Graph Representation Learning

1044704
Honour School of Mathematics and Computer Science (Part C)

Question 1

Over-smoothing:

(a) The message-passing neural network (MPNN) framework captures many popular types of graph neural network (GNN). After several iterations of an MPNN, node representations may become so similar as to be indistinguishable, a phenomenon known as "over-smoothing" (Li et al. (2018)). This is an issue because if nodes cannot be distinguished, then node classification and any downstream tasks become impossible.

This limitation is particularly prominent for models making homophily assumptions such as Graph Convolution Networks (GCNs) and Graph Attention Networks (GATs). On baseline datasets such as Cora, over-smoothing can degrade GCN performance to random levels even for networks $\sim$10 layers deep (see GRL Practical 4). In some heterophilic datasets such as the Actor co-occurrence network dataset, Pei et al. (2020) found GCN to be inferior to a simple multi-layer perceptron.

(b) Li et al. (2018) analysed the theoretical working of Graph Convolution Networks (GCNs), and showed that the graph convolution step on feature matrix $\mathbf{X}$ is equivalent to a special form of Laplacian smoothing, resulting in convolved matrix $\tilde{\mathbf{X}} = \tilde{\mathbf{A}}_{sym}\mathbf{X}$. In $\tilde{\mathbf{X}}$, each node's features is the weighted average of its original features and its neighbour's features. This provides a formal explanation for why features from different clusters appear more similar after convolution.

Zhao and Akoglu (2020) present a complementary viewpoint, based on the observation that for any initial feature vector $\mathbf{x}_j$

$$\lim_{k\to\infty} \tilde{\mathbf{A}}^k_{sym}\mathbf{x}_j \propto \boldsymbol{\pi} \qquad \text{for some vector } \boldsymbol{\pi} \text{ depending only upon the node degrees.}$$

As a result, over-smoothing can also be seen to remove all initial information from each node.

(c) One proposal for alleviating the over-smoothing limitation is PairNorm (Zhao and Akoglu (2020)). PairNorm introduces a normalisation layer between graph convolution layers to prevent all representations being smoothed together. This normalisation layer enforces a constant total pairwise squared distance between node feature representations, which is achieved through a two-step, centre-and-scale normalization procedure carried out node-wise.

PairNorm is simple to implement, parameterless and integrates easily with many popular GNN architectures. It is also a fast operation to carry out (linear time in inputs). Furthermore, PairNorm has a strong theoretical basis, with conceptual links to regularisation in traditional machine learning as well as graph optimisation problems.

Zhao and Akoglu (2020) show that PairNorm is effective at preventing accuracy drop-off due to over-smoothing as layers increase on a number of baseline node-classification datasets, although it doesn't improve on the accuracy of GCN with few layers. The authors convincingly justify this by the lack of necessity for many-layer GNNs in these cases. On a problem instance (SSNC-MV) specially selected to favour higher depth GNNs, they do observe improvements in accuracy with PairNorm using more layers than possible with GCNs alone.

While PairNorm was one of the first, many normalisation methods for GNNs are now present in the literature. Data from Chen et al. (2022) suggests that alternative normalisation methods that exploit a graph's structural information consistently outperform PairNorm. This is unsurprising

given the motivation behind GNNs, that models utilising graph structure outperform those that do not. PairNorm remains an effective approach for alleviating over-smoothing, particularly for node classification tasks, but is no longer a state-of-the-art method.

Over-squashing:

(a) In a GNN, a node u's layer-k representation $\mathbf{h}_u^{(k)}$ is calculated using information from all of the nodes in its receptive field. As k increases, the computation graph for calculating $\mathbf{h}_u^{(k)}$ is tree-like and so $\mathbf{h}_u^{(k)}$'s receptive field grows exponentially (in most non-trivial examples). As a result, the fixed-length vector $\mathbf{h}_u^{(k)}$ compresses information from exponentially many sources, leading to a bottleneck. This phenomenon is known as over-squashing (Alon and Yahav (2022)).

The consequence of this bottleneck is that messages from distant nodes are not propagated through the network. For example, long-range interaction is required to predict chemical properties of molecules (see Gilmer et al. (2017)), as atoms of relevance may be spatially close but far apart in a graph - over-squashing severely limits GNN performance in this case.

(b) If information has been over-squashed, then each pair of far away nodes will have minimal influence on one another. Take some nodes u, v at distance r from one another. The influence of initial feature $\mathbf{h}_u^0$ on final feature $\mathbf{h}_v^r$ can be quantified with the Jacobian $\frac{\partial \mathbf{h}_v^r}{\partial \mathbf{h}_u^0}$, which is calculated as part of backpropagation. Topping et al. (2022) showed that for typical graphs (i.e. those with node degrees much larger than 1), under reasonable boundedness conditions on an MPNN, the norm of the Jacobian $\left|\frac{\partial \mathbf{h}_v^r}{\partial \mathbf{h}_u^0}\right|$ is upper bounded by a term that decays exponentially in r.

This shows that for MPNNs on most graphs of interest, the influence node u can have on node v decays exponentially as the distance between them increases. This gives a formal explanation for why over-squashing limits the ability of GNNs to learn long-range dependencies.

(c) Abboud et al. (2022) proposed *shortest path* MPNNs (SP-MPNNs) as a method for alleviating over-squashing. In this generalisation of the message passing framework, node representations are calculated using aggregated representations from each of the shortest path neighbourhoods $N_1(u), ..., N_k(u)$ (where in MPNNs we use just the aggregation over $N_1(u) = N(u)$).

When considering neighbourhoods up to $k = \frac{r}{q}$ hops away, the authors show that the upper bound on the Jacobian $\frac{\partial \mathbf{h}_v^r}{\partial \mathbf{h}_u^0}$ may be weakened to an exponential decay in $q = \frac{r}{k}$. Intuitively this is because the SP-MPNNs span k hops per iterations. Crucially, the exponential dependence is reduced to linear if $k \geq r$. This analysis shows that theoretical restrictions to gradient flow are weakened by SP-MPNNs, thus alleviating over-squashing. This theoretical result is backed up empirically by the paper. Furthermore, the SP-MPNNs framework subsumes models that have achieved state-of-the-art performance such as Graphormers (Ying et al. (2021)), so is a cutting-edge framework.

Compared to alternatives, the number of iterations required is minimised by considering k-hops instead of graph re-wiring, and so SP-MPNNs retain more structural information. However, this comes at the cost of an increase in computational complexity. For a task with problem radius R, to maximally alleviate over-squashing $k \approx R$ is required, but in the worst case, space complexity with SP-MPPNs is $O(k)$ worse than MPNNs. Furthermore, all-pairs of shortest paths must be calculated for an input graph, which requires time $O(|V||E|)$ - this is prohibitive for large graphs.

Overall, SP-MPPNs are a natural extension of MPPNs that provide a theoretical and practical solution to the over-squashing problem in all but the largest graphs.

Limited expressive power:

(a) In the traditional machine learning setting, it is well known that multilayer feedforward networks are universal function approximators (e.g. see Hornik et al. (1989)). In contrast, GNNs trade universality for powerful inductive biases on graph data, and so have limited expressive power.

When nodes aren't uniquely identifiable, then two graphs are in general hard to distinguish from one another (c.f. NP-hardness of GRAPH-ISOMORPHISM). In the biological context, MPNNs lack the expressive power to distinguish the organic compounds Decalin and Bicyclopentyl (Balcilar et al. (2021)), and so can't accurately predict their biological interactions.

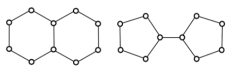


Figure 1: Decalin (left) and Bicyclopentyl (right)

(b) We can classify the expressiveness of models by considering which sets of graphs they can distinguish. MPNNs have been shown to unable to distinguish non-isomorphic graphs that are indistinguishable by the first-order Weisfeiler-Leman (1-WL) graph isomorphism test (Xu et al. (2018)). We can therefore quantify the limitations of MPNNs by considering the expressivity of 1-WL (a well-understood algorithm).

We can also characterise the expressiveness of GNNs by the class of functions they may represent. This class of functions may be expressed as a fragment of first-order logic over graphs. For example, Barcelo et al. (2020) showed that each logical formula in $\mathbf{C}^2$ with one free variable can be represented by an MPNN with global readout.

(c) The limited expressive power of GNNs is often addressed by the construction of higher-order models. An alternative to such models to enhance MPNNs with partial random node initialisation (RNI), where, as well as our given node features, we randomly initialise some additional node features. This approach individualises graphs with high probability, while retaining strong inductive biases (in particular permutation-invariance in expectation).

While giving similar performance to higher-order models such as k-GNNs, partial RNI is significantly more space and time efficient. Furthermore, Abboud et al. (2021) showed that GNNs with RNI are universal function approximators for real-valued invariant functions on graphs (in contrast to many higher-order models), thus showing RNI is an easy way to increase model expressibility.

This increase in expressive power comes at the cost of slower model convergence; this is unsurprising as partial RNI essentially adds some noisy features to a model. In the case that a graph is not unique under the 1-WL test, the model fundamentally faces a harder learning task: it must first use RNI features to learn structural features not available to it through MPNN without RNI, before then becoming robust against RNI variability. Despite the universality result, it is not guaranteed that this more difficult learning task is always achievable in computationally reasonable timeframes.

Overall, the successful use application of partial RNI is highly dataset dependent. Zopf (2022) finds that in a variety of popular graph datasets 1-WL expressiveness is sufficient to achieve near-perfect training accuracy. This suggests that in practical applications, generalisability is usually a greater constraint than expressiveness. Nonetheless, RNI remains a useful method as it is can be a computationally inexpensive way to address under-fitting due to insufficient expressive power.

Question 2

(a) We only require a single layer for our parametrisation (i.e. $T = 1$), so we omit the superscripts for our parameters where there is no confusion.

We let $\mathbf{W}_{\text{self}} = \mathbf{0}_{1\times d}$, $\mathbf{W}_{\text{neigh}} = \mathbf{1}_{1\times d}$ and $b = 0$. Then we define our non-linearity $\sigma : \mathbb{R} \to \mathbb{R}$ by $\sigma(x) = \sin^2(\frac{\pi}{2}x)$ (a differentiable function).

For $u \in V$, consider the final representations $\mathbf{z}_u \in \mathbb{R}$:

$$
\begin{aligned}
\mathbf{z}_u &= \sigma\left(\mathbf{W}_{\text{self}}\mathbf{x}_u + \mathbf{W}_{\text{neigh}} \sum_{v\in N(u)} \mathbf{x}_v + b\right) && \text{(definition of base GNN)} \\
&= \sigma\left(\mathbf{1}_{1\times d} \sum_{v\in N(u)} \mathbf{x}_v\right) && \text{(definitions of parameter matrices)} \\
&= \sigma\left(|N(u)|\right) && \text{(as } \mathbf{x}_v \text{ are 1-hot features)} \\
&= \sin^2\left(\frac{\pi|N(u)|}{2}\right) && \text{(definition of } \sigma \text{)} \\
&= \begin{cases} 1, & \text{if } |N(u)| \equiv 1 \pmod 2 \\ 0, & \text{otherwise.} \end{cases} && \text{(considering the definition of } \sin^2(\frac{\pi}{2}x) \text{)} \\
&= f(u)
\end{aligned}
$$

Hence our parametrisation precisely represents the function f.

(b) At no point does the parametrised model from part (a) use any information specific to the graph G except the number of nodes $|V|$ which is equal to d (the size of the one-hot feature vectors). This means the parametrisation is independent of the choice of one-hot vectors (i.e. the permutation of nodes) and the graph structure itself. So provided graph G_1 has the same number of nodes, the model may be used to represent f on G_1 without modification.

If G_1 has a different number of nodes to d, clearly we run into dimension errors immediately with our 1-hot vectors. However, we may use the model specified identically except with a different value of $d = |V_1|$, because the model in part (a) is specified to work with any fixed value of d.

(c) Define the function $g : V \to \mathbb{B}$ by

$$
g(u) = \begin{cases} 1, & \text{if } \mathbf{x}_u = [1,0]^\top \text{ and } \forall v \in N(u),\ \mathbf{x}_v = [0,1]^\top \\ 0, & \text{otherwise.} \end{cases}
$$

In words, g evaluates to 1 on node u if and only if u has initial node feature $[1,0]^\top$ and all nodes in its neighbourhood have initial node feature $[0,1]^\top$.

Base GNN model (Eq 1):

We provide an explicit one-layer parametrisation to show that the base GNN can capture this function.

Let

$$\mathbf{W}_{\text{self}} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \quad \mathbf{W}_{\text{neigh}} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad \sigma = id \quad .$$

For $u \in V$, consider the value of $\mathbf{z}_u \in \mathbb{R}^2$.

If $\mathbf{x}_u = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$:

$$\begin{aligned}
\mathbf{z}_u &= id\left(\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}\sum_{v \in N(u)} \mathbf{x}_v\right) && \text{(GNN layer with our parameters)} \\
&= \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}\sum_{v \in N(u)} \mathbf{x}_v \\
&= \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \cdot |\{v \in N(u) \mid \mathbf{x}_v = [1,0]^\top\}| && \text{(as } \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \text{ and } \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \text{)} \\
&= \begin{bmatrix} 1 \\ |\{v \in N(u) \mid \mathbf{x}_v = [1,0]^\top\}| \end{bmatrix}
\end{aligned}$$

Else if $\mathbf{x}_u = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$:

$$\begin{aligned}
\mathbf{z}_u &= id\left(\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}\sum_{v \in N(u)} \mathbf{x}_v\right) && \text{(GNN layer with our parameters)} \\
&= \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}\sum_{v \in N(u)} \mathbf{x}_v \\
&= \begin{bmatrix} 0 \\ |\{v \in N(u) \mid \mathbf{x}_v = [1,0]^\top\}| \end{bmatrix} && \text{(as in the previous case)}
\end{aligned}$$

As $g(u) = 1 \iff \left(\mathbf{x}_u = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \text{ and } \forall v \in N(u), \mathbf{x}_v = \begin{bmatrix} 0 \\ 1 \end{bmatrix}\right)$, we have that $\mathbf{z}_u = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \iff g(u) = 1$. This is a trivially classifiable instance by $\mathfrak{C}$.

(In the event that $\mathfrak{C}$ must be a linear classifier, the inequality $-x_1 + x_2 \leq -1$ is sufficient to partition the node representations according to g.)

Base self-loop GNN model (Eq 2):

Let $n \geq 2$.

Let (V_1, E_1) and (V_2, E_2) be two distinct copies of the complete graph K_n with n nodes, where $V_i = \{\mathbf{v}_1^{(i)}, \mathbf{v}_2^{(i)}, ..., \mathbf{v}_n^{(i)}\}$. Let $\mathbf{X}_2$ be such that $\mathbf{x}_1 = [1,0]^\top$ and all other $\mathbf{x}_u = [0,1]^\top$. Let $\mathbf{X}_2$ be such that all $\mathbf{x}_u = [1,0]^\top$.

Then let $G = (V_1 \cup V_2, E_1 \cup E_2, \mathbf{X}_1 \cup \mathbf{X}_2)$. We observe that initially we can't represent g with a simple transformation of the node features, because in V_1 the node with initial representation

$[1,0]^\top$ should map to 1 under g, whereas in V_2 nodes with initial representation $[1,0]^\top$ should map to 0.

We now show that adding GNN layers does not help. As K is complete, for any $u \in V_1$, $V_1 = N(u) \cup \{u\}$. This means that if $u \in V_1$, for any t

$$\sum_{v \in N(u) \cup \{u\}} \mathbf{h}_v^{(t)} = \sum_{v \in V_1} \mathbf{h}_v^{(t)} \quad .$$

As a consequence, for all $u, v \in V_1, t \in \mathbb{N}^{\geq 0}$, each $\mathbf{h}_u^{(t+1)} = \mathbf{h}_v^{(t+1)}$. This means that at any time $t \geq 1$, all node representations within V_1 are equal, and so we cannot distinguish which node in V_1 should map to 1 under g and which nodes should map to 0.

This proves that we cannot represent g with the base self-loop GNN model.

(d) By Barcelo et al. (2020) the base GNN model can capture graded modal logic.

In our context, our nodes have identical node features, and so a graded modal logic formula is defined either as **true**, or inductively as one of the following, where ϕ, ψ are graded modal logic formulas and $N \in \mathbb{N}^{>0}$:

$$\neg\phi(x), \quad \phi(x) \wedge \psi(x), \quad \exists^{\geq N} y (E(x,y) \wedge \phi(y))$$

(See Section 4 in Barcelo et al. (2020).)

We may express f as a logical node classifier, i.e. a formula $\Phi(x)$ in $\mathbf{C}^2$ with one free variable, where $\forall u \in V, \quad \Phi(u) = f(u)$:

$$\Phi(x) = \neg\exists^{\geq 3} y \left(E(x,y) \wedge \exists^{\geq 3} x E(x,y)\right)$$

(In words this logical fragment reads "*there does not exist three or more nodes* $\boldsymbol{y}$*, where* $\boldsymbol{y}$ *is a neighbour of* $\boldsymbol{x}$ *and* $\boldsymbol{y}$ *has degree at least three* " - this clearly shows $\Phi \equiv f$.)

By inserting some superfluous brackets, we can clearly see that Φ is expressible as a formula in graded modal logic:

$$\Phi(x) \equiv \neg \big[\quad \exists^{\geq 3} y \left(E(x,y) \wedge \big(\quad \exists^{\geq 3} x E(x,y) \wedge \mathbf{true} \quad \big)\right) \quad \big]$$

Hence, by Proposition 4.1 in Barcelo et al. (2020), there exists some parametrisation of the base GNN model (Eq 1) that captures the graded modal logic classifier Φ. As $\Phi \equiv f$, we are done.

(e) <u>Initial node features set to $\mathbf{1}^d$:</u>

Yes, there does exist a Boolean function that the base GNN model cannot represent when initial node features are set to $\mathbf{1}^d$:

Take the simple graph $G = (V, E)$ with two nodes $\mathbf{v}_1, \mathbf{v}_2$ connected by a single edge, i.e $V = \{\mathbf{v}_1, \mathbf{v}_2\}$ and $E = \{(\mathbf{v}_1, \mathbf{v}_2)\}$.

Then define $g : V \to \mathbb{B}$ by $g(\mathbf{v}_1) = 1$ and $g(\mathbf{v}_2) = 0$. We can't represent g by a transformation of the initial node features as these are both $\mathbf{1}^d$.

Assume that at time step t the nodes share equal node representations, i.e. $\mathbf{h}_{\mathbf{v}_1}^{(t)} = \mathbf{h}_{\mathbf{v}_2}^{(t)}$. This is true for $t = 0$. Then

$$\begin{aligned}\mathbf{h}_{\mathbf{v}_1}^{(t+1)} &= \sigma\left(\mathbf{W}_{\text{self}}\mathbf{h}_{\mathbf{v}_1}^{(t)} + \mathbf{W}_{\text{neigh}}\mathbf{h}_{\mathbf{v}_2}^{(t)} + \mathbf{b}\right) \\ &= \sigma\left(\mathbf{W}_{\text{self}}\mathbf{h}_{\mathbf{v}_2}^{(t)} + \mathbf{W}_{\text{neigh}}\mathbf{h}_{\mathbf{v}_1}^{(t)} + \mathbf{b}\right) \\ &= \mathbf{h}_{\mathbf{v}_2}^{(t+1)}\end{aligned}$$

So by induction, at every time step t the nodes share equal node representations. Hence, the base GNN model cannot represent g as g is a non-constant function.

Initial node features 1-hot encoded:

No, the base GNN model can represent any Boolean function on V when initial node features are 1-hot encoded:

Suppose we wish to represent $g : V \to \mathbb{B}$. We use a single layer GNN. Define $\mathbf{W}_{\text{neigh}} = \mathbf{0}$, $\mathbf{b} = \mathbf{0}$ and $\sigma = id$. Then if our vertices $v_1, ..., v_n$ have one-hot features $\mathbf{e}_1, ..., \mathbf{e}_n$ (with $\mathbf{e}_i$ the standard i^{th} basis vector), we set $\mathbf{W}_{\text{self}} = \begin{bmatrix} g(v_1) & g(v_2) & ... & g(v_n) \end{bmatrix}$.

Then $\mathbf{h}_{v_i}^{(1)} = \mathbf{W}_{\text{self}}\ \mathbf{e}_i = g(v_i)$, and so our single layer GNN represents g exactly.

Question 3: How does Random Node Initialisation (RNI) affect attention aggregation in Graph Attention Networks (GATs)?

(a) Motivation

Abboud et al. (2021) report empirical results supporting "the superior performance of GNNs with RNI over standard GNNs." However, both this study and Sato et al. (2021) focus on Graph Convolution Networks (GCNs), and don't examine how GATs perform with RNI.

Noise typically makes learning tasks more difficult, and RNI fundamentally adds noise to features. It is therefore reasonable to expect that RNI would make learning an effective parametrisation of a GAT's attention mechanism more difficult. We hypothesise that *the introduction of random noise from RNI will drive the GAT's attention aggregation mechanism towards uniform aggregation.*

(b) Means to study the research question

This research question may be empirically studied by comparing the performance of a GAT and a GCN on a real-life node-classification task that favours attention aggregation over uniform aggregation. By adding more randomly initialised features, we can measure the performance of each model as the proportion of randomly initialised features increases. If the hypothesis is correct, we would expect the GCN to be more resilient to the introduction of random node features than the GAT, and that the performance of the GAT converges towards that of the GCN.

In addition, we can examine the trained model's attention coefficients to determine whether aggregation is more or less uniform after the introduction of random node features.

We denote a GAT (resp. GCN) with x RNI features as an "rGAT-x" (resp. GCN).

(c) Methodology — (codebase linked at : github.com/1044704/GRL_mini_project)

For this study we use the protein-protein interaction (PPI) dataset (Zitnik and Leskovec (2017)). Each of the 24 graphs in the PPI dataset correspond to the protein interactions within a different human tissue. We consider the node classification task of classifying proteins by their cellular function within the tissue. The original GAT paper (Veličković et al. (2018)) shows that attention aggregation outperforms uniform aggregation on this task in PPI.

We take an inductive approach to generating node embeddings. The default PPI train/val/test masks are used, with 20 graphs for training and 2 each for validation and test. Experiments are run in a Jupyter notebook using Python 3.9 and PyTorch 1.12 on a cloud-hosted RTX4000 GPU.

Model architectures and hyperparameters are based on the example GAT architecture for PPI at PyTorch Geometric. Models are trained by stochastic gradient descent using the Adam optimiser, learning rate 0.005 and early stopping for up to 350 epochs. The batch size is 1 and the loss function is binary cross entropy. GAT statistics are taken as the average over 5 independent models.

Architecturally, both rGAT-x and rGCN-x first have an RNI layer that concatenates x random features (drawn from $U[0,1]^x$) to the initial features, then 3 graph layers, respectively with attention or convolutional aggregation. Each layer has 256 hidden channels (multiplied by the number of heads for the GAT). The activations for layers 1,2 and 3 are ELU, ELU and sigmoid. Pre-normalisation attention in a GAT layer is calculated as $e_{u,v} = \mathbf{a}^\top[\mathbf{W}\mathbf{h}_u \oplus \mathbf{W}\mathbf{h}_v]$. The first 2 GAT layers have 4 heads and concatenate multi-head output; the final layer has 6 heads and sums their output.

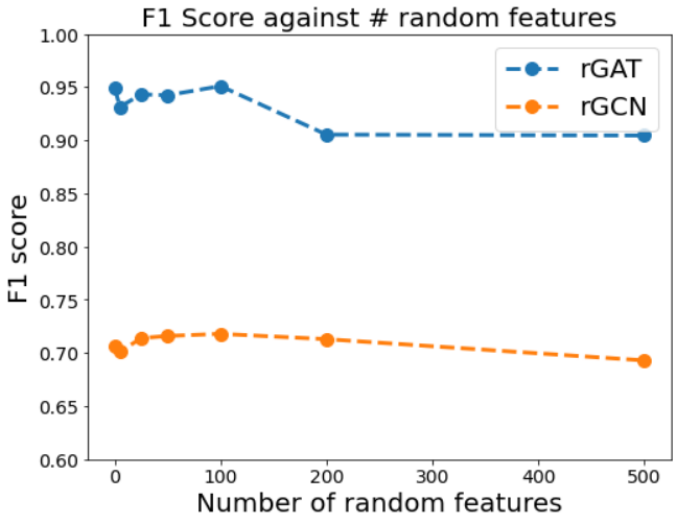
By using GCNs as our benchmark, our study assumes that degree-normalised aggregation and uniform aggregation will behave similarly on PPI. We also assume that the choice of hyperparameters

does not benefit one model (either GAT or GCN) disproportionately. Lastly, we assume that the PPI dataset is representative of other real-life datasets where attention aggregation is beneficial.

(d) Results

Figure 2 shows that as expected, GAT outperforms GCN on PPI. Veličković et al. (2018) report higher F1 scores for both models, likely due to a more optimal choice of hyperparameters.

While a limited number of random features doesn't affect each model's F1 score, with 500 random features the rGAT and rGCN model performance both degrade. In line with the hypothesis, the rGAT appears to degrade more than the rGCN, though not by enough to suggest nodes are being uniformly aggregated. Similarly to Sato et al. (2021), we find only slight difference in GCN performance with RNI on real-world biological datasets.



x	rGAT-x	rGCN-x
0	0.95	0.71
5	0.93	0.70
25	0.94	0.71
50	0.94	0.72
100	0.95	0.72
200	0.91	0.71
500	0.90	0.69

Figure 2: F1 score for rGAT and rGCN

We consider whether the degradation is due to the attention mechanism of the GAT. Restricting to the first layer of the GAT, we calculate the attention coefficients for each node, and then calculate the average (normalised) entropy on the test dataset (over all four heads). Entropy closer to 100% signifies that our coefficients are more uniform. We summarise the results in Table 1:

GAT-x : $x =$	0	5	25	50	100	200	500
Average Normalised Entropy	56%	70%	61%	58%	48%	42%	18%

Table 1: Normalised entropy of first-layer attention coefficients in rGAT-x models in test graphs

We observe that introducing a few random features initially causes the model to learn more uniform attention coefficients, in line with the hypothesis. However, for high x, the rGAT-x model is learning very sharp coefficients, contradicting our uniformity hypothesis. Figure 3 illustrates this finding, visualising the rGAT-x model's learnt attention coefficients for a single node for $x = 0, 5, 50$ and 500.

(e) Further Discussion

We find that rGAT performance is remarkably robust with random features numbering up to 2×

(a) 0 random features (b) 5 random features (c) 50 random features (d) 500 random features

Figure 3: Learned first-layer attention coefficients for the same node with rGATs with an increasing number of random features

the initial features (50), although the random features do affect the attention coefficients learnt.

Our results suggest the initial hypothesis is partially correct: the introduction of a few random features drives the attention coefficients towards uniformity; with more random features the trend reverses and our attention weights are much sharper. One explanation could be that with higher levels of RNI noise, to retain good performance the attention mechanism learns to put weight only on edges with the strongest links as more subtle links can't be found amid the noise.

Neither model can be conclusively said to benefit from RNI, suggesting that the expressivity of the models on PPI is not the performance limiting factor. However, feature data on PPI is very sparse, which suggests that RNI may be beneficial for determining relevant structural features that are otherwise indistinguishable to GNNs. Hence, our results do relate to a realistic use case for rGATs.

We can't place strong confidence in our comparisons with the rGCN. Firstly, the GAT has several heads leading to more parameters than the GCN. Secondly, GCN aggregation is degree normalised so is not a direct parallel for attention-less GAT. Lastly, we don't repeat GCN results. However, our results are indicative that the attention mechanism still outperforms uniform aggregation despite the presence of RNI.

The analysis of GAT performance and attention coefficients is much more robust. Results are taken as an average of 5 models, and strong trends are identified across 7 different levels of RNI.

To conclude, the study allows us to conclude with reasonable confidence that adding a few random features drives GAT aggregation towards uniform aggregation without performance loss, while large numbers of random features causes large performance loss and much sharper attention coefficients. This finding complements Sato et al. (2021) by showing the relative resilience of GATs to RNI, though no improvement is found to the model's classification ability in our study.

(f) Outlook

Replicating the results of the study on another real-life benchmarking task (e.g. transductive node-classification on Cora) would further increase confidence that trends observed aren't limited only to the PPI dataset. Further insight could be gained from a more detailed analysis of the attention coefficients learnt, e.g. by considering the distribution of coefficient entropy over nodes.

In addition, this research would benefit from a study on a synthetic dataset where both attention and recognising structure (e.g. triangles) are required. This would allow us to better study interactions between attention and RNI when greater than 1-WL expressibility is strictly required.

Bibliography

Ralph Abboud, İsmail İlkan Ceylan, Martin Grohe, and Thomas Lukasiewicz. The Surprising Power of Graph Neural Networks with Random Node Initialization. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pages 2112–2118, Montreal, Canada, August 2021. International Joint Conferences on Artificial Intelligence Organization. ISBN 978-0-9992411-9-6. doi: 10.24963/ijcai.2021/291.

Ralph Abboud, Radoslav Dimitrov, and Ismail Ilkan Ceylan. Shortest Path Networks for Graph Property Prediction. In *The First Learning on Graphs Conference*, December 2022.

Uri Alon and Eran Yahav. On the Bottleneck of Graph Neural Networks and its Practical Implications. In *International Conference on Learning Representations*, February 2022.

Muhammet Balcilar, Pierre Héroux, Benoit Gaüzère, Pascal Vasseur, Sébastien Adam, and Paul Honeine. Breaking the Limits of Message Passing Graph Neural Networks. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139, page 599, July 2021.

Pablo Barcelo, Jorge Perez, Egor V Kostylev, and Juan Reutter. THE LOGICAL EXPRESSIVENESS OF GRAPH NEURAL NETWORKS. *ICLR 2020*, 2020.

Yihao Chen, Xin Tang, Xianbiao Qi, Chun-Guang Li, and Rong Xiao. Learning graph normalization for graph neural networks. *Neurocomputing*, 493:613–625, July 2022. ISSN 0925-2312. doi: 10.1016/j.neucom.2022.01.003.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural Message Passing for Quantum Chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1263–1272. PMLR, July 2017.

William L. Hamilton, Z. Ying, and J. Leskovec. Inductive Representation Learning on Large Graphs. In *NIPS*, June 2017.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989. ISSN 0893-6080. doi: 10.1016/0893-6080(89)90020-8.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, pages 3538–3545. Association for the Advancement of Artificial Intelligence, February 2018. ISBN 978-1-57735-800-8.

Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-GCN: Geometric Graph Convolutional Networks, February 2020.

Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Random Features Strengthen Graph Neural Networks. In Carlotta Demeniconi and Ian Davidson, editors, *Proceedings of the 2021 SIAM International Conference on Data Mining, SDM 2021, Virtual Event, April 29 - May 1, 2021*, pages 333–341. SIAM, 2021. doi: 10.1137/1.9781611976700.38.

Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. In *International Conference on Learning Representations*, March 2022.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. In *International Conference on Learning Representations*, February 2018.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks? In *International Conference on Learning Representations*, September 2018.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do Transformers Really Perform Badly for Graph Representation? In *Advances in Neural Information Processing Systems*, May 2021.

Lingxiao Zhao and Leman Akoglu. PairNorm: Tackling Oversmoothing in GNNs. In *International Conference on Learning Representations*, March 2020.

Marinka Zitnik and Jure Leskovec. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics*, 33(14):i190–i198, July 2017. ISSN 1367-4803, 1460-2059. doi: 10.1093/bioinformatics/btx252.

Markus Zopf. 1-WL Expressiveness Is (Almost) All You Need, February 2022.