From Multi-Class to Multi-Label: Revisiting Edge Dropping for Graph Neural Networks

Alexandru Andrei[1]

Supervisor(s): Megha Khosla[1]**, Elena Congeduti**[1]

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 15, 2026

Name of the student: Alexandru Andrei
Final project course: CSE3000 Research Project
Thesis committee: Megha Khosla, Elena Congeduti , Christoph Lofi

An electronic version of this thesis is available at http://repository.tudelft.nl/.

Abstract

Many real-world tasks involve data that is naturally shaped like a graph, with proteins linked by interactions, papers linked by citations or people linked in a social network, and a common goal is to predict properties of each entity from how it is connected to the rest. The models that do this, called graph neural networks, work by letting every node look at its neighbours, then at its neighbours' neighbours, and so on. Looking too far ends up blurring distant nodes together, so a common remedy is to train the model on a random subset of the links and discard the rest, in the hope of stopping it from overrelying on any single part of the graph. This idea was proposed and tested only on the simpler problem in which each entity carries exactly one label. Many real problems are not like that. A single protein takes part in many biological processes at once, so a useful predictor has to assign several labels at the same time. Whether dropping links still helps in this more realistic multi-label setting has not been studied. We answer the question on synthetic graphs whose structure can be controlled exactly, ranging from strongly clustered to almost random, and on three real biological and bibliographic datasets that span the same range. In almost every case, dropping links hurts rather than helps, and the damage grows with the fraction removed, with a single weak exception on the most strongly multi-label, lowest-homophily real graph. The cause is not a flaw in the technique but a property of the multi-label task: when many labels must be predicted from the same node representation, each label gets only a fraction of the learning signal it would receive if it were the only one, and discarding links reduces that already-thin signal further.

1 Introduction

Many real-world systems can be naturally described as graphs whose nodes are the entities and whose edges record their relationships. A protein interaction network is one such example, where each node is a protein and each edge a measured interaction between them. A recurring task on such graphs is node classification, where the labels of unlabeled nodes are inferred from a small labeled subset, and graph neural networks (GNNs) address it by passing messages along edges so that the prediction for each node depends on its neighbourhood [Kipf and Welling, 2017]. Most research assumes the single-label setting, in which every node belongs to exactly one class, yet many important problems are multi-label, where a node belongs to several classes at once. A protein, for instance, is annotated with many Gene Ontology functions rather than a single one. Despite its prevalence, multi-label node classification has received far less attention than its single-label counterpart [Zhao *et al.*, 2023].

The dominant obstacle to building deep GNNs is oversmoothing. Each layer updates a node by averaging it with its neighbours, so after D layers a node's representation mixes information from its entire D-hop neighbourhood. Real graphs have a small diameter, meaning few hops separate any two nodes, so only a handful of layers are needed before these neighbourhoods cover almost the whole graph and every node aggregates from essentially the same set of nodes. The node representations then become nearly identical to one another and lose the local detail a classifier needs, so accuracy degrades once the network grows beyond a few layers [Li *et al.*, 2018; Oono and Suzuki, 2020]. The standard remedy is edge dropping, which removes a random fraction of edges at each training step so that every forward pass sees a sparser subgraph instead of the full graph. DropEdge removes edges uniformly at random [Rong *et al.*, 2020], whereas TADropEdge biases removal toward structurally redundant edges and protects the few edges that bridge otherwise separate regions of the graph, which it identifies from the graph's global connectivity structure [Gao *et al.*, 2021].

Both methods were justified and validated only in the single-label setting. It is far from obvious that this justification carries over to multi-label graphs, for two reasons. First, the single-label argument for edge dropping depends on a binary view of edges. An edge is homophilic when it joins two nodes of the same class and heterophilic otherwise, the former carrying reinforcing signal and the latter injecting noise. Therefore, edge-dropping trades signal for a reduction in noise. This view dissolves once each node carries several labels, because the same edge can be homophilic for one label and heterophilic for another. Homophily then becomes a continuous quantity in $[0, 1]$ rather than a binary attribute [Zhao *et al.*, 2023]. Moreover, since a GNN maintains a single shared representation per node, dropping one edge perturbs the prediction for every label simultaneously. Second, the two settings may differ in their fitting regime. Standard GNNs have been reported to underfit multi-label benchmarks even when labeled data is plentiful [Zhao and Khosla, 2024]. Edge dropping is by construction a regulariser addressing overfitting, and a regulariser applied to a model that is already underfitting can only remove information the model needs. Both observations suggest that edge dropping may hurt rather than help in the multi-label regime.

These considerations lead to our central question: *How does the choice of edge-dropping strategy interact with the label homophily of the graph in multi-label node classification?* We answer it in two phases. In the first phase we run a controlled experiment on synthetic graphs from the MLGNC generator [Zhao *et al.*, 2023] that span low, medium and high homophily with different label counts, sweeping the depth of the GNN architecture and the drop rate of the two edge-dropping methods. In the second phase we test whether the resulting trends transfer to three real datasets that span low, medium and high homophily. This leads us to the following contributions:

- We show that edge dropping does not improve on a no-drop baseline in this setting, and that the performance loss grows monotonically with the drop rate, while being roughly constant across depth. In particular, the depth-rescue that motivates DropEdge in the single label case does not appear in the multi-label case.

- We find that which of the two methods is less harmful depends on homophily. In particular, TADropEdge is the harsher of the two at medium homophily and the milder at high homophily. The reason is that TADropEdge decides which edges to keep from graph structure alone, ignoring labels. Only on a highly homophilic graph does keeping the structurally important edges also keep the label-informative ones, while at medium homophily the two come apart.

2 Background

This section sets up the concepts the rest of the paper builds on. We first state the node-classification problem and the graph convolutional network used to solve it, then define homophily and over-smoothing, and finally describe the two edge-dropping methods we compare.

2.1 Node classification

We consider an undirected graph $G = (V, E)$ with $|V|$ nodes. The rows of a feature matrix $\mathbf{X} \in \mathbb{R}^{|V| \times d}$ hold the node features, and the rows of a label matrix $\mathbf{Y} \in \{0, 1\}^{|V| \times L}$ hold the node labels, where L is the number of labels and $y_i^{(\ell)} = 1$ indicates that node i carries label ℓ. We write $\mathcal{N}(i)$ for the set of neighbors of node i.

We study node classification in the semi-supervised, transductive setting: the labels are known for only a small subset of the nodes, and the task is to predict the labels of the remaining nodes from the graph structure and the node features, with all nodes belonging to a single graph. In the single-label setting each node belongs to exactly one of the L classes; in the multi-label setting studied here each node may carry any subset of the L labels at once, so the prediction for node i is a binary vector in $\{0, 1\}^L$ rather than a single class.

A graph convolutional network (GCN) solves this by building a vector representation for each node and refining it layer by layer. Writing $\mathbf{h}_i^{(l)}$ for the representation of node i at layer l and d_i for its degree, the propagation rule is

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\sum_{j \in \mathcal{N}(i) \cup \{i\}} \frac{1}{\sqrt{d_i\, d_j}} \mathbf{W}^{(l)} \mathbf{h}_j^{(l)}\right), \qquad (1)$$

where $\mathbf{W}^{(l)}$ is a trainable weight matrix and σ a non-linearity, taken to be ReLU in the hidden layers and a sigmoid in the output layer. The output layer produces one score per label.

Whether a GNN benefits from an edge depends on whether that edge joins nodes with related labels, a property captured by homophily. In the single-label setting, label homophily is the fraction of edges that connect two nodes of the same class. It carries a direct interpretation, since a homophilic edge reinforces the shared class while a heterophilic edge introduces a competing one, and this interpretation underlies the usual argument that removing edges removes noise. This no longer holds in the multi-label setting when each node carries several labels, because the same edge can agree with its endpoints on some labels and disagree on others.

Multi-label homophily generalises the single-label homophily by averaging, over all edges, the overlap between the label sets of the two endpoints [Zhao *et al.*, 2023],

$$h = \frac{1}{|E|} \sum_{(i,j) \in E} \frac{|Y_i \cap Y_j|}{|Y_i \cup Y_j|}, \qquad (2)$$

where $Y_i = \{\ell : y_i^{(\ell)} = 1\}$ is the set of labels active at node i. This makes the value of h a continuous quantity in $[0, 1]$, with a value near one meaning that connected nodes share almost all of their labels, and a value near zero meaning that they share almost none.

The depth of the GCN comes at a cost. After D layers, a node's representation depends on its entire D-hop neighbourhood, so depth controls how far information propagates through the graph. Repeated normalised aggregation acts as a low-pass filter that pulls node representations towards the dominant eigenvector of the propagation operator, so they lose the local detail that distinguishes one node from another [Li *et al.*, 2018; Oono and Suzuki, 2020]. Once the representations have collapsed in this way, a classifier built on top of them can no longer separate the nodes and accuracy falls. This effect, known as over-smoothing, is the failure mode that edge dropping is designed to counter.

However, edge dropping was developed and justified only for single-label graphs, and two features of the multi-label setting put that justification in doubt. First, homophily is no longer binary but the continuous quantity of Eq. (2), so the same edge can agree with its endpoints on some labels and disagree on others. Second, because a GCN keeps one shared representation per node, dropping an edge shifts the prediction for all L labels at once, so the value of an edge is a compromise across the labels its endpoints carry rather than a property of the edge alone. For both reasons the single-label theory of edge dropping need not carry over, and homophily is the natural axis along which to study it since h is the single number that sets how much a typical edge's endpoints agree on their labels, and hence the signal-versus-noise balance that edge dropping depends on.

2.2 Edge-dropping strategies

We compare a no-drop baseline against the two edge-dropping methods. The baseline trains on the full graph and serves as the reference against which every other method is measured.

DropEdge sparsifies the graph by retaining each edge independently with probability $1 - r$ and removing it with probability r, where r is the nominal drop rate [Rong *et al.*, 2020]. Every edge is treated identically, so the expected degree of each node is scaled by $(1 - r)$ and the model sees a different random subgraph at every training step.

TADropEdge improves on this framework by replacing uniform removal with a topologically informed one [Gao *et al.*, 2021]. Its premise is that edges differ in structural importance. Inter-cluster edges connect distinct regions of the graph and carry long-range information, whereas redundant intra-cluster edges can be removed with little consequence. Each edge (i, j) is assigned a weight

$$\omega_{ij} = \sum_{k<q} \left(\mathbf{V}_{ik} - \mathbf{V}_{jk}\right)^2, \qquad (3)$$

computed from the q lowest eigenvectors $\mathbf{V}$ of the unnormalised graph Laplacian. A high ω_{ij} marks an edge that should be preserved, and a low ω_{ij} marks a redundant edge that can be dropped. The cutoff q is selected automatically from the largest gap between consecutive eigenvalues of the Laplacian near the number of labels, so the basis adapts to the graph rather than being fixed in advance. The weights are turned into per-edge keep probabilities by one of three rules:

- The cutoff rule places a hard threshold at the median weight γ, dropping edges below it with the full probability r and never dropping edges above it.
- The division rule replaces the threshold with a smooth keep probability $1-r\,\gamma/(\gamma+\omega)$, so that protecting edges grows continuously with weight.
- The cumulative distribution function (CDF) rule sets the keep probability to $(1-r)+r\,F(\omega)$, where F is the empirical cumulative distribution function of the weights; it depends only on the rank of an edge and is therefore insensitive to the scale of the weights.

The three rules share the edge weights of Eq. (3) and differ only in how sharply they protect the heavier edges. A consequence common to all three is that the realised average drop rate stays below the nominal r, because the protected high-weight edges lower the global average.

3 Methodology

We study edge dropping in two phases. We first run a controlled experiment on synthetic graphs, in which homophily and every other factor can be varied in isolation, and then ask whether the same trends transfer to three real datasets.

3.1 Synthetic datasets

To study homophily as a controlled variable we generate synthetic graphs with the MLGNC framework [Zhao *et al.*, 2023], which sets the label homophily of a graph while holding its other properties fixed. The generator works in two stages.

In the first stage, labels and features are produced jointly with the hypersphere method of Tomas et al. [2014]. We place L sphere centres in feature space, sample each node inside one of them, and assign node i label ℓ when its feature vector falls within sphere ℓ, so that nearby features imply shared labels. Each label is then flipped independently with probability 0.20 to inject noise, and ten irrelevant feature columns are appended.

In the second stage, edges are drawn with the Social Distance Attachment sampler of Boguñá et al. [2004], which connects two nodes with probability

$$p(i,j) = \frac{1}{1+\big(\mathrm{dist}(i,j)/b\big)^{\alpha}}, \tag{4}$$

where $\mathrm{dist}(i,j)$ is the normalised Hamming distance between the label vectors of i and j, b is a characteristic distance and α controls how sharply the probability decays with label disagreement. A small α connects nodes almost regardless of their labels and yields a low-homophily graph, whereas a large α suppresses connections between dissimilar nodes and yields a high-homophily graph. For each target homophily we find the matching α by binary search, evaluating the homophily of Eq. (2) on sampled graphs at each step, so everything is calibrated rather than assumed.

Dataset	Homophily h	Nodes	Edges	Labels
DBLP	0.76 (high)	28 702	66 832	4
HumanGo	0.40 (medium)	3 106	15 978	14
PCG	0.17 (low)	3 233	37 351	15

Table 1: The three real multi-label datasets, with multi-label homophily (Eq. 2), node count, undirected edge count and number of labels. The three span the low, medium and high homophily regimes of the synthetic sweep.

Following this approach, we make graphs of $1\,000$ nodes each having homophily $h \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$ and label count $L \in \{4, 8, 16\}$, with ten seeds per cell. After generating the graphs, for each of them we train a GCN with network depth $D \in \{2, 4, 6, 8\}$ using no edge-dropping as well as DropEdge or each of the three TADropEdge variants with a drop rate $r \in \{0.1, 0.3, 0.5, 0.7\}$. This gives 10,200 training runs in total for the 17 method/rate combinations at each of the 600 (L, h, D, seed) cells. Each synthetic seed regenerates a brand-new graph (new labels, features and edges), not just a new model initialisation. So the F1 spread across seeds mostly reflects which graph was drawn, and that spread is larger than the effect of edge dropping within any single (L, h, D) condition. We therefore trust the averaged trends across conditions, not individual cells.

3.2 Real-world datasets

We then test whether the synthetic trends transfer to real data, using the three multi-label benchmarks in Table 1. They were chosen to span the synthetic homophily range, with DBLP as a high-homophily co-authorship network, HumanGo as a medium-homophily protein interaction network and PCG as a low-homophily protein–phenotype network. The graphs differ widely in size and in label cardinality too. On average, PCG is the only one that is genuinely multi-label, with on average 1.9 labels per node, whereas DBLP and HumanGo average close to a single label per node despite using a multi-label encoding.

We load each graph with its fixed on-disk train, validation and test split and average over three seeds that vary only model initialisation and training-time randomness. Because the graph and the split are identical across these seeds, real per-condition variance is tighter than synthetic by construction. The purpose of this phase is to check transfer, not to reach the best possible accuracy on any benchmark.

3.3 Model training and evaluation

All experiments share a GCN backbone. The depth study uses 4, 6 and 8 layers. The characterisation we present sweeps all four depths, with the pooled effect figure and homophily table averaging over them, while the the per-condition breakdown in the appendix fixing depth at 2. The hidden width is 256, the output layer applies a sigmoid to each label and the loss is binary cross-entropy summed over the L labels. We optimise

with Adam at learning rate 0.01 and weight decay 5×10^{-4}, and stop early on the validation F1-macro with patience 50 and at most 300 epochs. Edge dropping is applied only during training, and evaluation always uses the full graph, so the reported metrics reflect generalisation rather than the sparsified subgraph seen during a training step.

We follow the top-k protocol of Zhao et al. [2023]. For each node the model predicts exactly k labels, where k is the node's true label count, which removes the need for a decision threshold. The primary metric is F1-macro, which weights every label equally and therefore does not reward a model for fitting only the common labels. We report F1-micro, AUROC and average precision in the appendix. The effect of a method is measured as a paired difference against the baseline trained on the same graph,

$$\Delta\mathrm{F1} = \mathrm{F1}(\text{method}) - \mathrm{F1}(\text{baseline}), \tag{5}$$

evaluated at the same (L, h, D, seed). Pairing in this way cancels the variance due to the graph draw and isolates the contribution of edge dropping.

4 Results

This sections reports the synthetic experiments and draws conclusions from them, then analyzes if the real-data experiments follow the same trends found.

4.1 Synthetic experiments

We analyze the synthetic sweep by looking at what regime the untouched baseline is in, then how the loss depends on the drop rate, and finally how it varies with homophily.

Baseline regime

Before any edge is dropped we characterise the baseline GCN across the grid, since the operating regime determines whether a regulariser has anything to improve. Figure 1 reports baseline F1-macro against depth. We make two observations based on it. The first problem is over-smoothing, which appears across almost the whole grid. F1-macro is highest at shallow depth and falls as the network deepens, and the depth at which it falls depends on homophily. For example, at our lowest multi-label count $L = 4$, lowering homophily to $h = 0.6$ makes the collapse visible at 6 layers compared to 8 layers for $h = 0.8$. Raising the label count lowers F1 and speeds up the collapse, since more label heads share one embedding. The one corner where collapse does not appear is the fully homophilic graph at $L = 4$, where the model with still reaches F1-macro of over 0.95, regardless of having $2, 4, 6$ or 8 layers. Aggregation blurs nodes together by averaging neighbours that disagree on labels, and a perfectly homophilic graph has no disagreement to average in. This is not an exception to over-smoothing but a consequence of how it works: aggregation washes representations out by mixing nodes that disagree on labels, and a perfectly homophilic graph has no disagreement to mix in. Once even a small amount of heterophily is reintroduced, the collapse returns.

The second problem is that even when the graph is perfectly homophilic, the two-layer model does not fit it well, reaching only F1-macro 0.42 at $L = 16$ and $h = 1.0$. We think this is underfitting. The gradient for each label is spread across sixteen output heads that share one node embedding, so it is too weak to fit the data even when the graph offers the strongest signal. Therefore, the collapse from two to eight layers (Figure 5 in the appendix) is deepest where the graph carries the most signal to lose, reaching -0.48, and negligible where the baseline already sits at the floor. The grid therefore splits into two regions. In one the baseline still holds real signal that a regulariser could damage, and in the other it is already no better than random guessing. We take this into account when interpreting our results.

Effect of homophily

We ask whether the harm changes with homophily. Table 2 reports the mean ΔF1 at each homophily level, together with the signed advantage of TADropEdge over DropEdge in the final TA$-$DE column. At $h = 0.2$ both methods are flat at zero, because the baseline is already at the random floor and there is nothing left to remove. Above that floor both methods are harmful at every homophily level, and the only structured difference between them is in the TA$-$DE column.

Two things are worth keeping in mind here. First, the numbers are so small that at any single homophily level the gap between the two methods is smaller than the variation we get just by changing the random seed, so we do not read anything into one cell on its own. Second, even if we can't trust a single number, what we trust is the trend. The column moves steadily in one direction as homophily rises, from -0.011 at $h = 0.6$ to $+0.006$ at $h = 0.8$ and $+0.017$ at $h = 1.0$. A steady move across five homophily levels is harder to dismiss as noise than any single value.

This observed trend itself has a simple reading. The one thing TADropEdge does differently from random dropping is keep the edges that are most important to the shape of the graph. Whether that helps depends on the graph. When the graph is almost fully homophilic, the edges that hold it together are also the ones that carry the most label information, so keeping them is the right call and TADropEdge comes out ahead. In the middle of the range the two stop coinciding, because the same edge can agree with its endpoints on some labels and disagree on others, so TADropEdge ends up protecting edges that matter for the graph's shape but not for the labels, and it does slightly worse than random dropping.

h	DropEdge	TADropEdge	TA $-$ DE
0.2	0.000	0.000	0.000
0.4	-0.012	-0.014	-0.002
0.6	-0.011	-0.022	-0.011
0.8	-0.025	-0.019	$+0.006$
1.0	-0.023	-0.006	$+0.017$

Table 2: Mean ΔF1 by homophily level, pooled over label count, depth and drop rate. The final column is the signed gap between TADropEdge and DropEdge, which is negative in the mid-homophily range and turns positive only at high homophily.

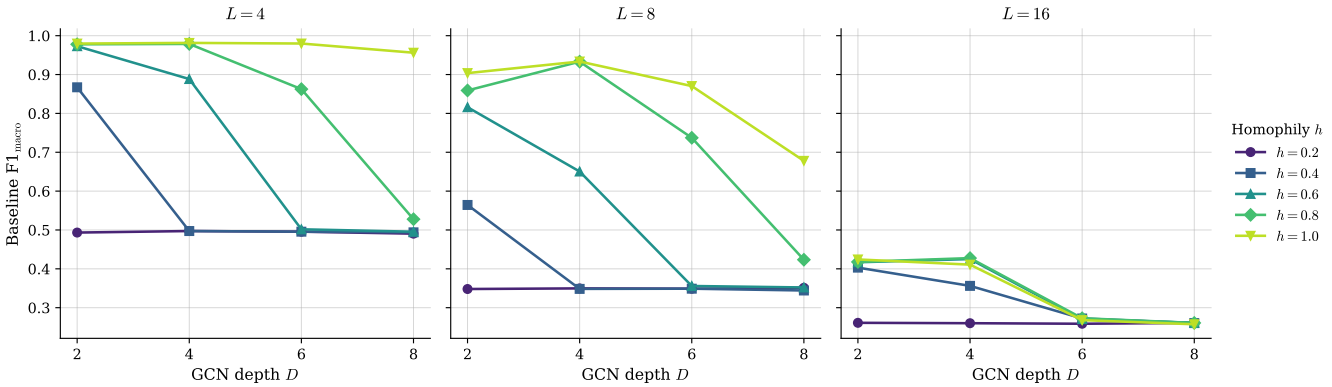

Figure 1: Baseline GCN F1-macro against network depth, with one panel per label count L and one curve per homophily level h, averaged over ten seeds. Performance decays with depth in every cell except the most homophilic ones, where a perfectly homophilic graph has no disagreement to mix in, and the depth at which it decays grows with homophily.

Effect of the drop rate

We next ask whether the failure has structure across drop-rate. Figure 2 reports the mean paired difference ΔF1 between each method and the no-drop baseline against the nominal drop rate, pooled over the whole grid. The four curves, DropEdge and the three TADropEdge rules, all lie below zero at every rate and run close together, with the TADropEdge rules clustered around DropEdge and mostly just above it. The pooled effect grows monotonically with the rate, from -0.0027 for DropEdge and -0.0017 for TADropEdge at $r = 0.1$ to -0.029 and -0.024 respectively at $r = 0.7$, and there is no rate at which dropping becomes beneficial.

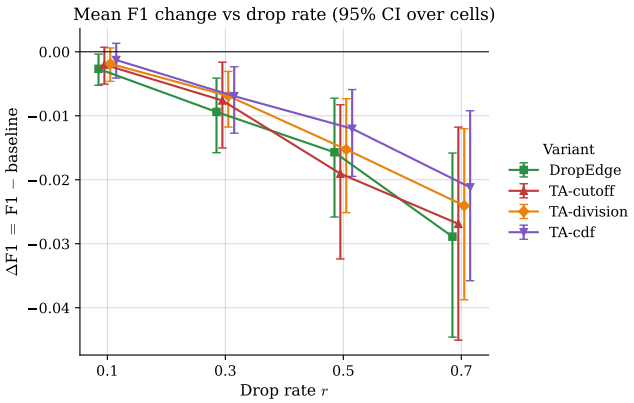

Figure 2: Mean paired difference ΔF1 between each method and the no-drop baseline against the nominal drop rate, pooled over the synthetic grid, with 95% bootstrap confidence intervals over cells. Every variant is negative at every rate.

A central claim of the original DropEdge work is that dropping edges lets deeper networks escape the over-smoothing collapse and recover accuracy that the baseline cannot reach. This does not hold here. Table 3 reports the mean ΔF1 at each depth, pooled over the other factors. The harm is essentially flat across depth, between -0.01 and -0.02 throughout, with no sign of a rescue at six or eight layers. Where the baseline has already collapsed to the random floor, the F1-macro a model reaches by predicting k labels at random, dropping edges simply arrives at the same floor. This is consistent with the underfitting reading. The depth-dependent collapse is a shortage of usable signal that a regulariser cannot replace, not an excess of capacity that it can absorb.

Depth D	DropEdge	TADropEdge
2	-0.012	-0.009
4	-0.019	-0.021
6	-0.015	-0.009
8	-0.011	-0.009

Table 3: Mean ΔF1 by network depth, pooled over label count, homophily and drop rate, with TADropEdge averaged over its three sampling rules. The harm is approximately constant across depth.

4.2 Real-world experiments

The three real datasets test whether these synthetic trends survive on graphs the generator did not produce. What we found is that they do. The baseline depth pattern transfers (Figure 6 in the appendix), with high-homophily DBLP reaching F1-macro 0.89 at four layers before collapsing to 0.31 at eight, while lower-homophily HumanGo and PCG peak at two layers and even there near 0.22 and 0.27. Edge dropping again only hurts as seen in Figure 3. On DBLP both methods drop sharply, and TADropEdge is the milder at the gentlest rate, losing 0.016 against DropEdge's 0.047, before collapsing to -0.094 at the highest rate as its few protected high-weight edges are used up. On HumanGo both methods hurt less and TADropEdge is again milder, with a pooled effect of -0.006 against -0.016. Same holds for PCG.

The homophily ordering also transfers. The high-homophily DBLP is harmed most and the low-homophily PCG least, exactly as the synthetic grid predicts. PCG in fact provides the single positive result in the study, where TADropEdge reaches a small but consistent ΔF1 of $+0.002$ that stays non-negative at every rate while DropEdge turns

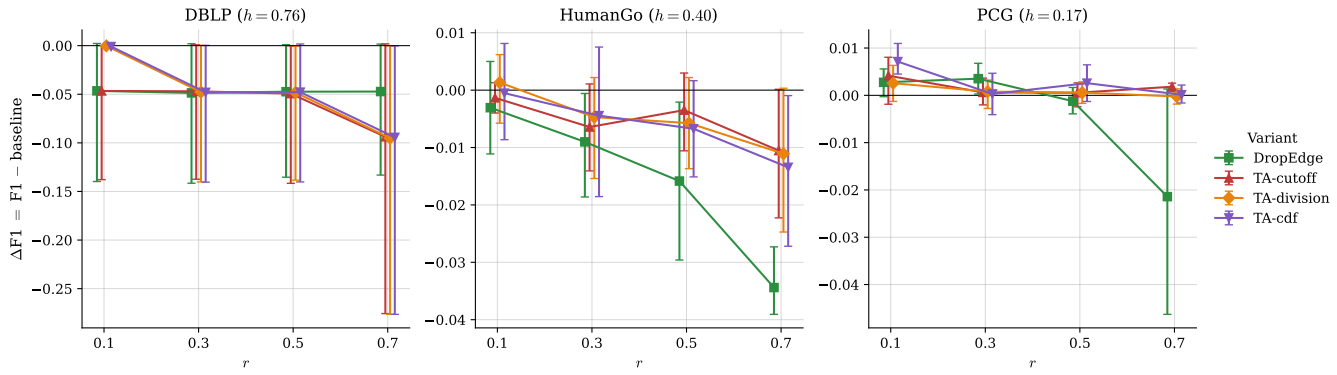

Figure 3: Mean ΔF1 against the nominal drop rate on the three real datasets, with mean and standard deviation over three seeds. Note the different vertical scale on DBLP.

harmful at the highest rate. PCG is the only genuinely multi-label graph here, with close to two labels per node, and its low homophily places it in the regime where the synthetic baseline collapses to the random floor and the paired difference is therefore uninformative. It fills that gap with a real graph whose baseline sits a finite amount above random, and there the original TADropEdge hypothesis holds weakly, that on a noisy low-homophily graph removing the least reliable edges can help. The small effect and the unusual label cardinality make this a single data point rather than a general claim.

4.3 Potential confounds

We note that the two methods are not compared on equal terms. Even though we use the same drop rate r for both, it does not mean the same thing. For DropEdge, r is the probability that any edge is dropped, so on average a fraction r of the edges is removed. For TADropEdge, r is only the highest drop probability, applied to the least important edges, while the most important edges are kept almost regardless of r. Averaged over all edges, the rules defined above remove only about $r/2$: under the cutoff rule only the less important half of the edges can be dropped at all, each with probability r, and under the rank-based CDF rule the drop probability slides from r for the least important edge down to zero for the most important. Either way, TADropEdge removes about half as many edges as DropEdge at the same r. Figure 4 confirms this gap, with DropEdge on the diagonal and the three TADropEdge rules at roughly half the nominal rate.

This is intended, not accidental. Gao et al. [2021] weight the edges precisely so that the important ones survive and the sparser training graph keeps the connectivity of the original. This matters for two reasons. First, the original paper compares the two methods at the same nominal r rather than at the same number of edges actually removed, so part of the advantage it reports for TADropEdge could come simply from TADropEdge dropping fewer edges. Second, the same confound applies to us, since removing less would on its own make TADropEdge look gentler. In our results removing fewer edges does help TADropEdge, and it is the reason it is the gentler of the two at most homophily levels. However, we also note that this cannot be the whole story. If it were, TADropEdge would always beat DropEdge, and at $h = 0.6$ it does not, where it is worse by 0.011. The change of sign across homophily therefore comes from which edges each method drops, not how many.

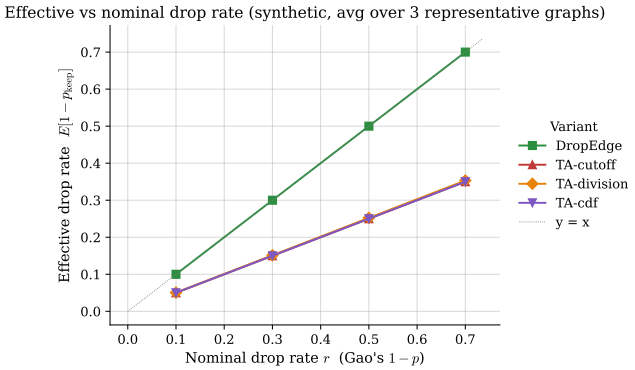

Figure 4: Effective (realised) average drop rate against the nominal drop rate on representative synthetic graphs. DropEdge follows the diagonal, while the three TADropEdge rules drop about half of the nominal fraction because they preserve high-weight edges.

The three TADropEdge rules behave almost identically. They nearly coincide in realised drop rate (Figure 4), so they remove close to the same fraction of edges, and they give the same qualitative picture at every homophily level and depth. Their aggregate harm separates only slightly, in the order set by how aggressively each exposes the low-weight edges, with mean ΔF1 of -0.014 for the hard-threshold cutoff rule, -0.012 for division, and -0.011 for the rank-based CDF rule. Because the three agree, we report TADropEdge pooled over them in the tables and treat the gentle ordering as a robustness check rather than a finding of its own.

5 Discussion

We interpret what our results mean more broadly by asking why edge dropping fails at all, why the two methods differ, and finally how homophily organises the effect. Then, we present the limitations of our work.

5.1 Interpretation

Underfitting, not overfitting

The consistent failure of edge dropping is best explained by the fitting regime of the multi-label model rather than by any flaw in the methods, which behave exactly as their specifications predict. In the single-label setting the loss is a single softmax cross-entropy, so each node produces one clean gradient directed towards its correct class. A model trained this way can overfit specific neighbourhood patterns, and reducing connectivity during training is a sensible way to regularise it. The multi-label loss is instead a sum of L binary cross-entropies applied to one shared node embedding. The L gradients compete for the same representation and often pull in different directions, so the effective signal per label is roughly $1/L$ of what it would be in the single-label case. As L grows the model moves further from saturation and closer to a regime in which the bottleneck is too little signal rather than too much capacity, and a regulariser applied to such a model can only remove information it needs.

Two predictions follow, and both hold in our results. First, more aggressive dropping should hurt more, and the dose-response of Figure 2 is monotone with no beneficial regime at low rates. Second, any method that reduces connectivity should land on the same downward trend regardless of which edges it removes, and DropEdge and TADropEdge indeed run roughly parallel, separated by a roughly constant offset. The baseline shows the same signature, since even a two-layer network on a perfectly homophilic graph reaches only F1-macro 0.42 at $L = 16$. This reading agrees with the underfitting that Zhao and Khosla [2024] identify at the architectural level, and it refines the usual account of over-smoothing [Li *et al.*, 2018; Oono and Suzuki, 2020]. The depth-dependent collapse still occurs, but its cause shifts from a convergence of representations that a regulariser can slow to a starvation of gradient that it cannot.

Why TADropEdge's gap with DropEdge changes sign?

TADropEdge differs from DropEdge in two ways: it removes only about half as many edges at a given nominal rate (Figure 4), and it chooses which edges to remove by a structural weight ω_{ij} rather than uniformly. The first difference is a roughly constant offset in its favour, but it cannot set the sign of the comparison: if dropping fewer edges were the whole story TADropEdge could never do worse than DropEdge, and in the mid-homophily range it does. The sign is set by the second difference, and ω_{ij} scores only the structural importance of an edge, not whether removing it helps the labels. In a multi-label graph these two properties decouple, since a single edge is homophilic for some labels and heterophilic for others, so in the middle of the range the criterion keeps structurally important but label-irrelevant edges and falls behind uniform sampling. Only at high homophily, where every edge reinforces its endpoints' labels, do structural importance and label-helpfulness coincide, and there the same criterion keeps the right edges and TADropEdge becomes the milder of the two.

None of this contradicts the original method, in fact it maps its behaviour onto an axis it was never tested on. Gao et al. [2021] motivated this weighting purely as connectivity preservation, keeping the edges most critical to graph topology so that augmented subgraphs stay close to the original, and computed it from the Laplacian alone, with no reference to labels or homophily. Their benchmarks are single-label graphs that all sit above $h = 0.7$, so the regime where a label-blind structural criterion comes apart from label-helpfulness, the low and middle of the homophily range, had simply never been tested.

How homophily summarizes this effect?

The real datasets confirm that homophily predicts the direction of the effect, since the high-homophily DBLP is harmed most and the low-homophily PCG least, exactly as the synthetic sweep orders them. Homophily is not a complete description of the regime, however. The single positive result, TADropEdge on PCG, coincides with both the lowest homophily and the highest label cardinality in the study, because PCG is the only genuinely multi-label graph at close to two labels per node. With three datasets we cannot separate these two explanations, and the gain may reflect the label structure as much as the homophily. Homophily is therefore a useful one-number axis for organising the multi-label regime, but not a sufficient summary of it.

5.2 Limitations and future work

Several factors bound these conclusions and point to natural next steps. All experiments use a single GCN backbone, and architectures with different aggregation [Hamilton *et al.*, 2017; Veličković *et al.*, 2018], may interact with edge dropping differently, so repeating the study under them would show how far the underfitting account extends. The MLGNC generator varies homophily while holding degree and community structure fixed, so the synthetic trends speak to homophily specifically rather than to every property that varies across real graphs. The real-data evidence rests on three datasets and three seeds, and the single positive result confounds low homophily with high label cardinality, which a wider panel of low-homophily and genuinely multi-label graphs would separate. Most directly, the underfitting explanation predicts that interventions which add per-label capacity, such as independent label heads or frequency-weighted losses, should restore a regime in which edge dropping is at worst neutral, and testing whether other regularisers such as PairNorm or DropNode meet the same ceiling would show whether the obstacle is specific to edge methods.

6 Conclusion

We investigated how the choice of edge-dropping strategy interacts with label homophily in multi-label node classification. Across a controlled synthetic sweep of $10\,200$ runs and three real datasets, the answer is that edge dropping does not help. The loss grows monotonically with the drop rate, the depth-rescue effect that motivates DropEdge does not appear, and homophily sets how much performance is lost without changing its sign, with a single weak exception at the lowest homophily. We trace this to underfitting rather than over-smoothing, since spreading the gradient across L label heads that share one embedding starves the model of signal, and a regulariser can then only remove what little signal remains.

The one structured difference between the methods is that the gap between TADropEdge and DropEdge changes sign with homophily. TADropEdge is the harsher of the two in the mid range and the milder only at high homophily, because its label-blind structural weight keeps the right edges only where structural importance and label signal coincide.

7 Responsible Research

Compute. The experiments comprise $10\,200$ synthetic and 612 real trainings, run as SLURM job arrays with one GPU, two CPU cores and 8 GB of memory per task on a shared university cluster. The total cost is approximately 60 GPU-hours, a modest footprint at this scale.

Data. The three real benchmarks come from the public ML-GNC release of Zhao et al. [2023] and contain no personally identifiable information beyond anonymised node identifiers. The synthetic graphs are produced from published algorithms [Tomas *et al.*, 2014; Boguñá *et al.*, 2004] at fixed seeds, so they can be regenerated exactly.

Use of LLMs. LLMs have been used during this research for improving the grammar, style and spelling of the paper contents and for improving the visual figures of the obtained results. Some of the most representative prompts are presented in Appendix B.

Reproducibility. Every result follows from public components and a single configuration file, with no undocumented changes between the description in this paper and the code that produced the numbers. The random seed is treated as a controlled factor, ten seeds for the synthetic experiments and three for the real ones, and the full set of completed runs is reported without any post-hoc selection of seeds or configurations. The one DBLP seed that failed to converge is disclosed in the results rather than removed.

References

[Boguñá *et al.*, 2004] Marián Boguñá, Romualdo Pastor-Satorras, Albert Díaz-Guilera, and Alex Arenas. Models of social networks based on social distance attachment. *Physical Review E*, 70(5), 2004.

[Gao *et al.*, 2021] Zheng Gao, Souvik Bhattacharya, Limin Zhang, Rick S. Blum, Alejandro Ribeiro, and Brian M. Sadler. Training robust graph neural networks with topology adaptive edge dropping. *arXiv preprint arXiv:2106.02892*, 2021.

[Hamilton *et al.*, 2017] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*, 2017.

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

[Li *et al.*, 2018] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. *AAAI Conference on Artificial Intelligence*, 2018.

[Oono and Suzuki, 2020] Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. *International Conference on Learning Representations*, 2020.

[Rong *et al.*, 2020] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. *International Conference on Learning Representations*, 2020.

[Tomas *et al.*, 2014] Jimena Tomas, Francisco Charte, Antonio Rivera, and María J. del Jesus. Using hypersphere information to generate synthetic multi-label data. *International Conference on Hybrid Artificial Intelligence Systems*, 2014.

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

[Zhao and Khosla, 2024] Tianqi Zhao and Megha Khosla. Gnn-multifix: Addressing the pitfalls for gnns for multi-label node classification. *arXiv preprint arXiv:2411.14094*, 2024.

[Zhao *et al.*, 2023] Tianqi Zhao, Nhat Tran Dong, Alan Hanjalic, and Megha Khosla. Multi-label node classification on graph-structured data. *Transactions on Machine Learning Research*, 2023.

A Hyperparameters

Table 4 lists the model, training and synthetic-generator settings used throughout. They are held fixed across the study except for the swept factors at the bottom of the table.

Setting	Value
Backbone	GCN
Hidden width	256
Depth (layers)	2 default; $4, 6, 8$ for the depth study
Dropout	0.5
Output / loss	sigmoid / binary cross-entropy
Optimiser	Adam
Learning rate	0.01
Weight decay	5×10^{-4}
Max epochs / patience	300 / 50
Train / val / test	0.6 / 0.2 / 0.2
Nodes (synthetic)	1 000
Feature dim / irrelevant	10 / 10
Label noise	0.20
SDA distance b	0.03
Homophily h	$\{0.2, 0.4, 0.6, 0.8, 1.0\}$
Label count L	$\{4, 8, 16\}$
Drop rate r	$\{0.1, 0.3, 0.5, 0.7\}$
Seeds	10 synthetic, 3 real

Table 4: Hyperparameters and sweep grid.

B Representative LLM prompts

Some of the most representative prompts are presented in Table 5.

Scope	Prompt
Grammar, style and spelling	"Rewrite this paragraph more concisely in the style of a top machine-learning conference, without changing the claims or the numbers." "Check this section for grammar, spelling and consistent notation."
Figures and tables	"Make each figure caption self-contained and descriptive rather than interpretive." "Refactor this LaTeX table to remove the vertical rules and use booktabs."

Table 5: Representative LLM prompts used in preparing this paper.

C Additional results

This section collects the baseline characterisation referenced from the main text, the secondary metrics and a per-condition breakdown of the main effect.

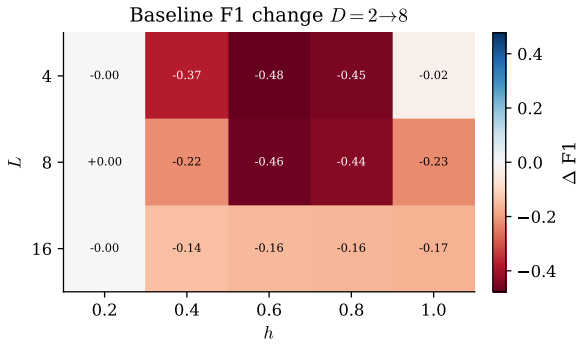

Figure 5: Change in baseline F1-macro from two to eight layers across the label-count and homophily grid. Negative values indicate a depth-induced collapse, which is largest at low label count and middle-to-high homophily.

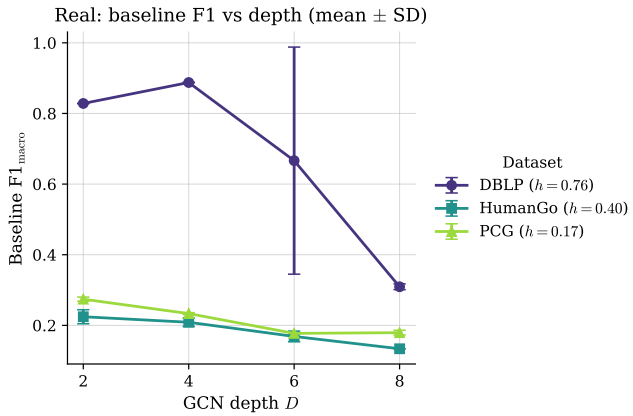

Figure 6: Baseline F1-macro against network depth on the three real datasets, with mean and standard deviation over three seeds. The homophily of each dataset is given in the legend.

Metric	Method	r=0.1	0.3	0.5	0.7
F1-micro	DropEdge	−0.002	−0.007	−0.011	−0.021
	TADropEdge	−0.002	−0.005	−0.011	−0.017
AUROC	DropEdge	−0.000	−0.005	−0.009	−0.017
	TADropEdge	−0.001	−0.005	−0.009	−0.016
AP	DropEdge	+0.000	−0.006	−0.010	−0.020
	TADropEdge	−0.001	−0.005	−0.011	−0.017

Table 6: Mean change in the secondary metrics relative to the no-drop baseline, by drop rate and pooled over the grid. F1-micro, AUROC and average precision all decline monotonically with the rate, matching the F1-macro result.

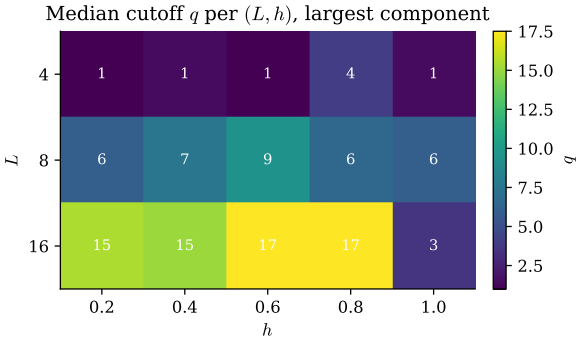

Figure 7: Median cutoff q used by TADropEdge per (L, h) cell. The basis adapts to the graph and grows with the label count rather than being fixed in advance.

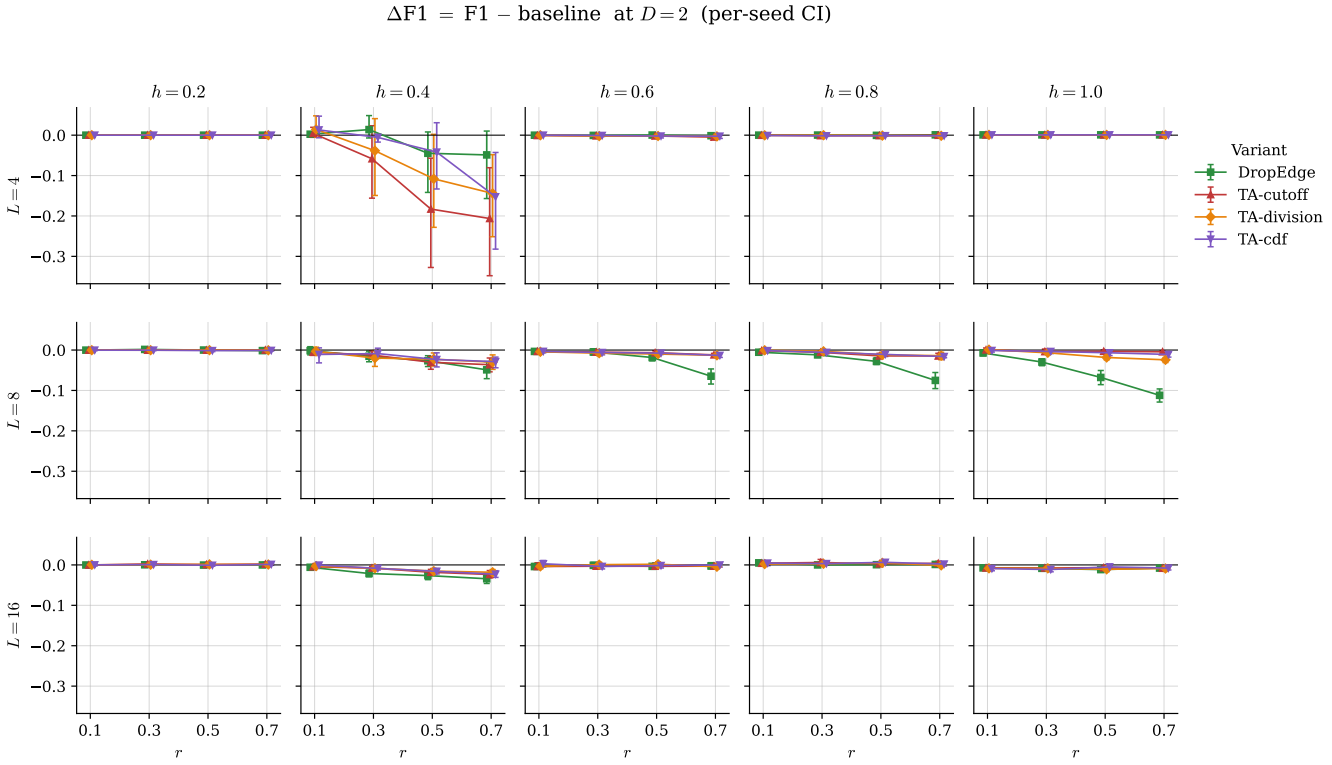

Figure 8: ΔF1 against drop rate at depth two, broken down by label count L (rows) and homophily h (columns), with per-seed confidence intervals. The pooled trend of the main text holds across individual conditions.