We follow once more the setup of [8]. The network we use has 6 GNN layers of hidden size 64. Dropout [35] is set to 0.25 between layers, with an input dropout of 0.1. At the beginning of the network, we place a linear layer followed by a ReLU activation to encode the nodes, and a linear layer at the end of the network to predict the class. Moreover, we use batch normalization [22] between layers and ReLU as the activation function. We train the model for 1200 epochs at most, with early stopping after not improving for 10 epochs.

E.2 Additional results

We show in Tables 8 to 10 the results of the main paper for the *arxiv*, *mag*, *products* datasets, respectively, without selecting the best configuration for each type of model. That is, we show the results for both number of heads. Note that we already show the full table of results for the *protein* datasets in the main paper (Table 3). All the trends discussed in the main paper hold.

Extrapolation ablation study. Due to page constraints, these results were not added to the main paper. Here, we study two questions. First, how important are λ_1 and λ_2 in the formulation of L-CAT (Eq. 7)? For the sake of completeness, the second question we attempt to answer here is whether we

Table 8: Test accuracy on the *arxiv* dataset for attention models using 1 head and 8 heads.

	GCN	GAT	CAT	L-CAT	GATv2	CATv2	L-CATv2
1h	71.58 ± 0.19	71.58 ± 0.15	$\mathbf{72.04 \pm 0.20}$	$\mathbf{72.00 \pm 0.11}$	71.70 ± 0.14	$\mathbf{72.02 \pm 0.08}$	71.96 ± 0.21
8h	–	71.63 ± 0.11	$\underline{\mathbf{72.14 \pm 0.20}}$	$\mathbf{71.98 \pm 0.08}$	71.72 ± 0.24	71.76 ± 0.14	71.91 ± 0.16

Table 9: Test accuracy on the *mag* dataset for attention models using 1 head and 8 heads.

	GCN	GAT	CAT	L-CAT	GATv2	CATv2	L-CATv2
1h	$\underline{32.77 \pm 0.36}$	32.35 ± 0.24	31.98 ± 0.46	32.47 ± 0.38	32.76 ± 0.18	$\mathbf{32.43 \pm 0.22}$	32.68 ± 0.50
8h	–	32.15 ± 0.31	$\mathbf{31.58 \pm 0.22}$	32.49 ± 0.21	$\underline{32.85 \pm 0.21}$	$\mathbf{32.34 \pm 0.18}$	$\mathbf{32.38 \pm 0.28}$

Table 10: Test accuracy on the *products* dataset for attention models using 1 head and 8 heads.

	GCN	GAT	CAT	L-CAT	GATv2	CATv2	L-CATv2
1h	74.12 ± 1.20	$\underline{78.53 \pm 0.91}$	$\mathbf{77.38 \pm 0.36}$	77.19 ± 1.11	73.81 ± 0.39	74.81 ± 1.12	$\mathbf{76.37 \pm 0.92}$
8h	–	$\underline{78.23 \pm 0.25}$	$\mathbf{76.63 \pm 1.15}$	$\mathbf{76.56 \pm 0.45}$	76.40 ± 0.71	75.20 ± 0.92	$\mathbf{74.70 \pm 0.28}$

can obtain similar performance by just interpolating between GCN and GAT (fixing $\lambda_2 = 0$)? Note that we theoretically showed in §§4 and 6.1 that CAT fills up a gap between GCN and GAT, making it preferable in certain settings.

To this end, we repeat the experiments for network-initialization robustness in §6.3.2, since they showed to be the best ones to tell apart the performance across models. We include three additional models: GCN-GAT, which interpolates between GCN and GAT (or GATv2) by learning λ_1 and fixing $\lambda_2 = 0$; CAT-λ_1 which interpolates between GCN and CAT by learning λ_1 and fixing $\lambda_2 = 1$; and CAT-λ_2, which interpolates between GAT and CAT by learning λ_2 and fixing $\lambda_1 = 1$.

Results using GAT and shown in Table 11, and using GATv2 in Table 12. We can observe that GCN-GAT obtains results in between GCN and GAT for all settings, despite being able to interpolate between both layers in each of the six layers of the network. Regarding learning λ_1 and λ_2, we can observe that there is a clear difference between learning boths (L-CAT), and learning a single one. For both attention models, CAT-λ_1 obtains better results than CAT-λ_2 in all settings, but *uniform* with 8 heads. Still, the results of both variants are substantially worse than those of L-CAT in all cases, *demonstrating the importance of learning to interpolate between the three layer types*.

Table 11: Test accuracy on the *proteins* dataset for GCN [25] and GAT [38] attention models using two network initializations, and two numbers of heads (1 and 8).

	GCN	GCN-GAT	GAT	CAT	L-CAT	CAT-λ_1	CAT-λ_2
				uniform initialization			
1h	61.08 ± 2.86	$\mathbf{70.44 \pm 1.56}$	59.73 ± 4.04	$\mathbf{74.19 \pm 0.72}$	$\underline{\mathbf{77.77 \pm 1.44}}$	$\mathbf{71.97 \pm 3.78}$	$\mathbf{73.55 \pm 1.36}$
8h	–	$\mathbf{68.51 \pm 0.91}$	72.23 ± 3.20	73.60 ± 1.27	$\underline{\mathbf{78.85 \pm 1.76}}$	$\mathbf{76.43 \pm 2.47}$	72.76 ± 2.79
				normal initialization			
1h	$\underline{80.10 \pm 0.61}$	66.51 ± 3.23	66.38 ± 7.76	73.26 ± 1.84	$\mathbf{78.06 \pm 1.40}$	$\mathbf{76.77 \pm 1.91}$	73.39 ± 1.25
8h	–	$\mathbf{69.93 \pm 1.93}$	79.08 ± 1.64	$\mathbf{74.67 \pm 1.29}$	$\underline{79.63 \pm 0.79}$	78.86 ± 1.07	$\mathbf{73.32 \pm 1.15}$

Table 12: Test accuracy on the *proteins* dataset for GCN [25] and GATv2 [8] attention models using two network initializations, and two numbers of heads (1 and 8).

	GCN	GCN-GATv2	GATv2	CATv2	L-CATv2	CATv2-λ_1	CATv2-λ_2
				uniform initialization			
1h	61.08 ± 2.86	$\mathbf{69.69 \pm 1.59}$	59.85 ± 3.05	$\mathbf{64.32 \pm 2.61}$	$\underline{\mathbf{79.08 \pm 1.06}}$	63.24 ± 1.55	$\mathbf{73.41 \pm 0.34}$
8h	–	$\mathbf{69.94 \pm 1.62}$	75.21 ± 1.80	74.16 ± 1.45	$\underline{\mathbf{78.77 \pm 1.09}}$	$\mathbf{77.61 \pm 1.32}$	73.96 ± 1.27
				normal initialization			
1h	$\underline{80.10 \pm 0.61}$	68.54 ± 1.63	69.13 ± 9.48	74.33 ± 1.06	$\mathbf{79.07 \pm 1.09}$	78.41 ± 0.93	74.07 ± 1.17
8h	–	$\mathbf{68.71 \pm 1.96}$	78.65 ± 1.61	$\mathbf{73.40 \pm 0.62}$	$\underline{79.30 \pm 0.55}$	78.76 ± 1.41	$\mathbf{73.22 \pm 0.77}$

References

[1] T.W. Anderson. *An introduction to multivariate statistical analysis*. John Wiley & Sons, 2003.

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[3] Aseem Baranwal, Kimon Fountoulakis, and Aukosh Jagannath. Graph convolution for semi-supervised classification: Improved linear separability and out-of-distribution generalization. In *International Conference on Machine Learning (ICML)*. PMLR, 2021.

[4] Aseem Baranwal, Kimon Fountoulakis, and Aukosh Jagannath. Effects of graph convolutions in deep networks. *arXiv preprint arXiv:2204.09297*, 2022.

[5] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[6] K. Bhatia, K. Dahiya, H. Jain, P. Kar, A. Mittal, Y. Prabhu, and M. Varma. The extreme classification repository: Multi-label datasets and code, 2016. URL `http://manikvarma.org/downloads/XC/XMLRepository.html`.

[7] Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. *arXiv preprint arXiv:1707.03815*, 2017.

[8] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? In *International Conference on Learning Representations (ICLR)*, 2022.

[9] Dan Busbridge, Dane Sherburn, Pietro Cavallo, and Nils Y Hammerla. Relational graph attention networks. *arXiv preprint arXiv:1904.05811*, 2019.

[10] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. *Advances in Neural Information Processing Systems (NeurIPS)*, 33, 2020.

[11] Yash Deshpande, Subhabrata Sen, Andrea Montanari, and Elchanan Mossel. Contextual stochastic block models. *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018.

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

[13] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[14] Kimon Fountoulakis, Amit Levi, Shenghao Yang, Aseem Baranwal, and Aukosh Jagannath. Graph attention retrospective. *arXiv preprint arXiv:2202.13060*, 2022.

[15] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning (ICML)*, 2017.

[16] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010.

[17] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

[18] William L. Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *IEEE Data Eng. Bull.*, 40, 2017.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, 2015.

[20] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:22118–22133, 2020.

[21] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. In *The Web Conference (WWW)*, 2020.

[22] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*. PMLR, 2015.

[23] Dongkwan Kim and Alice Oh. How to find your friendly neighborhood: Graph attention design with self-supervision. In *International Conference on Learning Representations (ICLR)*, 2021.

[24] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Machine Learning (ICML)*, 2015. URL http://arxiv.org/abs/1412.6980.

[25] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[26] Boris Knyazev, Graham W Taylor, and Mohamed Amer. Understanding attention and generalization in graph neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019.

[27] Devin Kreuzer, Dominique Beaini, Will Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. *Advances in Neural Information Processing Systems (NeurIPS)*, 34, 2021.

[28] John Boaz Lee, Ryan A Rossi, Sungchul Kim, Nesreen K Ahmed, and Eunyee Koh. Attention models in graphs: A survey. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 13, 2019.

[29] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *AAAI conference on artificial intelligence*, volume 33, 2019.

[30] P. Rigollet and J.-C. Hütter. High dimensional statistics. *Lecture notes for course 18S997*, 813: 814, 2015.

[31] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *Journal of Complex Networks*, 2021.

[32] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20, 2008.

[33] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3), 2008.

[34] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

[35] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[36] Kiran K Thekumparampil, Chong Wang, Sewoong Oh, and Li-Jia Li. Attention-based graph neural network for semi-supervised learning. *arXiv preprint arXiv:1803.03735*, 2018.

[37] 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 (NeurIPS)*, 30, 2017.

[38] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[39] Guangtao Wang, Rex Ying, Jing Huang, and Jure Leskovec. Multi-hop attention graph neural networks. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2021.

[40] Kuansan Wang, Zhihong Shen, Chiyuan Huang, Chieh-Han Wu, Yuxiao Dong, and Anshul Kanakia. Microsoft academic graph: When experts are not enough. *Quantitative Science Studies*, 2020.

[41] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*, 2019.

[42] 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 (WWW)*, 2019.

[43] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. Graph neural networks in recommender systems: a survey. *ACM Computing Surveys (CSUR)*, 2020.

[44] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32, 2020.

[45] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations (ICLR)*, 2019.

[46] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, 2016.

[47] Jiaxuan You, Zhitao Ying, and Jure Leskovec. Design space for graph neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 33, 2020.

[48] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. Graph transformer networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019.