THE IMPACT OF EGO-NEIGHBOR SEPARATION AND GRAPH TOPOLOGY ON GRAPH NEURAL NETWORK MEMORIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent studies on memorization in Graph Neural Networks (GNNs) indicate that nodes with high label inconsistency in their feature-space neighborhood are more susceptible to being memorized. However, this perspective overlooks the interplay between diverse topological factors and the inductive biases of GNN that dictate how ego-node and neighborhood representations are integrated during the aggregation phase. In this work, we systematically dissect GNN memorization by categorizing GNNs into two paradigms: *Ego-Neighbor Integration* GNNs, which mix ego-node and neighborhood representations during aggregation (e.g., GCN), and *Ego-Neighbor Separation* GNNs, which utilize independent ego-weight layers to process ego-node representations without mixing them during aggregation (e.g., GraphSAGE). Our findings indicate that the interplay between topological factors such as node degree can suppress memorization in both paradigms; however, Ego-Neighbor Separation GNNs still retain their memorization ability more effectively than Integration GNNs. Finally, bridging these architectural insights with privacy concerns, we show that GNNs equipped with ego-weight layers exhibit substantially increased susceptibility to Membership Inference Attacks. Ultimately, our findings demonstrate that GNN memorization is not merely a topology-dependent phenomenon. Rather, it is the result of a complex interplay between the topological environment and how ego-node and neighborhood representations are integrated.

1 INTRODUCTION

Memorization in neural networks is *not* distributed uniformly across training samples. Prior work has shown that it tends to concentrate on a small subset of long-tailed examples (Feldman & Zhang, 2020), suggesting that memorization is fundamentally a sample-dependent phenomenon rather than a uniform property of the dataset as a whole. One natural explanation is that these samples are too distributionally underrepresented to support stable pattern learning, making sample-specific fitting more likely. In other words, samples that are weakly supported by dominant patterns in the training data are harder for models to learn and therefore more susceptible to memorization.

In graph neural networks (GNNs), this phenomenon manifests itself in a more structured, relational form, where a node's rarity is not defined solely by frequency but by *weak distributional support from its local features, labels, and structural context.* This is doubly concerning: memorization not only undermines generalization, but also exposes GNNs to privacy risks (Haim et al., 2022; Buzaglo et al., 2023).

Although recent studies Jamadandi et al. (2025) provide insight by linking memorization to outcome-level properties such as label homophily positing that label inconsistency in their feature-space neighborhood is the primary driver, we propose expanding this perspective. We introduce a broader view based on *Node Support*, suggesting that structural factors such as feature homophily (Zheng et al., 2024) and degree bias (Subramonian et al., 2024) act alongside label homophily to affect a model's memorization capability. *Yet node support alone does not determine memorization.* Another key factor is the learning process that combines information from the ego-node and its neighborhood. While graph properties determine the support available to a node, this

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

process determines how that support is translated into learning. Early GNNs like GCN mix a target node's features with those of its neighbors, performing well when strong node support exists, such as in high label homophily graphs (Yang et al., 2021; Li et al., 2022). However, this strict mixing degrades performance when neighbors do not support useful information. To handle these lack of node support environments (e.g., low label homophily), architectures often employ *Ego-Neighbor Separation* using dedicated ego-weight layers that process ego-node features independently (Luan et al., 2023; Ma et al., 2022; Zhu et al., 2020). Although this design successfully maintains performance in the absence of node support, it fundamentally alters how the model utilizes structural support. This architectural shift inspires a critical, yet largely unexplored, question: when a model provides an independent learning pathway that bypasses neighborhood mixing, how does this affect its propensity to memorize noisy or structurally isolated data ?

To systematically investigate this, we categorize architectures into *Ego-Integration* and *Ego-Separation* GNNs. This concept allows us to empirically study how the ego-weight layer affects memorization across diverse topological environments defined by label homophily, feature homophily, and node degree. Furthermore, we extend our empirical analysis to evaluate the privacy risks associated with these designs, utilizing Membership Inference Attacks (MIAs) to observe how the presence of an ego-weight layer might exacerbate privacy vulnerabilities under different topological conditions.

Our Contributions

Identification of novel topological interplay factors affecting memorization: We demonstrate that other topology acts alongside label homophily significantly influence GNN memorization. Notably, we find that high-density graphs (characterized by high average node degrees) with high label homophily are inherently more robust to memorization for GNNs. However, Ego-Separation models were able to memorize noisy label nodes when training to the overfitting phase. Furthermore, we also found that in high-density graphs with both label and feature homophily being high, it can mitigate the memorization capacity of GNN on both model types.

Introducing the Memorization concept between Ego-Neighbor Integration and Separation GNNs: We establish that the Ego-Neighbor Separation GNN, which incorporates ego-weight layers, exhibits a higher propensity to memorize noisy labels compared to the Ego-Neighbor Integration GNN that mixes an ego-node feature with those of its neighbors, especially in heterophily graph.

Ego-Neighbor Separation GNNs are more vulnerable to privacy attacks than Integration ones: Beyond memorization, we conducted an empirical study of Membership Inference Attacks (MIAs) on GNNs. The results reveal that using ego-weight layers significantly increases a model's vulnerability to MIAs in heterophily graph.

2 Related work

Node Distinguishability: Samples that are ambiguous or closely resemble instances from other classes (e.g., atypical or mislabeled examples) are inherently challenging for neural networks to learn (Baldock et al., 2021). Since these samples share little commonality with others in their ground-truth class, they receive weak distributional support during training.

In graph-structured data, this concept directly translates into node distinguishability. For example, in low- to mid-label homophily graphs, inter-class nodes (nodes whose neighborhoods are dominated by different classes) are harder to learn than intra-class nodes (nodes whose neighborhoods are dominated by the same class), which are typically found more often in high label homophily graphs since they are supported by consistent neighbors within the same class.

In addition to label homophily, Zheng et al. (2024) investigate two additional types of homophily: structural and feature homophily. While higher structural homophily consistently improves node distinguishability for GNNs since it quantifies neighborhood structure consistency among nodes of the same class, feature homophily measures how node features depend on neighboring nodes across the topology. Higher feature homophily exhibits a synergy with label homophily: under high label homophily, it draws intra-class features closer together to enhance distinguishability, whereas under low label homophily, it can pull inter-class representations together and degrade the performance of GNNs.

GNNs for Heterophily Graphs: The effect of heterophily graphs on GNNs has been extensively studied, since the impact of these structural factors strongly depends on the GNN architecture. Although GCNs perform well on high label homophily graphs, their efficacy degrades rapidly in low- to mid-label homophily graphs (Luan et al., 2023) because GCNs are designed to smooth features across the graph by aggregating information from neighboring nodes, making their performance dependent on the support of neighbors. To address this, Zhu et al. (2020) demonstrate that introducing *Ego-Neighbor Separation*, an architectural design also inherent in GraphSAGE (Hamilton et al., 2017), can significantly improve GNN performance in heterophily graphs because it can maintain distinct representations, which is crucial when connected nodes have different class labels or features. However, while prior literature has primarily focused on how these architectural choices affect predictive performance under heterophily, our work pivots to investigate how Ego-Neighbor Separation influences the *memorization* behavior of GNNs.

Memorization in GNNs: Extensive research has explored memorization in deep learning based on vision models (Feldman & Zhang, 2020; Stephenson et al., 2021b; Maini et al., 2023b) and language models (Carlini et al., 2019; 2023; Tirumala et al., 2022; Biderman et al., 2023; Kiyomaru et al., 2024; Zeng et al., 2024). In contrast, the study of memorization in graph-structured data remains largely unexplored, with Jamadandi et al. (2025) providing a foundational investigation. Their work identifies an inverse relationship between memorization and graph *label homophily*, showing that memorized nodes are found more frequently in low label homophily graphs than in high label homophily graphs, and indicating that nodes with high label inconsistency in their feature-space neighborhood property are more prone to memorization. This node property is usually found in nodes that are difficult to distinguish (Wei et al., 2023; Luan et al., 2023).

Therefore, our work investigates additional topological properties that can affect node distinguishability, such as feature homophily and node degree, since Zheng et al. (2024) demonstrate that the synergy between feature homophily and label homophily can enhance or degrade the performance of GNNs, and Ma et al. (2022); Subramonian et al. (2024) reveal a degree bias in GCNs, demonstrating that high-degree nodes are less prone to misclassifications. This indicates that different GNN architectures exhibit different inductive biases when processing structural signals. Furthermore, we investigate the inductive biases of GNNs, since each GNN yields different performance results across different topologies; for example, GCNs perform better in graphs with high label homophily and high node degrees, but exhibit lower performance in low label homophily settings, where GraphSAGE is able to achieve better accuracy. We further contrast *Ego-Neighbor Integration* and *Ego-Neighbor Separation* architectures to isolate how ego-neighbor mixing bias governs these dynamics.

3 BACKGROUND

Let $G = (\mathcal{V}, \mathcal{E})$ denote an undirected graph where $\mathcal{V}$ is the set of N nodes and $\mathcal{E}$ is the set of edges. We represent the graph topology using an adjacency matrix $A \in \{0,1\}^{N \times N}$, where $A_{uv} = 1$ if $(u, v) \in \mathcal{E}$ and 0 otherwise. Each node $v \in \mathcal{V}$ is associated with a feature vector $x_v \in \mathbb{R}^d$ and a ground-truth label $y_v \in \{0, 1, \ldots, C-1\}$, where C is the number of classes. We define the local neighborhood of node v as $\mathcal{N}(v) = \{u \in \mathcal{V} \mid (u, v) \in \mathcal{E}\}$, and its degree as $d_v = |\mathcal{N}(v)|$.

3.1 TOPOLOGICAL GRAPH

We introduce the concept that a node's susceptibility to memorization depends heavily on its *Node Support*, defined as the amount of useful support it receives from its neighborhood, which can vary depending on the structural topology. In this work, we focus on three primary topological factors that affect memorization in GNNs. The equations for measuring both node- and graph-level topological factors are reported in Appendix F.

(i) Label Homophily (hL): Label homophily represents how much a neighborhood shares the same label with the ego-node, while the high label homophily represents that the ego-node has more label support from the neighborhood. In contrast, low to mid label homophily will represent conflict or inconsistency of label support from the neighborhood.

(ii) Feature Homophily (hF): Feature Homophily is the degree of feature dependencies among nodes across a graph's topology, specifically measuring how much an ego-node's features are influenced by its surrounding neighbors. The more dependency, or positive feature homophily, means a node's features become more similar to those of its neighbors, while independence or zero feature homophily means a node's features are not dependent on those of its neighbors.

(iii) Node Degree (Deg): Node Degree represents the number of connections between the ego-node and its neighborhood. In graphs with high label homophily, a higher node degree amplifies the supporting label information from neighboring nodes, resulting in more accurate GNN predictions.

3.2 Quantifying node-level memorization

Memorization: In the context of classification tasks, deep neural networks are known to possess enough capacity to perfectly fit random labels (Zhang et al., 2017). In this work, we formalize memorization through the lens of label perturbation. Suppose a subset of training nodes $\mathcal{V}_{\text{noise}} \subset \mathcal{V}$ is assigned uniformly random corrupted labels $\tilde{y}_v \neq y_v$. A GNN model f_θ is said to memorize a perturbed node v if it successfully predicted the class for that specific node $\hat{y}_v = \arg\max f_\theta(x_v, G)$ that perfectly matches the corrupted label $\tilde{y}_v$.

Label Corruption Strategy: We evaluate memorization in GNNs using the widely adopted strategy of introducing corrupted labels (Arpit et al., 2017; Stephenson et al., 2021a; Maini et al., 2023a). A subset of training samples (10% of the training set) is randomly selected and assigned incorrect labels. Mislabeled nodes are atypical and do not share generalized patterns with other nodes within the same class, so the model needs to memorize them to minimize the training loss.

Metric: We define the Memorization Rate (MR) as the fraction of corrupted nodes that the model successfully fits during training:

$$\text{MR } (\%) = \frac{\text{Number of memorized corrupted nodes}}{\text{Total number of corrupted nodes}} \tag{1}$$

4 The Impact of Graph Topological Characteristics on Memorization in GNNs

In this section, we empirically investigate how each distinct topological graph property (including label homophily, feature homophily, and node degree) affects the ability of GNNs to memorize noisy label nodes.

Experimental Setup: To systematically isolate how different topological graph environments, including node degree, feature homophily, and label homophily, affect memorization in GNNs, we generate these graphs using the Contextual Stochastic Block Model with three types of Homophily (CSBM-3H) (Zheng et al., 2024). This synthetic method allows us to config the generated graph property such as range of node degree (Deg), feature homophily (hF), and label homophily (hL). To obtain more realistic node features instead of generating them from a Gaussian distribution, we also use the method from Zhu et al. (2020) by sampling node features from the Cora dataset. The details of the experimental settings are explained in Appendix C.

After generating synthetic, we apply the label corruption, as described in Section 3.2, to these synthetic graphs. Then, we train GNNs and measure the memorization rate using Equation 1 on GCN, GAT, and GraphSAGE; see Appendix D for the details of GNN training configuration.

Increasing Node Degree and Label Homophily can suppress memorization: As shown in Figure 3, we observe that when feature homophily is low, the ability of GCN, GAT, and GraphSAGE to memorize noisy labels is highly sensitive to node degree (Deg) and label homophily (hL). When the models are early-stopped at the best validation epoch, the figures show that higher levels of either node degree or label homophily can suppress memorization. However, when the models are trained to overfit, all models are able to fully memorize noisy-labeled nodes in low-node-degree settings. In contrast, in high-node-degree settings, only GraphSAGE is able to fully memorize noisy labels regardless of the level of label homophily. This indicates that when a GNN has more support from

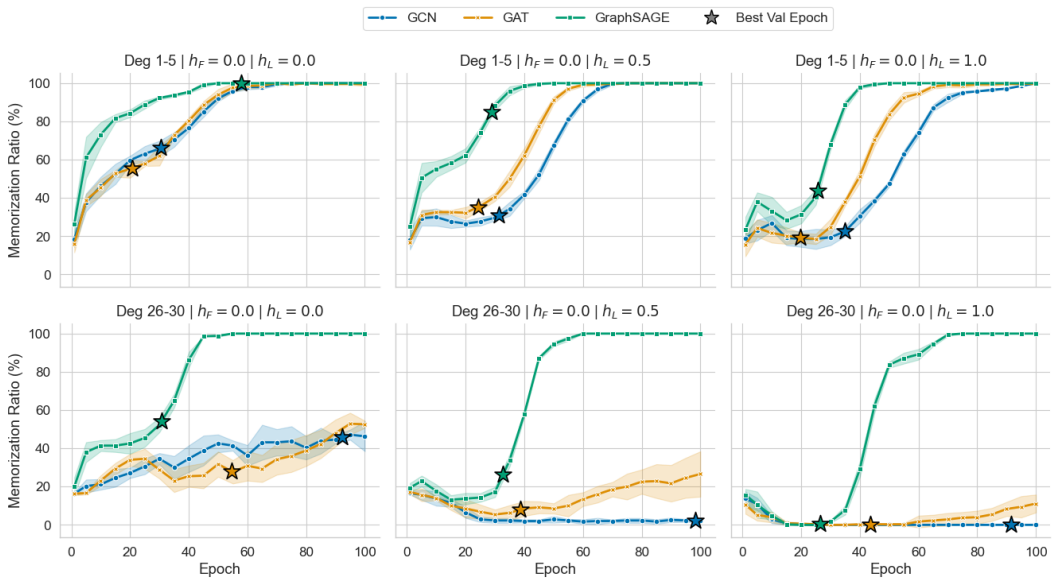

Figure 1: relationship between Memorization Rate and interplay of node degree and label homophily

neighbors within the same class (high node degree and high label homophily), the model is less likely to memorize noisy-labeled nodes.

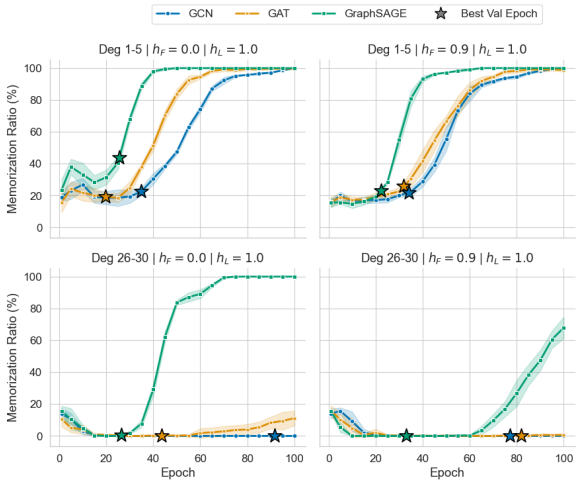

Figure 2: relationship between Memorization Rate and interplay of feature homophily and label homophily

Interplay between Feature and Label Homophily can suppress memorization: As shown in Figure 2, when the level of feature homophily increases in low-node-degree and high-label-homophily settings, it can decrease the memorization rate of GraphSAGE at the best validation epoch. Furthermore, as the level of feature homophily increases in high-node-degree and high-label-homophily settings, it also decreases the memorization rate of GraphSAGE at both the best validation epoch and the last epoch. This indicates a synergy among all three graph topological properties: when there is more useful support from neighbors that are within the same class and also share similar features, it drastically suppresses the memorization of noisy labels, even during the overfitting phase.

5 The Impact of Ego-Neighbor Separation Weight Layer in GNNs to Memorization

As shown in Section 4, GraphSAGE exhibits stronger memorization than GCN and GAT in high-node-degree settings during the overfitting phase. A key architectural difference among these models is the presence of the *ego-weight*. We therefore hypothesize that the ego-weight plays a significant role in influencing memorization.

5.1 EGO-NEIGHBOR INTEGRATION VS EGO-NEIGHBOR SEPARATION

To isolate the effect of ego-weights, we formalize GNN architectures into two categories: Ego-Neighbor Integration and Ego-Neighbor Separation.

Ego-Neighbor Integration (do not contain ego-weight): Architectures in this category fundamentally entangle the ego and neighbor representations. A prominent example is the GCN (Kipf & Welling, 2017). By adding a self-loop to the adjacency matrix ($\tilde{A} = A + I_N$), the central node v is explicitly included in its own neighborhood during the message-passing step. The update rule thus enforces a shared transformation:

$$z_v^{(l+1)} = W_{\text{share}}^{(l)} \cdot \text{AGG}\big(\{z_u^{(l)} \mid u \in \mathcal{N}(v) \cup \{v\}\}\big) \tag{2}$$

In Eq. 2, a single weight matrix $W_{\text{share}}^{(l)}$ transforms the aggregated AGG representation of the central node v and its neighbors $\mathcal{N}(v)$. This shared transformation makes it mathematically difficult for the model to disentangle the distinct ego-signal from the broader neighborhood consensus, inherently inducing a strong smoothing bias.

Ego-Neighbor Separation (contains ego-weight): In contrast, architectures of this type explicitly decouple the ego-transformation from the neighborhood aggregation. A representative example is GraphSAGE (Hamilton et al., 2017), which isolates these components in its update rule:

$$z_v^{(l+1)} = W_{\text{ego}}^{(l)} z_v^{(l)} + W_{\text{neigh}}^{(l)} \cdot \text{AGG}\big(z_u^{(l)} \mid u \in \mathcal{N}(v)\big) \tag{3}$$

In Eq. 3, the use of distinct matrices ($W_{\text{ego}}^{(l)} \neq W_{\text{neigh}}^{(l)}$) allows the model to weigh ego-features independently. This provides the architectural flexibility to preserve node-specific information, even against a contradictory neighborhood.

5.2 MEMORIZATION OF EGO-NEIGHBOR INTEGRATION AND SEPARATION GNNS

This experiment reveal a fundamental architectural tension in Graph Neural Networks: the mechanism that dictates how a model integrates or separates an ego node from its neighborhood fundamentally controls its generalization capability and its vulnerability to overfitting noise. By comparing variants of GNN that either explicitly decouple the ego-node (Ego-Neighbor Separation) or enforce shared transformations (Ego-Neighbor Integration), we uncover distinct memorization behavior driven by the underlying dataset properties.

Experimental Setup: We systematically modify the canonical implementations of GCN, GAT, and GraphSAGE. For each architecture, we construct two strict variants: an **Integration** variant that enforces a shared weight matrix for both the ego node and its neighbors, and a **Separation** variant that explicitly decouples the ego node using an independent weight matrix. Detailed equations for each variant of GCN, GAT, and GraphSAGE are provided in Appendix G. To ensure that Separation GNNs do not have an advantage from the additional parameters of the ego-weight, we increase the parameters of the Integration GNNs to match that of the Separation models by increasing the hidden size of the neighbor layer. Table 1 presents the results for GCN. We observe similar trends for other GNN architectures, which are reported in Tables 5–6.

Ego-Neighbor Separation Performance Trade-off in the Cost of Memorization: While Ego-Neighbor Separation consistently outperforms its integrated counterparts in low label homophily graph, it also introduces a severe vulnerability to memorize noisy labels, especially in even in high node degree environments. This vulnerability arises because the independent ego-weight allows the model to fit noisy labels without affecting neighborhood representations.

Memorization rate decreases with high feature homophily in high-degree graphs for Ego-Neighbor Separation: While Ego-Neighbor Separation creates severe vulnerabilities in almost every topology environment, increasing feature homophily in high node degree graphs drastically mitigates these vulnerabilities. This pattern reveals that highly similar features act as a powerful defense against overfitting, even when the architecture provides a dedicated ego-pathway due to the target node shares nearly identical features with its neighbors, the model cannot arbitrarily manipulate the ego-weights to fit a single mislabeled node without catastrophically misclassifying the surrounding clean nodes.

Table 1: Separation vs Integration GCN on Synthetic Cora dataset

				Train Clean Acc (%)		Test Acc (%)		MR (%)	
				Best Val	Last	Best Val	Last	Best Val	Last
Deg	hF	hL	Model						
1 5	0.00	0.00	Sep.	**100.00 ± 0.00**	**100.00 ± 0.00**	**57.46 ± 0.76**	**57.53 ± 0.69**	**100.00 ± 0.00**	**100.00 ± 0.00**
			Int.	82.70 ± 3.88	99.84 ± 0.00	32.35 ± 0.54	26.68 ± 0.11	64.57 ± 6.72	**100.00 ± 0.00**
		0.50	Sep.	**99.62 ± 0.76**	**100.00 ± 0.00**	**62.60 ± 0.69**	**62.35 ± 0.40**	**96.29 ± 7.43**	**100.00 ± 0.00**
			Int.	84.92 ± 1.41	**100.00 ± 0.00**	58.61 ± 0.42	48.61 ± 0.17	29.14 ± 4.20	**100.00 ± 0.00**
		1.00	Sep.	**97.71 ± 0.54**	**100.00 ± 0.00**	85.34 ± 0.42	**82.21 ± 0.20**	**38.00 ± 3.68**	**100.00 ± 0.00**
			Int.	93.56 ± 1.91	**100.00 ± 0.00**	**85.39 ± 0.32**	77.46 ± 0.09	23.14 ± 6.85	**100.00 ± 0.00**
	0.90	0.00	Sep.	**99.97 ± 0.06**	99.97 ± 0.06	**32.57 ± 0.43**	**32.61 ± 0.44**	**99.71 ± 0.57**	99.71 ± 0.57
			Int.	86.29 ± 4.22	**100.00 ± 0.00**	26.54 ± 0.33	24.06 ± 0.29	81.14 ± 2.77	**100.00 ± 0.00**
		0.50	Sep.	**86.51 ± 3.65**	**100.00 ± 0.00**	52.88 ± 0.84	**51.33 ± 0.16**	**44.86 ± 8.79**	99.71 ± 0.57
			Int.	78.79 ± 0.84	99.84 ± 0.00	**53.50 ± 0.31**	45.97 ± 0.28	29.43 ± 3.45	**100.00 ± 0.00**
		1.00	Sep.	**89.08 ± 1.60**	99.90 ± 0.08	81.21 ± 0.57	**74.83 ± 0.13**	**20.00 ± 2.39**	**100.00 ± 0.00**
			Int.	85.84 ± 0.87	**100.00 ± 0.00**	**82.61 ± 0.25**	73.20 ± 0.26	18.86 ± 1.07	**100.00 ± 0.00**
26 30	0.00	0.00	Sep.	**99.02 ± 0.58**	**100.00 ± 0.00**	**78.60 ± 1.30**	**75.36 ± 0.86**	**51.71 ± 9.14**	**100.00 ± 0.00**
			Int.	72.19 ± 2.50	65.56 ± 5.35	33.94 ± 1.16	30.15 ± 2.86	50.86 ± 5.90	41.14 ± 2.29
		0.50	Sep.	**99.17 ± 0.81**	**100.00 ± 0.00**	**93.45 ± 0.53**	87.67 ± 0.42	**11.71 ± 6.22**	**100.00 ± 0.00**
			Int.	91.97 ± 0.38	92.19 ± 0.21	89.45 ± 0.21	**89.53 ± 0.09**	2.29 ± 0.70	2.29 ± 0.70
		1.00	Sep.	**100.00 ± 0.00**	**100.00 ± 0.00**	99.59 ± 0.36	96.76 ± 0.07	**7.43 ± 9.14**	**100.00 ± 0.00**
			Int.	99.84 ± 0.00	99.84 ± 0.00	**99.80 ± 0.02**	**99.78 ± 0.01**	0.00 ± 0.00	0.00 ± 0.00
	0.90	0.00	Sep.	**94.63 ± 2.36**	**95.37 ± 2.55**	**65.32 ± 3.07**	**65.45 ± 2.95**	**52.57 ± 5.06**	**54.57 ± 3.55**
			Int.	23.02 ± 1.06	20.41 ± 2.17	15.54 ± 0.41	14.88 ± 0.41	27.71 ± 1.46	22.86 ± 2.86
		0.50	Sep.	**98.25 ± 0.22**	**98.57 ± 0.14**	**93.71 ± 0.67**	**93.79 ± 0.65**	**5.14 ± 1.46**	4.86 ± 1.46
			Int.	79.87 ± 3.69	57.78 ± 17.78	77.84 ± 3.24	56.61 ± 17.18	3.14 ± 0.57	**6.57 ± 4.66**
		1.00	Sep.	**100.00 ± 0.00**	**99.97 ± 0.06**	99.76 ± 0.10	99.62 ± 0.24	**1.14 ± 2.29**	**1.71 ± 2.29**
			Int.	99.87 ± 0.06	99.87 ± 0.06	**99.84 ± 0.03**	**99.85 ± 0.03**	0.00 ± 0.00	0.00 ± 0.00

5.3 How Much the Ego-Neighbor Separation Layer Contributes to Memorization

Based on the experimental results in Section 5.2, the Ego-Separation model has a higher memorization rate than the Ego-Integration model in many environments. In this subsection, we further investigate how much the ego-weight layer contributes to the memorization of these noisy-labeled nodes across different topologies.

Experimental Setup: We perform Layer Rewinding (Stephenson et al., 2021a; Maini et al., 2023b) and Layer Retraining (Maini et al., 2023b) experiments by initializing the weights of a specific layer ($W^{(l)}_{\text{ego}}$ or $W^{(l)}_{\text{neigh}}$) and freezing the weights of the other layers, then retraining the model with only clean training samples (excluding noisy-labeled training samples) and observing the memorization rate of the noisy-labeled samples. After retraining the model, if the memorization rate of a specific layer decreases, it implies that the specific layer is responsible for or contributes to the memorization of these noisy-labeled samples.

The ego-layer contributes to memorization in high label homophily and low-node-degree environments: Based on the results in Figure 3, we observe that when we initialize the weights of the ego-layer (W_{ego}), freeze the other layers, and retrain with clean training nodes, the memorization rate decreases as the model is trained further. This phenomenon is more pronounced in settings with high label homophily. In contrast, when we perform the same procedure with the convolutional layer, we find that the memorization rate only slightly decreases. This indicates that the convolutional layer does not contribute to the memorization of noisy-labeled nodes as much as the ego-layer.

The ego-layer contributes less to memorization in low feature homophily and high-node-degree environments: Based on the results in Figure 3, for high-node-degree environments, rewinding and retraining the convolutional layer show a similar pattern to that observed in low-node-degree environments. However, for the ego-layer (W_{ego}), we find that the previous pattern of the memorization rate observed in low-node-degree environments applies only to low feature homophily; when feature homophily increases, it results in only a slight decrease in the memorization rate.

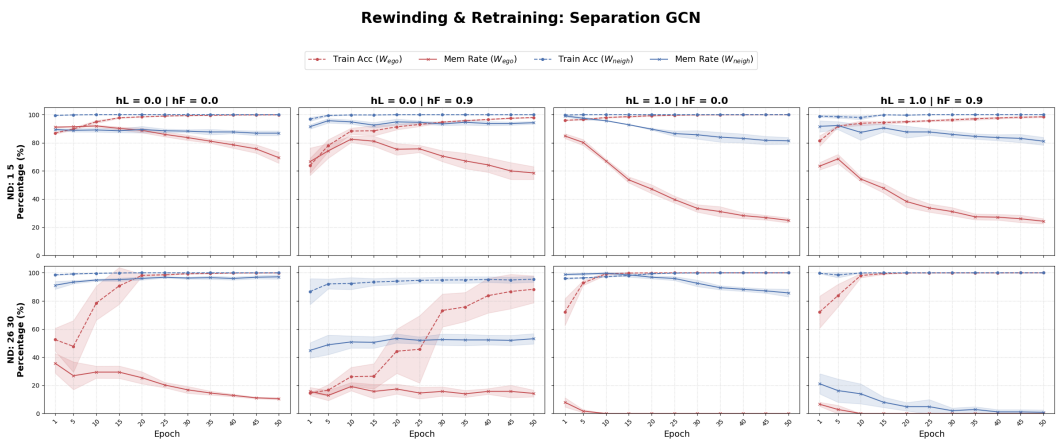

Figure 3: Ego-layer Retraining: Clean Training accuracy vs Memorization Rate of Separation GCN on Syn-Cora dataset.

6 Privacy Attacks on Ego-Neighbor Separation GNNs

As demonstrated in Section 5, architectures utilizing Ego-Neighbor Separation generally exhibit higher memorization rates than their Integration counterparts. Because extensive literature (Haim et al., 2022; Buzaglo et al., 2023) establishes that memorization can lead to privacy vulnerabilities, this section investigates the susceptibility of these models to privacy attacks. Specifically, we conduct an empirical study to demonstrate that Ego-Neighbor Separation GNNs carry significantly higher privacy risks.

Experimental Setup: To demonstrate the impact of the ego-weight layer on privacy vulnerabilities, we employ a Membership Inference Attack (MIA) against both Ego-Neighbor Integration and Separation GCNs on the synthetic data from Section 5.2 and on real-world datasets (the dataset statistics are reported in Appendix H). Specifically, we utilize the Likelihood Ratio Attack (LiRA) method in an online setting (Carlini et al., 2022). Full experimental details are provided in Appendix E. We measure the attack success rate using the True Positive Rate at a 1% False Positive Rate (TPR@FPR1%), where higher values indicate a greater privacy risk.

Ego-Neighbor Separation GNNs are more vulnerable to privacy attacks than their Integration counterparts: The results in Tables 2–3 show consistently higher attack success rates for Ego-Neighbor Separation models than for Ego-Neighbor Integration models for every real-world dataset and for synthetic datasets with low to mid label homophily. Notably, while the Separation GCN achieves better train and test accuracy on heterophily datasets (Chameleon, Squirrel, and Actor), the gap in attack scores between these two models is also significantly larger when compared to homophily datasets (Cora, CiteSeer, PubMed, Computers, and Photo).

7 Conclusion

This work demonstrates that memorization in Graph Neural Networks is driven not by topological characteristics alone, but by the critical interplay between a node's structural support and the model's architectural inductive bias. Expanding beyond label homophily, we establish that broader structural dimensions, specifically node degree and feature homophily, also affect memorization in GNNs. Notably, high-node-degree environments, especially those with high feature homophily and high label homophily, can effectively suppress the memorization ability of GNNs. By formalizing GNN architectures into Ego-Integration and Ego-Separation paradigms, we reveal a fundamental trade-off. Ego-Neighbor Separation models (e.g., GraphSAGE) maintain predictive performance in heterophily graphs by processing the ego-node independently; however, this dedicated weight layer acts as a structural bypass that exacerbates the memorization of noisy labels. Conversely, Ego-Neighbor Integration models (e.g., GCN) that use a shared transformation inherently suppress noisy-label memorization when surrounded by abundant useful neighbor support. Crucially, we

Table 2: Separation vs Integration GCN: MIA on Synthetic Cora dataset

Deg	hF	hL	Model	Train Clean Acc (%)	Test Acc (%)	TPR@FPR1% (%)
1 5	0.0	0.0	Sep.	**100.00 ± 0.00**	**57.53 ± 0.69**	**26.31 ± 0.50**
			Int.	99.84 ± 0.00	26.68 ± 0.11	6.11 ± 0.19
		0.5	Sep.	**100.00 ± 0.00**	**62.35 ± 0.40**	**35.74 ± 1.37**
			Int.	**100.00 ± 0.00**	48.61 ± 0.17	11.71 ± 0.24
		1.0	Sep.	**100.00 ± 0.00**	**82.21 ± 0.20**	**30.34 ± 0.64**
			Int.	**100.00 ± 0.00**	77.46 ± 0.09	17.83 ± 0.33
	0.9	0.0	Sep.	99.97 ± 0.06	**32.61 ± 0.44**	4.14 ± 0.56
			Int.	**100.00 ± 0.00**	24.06 ± 0.29	**4.26 ± 0.26**
		0.5	Sep.	**100.00 ± 0.00**	**51.33 ± 0.16**	**4.77 ± 0.71**
			Int.	99.84 ± 0.00	45.97 ± 0.28	1.91 ± 0.07
		1.0	Sep.	99.90 ± 0.08	**74.83 ± 0.13**	**18.69 ± 0.55**
			Int.	**100.00 ± 0.00**	73.20 ± 0.26	11.57 ± 0.87
26 30	0.0	0.0	Sep.	**100.00 ± 0.00**	**75.36 ± 0.86**	**19.00 ± 0.63**
			Int.	65.56 ± 5.35	30.15 ± 2.86	8.63 ± 1.98
		0.5	Sep.	**100.00 ± 0.00**	87.67 ± 0.42	**17.97 ± 0.40**
			Int.	92.19 ± 0.21	**89.53 ± 0.09**	6.66 ± 0.11
		1.0	Sep.	**100.00 ± 0.00**	96.76 ± 0.07	9.63 ± 0.21
			Int.	99.84 ± 0.00	**99.78 ± 0.01**	**10.57 ± 0.00**
	0.9	0.0	Sep.	**95.37 ± 2.55**	**65.45 ± 2.95**	**13.31 ± 0.78**
			Int.	20.41 ± 2.17	14.88 ± 0.41	0.71 ± 0.00
		0.5	Sep.	**98.57 ± 0.14**	**93.79 ± 0.65**	**11.40 ± 1.17**
			Int.	57.78 ± 17.78	56.61 ± 17.18	0.66 ± 0.07
		1.0	Sep.	**99.97 ± 0.06**	99.62 ± 0.24	**12.83 ± 0.52**
			Int.	99.87 ± 0.06	**99.85 ± 0.03**	9.51 ± 0.11

Table 3: Separation vs Integration GCN: MIA on Realworld dataset

Dataset	Model	Train Clean Acc (%)	Test Acc (%)	TPR@FPR1% (%)
Cora	Sep.	**96.08 ± 0.20**	79.56 ± 0.82	**8.95 ± 1.49**
	Int.	94.42 ± 0.50	**80.94 ± 0.51**	8.80 ± 0.18
CiteSeer	Sep.	**95.08 ± 1.14**	68.24 ± 0.63	**15.58 ± 3.92**
	Int.	93.60 ± 1.00	**69.15 ± 0.31**	6.48 ± 0.41
PubMed	Sep.	**89.67 ± 0.75**	**83.83 ± 0.08**	**4.90 ± 0.23**
	Int.	87.70 ± 0.56	83.83 ± 0.16	4.02 ± 0.05
Computers	Sep.	**97.50 ± 0.59**	**85.09 ± 0.50**	**10.86 ± 0.38**
	Int.	93.40 ± 0.50	81.86 ± 0.95	7.53 ± 0.16
Photo	Sep.	**97.72 ± 0.59**	**91.63 ± 0.29**	**11.47 ± 0.23**
	Int.	93.57 ± 0.45	90.40 ± 0.61	8.92 ± 0.13
Chameleon	Sep.	**95.76 ± 1.61**	**36.63 ± 1.31**	**42.76 ± 15.14**
	Int.	78.72 ± 4.13	34.99 ± 0.67	1.16 ± 0.22
Squirrel	Sep.	**98.29 ± 1.07**	**29.30 ± 0.31**	**28.00 ± 0.87**
	Int.	93.08 ± 2.48	25.34 ± 0.43	4.19 ± 0.28
Actor	Sep.	**94.82 ± 1.37**	**27.63 ± 0.39**	**20.08 ± 4.84**
	Int.	67.43 ± 12.34	22.62 ± 0.79	6.55 ± 0.47

show that equipping GNN models with an ego-weight layer increases their vulnerability to Membership Inference Attacks (MIAs). This indicates that Ego-Neighbor Separation can improve predictive performance but comes with a hidden cost to model robustness. This highlights the need to carefully design future GNN architectures; specifically, models employing the Ego-Neighbor Separation concept must incorporate mitigation methods to avoid compromising generalization and privacy.

REFERENCES

Devansh Arpit, Stanisław Jastrzundefinedbski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, and Simon Lacoste-Julien. A closer look at memorization in deep networks. ICML'17, pp. 233–242. JMLR.org, 2017.

Robert J. N. Baldock, Hartmut Maennel, and Behnam Neyshabur. Deep learning through the lens of example difficulty. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 10876–10889, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/5a4b25aaed25c2ee1b74de72dc03c14e-Abstract.html.

Stella Biderman, USVSN Sai Prashanth, Lintang Sutawika, Hailey Schoelkopf, Quentin Gregory Anthony, Shivanshu Purohit, and Edward Raff. Emergent and predictable memorization in large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=Iq0DvhB4Kf.

Gon Buzaglo, Niv Haim, Gilad Yehudai, Gal Vardi, Yakir Oz, Yaniv Nikankin, and Michal Irani. Deconstructing data reconstruction: Multiclass, weight decay and general losses. *Advances in Neural Information Processing Systems*, 36:51515–51535, 2023.

Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: evaluating and testing unintended memorization in neural networks. In *Proceedings of the 28th USENIX Conference on Security Symposium*, SEC'19, pp. 267–284, USA, 2019. USENIX Association. ISBN 9781939133069.

Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramèr. Membership inference attacks from first principles. In *2022 IEEE Symposium on Security and Privacy (SP)*, pp. 1897–1914, 2022. doi: 10.1109/SP46214.2022.9833649.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. Quantifying memorization across neural language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=TatRHT_1cK.

Vitaly Feldman and Chiyuan Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/1e14bfe2714193e7af5abc64ecbd6b46-Abstract.html.

Matthias Fey, Jinu Sunil, Akihiro Nitta, Rishi Puri, Manan Shah, Blaz Stojanovic, Ramona Bendias, Alexandria Barghi, Vid Kocijan, Zecheng Zhang, Xinwei He, Jan Eric Lenssen, and Jure Leskovec. Pyg 2.0: Scalable learning on real world graphs. *CoRR*, abs/2507.16991, 2025. doi: 10.48550/ARXIV.2507.16991. URL https://doi.org/10.48550/arXiv.2507.16991.

Niv Haim, Gal Vardi, Gilad Yehudai, Ohad Shamir, and Michal Irani. Reconstructing training data from trained neural networks. *Advances in Neural Information Processing Systems*, 35:22911–22924, 2022.

William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 1024–1034, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html.

Adarsh Jamadandi, Jing Xu, Adam Dziedzic, and Franziska Boenisch. Memorization in graph neural networks. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=jAJv3CaS4f.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Hirokazu Kiyomaru, Issa Sugiura, Daisuke Kawahara, and Sadao Kurohashi. A comprehensive analysis of memorization in large language models. In Saad Mahamood, Nguyen Le Minh, and Daphne Ippolito (eds.), *Proceedings of the 17th International Natural Language Generation Conference*, pp. 584–596, Tokyo, Japan, September 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.inlg-main.45. URL https://aclanthology.org/2024.inlg-main.45/.

Xiang Li, Renyu Zhu, Yao Cheng, Caihua Shan, Siqiang Luo, Dongsheng Li, and Weining Qian. Finding global homophily in graph neural networks when meeting heterophily. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato (eds.), *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, Proceedings of Machine Learning Research, pp. 13242–13256. PMLR, 2022. URL https://proceedings.mlr.press/v162/li22ad.html.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Sitao Luan, Chenqing Hua, Minkai Xu, Qincheng Lu, Jiaqi Zhu, Xiao-Wen Chang, Jie Fu, Jure Leskovec, and Doina Precup. When do graph neural networks help with node classification? investigating the homophily principle on node distinguishability. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/5ba11de4c74548071899cf41dec078bf-Abstract-Conference.html.

Yao Ma, Xiaorui Liu, Neil Shah, and Jiliang Tang. Is homophily a necessity for graph neural networks? In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=ucASPPD9GKN.

Pratyush Maini, Michael C Mozer, Hanie Sedghi, Zachary C Lipton, J Zico Kolter, and Chiyuan Zhang. Can neural network memorization be localized? *arXiv preprint arXiv:2307.09542*, 2023a.

Pratyush Maini, Michael Curtis Mozer, Hanie Sedghi, Zachary Chase Lipton, J. Zico Kolter, and Chiyuan Zhang. Can neural network memorization be localized? In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 23536–23557. PMLR, 2023b. URL https://proceedings.mlr.press/v202/maini23a.html.

Jiawen Qin, Haonan Yuan, Qingyun Sun, Lyujin Xu, Jiaqi Yuan, Pengfeng Huang, Zhaonan Wang, Xingcheng Fu, Hao Peng, Jianxin Li, and Philip S. Yu. IGL-bench: Establishing the comprehensive benchmark for imbalanced graph learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=uTqnyF0JNR.

Cory Stephenson, Suchismita Padhy, Abhinav Ganesh, Yue Hui, Hanlin Tang, and SueYeon Chung. On the geometry of generalization and memorization in deep neural networks. *arXiv preprint arXiv:2105.14602*, 2021a.

Cory Stephenson, Suchismita Padhy, Abhinav Ganesh, Yue Hui, Hanlin Tang, and SueYeon Chung. On the geometry of generalization and memorization in deep neural networks. In *9th International*

Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021. OpenReview.net, 2021b. URL https://openreview.net/forum?id=V8jrrnwGbuc.

Arjun Subramonian, Jian Kang, and Yizhou Sun. Theoretical and empirical insights into the origins of degree bias in graph neural networks. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/0f94c552e5fe82bc152494985e34bd48-Abstract-Conference.html.

Kushal Tirumala, Aram H. Markosyan, Luke Zettlemoyer, and Armen Aghajanyan. Memorization without overfitting: analyzing the training dynamics of large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Xiaowen Wei, Xiuwen Gong, Yibing Zhan, Bo Du, Yong Luo, and Wenbin Hu. Clnode: Curriculum learning for node classification. In Tat-Seng Chua, Hady W. Lauw, Luo Si, Evimaria Terzi, and Panayiotis Tsaparas (eds.), *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining, WSDM 2023, Singapore, 27 February 2023 - 3 March 2023*, pp. 670–678. ACM, 2023. doi: 10.1145/3539597.3570385. URL https://doi.org/10.1145/3539597.3570385.

Liang Yang, Mengzhe Li, Liyang Liu, Bingxin Niu, Chuan Wang, Xiaochun Cao, and Yuanfang Guo. Diverse message passing for attribute with heterophily. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 4751–4763, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/253614bbac999b38b5b60cae531c4969-Abstract.html.

Shenglai Zeng, Yaxin Li, Jie Ren, Yiding Liu, Han Xu, Pengfei He, Yue Xing, Shuaiqiang Wang, Jiliang Tang, and Dawei Yin. Exploring memorization in fine-tuned language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3917–3948, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.216. URL https://aclanthology.org/2024.acl-long.216/.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Sy8gdB9xx.

Yilun Zheng, Sitao Luan, and Lihui Chen. What is missing for graph homophily? disentangling graph homophily for graph neural networks. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=GmdGEF8xxU.

Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/58ae23d878a47004366189884c2f8440-Abstract.html.

A LIMITATIONS

While our study provides valuable insights into memorization vulnerabilities within Graph Neural Networks (GNNs), we acknowledge certain limitations in our current scope. First, our empirical

observations and theoretical analyses focus exclusively on node classification rather than graph-level classification. Although both domains utilize graph-structured data, the theoretical frameworks we employ particularly those concerning local neighborhood topology and homophily metrics are fundamentally rooted in node-level concepts. Extending these analyses to graph-level tasks would require redefining these topological metrics for entire graph structures.

Furthermore, our experiments are restricted to homogeneous graphs. We leave the exploration of heterogeneous graphs to future work, primarily because the theoretical foundations for defining and accurately estimating homophily in heterogeneous networks remain largely underdeveloped. Adapting our topology-aware methodology to heterogeneous structures represents a non-trivial challenge beyond the scope of this paper.

B Experimental Details & Compute Resources

Every experiment was conducted using Pytorch Geometric 2.0 (Fey et al., 2025) and executed on an RTX 5060 Ti GPU with 16GB memory. Models were trained under consistent settings across all experiments, including fixed random 5 seeds, identical data splits, and standardized optimization procedures to ensure fair comparison and reproducibility. Hyperparameters were selected based on validation performance and kept consistent unless otherwise specified.

Hyperparameters For our Graph Neural Network backbones (GCN, GAT, and GraphSAGE), we employ a 3-layer architecture with a hidden dimensionality of 128. Specifically for the GAT model, we configure 1 attention heads. All models are trained for a maximum of 100 epochs using the AdamW optimizer (Loshchilov & Hutter, 2019) with a learning rate of 0.001. We determine the final model weights via early stopping, retaining the checkpoint that achieves the highest accuracy on the validation set.

Compute Time of Experiments For the experiments analyzing topology and memorization in GNNs (Section 4), each individual run takes approximately one hour per model architecture, consistent across both the synthetic and real-world datasets. Similarly, the experiments evaluating memorization under the Ego-Neighbor Separation and Ego-Neighbor Integration paradigms in Section 5 also require roughly one hour per GNN model type. The computational overhead increases significantly for the Membership Inference Attack (MIA) experiments detailed in Section 6. Specifically, the complete MIA pipeline takes about 5 hours per run for the GCN and GraphSAGE architectures, whereas the GAT architecture requires approximately 10 hours per run due to the added complexity of computing attention weights.

C Synthetic Graph Generation Details

To rigorously evaluate the model under varying and isolated graph conditions, we generate synthetic datasets based on Contextual Stochastic Block Models (CSBMs). This generation process allows us to explicitly disentangle and control three distinct dimensions of graph homophily: Label Homophily (h_L), Structural Homophily (h_S), and Feature Homophily (h_F), following the method introduced in prior work (Zheng et al., 2024).

C.1 General Dataset Statistics

Each synthetic graph is generated with a fixed size and feature space to maintain consistency across comparative experiments:

- **Total Nodes:** $7,000$ nodes.
- **Total Classes:** 7 distinct classes, with labels assigned uniformly at random.
- **Node Feature Dimension:** 1433 dimensions.

C.2 Disentangled Homophily Configurations

We systematically vary the homophily parameters to observe their independent and combined effects on the models. The configurations evaluated in our experiments include:

- **Label Homophily (h_L):** Evaluated at 0.0, 0.5, and 1.0 to represent low label homophily, mid label homophily, and high label homophily, respectively.
- **Structural Homophily (h_S):** Fixed at 0.5 across all graphs to maintain a consistent neighborhood class distribution baseline.
- **Feature Homophily (h_F):** Evaluated at 0.0, which refers to low feature homophily (feature distributions are independent of the graph topology) and 0.9 refer to high feature homophily (feature distributions are dependent on the graph topology).

C.3 Topology and Edge Generation

To observe the impact of graph density and average degree, we generate topologies under two distinct node degree distributions:

- **Sparse Graphs:** Node degrees are uniformly sampled from the range $[1, 5]$.
- **Dense Graphs:** Node degrees are uniformly sampled from the range $[26, 30]$.

Edges are formed by sampling from a Bernoulli distribution. The connection probability between any two nodes is scaled by their respective generated node degrees and a neighborhood distribution matrix. This matrix explicitly incorporates both the defined label homophily (h_L) and the structural homophily variance (h_S).

C.4 Feature Generation

Node features are initially drawn from a multivariate normal distribution $\mathcal{N}(\mu, \Sigma)$. For each feature dimension, a unique mean and variance are randomly generated for each of the 10 classes to create the base feature matrix X_0.

To inject the desired feature homophily (h_F), the base features are propagated over the graph topology using the following relationship:

$$X = \left(I - \frac{h_F}{\rho} A\right)^{-1} X_0 \tag{4}$$

where A is the symmetric adjacency matrix and ρ is the spectral radius of A. This spatial propagation ensures that the final node features X are contextually tied to the graph structure strictly according to the designated h_F parameter.

C.5 Data Splitting

For our evaluation, we adopt the rigorous data splits and methodology established in recent graph imbalanced learning research (Qin et al., 2025). Specifically, we operate under a low local topology imbalance setting to guarantee that both class distributions and node degrees remain consistent and balanced across the training, validation, and test sets.

- **Split Ratios:** The dataset is divided into 10% training, 10% validation, and 80% testing nodes.
- **Class Balancing:** The 700 training and 700 validation nodes are perfectly balanced across the 7 classes, ensuring exactly 100 nodes per class in each respective set.

D GNN Training Configuration

D.1 Target and Shadow Model Architecture

The standardized GNN configuration is as follows:

- **Network Depth:** 3 message-passing layers.
- **Hidden Dimensionality:** 128 dimensions per hidden layer.
- **Attention Heads:** 1 attention heads per layer (GAT only).

D.2 Training Hyperparameters

Each GNN share an identical training configuration:

- **Optimization:** Models are trained using the AdamW optimizer with a learning rate of 0.001 and a weight decay of 0.01.
- **Loss Function:** Standard Cross-Entropy Loss.
- **Epochs & Selection:** Models are trained for a maximum of 100 epochs. The final model weights are selected based on the epoch that achieves the highest accuracy on the validation set (early stopping via best checkpointing).

E Online LiRA Configuration

In this section, we provide a comprehensive overview of the experimental setup for the Likelihood Ratio Attack (LiRA) on Graph Neural Networks (GNNs) in Section 6, including model architecture, training hyperparameters, and the attack configuration.

The attacker employs an online Likelihood Ratio Attack (LiRA). In the online setting, the attacker relies solely on "IN" and "OUT" shadow models to estimate the prediction distribution.

E.1 Dataset Configuration

The experiments are conducted on synthetic graph datasets specifically designed to evaluate the impact of various graph properties and homophily metrics on membership inference vulnerability.

- **Graph Size & Features:** Each generated graph consists of $7,000$ nodes. Node features are generated with a dimensionality of 1433, and the nodes are uniformly distributed across 7 distinct classes.
- **Topological Properties:** The graphs are generated with two varying node degree ranges to simulate dense and sparse environments: dense graphs have an average degree range of $[26, 30]$, while sparse graphs have an average degree range of $[1, 5]$.
- **Homophily Settings:** The graphs are synthesized with controlled levels of homophily across three dimensions:

E.2 Target and Shadow Model Architecture

The primary architecture utilized for both the target model and the shadow models is a GNN. We note that our framework also supports alternative GNN backbones. The standardized GNN configuration is as follows:

- **Network Depth:** 3 message-passing layers.
- **Hidden Dimensionality:** 128 dimensions per hidden layer.
- **Attention Heads:** 2 attention heads per layer (GAT only).

E.3 Training Hyperparameters

Both the target and shadow models share an identical training configuration to mimic a online attacker scenario where the adversary has white-box knowledge of the training pipeline:

- **Optimization:** Models are trained using the AdamW optimizer with a learning rate of 0.001 and a weight decay of 0.01.

- **Loss Function:** Standard Cross-Entropy Loss.
- **Epochs & Selection:** Models are trained for a maximum of 100 epochs. The final model weights are selected based on the epoch that achieves the highest accuracy on the validation set (early stopping via best checkpointing).

Shadow Model Generation. A total of 256 shadow models are trained per experiment. For each shadow model pair, a binary mask is generated with a 0.5 inclusion probability for each node from the pool of available training data. The exact inverse mask ($1 - \text{mask}$) is used for the corresponding paired shadow model. This methodology ensures approximately 128 "IN" and 128 "OUT" models for any given target node.

Confidence Scoring. The raw output probabilities of the shadow models are scaled to prevent numerical instability and bounded logits. We use a stable logit scaling method for the confidence score ϕ:

$$\phi(x, y) = \log(P(y|x)) - \log\left(\sum_{y' \neq y} P(y'|x)\right) \tag{5}$$

Likelihood Estimation. For a given target node, the attacker fits a Gaussian distribution $\mathcal{N}(\mu_{\text{out}}, \sigma^2_{\text{out}})$ using the scaled ϕ scores generated strictly by the shadow models that did *not* include the target node in their training set (the IN and OUT models). The final membership inference score is derived by computing the log probability density of the target model's confidence score against this fitted IN and OUT distribution.

Evaluation Metrics To account for variance in random initializations and data splits, the entire experiment is averaged over 5 independent random seeds. The effectiveness of the attack is evaluated using standard membership inference metrics:

- **TPR @ Low FPR:** The True Positive Rate (Attack Success Rate) measured at fixed low False Positive Rates of 1% and 10% to evaluate the attack's practical precision and reliability.

F Topological Factors

Node-level label homophily measures the proportion of a node's neighbors that share its ground-truth class. Formally, for a given node v, it is defined as:

$$h_{\text{label}}(v) = \frac{|\{u \in \mathcal{N}(v) \mid y_u = y_v\}|}{d_v} \tag{6}$$

where d_v denotes its degree and y_v is its corresponding class label. Nodes with a low $h_{\text{label}}(v)$ reside in heterophilous neighborhoods and often represent harder examples for standard GNNs to classify. Consequently, the graph-level label homophily for a graph $G = (\mathcal{V}, \mathcal{E})$ is defined as the average node-level homophily across all nodes in the graph:

$$h_{\text{label}}(G) = \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} h_{\text{label}}(v) \tag{7}$$

(ii) Feature Homophily (hF): Node-level feature homophily captures the similarity between a node's own features and the aggregated features of its neighborhood. We measure this using cosine similarity:

$$h_{\text{feat}}(v) = \cos\left(x_v, \frac{1}{d_v} \sum_{u \in \mathcal{N}(v)} x_u\right) \tag{8}$$

Consequently, the graph-level feature homophily for a graph $G = (\mathcal{V}, \mathcal{E})$ is defined as the average node-level homophily across all nodes in the graph:

$$h_{\text{feat}}(G) = \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} h_{\text{feat}}(v) \tag{9}$$

G Ego-Neighbor Integration and Ego-Neighbor Separation Equations

GCN Variants. The standard GCN relies on isotropic smoothing via the renormalization trick, serving as our *Integration (Neighbor)* baseline. It injects self-loops into the adjacency matrix and applies a single, shared weight matrix $\mathbf{W}$ to both the ego node and its augmented neighborhood:

$$\mathbf{x}_i^{(k)} = \sum_{j \in \mathcal{N}(i) \cup \{i\}} \frac{1}{\sqrt{\deg(i)\deg(j)}} \mathbf{W}\mathbf{x}_j^{(k-1)} \tag{10}$$

To construct the *Separation (Ego)* variant, we remove the self-loops from the graph structure and introduce an independent weight matrix $\mathbf{W}_{\text{ego}}$ exclusively for the ego node. The update rule is thus decoupled as follows:

$$\mathbf{x}_i^{(k)} = \mathbf{W}_{\text{ego}}\mathbf{x}_i^{(k-1)} + \sum_{j \in \mathcal{N}(i)} \frac{1}{\sqrt{\deg(i)\deg(j)}} \mathbf{W}_{\text{neighbor}}\mathbf{x}_j^{(k-1)} \tag{11}$$

GAT Variants. The original GAT formulation represents our *Integration (Neighbor)* model. It computes dynamic attention coefficients $\alpha_{i,j}$ over the local neighborhood, including a self-loop, and projects all features using a single shared weight matrix $\mathbf{W}$:

$$\mathbf{x}_i^{(k)} = \sum_{j \in \mathcal{N}(i) \cup \{i\}} \alpha_{i,j} \mathbf{W}\mathbf{x}_j^{(k-1)} \tag{12}$$

For the *Separation (Ego)* variant, we eliminate the self-loop from the attention graph, restricting the attention computation to strictly structural neighbors. The ego node's features are instead projected via a dedicated ego-weight matrix $\mathbf{W}_{\text{ego}}$:

$$\mathbf{x}_i^{(k)} = \mathbf{W}_{\text{ego}}\mathbf{x}_i^{(k-1)} + \sum_{j \in \mathcal{N}(i)} \alpha_{i,j} \mathbf{W}_{\text{neighbor}}\mathbf{x}_j^{(k-1)} \tag{13}$$

GraphSAGE Variants. Unlike GCN and GAT, the original GraphSAGE algorithm naturally decouples the ego node, serving as our *Separation (Ego)* baseline. It transforms the central node via $\mathbf{W}_{\text{ego}}$ and the mean-aggregated neighborhood via $\mathbf{W}_{\text{neighbor}}$:

$$\mathbf{x}_i^{(k)} = \mathbf{W}_{\text{ego}}\mathbf{x}_i^{(k-1)} + \mathbf{W}_{\text{neighbor}} \cdot \text{MEAN}_{j \in \mathcal{N}(i)} \left(\mathbf{x}_j^{(k-1)} \right) \tag{14}$$

To construct the *Integration (Neighbor)* variant, we disable the dedicated ego-weight parameter and explicitly inject self-loops into the graph topology. This forces the ego node to be averaged alongside its neighbors and projected through a single, shared transformation matrix $\mathbf{W}$, enforcing an integration bias analogous to a standard GCN:

$$\mathbf{x}_i^{(k)} = \mathbf{W} \cdot \text{MEAN}_{j \in \mathcal{N}(i) \cup \{i\}} \left(\mathbf{x}_j^{(k-1)} \right) \tag{15}$$

H Real-world Dataset Statistics

This appendix section provides detailed statistics of the real-world benchmark datasets (Qin et al., 2025) used in our experiments. Specifically, we report the number of nodes, edges, classes, feature dimensions, average node degree, label homophily, and feature homophily for each dataset.

I Memorization: Separation vs Integration

In this section, we provide the complete experimental results of the Memorization for the Ego-Neighbor Separation and Ego-Neighbor Integration experiments discussed in Section 5.2.

Table 4: Statistics of benchmark real-world datasets for node classification.

Dataset	Nodes	Edges	Classes	Features	Avg. Degree	Label Homophily	Feature Homophily
Cora	2,708	5,278	7	1,433	3.90	0.81	0.61
CiteSeer	3,327	4,614	6	3,703	2.77	0.74	0.63
PubMed	19,717	44,325	3	500	4.50	0.80	0.98
Computers	13,752	245,861	10	767	35.76	0.78	0.99
Photo	7,487	119,081	8	745	31.13	0.82	0.99
ogbn-arXiv	169,343	1,157,799	40	767	13.67	0.65	0.15
Actor	7,600	26,659	5	932	7.02	0.22	0.06
Chameleon	2,277	36,101	5	2,325	27.60	0.23	-0.04
Squirrel	5,201	217,073	5	2,089	76.33	0.22	-0.06

Table 5: Separation vs Integration GAT on Synthetic Cora dataset

				Train Clean Acc (%)		Test Acc (%)		MR (%)	
				Best Val	Last	Best Val	Last	Best Val	Last
Deg	hF	hL	Model						
1 5	0.00	0.00	Sep.	**100.00 ± 0.00**	**100.00 ± 0.00**	**49.39 ± 1.05**	**49.47 ± 1.09**	**100.00 ± 0.00**	**100.00 ± 0.00**
			Int.	75.97 ± 1.01	99.62 ± 0.08	29.80 ± 0.26	24.28 ± 0.49	59.43 ± 3.79	**100.00 ± 0.00**
		0.50	Sep.	**98.57 ± 1.78**	**100.00 ± 0.00**	55.04 ± 1.11	**54.74 ± 1.21**	**89.43 ± 13.02**	**100.00 ± 0.00**
			Int.	80.76 ± 1.45	99.78 ± 0.16	**55.10 ± 0.32**	42.34 ± 0.32	28.57 ± 3.50	**100.00 ± 0.00**
		1.00	Sep.	**94.67 ± 1.33**	**100.00 ± 0.00**	81.48 ± 1.21	**74.29 ± 1.27**	**28.86 ± 4.28**	**100.00 ± 0.00**
			Int.	88.86 ± 1.21	99.97 ± 0.06	**82.50 ± 0.33**	67.18 ± 0.64	19.43 ± 3.45	99.71 ± 0.57
	0.90	0.00	Sep.	**98.22 ± 3.17**	**100.00 ± 0.00**	**29.16 ± 2.36**	**30.19 ± 1.35**	**95.71 ± 6.64**	**100.00 ± 0.00**
			Int.	76.63 ± 3.28	99.59 ± 0.22	25.93 ± 0.42	22.67 ± 0.61	67.14 ± 5.71	99.71 ± 0.57
		0.50	Sep.	**92.67 ± 6.50**	**100.00 ± 0.00**	49.88 ± 1.67	**49.30 ± 1.04**	**71.14 ± 23.67**	**100.00 ± 0.00**
			Int.	76.19 ± 2.79	99.56 ± 0.23	**50.51 ± 1.34**	41.75 ± 1.12	30.29 ± 1.90	**100.00 ± 0.00**
		1.00	Sep.	**90.86 ± 3.47**	99.87 ± 0.06	**79.58 ± 0.74**	**72.95 ± 0.62**	**25.71 ± 6.19**	**100.00 ± 0.00**
			Int.	88.22 ± 4.33	**99.94 ± 0.08**	77.96 ± 1.71	65.52 ± 2.46	24.57 ± 5.95	99.71 ± 0.57
26 30	0.00	0.00	Sep.	**97.21 ± 0.84**	**100.00 ± 0.00**	**77.09 ± 0.74**	**64.50 ± 2.17**	**38.29 ± 5.53**	**100.00 ± 0.00**
			Int.	42.60 ± 2.91	74.29 ± 3.51	23.19 ± 1.72	21.55 ± 1.13	26.86 ± 4.18	66.29 ± 8.55
		0.50	Sep.	**98.29 ± 0.42**	**100.00 ± 0.00**	**92.09 ± 0.73**	**80.40 ± 1.21**	**8.57 ± 3.73**	**100.00 ± 0.00**
			Int.	81.14 ± 2.92	90.92 ± 1.47	77.32 ± 3.95	63.50 ± 1.51	4.29 ± 2.02	32.29 ± 4.30
		1.00	Sep.	**100.00 ± 0.00**	**100.00 ± 0.00**	**99.75 ± 0.14**	93.29 ± 0.42	**0.29 ± 0.57**	**100.00 ± 0.00**
			Int.	99.71 ± 0.35	99.59 ± 0.19	99.19 ± 0.21	**95.67 ± 1.65**	0.00 ± 0.00	15.71 ± 8.94
	0.90	0.00	Sep.	**95.27 ± 1.80**	**99.17 ± 0.59**	**60.41 ± 4.53**	**59.88 ± 4.88**	**57.14 ± 8.03**	**81.14 ± 13.38**
			Int.	29.59 ± 7.09	33.97 ± 9.89	18.05 ± 1.89	18.08 ± 2.21	29.43 ± 6.30	33.43 ± 8.93
		0.50	Sep.	**96.86 ± 1.88**	**98.83 ± 0.36**	**90.08 ± 1.94**	**89.14 ± 2.78**	10.00 ± 6.32	**22.00 ± 14.10**
			Int.	69.37 ± 6.23	59.78 ± 5.48	62.74 ± 10.98	50.28 ± 2.67	**11.71 ± 9.58**	16.57 ± 6.49
		1.00	Sep.	**99.97 ± 0.06**	**99.97 ± 0.06**	**99.57 ± 0.30**	98.71 ± 0.55	**0.86 ± 1.71**	**17.71 ± 14.27**
			Int.	99.94 ± 0.13	99.94 ± 0.13	99.51 ± 0.21	**99.36 ± 0.29**	0.29 ± 0.57	0.86 ± 1.14

J LLM USAGE

In accordance with recent conference guidelines, we disclose the use of Large Language Models during the preparation of this manuscript strictly for writing assistance and data visualization. Specifically, LLMs were employed to improve the grammar, clarity, and overall readability of the text, as well as to assist in writing boilerplate scripts for formatting and visualizing our experimental results. We affirm that no LLMs were used to synthesize the datasets, conduct the primary experiments, or formulate the core scientific contributions and theoretical claims of this work. The authors have rigorously reviewed, edited, and verified all AI-assisted outputs and assume full responsibility for the accuracy and integrity of the final content presented in this paper.

Table 6: Separation vs Integration GraphSAGE on Synthetic Cora dataset

				Train Clean Acc (%)		Test Acc (%)		MR (%)	
				Best Val	Last	Best Val	Last	Best Val	Last
Deg	hF	hL	Model						
1 5	0.00	0.00	Sep.	**99.33 ± 0.71**	**100.00 ± 0.00**	**59.05 ± 1.05**	**59.04 ± 0.98**	**94.86 ± 4.57**	**100.00 ± 0.00**
			Int.	79.81 ± 1.88	99.84 ± 0.00	30.25 ± 0.28	24.66 ± 0.25	62.86 ± 4.61	**100.00 ± 0.00**
		0.50	Sep.	**98.86 ± 0.87**	**100.00 ± 0.00**	**65.51 ± 0.74**	**63.84 ± 0.75**	**82.86 ± 10.14**	**100.00 ± 0.00**
			Int.	84.22 ± 2.25	**100.00 ± 0.00**	57.50 ± 0.28	47.12 ± 0.08	32.57 ± 4.55	**100.00 ± 0.00**
		1.00	Sep.	**98.70 ± 0.82**	**100.00 ± 0.00**	**85.62 ± 0.35**	**82.32 ± 0.11**	**49.71 ± 14.64**	**100.00 ± 0.00**
			Int.	92.06 ± 0.63	**100.00 ± 0.00**	85.14 ± 0.11	77.22 ± 0.06	20.86 ± 1.14	**100.00 ± 0.00**
	0.90	0.00	Sep.	**97.43 ± 5.06**	**100.00 ± 0.00**	**34.41 ± 1.60**	**34.88 ± 0.93**	**91.43 ± 17.14**	**100.00 ± 0.00**
			Int.	77.43 ± 4.96	99.81 ± 0.06	25.26 ± 0.26	23.37 ± 0.11	67.14 ± 6.70	**100.00 ± 0.00**
		0.50	Sep.	**88.57 ± 1.83**	**100.00 ± 0.00**	**55.45 ± 0.14**	**53.43 ± 0.27**	**43.14 ± 5.14**	**100.00 ± 0.00**
			Int.	76.29 ± 1.74	99.68 ± 0.00	51.72 ± 0.41	44.50 ± 0.16	27.14 ± 1.56	**100.00 ± 0.00**
		1.00	Sep.	**91.46 ± 1.34**	**100.00 ± 0.00**	**81.97 ± 0.36**	**75.74 ± 0.20**	**24.29 ± 3.73**	**100.00 ± 0.00**
			Int.	87.59 ± 2.24	99.97 ± 0.06	81.15 ± 0.34	72.03 ± 0.22	21.71 ± 2.77	**100.00 ± 0.00**
26 30	0.00	0.00	Sep.	**98.60 ± 0.91**	**100.00 ± 0.00**	**75.92 ± 0.49**	**73.88 ± 0.33**	**58.86 ± 15.58**	**100.00 ± 0.00**
			Int.	64.60 ± 3.41	63.68 ± 2.02	31.56 ± 1.01	30.75 ± 1.06	42.00 ± 6.61	41.43 ± 4.14
		0.50	Sep.	**99.62 ± 0.08**	**100.00 ± 0.00**	**91.14 ± 0.46**	85.57 ± 0.36	**25.43 ± 4.90**	**100.00 ± 0.00**
			Int.	91.17 ± 0.33	91.27 ± 0.35	88.36 ± 0.23	**88.35 ± 0.25**	2.57 ± 0.57	2.57 ± 0.57
		1.00	Sep.	**100.00 ± 0.00**	**100.00 ± 0.00**	99.79 ± 0.12	95.57 ± 0.09	**2.00 ± 2.80**	**100.00 ± 0.00**
			Int.	99.87 ± 0.06	99.84 ± 0.00	**99.80 ± 0.01**	**99.79 ± 0.01**	0.00 ± 0.00	0.00 ± 0.00
	0.90	0.00	Sep.	**98.03 ± 0.90**	**99.71 ± 0.12**	**67.62 ± 0.99**	**67.50 ± 0.81**	**68.29 ± 8.45**	**94.00 ± 1.40**
			Int.	26.41 ± 2.21	16.29 ± 2.14	17.45 ± 0.62	14.08 ± 0.21	28.57 ± 4.04	17.14 ± 1.28
		0.50	Sep.	**99.17 ± 0.25**	**99.37 ± 0.17**	**94.03 ± 0.38**	**91.85 ± 0.47**	**7.43 ± 2.29**	**69.43 ± 8.88**
			Int.	80.86 ± 5.64	64.95 ± 13.09	80.20 ± 5.62	64.78 ± 13.54	3.14 ± 1.90	6.00 ± 3.18
		1.00	Sep.	**100.00 ± 0.00**	**99.97 ± 0.06**	**99.79 ± 0.15**	97.77 ± 0.13	**2.86 ± 5.71**	**68.00 ± 6.03**
			Int.	99.84 ± 0.17	99.90 ± 0.08	99.76 ± 0.10	**99.84 ± 0.04**	0.00 ± 0.00	0.00 ± 0.00