GATE: How to Keep Out Intrusive Neighbors

Nimrah Mustafa [1] **Rebekka Burkholz** [1]

Abstract

Graph Attention Networks (GATs) are designed to provide flexible neighborhood aggregation that assigns weights to neighbors according to their importance. In practice, however, GATs are often unable to switch off task-irrelevant neighborhood aggregation, as we show experimentally and analytically. To address this challenge, we propose GATE, a GAT extension that holds three major advantages: i) It alleviates over-smoothing by addressing its root cause of unnecessary neighborhood aggregation. ii) Similarly to perceptrons, it benefits from higher depth as it can still utilize additional layers for (non-)linear feature transformations in case of (nearly) switched-off neighborhood aggregation. iii) By down-weighting connections to unrelated neighbors, it often outperforms GATs on real-world heterophilic datasets. To further validate our claims, we construct a synthetic test bed to analyze a model's ability to utilize the appropriate amount of neighborhood aggregation, which could be of independent interest.

1. Introduction

Graph neural networks (GNNs) (Gori et al., 2005) are a standard class of models for machine learning on graph-structured data that utilize node feature and graph structure information jointly to achieve strong empirical performance, particularly on node classification tasks. Input graphs to GNNs stem from various domains of real-world systems such as social (Bian et al., 2020), commercial (Zhang & Chen, 2020), academic (Hamaguchi et al., 2017), economic (Monken et al., 2021), biochemical(Kearnes et al., 2016), physical (Shlomi et al., 2021), and transport (Wu et al., 2019) networks that are diverse in their node feature and graph structure properties.

[1]CISPA Helmholtz Center for Information Security, 66123 Saarbrücken, Germany. Correspondence to: Nimrah Mustafa <nimrah.mustafa@cispa.de>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

The message-passing mechanism of GNNs (Kipf & Welling, 2017; Xu et al., 2019) involves two key steps: a transformation of the node features, and the aggregation of these transformed features from a node's neighborhood to update the node's representation during training. While this has proven to be largely successful in certain cases, it generally introduces some problems for learning with GNNs, the most notorious of which is over-smoothing (Li et al., 2018). The enforced use of structural information in addition to node features may be detrimental to learning the node classification task, as shown by recent results where state-of-the-art GNNs perform the same as or worse than multi-layer perceptrons (MLPs) (Gomes et al., 2022; Yan et al., 2022; Ma et al., 2022). One such task is where node labels can be easily determined by informative node features and require no contribution from the neighborhood. Here, standard neighborhood aggregation, as in most GNN architectures, would impair model performance, particularly with an increase in model depth.

A popular standard GNN architecture that, in principle, tries to resolve this problem is the Graph Attention Network (GAT) (Veličković et al., 2018; Brody et al., 2022). By design, neighborhood aggregation in GATs is characterized by learnable coefficients intended to assign larger weights to more important neighboring nodes (including the node itself) in order to learn better node representations. Therefore, in the above example, GATs should ideally resort to assigning near-zero importance to neighbor nodes, effectively switching off neighborhood aggregation. However, we find that, counter-intuitively, GATs are unable to do this in practice and continue to aggregate the uninformative features in the neighborhood which impedes learning.

One may ask why one would employ a GAT (or any GNN architecture) if an MLP suffices. In practice, we do not know whether neighborhood aggregation (of raw features or features transformed by a perceptron or MLP), would be beneficial or not beforehand. This raises a pertinent question for the GNN research community: *How much neighborhood aggregation is needed for a given task?*. Ideally, it is what we would want a model to learn. Otherwise, the right task-specific architecture would need to be identified by time and resource-intensive manual tuning.

We address the challenge faced by GAT to effectively de-

termine how well a node is represented by its own features compared to the features of neighboring nodes, i.e., distinguish between the relative importance of available node features and graph structure information for a given task.

Firstly, we provide an intuitive explanation for the problem based on a conservation law of GAT gradient flow dynamics derived by Mustafa & Burkholz (2023). Building on this insight, we present GATE, an extension of the GAT architecture that can switch neighborhood aggregation on and off as necessary. This allows our proposed architecture to gain the following advantages over GAT:

1. It alleviates the notorious over-smoothing problem by addressing the root cause of unnecessarily repeated neighborhood aggregation.
2. It allows the model to benefit from more meaningful representations obtained solely by deeper non-linear transformations, similarly to perceptrons, in layers with little to no neighborhood aggregation.
3. It often outperforms GATs on real-world heterophilic datasets by weighing down unrelated neighbors.
4. It offers interpretable learned self-attention coefficients, at the node level, that are indicative of the relative importance of feature and structure information in the locality of the node.

In order to validate these claims, we construct a synthetic test bed of two opposite types of learning problems for node classification where label-relevant information is completely present only in a node's i) own features and ii) neighboring nodes' features (see Fig. 2). GATE is able to adapt to both cases as necessary. On real-world datasets, GATE performs competitively on homophilic datasets and is substantially better than GAT on heterophilic datasets. Furthermore, up to our knowledge, it achieves a new state of the art on the relatively large OGB-arxiv dataset (Hu et al., 2021) (i.e., $79.57 \pm 0.84\%$ test accuracy). In summary, our contributions are as follows:

- We identify and experimentally demonstrate a structural limitation of GAT, i.e., its inability to switch off neighborhood aggregation.
- We propose GATE, an extension of GAT, that overcomes this limitation and, in doing so, unlocks several benefits of the architecture.
- We update an existing conservation law relating the structure of gradients in GAT to GATE.
- We construct a synthetic test bed to validate our claims, which could be of independent interest to measure progress in developing adaptive neighborhood aggregation schemes.

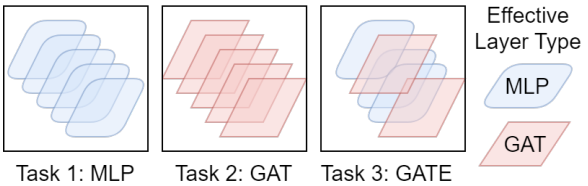


Figure 1: MLP only performs node feature transformations, whereas GAT also always aggregates over the neighborhood. With the ability to switch off neighborhood aggregation, GATE can learn to emulate MLP behavior and potentially interleave effective perceptron and standard layers in a flexible manner. This allows for more expressive power that we find to benefit real-world tasks (see Table 3).

2. Related Work

To relieve GNNs from the drawbacks of unnecessarily repeated neighborhood aggregation in deeper models, initial techniques were inspired by classical deep learning of MLPs such as normalization (Cai et al., 2021; Zhao & Akoglu, 2020; Zhou et al., 2020; 2021) and regularization (Papp et al., 2021; Rong et al., 2020; Yang et al., 202; Zou et al., 2019).

More recently, the need for deeper models and architectural changes to limit neighborhood aggregation as necessary has been recognized. Some approaches use linear combinations of initial features and current layer representation (Gasteiger et al., 2019), add skip connections and identity mapping (Chen et al., 2020; Cong et al., 2021), combine representations of all previous layers at the last layer (Xu et al., 2018), aggregate information from a node-wise defined range of k-hop neighbors(Liu et al., 2020), and limit the number of aggregation iterations based on node influence scores (Zhang et al., 2021). However, these architectures are not flexible enough to utilize additional network layers to simulate perceptron behavior, which, as we find, benefits heterophilic tasks. (Ma et al., 2023) discuss 'good' and 'bad' heterophily, that are also task-dependent.

Other contemporary works for general GNNs propose the use of bilevel optimization to determine a node's strategic discrete action to a received message (Finkelshtein et al., 2023) and a variational inference framework for adaptive message passing (Errica et al., 2023). While this non-attentive architectures improve message passing in GNNs, we focus on identifying and explaining a structural limitation of self-attention in GATs, that continue to be used as a strong baseline architecture.

An orthogonal line of research uses graph structural learning (Yang et al., 2019; Stretcu et al., 2019; Franceschi et al., 2020) to amend the input graph structure such that neighborhood aggregation is beneficial for the given task. Such

approaches are difficult to scale, more susceptible to over-smoothing, and potentially destroy any inherent information in the original graph structure. On the contrary, a standard GNN architecture empowered to selectively perform neighborhood aggregation avoids these pitfalls. Self-supervision of the attention mechanism has also been proposed (Wang et al., 2019; Kim & Oh, 2021). Methods such as graph rewiring (Deac et al., 2022) to overcome problems such as over-squashing (Alon & Yahav, 2021) are complementary and may also be combined with GATE.

While we focus our insights on GAT, architectures based on GAT such as ωGAT (Eliasof et al., 2023) also suffer from the same problem (see Fig. 14 in Appendix C). This further confirms that the universal problem with GAT has been correctly identified. In general, recent works direct effort to understand the current limitations of graph attention (Lee et al., 2023; Fountoulakis et al., 2023).

3. Architecture

Notation Consider a graph $G = (V, E)$ with node set $\mathbb{V}$ and edge set $\mathbb{E} \subseteq \mathbb{V} \times \mathbb{V}$, where for a node $v \in \mathbb{V}$ the neighborhood is $\mathbb{N}(v) = \{u|(u, v) \in \mathbb{E}\}$ and input features are $\mathbf{h}_v^0$. A GNN layer updates each node's representation by aggregating over its neighbors' representation and combining it with its own features. The aggregation and combination steps can be performed together by introducing self-loops in G such that, $\forall v \in \mathbb{V}$, $(v, v) \in \mathbb{E}$. We assume the presence of self-loops in G unless specified otherwise. In GATs, this aggregation is weighted by parameterized attention coefficients α_{uv}, which indicate the importance of node u for v. A network is constructed by stacking L layers, defined as follows, using a non-linear activation function ϕ that is homogeneous (i.e $\phi(x) = x\phi'(x)$) and consequently, $\phi(ax) = a\phi(x)$ for positive scalars a) such as ReLU $\phi(x) = \max\{x, 0\}$ or LeakyReLU $\phi(x) = \max\{x, 0\} + -\alpha \max\{-x, 0\}$.

GAT Given input representations $\mathbf{h}_v^{l-1}$ for $v \in \mathbb{V}$, a GAT [1] layer $l \in [L]$ transforms those to:

$$\mathbf{h}_v^l = \phi\left(\sum_{u\in\mathbb{N}(v)} \alpha_{uv}^l \cdot \mathbf{W}_s^l \mathbf{h}_u^{l-1}\right), \quad \text{where} \tag{1}$$

$$\alpha_{uv}^l = \frac{\exp\left(e_{uv}^l\right)}{\sum_{u'\in\mathbb{N}(v)} \exp\left(e_{u'v}^l\right)}, \quad \text{and} \tag{2}$$

$$e_{uv}^l = \left(\mathbf{a}^l\right)^\top \cdot \phi\left(\mathbf{W}_s^l \mathbf{h}_u^{l-1} + \mathbf{W}_t^l \mathbf{h}_v^{l-1}\right) \tag{3}$$

The feature transformation weights $\mathbf{W}_s$ and $\mathbf{W}_t$ for source and target nodes, respectively, may also be shared such that $\mathbf{W}_s = \mathbf{W}_t$. We denote this variant of GAT by GAT$_S$.

[1] Throughout, we refer to GATv2 (Brody et al., 2022) as GAT.

GATE In addition, we propose GATE, a GAT variant that flexibly weights the importance of node features and neighborhood features. A GATE layer is also defined by Eq. (1) and (2) but modifies e_{uv} in Eq. (3) to Eq. (4).

$$e_{uv}^l = \left(\mathbb{1}_{u\neq v}\mathbf{a}_s^l + \mathbb{1}_{u=v}\mathbf{a}_t^l\right)^\top \cdot \phi\left(\mathbf{U}^l\mathbf{h}_u^{l-1} + \mathbf{V}^l\mathbf{h}_v^{l-1}\right). \tag{4}$$

We denote e_{uv} in Eq. (3) and (4) as e_{vv}^l if $u = v$. For GATE, $\mathbf{W}_s^l$ in Eq. (1) is denoted as $\mathbf{W}^l$. A weight-sharing variant of GATE, GATE$_S$, is characterized by all feature transformation parameters being shared in a layer (i.e. $\mathbf{W}^l = \mathbf{U}^l = \mathbf{V}^l$). Note that, then, for a d-dimensional layer, GATE adds only d more parameters to GAT.

We next present theoretical insights into the reasoning behind the inability of GATs to switch off neighborhood aggregation, which is rooted in norm constraints imposed by the inherent conservation law for GATs. The gradients of GATE fulfill an updated conservation law (Thoerem 4.3) that enables switching off neighborhood aggregation in a parameter regime with well-trainable attention.

4. Theoretical Insights

For simplicity, we limit our discussion here to GATs with weight sharing. We derive similar arguments for GATs without weight sharing in Appendix A.1. The following conservation law was recently derived for GATs to explain trainability issues of standard initialization schemes. Even with improved initializations, we argue that this law limits the effective expressiveness of GATs and their ability to switch off neighborhood aggregation when necessary.

Theorem 4.1 (Thm. 2.2 by Mustafa & Burkholz (2023))**.** *The parameters θ of a layer $l \in [L-1]$ in a GAT network and their gradients $\nabla_\theta \mathcal{L}$ w.r.t. loss $\mathcal{L}$ fulfill:*

$$\langle \mathbf{W}_{[i,:]}^l, \nabla_{\mathbf{W}_{[i,:]}^l} \rangle = \langle \mathbf{W}_{[:,i]}^{l+1}, \nabla_{\mathbf{W}_{[:,i]}^{l+1}} \rangle + \langle \mathbf{a}_{[i]}^l, \nabla_{\mathbf{a}_{[i]}^l} \rangle. \tag{5}$$

Intuitively, this equality limits the budget for the relative change of parameters and imposes indirectly a norm constraint on the parameters. Under gradient flow that assumes infinitesimally small learning rates, this law implies that the relationship $\left\|\mathbf{W}^l[i,:]\right\|^2 - \left\|\mathbf{a}^l[i]\right\|^2 - \left\|\mathbf{W}^{l+1}[:i]\right\|^2 = c$ stays constant during training, where c is defined by the initial norms. Other gradient-based optimizers fulfill this norm balance also approximately. Note that the norms $\left\|\mathbf{W}^l[i,:]\right\|$ generally do not assume arbitrary values but are determined by the required scale of the output. Deeper models are especially less flexible in varying these norms as deviations could lead to exploding or diminishing outputs and/or gradients. In consequence, the norms of the attention parameters are also bounded. Furthermore, a parameter becomes harder to change during training when its

magnitude increases. This can be seen by transforming the law with respect to the relative change of a parameter defined as $\Delta\theta = \nabla_\theta \mathcal{L}/\theta$ for $\theta \neq 0$ or $\Delta\theta = 0$ for $\theta = 0$.

$$\sum_{j=1}^{n_{l+1}} {\mathbf{W}_{ij}^l}^2 \Delta \mathbf{W}_{ij}^l = \sum_{k=1}^{n_{l+2}} {\mathbf{W}_{ki}^{l+1}}^2 \Delta \mathbf{W}_{ki}^{l+1} + {\mathbf{a}_i^l}^2 \Delta \mathbf{a}_i^l. \quad (6)$$

The higher the magnitude of an attention parameter $(\mathbf{a}_i^l)^2$, the smaller will be the relative change $\Delta \mathbf{a}_i^l$ and vice versa. This restricts the attention mechanism in the network to a less-trainable regime without converging to a meaningful model. We next explain why large $\left\|\mathbf{a}^l\right\|$ are required to switch off neighborhood aggregation in a layer.

Insight 4.2 (Effective expressiveness of GATs)**.** GATs are challenged to switch off neighborhood aggregation during training, as this requires the model to enter a less trainable regime with large attention parameters $\|\mathbf{a}\|^2 >> 1$.

An intuitive derivation of this insight is presented in the appendix. Here, we outline the main argument based on the observation that to make the contribution of neighbor j insignificant relative to node i, we require $\alpha_{ij}/\alpha_{ii} << 1$. We use relative α_{ij}/α_{ii} instead of α_{ij} and α_{ii} to cancel out normalization constants and simplify the analysis.

Our key observation is that, given an insignificant link (i, j), its relative contribution to its two neighborhoods $\alpha_{ij}/\alpha_{ii} << 1$ and $\alpha_{ji}/\alpha_{jj} << 1$ are affected in opposite ways by a feature f of the attention parameters $\mathbf{a}$, i.e. if $\mathbf{a}[f]$ contributes to reducing α_{ij}/α_{ii}, it automatically increases α_{ji}/α_{jj}. However, we require multiple features that contribute to reducing only α_{ij} without strengthening α_{ji} that may only be possible in a high dimensional space requiring large norms of $\mathbf{a}$. Yet, the norms $\|\mathbf{a}\|^2$ are constrained by the parameter initialization and cannot increase arbitrarily due to the derived conservation law. Note that to switch off all neighborhood aggregation, we require $\alpha_{ij}/\alpha_{ii} << 1, \forall\, j \in \mathbb{N}(i)$, further complicating the task.

To address this challenge, we modify the GAT architecture by GATE that learns separate attention parameters for the node and the neighborhood contribution. As its conservation law indicates, it can switch off neighborhood aggregation in the well-trainable parameter regime.

Theorem 4.3 (Structure of GATE gradients)**.** *The parameters and gradients of a GATE network w.r.t. to loss $\mathcal{L}$ for layer $l \in [L-1]$ are conserved according to the following laws. Given $\Theta(\theta) = \langle \theta, \nabla_\theta \mathcal{L} \rangle$, it holds that:*

$$\Theta(\mathbf{W}_{[i,:]}^l) - \Theta({\mathbf{a}_s^{l+1}}_{[i]}) - \Theta({\mathbf{a}_t^{l+1}}_{[i]}) = \Theta(\mathbf{W}_{[:,i]}^{l+1}). \quad (7)$$

and, if additional independent matrices $\mathbf{U}^l$ and $\mathbf{V}^l$ are trainable, it also holds that:

$$\Theta(\mathbf{a}_{s\,[i]}^l) + \Theta(\mathbf{a}_{t\,[i]}^l) = \Theta({\mathbf{U}^l}_{[i,:]}) + \Theta({\mathbf{V}^l}_{[i,:]}). \quad (8)$$

The proof is provided in the appendix. We utilize this theorem for two purposes. Firstly, it induces an initialization that enables at least the initial trainability of the network. Similarly to GAT (Mustafa & Burkholz, 2023), we initialize all $\mathbf{a}$ parameters with zero and matrices $\mathbf{W}$ with random orthogonal looks-linear structure in GATE. This also ensures that we have no initial inductive bias or preference for specific neighbor or node features. As an ablation, we also verify that the initialization of the attention parameters in GAT with zero alone can not switch off neighborhood aggregation in GAT (see Fig. 7 in Appendix C).

Secondly, the conservation law leads to the insight that a GATE network is more easily capable of switching off neighborhood aggregation or node feature contributions in comparison with GAT.

Insight 4.4 (GATE is able to switch off neighborhood aggregation.)**.** GATE can flexibly switch off neighborhood aggregation or node features in the well-trainable regime of the attention parameters.

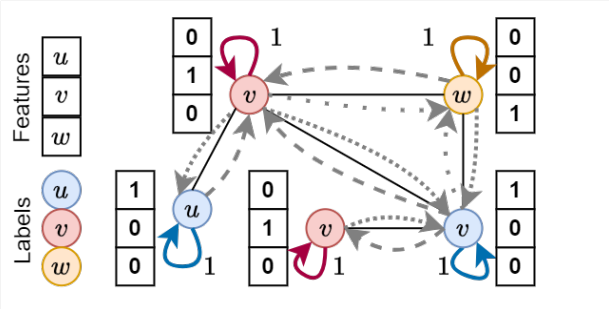
This insight follows immediately from the related conservation law for GATE that shows that $\mathbf{a}_t$ and $\mathbf{a}_s$ can interchange the available budget for relative change among each other. Furthermore, the contribution of neighbors and the nodes are controlled separately. We show how the respective switch-off can be achieved with relatively small attention parameter norms that correspond to the well-trainable regime in Appendix A.3. To verify these insights in experiments, we next design synthetic data generators that can test the ability of GNNs to take structural infromation into account in a task-appropriate manner.

5. Experiments

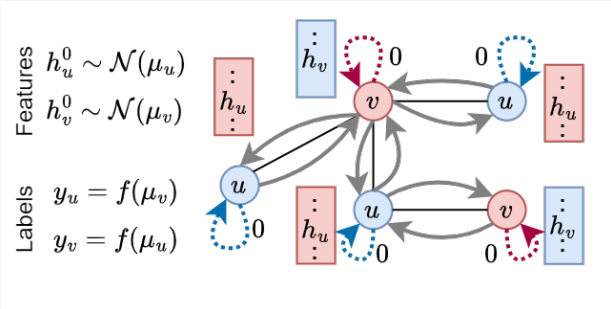
We validate the ability of GATE to perform the appropriate amount of neighborhood aggregation, as relevant for the given task and input graph, on both synthetic and real-world graphs. In order to gauge the amount of neighborhood aggregation, we study the distribution of α_{vv} values (over the nodes) at various epochs during training and layers in the network. This serves as a fair proxy since $\forall\, v \in \mathbb{V}$, $\alpha_{vv} = 1 - \sum_{u \in \mathbb{N}(v), u \neq v} \alpha_{uv}$. Thus, $\alpha_{vv} = 1$ implies no neighborhood aggregation (i.e. only $\mathbf{h}_v$ is used) whereas $\alpha_{vv} = 0$ implies only neighborhood aggregation (i.e. $\mathbf{h}_v$ is not used). Figure 2 shows an exemplary construction of both these cases. We defer a discussion of the experimental setup to Appendix B.

5.1. Synthetic Test Bed

We construct the synthetic test bed as a node classification task for two types of problems: *self-sufficient* learning and *neighbor-dependent* learning. In the self-sufficient learning problem, complete label-relevant information is present in



(a) No neighborhood contribution required.



(b) Only neighborhood contribution required.

Figure 2: Examples of synthetic input graphs constructed for learning tasks that are (a) self-sufficient and can be better solved by switching off neighborhood aggregation, i.e. $\alpha_{vv} = 1$ and (b) neighbor-dependent that benefit from ignoring the node's own features, i.e. $\alpha_{vv} = 0$. In both cases, $\forall\, v \in \mathbb{V}$, $\sum_{u \in \mathbb{N}(v), u \neq v} \alpha_{uv} + \alpha_{vv} = 1$. These represent opposite ends of the spectrum whereas real-world tasks often lie in between and require $\alpha_{ii} \in [0, 1]$. GATE's attention mechanism is more flexible than GAT's in learning the level of neighborhood aggregation required for a task.

a node's own features. On the contrary, in the neighbor-dependent learning problem, label-relevant information is present in the node features of the k-hop neighbors. We discuss both cases in detail, beginning with the simpler one.

Learning self-sufficient node labels In order to model this task exactly, we generate an Erdős–Rényi (ER) graph structure G with $N = 1000$ nodes and edge probability $p = 0.01$. Node labels y_v are assigned uniformly at random from $C = [2, 8]$ classes. Input node features $\mathbf{h}_v^0$ are generated as one-hot encoded node labels in both cases, i.e., $\mathbf{h}_v^0 = \mathbf{1}_{y_v}$. Nodes are divided randomly into train/validation/test split with a $2:1:1$ ratio.

We also use a real-world graph structure of the Cora dataset. Two cases using this graph structure are tested: i) using the original node labels consisting of 7 classes, and ii) randomized labels of 7 classes. Input node features are generated as one-hot encoding of node labels in both cases. The standard splits of Cora are used.

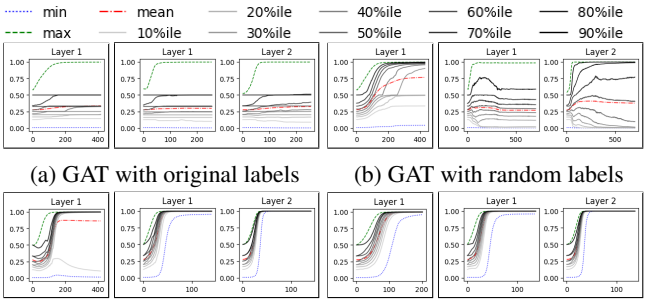
Table 1: Self-sufficient learning: S, C and L denote graph structure, number of label classes, and number of network layers, respectively. Original (Orig.) and randomized (Rand.) labels are used for the Cora structure. In all cases, 100% train accuracy is achieved except in ones marked with * and GATE eventually achieves 100% test accuracy a few epochs later except in one marked with ‡. GAT$_S$ and GAT models marked with † also eventually achieve 100% test accuracy. GATE$_S$ behaves similarly to GATE and achieves 100% train and test accuracy.

S	C	L	Test Acc.@Epoch of Min. Train Loss		
			GAT$_S$	GAT	GATE
Cora	O,7	1	**99.1@215**†	97.7@166†	99.0@127
		2	93.4@218	94.5@158	**99.6@35**
		5	85.9@92	85.5@72	**98.4@36**‡
	R,7	1	99.4@263†	99.8@268†	**100@104**
		2	61.7@2088*	52.8@341*	**99.9@36**
		5	35.1@609	32.1@1299	**99.9@23**
ER($p = 0.01$)	R,2	1	100@341†	**100@182**†	100@1313
		2	99.2@100†	99.2@119†	**99.6@79**
		5	64.0@7778*	99.6@239	**100@45**
	R,8	1	88.8@9578*	98.4@3290	**99.2@1755**
		2	90.4@2459*	94.8@2237	**99.6@44**
		5	23.6@8152	26.0@8121	**100@28**

As evident in Table 1, GAT is unable to perfectly learn this task whereas GATE easily achieves 100% train and test accuracy, and often in fewer training epochs.

In line with the homophilic nature of Cora, GAT achieves reasonably good accuracy when the original labels of the Cora graph structure are used as neighborhood aggregation is relatively less detrimental, particularly in a single-layer model. Nevertheless, in the same case, GATE generalizes better than GAT with an increase in model depth. This indicates that over-smoothing, a major cause of performance degradation with model depth in GNNs, is also alleviated due to reduced neighborhood aggregations (see Fig. 3).

On the contrary, random labels pose a real challenge to GAT. Since the neighborhood features are fully uninformative about a node's label in the randomized case, aggregation over such a neighborhood distorts the fully informative features of the node itself. This impedes the GAT network from learning the task, as it is unable to effectively switch off aggregation (see Fig. 3), whereas GATE is able to adapt to the required level of neighborhood aggregation (i.e. none, in this case). Interestingly, note that a single



(a) GAT with original labels (b) GAT with random labels

(c) GATE with original labels (d) GATE with random labels

Figure 3: Distribution of α_{vv} against training epoch for self-sufficient learning problem using Cora structure and input node features as the one-hot encoding of labels for 1 and 2 layer models. Due to space limitation, we defer the plots of 5 layer networks to Fig. 8 in Appendix C.

layer GAT in the case of random labels can almost, though not completely, switch off neighborhood aggregation (see Fig. 3) and achieve (near) perfect accuracy in the simpler cases. This is in line with our theoretical analysis (see Insight 4.2), as the norms of a single-layer model are not constrained and thus the attention parameters have more freedom to change.

Overall, the accuracy of GAT worsens drastically along two dimensions simultaneously: i) an increase in the depth of the model (due to increased unnecessary aggregation), and ii) an increase in the complexity of the task (due to an increase in the number of classes in an ER graph and consequently in node neighborhoods). In the interest of space, we defer results for GATE$_S$ to Fig. 9 in Appendix C as aggregation patterns similar to GATE are observed.

Having established that GATE excels GAT in avoiding task-irrelevant neighborhood aggregation, it is also important to verify whether GATE can perform task-relevant neighborhood aggregation *when* required, and *as much as* required. We answer this question next by studying the behavior of GATE, in comparison to GAT, on a synthetically constructed neighbor-dependent learning problem.

Learning neighbor-dependent node labels To model this task, we generate an ER graph structure with $N = 1000$ nodes and edge probability $p = 0.01$. Input node features $\mathbf{h}_v^0 \in \mathbb{R}^d$ are sampled from a multivariate normal distribution $\mathcal{N}(\mathbf{0}_d, \mathbf{I}_d)$. For simplicity, $d = 2$.

This input graph G is fed to a random GAT network M_k with k layers of width d. Note that this input graph G has no self-loops on nodes (i.e. $v \notin \mathbb{N}(v)$).The parameters of M_k are initialized with the standard Xavier (Glorot & Bengio, 2010) initialization. Thus, for each node v, the node embedding output by M_k, $\mathbf{h}_v^{M_k}$ is effectively a function f of the k-hop neighboring nodes of node v represented by a random GAT network. Let $\mathbb{N}_k(v)$ denote the set of k-hop neighbors of v and $v \notin \mathbb{N}_k(v)$.

Table 2: Neighbor-dependent learning: k and L denote the number of aggregation steps of the random GAT used for label generation and the number of layers of the evaluated network, respectively. Mean test accuracy $\pm 95\%$ confidence interval over 5 runs is reported. In all cases, a GATE variant outperforms the GAT variants. We further analyze one experimental run in detail in Table 8 in Appendix C.

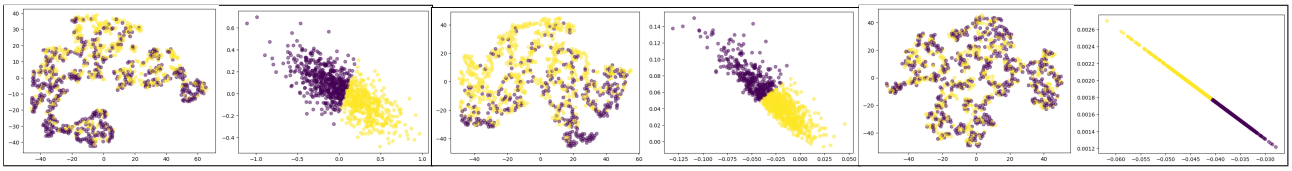
k	L	GAT$_S$	GAT	GATE$_S$	GATE
	1	93.6±1.3	92.3±1.3	**96.4 ± 0.7**	93.5±1.3
1	2	93.5±0.7	92.7±2.7	**97.9 ± 0.8**	94.6±2.1
	3	88.2±4.9	91.8±3.4	92.1±4.6	**94.0 ± 1.5**
	2	90.4±1.3	87.7±1.6	**93.8 ± 0.5**	88.7±2.5
2	3	82.2±4.5	88.9±2.1	85.8±2.5	**93.4 ± 3.3**
	4	84.0±5.0	83.0±4.8	**89.2 ± 2.3**	87.8±2.4
	3	84.3±3.2	83.8±2.7	87.5±1.8	**88.6 ± 2.0**
3	4	71.4±3.9	75.9±7.6	**89.2 ± 1.0**	89.0±0.5
	5	80.2±4.8	83.9±2.2	86.1±0.8	**87.8 ± 1.6**

Finally, we run K-means clustering on the neighborhood aggregated representation of nodes $\mathbf{h}_v^{M_k}$ to divide nodes into C clusters. For simplicity, we set $C = 2$. This clustering serves as the node labels (i.e. $y_v = \arg_{c \in [C]}(v \in c)$ for our node classification task. Thus, the label y_v of a node v to be learned is highly dependent on the input features of the neighboring nodes $\mathbf{h}_u^0 \in \mathbb{N}_k(v)$ rather than the node's own input features $\mathbf{h}_v^0$.

The generated input data and the real decision boundary for varying k are shown in Fig. 4. Corresponding results in Table 2 and Fig. 5 exhibit that GATE can better detect the amount of necessary neighborhood aggregation than GAT. However, this task is more challenging than the previous one, and GATE too can not achieve perfect 100% test accuracy. This could be attributed to data points close to the real decision boundary which is not crisply defined (see Fig. 4).

5.2. Real-World Data

To demonstrate that the ability of GATE to switch-off neighborhood aggregation has real application relevance, we evaluate GATE on relatively large-scale real-world node classification tasks, namely on five heterophilic benchmark datasets (Platonov et al., 2023) (see Table 3) and three OGB datasets (Hu et al., 2021) (see Table 5). We defer results and discussion on five small-scale datasets with varying homophily levels to Table 7 in Appendix C. To analyze



(a) $k = 1$ (b) $k = 2$ (c) $k = 3$

Figure 4: (a)-(c): Distribution of node labels of a synthetic dataset, with neighbor-dependent node labels, based on nodes' own random features (left) and neighbors' features aggregated k times (right).

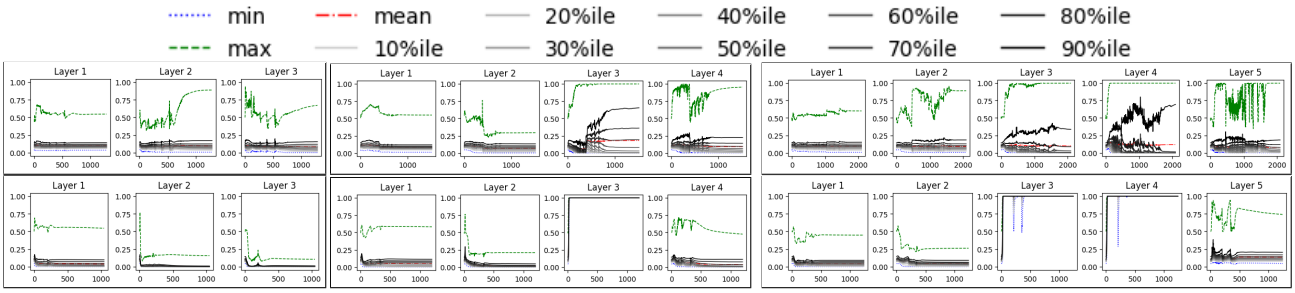


Figure 5: Distribution of α_{vv} against training epoch for the neighbor-dependent learning problem with $k = 3$. Rows: GAT (top) and GATE (bottom) architecture. Columns (left to right): 3, 4, and 5 layer models. While GAT is unable to switch off neighborhood aggregation in any layer, only 3 layers of the 4 and 5 layer models perform neighborhood aggregation.

the potential benefits of combining MLP and GAT layers in GATE, we compare its behavior with GAT and MLP. We argue that the better performance of GATE, by a large margin in most cases, can be attributed to down-weighting unrelated neighbors, leveraging deeper non-linear feature transformations, and reducing over-smoothing.

While we focus our exposition on the neighborhood aggregation perspective of GATs, we also consider the FAGCN architecture (Bo et al., 2021), which relies on a similar attention mechanism and, in theory, could switch off neighborhood aggregation when positive and negative contributions of neighbors cancel out. In contrast to GATE, it requires tuning a hyperparameter ϵ, which controls the contribution of raw node features to each layer. Furthermore, on our synthetic tasks, we find that, like GAT, FAGCN is also unable to limit neighborhood contribution. We also provide a detailed qualitative and quantitative discussion comparing GATE and FAGCN using the synthetic testbed in Appendix C.

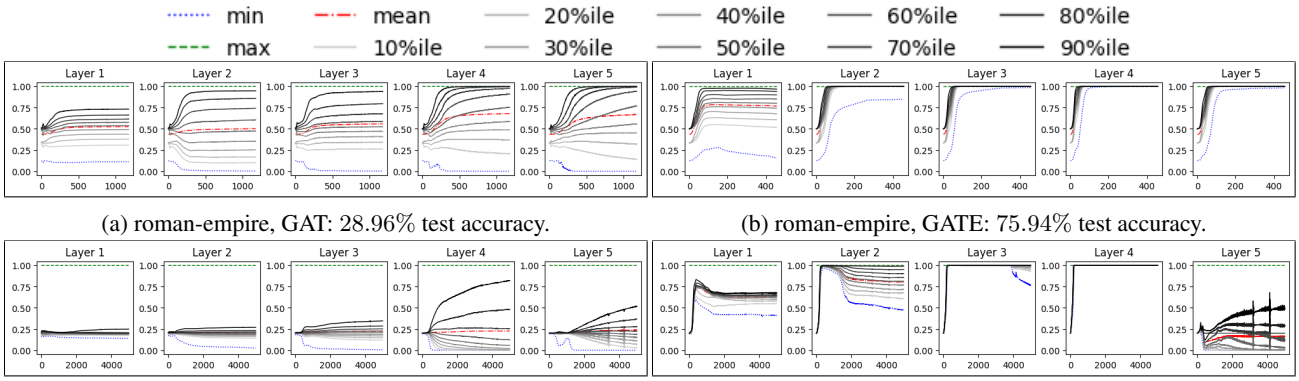
Next, we analyze GATE's ability to mix MLP and GAT layers. To this end, we evaluate a GNN architecture constructed by alternately placing GAT and MLP layers in the network that we denote by MLP_{+GAT} on various heterophilic tasks. The purpose of this experiment is twofold. Firstly, we observe in Table 3 that MLP_{+GAT} outperforms both GAT and MLP in most cases. This highlights the benefit of only performing non-linear transformations on raw or aggregated neighborhood features without immediate further neighborhood aggregation to learn potentially more complex features. Secondly, we find GATE to outperform MLP_{+GAT} (see Table 3). This illustrates that rigidly embedding MLP layers in a GNN with arbitrary predefined roles is not ideal as the appropriate degree and placement of neighborhood aggregation is unknown a-priori. In contrast, GATE offers more flexibility to learn intricate combinations of GNN layers and nonlinear feature transformations that define more adequate models for a given task, as exemplified in Fig. 6.

The distributions of α_{vv} in Fig. 6 across layers in GATE reveal information about the relative importance of node feature and graph structure at the node level, which allows us to analyze the question to which degree graph information is helpful for a task. For example, in Fig. 6, we observe that the α_{vv} values are mostly lower in the minesweeper dataset than the roman-empire dataset. This indicates that aggregation, particularly over the input node features and the final layer's learned representations, is more beneficial compared to the node's own features for the minesweeper dataset. On the other hand, for roman-empire, the model has a higher preference to utilize features of the node itself (as most values of α_{vv} approach 1) over features of the neighbors. This aligns with the homophily levels, 0.05 and 0.68, of the roman-empire and minesweeper datasets, respectively. A similar analysis for datasets Texas and Actor can be found in Fig. 13 in Appendix C.

We also observe in Fig. 6 that when neighborhood aggrega-

Table 3: We report mean test accuracy $\pm 95\%$ confidence interval for roman-empire and amazon-ratings and AUC-ROC for the other three datasets over the standard 10 splits, following (Platonov et al., 2023). All architectures were run with networks of depth 5 and 10 layers. The better performance for each architecture is shown with the number of network layers used in parentheses. GATE outperforms GAT and other baselines on all datasets, mostly by a significant margin.

	roman-empire	amazon-ratings	questions	minesweeper	tolokers
GAT	26.10 ± 1.25 (5)	45.58 ± 0.41 (10)	57.72 ± 1.58 (5)	50.83 ± 0.41 (5)	63.57 ± 1.03 (10)
MLP	65.12 ± 0.25 (5)	43.26 ± 0.34 (5)	59.44 ± 0.94 (10)	50.74 ± 0.56 (5)	62.67 ± 1.06 (10)
MLP$_{+GAT}$	70.83 ± 0.39 (5)	45.25 ± 0.17 (10)	59.12 ± 1.57 (10)	60.07 ± 1.11 (5)	65.85 ± 0.64 (10)
FAGCN	67.55 ± 0.81 (5)	42.85 ± 0.83 (10)	60.38 ± 1.21 (5)	63.38 ± 0.91 (10)	60.89 ± 1.12 (5)
GATE	$\mathbf{75.55 \pm 0.30}$ (5)	$\mathbf{45.73 \pm 0.24}$(10)	$\mathbf{62.95 \pm 0.71}$ (5)	$\mathbf{66.14 \pm 1.57}$ (5)	$\mathbf{66.63 \pm 1.15}$(10)



(a) roman-empire, GAT: 28.96% test accuracy.

(b) roman-empire, GATE: 75.94% test accuracy.

(c) Minesweeper, GAT: 50.50% test AUROC.

(d) Minesweeper, GATE: 67.57% test AUROC.

Figure 6: Distribution of α_{vv} against training epoch for a run of 5 layer networks on real-world heterophilic tasks. As expected, GAT is unable to significantly vary neighborhood aggregation whereas GATE exhibits varying aggregation patterns across layers and tasks. These could be interpreted to indicate the inherent importance of raw node features relative to their neighborhoods for a given task. We defer the plots of the 10-layer models for tolokers dataset to Fig. 11 in Appendix C.

tion takes place, the level of aggregation across all nodes, as indicated by the shape of α_{vv} distribution, varies over network layers. This is expected as different nodes need different levels of aggregation depending on where they are situated in the graph topology. For example, peripheral nodes would require more aggregation than central nodes to obtain a similar amount of information. Therefore, as already observed with purposefully constructed synthetic data, GATE offers a more interpretable model than GAT in a real-world setting.

While we focus our evaluation in Table 7 on comparison with the most relevant baselines such as attention-based architectures, we next present a more extensive comparison with 14 other baseline architectures in Table 4. For the results reported in Table 7, we conduct experiments in a simple setting without additional elements that may impact the performance such as skip connections, normalization, etc., to isolate the effect of the architecture and evaluate solely the impact of GATE's ability to switch off neighborhood aggregation on real-world data. However, for the results in Table 4, we adopt the original codebase of (Platonov et al., 2023), which utilizes such elements to evaluate the performance of baseline GNNs and architectures specifically designed for heterophilic datasets. We evaluate GATE in the same settings optimized for their experiments. For easy comparison, we replicate their results from Table 4 in (Platonov et al., 2023).

We observe in Table 4 that while GATE outperforms GAT (and other baselines) significantly, GATE has comparable performance to GAT-sep, a variant of GAT, despite GATE being more parameter efficient by an order of magnitude. More specifically, GAT-sep and GATE introduce d^2 and d additional parameters, respectively, in a layer. By correspondingly adapting GATE, we find GATE-sep to achieve the best performance in most cases. Therefore, additional techniques generally employed to boost performance are compatible and complementary to GATE.

GATE's ability to benefit from depth in terms of generalization is demonstrated on OGB datasets (see Table 5). In par-

Table 4: An extensive comparison of GATE with baseline GNNs using the experimental setup of (Platonov et al., 2023). Accuracy is reported for roman-empire and amazon-ratings, and ROC AUC is reported for the remaining three datasets.

	roman-empire	amazon-ratings	minesweeper	tolokers	questions
GATE	89.51 ± 0.49	52.49 ± 0.46	92.82 ± 0.90	84.62 ± 0.69	78.46 ± 1.17
GAT	80.87 ± 0.30	49.09 ± 0.63	92.01 ± 0.68	83.70 ± 0.47	77.43 ± 1.20
GATE-sep	$\mathbf{89.78 \pm 0.54}$	$\mathbf{54.51 \pm 0.38}$	$\mathbf{94.18 \pm 0.43}$	$\mathbf{84.48 \pm 0.57}$	78.20 ± 1.00
GAT-sep	88.75 ± 0.41	52.70 ± 0.62	93.91 ± 0.35	83.78 ± 0.43	76.79 ± 0.71
GT	86.51 ± 0.73	51.17 ± 0.66	91.85 ± 0.76	83.23 ± 0.64	77.95 ± 0.68
GT-sep	87.32 ± 0.39	52.18 ± 0.80	92.29 ± 0.47	82.52 ± 0.92	78.05 ± 0.93
GCN	73.69 ± 0.74	48.70 ± 0.63	89.75 ± 0.52	83.64 ± 0.67	76.09 ± 1.27
SAGE	85.74 ± 0.67	53.63 ± 0.39	93.51 ± 0.57	82.43 ± 0.44	76.44 ± 0.62
H_2GCN	60.11 ± 0.52	36.47 ± 0.23	89.71 ± 0.31	73.35 ± 1.01	63.59 ± 1.46
CPGNN	63.96 ± 0.62	39.79 ± 0.77	52.03 ± 5.46	73.36 ± 1.01	65.96 ± 1.95
GPR-GNN	64.85 ± 0.27	44.88 ± 0.34	86.24 ± 0.61	72.94 ± 0.97	55.48 ± 0.91
FSGNN	79.92 ± 0.56	52.74 ± 0.83	90.08 ± 0.70	82.76 ± 0.61	$\mathbf{78.86 \pm 0.92}$
GloGNN	59.63 ± 0.69	36.89 ± 0.14	51.08 ± 1.23	73.39 ± 1.17	65.74 ± 1.19
FAGCN	65.22 ± 0.56	44.12 ± 0.30	88.17 ± 0.73	77.75 ± 1.05	77.24 ± 1.26
GBK-GNN	74.57 ± 0.47	45.98 ± 0.71	90.85 ± 0.58	81.01 ± 0.67	74.47 ± 0.86
JacobiConv	71.14 ± 0.42	43.55 ± 0.48	89.66 ± 0.40	68.66 ± 0.65	73.88 ± 1.16
ResNet	65.88 ± 0.38	45.90 ± 0.52	50.89 ± 1.39	72.95 ± 1.06	70.34 ± 0.76
ResNet+SGC	73.90 ± 0.51	50.66 ± 0.48	70.88 ± 0.90	80.70 ± 0.97	75.81 ± 0.96
ResNet+adj	52.25 ± 0.40	51.83 ± 0.57	50.42 ± 0.83	78.78 ± 1.11	75.77 ± 1.24

Table 5: Mean test accuracy $\pm 95\%$ confidence interval (and number of network layers). We replicate the results for GAT reported by (Brody et al., 2022). GATE leverages deeper networks to substantially outperform GAT.

OGB-	GAT	GATE
arxiv	71.87 ± 0.16 (3)	$\mathbf{79.57 \pm 0.84}$ (12)
products	80.63 ± 0.46 (3)	$\mathbf{86.24 \pm 1.01}$ (8)
mag	32.61 ± 0.29 (2)	$\mathbf{35.29 \pm .36}$ (5)

ticular, GATE improves the SOTA test accuracy (78.03%) on the arxiv dataset achieved by a model using embeddings learned by a language model instead of raw node features(Duan et al., 2023), as reported on the OGB leaderboard. While the better performance of deeper models with limited neighborhood aggregation in certain layers indicates reduced over-smoothing, we also verify this insight quantitatively (see Table 9 in Appendix C).

Our experimental code is available at `https://github.com/RelationalML/GATE.git`.

6. Conclusion

We experimentally illustrate a structural limitation of GAT that disables the architecture, in practice, to switch off task-irrelevant neighborhood aggregation. This obstructs GAT from achieving its intended potential. Based on insights from an existing conservation law of gradient flow dynamics in GAT, we have explained the source of this problem. To verify that we have identified the correct issue, we resolve it with a modification of GAT, which we call GATE, and derive the corresponding modified conservation law. GATE holds multiple advantages over GAT, as it can leverage the benefits of depth as in MLPs, offer interpretable, learned self-attention coefficients, and adapt the model to the necessary degree of neighborhood aggregation for a given task. We verify this on multiple synthetic and real-world tasks, where GATE significantly outperforms GAT and also achieves a new SOTA test accuracy on the OGB-arxiv dataset. Therefore, we argue that GATE is a suitable candidate to answer highly debated questions related to the importance of a given graph structure for standard tasks.

Acknowledgements

We gratefully acknowledge funding from the European Research Council (ERC) under the Horizon Europe Framework Programme (HORIZON) for proposal number 101116395 SPARSE-ML.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

Alon, U. and Yahav, E. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations*, 2021.

Bian, T., Xiao, X., Xu, T., Zhao, P., Huang, W., Rong, Y., and Huang, J. Rumor detection on social media with bi-directional graph convolutional networks. In *AAAI Conference on Artificial Intelligence*, 2020.

Bo, D., Wang, X., Shi, C., and Shen, H. Beyond low-frequency information in graph convolutional networks. In *AAAI Conference on Artificial Intelligence*, 2021.

Brody, S., Alon, U., and Yahav, E. How attentive are graph attention networks? In *International Conference on Learning Representations*, 2022.

Burkholz, R. and Dubatovka, A. Initialization of ReLUs for dynamical isometry. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Cai, C. and Wang, Y. A note on over-smoothing for graph neural networks. In *Graph Representation Learning Workshop, International Conference on Machine Learning*, 2020.

Cai, T., Luo, S., Xu, K., He, D., Liu, T.-Y., and Wang, L. Graphnorm: A principled approach to accelerating graph neural network training. In *International Conference on Machine Learning*, 2021.

Chen, M., Wei, Z., Huang, Z., Ding, B., and Li, Y. Simple and deep graph convolutional networks. In *International Conference on Machine Learning*, 2020.

Cong, W., Ramezani, M., and Mahdavi, M. On provable benefits of depth in training graph convolutional networks. In *Advances in Neural Information Processing Systems*, 2021.

Deac, A., Lackenby, M., and Veličković, P. Expander graph propagation. In *Learning on Graphs Conference*, 2022.

Duan, K., Liu, Q., Chua, T.-S., Yan, S., Ooi, W. T., Xie, Q., and He, J. Simteg: A frustratingly simple approach improves textual graph learning, 2023.

Eliasof, M., Ruthotto, L., and Treister, E. Improving graph neural networks with learnable propagation operators. In *International Conference on Machine Learning*, 2023.

Errica, F., Christiansen, H., Zaverkin, V., Maruyama, T., Niepert, M., and Alesiani, F. Adaptive message passing: A general framework to mitigate oversmoothing, over-squashing, and underreaching, 2023.

Finkelshtein, B., Huang, X., Bronstein, M., and İsmail İlkan Ceylan. Cooperative graph neural networks, 2023.

Fountoulakis, K., Levi, A., Yang, S., Baranwal, A., and Jagannath, A. Graph attention retrospective. In *Journal of Machine Learning Research*, 2023.

Franceschi, L., Niepert, M., Pontil, M., and He, X. Learning discrete structures for graph neural networks. In *International Conference on Machine Learning*, 2020.

Gasteiger, J., Bojchevski, A., and Günnemann, S. Predict then propagate: Graph neural networks meet personalized pagerank. In *International Conference on Learning Representations*, 2019.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics*, volume 9, pp. 249–256, May 2010.

Gomes, D., Ruelens, F., Efthymiadis, K., Nowe, A., and Vrancx, P. When are graph neural networks better than structure agnostic methods? In *Neural Information Processing Systems Workshop ICBINB*, 2022.

Gori, M., Monfardini, G., and Scarselli, F. A new model for learnig in graph domains. In *IEEE International Joint Conference on Neural Networks*, 2005.

Hamaguchi, T., Oiwa, H., Shimbo, M., and Matsumoto, Y. Knowledge transfer for out-of-knowledge-base entities : A graph neural network approach. In *International Joint Conference on Artificial Intelligence*, 2017.

Hamilton, W. L., Ying, R., and Leskovec, J. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, 2018.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. Open graph benchmark: Datasets for machine learning on graphs, 2021.

Kearnes, S., McCloskey, K., Berndl, M., Pande, V., and Riley, P. Molecular graph convolutions: Moving beyond fingerprints. In *Journal of Computer-Aided Molecular Design*, 2016.

Kim, D. and Oh, A. How to find your friendly neighborhood: Graph attention design with self-supervision. In *International Conference on Learning Representations*, 2021.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Lee, S. Y., Bu, F., Yoo, J., and Shin, K. Towards deep attention in graph neural networks: Problems and remedies. In *International Conference on Machine Learning*, 2023.

Li, Q., Han, Z., and Wu, X.-M. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI Conference on Artificial Intelligence*, 2018.

Liu, M., Gao, H., and Ji, S. Towards deeper graph neural networks. In *SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2020.

Ma, Y., Liu, X., Shah, N., and Tang, J. Is homophily a necessity for graph neural networks? In *International Conference on Learning Representations*, 2022.

Ma, Y., Liu, X., Shah, N., and Tang, J. Is homophily a necessity for graph neural networks? In *International Conference on Learning Representations*, 2023.

Monken, A., Haberkorn, F., Gopinatha, M., Freeman, L., and Batarseh, F. A. Graph neural networks for modeling causality in international trade. In *AAAI Conference on Artificial Intelligence*, 2021.

Mustafa, N. and Burkholz, R. Are GATS out of balance? In *Advances in Neural Information Processing Systems*, 2023.

Papp, P. A., Martinkus, K., Faber, L., and Wattenhofer, R. Dropgnn: Random dropouts increase the expressiveness of graph neural networks. In *Advances in Neural Information Processing Systems*, 2021.

Pei, H., Wei, B., Chang, K. C.-C., Lei, Y., and Yang, B. Geom-gcn: Geometric graph convolutional networks. In *International Conference on Learning Representations*, 2020.

Platonov, O., Kuznedelev, D., Diskin, M., Babenko, A., and Prokhorenkova, L. A critical look at the evaluation of gnns under heterophily: are we really making progress? In *International Conference on Learning Representations*, 2023.

Rong, Y., Huang, W., Xu, T., and Huang, J. Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*, 2020.

Shlomi, J., Battaglia, P., and Vlimant, J.-R. Graph neural networks in particle physics. 2021.

Stretcu, O., Viswanathan, K., Movshovitz-Attias, D., Platanios, E., Ravi, S., and Tomkins, A. Graph agreement models for semi-supervised learning. In *Advances in Neural Information Processing Systems*, 2019.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Wang, G., Ying, R., Huang, J., and Leskovec, J. Improving graph attention networks with large margin-based constraints. In *Graph Representation Learning Workshop, Neural Information Processing Systems*, 2019.

Wu, Z., Pan, S., Long, G., Jiang, J., and Zhang, C. Graph wavenet for deep spatial-temporal graph modeling. In *International Joint Conference on Artificial Intelligence*, 2019.

Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K.-i., and Jegelka, S. Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning*, 2018.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

Yan, Y., Hashemi, M., Swersky, K., Yang, Y., and Koutra, D. Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks. In *IEEE International Conference on Data Mining*, 2022.

Yang, H., Ma, K., and Cheng, J. Rethinking graph regularization for graph neural networks. In *Advances in Neural Information Processing Systems*, 202.

Yang, L., Kang, Z., Cao, X., Jin, D., Yang, B., and Guo, Y. Topology optimization based graph convolutional network. In *International Joint Conference on Artificial Intelligence*, 2019.

Zhang, M. and Chen, Y. Inductive matrix completion based on graph neural networks. In *International Conference on Learning Representations*, 2020.

Zhang, W., Yang, M., Sheng, Z., Li, Y., Ouyang, W., Tao, Y., Yang, Z., and Cui, B. Node dependent local smoothing for scalable graph learning. In *Advances in Neural Information Processing Systems*, 2021.

Zhao, L. and Akoglu, L. Pairnorm: Tackling oversmoothing in gnns. In *International Conference on Learning Representations*, 2020.

Zhou, K., Huang, X., Li, Y., Zha, D., Chen, R., and Hu, X. Towards deeper graph neural networks with differentiable group normalization. In *Advances in Neural Information Processing Systems*, 2020.

Zhou, K., Dong, Y., Wang, K., Lee, W. S., Hooi, B., Xu, H., and Feng, J. Understanding and resolving performance degradation in graph convolutional networks. In *Conference on Information and Knowledge Management*, 2021.

Zou, D., Hu, Z., Wang, Y., Jiang, S., Sun, Y., and Gu, Q. Layer-dependent importance sampling for training deep and large graph convolutional networks. In *Advances in Neural Information Processing Systems*, 2019.

A. Theoretical Derivations

A.1. Derivation of Insight 4.2

Statement (Restated Insight 4.2). GATs are challenged to switch off neighborhood aggregation during training, as this would require the model to enter a less trainable regime with large attention parameters $\|a\|^2 >> 1$.

We have to distinguish GATs with and without weight sharing in our analysis.

GATs with weight sharing:

To investigate the ability of a GAT to switch off neighborhood aggregation, let us focus on a link (i, j) that should neither contribute to the feature transformation of i nor j.

This implies that we need to find attention parameters $\mathbf{a}$ (and potentially feature transformations W) so that $\alpha_{ij}/\alpha_{ii} << 1$ with $\alpha_{ij}/\alpha_{ii} = \exp(e_{ij} - e_{ii})$. This implies that we require $e_{ij} - e_{ii} << 0$ and thus $\mathbf{a}^T\phi(\mathbf{W}(\mathbf{h}_i + \mathbf{h}_j)) - 2\mathbf{a}^T\phi(\mathbf{W}(\mathbf{h}_i)) << 0$.

Since we also require $\alpha_{ij}/\alpha_{jj} << 1$, it follows from adding both inequalities that $\mathbf{a}^T[\phi(\mathbf{W}(\mathbf{h}_i + \mathbf{h}_j)) - (\phi(\mathbf{W}\mathbf{h}_i) + \phi(\mathbf{W}\mathbf{h}_j))] << 0$.

This inequality can only be fulfilled if there exists at least one feature f for which

$$\Delta_{fij}; = a[f]\left[\phi\left(\mathbf{W}[f,:]\left(\mathbf{h}_i + \mathbf{h}_j\right)\right) - \left(\phi\left(\mathbf{W}[f,:]\mathbf{h}_i\right) + \phi\left(\mathbf{W}[f,:]\mathbf{h}_i\right)\right)\right]$$

fulfills $\Delta_{fij} << 0$. Yet, note that if both $\phi(\mathbf{W}[f,:]\mathbf{h}_i)$ and $\phi(\mathbf{W}[f,:]\mathbf{h}_j)$ are positive or both are negative, we just get $\Delta_{fij} = 0$ because of the definition of a LeakyReLU. Thus, there must exist at least one feature f so that without loss of generality $\phi(\mathbf{W}[f,:]\mathbf{h}_i) < 0$ and $\phi(\mathbf{W}[f,:]\mathbf{h}_j) > 0$.

It follows that if $a[f] > 0$ that

$$\begin{aligned} 0 &> a[f]\phi\left(\mathbf{W}[f,:]\mathbf{h}_i\right) >> a[f]\left(\phi\left(\mathbf{W}[f,:]\left(\mathbf{h}_i + \mathbf{h}_j\right)\right) - \phi\left(\mathbf{W}[f,:]\mathbf{h}_j\right)\right) \\ &> a[f]\left(\phi\left(\mathbf{W}[f,:]\left(\mathbf{h}_i + \mathbf{h}_j\right)\right) - 2\phi\left(\mathbf{W}[f,:]\mathbf{h}_j\right)\right) \end{aligned}$$

also receives a negative contribution that makes α_{ij}/α_{jj} smaller. Yet, what happens to α_{ij}/α_{ii}? By distinguishing two cases, namely $\mathbf{W}[f,:](\mathbf{h}_i + \mathbf{h}_j) > 0$ or $\mathbf{W}[f,:](\mathbf{h}_i + \mathbf{h}_j) < 0$ and computing

$$a[f]\left[\phi\left(\mathbf{W}\left(\mathbf{h}_i + \mathbf{h}_j\right)\right) - 2\phi\left(\mathbf{W}[f,:]\mathbf{h}_j\right)\right] > 0$$

we find the feature contribution to be positive.

If $a[f] < 0$, then

$$\begin{aligned} 0 &> a[f]\phi\left(\mathbf{W}[f,:]\mathbf{h}_j\right) >> a[f]\left(\phi\left(\mathbf{W}[f,:]\left(\mathbf{h}_i + \mathbf{h}_j\right)\right) - \phi\left(\mathbf{W}[f,:]\mathbf{h}_i\right)\right) \\ &> a[f]\left(\phi\left(\mathbf{W}[f,:]\left(\mathbf{h}_i + \mathbf{h}_j\right)\right) - 2\phi\left(\mathbf{W}[f,:]\mathbf{h}_i\right)\right) \end{aligned}$$

and α_{ij}/α_{jj} is reduced. Similarly, we can derive that at the same time α_{ij}/α_{ii} is increased, however.

This implies that any feature that contributes to reducing Δ_{fij} automatically increases one feature while it increases another. We therefore need multiple features f to contribute to reducing either α_{ij}/α_{ii} or α_{ij}/α_{jj} to compensate for other increases.

This implies, in order to switch off neighborhood aggregation, we would need a high dimensional space of features that cater to switching off specific links without strengthening others. Furthermore, they would need large absolute values of $a[f]$ and norms of $\mathbf{W}[f,:]$ or exploding feature vectors $\mathbf{h}$ to achieve this.

Yet, all these norms are constrained by the derived conservation law and therefore prevent learning a representation that switches off full neighborhoods.

GATs without weight sharing:

The flow of argumentation without weight sharing is very similar to the one above with weight sharing. Yet, we have to distinguish more cases.

Similarly to before, we require $\alpha_{ij}/\alpha_{jj} << 1$ and $\alpha_{ji}/\alpha_{ii} << 1$. It follows from adding both related inequalities that

$$\mathbf{a}^T \left[\phi\left(\mathbf{W}_s\mathbf{h}_i + \mathbf{W}_t\mathbf{h}_j\right) + \phi\left(\mathbf{W}_s\mathbf{h}_j + \mathbf{W}_t\mathbf{h}_i\right) - \phi\left(\left(\mathbf{W}_s + \mathbf{W}_t\right)\mathbf{h}_i\right) - \phi\left(\left(\mathbf{W}_s + \mathbf{W}_t\right)\mathbf{h}_j\right)\right] << 0.$$

This implies that for at least one feature f, we require

$$\begin{aligned} a[f][\phi\left(\mathbf{W}_s[f,:]\mathbf{h}_i + \mathbf{W}_t[f,:]\mathbf{h}_j\right) + \phi\left(\mathbf{W}_s[f,:]\mathbf{h}_j + \mathbf{W}_t[f,:]\mathbf{h}_i\right) \\ - \phi\left(\left(\mathbf{W}_s[f,:] + \mathbf{W}_t[f,:]\right)\mathbf{h}_i\right) - \phi\left(\left(\mathbf{W}_s[f,:] + \mathbf{W}_t[f,:]\right)\mathbf{h}_j\right)] << 0. \end{aligned} \tag{9}$$

Again, our goal is to show that this feature automatically decreases the contribution of one feature while it increases another. As argued above, switching off neighborhood aggregation would therefore need a high dimensional space of features that cater to switching off specific links without strengthening others. Furthermore, they would need large absolute values of $a[f]$ and norms of $\mathbf{W}[f,:]$ or exploding feature vectors $\mathbf{h}$ to achieve this. Our derived norm constraints, however, prevent learning such a model representation.

Concretely, without loss of generality, we therefore have to show that if

$$a[f][\phi\left(\mathbf{W}_s[f,:]\mathbf{h}_i + \mathbf{W}_t[f,:]\mathbf{h}_j\right) - \phi\left(\left(\mathbf{W}_s[f,:] + \mathbf{W}_t[f,:]\right)\mathbf{h}_j\right) < 0, \tag{10}$$

at the same time, we receive

$$a[f][\phi\left(\mathbf{W}_s[f,:]\mathbf{h}_j + \mathbf{W}_t[f,:]\mathbf{h}_i\right) - \phi\left(\left(\mathbf{W}_s[f,:] + \mathbf{W}_t[f,:]\right)\mathbf{h}_i\right) > 0, \tag{11}$$

(or vice versa).

In principle, we have to show this for 16 different cases of pre-activation sign configurations for the four terms in Eq. (9). Yet, since the argument is symmetric with respect to exchanging i and j, only 8 different cases remain. Two trivial cases are identical signs for all four terms. These are excluded, as the left hand side (LHS) of Eq. (9) would become zero and thus not contribute to our goal to switch off neighborhood aggregation. In the following, we will discuss the remaining six cases. Please note that for the remainder of this derivation $\alpha > 0$ denotes the slope of the leakyReLU and not the attention weights α_{ij}.

1. **Case** $(+\ -\ ++)$**:** Let us assume that $\mathbf{W}_s[f,:]\mathbf{h}_i + \mathbf{W}_t[f,:]\mathbf{h}_j > 0$, $\mathbf{W}_s[f,:]\mathbf{h}_j + \mathbf{W}_t[f,:]\mathbf{h}_i < 0$, $\left(\mathbf{W}_s[f,:] + \mathbf{W}_t[f,:]\right)\mathbf{h}_i > 0$, and $\left(\mathbf{W}_s[f,:] + \mathbf{W}_t[f,:]\right)\mathbf{h}_j > 0$.

   From this assumption and the fact that ϕ is a leakyReLU it follows that the LHS of Eq. (9) becomes: $a[f][\phi\left(\mathbf{W}_s[f,:]\mathbf{h}_i + \mathbf{W}_t[f,:]\mathbf{h}_j\right) + \phi\left(\mathbf{W}_s[f,:]\mathbf{h}_j + \mathbf{W}_t[f,:]\mathbf{h}_i\right) - \phi\left(\left(\mathbf{W}_s[f,:] + \mathbf{W}_t[f,:]\right)\mathbf{h}_i\right) - \phi\left(\left(\mathbf{W}_s[f,:] + \mathbf{W}_t[f,:]\right)\mathbf{h}_j\right)] = a[f](\alpha-1)[\mathbf{W}_s[f,:]\mathbf{h}_j + \mathbf{W}_t[f,:]\mathbf{h}_i]$. Since $\alpha - 1 < 0$ and $[\mathbf{W}_s[f,:]\mathbf{h}_j + \mathbf{W}_t[f,:]\mathbf{h}_i] < 0$ according to our assumption, Eq. (9) demands $a[f] < 0$. To switch off neighborhood aggregation, we would need to be able to make the LHS of Eq. (10) and Eq. (11) Eq. (11) negative. Yet, a negative $a[f]$ leads to a positive LHS of Eq. (11). Thus, the assumed sign configuration cannot support switching off neighborhood aggregation.

2. **Case** $(+\ -\ --)$**:** Let us assume that $\mathbf{W}_s[f,:]\mathbf{h}_i + \mathbf{W}_t[f,:]\mathbf{h}_j > 0$, $\mathbf{W}_s[f,:]\mathbf{h}_j + \mathbf{W}_t[f,:]\mathbf{h}_i < 0$, $\left(\mathbf{W}_s[f,:] + \mathbf{W}_t[f,:]\right)\mathbf{h}_i < 0$, and $\left(\mathbf{W}_s[f,:] + \mathbf{W}_t[f,:]\right)\mathbf{h}_j < 0$.

   The LHS of Eq. (9) becomes $a[f](1-\alpha)[\mathbf{W}_s[f,:]\mathbf{h}_i + \mathbf{W}_t[f,:]\mathbf{h}_j]$, which demands $a[f] < 0$. Accordingly, the LHS of Eq. (10) is clearly negative, while the LHS of Eq. (11) is $a[f]\alpha\mathbf{W}_s[f,:](\mathbf{h}_j - \mathbf{h}_i) > 0$. The last inequality follows from our assumptions that imply $\mathbf{W}_s[f,:]\mathbf{h}_j < \mathbf{W}_s[f,:]\mathbf{h}_i$ by combining the assumptions $\left(\mathbf{W}_s[f,:] + \mathbf{W}_t[f,:]\right)\mathbf{h}_j < 0$ and $\mathbf{W}_s[f,:]\mathbf{h}_i + \mathbf{W}_t[f,:]\mathbf{h}_j > 0$. Again, this result implies that the considered sign configuration does not support switching off neighborhood aggregation.

3. **Case** $(+\ +\ +-)$**:** Let us assume that $\mathbf{W}_s[f,:]\mathbf{h}_i + \mathbf{W}_t[f,:]\mathbf{h}_j > 0$, $\mathbf{W}_s[f,:]\mathbf{h}_j + \mathbf{W}_t[f,:]\mathbf{h}_i > 0$, $\left(\mathbf{W}_s[f,:] + \mathbf{W}_t[f,:]\right)\mathbf{h}_i > 0$, and $\left(\mathbf{W}_s[f,:] + \mathbf{W}_t[f,:]\right)\mathbf{h}_j < 0$.

   The LHS of Eq. (9) becomes $a[f](1-\alpha)[\mathbf{W}_s[f,:]\mathbf{h}_j + \mathbf{W}_t[f,:]\mathbf{h}_j]$, which demands $a[f] > 0$. Accordingly, the LHS of Eq. (10) becomes positive, which hampers switching-off neighborhood aggregation as discussed.

4. **Case** $(- \; - \; -+)$**:** Let us assume that $\mathbf{W}_s[f,:]\mathbf{h}_i + \mathbf{W}_t[f,:]\mathbf{h}_j < 0$, $\mathbf{W}_s[f,:]\mathbf{h}_j + \mathbf{W}_t[f,:]\mathbf{h}_i < 0$, $(\mathbf{W}_s[f,:] + \mathbf{W}_t[f,:])\,\mathbf{h}_i < 0$, and $(\mathbf{W}_s[f,:] + \mathbf{W}_t[f,:])\,\mathbf{h}_j > 0$.

   The LHS of Eq. (9) becomes $a[f](\alpha - 1)[\mathbf{W}_s[f,:]\mathbf{h}_j + \mathbf{W}_t[f,:]\mathbf{h}_j]$, which demands $a[f] > 0$. Accordingly, the LHS of Eq. (10) becomes clearly negative. However, the LHS of Eq. (11) is positive, as $a[f]\alpha\mathbf{W}_s[f,:](\mathbf{h}_j - \mathbf{h}_i) > 0$.

   The last inequality follows from our assumptions that imply $\mathbf{W}_s[f,:]\mathbf{h}_j > \mathbf{W}_s[f,:]\mathbf{h}_i$ by combining the assumptions $(\mathbf{W}_s[f,:] + \mathbf{W}_t[f,:])\,\mathbf{h}_j > 0$ and $\mathbf{W}_s[f,:]\mathbf{h}_i + \mathbf{W}_t[f,:]\mathbf{h}_j < 0$. Again, this analysis implies that the considered sign configuration does not support switching off neighborhood aggregation.

5. **Case** $(+ \; - \; +-)$**:** Let us assume that $\mathbf{W}_s[f,:]\mathbf{h}_i + \mathbf{W}_t[f,:]\mathbf{h}_j > 0$, $\mathbf{W}_s[f,:]\mathbf{h}_j + \mathbf{W}_t[f,:]\mathbf{h}_i < 0$, $(\mathbf{W}_s[f,:] + \mathbf{W}_t[f,:])\,\mathbf{h}_i > 0$, and $(\mathbf{W}_s[f,:] + \mathbf{W}_t[f,:])\,\mathbf{h}_j < 0$.

   According to our assumptions the LHS of Eq. (10) can only be negative if $a[f] < 0$. Yet, the LHS of Eq. (11) can only be negative if $a[f] > 0$. Thus, this case clearly cannot contribute to switching off neighborhood aggregation.

6. **Case** $(+ \; - \; -+)$**:** Let us assume that $\mathbf{W}_s[f,:]\mathbf{h}_i + \mathbf{W}_t[f,:]\mathbf{h}_j > 0$, $\mathbf{W}_s[f,:]\mathbf{h}_j + \mathbf{W}_t[f,:]\mathbf{h}_i < 0$, $(\mathbf{W}_s[f,:] + \mathbf{W}_t[f,:])\,\mathbf{h}_i < 0$, and $(\mathbf{W}_s[f,:] + \mathbf{W}_t[f,:])\,\mathbf{h}_j > 0$.

   Eq. (9) becomes $a[f](1-\alpha)\mathbf{W}_s[f,:]\,(\mathbf{h}_i - \mathbf{h}_j) < 0$. At the same time, the LHS of Eq. (10) simplifies to $a[f]\mathbf{W}_s[f,:](\mathbf{h}_i - \mathbf{h}_j)$ and the LHS of Eq. (11) is $a[f]\alpha\mathbf{W}_s[f,:](\mathbf{h}_j - \mathbf{h}_i) > 0$.

   Hence, a negative Eq. (9) leads to a positive Eq. (11). Accordingly, the last possible sign configuration also does not support switching off neighborhood aggregation, which concludes our derivation.

A.2. Proof of Theorem 4.3

Statement (Restated Theorem 4.3). The gradients and parameters of GATE for layer $l \in [L-1]$ are conserved according to the following laws:

$$\langle W^l[i,:], \nabla_{W^l[i,:]}\mathcal{L}\rangle = \langle W^{l+1}[:,i], \nabla_{W^{l+1}[:,i]}\mathcal{L}\rangle + \langle a_s^{l+1}[i], \nabla_{a_s^{l+1}[i]}\mathcal{L}\rangle + \langle a_t^{l+1}[i], \nabla_{a_t^{l+1}[i]}\mathcal{L}\rangle. \tag{12}$$

and, if additional independent matrices $\mathbf{U}^l$ and $\mathbf{V}^l$ are trainable, it also holds

$$\langle a_s^l[i], \nabla_{a_s^l[i]}\mathcal{L}\rangle + \langle a_t^l[i], \nabla_{a_t^l[i]}\mathcal{L}\rangle = \langle U^l[i,:], \nabla_{U^l[i,:]}\mathcal{L}\rangle + \langle V^l[i,:], \nabla_{V^l[i,:]}\mathcal{L}\rangle. \tag{13}$$

The proof is analogous to the derivation of Theorem 2.2 by (Mustafa & Burkholz, 2023) that is restated in this work as Theorem 4.1. For ease, we replicate their notation and definitions here.

Statement (Rescale invariance: Def 5.1 by Mustafa & Burkholz (2023)). The loss $\mathcal{L}(\theta)$ is rescale-invariant with respect to disjoint subsets of the parameters θ_1 and θ_2 if for every $\lambda > 0$ we have $\mathcal{L}(\theta) = \mathcal{L}((\lambda\theta_1, \lambda^{-1}\theta_2, \theta_d))$, where $\theta = (\theta_1, \theta_2, \theta_d)$.

Statement (Gradient structure due to rescale invariance Lemma 5.2 in (Mustafa & Burkholz, 2023)). The rescale invariance of $\mathcal{L}$ enforces the following geometric constraint on the gradients of the loss with respect to its parameters:

$$\langle \theta_1, \nabla_{\theta_1}\mathcal{L}\rangle - \langle \theta_2, \nabla_{\theta_2}\mathcal{L}\rangle = 0. \tag{14}$$

We first consider the simpler case of GATE$_S$, i.e. $W = U = V$

Theorem A.1 (Structure of GATE$_S$ gradients)**.** *The gradients and parameters of GATE$_S$ for layer $l \in [L-1]$ are conserved according to the following laws:*

$$\langle W^l[i,:], \nabla_{W^l[i,:]}\mathcal{L}\rangle = \langle W^{l+1}[:,i], \nabla_{W^{l+1}[:,i]}\mathcal{L}\rangle + \langle a_s^l[i], \nabla_{a_s^l[i]}\mathcal{L}\rangle + \langle a_t^l[i], \nabla_{a_t^l[i]}\mathcal{L}\rangle. \tag{15}$$

Following a similar strategy to (Mustafa & Burkholz, 2023), we identify rescale invariances for every neuron i at layer l that induce the stated gradient structure.

Given the following definition of disjoint subsets θ_1 and θ_2 of the parameter set θ, associated with neuron i in layer l,

$$\theta_1 = \{x | x \in W^l[i,:]\}$$
$$\theta_2 = \{w | w \in W^{l+1}[:,i]\} \cup \{a_s^l[i]\} \cup \{a_t^l[i]\}$$

We show that the loss of GATE$_S$ remains invariant for any $\lambda > 0$.

The only components of the network that potentially change under rescaling are $h_u^l[i]$, $h_v^{l+1}[j]$, and α_{uv}^l.

The scaled network parameters are denoted with a tilde as $\tilde{a_s}^l[i] = \lambda^{-1} a_s^l[i]$, $\tilde{a_t}^l[i] = \lambda^{-1} a_t^l[i]$, and $\tilde{W}^l[i,j] = \lambda W^l[i,j]$, and the corresponding networks components scaled as a result are denoted by $\tilde{h}_u^l[i]$, $\tilde{h}_v^{l+1}[k]$, and $\tilde{\alpha}_{uv}^l$.

We show that the parameters of upper layers remain unaffected, as $\tilde{h}_v^{l+1}[k]$ coincides with its original non-scaled variant $\tilde{h}_v^{l+1}[k] = h_v^{l+1}[k]$.

Also recall Eq. (4) for $W = U = V$ as:

$$e_{uv}^l = ((1 - q_{uv}) a_s^l + (q_{uv}) a_t^l)^\top \cdot \phi(W^l h_u^{l-1} + W^l h_v^{l-1})$$

where $q_{uv} = 1$ if $u = v$ and $q_{uv} = 0$ if $u \neq v$.

For simplicity, we rewrite this as:

$$e_{uv,u\neq v}^l = (a_s^l)^\top \cdot \phi(W^l h_u^{l-1} + W^l h_v^{l-1}) \tag{16}$$
$$e_{uv,u=v}^l = (a_t^l)^\top \cdot \phi(W^l h_u^{l-1} + W^l h_v^{l-1}) \tag{17}$$

We show that

$$\tilde{\alpha}_{uv}^l = \frac{\exp(\tilde{e}_{uv}^l)}{\sum_{u' \in \mathcal{N}(v)} \exp(\tilde{e}_{uv}^l)} = \alpha_{uv}^l \ , \ \text{ because} \tag{18}$$
$$\tilde{e}_{uv,u\neq v}^l = e_{uv,u\neq v}^l \ , \ \text{ and } \ \tilde{e}_{uv,u=v}^l = e_{uv,u=v}^l \tag{19}$$

which follows from the positive homogeneity of ϕ that allows

$$\begin{aligned}
\tilde{e}_{uv,u=v}^l &= \lambda^{-1} a_s^l[i] \phi(\sum_j^{n_{l-1}} \lambda W^l[i,j](h_u^{l-1}[j] + h_v^{l-1}[j]) \\
&+ \sum_{i' \neq i}^{n_l} a_s^l[i'] \phi(\sum_j^{n_{l-1}} W^l[i',j](h_u^{l-1}[j] + h_v^{l-1}[j]) && (20) \\
&= \lambda^{-1} \lambda a_s^l[i] \phi(\sum_j^{n_{l-1}} W^l[i,j](h_u^{l-1}[j] + h_v^{l-1}[j]) \\
&+ \sum_{i' \neq i}^{n_l} a_s^l[i'] \phi(\sum_j^{n_{l-1}} W^l[i',j](h_u^{l-1}[j] + h_v^{l-1}[j]) && (21) \\
&= e_{uv,u\neq v}^l. && (22)
\end{aligned}$$

and similarly,

$$\begin{aligned}
\tilde{e}^l_{uv,u=v} &= \lambda^{-1} a^l_t[i]\phi(\sum_j^{n_{l-1}} \lambda W^l[i,j](h^{l-1}_u[j] + h^{l-1}_v[j]) \\
&+ \sum_{i' \neq i}^{n_l} a^l_t[i']\phi(\sum_j^{n_{l-1}} W^l[i',j](h^{l-1}_u[j] + h^{l-1}_v[j]) && (23) \\
&= \lambda^{-1}\lambda a^l_t[i]\phi(\sum_j^{n_{l-1}} W^l[i,j](h^{l-1}_u[j] + h^{l-1}_v[j]) \\
&+ \sum_{i' \neq i}^{n_l} a^l_t[i']\phi(\sum_j^{n_{l-1}} W^l[i',j](h^{l-1}_u[j] + h^{l-1}_v[j]) && (24) \\
&= e^l_{uv,u=v}. && (25)
\end{aligned}$$

Since $\tilde{\alpha}^l_{uv} = \alpha^l_{uv}$, it follows that

$$\begin{aligned}
\tilde{h_u}^l[i] &= \phi_1\left(\sum_{z\in\mathcal{N}(u)} \alpha^l_{zu} \sum_j^{n_{l-1}} \lambda W^l[i,j] h^{l-1}_z[j]\right) \\
&= \lambda\phi_1\left(\sum_{z\in\mathcal{N}(u)} \alpha^l_{zu} \sum_j^{n_{l-1}} W^l[i,j] h^{l-1}_z[j]\right) \\
&= \lambda h^l_u[i].
\end{aligned}$$

In the next layer, we therefore have

$$\begin{aligned}
\tilde{h}^{l+1}_v[k] &= \phi_1\left(\sum_{u\in\mathcal{N}(v)} \alpha^{l+1}_{uv} \sum_i^{n_l} \lambda^{-1} W^{l+1}[k,i] \tilde{h}^l_u[i]\right) \\
&= \phi_1\left(\sum_{u\in\mathcal{N}(v)} \alpha^{l+1}_{uv} \sum_i^{n_l} \lambda^{-1} W^{l+1}[k,i] \lambda h^l_u[i]\right) \\
&= \phi_1\left(\sum_{u\in\mathcal{N}(v)} \alpha^{l+1}_{uv} \sum_i^{n_l} W^{l+1}[k,i] h^l_u[i]\right) \\
&= h^{l+1}_v[k].
\end{aligned}$$

Thus, the output node representations of the network remain unchanged, and the loss $\mathcal{L}$ is rescale-invariant.

Next consider the case that W^l, U^l, and V^l are independent matrices. Similarly to the previous reasoning, we see that if we scale $\tilde{W}^l[i,:] = W^l[i,:]\lambda$, then also scaling $\tilde{W}^{l+1}[:,i] = W^{l+1}[:,i]\lambda^{-1}$ and $\tilde{a}^{l+1}_s[i] = a^{l+1}_s[i]\lambda^{-1}$ and $\tilde{a}^{l+1}_t[i] = a^{l+1}_t[i]\lambda^{-1}$ will keep the GATE layer unaltered.

In this case, we obtain an additional rescaling relationship between a^l_s, a^l_t and U^l, V^l. A rescaling of the form $\tilde{a_s}^l[i] = \lambda^{-1}a^l_s[i]$, $\tilde{a_t}^l[i] = \lambda^{-1}a^l_t[i]$ could be compensated by $\tilde{U}^l[i,:] = U^l[i,:]\lambda$ and $\tilde{V}^l[i,:] = V^l[i,:]\lambda$. It follows immediately that $\tilde{e}_{uv} = e_{uv}$.

A.3. Derivation of Insight 4.4

Following the analysis in A.1, in contrast to GAT, $\alpha_{ij}/\alpha_{ii} << 1$ can be easily realized in GATE with $a_s[f] < 0$ and $a_t[f] > 0$ for all or only a subset of the features. Note that for the non-weight-sharing case, $\mathbf{U}$ and $\mathbf{V}$ in GATE would simply correspond to $\mathbf{W}_s$ and $\mathbf{W}_t$, respectively, in GATE and the same line of reasoning holds. Large norms are usually not required to create a notable difference in size between e_{ii} and e_{ij}.

B. Experimental Settings

Our complete experimental setup is described as follows.

Non-linearity For GAT_S and GAT networks, we substitute ϕ in Eq. (3) with LeakyReLU as defined in the standard architecture. For GATE, we substitute ϕ in Eq. (4) with ReLU in order to be able to interpret the sign of $\mathbf{a_s}$ and $\mathbf{a_t}$ parameters as contributing positively or negatively to neighborhood aggregation. MLP, MLP_{+GAT}, and FAGCN also all use ReLU after every hidden layer.

Network Width We vary the depth of GAT and GATE networks in all our experiments as specified. For synthetic datasets, the network width is fixed to 64 in all cases. For OGB datasets, we use the hidden dimensions used by (Brody et al., 2022). For the remaining datasets, the network width is also fixed to 64.

Initialization The feature transformation parameter matrices, i.e., $\mathbf{W}$, $\mathbf{U}$, and $\mathbf{V}$ are initialized randomly with an orthogonal looks-linear structure (Burkholz & Dubatovka, 2019) for MLP, MLP_{+GAT}, $\text{GAT}_{(S)}$ and $\text{GATE}_{(S)}$. The parameters $\mathbf{a}$ in $\text{GAT}_{(S)}$ use Xavier initialization (Glorot & Bengio, 2010), as is the standard. In $\text{GATE}_{(S)}$, $\mathbf{a}_s$ and $\mathbf{a}_t$ are initialized to 0 to initially give equal weights to the features of a node itself and its neighboring nodes.

Optimization Synthetic, OGB, and remaining real-world tasks are run for a maximum of 10000, 2000, 5000 epochs, respectively, using the Adam optimizer. To isolate the effect of the architecture and study the parameter dynamics during training as best as possible, we do not use any additional elements such as weight decay and dropout regularization. We also do not perform any hyperparameter optimization. However, the learning rate is adjusted for different real-world datasets to enable stable training of models as specified in Table 6. Nevertheless, for a fair comparison, the same learning rate is used for a given problem across all architectures. For all synthetic data, a learning rate of 0.005 is used. Real-world datasets use their standard train/test/validation splits, i.e. those provided by Pytorch Geometric for Planetoid datasets Cora and Citeseer, by OGB framework for OGB datasets, and by (Platonov et al., 2023) for all remaining real-world datasets.

Code Our experimental code and synthetic data generators are available at `https://github.com/RelationalML/GATE.git`.

Table 6: Details of real-world datasets used in experiments.

dataset	# nodes	# edges	# features	# classes	learning rate used for L layer networks
ogb-arxiv	$169,343$	$2,315,598$	128	40	$L=[12]:0.001$
ogb-products	$2,449,029$	$123,718,152$	100	47	$L=[8]:0.001$
ogb-mag	$736,389$	$10,792,672$	128	349	$L=[5]:0.005$
roman-empire	$22,662$	$32,927$	300	18	$L=[5]:0.001, L=[10]:0.0005$
amazon-ratings	$24,492$	$93,050$	300	5	$L=[5]:0.001, L=[10]:0.0005$
questions	$48,921$	$153,540$	301	2	$L=[5]:0.001, L=[10]:0.0005$
minesweeper	$10,000$	$39,402$	7	2	$L=[5]:0.001, L=[10]:0.0005$
tolokers	$11,758$	$519,000$	10	2	$L=[5]:0.001, L=[10]:0.0005$
cora	$2,708$	$10,556$	$1,433$	7	$L=[2,5]:0.005, L=[10,20]:0.0005$
citeseer	$3,327$	$9,104$	$3,703$	6	$L=[2,5]:0.001, L=[10,20]:0.0001$
actor	$7,600$	$26,659$	932	5	$L=[2,5,10,20]:0.005$
texas	183	279	$1,703$	5	$L=[2,5]:0.01, L=[10,20]:0.0005$
wisconsin	251	450	$1,703$	5	$L=[2,5]:0.01, L=[10,20]:0.005$

C. Additional Results

Smaller Real-World Datasets We evaluate GAT and GATE on five small-scale real-world datasets with varying homophily levels β as defined in (Pei et al., 2020) and report results in Table 7. Higher values of β indicate higher homophily, i.e. similar nodes (with the same label) tend to be connected. We note that a 2-layer network of a baseline method for heterophilic datasets, Geom-GCN (Pei et al., 2020), attains test accuracy (%) of 64.1, 67.6, and 31.6 for Wisconsin, Texas, and Actor datasets, respectively, which is in line with that achieved by GATE. Except for Citeseer, the best overall performance for each dataset is achieved on a shallow model. This is not surprising as these datasets are small-scale and potentially prone to over-fitting in large models, particularly since we do not use any skip connections or regularization to retain model performance. Furthermore, the three heterophilic datasets have been recently shown to be problematic (Platonov et al., 2023). Therefore, a better evaluation of GATE is on relatively large-scale OGB datasets (Hu et al., 2021) and more recent heterophilic datasets (Platonov et al., 2023) that can exploit the flexibility of GATE. Although GATE is more parameterized than GAT, it usually requires fewer training epochs and generalizes better, in addition to other advantages over GAT as discussed in the paper.

Table 7: Test accuracy (%) of GAT and GATE models for network depth L on small-scale real-world datasets with varying homophily levels β. Entries marked with * indicate models that achieve 100% training accuracy and stable test accuracy. Otherwise, test accuracy at max. validation accuracy is reported.

Data	β	$L = 2$			$L = 5$			$L = 10$			$L = 20$		
		GAT	GATE$_S$	GATE	GAT	GATE$_S$	GATE	GAT	GATE$_S$	GATE	GAT	GATE$_S$	GATE
Wisc.	0.21	62.7*	**80.4**	70.5*	51.0	**70.5**	60.7*	45.1	**62.7**	58.8	47.1	**62.7**	60.7
Texas	0.11	56.7*	**67.6***	**67.6***	51.4	**67.6***	**67.6***	56.7*	62.2*	**62.3***	59.4*	62.1*	**64.9**
Actor	0.24	27.1	**32.2**	31.6	25.4	27.5	**29.2**	25.3	27.4	**27.9**	24.5	24.6	**29.4**
Cora	0.83	80.0	**81.0***	80.8	79.8	**80.8***	80.4	77.6	**80.0***	79.2	77.7	77.2*	**79.0**
Cite.	0.71	68.0	67.6*	**68.3**	67.2	**68.7***	67.8	66.9	**67.6***	**67.6**	68.2	67.1*	**69.2**

Initialization of attention parameters in GAT We show in Fig. 7 that setting the initial value of attention parameters a_s and a_t in GATE to zero is, in fact, not what enables neighborhood aggregation but rather the separation of a into a_s and a_t as discussed in Insight 4.4.

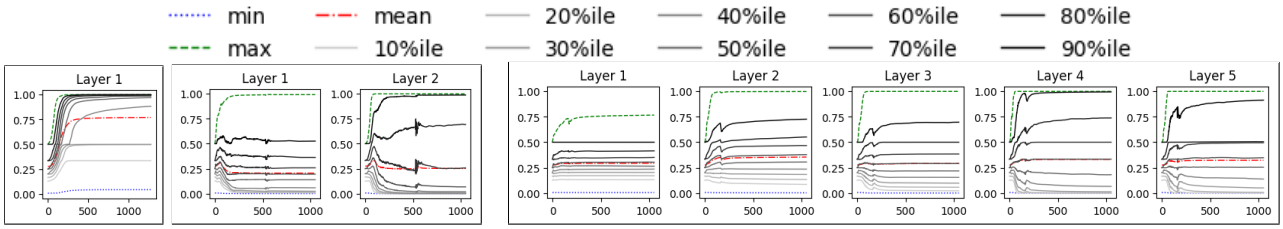
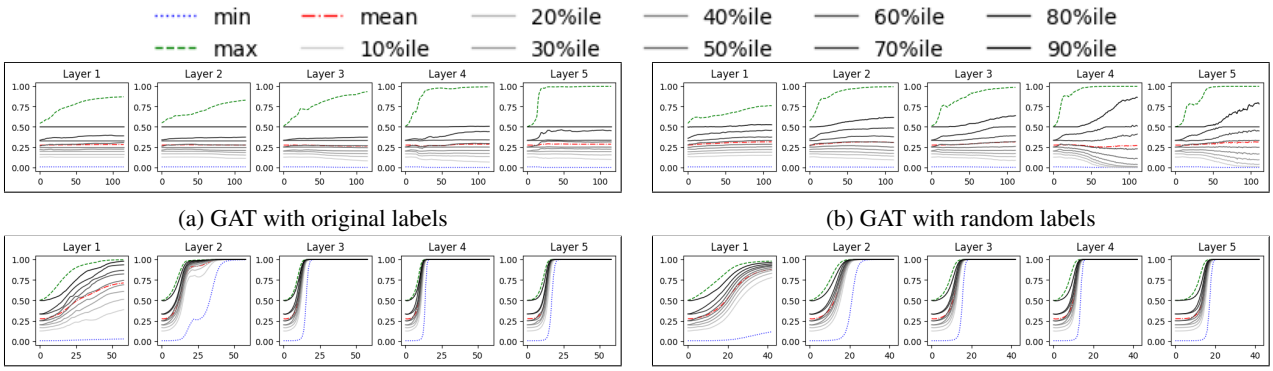


Figure 7: Distribution of α_{vv} against training epoch for self-sufficient learning problem using the Cora structure with random labels, where input node features are a one-hot encoding of node labels for GAT with attention parameters $\mathbf{a}$ initialized to zero. Left to right: 1, 2 and 5 layer models that achieve test accuracy of 100%, 52.7%, and 36.2%, respectively, which is similar to the results obtained by standard Xavier initialization of attention parameters in GAT.

Further analysis of experiments We present the analysis of α coefficients learned for some experiments in the main paper that were deferred to the appendix due to space limitations.



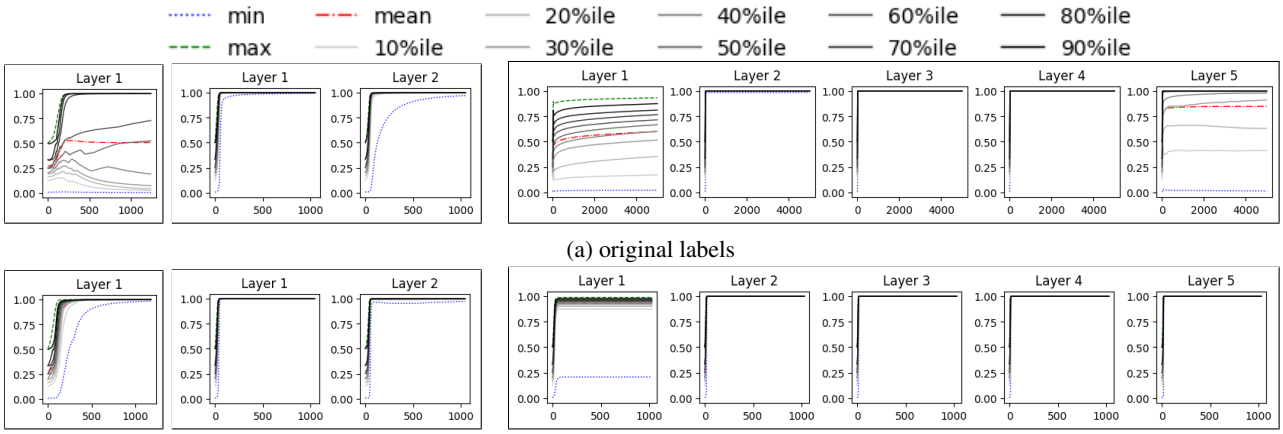
(a) GAT with original labels

(b) GAT with random labels

(c) GATE with original labels

(d) GATE with random labels

Figure 8: Distribution of α_{vv} against training epoch for self-sufficient learning problem using Cora structure and input node features as the one-hot encoding of labels for 5 layer models.



(a) original labels

(b) Random labels

Figure 9: Distribution of α_{vv} against training epoch for the self-sufficient learning problem using Cora graph structure with original (top) and random (bottom) node labels and input node features as a one-hot encoding of labels. Left to right: 1, 2, and 5 layer GATE$_S$ models that all 100% test accuracy except in the case of 5 layer model using original labels. In this case, although a training accuracy if 100% is achieved at 32 epochs with test accuracy 97.3%, a maximum test accuracy of 98.4% is reached at 7257 epochs. Training the model to run to 15000 epochs only increases it to 98.4%. An increased learning rate did not improve this case. However, we also run the GAT model for 15000 epochs for this case, and it achieves 85.9% test accuracy at epoch 47 where the model achieves 100% accuracy and only achieves a maximum test accuracy of 89.3% briefly at epoch 8.

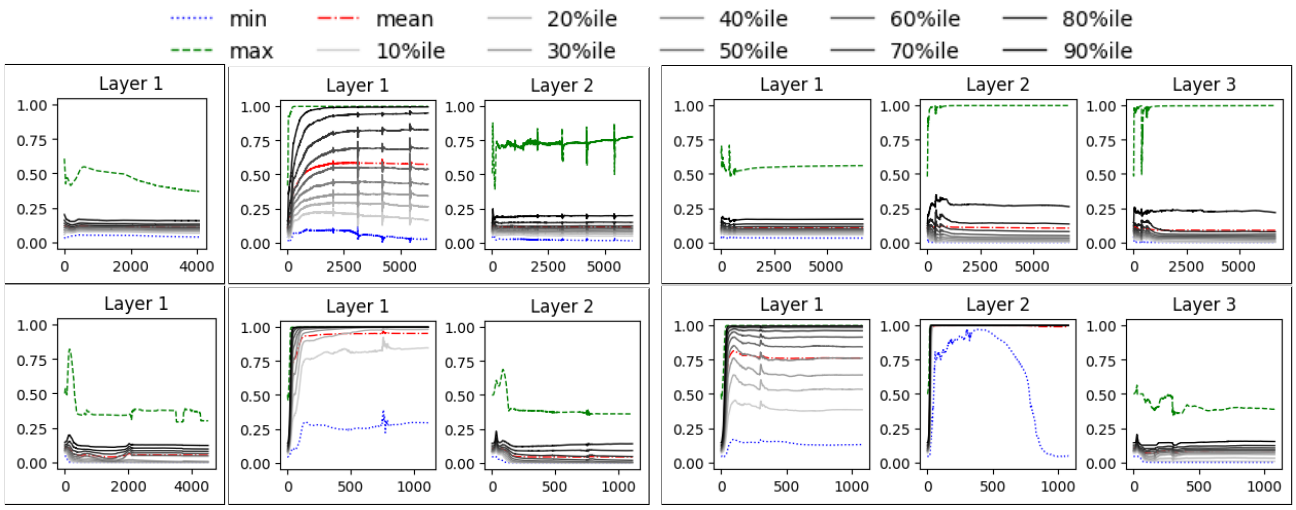
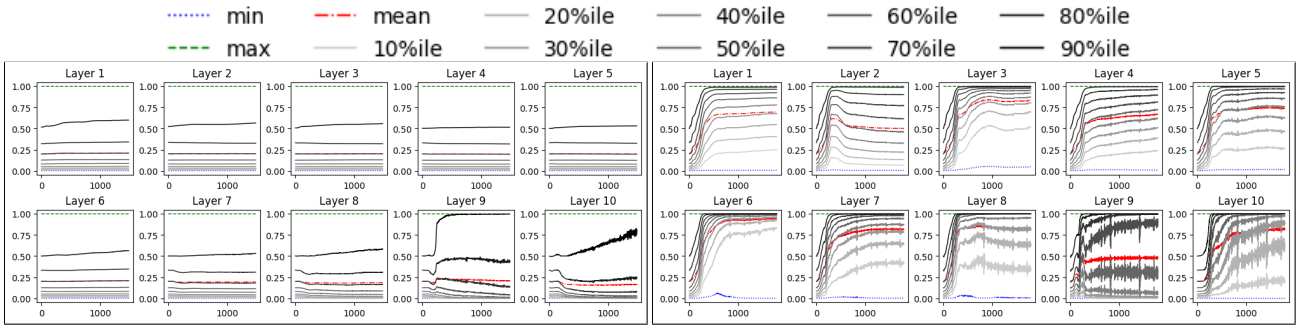


Figure 10: Distribution of α_{vv} against training epoch for the neighbor-dependent learning problem with $k = 1$. Rows: GAT (top) and GATE (bottom) architecture. Columns (left to right): 1, 2, and 3 layer models. While GAT is unable to switch off neighborhood aggregation in any layer, only 1 layer of the 2 and 3 layer models perform neighborhood aggregation.

Table 8: Neighbor-dependent learning: k and L denote the number of aggregation steps of the random GAT used for label generation and the number of layers of the evaluated network, respectively. Entries marked with * identify models where 100% train accuracy is not achieved. Underlined entries identify the model with the highest train accuracy at the epoch of max. test accuracy. This provides an insight into how similar the function represented by the trained model is to the function used to generate node labels, i.e. whether the model is simply overfitting to the train data or really learning the task. Higher training and test accuracy simultaneously indicate better learning. In this regard, the difference in train accuracy at max. test accuracy between GATE and GAT$_S$ or GAT is only 0.4, 1.0 and 0.6 for the settings $(k = 1, L = 3)$, $(k = 2, L = 4)$ and $(k = 3, L = 3)$, respectively.

k	L	Test Acc. @ Epoch of Max. Train Acc.			Max Test Acc. @ Epoch		
		GAT$_S$	GAT	GATE	GAT$_S$	GAT	GATE
1	1	92.0@2082*	91.2@6830*	**93.2@3712***	93.2@1421	92.0@9564	**<u>93.6@3511</u>**
	2	89.6@8524*	88.0@8935	**91.2@942**	91.6@5188	92.8@4198	**<u>95.6@111</u>**
	3	86.4@9180*	88.8@997	**92.8@618**	91.2@6994	<u>92.8@437</u>	**97.2@82**
2	2	88.8@6736*	**89.6@3907**	88.8@467	93.2@151	**93.2@95**	<u>92.0@105</u>
	3	82.0@7612	89.2@1950	**91.6@370**	91.6@1108	93.2@856	**<u>95.2@189</u>**
	4	84.8@4898	82.4@739	**87.2@639**	<u>88.0@1744</u>	88.4@423	**90.4@447**
3	3	80.8@8670	80.4@737	**85.2@391**	<u>86.4@1578</u>	88.8@285	**92.0@47**
	4	78.0@3012	80.4@767	**89.6@480**	86.8@1762	85.6@469	**<u>91.6@139</u>**
	5	80.0@6611	74.4@1701	**86.0@447**	85.6@921	83.6@1098	**<u>91.2@243</u>**



(a) Tolokers, GAT: 61.6% test AUROC.

(b) Tolokers, GATE: 69.2% test AUROC.

Figure 11: Distribution of α_{vv} against training epoch for one run of 10 layer networks on real-world heterophilic task.

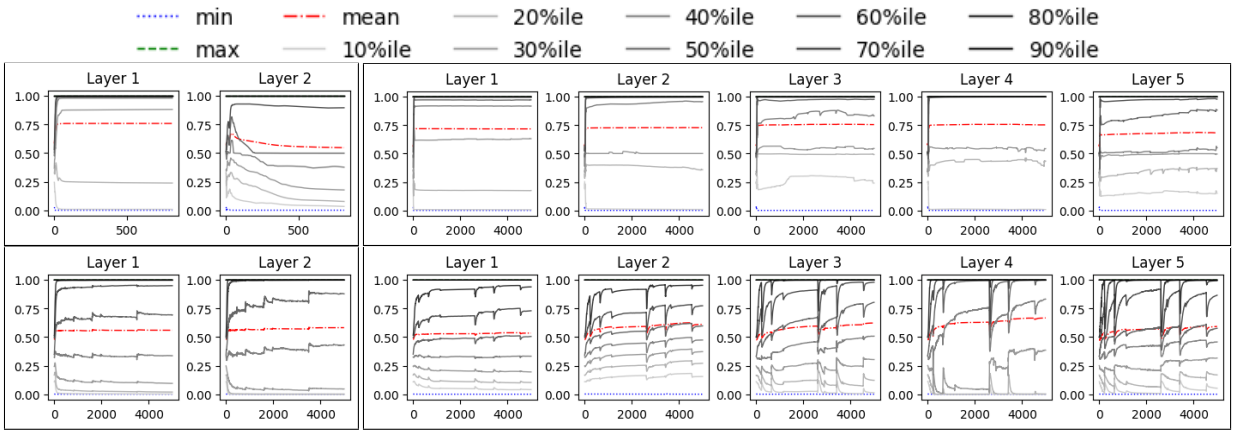


Figure 12: Distribution of α_{vv} against training epoch of 2-layer (left) and 5-layer (right) GAT networks for heterophilic datasets Texas (top) and Actor (bottom) 2-layer modes. Despite having connections to unrelated neighbors, GAT is unable to switch off neighborhood aggregation.

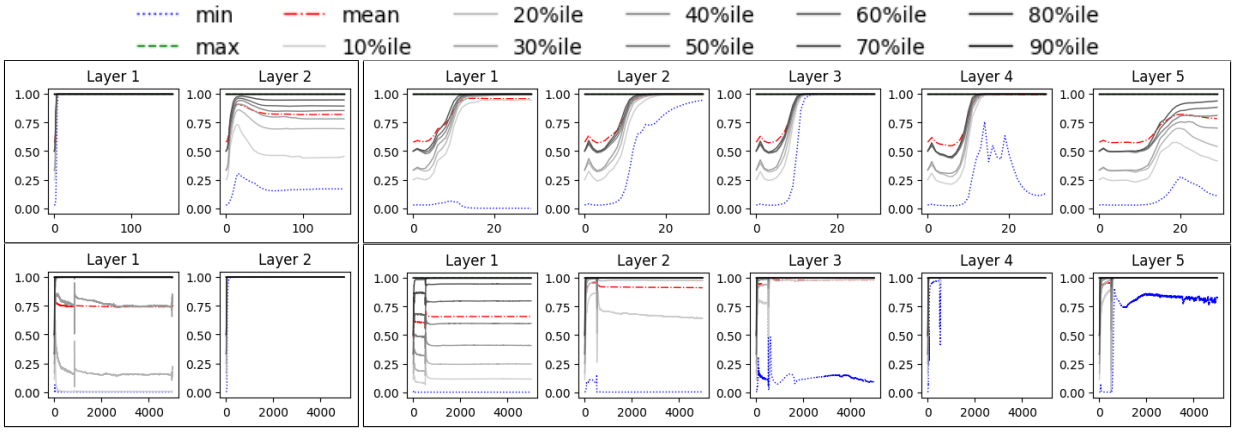


Figure 13: Distribution of α_{vv}, against training epoch of 2-layer (left) and 5-layer (right) GATE networks for heterophilic datasets Texas (top) and Actor (bottom), across layers could be interpreted to indicate the inherent importance of raw node features relative to their neighborhoods. For instance, in the case of Texas, GATE carries out little to no neighborhood aggregation in the first layer over input node features. Instead, aggregation is mainly done over node features transformed in earlier layers that effectuate non-linear feature learning as in perceptrons. However, in the case of Actor, GATE prefers most of the neighborhood aggregation to occur over the input node features, indicating that they are more informative for the task at hand.

Over-smoothing analysis In the main paper, we have already established the superior performance of GATE, compared to GAT, on several tasks. Intuitively, this can partially be attributed to reduced over-smoothing as its root cause, unnecessary neighborhood aggregation, is alleviated. Here, we verify this insight quantitatively.

A widely accepted measure of over-smoothing is the Dirichlet energy (DE) (Cai & Wang, 2020). However, Eliasof et al. (2023) propose a modification of DE to measure GAT energy E_{GAT}, that we use to evaluate over-smoothing in our experiments (see Table 9). We note that the notion of 'over-smoothing' is itself task-dependent. It is difficult to determine the optimal degree of smoothing for a task and the threshold that determines 'over'-smoothing. This merits an in-depth analysis and curation of task-dependent smoothness measures that are not our focus. To show that GATE reduced over-smoothing relative to GAT, it suffices that a decrease in smoothing and an increase in accuracy occur simultaneously.

Table 9: The measures E_{input}, E_{GAT} and E_{GATE} denote the smoothness of input node features, node features at the last layer L of the trained GAT and GATE models, respectively. Two cases are considered: All node pairs and only adjacent node pairs to measure smoothing at the global graph and local node level. Higher values indicate less smoothing. Node representations learned by GATE achieve higher test accuracy on all these tasks, as reported in the main paper, and are simultaneously less smooth than GAT in most cases, indicating that GATE potentially alleviates over-smoothing in GATs.

(a) Synthetic self-sufficient task: varying graph structure and label distribution, node features as one-hot encoding of labels, $L = 5$.

Experiment setting	All node pairs			Adjacent node pairs		
	E_{input}	E_{GAT}	E_{GATE}	E_{input}	E_{GAT}	E_{GATE}
Cora, Original Labels	6.016 e + 06	1.281 e + 08	**2.037** e + **09**	2.006 e + 03	7.903 e + 03	**1.971** e + **05**
Cora, Random Labels	6.283 e + 06	3.472 e + 09	**3.747** e + **09**	9.080 e + 03	9.306 e + 05	**3.966** e + **06**
ER($p = .01$), 2 Classes	4.994 e + 05	2.701 e + 06	**4.478** e + **07**	5.042 e + 03	2.272 e + 04	**3.229** e + **05**
ER($p = .01$), 8 Classes	8.745 e + 05	3.350 e + 07	**2.615** e + **08**	8.694 e + 03	1.762 e + 05	**1.960** e + **06**

(b) Synthetic neighbor-dependent task: graph structure, node features, generator parameter k, and label distribution as in Section 5.

Experiment setting	All node pairs			Adjacent node pairs		
	E_{input}	E_{GAT}	E_{GATE}	E_{input}	E_{GAT}	E_{GATE}
$k = 1, L = 3$	1.953 e + 06	**5.306** e + **07**	5.095 e + 07	1.957 e + 05	**4.234** e + **05**	3.610 e + 05
$k = 2, L = 4$	1.975 e + 06	1.193 e + 07	**2.198** e + **07**	2.012 e + 04	1.016 e + 05	**1.939** e + **05**
$k = 3, L = 5$	1.951 e + 06	1.645 e + 07	**1.053** e + **08**	1.966 e + 04	1.408 e + 05	**9.096** e + **05**

(c) Real-world tasks.

Dataset	L	All node pairs			Adjacent node pairs		
		E_{input}	E_{GAT}	E_{GATE}	E_{input}	E_{GAT}	E_{GATE}
roman-empire	5	1.274 e + 09	1.002 e + 11	**7.491** e + **11**	7.878 e + 04	2.441 e + 06	**4.009** e + **07**
amazon-ratings	10	3.844 e + 08	1.187 e + 10	**2.272** e + **10**	4.933 e + 04	3.848 e + 05	**7.430** e + **05**
minesweeper	5	6.869 e + 07	1.386 e + 09	**2.531** e + **10**	2.628 e + 04	1.946 e + 05	**7.017** e + **06**
tolokers	10	1.391 e + 08	1.044 e + 11	**1.042** e + **11**	3.423 e + 05	1.249 e + 08	**1.397** e + **08**
cora	10	5.088 e + 05	1.437 e + 07	**2.783** e + **08**	6.490 e + 02	5.959 e + 02	**1.226** e + **04**
citeseer	10	3.463 e + 05	2.916 e + 05	**1.126** e + **07**	2.360 e + 02	2.426 e + 00	**1.030** e + **02**
texas	10	1.945 e + 06	3.280 e + 04	**3.877** e + **04**	1.758 e + 04	8.487 e + 01	**9.695** e + **01**
actor	10	2.612 e + 08	**1.800** e + **07**	1.364 e + 07	1.237 e + 05	**2.810** e + **03**	2.215 e + 03
wisconsin	10	4.438 e + 06	1.057 e + 07	**1.008** e + **08**	3.299 e + 04	3.765 e + 04	**7.363** e + **08**

Comparison with other GNNs Other GNN architectures could potentially switch off neighborhood aggregation, as we show here. However, they are less flexible in assigning different importance to neighbors, suffer from over-smoothing, or come at the cost of an increased parameter count by increasing the size of the hidden dimensions (e.g. via a concatenation operation). We evaluate the performance of three such architectures that, in principle, employ different aggregation methods, which are likely to be capable of switching off neighborhood aggregation, on synthetic datasets empirically and discuss their ability or inability to switch off neighborhood aggregation qualitatively as follows.

1. ωGAT (Eliasof et al., 2023) introduces an additional feature-wise layer parameter ω that can, in principle, switch off neighborhood aggregation by setting ω parameters to 0, in addition to the attention mechanism based on GAT. However, in practice, as we verify on our synthetic dataset in Figure 14, it is unable to effectively switch off neighborhood aggregation. Although it outperforms GAT, it is still substantially worse than GATE, especially for the deeper model due to unnecessary neighborhood aggregations. Another architecture based on graph attention, superGAT(Kim & Oh, 2021), falls under the paradigm of structural learning as it uses a self-supervised attention mechanism essentially for link prediction between nodes, and therefore its comparison with GATE is infeasible.

2. GraphSAGE (Hamilton et al., 2018) uses the concatenation operation to combine the node's own representation with the aggregated neighborhood representation. Therefore, it is usually (but not always) able to switch off the neighborhood aggregation for the synthetic datasets designed for the self-sufficient learning task (see Table 10). Mostly, GATE performs better on the neighbor-dependent task, in particular for deeper models, where the performance of GraphSAGE drops likely due to over-smoothing (see Table 11).

3. FAGCN (Bo et al., 2021) requires a slightly more detailed analysis. Authors of FAGCN state in the paper that: 'When $\alpha_{ij}^G \approx 0$, the contributions of neighbors will be limited, so the raw features will dominate the node representations.' where α_{ij}^G defined in the paper can be considered analogous to α_{ij} in GAT, though they are defined differently. Thus, from an expressivity point of view, FAGCN should be able to assign parameters such that all $\alpha_{ij}^G = 0$. However, we empirically observe on synthetic datasets designed for the self-sufficient learning task, values of α_{ij}^G do not, in fact, approach zero. Despite being unable to switch off neighborhood aggregation, FAGCN, in its default implementation, achieves 100% test accuracy on the task. We discover this is so because FAGCN introduces direct skip connections of non-linearly transformed raw node features to every hidden layer. Given the simplicity of the one-hot encoded features in the datasets and the complete dependence of the label on these features, FAGCN is able to represent the desired function. In order to better judge its ability to *switch off neighborhood aggregation by setting* $\alpha_{ij}^G = 0$, we remove this skip connection. From an expressivity point of view, FAGCN should still be able to achieve 100% test accuracy by using only the (non-)linear transformations of raw features initially and performing no neighborhood aggregation in the hidden layers. However, we find that FAGCN was unable to emulate this behavior in practice. For a fair comparison of the differently designed attention mechanism in FAGCN with GATE, we introduce self-loops in the data so FAGCN may also receive a node's own features in every hidden layer. Even then, FAGCN fails to achieve perfect test accuracy as shown in Table 10. Therefore, we suspect the attention mechanism in FAGCN may also be susceptible to the trainability issues we have identified for the attention mechanism in GAT. Nevertheless, the capacity of FAGCN to learn negative associations with neighboring nodes is complementary to GATE and both could be combined. It would be interesting to derive conservation laws inherent to other architectures such as FAGCN and GraphSAGE and study how they govern the behaviour of parameters. Furthermore, by design, FAGCN does not perform any non-linear transformations of aggregated neighborhood features which may be necessary in some tasks, such as our synthetic dataset for the neighbor-dependent learning task. As Table 11 shows, GATE outperforms FAGCN on such a task.

Lastly, we would like to emphasize that our aim is to provide insights into the attention mechanism of GAT and understand its limitations. While it should be able to flexibly assign importance to neighbors and the node itself without the need for concatenated representation or explicit skip connections of the raw features to every layer, it is currently unable to do so in practice. In order to verify our identification of trainability issues, we modify the GAT architecture to enable the trainability of attention parameters which control the trade-off between node features and structural information.

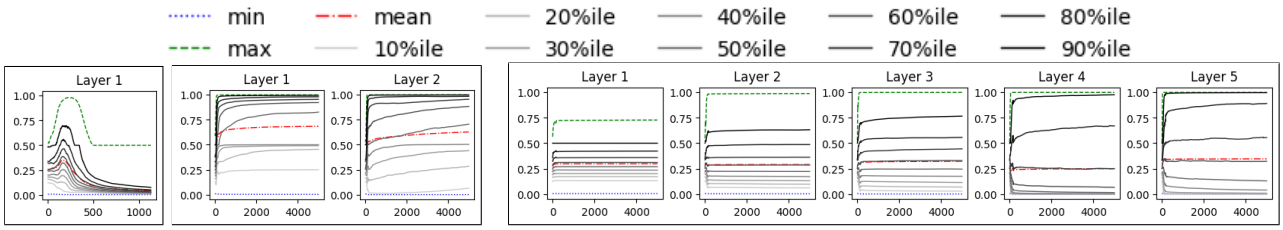


Figure 14: Distribution of α_{vv} against training epoch for self-sufficient learning problem using the Cora structure with random labels, where input node features are a one-hot encoding of node labels, for the ωGAT architecture for the 1, 2 and 5 layer models that achieve test accuracy of 100%, 98.5%, and 49.3%, respectively.

Table 10: Self-sufficient learning: S, C and L denote graph structure, number of label classes, and number of network layers, respectively. Original (Orig.) and Randomized (Rand.) labels are used for the Cora structure. The FAGCN model is implemented without skip connections from the input layer to every other layer and without any self-loops in input data, whereas FAGCN* denotes the model also without skip connections but with self-loops introduced for all nodes in input data.

Structure	C	L	Max. Test Accuracy (%)				
			GAT	GATE	FAGCN	FAGCN*	SAGE
Cora	O,7	1	**100**	**100**	90.1	97.6	**100**
		2	94.6	**100**	94.2	94.9	98.8
		5	88.5	**99.7**	87.1	89.1	92.4
	R,7	1	**100**	**100**	61.6	97.8	**100**
		2	57.0	**100**	69.2	70.5	**100**
		5	36.7	**100**	21.2	36.7	99.6
ER ($p = 0.01$)	R,2	1	**100**	**100**	**100**	**100**	**100**
		2	**100**	**100**	**100**	**100**	**100**
		5	99.6	**100**	96.4	99.2	**100**
	R,8	1	99.2	**100**	86.4	98.8	**100**
		2	97.6	**100**	86.0	91.6	**100**
		5	38.4	**100**	31.6	40.4	**100**

Table 11: Neighbor-dependent learning: k and L denote the number of hops aggregated in the neighborhood to generate labels, and the number of layers of the evaluated network, respectively.

k	L	Max Test Accuracy (%) @ Epoch			
		GAT	GATE	SAGE	FAGCN
1	1	92@9564	**93.6 @ 3511**	93.2@2370	93.2@1618
	2	92.8@4198	**95.6 @ 111**	95.6@723	94.1@1455
	3	92.8@437	**97.2 @ 82**	96.8@100	81.2@573
2	2	**93.2 @ 95**	92.0@105	90.8@199	90.4@170
	3	93.2@856	**95.2 @ 189**	94.4@113	88.8@283
	4	88.4@423	90.4@447	**92.4 @ 139**	87.6@549
3	3	88.8@285	**92.0 @ 47**	87.6@45	89.2@528
	4	85.6@469	**91.6 @ 139**	88@60	89.2@3191
	5	83.6@1098	**91.2 @ 243**	86.0@35	88.8@205