LOCALIZING MEMORIZATION IN GRAPH NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph Neural Networks (GNNs) have been shown to memorize individual training examples, with recent work quantifying their memorization. However, *where* memorization occurs within the network, *i.e.,* which layers and weights are responsible, remains an open question. While studies in Deep Neural Networks (DNNs) have established that memorization often concentrates on a small number of model weights across all model layers, no such analysis has been explored for GNNs. This paper presents the first systematic study of localizing memorization in GNNs at both the layer and weight levels. At the layer level, we employ *gradient accounting* to measure per-layer gradient norm contributions from memorized and non-memorized nodes. At the weight level, we analyze memorization of a training node by measuring the minimum number of important weights whose ablation leads to the node's prediction flip. By localizing memorization in multiple GNN architectures trained on diverse datasets of varying homophily levels, we find that (1) memorized nodes dominate the gradient contributions over non-memorized nodes at every layer, with the gap generally increasing with layer depth, (2) the graph homophily determines whether memorized nodes produce gradients aligned with or opposed to those of non-memorized nodes, revealing a GNN-specific memorization mechanism, (3) the memorization of individual nodes in GNNs is concentrated on a few model weights, and (4) finally, the memorization localization analysis in GNNs enables targeted mitigation of privacy leakage. Overall, our work contributes to the fundamental understanding of memorization in GNNs.

1 INTRODUCTION

Graph Neural Networks (GNNs) have demonstrated remarkable performance across a wide range of domains, including social network analysis, recommender systems, fraud detection, and molecular property prediction in drug discovery (Kipf & Welling, 2017; Hamilton et al., 2017; Veličković et al., 2018; Zhou et al., 2020). By operating directly on graph-structured data through a message-passing mechanism (Gori et al., 2005; Scarselli et al., 2008; Gilmer et al., 2017; Bronstein et al., 2021), GNNs learn representations that capture both node features and topological structure, enabling strong generalization across diverse tasks. Despite this empirical success, GNNs are known to memorize individual training examples. Jamadandi et al. (2025) introduce NCMemo, the first framework which utilizes the leave-one-out mechanism (Feldman, 2020) to measure label memorization in semi-supervised node classification, establishing that memorization is inversely correlated with graph homophily. However, NCMemo focuses on *quantifying* memorization at the node level and does not address *where* within the GNN this memorization occurs. The internal components in GNNs, *e.g.,* layers and individual weights, responsible for memorization remain entirely unexplored for GNNs.

This gap is significant given the progress made in understanding memorization localization in other model families. Maini et al. (2023) show that in standard deep neural networks (DNNs) for image classification, memorization arises at a small number of neurons across all model layers, rather than being confined to individual layers. Wang et al. (2024) extend exploration of memorization localization to self-supervised vision *encoders*, proposing LayerMem and UnitMem metrics to localize memorization at the layer and unit level, respectively, finding that highly memorizing units are distributed across the encoder and that atypical data points cause significantly higher memorization than standard data points. Yet, no previous study on memorization localization exists for GNNs,

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.

where the unique data structure and message-passing architecture may lead to fundamentally different memorization patterns compared to general vision or language models.

In order to bridge the gap in understanding memorization localization in GNNs, we present the first systematic study of *localizing memorization in GNNs*, analyzing where memorization occurs at both the layer and weight levels. To yield general insights, we identify memorized nodes through two complementary approaches and then localize them within the model. First, following DNN memorization studies (Feldman, 2020; Feldman & Zhang, 2020; Maini et al., 2023), we flip the labels of a subset of training nodes to create outliers that are known to be prone to memorization. Second, we train normally and score every node with the GNN-specific NCMemo memorization score (Jamadandi et al., 2025), treating nodes above a threshold as memorized. The two approaches are complementary in the sense that label flipping is *controllable* and directly comparable to DNN analyses, while NCMemo *introduces no label* noise and is specifically designed for graphs. To localize memorization at the layer and individual weight level, we employ *layer-wise gradient accounting* during training, and *weight ablation* during inference time, respectively.

Our analysis reveals that GNNs exhibit memorization patterns that differ qualitatively from those observed in DNNs, with graph homophily playing an important role in shaping these patterns. Specifically, we find memorized nodes consistently have larger gradients than non-memorized nodes across all layers, with the gap increasing with layer depth. This suggests that memorization is mainly located in the final layer, which is in line with some observations from DNNs (Baldock et al., 2021; Stephenson et al., 2021). We also observe that the graph homophily determines whether memorized nodes produce gradients aligned with or opposed to those of non-memorized nodes. Finally, we show that localizing *where* memorization occurs can also have practical implications: a model that memorizes particular nodes can leak private information about them, enabling membership inference attacks (MIAs) (Shokri et al., 2017) and data extraction attacks (Wu et al., 2022; Zhang et al., 2021b). In graphs this risk is amplified, since a node's representation depends on its neighbors, so leakage about one node implicates the others around it (Olatunji et al., 2021; Conti et al., 2022; He et al., 2021). We show that our localization of memorization to a specific subset of layers or model weights enables us to prune these components to reduce privacy leakage while leaving the generalizable parts of the model intact, hence maintaining overall performance.

Our main contributions are as follows:

1. We perform the first systematic study to localize memorization in GNNs. Therefore, we rely on both training and inference time localization measures, and analyze different types of memorized nodes and various homophily levels.
2. We find that memorized nodes have larger gradients than non-memorized nodes across all layers, with the gap widening with layer depth, suggesting that memorization is mainly located in the final layer of GNNs.
3. We show empirically and theoretically that the gradients of memorized and non-memorized nodes are aligned in heterophilic graphs but become opposed in homophilic graphs, revealing a GNN-specific mechanism that does not exist in standard DNNs.
4. We find that memorized nodes exhibit concentrated memorization, *i.e.,* memorization of memorized nodes is stored in a small number of weights, and the memorization-critical weights are located primarily in the final layer.
5. As a proof of concept, we use our new insights that memorization is located mainly in the final-layer to design targeted defense against privacy leakage and show that it yields stronger privacy-utility trade-offs than global regularization baselines.

2 Related Work

2.1 Memorization in DNNs

Memorization has been extensively studied in DNNs, with a focus on understanding how models remember specific training data points (Chatterjee, 2018; Zhang et al., 2021a; Arpit et al., 2017; Feldman, 2020; Baldock et al., 2021; Maini et al., 2023; Bombari et al., 2022; Wang et al., 2024; Bayat et al., 2025). Zhang et al. (2021a) first highlighted that DNNs can easily fit random labels,

demonstrating their capacity for memorization. Subsequent work has explored the dynamics of memorization during training (Arpit et al., 2017), the role of model architecture and regularization in controlling memorization (Neyshabur et al., 2017; Sadrtdinov et al., 2021), and the implications of memorization for generalization (Belkin et al., 2019; Feldman & Zhang, 2020). Unlike overfitting where the model shows poor generalization on unseen data, memorization refers to the model's tendency to remember specific training examples, which can occur even when the model generalizes well on unseen data (Feldman, 2020; Tirumala et al., 2022).

2.2 Memorization in GNNs

Bechler-Speicher et al. (2024) studies how the GNNs' inductive bias towards graph structure can lead to poor generalization particularly in graph classification tasks, which is a form of overfitting. Jamadandi et al. (2025) introduce the first framework, NCMemo, to quantify label memorization in GNNs under semi-supervised node classification, and find that nodes in low-homophily graphs exhibit significantly higher memorization, as GNNs resort to label memorization when graph structure is not informative. However, NCMemo focuses on *quantifying* memorization at the node level and does not investigate *where* within a GNN this memorization occurs, which is the question we study.

2.3 Localizing Memorization

Prior work on DNNs has suggested that memorization concentrates in deeper layers, with early layers learning generalizable features and later layers memorizing specific examples (Baldock et al., 2021; Stephenson et al., 2021). In particular, some work explores memorization in DNNs by studying individual units (Erhan et al., 2009; Zeiler & Fergus, 2014; Bau et al., 2017; Morcos et al., 2018; Zhou et al., 2018). By studying memorization on a per-unit granularity, Maini et al. (2023) finds that the memorization of outlier data points is not confined to the final layers but is instead determined by a small number of neurons distributed across the model layers. Wang et al. (2024) extend memorization localization to SSL vision encoders, proposing metrics that operate in a single forward pass without labels. Another line of work localizes memorization in LLMs (Meng et al., 2022; 2023; Dai et al., 2022; Chang et al., 2024), as introduced in Appx. A. Despite this progress across DNNs, SSL encoders, and LLMs, no prior work has investigated memorization localization in GNNs, where the unique data structure and message-passing architecture may lead to fundamentally different memorization patterns compared to these models. Our work fills this gap by providing the first systematic study of localizing memorization in GNNs, analyzing where memorization occurs at both the layer and weight levels, and revealing unique patterns shaped by graph homophily.

3 Problem Setup

Graph Neural Networks and Notation. We define a l-layer GNN f: $\mathbb{R}^{|\mathbf{V}|\times d_i} \to \mathbb{R}^{|\mathbf{V}|\times d_o}$, trained on the graph $\mathbf{G} = (\mathbf{X}, \mathbf{A})$, where $\mathbf{X}$ is the node feature matrix, $\mathbf{A}$ is the adjacency matrix, and $|\mathbf{V}|$, d_i, d_o are the number of nodes, input and output node feature dimensions, respectively. Let f be parametrized by weights $\theta = (\theta_1, \cdots, \theta_l)$. We denote by θ_k, $k \in \{1, 2, \cdots, l\}$ the weight of kth layer in the GNN f. The full notations are presented in Appx. B.

Memorized Nodes. We identify memorized nodes through two complementary approaches: (1) *Controlled label noise*. Prior work on memorization in DNNs usually generates memorized data by flipping the labels of a subset of training data (Feldman, 2020; Feldman & Zhang, 2020; Maini et al., 2023). Following this convention and especially the setup in Maini et al. (2023), we generate memorized nodes for measuring memorization by randomly selecting a subset of training nodes and replacing their labels with uniformly sampled incorrect labels. We refer to these mislabeled nodes as *memorized nodes*, and the remaining nodes with correct labels as *non-memorized nodes*. We denote the proportion of mislabeled nodes as ρ. The whole training dataset $\mathcal{D}$ consists of the set of memorized nodes $\mathcal{D}_m$ and the set of non-memorized nodes $\mathcal{D}_n$, *i.e.,* $\mathcal{D} = \mathcal{D}_m \cup \mathcal{D}_n$. This approach is applied because it is *controllable* (the number of memorized nodes is set by the memorization rate ρ) and *directly comparable* to prior DNN memorization studies using the same protocol. We vary the memorization rate ρ from 1% to 5% to analyze how different proportions of memorized data affect the memorization patterns in GNNs. To further verify that our assumption that mislabeled nodes are a good proxy for memorized nodes is valid, we also run inference on all mislabeled nodes to separate

them into *fitted mislabeled nodes* (model predicts the mislabeled label, confirming memorization) and *non-fitted mislabeled nodes* (model does not predict the mislabeled label). Across all settings, over 85% of mislabeled nodes are fitted (memorized), confirming that the mislabeled nodes are indeed memorized by the model (see Appx. I). (2) *GNN-specific* memorization score, NCMemo (Jamadandi et al., 2025). In this approach, we train without any manipulation to the training nodes and then identify highly memorized nodes based on their respective NCMemo score. Specifically, following Jamadandi et al. (2025), we consider a node as memorized if its NCMemo score is above 0.5.

Experimental Setup. We first perform our experiments on a *synthetic benchmark* proposed in Zhu et al. (2020), where we can control the level of homophily in the graph. It has been shown that homophily plays a critical role in GNNs' training dynamics (Lim et al., 2021). Thus, we evaluate the memorization localization patterns across a range of homophily levels, from 0.0 (completely heterophilic) to 1.0 (completely homophilic), to understand how graph structure influences memorization localization within GNNs. For each dataset, we divide the nodes into 60% training, 20% testing, and 20% validation sets. We leverage different GNN architectures, including GCN (Kipf & Welling, 2017), GAT (Veličković et al., 2018), and GraphSAGE (Hamilton et al., 2017) to ensure the generality of our findings across different GNN designs.

We also validate our findings on *real-world graph datasets* with varying levels of homophily in Appx. H, and on *larger datasets* and *more advanced GNN backbones* in Appx. K. Results are averaged across 5 independent trials. The detailed hyperparameter configurations are provided in Appx. F.

4 Memorization Localization Measures

In this section, we introduce the measures used to localize memorization in GNNs at both the layer and weight levels. We first describe how to localize memorization at layer level using gradient accounting, and then explain how to assess memorization at weight level through weight ablation.

4.1 Memorization by individual layers

In order to localize memorization at the layer level, we employ *gradient accounting* (Katharopoulos & Fleuret, 2018; Maini et al., 2023; Paul et al., 2021) to GNNs. During training, let $f_\theta^t = (\theta_1^t, \cdots, \theta_l^t)$ denote the model weights at the epoch t. The parameter update to layer k using gradient descent with a learning rate η is given by:

$$\theta_k^{t+1} = \theta_k^t - \eta \nabla_{\theta_k} \mathcal{L}(f_{\theta_k}^t, \mathcal{D}) \tag{1}$$

where the gradient $\nabla_{\theta_k} \mathcal{L}(f_{\theta_k}^t, \mathcal{D})$ determines the magnitude and direction of the weight update at layer k, and thus can serve as an indicator of that layer's memorization potential.

In order to assess the contribution of gradients from memorized and non-memorized nodes respectively, we measure the gradient norms $\|\nabla_{\theta_k} \mathcal{L}(f, \mathcal{D}_m)\|_2$ and $\|\nabla_{\theta_k} \mathcal{L}(f, \mathcal{D}_n)\|_2$ on memorized and non-memorized nodes separately per epoch. To enable fair comparison across layers with different numbers of parameters, each gradient norm is normalized by $\sqrt{|\theta_k|}$, the square root of the number of parameters in layer k. The normalized gradient norms are then averaged across all training epochs to obtain a single per-layer memorization score for each data subset. We define the layer-wise memorization score $\texttt{LMS}_k$ for memorized and non-memorized nodes at layer k as follows:

$$\texttt{LMS}_k(\mathcal{D}_m) = \frac{1}{T} \sum_{t=1}^{T} \frac{\left\|\nabla_{\theta_k} \mathcal{L}(f_{\theta_k}^t, \mathcal{D}_m)\right\|_2}{\sqrt{|\theta_k|}}, \quad \texttt{LMS}_k(\mathcal{D}_n) = \frac{1}{T} \sum_{t=1}^{T} \frac{\left\|\nabla_{\theta_k} \mathcal{L}(f_{\theta_k}^t, \mathcal{D}_n)\right\|_2}{\sqrt{|\theta_k|}} \tag{2}$$

where T is the total number of training epochs.

4.2 Memorization by individual weights

LMS reveals how much gradient effort the model expends on memorizing training nodes at each layer during training. A complementary and even more fundamental question concerns the *distribution* of the memorization inside the model after training, *i.e., is memorization in GNNs concentrated in a small set of weights, or distributed broadly across the model?* To answer this question, we introduce the Ablated Weight Score (AWS), which applies a greedy ablation procedure (*i.e.,* a procedure that iteratively zeroes out weights) to quantify the number of important weights involved in flipping the

prediction of a given node. The intuition is that the model is extremely certain about the predicted label of nodes that it memorizes. Hence, if we ablate, *i.e.,* remove model weights that are responsible for this prediction until a label flip occurs, we identify (and remove) the weights that were responsible for memorization. More importantly, we investigate the layer-wise distribution of these important weights to characterize the localization of memorization within the network architecture.

To obtain the AWS of a target node, we first measure the importance for each weight and then apply a greedy ablation procedure that iteratively zeroes out the single most important weight across the entire model. We do so until the prediction for the target node flips. At this point, we count the total number of ablated weights, which we normalize by the total number of model weights as the final score for that node. In Appx. M, we analyze the computational efficiency of AWS and also implement two variants to calculate AWS with scalable approximation, *i.e.,* ablating the top-k most important weights simultaneously at each step, and batch multiple nodes together and share a single forward pass to compute gradients jointly.

Measuring importance. At each ablation step, we rank all non-zeroed weights by an importance score based on a first-order Taylor approximation of the change in loss induced by zeroing w_{ij} (Molchanov et al., 2017; LeCun et al., 1989):

$$s(w_{ij}) = \left| \frac{\partial \mathcal{L}_v}{\partial w_{ij}} \cdot w_{ij} \right|, \tag{3}$$

where w_{ij} is the weight connecting neuron i to j, $s(w_{ij})$ is the importance score of weight w_{ij}, and $\mathcal{L}_v$ is the cross-entropy loss evaluated on node v's output. This score jointly captures the sensitivity of the loss to w_{ij} via the gradient and the current magnitude of w_{ij} via the weight itself, providing a stable signal that is less susceptible to vanishing gradients than gradient-only attribution. Weights that have already been zeroed are excluded from selection by assigning them an importance score of $-\infty$.

Greedy Ablation. The greedy ablation procedure is summarized in Algorithm 1 (Appx. C). Given a trained GNN f_θ and a target node v, we enter an iterative loop in which, at each step, we compute the cross-entropy loss $\mathcal{L}_v$ on node v and backpropagate to obtain gradients with respect to all weights. For each non-zeroed weight w_{ij}, we compute an importance score from Eq. (3). The single weight with the highest importance score across all layers is then permanently zeroed, and the ablation counter is incremented. This process repeats until the model's prediction on v changes, at which point all zeroed weights are restored to their original values[1] and $\texttt{AWS}(v)$, the percentage of model weights ablated required to induce the flip, is returned alongside the per-layer ablation counts which supports investigation of the layer-wise distribution of the important weights. Crucially, the selection is *global*: at each step, the most important weight is chosen from *the entire model*, regardless of which layer it belongs to. This means the per-layer distribution of ablated weights is not imposed by design but emerges naturally from the model's own weight importance structure, making it a faithful reflection of where memorization-critical weights actually concentrate.

Ablated Weight Score. We define the AWS for node v as the percentage of model weights whose ablation is required to flip the model's prediction on v:

$$\texttt{AWS}(v) = \frac{c}{|\theta|} \times 100\%, \tag{4}$$

where c is the number of ablated weights required for prediction flip and $|\theta| = \sum_{k=1}^{l} |\theta_k|$ is the total number of weights in the model. A *lower* AWS indicates that the prediction is supported by fewer, more concentrated memorization-critical weights, suggesting a sign of highly localized memorization. A *higher* AWS indicates that the prediction is supported by a broader, more distributed set of weights, suggesting a more distributed memorization pattern. Crucially, AWS and the layer-wise memorization score (LMS) from Sec. 4.1 measure fundamentally different aspects of memorization. LMS operates at the *layer* level during training, identifying which layers are more engaged in fitting memorized nodes. AWS is finer-grained and operates at the *weight* level after training, identifying which individual weights a memorized node depends on for its prediction. Therefore, AWS additionally reveals how *concentrated* the resulting memorization is. As we will show, these two dimensions can diverge, *e.g.,* a node can have a high layer-wise memorization score during training, yet its prediction can only

[1] It indicates that the calculation of AWS is a post-training process and it doesn't change the model's weights.

be flipped by ablating a large number of weights spread across the model , and their joint analysis provides a more complete picture of *how and where* GNNs memorize training nodes.

5 EVALUATION

We begin our evaluations by using a synthetic dataset benchmark, which is the `syn-cora` dataset proposed in (Zhu et al., 2020). It contains graphs with different homophily levels and the node features for the graphs are assigned from a real-world graph such as Cora (Yang et al., 2016). We choose five graphs with increasing homophily levels $h = \{0.0, 0.3, 0.5, 0.7, 1.0\}$, where, for example, `syn-cora-h0.0` indicates a synthetic graph with homophily level 0.0. Note that low-homophily graphs are also called *heterophilic* graphs, where the connected nodes have different labels/features.

5.1 LAYER-WISE MEMORIZATION SCORE

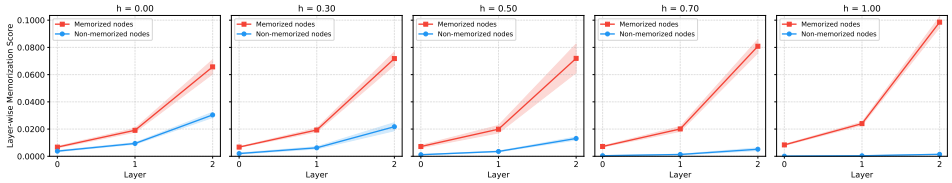


Figure 1: **Layer-wise memorization scores of memorized (mislabeled) and non-memorized (clean) nodes across homophily levels and model layers for a 3-layer GCN trained on `syn-cora`,** $\rho = 0.01$**.** Overall, memorized nodes dominate gradient contributions across all layers.

We first evaluate the `LMS` under controlled label noise by introducing a certain fraction of mislabeled nodes into the training data. We train a three-layer GNN model, *e.g.,* GCN, on each *syn-cora* graph and compute layer-wise memorization scores for memorized (mislabeled) and non-memorized (clean) nodes using Eq. (2). We plot the memorization scores for memorized and non-memorized nodes across different layers and homophily levels, as shown in Fig. 1. Results for other GNN models are provided in Appx. G.1. We observe that the layer-wise memorization of memorized nodes exceeds that of non-memorized nodes at every layer and across all homophily levels, with the gap widening with layer depth. This stands in consistent with the findings of Baldock et al. (2021); Stephenson et al. (2021) for standard DNNs, where memorization concentrates in deeper layers of the network. We also find that memorized and non-memorized nodes exhibit opposite layer-wise memorization trends with respect to homophily, *i.e.,* the average gradient norm of non-memorized nodes decreases as homophily increases while that of memorized nodes exhibits a general increase (see Appx. G.2).

The persistently higher `LMS` for memorized nodes can be attributed to the fact that, in this setup, the memorized nodes are assigned labels that contradict both their node features and the class-consistent signal aggregated from their neighborhood by message passing. Therefore, fitting these wrong labels requires overriding a well-learned representation, producing persistently high cross-entropy and thus large individual gradient contributions throughout training. Importantly, in Appx. J, we show that the effect of obtaining a higher `LMS` for memorized nodes cannot be explained through sample difficulty. We show that difficult non-memorized nodes, matched by their loss or prediction margin to equally difficult memorized nodes, still yield a substantially smaller `LMS`.

The results for memorized and non-memorized nodes based on NCMemo score are presented in Appx. G.1. We observe an equivalent pattern to that seen with the controlled label noise approach, *i.e.,* memorized nodes exhibit higher layer-wise memorization scores than non-memorized nodes, and the gap widens with increasing layer depth. This observation leads to our first novel insight:

Finding 1 (Memorized nodes dominate gradient contributions across all layers.) *At each layer of a GNN, the memorized nodes exhibit a larger `LMS` score than non-memorized nodes, and the gap between them increases with layer depth, indicating that in GNNs, memorization is distributed across all layers but becomes increasingly dominant in deeper layers.*

We further calculate and analyze the cosine similarity between the average gradients of memorized (mislabeled) and non-memorized nodes (clean) per layer, per epoch, across homophily levels. We

plot the cosine similarity in Fig. 2, which reveals an obvious relationship between graph homophily and the alignment of memorized and non-memorized gradients. At low homophily ($h = 0.00$), the cosine similarity is consistently positive across all layers and epochs, indicating that memorized and non-memorized nodes push the model weights in similar directions throughout training. As homophily increases, the cosine similarity decreases progressively, becoming strongly negative at $h = 1.00$, where memorized and non-memorized gradients actively oppose each other across all layers. We formalize this observation as a proposition below:

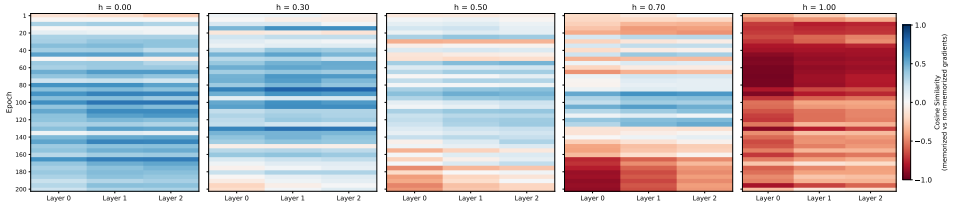


Figure 2: **Cosine similarity between the average gradients of memorized (mislabeled) and non-memorized nodes (clean) per layer, per epoch, across homophily levels, for a 3-layer GCN trained on `syn-cora`,** $\rho = 0.01$**.** Blue indicates positive similarity; red indicates negative similarity.

Proposition 1 (Inverse relationship between gradients cosine similarity and homophily.) *The cosine similarity between the gradients of memorized and non-memorized nodes decreases with graph homophily, transitioning from consistently positive on heterophilic graphs to strongly negative on homophilic graphs.*

The key intuition is the direction in which each node type pulls the model. In a homophilic graph, neighbouring aggregation produces a strong class-consistent signal, so the model is likely to predict the true class for a memorized (mislabeled) node from its neighbouring context. Non-memorized (clean) nodes reinforce this signal, while memorized nodes need to push against it to fit their wrong label. Crucially, the stronger the neighbouring node consensus, the more forcefully the memorized gradient must oppose it. Therefore, as homophily increases, the model's true-class prediction for the memorized node grows stronger, requiring an increasingly large opposing update (gradient) to fit the wrong label. In a heterophilic graph, neighbouring aggregation yields a class-uninformative signal, leaving both memorized and non-memorized nodes to rely on individual node features, resulting in compatible gradient directions and positive cosine similarity. A formal proof under a linearized GCN is provided in Appx. D.

5.2 Ablated weight score

Fig. 3 shows the distribution of `AWS` for memorized (mislabeled) and non-memorized (clean) nodes, based on the controlled label noise approach with a memorization ratio $\rho = 0.01$ across all homophily levels. Results on other GNN models are provided in Appx. G.4. We can observe that the `AWS` of memorized nodes is significantly lower than that of non-memorized nodes across all homophily levels. The distribution of `AWS` for memorized and non-memorized nodes based on NCMemo score is presented in Appx. G.4 (Fig. 11), and we can observe equivalent trends to those seen under the controlled label noise setup. This finding is consistent with Maini et al. (2023) and Frankle & Carbin (2019), suggesting that memorization in GNNs, like in DNNs, is concentrated in a small number of weights, supporting Finding 2. Table 1 confirms this quantitatively: across all memorization rates and homophily levels, the average `AWS` of memorized nodes is consistently and substantially lower than that of non-memorized nodes, with non-memorized nodes requiring on average 2–3 times more model weight ablations (as a percentage of total model weights) than memorized nodes. We also find a positive relationship between `AWS` and homophily (see Appx. G.6).

Finding 2 (Memorized nodes exhibit more concentrated memorization.) *In GNNs, memorized nodes exhibit significantly lower* `AWS` *than non-memorized nodes across all homophily levels, indicating that memorization in GNNs is concentrated in a small percentage of model weights, consistent with the findings of Maini et al. (2023) in DNNs.*

We also investigate the layer-wise distribution of ablated weights to further localize the memorization within the model architecture, as shown in Fig. 12 in Appx. G.5. We find that, for both memorized

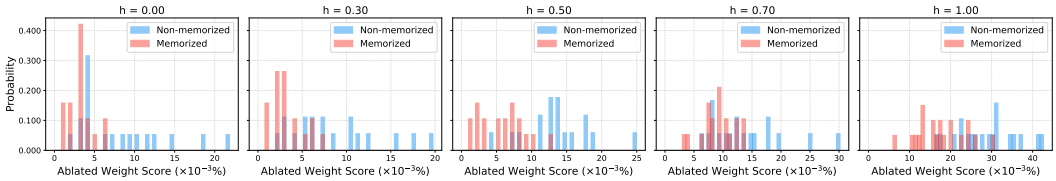


Figure 3: **Distribution of `AWS`(v) ($\times 10^{-3}\%$ of total model weights ablated) for memorized (mislabeled) and non-memorized (clean) nodes (GCN, `syn-cora`, $\rho = 0.01$).**

Table 1: **Average `AWS` ($\times 10^{-3}\%$ of total model weights, mean $\pm$ std) for memorized (mislabeled) and non-memorized (clean) nodes across memorization rates and homophily levels (GCN, `syn-cora`).** Lower `AWS` indicates more concentrated memorization.

Memorization Rate	Node Type	Homophily Level				
		h=0.00	h=0.30	h=0.50	h=0.70	h=1.00
0.01	Non-memorized	8.10 ± 5.60	8.67 ± 5.12	13.52 ± 4.77	13.81 ± 6.38	28.85 ± 7.36
	Memorized	3.18 ± 1.53	3.29 ± 1.84	5.69 ± 3.33	9.14 ± 2.89	17.48 ± 5.85
0.02	Non-memorized	9.64 ± 5.57	8.80 ± 4.31	14.76 ± 7.45	21.02 ± 11.70	27.75 ± 9.27
	Memorized	4.09 ± 2.54	3.45 ± 2.25	5.28 ± 2.54	8.70 ± 4.47	15.34 ± 7.35
0.03	Non-memorize	7.38 ± 4.33	9.38 ± 5.45	16.63 ± 7.04	20.59 ± 8.98	26.13 ± 6.83
	Memorized	3.31 ± 1.78	4.06 ± 2.16	5.78 ± 3.35	7.86 ± 4.80	15.01 ± 5.73
0.04	Non-memorize	7.89 ± 3.66	10.13 ± 5.01	11.88 ± 5.50	17.12 ± 6.09	23.48 ± 8.49
	Memorized	3.34 ± 1.72	3.92 ± 2.52	4.48 ± 2.27	7.16 ± 3.72	13.99 ± 6.26
0.05	Non-memorize	7.23 ± 4.02	11.01 ± 5.72	12.05 ± 6.97	14.83 ± 8.31	23.29 ± 8.76
	Memorized	4.11 ± 2.20	3.90 ± 2.45	5.39 ± 3.08	6.91 ± 3.57	13.39 ± 6.53

and non-memorized nodes, most ablated weights generally fall in the final layer across all homophily levels, supporting Proposition 2. This observation is complementary rather than contradictory with the gradient accounting results of Sec. 5.1, where gradients of memorized nodes dominate across *all* layers, increasing with depth. The key distinction is that *gradient accounting measures which layers are most heavily updated to fit memorized nodes during training, while weight ablation measures which weights the trained model's predictions depend on after training.* These are two orthogonal aspects of memorization. We provide a theoretical analysis to support Proposition 2 in Appx. E. Similar to `LMS`, we also distinguish `AWS` from effects of per-example loss, prediction margin, and sample difficulty (see Appx. J).

Proposition 2 (The final layer is the primary site of memorization in GNNs.) *Memorization-critical weights are concentrated in the final layer for both memorized and non-memorized nodes, indicating that the final layer is the primary site of memorization in GNNs.*

Together, our results on two complementary approaches, *i.e.,* controlled label noise and NCMemo score, confirm that our `LMS` and `AWS` capture genuine memorization in GNNs. In Appx. L, we show that our findings also hold across other graph tasks, including graph classification (Appx. L.1) and link prediction (Appx. L.2). These results validate the generality and scalability of our findings.

6 Impact of Localized Memorization on Privacy Leakage

As memorization in GNNs is particularly relevant to privacy leakage in GNNs, where the memorization of training data can lead to privacy risks, measurable, for example, through membership inference attacks. Our analysis in Sec. 5 reveals that memorization in GNNs is highly concentrated in a small number of weights in the final layer. A natural question is whether this localization can be leveraged to mitigate privacy risks. If we know where memorization is concentrated, we might be able to remove it in a targeted way while preserving the generalization performance of the model. In this section, we analyze this question with a proof-of-concept defense, a *localization-guided pruning* that uses the `AWS` localization signal to selectively remove memorization-critical weights, and evaluate its effectiveness in reducing membership inference risk while preserving model utility.

Threat Model. We consider an adversary with access to the outputs of the trained GNN who conducts the state-of-the-art LiRA membership inference attack (Carlini et al., 2022). We measure

attack effectiveness via MIA AUC (Area Under Curve) and TPR (True Positive Rate) at 1% FPR (False Positive Rate), with higher values indicating greater privacy leakage. The goal of the defense is to reduce membership inference risk while preserving test accuracy.

Experimental Setup. We compare the performance of our *localization-guided pruning* against four baseline defenses against MIAs, including (1) *Weight decay*, a standard ℓ_2 weight decay with coefficient 5×10^{-4} applied to all parameters during training, (2) *Rewiring* (Jamadandi et al., 2025), which modifies the graph structure to reduce memorization, and two masking-based defenses: (3) *Uniform dropout*, which randomly zeros out weights across the entire model, and (4) *Final-layer dropout*, which randomly zeros out weights only in the final layer as memorization concentrates in the final layer. For our localization-guided pruning, and uniform, final-layer dropout defenses, we zero out the same number of weights, which is γ of the final-layer model weights. For rewiring defense, we apply a budget of 500 edge modifications, following the setup in Jamadandi et al. (2025).

Defense Results. Table 2 reports the performance of each defense, including train accuracy, test accuracy, AWS of memorized (mislabeled) nodes, MIA AUC, and TPR@1%FPR, on Cora with GCN at $\rho = 0.01$. Results are averaged across five independent seeds. Across all metrics, our localization-guided pruning consistently outperforms the other defenses. It achieves the lowest MIA AUC (0.51) and TPR@1%FPR (0.03), representing a 0.15 reduction in MIA AUC and a $\sim 4\times$ reduction in TPR@1%FPR compared with the model without defense. Simultaneously, it maintains the highest train and test accuracy among all defenses. The highest AWS on memorized nodes confirms that localization-guided pruning mitigates memorization effectively. Our preliminary results suggest that targeting the most memorization-critical weights in the final layer can be an effective strategy for mitigating memorization and protecting user privacy in GNNs, while preserving generalizable representations more effectively than untargeted defenses. We further visualize the privacy-utility trade-off of three weight-masking defenses, *i.e.,* Uniform dropout, Final-layer dropout, and our Localization-guided pruning, across $\gamma \in \{1\%, 5\%, 10\%, 20\%, 30\%\}$ of final-layer weights, in Fig. 4. The ideal operating point is the bottom-right corner, which indicates high test accuracy and low MIA risk. Fig. 4 shows that our localization-guided pruning generally achieves a better privacy-utility trade-off compared to the other defenses across all γ values.

Table 2: **Comparison of defenses (GCN, Cora, $\rho = 0.01$, $\gamma = 10\%$).** ↑ indicates higher is better; ↓ indicates lower is better. AWS values are reported in units of $10^{-3}\%$ of total model weights. Our localization-guided pruning consistently outperforms other defenses.

Defense	Train Acc ↑	Test Acc ↑	AWS (Mislabeled) ↑	MIA AUC ↓	TPR@1%FPR ↓
None	0.89 ± 0.05	0.86 ± 0.01	4.97 ± 0.82	0.66 ± 0.01	0.11 ± 0.01
Weight Decay	0.70 ± 0.05	0.68 ± 0.01	6.84 ± 5.15	0.57 ± 0.01	0.09 ± 0.00
Rewiring	0.67 ± 0.08	0.66 ± 0.07	10.73 ± 5.06	0.55 ± 0.02	0.08 ± 0.01
Uniform Dropout	0.78 ± 0.02	0.76 ± 0.02	8.83 ± 0.60	0.57 ± 0.01	0.09 ± 0.01
Final-layer Dropout	0.78 ± 0.00	0.76 ± 0.01	13.41 ± 9.05	0.54 ± 0.01	0.09 ± 0.01
Localization-guided Pruning	$\mathbf{0.83 \pm 0.00}$	$\mathbf{0.81 \pm 0.05}$	$\mathbf{17.88 \pm 1.81}$	$\mathbf{0.51 \pm 0.00}$	$\mathbf{0.03 \pm 0.01}$

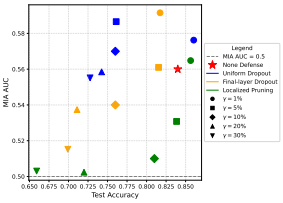


Figure 4: **Privacy-utility for three weight-masking defenses across various γ.**

7 Conclusion

In this work, we present the first systematic study of localizing memorization in GNNs at both the layer and weight levels. Identifying memorized nodes through two complementary approaches, namely controlled label noise and a label-free, GNN-specific memorization score (NCMemo), we show that memorized nodes dominate gradient contributions at every layer, with the gap over non-memorized nodes increasing with depth, in contrast to DNNs. We further show, empirically and theoretically, that the gradients of memorized and non-memorized nodes transition from aligned on heterophilic graphs to strongly opposed on homophilic graphs, revealing a GNN-specific mechanism. At the weight level, memorized nodes exhibit more concentrated memorization than non-memorized nodes, with memorization-critical weights being located mainly in the final layer. As a proof of concept, we leverage this final-layer concentration to design a targeted pruning defense that effectively reduces membership inference risk while preserving accuracy better than other untargeted baselines. Overall, our work establishes memorization localization as a principled lens for understanding and improving the trustworthiness of GNNs.

REFERENCES

Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. In *International conference on machine learning*, pp. 233–242. PMLR, 2017.

Robert Baldock, Hartmut Maennel, and Behnam Neyshabur. Deep learning through the lens of example difficulty. *Advances in Neural Information Processing Systems*, 34:10876–10889, 2021.

David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *2017 IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 3319–3327. IEEE, 2017.

Reza Bayat, Mohammad Pezeshki, Elvis Dohmatob, David Lopez-Paz, and Pascal Vincent. The pitfalls of memorization: When memorization hurts generalization. In *International Conference on Learning Representations*, volume 2025, pp. 86017–86043, 2025.

Maya Bechler-Speicher, Ido Amos, Ran Gilad-Bachrach, and Amir Globerson. Graph neural networks use graphs when they shouldn't. In *Proceedings of the 41st International Conference on Machine Learning*, Proceedings of Machine Learning Research, pp. 3284–3304. PMLR, 2024.

Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

Simone Bombari, Mohammad Hossein Amani, and Marco Mondelli. Memorization and optimization in deep neural networks with minimum over-parameterization. *Advances in Neural Information Processing Systems*, 35:7628–7640, 2022.

Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. Membership inference attacks from first principles. In *2022 IEEE symposium on security and privacy (SP)*, pp. 1897–1914. IEEE, 2022.

Ting-Yun Chang, Jesse Thomason, and Robin Jia. Do localization methods actually localize memorized data in llms? a tale of two benchmarks. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3190–3211, 2024.

Satrajit Chatterjee. Learning and memorization. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 755–763. PMLR, 10–15 Jul 2018.

Mauro Conti, Jiaxin Li, Stjepan Picek, and Jing Xu. Label-only membership inference attack against node-level graph neural networks. In *Proceedings of the 15th ACM Workshop on Artificial Intelligence and Security*, pp. 1–12, 2022.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8493–8502, 2022.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1, 2009.

Vitaly Feldman. Does learning require memorization? a short tale about a long tail. In *Proceedings of the 52nd annual ACM SIGACT symposium on theory of computing*, pp. 954–959, 2020.

Vitaly Feldman and Chiyuan Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation. *Advances in neural information processing systems*, 33:2881–2891, 2020.

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

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. PMLR, 2017.

Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE international joint conference on neural networks, 2005.*, volume 2, pp. 729–734. IEEE, 2005.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

Xinlei He, Rui Wen, Yixin Wu, Michael Backes, Yun Shen, and Yang Zhang. Node-level membership inference attacks against graph neural networks. *arXiv preprint arXiv:2102.05429*, 2021.

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.

Angelos Katharopoulos and François Fleuret. Not all samples are created equal: Deep learning with importance sampling. In *International conference on machine learning*, pp. 2525–2534. PMLR, 2018.

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

Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

Derek Lim, Felix Hohne, Xiuyu Li, Sijia Linda Huang, Vaishnavi Gupta, Omkar Bhalerao, and Ser Nam Lim. Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. *Advances in neural information processing systems*, 34:20887–20902, 2021.

Pratyush Maini, Michael C. Mozer, Hanie Sedghi, Zachary C. Lipton, J. Zico Kolter, and Chiyuan Zhang. Can neural network memorization be localized? In *Proceedings of the 40th International Conference on Machine Learning*, pp. 23536–23557. PMLR, 2023.

Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163, 2000.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372, 2022.

Kevin Meng, Arnab Sen Sharma, Alex J Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations*, 2023.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. In *International Conference on Learning Representations*, 2017.

Ari S Morcos, David GT Barrett, Neil C Rabinowitz, and Matthew Botvinick. On the importance of single directions for generalization. In *International Conference on Learning Representations*, 2018.

Christopher Morris, {Nils Morten} Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*, 2020.

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nathan Srebro. Exploring generalization in deep learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 5949–5958. Curran Associates Inc., 2017.

Iyiola E Olatunji, Wolfgang Nejdl, and Megha Khosla. Membership inference attack on graph neural networks. In *2021 Third IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, pp. 11–20. IEEE, 2021.

Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data diet: Finding important examples early in training. *Advances in neural information processing systems*, 34: 20596–20607, 2021.

Hongbin Pei, Bingzhe Wei, Kevin Chen Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *8th International Conference on Learning Representations, ICLR 2020*, 2020.

Ildus Sadrtdinov, Nadezhda Chirkova, and Ekaterina Lobacheva. On the memorization properties of contrastive learning. *arXiv preprint arXiv:2107.10143*, 2021.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

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

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pp. 3–18. IEEE, 2017.

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 *International Conference on Learning Representations*, 2021.

Kushal Tirumala, Aram Markosyan, Luke Zettlemoyer, and Armen Aghajanyan. Memorization without overfitting: Analyzing the training dynamics of large language models. *Advances in Neural Information Processing Systems*, 35:38274–38290, 2022.

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

Kuansan Wang, Zhihong Shen, Chiyuan Huang, Chieh-Han Wu, Yuxiao Dong, and Anshul Kanakia. Microsoft academic graph: When experts are not enough. *Quantitative Science Studies*, 1(1): 396–413, 2020.

Wenhao Wang, Adam Dziedzic, Michael Backes, and Franziska Boenisch. Localizing memorization in ssl vision encoders. *Advances in Neural Information Processing Systems*, 37:60475–60516, 2024.

Fan Wu, Yunhui Long, Ce Zhang, and Bo Li. Linkteller: Recovering private edges from graph neural networks via influence analysis. In *2022 ieee symposium on security and privacy (sp)*, pp. 2005–2024. IEEE, 2022.

Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pp. 40–48. PMLR, 2016.

Shi Yunsheng, Huang Zhengjie, Feng Shikun, Zhong Hui, Wang Wenjing, and Sun Yu. Masked label prediction: Unified message passing model for semi-supervised classification. *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pp. 1548–1554, 08 2021.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pp. 818–833. Springer, 2014.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021a.

Zaixi Zhang, Qi Liu, Zhenya Huang, Hao Wang, Chengqiang Lu, Chuanren Liu, and Enhong Chen. Graphmi: Extracting private graph data from graph neural networks. *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pp. 3749–3755, 08 2021b.

Bolei Zhou, Yiyou Sun, David Bau, and Antonio Torralba. Revisiting the importance of individual units in cnns via ablation. *arXiv preprint arXiv:1806.02891*, 2018.

Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.

Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in neural information processing systems*, 33:7793–7804, 2020.

AI USE STATEMENT

In this work, we used generative AI tools for polishing writing and research execution. We have not used generative AI tools for retrieval or discovery, drafting sections of the paper, or providing mathematical claims, and generating synthetic datasets is not applicable to this work. We have reviewed all AI-assisted work. For example, LLM-generated code was verified and tested for correctness by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

A ADDITIONAL RELATED WORK ON LOCALIZING MEMORIZATION

A significant line of work localizes where semantic knowledge is stored within large language transformers. Meng et al. (2022) identify that factual associations are mediated by middle-layer MLP modules, and demonstrate that targeted rank-one weight edits (ROME) can precisely update stored facts. Meng et al. (2023) extend this to mass editing across multiple transformer layers (MEMIT), enabling simultaneous updates to thousands of associations. Dai et al. (2022) identify individual *knowledge neurons* in pretrained transformers via gradient-based attribution, showing that suppressing these neurons degrades the expression of corresponding facts. Chang et al. (2024) provides the first systematic benchmarks for evaluating localization methods in LLMs, proposing the INJ Benchmark, which injects new information into a controlled subset of weights to create ground-truth localization targets, and the DEL Benchmark, which measures how much dropping identified neurons suppresses a memorized sequence.

B NOTATION

We present the notations used throughout the work in Table 3.

Table 3: **Notation used throughout the work.**

Notations	Descriptions
$\mathbf{G} = (\mathbf{X}, \mathbf{A})$	graph
$\mathbf{X}$	node feature matrix
$\mathbf{A}$	adjacency matrix
d_i, d_o	input and output node feature dimensions
f	GNN model
l	number of layers
θ	weights of model
$\theta_k, k \in \{1, 2, \cdots, l\}$	weights of kth layer of the model
$\mathcal{L}(\cdot)$	cross-entropy loss
$\nabla_{\theta_k}\mathcal{L}(\cdot)$	gradient on weight θ_k
t	epoch
T	total number of training epochs
θ_k^t	model weights at layer k at the epoch t
$\mathcal{D}_m$	memorized nodes
$\mathcal{D}_n$	non-memorized nodes
$\mathcal{D}$	the whole training set
ρ	memorization rate (proportion of $\mathcal{D}_m$ in $\mathcal{D}$)
η	learning rate
$\texttt{LMS}_k(\mathcal{D})$	layer-wise memorization score for nodes $\mathcal{D}$ at layer k
h	homophily level
w_{ij}	weight connecting neuron i to neuron j
$s(w_{ij})$	importance score of weight w_{ij}
$\texttt{AWS}(v)$	ablated weight score of node v
γ	percentage of zeroed-out weights within the last layer

C GREEDY WEIGHT ABLATION

The greedy weight ablation procedure for calculating the Ablated Weight Score (AWS) of a given node v is as shown in Algorithm 1:

Algorithm 1 Greedy Weight Ablation for Ablated Weight Score

Input: GNN f_θ, node v, graph $\mathbf{G}$
Output: ablated weight score $\texttt{AWS}(v)$, per-layer ablation counts $\{n_k\}_{k=1}^{l}$

$\hat{y}_v^{(0)} \leftarrow \arg\max f_\theta(\mathbf{G})_v$ ▷ Record original prediction
$c \leftarrow 0; \quad n_k \leftarrow 0, \forall k; \quad \mathcal{Z} \leftarrow \emptyset$ ▷ Initialise counters and zeroed-weight set
repeat
  $\hat{y}_v \leftarrow \arg\max f_\theta(\mathbf{G})_v$
  if $\hat{y}_v \neq \hat{y}_v^{(0)}$ **then**
    break ▷ Prediction has flipped
  end if
  $\mathcal{L}_v = \texttt{CrossEntropy}(f_\theta(\mathbf{G})_v, \hat{y}_v^{(0)})$ ▷ Compute cross-entropy loss on node v
  Backpropagate to obtain $\nabla_\theta \mathcal{L}_v$
  $s(w_{ij}) = |\frac{\partial \mathcal{L}_v}{\partial w_{ij}} \cdot w_{ij}|, \forall w_{ij} \notin \mathcal{Z}$ ▷ Compute importance scores
  $w_{i^*j^*}^{(k^*)} \leftarrow \arg\max_{w_{ij} \notin \mathcal{Z}} s(w_{ij})$ ▷ Select most important weight globally
  $w_{i^*j^*}^{(k^*)} \leftarrow 0; \quad \mathcal{Z} \leftarrow \mathcal{Z} \cup \{w_{i^*j^*}^{(k^*)}\}$ ▷ Zero out selected weight and add to zeroed-weight set
  $c \leftarrow c + 1; \quad n_{k^*} \leftarrow n_{k^*} + 1$ ▷ Increment flip count and layer-wise ablation count
until prediction flips
return $\texttt{AWS}(v) \leftarrow (c/|\theta|) \times 100, \ \{n_k\}_{k=1}^{l}$

D PROOF FOR PROPOSITION 1

In this section, we provide a simple proof to support our empirical results in Proposition 1 under a simplified linear GCN model. We here take the controlled label noise setup as the example, where the memorized and non-memorized nodes are the mislabeled and clean nodes, respectively.

D.1 SETUP AND ASSUMPTIONS

Consider a 2-layer linear GCN:

$$\mathbf{H} = \tilde{\mathbf{A}}\mathbf{X}\mathbf{W}_1, \qquad \mathbf{Z} = \mathbf{H}\mathbf{W}_2, \qquad \hat{\mathbf{Y}} = \texttt{softmax}(\mathbf{Z}), \tag{5}$$

where $\tilde{\mathbf{A}} = \mathbf{D}^{-1/2}(\mathbf{A} + \mathbf{I})\mathbf{D}^{-1/2}$ is the symmetrically normalised adjacency matrix with self-loops, $\mathbf{X} \in \mathbb{R}^{n \times d_0}$ is the node feature matrix, $\mathbf{W}_1 \in \mathbb{R}^{d_0 \times d}$ and $\mathbf{W}_2 \in \mathbb{R}^{d \times C}$ are learnable weight matrices, C is the number of classes. For node v, let $h_v \in \mathbb{R}^d$ denote its hidden representation (the v-th row of $\mathbf{H}$), $\hat{y}_v \in \mathbb{R}^C$ its predicted class distribution, $y_v \in \mathbb{R}^C$ its one-hot true label, and $\ell_v = \texttt{CrossEntropy}(\hat{y}_v, y_v)$ its cross-entropy loss. The gradient of the cross-entropy loss at node v with respect to $\mathbf{W}_2$ is:

$$\nabla_{\mathbf{W}_2} \ell_v = h_v(\hat{y}_v - y_v)^\top \in \mathbb{R}^{d \times C}. \tag{6}$$

This is a rank-1 matrix formed by the outer product of the hidden representation h_v and the residual error vector $(\hat{y}_v - y_v)$. For a non-memorized node c and a memorized node m with assigned label $\tilde{y}_m \neq y_m$, the Frobenius inner product between their gradients factorizes exactly as:

$$\langle \nabla_{\mathbf{W}_2} \ell_c, ; \nabla_{\mathbf{W}_2} \ell_m \rangle_F = \underbrace{\langle h_c, h_m \rangle}_{\text{representation term}} \cdot \underbrace{(\hat{y}_c - y_c)^\top (\hat{y}_m - \tilde{y}_m)}_{\text{error alignment term}}. \tag{7}$$

The cosine similarity between the two gradients therefore has the same sign as the product of these two scalar terms. We now analyse each term separately as a function of graph homophily h.

Assumption 1 (Feature clustering by homophily). In a graph with homophily level $h \in [0, 1]$, the expected inner product between the hidden representations of two nodes of the same true class increases with h, and that of nodes of different classes decreases with h:

$$\mathbb{E}[\langle h_u, h_v \rangle \mid y_u = y_v] = \mu_{\text{same}}(h), \quad \mathbb{E}[\langle h_u, h_v \rangle \mid y_u \neq y_v] = \mu_{\text{diff}}(h), \tag{8}$$

where $\mu_{\text{same}}(h)$ is non-decreasing in h and $\mu_{\text{diff}}(h)$ is non-increasing in h, with $\mu_{\text{same}}(h) > \mu_{\text{diff}}(h)$ for $h > 0$. This assumption is natural: in a homophilic graph, neighbourhood aggregation reinforces within-class similarity in node representations, as each node's hidden state is a weighted average of its (predominantly same-class) neighbours' features.

Assumption 2 (Error vector structure). For a well-trained model, the softmax output $\hat{y}_v$ approximates the true class posterior. For a clean node c of class k, the error vector satisfies:

$$\hat{y}_c - y_c \approx \epsilon_c \mathbf{e}_k^{\perp}, \tag{9}$$

where $\mathbf{e}_k^{\perp}$ denotes a vector with a small negative component on class k and small positive components elsewhere, and $\epsilon_c > 0$ is small (low loss for a well-learned example). For a memorized node m of true class k but assigned class $\tilde{k} \neq k$, the error vector satisfies:

$$\hat{y}_m - \tilde{y}_m \approx \hat{y}_m - \mathbf{e}_{\tilde{k}}, \tag{10}$$

where the model predicts something close to the true class k (due to structural aggregation) but is penalised toward class $\tilde{k}$. The error vector therefore has a large negative component on $\tilde{k}$ and a positive component on k.

Proposition (Gradient alignment decreases with homophily). *Under Assumptions 1–2, consider a non-memorized node c of class k and a memorized node m with true class k and assigned class $\tilde{k} \neq k$. The expected inner product $\langle \nabla_{\mathbf{W}_2} \ell_c, \nabla_{\mathbf{W}_2} \ell_m \rangle_F$ is a decreasing function of graph homophily h, and becomes negative in the high-homophily regime.*

D.2 Proof

We analyse the two terms in Eq. (7) separately.

Representation term $\langle h_c, h_m \rangle$. Since c and m share the same true class k, by Assumption 1:

$$\mathbb{E}[\langle h_c, h_m \rangle] = \mu_{\text{same}}(h), \tag{11}$$

which is non-decreasing in h and positive for $h > 0$.

Error alignment term $(\hat{y}_c - y_c)^\top (\hat{y}_m - \tilde{y}_m)$. By Assumption 2, $(\hat{y}_c - y_c)$ has a negative component on class k and small positive components elsewhere. The vector $(\hat{y}_m - \tilde{y}_m)$ has a large negative component on $\tilde{k}$ and a positive component on class k (since the model is pulled toward k by the neighbourhood signal in proportion to h). Therefore:

$$(\hat{y}_c - y_c)^\top (\hat{y}_m - \tilde{y}_m) \approx -\epsilon_c \cdot p_k + \text{small positive terms}, \tag{12}$$

where $p_k = [\hat{y}_m]_k$ is the model's predicted probability on the true class k for the mislabeled node. As homophily increases, neighbourhood aggregation drives $[\hat{y}_m]_k \to 1$ (the model's representation for m becomes increasingly class-k consistent due to same-class neighbours), so p_k increases with h. Consequently:

$$\frac{\partial}{\partial h} \left[(\hat{y}_c - y_c)^\top (\hat{y}_m - \tilde{y}_m) \right] < 0, \tag{13}$$

and for sufficiently large h, the error alignment term becomes dominated by the $-\epsilon_c \cdot p_k$ component and is negative.

Combining the two terms. The gradient inner product in Eq. (7) is:

$$\langle \nabla_{\mathbf{W}_2} \ell_c, ; \nabla_{\mathbf{W}_2} \ell_m \rangle_F = \mu_{\text{same}}(h) \cdot \left[(\hat{y}_c - y_c)^\top (\hat{y}_m - \tilde{y}_m) \right]. \tag{14}$$

Since $\mu_{\text{same}}(h) > 0$ for $h > 0$, the sign of the inner product is determined entirely by the error alignment term. As established above, the error alignment term is a decreasing function of h and becomes negative for large h. Therefore, the gradient inner product is a decreasing function of h, transitioning from positive (at low h, where the error alignment term is weakly positive due to the noisy, class-uninformative neighbourhood signal) to negative (at high h, where the neighbourhood signal strongly aligns the mislabeled node's representation with its true class, creating a large negative error alignment), which proves the proposition.

Low-homophily case. When $h \approx 0$, neighbourhood aggregation yields a class-uninformative signal and $[\hat{y}_m]_k \approx 1/C$ (near-uniform prediction on the true class). In this regime, the error alignment term is weakly positive and the gradient inner product is positive, consistent with Proposition 1.

Remark. The analysis above focuses on the final-layer gradient $\nabla_{\mathbf{W}_2}$. For the first-layer gradient $\nabla_{\mathbf{W}_1}$, an analogous factorization holds via the chain rule, with the representation term replaced by raw input features x_v and an additional propagation factor through $\tilde{\mathbf{A}}$. The qualitative conclusion, *i.e.,* gradient alignment decreases with homophily, extends to earlier layers, though the effect is attenuated as the direct label signal weakens through backpropagation. This is consistent with Fig. 2, where the cosine similarity pattern is observed across all layers but is most significant at Layer 2.

E Theoretical Analysis of Final-layer Memorization Concentration

We provide a theoretical analysis for Proposition 2. We here take the controlled label noise setup as the example, where the memorized and non-memorized nodes are the mislabeled and clean nodes, respectively.

E.1 Setup

Consider an l-layer GNN with hidden representations:

$$\mathbf{H}^{(k)} = \sigma\Big(\tilde{\mathbf{A}}\mathbf{H}^{(k-1)}\mathbf{W}_k\Big), \quad k = 1, \ldots, l, \tag{15}$$

with $\mathbf{H}^{(0)} = \mathbf{X}$ and final logits $\mathbf{Z} = \mathbf{H}^{(l-1)}\mathbf{W}_l$. For simplicity, we consider a linearized GNN (removing the nonlinearity σ), so that:

$$\mathbf{H}^{(k)} = \tilde{\mathbf{A}}^k \mathbf{X}\mathbf{W}_1\mathbf{W}_2 \cdots \mathbf{W}_k \triangleq \tilde{\mathbf{A}}^k \mathbf{X}\mathbf{W}_{1:k}. \tag{16}$$

The logit of node v at the output is:

$$z_v = h_v^{(l-1)}\mathbf{W}_l = [\tilde{\mathbf{A}}^{l-1}\mathbf{X}\mathbf{W}_{1:l-1}]_v \mathbf{W}_l. \tag{17}$$

E.2 Proof

Collateral influence of layer-k weight perturbations. Let $\delta\mathbf{W}_k$ denote a small perturbation to the weight matrix at layer k. The induced change in the logit of node u is:

$$\delta z_u^{(k)} = \frac{\partial z_u}{\partial \mathbf{W}_k} \cdot \delta\mathbf{W}_k = \Big[\tilde{\mathbf{A}}^{l-1}\mathbf{X}\mathbf{W}_{1:k-1}\Big]_u \delta\mathbf{W}_k \mathbf{W}_{k+1:l-1}\mathbf{W}_l. \tag{18}$$

The key observation is the factor $[\tilde{\mathbf{A}}^{l-1}\mathbf{X}\mathbf{W}_{1:k-1}]_u$: for early layers ($k \ll l$), this involves $\tilde{\mathbf{A}}^{l-1}$, a $(l-1)$-step propagation operator that distributes the perturbation across the $(l-1)$-hop neighbourhood of u. Specifically, since $\tilde{\mathbf{A}}$ is a doubly stochastic-like matrix, $[\tilde{\mathbf{A}}^{l-1}]_{uv} > 0$ for all nodes v within $(l-1)$ hops of u, so $\delta\mathbf{W}_k$ affects the logits of *all* such nodes.

We define the *collateral set* of a weight perturbation at layer k as:

$$\mathcal{C}(v,k) = \{u \in \mathcal{V} : [\tilde{\mathbf{A}}^{l-k}]_{vu} > 0\}, \tag{19}$$

which is the set of nodes whose logits are non-trivially affected by a perturbation to $\mathbf{W}_k$ that targets node v. For an early layer ($k = 1$), $|\mathcal{C}(v,1)|$ is the size of the $(l-1)$-hop neighbourhood of v in the graph, which can be $O(n)$ in a dense or highly connected graph.

Final-layer perturbations are node-specific. For the final layer ($k = l$), the induced change in the logit of node u is:

$$\delta z_u^{(l)} = h_u^{(l-1)} \delta \mathbf{W}_l, \tag{20}$$

where $h_u^{(l-1)}$ is the pre-computed embedding of node u after $l-1$ rounds of message passing. Importantly, Eq. (20) depends only on $h_u^{(l-1)}$: a perturbation to $\mathbf{W}_l$ affects node u's logit through u's own embedding and *no other node's embedding*.

Argument. Early layers are shared across all nodes and are strongly constrained by graph smoothing/neighborhood aggregation. Memorizing a mislabeled node in early layers would perturb representations of many structurally related nodes. The final layer, however, maps already-computed node embeddings to class logits and can create local decision boundary exceptions more cheaply. Therefore, the lowest-cost way to learn (fit) a rare mislabeled node is to adjust final-layer classifier weights. This is GNN-specific because in GNNs, early layers are not just feature extractors; they are graph-coupled propagation operators.

Remark 1 (Reconciling `LMS` and `AWS`). This result clarifies the distinction between the two memorization measures, *i.e.,* `LMS` and `AWS`. During training, gradient accounting (`LMS`) measures how much each layer's weights are updated in response to the loss on mislabeled nodes. Mislabeled nodes exert gradient pressure on *all* layers because the backpropagated gradient flows through the entire computational graph. However, the trained model concentrates *prediction-critical* weights in the final layer because, as shown above, adjusting final-layer weights is the minimum-collateral memorization strategy. The model learns to memorize mislabeled nodes where it is cheapest to do so, *i.e.,* the final layer, while the gradient pressure propagated to earlier layers during training reflects the difficulty of fitting the wrong label against the structural signal, not the location of the stored memorization.

Remark 2 (GNN-specificity). This argument is specific to GNNs and does not apply to standard MLPs or CNNs, where early layers are not graph-coupled propagation operators. In a standard MLP, each layer processes nodes independently, so $\mathcal{C}(v,k) = \{v\}$ for all layers k and there is no collateral cost argument favouring the final layer. This explains why Maini et al. (2023) find memorization distributed across layers in DNNs, whereas we observe concentration in the final layer for GNNs.

F Hyperparameters

We use PyTorch Geometric (Fey & Lenssen, 2019) for all of our experiments. The hyperparameters used for training our models, the dataset statistics such as the number of nodes and edges, and the homophily level are reported in Table 4. All of our experiments were done on an A100 GPU, highlighting the computational efficiency of our proposed framework.

G Additional Results

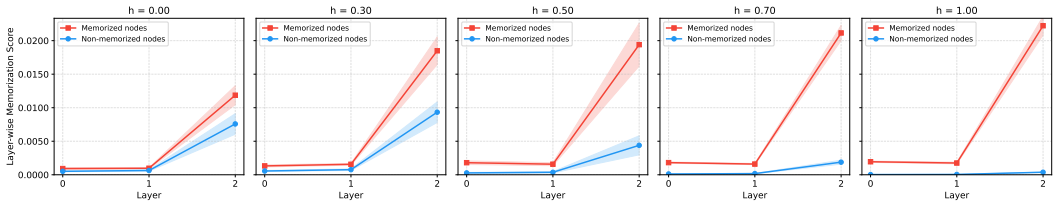
In this section, we provide additional results to complement the findings presented in the main text.

G.1 Layer-wise Memorization Scores on `syn-cora`

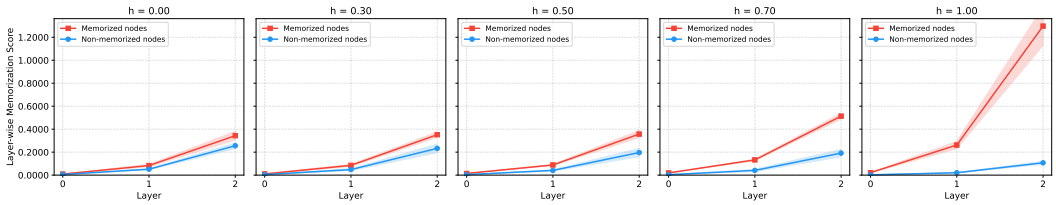
Fig. 5 shows the layer-wise memorization scores of memorized and non-memorized nodes (based on controlled label noise approach) on the other two GNN models, *i.e.,* GAT, and GraphSAGE, trained on `syn-cora` dataset, with $\rho = 0.01$. Consistent with the finding observed in GCN model, memorized nodes dominate gradient contributions across all layers.

Table 4: **Hyperparameters and dataset statistics.** η: learning rate, $|\mathcal{V}|$: number of nodes, $|\mathcal{E}|$: number of edges.

Dataset	Hidden Dimension	η	#Layers	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	# Classes	Homophily Level
Cora	64	0.01	3	2708	10138	7	0.7637
Citeseer	64	0.01	3	3327	7358	6	0.6620
Chameleon	128	0.001	3	2277	36101	5	0.0339
Squirrel	128	0.001	3	5201	217073	5	0.0074
Actor	128	0.001	3	7600	30019	5	0.0062
ogbn-arxiv	128	0.001	3	169343	1166243	40	0.5883
Coauthor-CS	128	0.001	3	18333	163788	15	0.7851
Coauthor-Physics	128	0.001	3	34493	495924	5	0.8722
`syn-cora`	64	0.01	3	1490	2965 to 2968	5	0.0 to 1.0



(a) GAT.



(b) GraphSAGE.

Figure 5: **Layer-wise memorization scores of memorized and non-memorized nodes (based on controlled label noise) across homophily levels and model layers for GAT and GraphSAGE trained on `syn-cora`, ρ = 0.01.** Overall, memorized nodes dominate gradient contributions across all layers.

Fig. 6 shows the `LMS` of memorized and non-memorized nodes (based on NCMemo score) across homophily levels and model layers. Consistent with the findings on controlled label noise setup, we can observe that the layer-wise memorization of memorized nodes exceeds that of non-memorized nodes at every layer, with the gap increasing with layer depth.

We present the cosine similarity between the average gradient norms of memorized (mislabeled) and non-memorized (clean) nodes with GAT and GraphSAGE models, in Fig. 7. It can be observed that the cosine similarity between the average gradient of memorized and non-memorized nodes decreases with graph homophily.

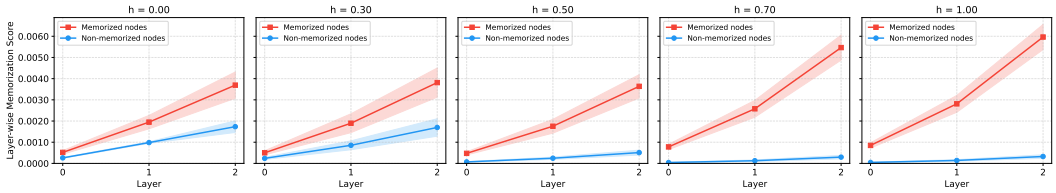
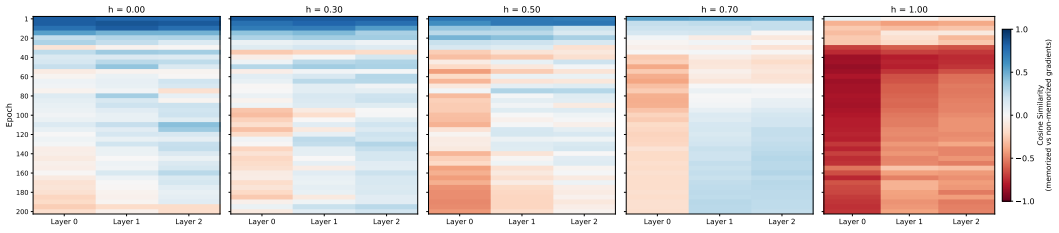
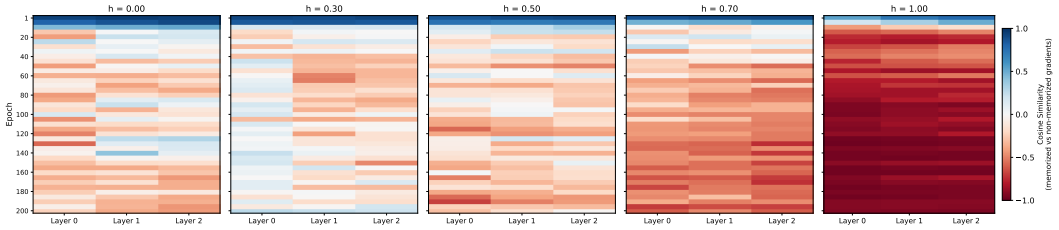


Figure 6: **Layer-wise memorization scores of memorized and non-memorized nodes (based on NCMemo score) across homophily levels and model layers (GCN, `syn-cora`).** Overall, memorized nodes dominate gradient contributions across all layers.



(a) GAT.



(b) GraphSAGE.

Figure 7: **Cosine similarity between the average gradients of memorized (mislabeled) and non-memorized (clean) nodes per layer, per epoch, with GAT and GraphSAGE trained on `syn-cora`, ρ = 0.01.** Blue indicates positive similarity; red indicates negative similarity. We can observe an inverse relationship between gradient cosine similarity over memorized and non-memorized nodes and the graph homophily.

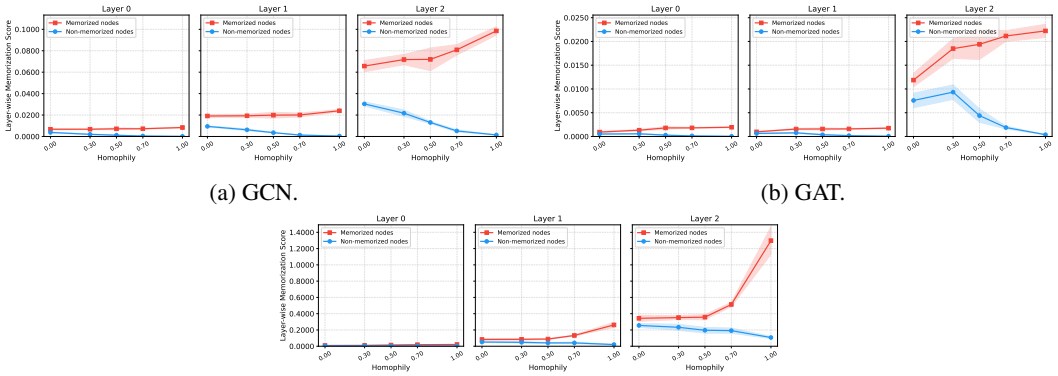
G.2 Effect of Homophily on Layer-wise Memorization

We also analyze the layer-wise memorization of memorized (mislabeled) and non-memorized (clean) nodes across homophily levels for each layer in Fig. 8. We can observe that the gradient norms for both memorized and non-memorized nodes increase with layer depth, consistent with the observations in Fig. 1. We also observe that the memorized and non-memorized nodes exhibit opposite layer-wise memorization trends with respect to homophily. The average gradient norm of non-memorized nodes decreases as homophily increases, approaching near-zero at $h = 1.00$ across all layers. In contrast, the gradient norm of memorized nodes exhibits a general *increase* with homophily, most visibly at Layer 2. This trend is explained by the increasing structural conflict faced by memorized nodes as homophily grows. In a highly homophilic graph (*e.g.,* $h = 1.00$), memorized nodes are surrounded almost by neighbours of the same true class, making them highly distinctive outliers whose labels strongly contradict the aggregated neighbourhood evidence. The model cannot rely on either node features or graph structure to learn these nodes in a generalizable way; the only path to reducing their training loss is to memorize them directly, resulting in large gradient norms throughout training.

Finding 3 (Opposite homophily trends in `LMS`.) *The layer-wise memorization of non-memorized nodes decreases with increasing graph homophily across all layers, while the average gradient norm of memorized nodes increases with graph homophily, reflecting heightened memorization pressure as memorized nodes become increasingly isolated outliers in strongly homophilic graphs.*

G.3 Effect of Memorization Rate on Layer-wise Memorization

Our previous analyses have focused on a fixed memorization rate of $\rho = 0.01$, where mislabeled nodes are rare outliers. We now investigate how increasing the memorization rate, and thus the proportion of mislabeled nodes, affects layer-wise memorization patterns. We plot the layer-wise memorization of memorized (mislabeled) and non-memorized (clean) nodes at Layer 2 across different memorization rates for each homophily level in Fig. 9. We can observe that the layer-wise memorization of non-memorized nodes remains largely stable as ρ increases from 0.01 to 0.05, confirming that the learning dynamics of non-memorized nodes are largely unaffected by the proportion of mislabeled nodes in the training set. In contrast, the `LMS` of memorized nodes gradually decreases with increasing memorization rate across all homophily levels, indicating a transition from individual memorization



(a) GCN.

(b) GAT.

(c) GraphSAGE.

Figure 8: **Layer-wise memorization scores of memorized (mislabeled) and non-memorized (clean) nodes across all homophily levels for each model layer (`syn-cora`, $\rho = 0.01$).** Overall, the layer-wise memorization score of memorized nodes increases with homophily.

at low memorization rates to pattern-based learning as memorized nodes become more numerous, but still remains higher than that of non-memorized nodes. Based on this observation, we formalize the following finding.

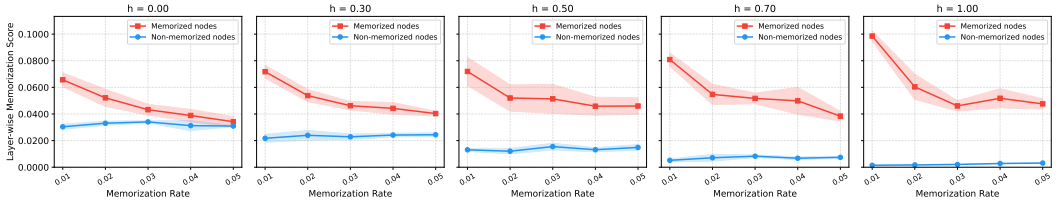
Finding 4 (Transition from individual memorization to pattern-based learning.) *The gradient norms of memorized (mislabeled) nodes decrease with increasing memorization rate across all homophily levels. As the memorization rate increases, the memorization on mislabeled nodes gradually decreases, indicating a transition from individual memorization at low memorization rates to pattern-based learning as memorized nodes become more numerous, but still remains higher than that of non-memorized (clean) nodes.*

This pattern reflects a qualitative shift in how the model handles memorized nodes as their proportion grows. At low memorization rates ($\rho = 0.01$), memorized nodes constitute rare outliers with no coherent pattern amongst themselves; the model has no choice but to memorize each one individually to reduce their training loss, resulting in large and sustained gradient norms. As the memorization rate increases, memorized nodes become more numerous and begin to exhibit a statistical pattern of their own, *i.e.,* a shared characteristic of being label-inconsistent with graph structure. The model can then exploit this pattern to make predictions for memorized nodes without fully memorizing each individual example, reducing the per-node loss and thus the gradient norms. This transition from individual memorization to pattern-based learning with increasing memorization rate mirrors the long-tail memorization hypothesis of Feldman (2020), and suggests that the memorization in GNNs is most acute precisely when memorized nodes are rare and structurally isolated.

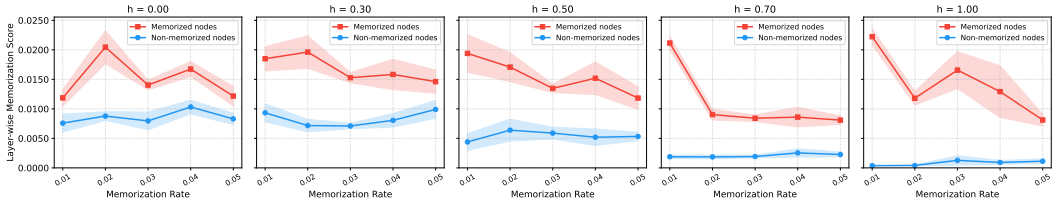
G.4 Ablated Weight Score on `syn-cora`

Fig. 10 shows the distribution of $\texttt{AWS}(v)$ for memorized (mislabeled) and non-memorized (clean) nodes across homophily levels for GAT and GraphSAGE models trained on `syn-cora` dataset, with $\rho = 0.01$. Consistent with the finding observed in GCN model, the $\texttt{AWS}(v)$ of memorized nodes is significantly lower than that of non-memorized nodes across all homophily levels, indicating that the memorization of memorized nodes is more concentrated on a smaller subset of weights compared to non-memorized nodes.

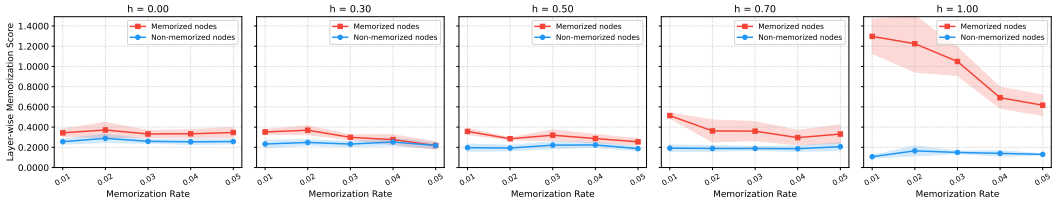
We also show the distribution of $\texttt{AWS}(v)$ for memorized and non-memorized nodes based on NCMemo across homophily levels on `syn-cora` dataset in Fig. 11. Equivalent to our findings with memorized and non-memorized nodes based on controlled label noise approach, `AWS` of memorized nodes is significantly lower than that of non-memorized nodes across all homophily levels.



(a) GCN.

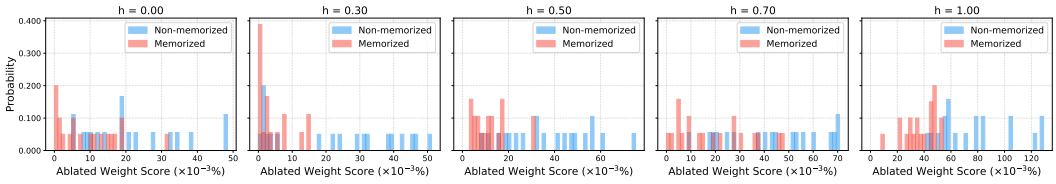


(b) GAT.

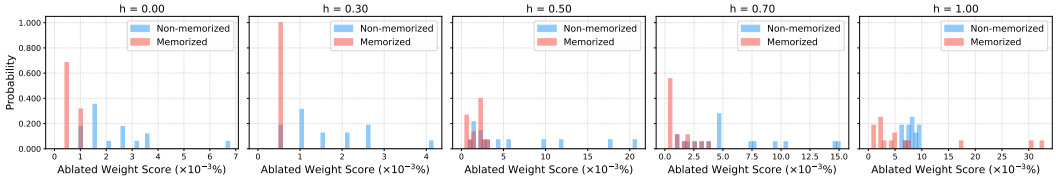


(c) GraphSAGE.

Figure 9: **Layer-wise memorization scores of memorized (mislabeled) and non-memorized (clean) nodes at Layer 2 across different memorization rates, for each homophily level.** Overall, the layer-wise memorization score of memorized nodes decreases with increasing memorization rate.



(a) GAT.



(b) GraphSAGE.

Figure 10: **Distribution of $\texttt{AWS}(v)$ ($\times 10^{-3}\%$ of total model weights ablated) for memorized (mislabeled) and non-memorized (clean) nodes across homophily levels, for a 3-layer GAT and GraphSAGE trained on `syn-cora`,** $\rho = 0.01$**.** Lower AWS indicates more concentrated memorization.

G.5 Layer Distribution of Ablated Weights

Fig. 12 shows the layer-wise distribution of ablated weights for memorized (mislabeled) and non-memorized (clean) nodes across all five homophily levels ($h \in 0.00, 0.30, 0.50, 0.70, 1.00$) and all five memorization rates ($\rho \in 0.01, 0.02, 0.03, 0.04, 0.05$). Across all memorization rates, almost ablated weights are concentrated in Layer 2 for both sample types and all homophily levels.

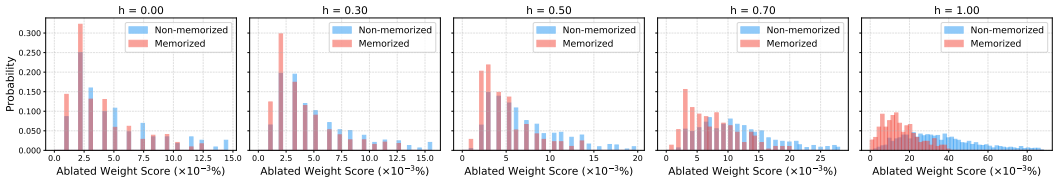


Figure 11: **Distribution of $\texttt{AWS}(v)$ ($\times 10^{-3}\%$ of total model weights ablated) for memorized and non-memorized nodes based on NCMemo (GCN, `syn-cora`).**

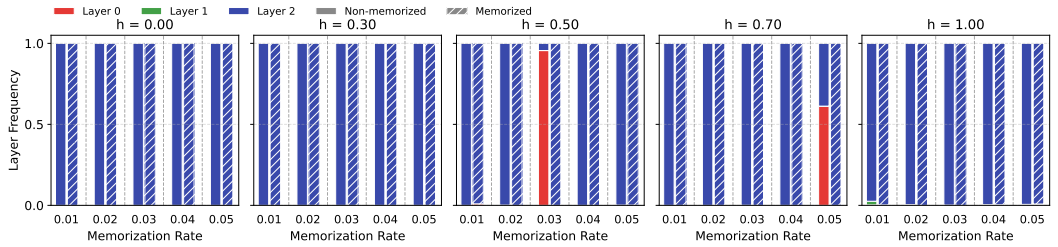
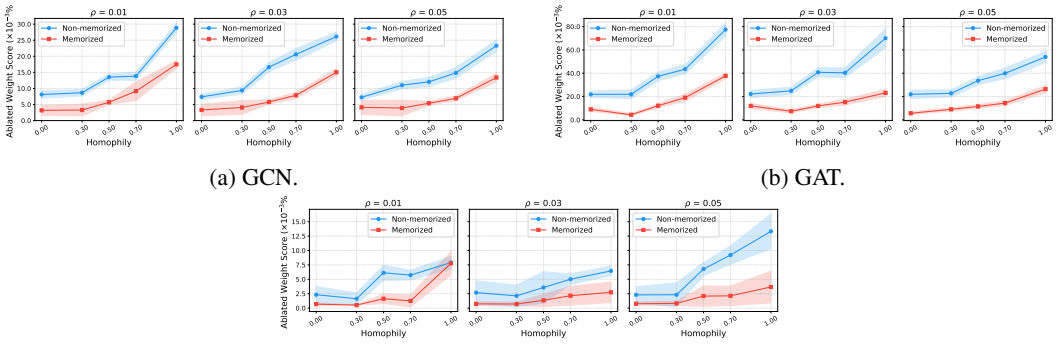


Figure 12: **Layer-wise distribution of ablated weights for memorized (mislabeled) and non-memorized (clean) nodes across all homophily levels and memorization rates (GCN, `syn-cora`).** Within each panel, bars for memorized and non-memorized nodes are placed side by side for each memorization rate group.

G.6 Effect of Homophily on Ablated Weight Score



(a) GCN.

(b) GAT.

(c) GraphSAGE.

Figure 13: **Average $\texttt{AWS}(v)$ ($\times 10^{-3}\%$ of total model weights ablated) for memorized (mislabeled) and non-memorized (clean) nodes, across graph homophily levels and memorization rates (`syn-cora`).**

We present the average AWS of both memorized (mislabeled) and non-memorized (clean) nodes across various homophily levels and memorization rates in Fig. 13. We can observe a clear positive relationship is evident for both node types, *i.e.,* AWS increases from $h = 0.00$ to $h = 1.00$ across all memorization rates. Higher homophily leads to more structurally entangled predictions that require more weights to be ablated for a prediction flip, resulting in higher AWS scores. Conversely, lower homophily results in less structurally entangled predictions that require fewer weights to be ablated for a prediction flip, leading to lower AWS scores. This suggests that the structural properties of the graph, such as homophily, play a significant role in shaping the memorization patterns of GNNs at not only the node level but also the weight level.

H EXTENSION TO REAL-WORLD DATASETS

In this section, we present results on layer-wise and weight-wise memorization on real-world datasets. We report results on real-world datasets with varying levels of homophily, including Cora (McCallum et al., 2000), Citeseer (Yang et al., 2016), which are homophilous, and Squirrel, Actor, Chameleon (Pei et al., 2020), which are heterophilous. Detailed dataset statistics, including the homophily level of each dataset, are presented in Appx. F. The observations are consistent with our findings on synthetic datasets

H.1 LAYER-WISE MEMORIZATION ON REAL-WORLD DATASETS

We present the layer-wise memorization scores of memorized (mislabeled) and non-memorized (clean) nodes across different layers and real-world datasets in Fig. 14. We can observe that the LMS of memorized nodes exceeds that of non-memorized nodes at every layer and across all datasets, with the gap increasing with layer depth. This observation is consistent with our findings on synthetic datasets.

The cosine similarities between memorized (mislabeled) and non-memorized (clean) gradients across layers for real-world datasets are presented in Fig. 15. We can observe that on homophilous graphs (Cora and Citeseer), the cosine similarities between memorized and non-memorized gradients are generally negative across all layers, while on heterophilous graphs (Actor, Squirrel, Chameleon), the cosine similarities are generally positive across all layers. This observation is consistent with our findings on synthetic datasets. The effect of graph homophily on layer-wise memorization scores for real-world datasets is presented in Fig. 16. We can observe that as homophily increases, the layer-wise memorization scores of non-memorized nodes decrease with homophily while that of memorized nodes increases.

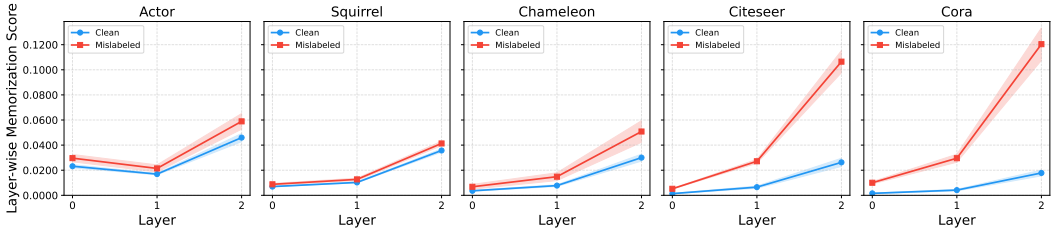


Figure 14: **Layer-wise memorization scores of memorized (mislabeled) and non-memorized (clean) nodes across model layers for a 3-layer GCN trained on 5 real-world datasets,** $\rho = 0.01$**.**

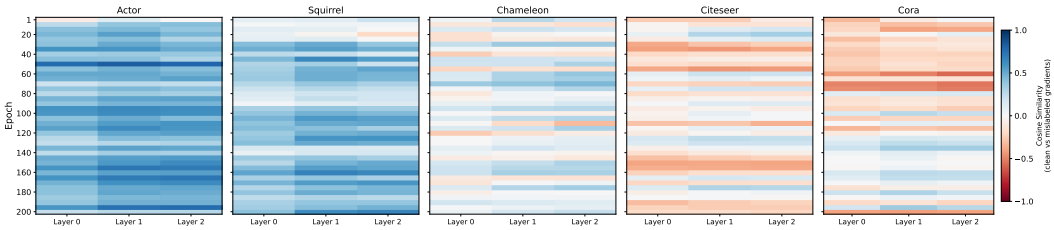


Figure 15: **Cosine similarity between the average gradients of memorized (mislabeled) and non-memorized (clean) nodes per layer, per epoch, across real-world datasets with different homophily levels (GCN,** $\rho = 0.01$**).** Blue indicates positive similarity; red indicates negative similarity.

H.2 ABLATED WEIGHT SCORE ON REAL-WORLD DATASETS

We show the distribution of Ablated Weight Score for memorized (mislabeled) and non-memorized (clean) nodes across real-world datasets in Fig. 17. We can observe that the AWS(v) of memorized nodes is significantly lower than that of non-memorized nodes across all datasets, indicating that the

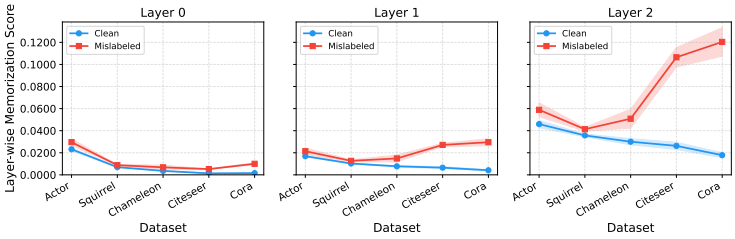


Figure 16: **Layer-wise memorization scores of memorized (mislabeled) and non-memorized (clean) nodes across real-world datasets for each layer, with GCN, $\rho = 0.01$.**

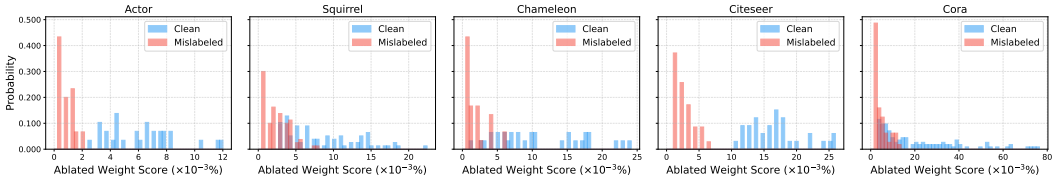


Figure 17: **Distribution of $\texttt{AWS}(v)$ ($\times 10^{-3}\%$ of total model weights ablated) for memorized (mislabeled) and non-memorized (clean) nodes for a 3-layer GCN trained on real-world datasets, $\rho = 0.01$.** Lower AWS indicates more concentrated memorization.

memorization of memorized nodes is more concentrated on a smaller subset of weights compared to non-memorized nodes. We also present the average $\texttt{AWS}(v)$ of memorized and non-memorized nodes across real-world dataset with different homophily levels in Fig. 18. We can observe that there is a positive relationship between the homophily level of a dataset and the average $\texttt{AWS}(v)$ of its nodes.

We also investigate our LMS and AWS on larger graph datasets, including *ogbn-arxiv* (Wang et al., 2020; Mikolov et al., 2013), *Coauthor-Physics* (Shchur et al., 2018), and *Coauthor-CS* (Shchur et al., 2018), ranging from $18,000$ to $169,000$ nodes and $82,000$ to $1,170,000$ edges, and a more advanced GNN backbone, *Graph Transformer* (Yunsheng et al., 2021) (Appx. K).

I LMS AND AWS OF FITTED AND NON-FITTED MISLABELED NODES

After training, we run inference on all memorized nodes to separate them into fitted mislabeled nodes (model predicts the mislabeled label, confirming memorization) and non-fitted mislabeled nodes (model does not predict the mislabeled label). Across all settings, over 85% of mislabeled nodes are fitted (memorized), confirming that the majority are indeed memorized by the model. We then report LMS and AWS separately for clean nodes, non-fitted mislabeled nodes, fitted mislabeled nodes, and all mislabeled nodes (5 real-world datasets, GCN, $\rho = 0.01$) in Fig. 19 and Table 5.

We find that fitted mislabeled nodes exhibit comparable LMS to all mislabeled nodes, *i.e.,* they dominate the gradients across all layers (see Finding 1). For AWS, fitted mislabeled nodes have even lower AWS than all mislabeled nodes, confirming that memorized nodes rely on fewer, more

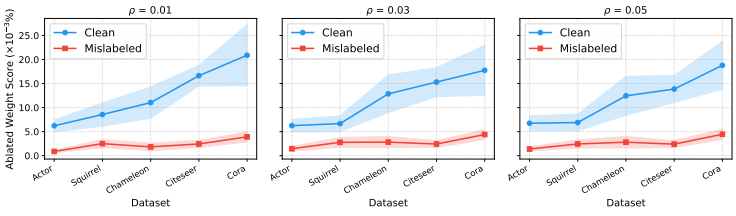


Figure 18: **Average $\texttt{AWS}(v)$ ($\times 10^{-3}\%$ of total model weights ablated) for memorized (mislabeled) and non-memorized (clean) nodes nodes, across different real-world datasets and noise rates (GCN).**

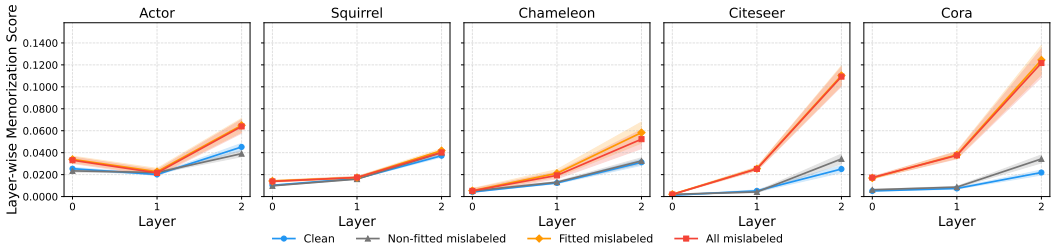


Figure 19: **Layer-wise memorization scores of clean, non-fitted, fitted and all mislabeled nodes across model layers for a 3-layer GCN trained on 5 real-world datasets,** $\rho = 0.01$**.**

Table 5: **Average AWS ($\times 10^{-3}\%$ of total model weights, mean) for clean, non-fitted, fitted and all mislabeled nodes across 5 real-world datasets (GCN,** $\rho = 0.01$**).** Fitted and all mislabeled nodes have similar AWS, which is much lower than clean nodes.

Node Type	Dataset				
	Actor	Squirrel	Chameleon	Citeseer	Cora
Clean	6.93	8.39	11.94	17.48	20.83
Non-fitted mislabeled	5.32	8.43	12.59	18.39	21.58
Fitted mislabeled	1.02	3.01	2.31	4.97	5.19
All mislabeled	1.35	3.04	2.48	5.04	5.27

concentrated weights (see Finding 2). These results confirm that our assumption that mislabeled nodes are a good proxy for memorized nodes is valid.

J Distinguish LMS and AWS from Other Effects

To distinguish LMS and AWS from effects of per-example loss, prediction margin, and sample difficulty, we conducted two complementary analyses: (1) *Loss-matched control experiment.* We first calculate the average cross-entropy loss of fitted mislabeled nodes (predicted as their assigned incorrect label) over all epochs, defined as L_m. Then, we select the subset of clean nodes with the closest cross-entropy loss to L_m. The size of the selected clean nodes is the same as that of the fitted mislabeled nodes; (2) *Margin-matched control experiment.* We further match clean nodes by prediction margin, *i.e.,* based on the difference between top-two logits. For both approaches, we refer the matched clean nodes as *difficult non-memorized nodes*.

We then compare the LMS and AWS of these two kinds of difficult non-memorized nodes against fitted mislabeled nodes, as shown in Fig. 20 and Table 6. We find that both loss-matched and margin-matched clean nodes, *i.e.,* difficult non-memorized nodes, exhibit significantly lower LMS and higher AWS than memorized nodes, confirming that our findings/analysis on LMS and higher AWS are not confounded by per-sample loss, prediction margin or sample difficulty, but are indeed driven by memorization.

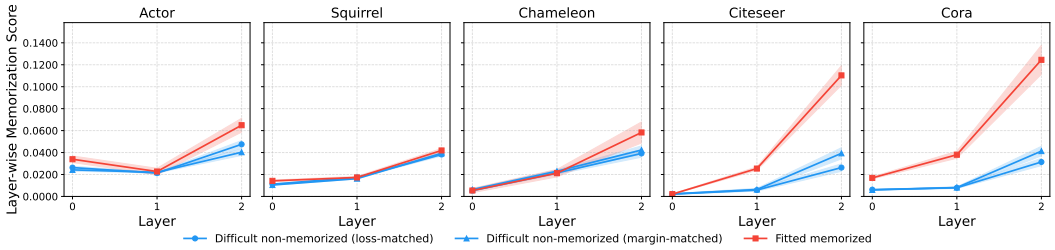


Figure 20: **Layer-wise memorization scores of difficult non-memorized nodes (loss-matched), difficult non-memorized nodes (margin-matched), and fitted mislabeled nodes across model layers for a 3-layer GCN trained on 5 real-world datasets,** $\rho = 0.01$**.**

Table 6: **Average AWS ($\times 10^{-3}\%$ of total model weights, mean) for two difficulty-matched non-memorized nodes, and fitted mislabeled nodes across 5 real-world datasets (GCN, $\rho = 0.01$).** Difficult non-memorized nodes have higher AWS than fitted memorized nodes.

Node Type	Dataset				
	Actor	Squirrel	Chameleon	Citeseer	Cora
Difficult non-memorized (loss-matched)	6.82	8.27	10.32	15.48	18.43
Difficult non-memorized (margin-matched)	5.29	7.93	11.48	16.43	19.43
Fitted mislabeled	1.02	3.01	2.31	4.97	5.19

Table 7: **Layer-wise memorization score of memorized (mislabeled) and non-memorized (clean) nodes on large-scale graph datasets (GCN, $\rho = 0.01$).** Memorized nodes exhibit higher LMS than non-memorized nodes at every layer, with the gap widening in deeper layers.

Dataset	Node type	Layer 0	Layer 1	Layer 2
ogbn-arxiv	Non-memorized	0.0020	0.0021	0.0028
	Memorized	0.0336	0.0367	0.0498
Coauthor-Physics	Non-memorized	0.0002	0.0016	0.0066
	Memorized	0.0059	0.0623	0.2695
Coauthor-CS	Non-memorized	0.0002	0.0028	0.0072
	Memorized	0.0044	0.0545	0.1429

K Extension to Larger Graphs and More Advanced Architecture

To support the practical applicability of our methods, we also evaluate our approaches on larger graph datasets, including *ogbn-arxiv* (Wang et al., 2020; Mikolov et al., 2013), *Coauthor-Physics* (Shchur et al., 2018), and *Coauthor-CS* (Shchur et al., 2018), ranging from $18,000$ to $169,000$ nodes and $82,000$ to $1,170,000$ edges. The LMS and AWS results are presented in Table 7 and Table 8, respectively.

We also evaluate our approaches on a more advanced GNN backbone, *Graph Transformer* (Yunsheng et al., 2021), on three real-world datasets (Actor, Chameleon, and Cora). The results of LMS and AWS are presented in Table 9 and Table 10, respectively.

From the above results, we can observe that memorized nodes still dominate the layer-wise gradients across all layers, and the AWS of memorized nodes is consistently lower than that of non-memorized nodes, demonstrating that our proposed memorization metrics, LMS and AWS, can be applied to more complex and large-scale scenarios.

L Extension to Other Graph Tasks

In this section, we discuss how LMS and AWS can be extended to other graph-related tasks beyond the node classification task. We show that our approaches can be adapted to *graph classification*, where the unit of memorization is an entire graph, and *link prediction*, where the unit is an edge.

L.1 Graph Classification

We adapt our LMS and AWS approaches to the graph classification task by changing the dataset to contain graphs rather than nodes. We evaluate on four graph classification datsets, namely MUTAG, AIDS, BZR, and COX2 (Morris et al., 2020), using a four-layer GCN model (3 convolutional layers followed by a linear head layer). We flip the labels of 1% of the entire training graphs as the memorized (mislabeled) graphs and compute LMS and AWS at the graph level. Tables 11 and 12 show the results for LMS and AWS, respectively. Consistent with the findings for node classification, memorized (mislabeled) graphs exhibit higher LMS than non-memorized (clean) graphs, with the gap increasing at deeper layers, and lower AWS than non-memorized graphs, indicating that our methods effectively capture memorization at the graph level.

Table 8: **Average AWS ($\times 10^{-3}\%$ of total model weights) for memorized (mislabeled) and non-memorized (clean) nodes on large-scale graph datasets (GCN, $\rho = 0.01$).**

Dataset	Non-memorized nodes	Memorized nodes
ogbn-arxiv	18.84	8.76
Coauthor-Physics	4.31	1.94
Coauthor-CS	2.56	0.70

Table 9: **Layer-wise memorization score of memorized (mislabeled) and non-memorized (clean) nodes across 3 real-world datasets (Graph Transformer, $\rho = 0.01$), in units of 10^{-2}.**

Dataset	Node type	Layer 0	Layer 1	Layer 2
Actor	Non-memorized	0.0453	0.0164	0.1732
	Memorized	0.1752	0.0673	0.6764
Chameleon	Non-memorized	0.0064	0.0081	0.0919
	Memorized	0.1188	0.0882	0.8050
Cora	Non-memorized	0.0077	0.0143	0.2441
	Memorized	0.0553	0.0778	1.2261

L.2 Link Prediction

For link prediction, the only conceptual change is that the unit of memorization becomes an edge between a pair of nodes, which can be present or absent. We evaluate on Cora, Citeseer, and Coauthor-CS, using an encoder-decoder model including an 3-layer GCN encoder and a 2-layer MLP decoder, and flip the labels (present/absent) of 1% of the training edges. Tables 13 and 14 show the LMS and AWS for memorized (mislabeled) and non-memorized (clean) edges, respectively. As in the node- and graph-level settings, memorized edges exhibit higher LMS and lower AWS than non-memorized edges, confirming that our findings extend to link prediction.

M Computational Efficiency

We analyze the computational efficiency of our two memorization measures, LMS and AWS, in terms of the time taken to compute them. It is notable that LMS is computed during training, which is obtained from the *same* forward and backward passes already performed for training the model. Therefore, the computation of LMS is in the same order as standard GNN training.

In contrast, AWS is computed after training, measured on a trained, frozen model, so it incurs no training cost. For each analyzed node, we greedily zero one weight at a time, selecting at each step the weight with the largest importance score, until the node's prediction flips. A node whose AWS is K (the number of weights removed before the flip) requires K steps, each consisting of one forward pass and one backward pass. The per-node cost is thus $O(K \cdot C_{fwd})$, where C_{fwd} is the cost of a forward pass. The cost is data-dependent, *i.e.,* more strongly memorized nodes (larger K) and larger models are more expensive to compute. Specifically, we report the average wall-clock time to compute AWS *per node* across all real-world datasets, on an A100 GPU, in Table 15. Times are averaged over analyzed nodes and seeds. The results highlight that determining the AWS score per node takes less than 0.15 seconds across all datasets and nodes.

We also implement two variants to calculate AWS with scalable approximation:

Batched top-k variant. Instead of ablating one weight per step, we ablate the top-k most important weights simultaneously at each step. This reduces the number of forward-backward passes per node from c (the number of ablation steps) to roughly c/k, yielding an approximately $k\times$ speedup in per-node wall-clock time. We calculate the relative AWS deviation for memorized and non-memorized nodes (based on controlled label noise setup), defined as $(\mathrm{AWS}_{\mathrm{top}-k} - \mathrm{AWS}_{\mathrm{exact}})$ / $\mathrm{AWS}_{\mathrm{exact}}$, where $\mathrm{AWS}_{\mathrm{top}-k}$ is the AWS value obtained with the top-k approximation, and $\mathrm{AWS}_{\mathrm{exact}}$ is the exact AWS value without approximation. The results are shown in Table 16 (taking the medium-scale real graph Citeseer, on GCN, as the example).

Table 10: **Average AWS (%) for memorized (mislabeled) and non-memorized (clean) nodes across 3 real-world datasets (Graph Transformer, $\rho = 0.01$).**

Dataset	Non-memorized nodes	Memorized nodes
Actor	0.0122	0.0015
Chameleon	0.0369	0.0034
Cora	0.1097	0.0103

Table 11: **Layer-wise memorization score of memorized (mislabeled) and non-memorized (clean) graphs (4-layer GCN, $\rho = 0.01$).** Memorized graphs exhibit higher LMS than non-memorized graphs, with the gap widening at deeper layers.

Dataset	Node type	Layer 0	Layer 1	Layer 2	Layer 3
MUTAG	Non-memorized	0.0014 ± 0.0003	0.0019 ± 0.0004	0.0083 ± 0.0027	0.0097 ± 0.0021
	Memorized	0.0091 ± 0.0047	0.0138 ± 0.0071	0.0523 ± 0.0181	0.0608 ± 0.0300
AIDS	Non-memorized	0.0029 ± 0.0011	0.0025 ± 0.0008	0.0073 ± 0.0038	0.0133 ± 0.0059
	Memorized	0.0267 ± 0.0054	0.0233 ± 0.0046	0.0513 ± 0.0182	0.1233 ± 0.0034
BZR	Non-memorized	0.0010 ± 0.0004	0.0019 ± 0.0009	0.0034 ± 0.0011	0.0072 ± 0.0057
	Memorized	0.0152 ± 0.0061	0.0246 ± 0.0072	0.0411 ± 0.0045	0.1252 ± 0.0476
COX2	Non-memorized	0.0019 ± 0.0009	0.0052 ± 0.0020	0.0106 ± 0.0042	0.0197 ± 0.0064
	Memorized	0.0151 ± 0.0044	0.0397 ± 0.0052	0.0833 ± 0.0222	0.1555 ± 0.0411

Batching across nodes. Instead of computing importance scores independently for each node, we batch B nodes together and share a single forward pass to compute gradients jointly. By doing so, the per-step gradient computation is amortized across the batch. This reduces runtime for AWS calculation, but batching computes importance from a summed gradient over B nodes rather than per-node gradients, so the ranking shifts and gradient cancellation across nodes introduces error. The results are shown in Table 17 (taking the medium-scale real graph Citeseer, on GCN, as the example).

These results show that it is possible to reduce the compute time of AWS significantly, while the relative deviation remains small, *i.e.,* even at the highest speedup of factor $\sim 25\times$ (when we ablate the top-10 most important weights), the relative AWS deviation remains below 13%, and we can still maintain the main conclusions in the paper (Finding 2). Additionally, we position AWS as a diagnostic/analysis tool for understanding memorization patterns, similar to how influence functions or leave-one-out mechanisms are used for model understanding rather than routine deployment.

Table 12: **Average AWS (%) for memorized (mislabeled) and non-memorized (clean) graphs (4-layer GCN, $\rho = 0.01$).**

Dataset	Non-memorized	Memorized
MUTAG	0.3140 ± 0.0168	0.1537 ± 0.0150
AIDS	0.0776 ± 0.0028	0.0345 ± 0.0058
BZR	0.1602 ± 0.0015	0.0617 ± 0.0013
COX2	0.2040 ± 0.0042	0.0736 ± 0.0041

Table 13: **LMS of memorized (mislabeled) and non-memorized (clean) edges (GCN encoder+MLP decoder, $\rho = 0.01$).**

Dataset	Edge type	Layer 0	Layer 1	Layer 2	Layer 3	Layer 4
Cora	Non-memorized	0.0019 ± 0.0002	0.0047 ± 0.0012	0.0060 ± 0.0030	0.0065 ± 0.0046	0.0377 ± 0.0287
	Memorized	0.0195 ± 0.0018	0.0403 ± 0.0144	0.0443 ± 0.0196	0.0695 ± 0.0564	0.6278 ± 0.0589
Citeseer	Non-memorized	0.0002 ± 0.0001	0.0018 ± 0.0000	0.0019 ± 0.0005	0.0012 ± 0.0005	0.0073 ± 0.0027
	Memorized	0.0058 ± 0.0006	0.0368 ± 0.0122	0.0457 ± 0.0070	0.0315 ± 0.0040	0.2215 ± 0.0470
Coauthor-CS	Non-memorized	0.0002 ± 0.0000	0.0016 ± 0.0000	0.0023 ± 0.0001	0.0031 ± 0.0008	0.0137 ± 0.0006
	Memorized	0.0067 ± 0.0017	0.0466 ± 0.0104	0.0848 ± 0.0162	0.1118 ± 0.0409	0.7194 ± 0.0885

Table 14: **Average AWS (%) for memorized (mislabeled) and non-memorized (clean) edges (GCN encoder+MLP decoder, $\rho = 0.01$).**

Dataset	Non-memorized edges	Memorized edges
Cora	0.0134 ± 0.0014	0.0086 ± 0.0008
Citeseer	0.0103 ± 0.0028	0.0062 ± 0.0003
Coauthor-CS	0.0174 ± 0.0032	0.0095 ± 0.0014

Table 15: **Average cost of AWS computation on GCN model (per node, in second).**

Dataset	Average AWS runtime for non-memorized (clean) nodes	Average AWS runtime for memorized (mislabeled) nodes
Actor	0.0242 ± 0.0013	0.0034 ± 0.0002
Squirrel	0.0880 ± 0.0072	0.0129 ± 0.0013
Chameleon	0.0588 ± 0.0050	0.0097 ± 0.0010
Citeseer	0.0505 ± 0.0034	0.0147 ± 0.0011
Cora	0.1333 ± 0.0201	0.0250 ± 0.0036

Table 16: **AWS computation cost based on batched top-k variant (Citeseer, GCN).**

k	Average AWS runtime for non-memorized nodes (per node, in second)	Average AWS runtime for memorized nodes (per node, in second)	Relative AWS deviation for non-memorized nodes (%)	Relative AWS deviation for memorized nodes (%)
Exact ($k = 1$)	0.0505 ± 0.0034	0.0147 ± 0.0011	-	-
5	0.0091 ± 0.0024	0.0066 ± 0.0011	3.56	6.03
10	0.0069 ± 0.0021	0.0023 ± 0.0007	4.22	8.46
20	0.0041 ± 0.0019	0.0020 ± 0.0005	6.84	12.33

Table 17: **AWS computation cost based on batching across nodes (Citeseer, GCN).**

Batch size B	Average AWS runtime for non-memorized nodes (per node, in second)	Average AWS runtime for memorized nodes (per node, in second)	Relative AWS deviation for non-memorized nodes (%)	Relative AWS deviation for memorized nodes (%)
Exact ($B = 1$)	0.0505 ± 0.0034	0.0147 ± 0.0011	-	-
4	0.0126 ± 0.0017	0.0037 ± 0.0006	1.45	2.36
8	0.0063 ± 0.0012	0.0018 ± 0.0004	4.45	8.13
16	0.0032 ± 0.0008	0.0009 ± 0.0003	10.25	14.10