COUNTERFACTUAL SUPERVISION FOR ADAPTIVE PROPAGATION CONTROLS IN GRAPH NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph neural networks (GNNs) propagate information between neighboring nodes, yet the most effective propagation behavior can differ across nodes and layers. Adaptive GNNs seek to address this variation through learned controls. However, standard end-to-end training supervises these controls through gradients evaluated at their current values, thereby capturing only local loss sensitivity and failing to directly compare the downstream predictions resulting from distinct propagation behaviors. This shortcoming weakens the training signal and complicates the understanding of the learned controls. To address this issue, we introduce a method known as *Counterfactual Supervision*. This training strategy utilizes node-local continuous controls with two meaningful alternatives. For a selected control, we temporarily set it to each alternative, recompute the subsequent layers, and utilize the difference in prediction loss as an additional supervision signal. This approach complements end-to-end optimization while maintaining continuous controls. Evaluating both alternatives at every node can be computationally costly. Therefore, we show that alternatives at sufficiently distant nodes can be assessed together without compromising the exact loss differences for each node. This strategy enables the derivation of multiple per-node comparisons from a single pair of recomputations. We implement this framework in *TriAxis-GNN*, which learns distinct controls for nodes and layers related to propagation direction, balancing contributions from self and neighbors, and distinguishing between ordinary and higher-order neighborhood structures. Experiments on homophilic and heterophilic node classification benchmarks indicate that counterfactual supervision enhances adaptive propagation models. Furthermore, the proposed batching strategy considerably reduces computational costs associated with this method. Our source code is available at: `https://anonymous.4open.science/r/TriAxis-GNN-7BE3/`.

1 INTRODUCTION

Graph neural networks (GNNs) learn from graph-structured data through message passing, where each node repeatedly combines its features with information from its neighboring nodes (Kipf and Welling, 2017; Gilmer et al., 2017). Traditional GNNs implement a uniform propagation rule across all nodes (Kipf and Welling, 2017; Hamilton et al., 2017), even though there is considerable variation in their structural attributes and feature relationships (Zhu et al., 2020; Luan et al., 2022; Hevapathige et al., 2026). However, different nodes may require tailored information-propagation strategies to account for their diverse characteristics. Further, propagation requirements can vary from layer to layer within the network, making a single, universal propagation rule overly restrictive.

Node- and layer-wise controls effectively customize information propagation based on each node's characteristics (Luan et al., 2022; Song et al., 2023; Yue and Zhang, 2026). Most adaptive GNNs learn these controls through end-to-end objective functions (Luan et al., 2022; Han et al., 2024; Zeng et al., 2024). Backpropagation trains each control using the loss gradient at its current value. This gradient describes behavior near the current setting, but it may not indicate which of two distinct propagation regimes produces a lower downstream loss, particularly when the loss is nonlinear with respect to the control. Direct interventions provide a natural way to make this comparison. Previous work has used interventions to guide graph attention (Wang et al., 2023b) and routing decisions (Yoon et al., 2026). However, finite interventions have not been used to directly supervise continuous, node-local propagation controls with meaningful endpoints.

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.

In this work, we introduce *counterfactual (CF) supervision*, a training strategy for continuous, node-local controls with two semantically meaningful endpoints. For a selected node, layer, and control, we alternately fix the control at each endpoint. The other controls at that layer retain their current values, while all subsequent representations and controls are recalculated. The difference between the two resulting losses quantifies the endpoint's utility, where the sign indicates the preferred endpoint, and the magnitude measures the strength of that preference. Because both losses come from complete downstream computations, the utility directly reflects the intervention's effect on the model's prediction. We use this finite contrast to guide the adaptive controller while maintaining continuous controls and standard task-loss optimization. Thus, counterfactual supervision enhances regular task optimization instead of replacing it. Unlike CSA-GNN (Wang et al., 2023b), which uses interventions to guide graph attention, our framework supervises node-local propagation controls through receiver-specific endpoint utilities obtained from the fully recomputed downstream network.

Evaluating the two endpoint evaluations separately for every node, layer, and control would be computationally expensive. We reduce this cost by adapting the non-interference principle from graph-coloring algorithms (Gebremedhin et al., 2005). For a fixed layer and control, we select nodes that are sufficiently far apart so that an intervention at one node cannot change the receiver loss at another selected node. We then clamp all selected controls jointly to endpoint zero and endpoint one. The resulting pair of suffix runs recovers exactly the same node-specific utilities as separate interventions, allowing multiple counterfactual utilities to be computed together.

We instantiate our framework in TRIAXIS-GNN, which targets three limitations of standard message passing: a fixed smoothing relation can be harmful under heterophily (Gong et al., 2026; Zheng et al., 2026), excessive exposure to neighborhood information can lead to oversmoothing (Zhang et al., 2023; Rusch et al., 2023), and propagation over ordinary edges alone may miss useful higher-order structure (Bouritsas et al., 2022). TriAxis-GNN addresses these limitations through three node- and layer-specific controls. In summary, we make four key contributions: **(1.)** We introduce finite-intervention supervision for continuous node-local propagation controls using receiver-specific downstream endpoint utilities. **(2.)** We establish an exact independent-set batching result that recovers multiple individual utilities from a single pair of suffix runs. **(3.)** We develop TriAxis-GNN as a structured implementation that features separate controls over propagation relation, exposure, and geometry. **(4.)** Our experiments on homophilic and heterophilic node classification benchmarks show that the proposed counterfactual supervision enhances adaptive propagation models, while our batching strategy keeps the additional supervision computationally efficient in practice.

2 PRELIMINARIES

Problem Setup We consider semi-supervised node classification on an undirected graph $G = (\mathcal{V}, \mathcal{E})$ with $n = |\mathcal{V}|$ nodes, features $X \in \mathbb{R}^{n \times d_0}$, and labels $y_i \in \{1, \dots, C\}$ given for a subset $\mathcal{V}_{\text{tr}} \subseteq \mathcal{V}$. A K-layer GNN starts with $H^{(0)} = X$ and computes

$$H^{(\ell+1)} = F_{\ell,\theta}\big(H^{(\ell)}, G; c^{(\ell)}\big) \in \mathbb{R}^{n \times d}, \qquad \ell = 0, \dots, K-1,$$

where d is the hidden dimension, $c^{(\ell)}$ holds the adaptive propagation controls. Here θ and ϕ are the GNN and controller parameters, respectively. The training minimizes

$$\mathcal{L}_{\text{task}}(\theta, \phi) = \frac{1}{|\mathcal{V}_{\text{tr}}|} \sum_{i \in \mathcal{V}_{\text{tr}}} \ell_{\text{cls}}\big(f_{\theta,\phi}(G)_i, y_i\big),$$

where ℓ_{cls} is the standard cross-entropy loss.

Continuous Propagation Controls Let $\mathcal{A}$ be the control set. For node i, layer ℓ, and $a \in \mathcal{A}$, the node-local control $c^a_{i\ell} \in (0, 1)$ smoothly interpolates between two fixed propagation endpoints, indexed by 0 and 1. Changing $c^a_{i\ell}$ directly affects only node i at layer ℓ. The controller computes

$$z^a_{i\ell} = g^a_\phi\big(r^{(\ell)}_i\big), \qquad c^a_{i\ell} = \text{sigmoid}\left(z^a_{i\ell}/T\right),$$

where $r^{(\ell)}_i \in \mathbb{R}^{d_r}$ is a local descriptor of dimension d_r, g^a_ϕ is the controller for axis a parameterized by ϕ, and $T > 0$ sets the decision sharpness temperature. This form covers any continuous node-local control with two fixed endpoints. Section 4 provides the three controls used by TRIAXIS-GNN.

3 COUNTERFACTUAL SUPERVISION

In our framework, we supervise each control by contrasting its endpoints under the complete downstream network. We start by defining the utility of the endpoints.

Definition 1 (Counterfactual endpoint utility)**.** *Fix the parameters and a decision (i, ℓ, a). Let $f_{\theta,\phi}^{(\ell,a,b,i)}$ denote the network with only $c_{i\ell}^a$ clamped to $b \in \{0, 1\}$. All remaining controls at layer ℓ retain their deployed values, and all subsequent states and controllers are recomputed. Under a shared stochastic realization ξ, define*

$$u_{i\ell}^a(\xi) = \ell_{\text{cls}}\Big(f_{\theta,\phi}^{(\ell,a,0,i)}(G,\xi)_i, y_i\Big) - \ell_{\text{cls}}\Big(f_{\theta,\phi}^{(\ell,a,1,i)}(G,\xi)_i, y_i\Big).$$

The sign of $u_{i\ell}^a > 0$ indicates a preference for a given endpoint, and $|u_{i\ell}^a|$ quantifies the strength of that preference.

We omit the variable ξ from our notation and treat utility as a fixed target, meaning that no gradients will flow through it. Both runs for the endpoints utilize the same model and the same sources of randomness, including dropout masks and augmentations. The only difference between the two runs is the clamped control value, which allows the difference in loss to be exact rather than noisy.

Magnitude-aware soft preference loss. For a sampled set Ω of decisions $q = (i, \ell, a)$, write $u_q = u_{i\ell}^a$. Let $\text{stopgrad}(x)$ return x while blocking its gradient, and let $\epsilon_{\text{num}} > 0$ be a small constant that prevents division by zero. We define

$$\tau = \text{stopgrad}(\text{median}_{q\in\Omega}\,|u_q|) + \epsilon_{\text{num}}, \qquad \widetilde{y}_q^{\text{CF}} = \frac{1}{2}\left(1 + \frac{u_q}{\tau + |u_q|}\right).$$

The target lies in $(0, 1)$. Its side of $1/2$ tells us which endpoint is preferred, and its distance from $1/2$ reflects how strong the preference is relative to the batch scale. Write $z_q = z_{i\ell}^a$ and $c_q = \text{sigmoid}(z_q/T)$. We optimize

$$\mathcal{L}_{\text{CF}}(\phi) = \frac{1}{|\Omega|}\sum_{q\in\Omega} \text{BCE}\big(\text{stopgrad}(\widetilde{y}_q^{\text{CF}}), c_q\big), \qquad \text{BCE}(y,c) = -y\log c - (1-y)\log(1-c). \quad (1)$$

where BCE denotes binary cross-entropy, applied here to a soft target in $(0, 1)$.

Finite contrast versus local derivative. For $q = (i, \ell, a)$, let $\mathcal{J}_q(c)$ represent the receiver loss when only $c_{i\ell}^a$ is set to $c \in [0, 1]$ and the entire suffix is recomputed. Then, we define $u_q = \mathcal{J}_q(0) - \mathcal{J}_q(1)$. The following proposition presents the theoretical foundation for counterfactual supervision. It illustrates that the endpoint contrast and the local derivative offer two distinct perspectives on the same loss curve, and that they may yield differing signs. The proof is in Appendix A.

Proposition 1 (Finite contrast and decision-level derivative)**.** *If $\mathcal{J}_q$ is continuously differentiable on $[0, 1]$, then $u_q = -\int_0^1 \mathcal{J}_q'(t)\,dt$. If $\mathcal{J}_q'$ is additionally L_q-Lipschitz, then, for every $c_q \in [0, 1]$,*

$$\big|u_q + \mathcal{J}_q'(c_q)\big| \le \frac{L_q}{2}\left(c_q^2 + (1-c_q)^2\right). \quad (2)$$

Consequently, if $|u_q|$ exceeds the right-hand side, then $\text{sign}(u_q) = -\text{sign}(\mathcal{J}_q'(c_q))$.

Figure 1 depicts the distinction formalized in Proposition 1: local and finite signals can disagree in sign (panels a, and b), and the best finite control value need not be an endpoint (panel c).

To clarify what the counterfactual loss optimizes, we analyze its population behavior, revealing that the optimal controller output is the conditional expectation of the soft endpoint preference.

Population interpretation. Let (U, R) denote the utility and descriptor obtained by sampling a decision from the target decision population. For fixed $\tau > 0$, define

$$\widetilde{Y} = \frac{1}{2}\left(1 + \frac{U}{\tau + |U|}\right), \qquad \mathcal{R}_{\text{CF}}(c) = \mathbb{E}\Big[\text{BCE}(\widetilde{Y}, c(R))\Big],$$

where $c(R)$ is the controller output.

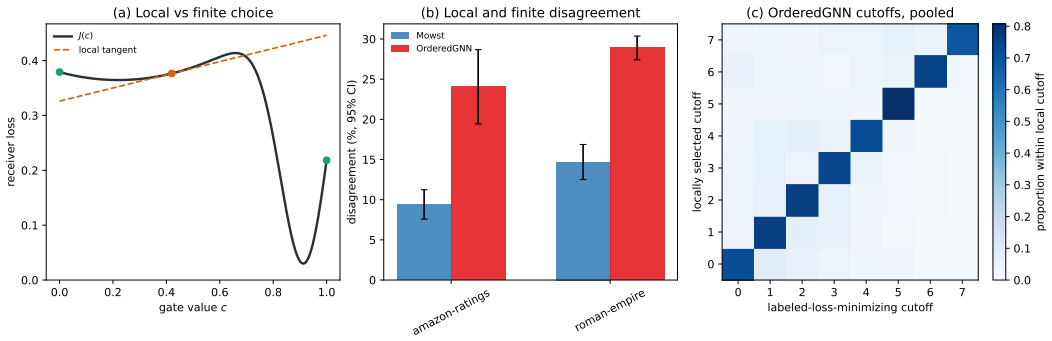

Figure 1: **Local and finite propagation preferences. (a)** A loss curve where the local derivative and the endpoint comparison favor different choices, as analyzed in Proposition 1. **(b)** Disagreement between local and finite signals for training receivers in Mowst and OrderedGNN on Amazon-Ratings and Roman-Empire, showing the phenomenon arises in practice. Bars show means and 95% confidence intervals over ten seeds. **(c)** OrderedGNN cutoff frequencies pooled across layers, seeds, and datasets, illustrating that the best finite choice need not lie at an endpoint.

Proposition 2 (Population optimum of soft preference supervision)**.** *For every descriptor value r, the pointwise minimizer is*

$$c^*(r) = \mathbb{E}[\widetilde{Y} \mid R = r] = \frac{1}{2}\left(1 + \mathbb{E}\left[\frac{U}{\tau + |U|}\,\middle|\, R = r\right]\right). \tag{3}$$

Thus $c^*(r) \geq 1/2$ if and only if $\mathbb{E}[U/(\tau + |U|) \mid R = r] \geq 0$.

Proposition 2 demonstrates that the population-optimal controller output corresponds to the conditional mean of the soft endpoint preference, thereby allowing the learned control to have a calibrated preference interpretation. The proof is provided in Appendix A.

During a CF update, only ϕ is optimized. The gradients are halted for the variables u_q, $\widetilde{y}_q^{\mathrm{CF}}$, and the descriptor inputs, while they are allowed to flow through the controller output c_q. Let $g_{\mathrm{task}} = \nabla_\phi \mathcal{L}_{\mathrm{task}}$ and $g_{\mathrm{CF}} = \nabla_\phi \mathcal{L}_{\mathrm{CF}}$. A joint update step of $-\eta(g_{\mathrm{task}} + \lambda_{\mathrm{CF}} g_{\mathrm{CF}})$, where $\eta > 0$ is the step size and $\lambda_{\mathrm{CF}} \geq 0$ is the CF weight, would reduce the task loss to first order whenever

$$\|g_{\mathrm{task}}\|^2 + \lambda_{\mathrm{CF}} \langle g_{\mathrm{task}}, g_{\mathrm{CF}} \rangle > 0,$$

a condition that need not hold at every step since g_{CF} can point away from g_{task}. We therefore do not rely on a joint step. Instead, a standard task-loss update on all parameters is followed periodically by a separate update on $\mathcal{L}_{\mathrm{CF}}$ restricted to ϕ alone, and validation-based early stopping discards any point in training where CF updates have degraded task performance.

4 TriAxis-GNN

In this section, we propose TriAxis-GNN, an adaptive GNN that uses factorized controls over multiple axes, namely, relation type, propagation exposure, and neighborhood geometry.

Propagation channels. Let A be the binary adjacency matrix of a simple undirected graph without self-loops. Define

$$(A_\triangle)_{ij} = A_{ij}(A^2)_{ij}, \qquad P = D^{-1}A, \qquad P_\triangle = D_\triangle^{-1} A_\triangle,$$

where $A_\triangle$ weights each edge by its number of common neighbors. The diagonal matrices D and $D_\triangle$ hold the row sums of A and $A_\triangle$, and $d_i = D_{ii}$ is the degree of node i. Inverse degrees are set to zero on zero-degree rows. Let $\mathcal{E}_\triangle$ be the support edge set of $A_\triangle$. Since $\mathrm{supp}(A_\triangle) \subseteq \mathrm{supp}(A)$, triangle propagation does not enlarge the receptive field. At layer ℓ, define

$$E = H^{(\ell)} W_E^{(\ell)}, \qquad N = P H^{(\ell)} W_N^{(\ell)}, \qquad M = P_\triangle H^{(\ell)} W_N^{(\ell)}.$$

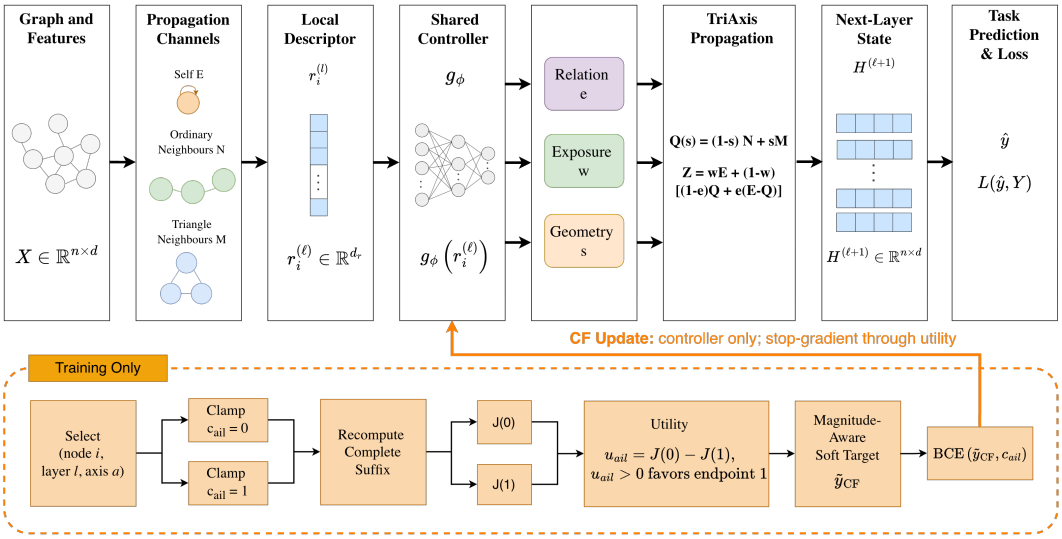

Figure 2: Overview of TriAxis-GNN with counterfactual supervision. A shared controller produces node- and layer-specific relation, exposure, and geometry controls. During training, each control is clamped to its endpoints, and the downstream loss difference informs updates to the controller.

Here $H^{(\ell)}$ is the layer input, and $W_E^{(\ell)}$ and $W_N^{(\ell)}$ are learned transformations. The shared neighbor transformation isolates the effect of geometry. Let $\chi_i = \mathbf{1}[(D_\triangle)_{ii} > 0]$. If $\chi_i = 0$, geometry is fixed to the ordinary channel and receives no CF supervision.

Factorized controls. Dropping node and layer indices, let $Q(s) = (1-s)N + sM$. TriAxis computes

$$\begin{aligned} Z(e,w,s) &= wE + (1-w)\big[(1-e)Q(s) + e(E - Q(s))\big] \\ &= E + (1-w)\big[(1-e)(Q(s) - E) - eQ(s)\big], \\ H_i^{(\ell+1)} &= \rho_\ell\Big(Z_i^{(\ell)}\Big). \end{aligned} \tag{4}$$

Here ρ_ℓ is the layer activation. The geometry control s selects between ordinary and triangle-informed neighborhoods. The relation control e selects smoothing through $Q(s) - E$, or self-referenced contrast through $-Q(s)$. The exposure control $1 - w$ determines the strength of the selected update, with $w = 1$ preserving only E and $w = 0$ applying the update fully. Thus, s selects the neighborhood, e selects the interaction type, and w controls its strength.

Decision descriptor. The descriptor is designed to answer three questions the controller needs: what each channel contains, how similar the channels are, and what the local graph structure looks like. We concatenate the corresponding quantities:

$$r_i^{(\ell)} = \big[\underbrace{\Pi E_i \| \Pi N_i \| \Pi M_i}_{\text{channel content}} \| \underbrace{\cos(E_i, N_i) \| \cos(E_i, M_i)}_{\text{channel alignment}} \| \underbrace{\log(1+d_i) \| C_i \| \chi_i \| q_\ell}_{\text{local structure}}\big],$$

where Π is a shared projection, C_i is local clustering, and q_ℓ is a layer embedding. A shared MLP with three heads generates the three controls. Figure 2 illustrates the TriAxis-GNN architecture.

5 Supervision by Independent-Set Batching

Counterfactual supervision requires evaluating both endpoints for every decision, and individual evaluation is costly because each endpoint needs a separate suffix recomputation. Under finite-radius downstream computation, two interventions interact only if their influence balls overlap. Distant nodes can therefore be clamped together, and their utilities can be recovered exactly from a single pair of suffix runs. This section formalizes this separation condition, establishes exactness for receiver and aggregate utilities, and describes the resulting training procedure. The proofs are in Appendix A.

An intervention at layer ℓ changes the state $H_i^{(\ell+1)}$ of node i. Its effect then spreads through the remaining layers. If each later layer t has receptive-field radius r_t, this spread reaches at most $R_\ell = \sum_{t=\ell+1}^{K-1} r_t$ hops downstream.

Assumption 1 (Local downstream computation)**.** *Each node loss depends only on its own final state. Every later layer has finite radius r_t, uses edges of G, and contains only node-local controllers, activations, and normalization.*

We identify that this assumption rules out operations that couple all nodes at once, such as graph-wide normalization, global attention, virtual nodes, and graph pooling. It allows residual connections, node-local normalization, and TriAxis, since triangle propagation stays within the same one-hop support as ordinary propagation. Running statistics, such as batch normalization, are held fixed during the two endpoint evaluations so that both runs execute the same model.

Affected-node aggregate utility. The receiver utility only measures the loss at node i. But an intervention can also change the losses of other nodes within its reach. Define the influence ball $B_{R_\ell}(i) = \{v \in \mathcal{V} : \mathrm{dist}_G(i, v) \le R_\ell\}$, where dist_G denotes shortest path length in G. The aggregate utility sums the loss changes over all labeled nodes in this ball:

$$u_{i\ell}^{a,\mathrm{agg}} = \frac{1}{|\mathcal{V}_{\mathrm{tr}}|} \sum_{v \in \mathcal{V}_{\mathrm{tr}} \cap B_{R_\ell}(i)} \left[\ell_{\mathrm{cls}}\left(f_{\theta,\phi}^{(\ell,a,0,i)}(G)_v, y_v \right) - \ell_{\mathrm{cls}}\left(f_{\theta,\phi}^{(\ell,a,1,i)}(G)_v, y_v \right) \right]. \tag{5}$$

By Assumption 1, nodes outside this ball are unaffected, so Eq. (5) is exactly the task-loss change caused by intervening on node i alone.

Proposition 3 (Exact batched interventions)**.** *Let $\mathcal{I}_\ell \subseteq \mathcal{V}_{\mathrm{tr}}$ satisfy $\mathrm{dist}_G(i, j) > R_\ell$ for all distinct $i, j \in \mathcal{I}_\ell$. For a fixed control a, jointly clamp all selected nodes to zero and then to one. Under Assumption 1, the loss difference at each selected receiver equals its individual utility $u_{i\ell}^a$ under the same stochastic realization.*

Corollary 4 (Exact batching of aggregate utilities)**.** *If $\mathrm{dist}_G(i, j) > 2R_\ell$, summing each jointly evaluated loss difference over $\mathcal{V}_{\mathrm{tr}} \cap B_{R_\ell}(i)$ recovers $u_{i\ell}^{a,\mathrm{agg}}$ exactly.*

If two intervened nodes are far enough apart, their downstream computations do not overlap, so clamping them together gives the same result as intervening on each one separately.

The stronger condition $\mathrm{dist}_G(i, j) > 2R_\ell$ aligns better with the task loss but allows fewer nodes per batch. At the final propagation layer, $R_{K-1} = 0$, so all valid labeled receivers can be batched together. We build $\mathcal{I}_\ell$ with a shuffled greedy procedure and rotate the layer, control, and node orderings across updates. For sampled training, separation is checked on the realized computation graph, so exactness holds conditional on that sample. Greedy inclusion is not uniform and may favour nodes in sparse regions. Balanced schedules or inverse probability weighting can correct this bias when inclusion probabilities are available.

Training Procedure. Training alternates ordinary task updates with periodic CF updates. Every T_{CF} steps, we pick a layer-control pair, compute representations up to that layer, and greedily build an independent set of receivers satisfying the distance rule of Proposition 3. We then evaluate both endpoints with a single pair of suffix recomputations, which gives exact per-node utilities, and update ϕ with one gradient step on $\mathcal{L}_{\mathrm{CF}}$. Because utilities are recomputed from the current parameters at every CF step, training is an alternating stop-gradient procedure. Appendix D gives the full pseudocode.

Implementation and Complexity We analyze the cost of TriAxis and counterfactual supervision relative to standard message passing. Exact triangle weights cost $O(\sum_{(i,j)\in\mathcal{E}} \min(d_i, d_j))$ expected time and $O(|\mathcal{E}|)$ memory using sorted adjacency lists, or can be approximated by wedge sampling on large graphs. With hidden and descriptor dimensions d and d_r, one TriAxis layer costs $O(nd^2 + (|\mathcal{E}| + |\mathcal{E}_\triangle|)d + ndd_r)$ with $|\mathcal{E}_\triangle| \le |\mathcal{E}|$. Counterfactual supervision runs every T_{CF} updates and has amortized overhead $(C_{<\ell} + 2C_{\ge\ell} + B_\phi)/(T_{\mathrm{CF}}C_{\mathrm{task}})$, where $C_{<\ell}$ and $C_{\ge\ell}$ are prefix and suffix forward costs, B_ϕ is the controller backward cost, and C_{task} is the cost of one task update. Prefix caching removes $C_{<\ell}$, and independent-set construction adds only bounded searches to depth R_ℓ. All of this is training-only, and inference uses a single TriAxis forward pass with no labels, stored targets, or counterfactual branches.

6 EXPERIMENTS

Experimental Setup We conduct experiments on node classification, evaluating nine datasets: the homophilous PubMed, DBLP, and Amazon Photo (Bojchevski and Günnemann, 2018; Shchur et al., 2018), and the heterophilous Film, Amazon-Ratings, Roman-Empire, Minesweeper, Tolokers, and Questions (Pei et al., 2020; Platonov et al., 2023). For scalability, we employ two large-scale datasets: ogbn-arxiv and ogbn-products (Hu et al., 2020). Our baselines include a variety of approaches. GAT (Veličković et al., 2018), GCN (Kipf and Welling, 2017), and GCNII (Chen et al., 2020) represent standard deep message-passing architectures. APPNP (Gasteiger et al., 2019) and GPRGNN (Chien et al., 2021) utilize personalized PageRank for propagation. Models designed for heterophilous graphs include H2GCN (Zhu et al., 2020), MixHop (Abu-El-Haija et al., 2019), OrderedGNN (Song et al., 2023), DirGNN (Rossi et al., 2024) and ACM-GCN (Luan et al., 2022). NODE-MOE (Han et al., 2024) and GMoE (Wang et al., 2023a) adopt mixture-of-experts approaches, while CSA-GCN (Wang et al., 2023b) provides intervention-based supervision. We pair GCN, DirGNN, and GPRGNN with TriAxis variants under task-only and counterfactual-supervision settings to assess the contribution of each component. To examine transfer beyond TriAxis, we apply the framework directly to Mowst, whose router is a free node-local continuous scalar with two expert-selection endpoints. We additionally report a grouped-edge adaptation for FAGCN: each receiver-level intervention jointly clamps its incoming edge scores and supervises their mean raw score. This FAGCN adaptation is an empirical robustness check, rather than an individual edge-control instance of Definition 1 or the exact batching result. For each dataset, we generate ten independent random 60%/20%/20% train/validation/test splits and report the mean and standard deviation of the evaluation metrics. The dataset statistics, implementation details, and model hyperparameters are provided in Appendix B.

Table 1: Node-classification performance (%). Minesweeper, Tolokers, and Questions use ROC–AUC; all other datasets use accuracy. The best result in each column, across all methods, is shaded and bolded. Within each +CF row, text color shows whether counterfactual supervision improved (blue) or regressed (red) relative to its own task-only or base counterpart directly above it.

	Homophilous			Heterophilous					
Method	PubMed	DBLP	Amazon Photo	Film	Amazon-Ratings	Roman-Empire	Minesweeper	Tolokers	Questions
GAT	87.24 ± 0.55	80.61 ± 1.21	93.87 ± 0.11	32.63 ± 1.61	42.52 ± 1.22	77.40 ± 1.53	80.12 ± 1.11	81.45 ± 0.92	65.47 ± 0.88
GCNII	88.82 ± 0.08	84.74 ± 0.65	94.38 ± 0.44	36.18 ± 0.61	48.85 ± 1.38	69.54 ± 1.06	79.16 ± 1.66	80.46 ± 0.61	73.61 ± 1.57
APPNP	83.37 ± 0.42	85.36 ± 0.40	93.74 ± 0.42	36.44 ± 1.78	43.48 ± 0.68	64.98 ± 0.44	62.62 ± 1.66	68.98 ± 0.96	65.04 ± 1.42
H2GCN	87.78 ± 0.28	86.81 ± 1.36	95.62 ± 0.30	38.85 ± 1.17	45.46 ± 0.55	77.48 ± 0.51	**91.32 ± 0.44**	79.86 ± 0.74	75.00 ± 1.02
MixHop	80.75 ± 2.29	84.27 ± 0.31	94.83 ± 0.41	32.22 ± 2.34	42.81 ± 0.90	81.78 ± 0.52	90.46 ± 0.51	74.87 ± 0.61	75.90 ± 1.10
OrderedGNN	89.79 ± 0.48	85.19 ± 0.42	95.67 ± 0.41	35.36 ± 0.97	36.78 ± 1.00	73.18 ± 1.69	88.79 ± 1.38	82.61 ± 0.61	63.04 ± 3.39
ACM-GCN	89.32 ± 0.51	83.37 ± 0.70	94.97 ± 0.29	35.40 ± 1.42	36.76 ± 0.07	82.39 ± 0.79	90.28 ± 0.68	80.22 ± 0.72	64.09 ± 2.49
CSA-GCN	87.38 ± 0.59	85.54 ± 0.33	93.27 ± 0.44	28.39 ± 1.18	37.04 ± 0.34	54.37 ± 0.46	71.27 ± 1.00	80.56 ± 3.59	64.28 ± 1.47
NODE-MOE	88.94 ± 0.59	85.93 ± 0.72	95.19 ± 0.55	40.55 ± 1.40	45.25 ± 1.43	78.78 ± 0.55	90.38 ± 0.52	81.31 ± 0.89	74.20 ± 0.73
GMoE	89.04 ± 0.69	84.63 ± 0.43	**95.91 ± 0.18**	37.12 ± 2.84	47.48 ± 1.55	76.50 ± 1.12	87.66 ± 1.49	80.50 ± 1.21	72.54 ± 1.33
FAGCN	87.53 ± 0.26	85.82 ± 0.37	93.67 ± 0.50	37.34 ± 1.47	42.63 ± 1.20	71.46 ± 0.74	85.19 ± 1.27	79.56 ± 0.77	72.22 ± 1.76
FAGCN + grouped CF†	88.67 ± 0.56	86.93 ± 1.32	94.53 ± 0.59	38.63 ± 1.62	41.54 ± 1.36	72.25 ± 0.46	88.28 ± 0.84	80.95 ± 1.20	73.21 ± 2.94
Mowst	87.41 ± 1.24	83.77 ± 3.67	92.33 ± 1.73	39.92 ± 1.33	45.60 ± 1.75	71.91 ± 0.50	75.95 ± 1.88	79.96 ± 0.41	70.74 ± 1.52
Mowst + CF	89.32 ± 0.53	86.33 ± 1.00	94.85 ± 0.60	40.65 ± 1.40	50.24 ± 0.77	75.59 ± 0.36	77.68 ± 1.97	83.81 ± 0.60	72.46 ± 1.49
GCN	86.74 ± 0.47	83.93 ± 0.34	92.70 ± 0.20	30.26 ± 0.79	37.99 ± 0.61	71.23 ± 0.22	74.79 ± 1.78	79.61 ± 0.66	68.14 ± 2.18
TriAxis-GCN (task only)	89.38 ± 0.68	86.57 ± 1.99	93.77 ± 1.82	40.54 ± 2.22	46.98 ± 2.49	82.76 ± 1.12	89.40 ± 1.39	**84.24 ± 0.75**	75.93 ± 1.67
TriAxis-GCN + CF	90.26 ± 0.87	**87.06 ± 1.19**	95.71 ± 0.61	40.74 ± 2.15	47.93 ± 1.16	**83.36 ± 0.47**	90.63 ± 0.59	83.60 ± 0.89	**76.10 ± 1.05**
DirGNN	86.94 ± 0.55	81.22 ± 0.54	95.38 ± 0.32	35.76 ± 1.68	46.66 ± 0.61	80.08 ± 1.07	81.52 ± 0.41	82.64 ± 0.75	59.95 ± 0.79
TriAxis-DirGNN (task only)	89.61 ± 0.37	85.36 ± 1.81	95.02 ± 0.54	40.44 ± 2.41	47.64 ± 1.85	82.90 ± 0.96	89.95 ± 0.85	83.42 ± 0.53	70.90 ± 1.38
TriAxis-DirGNN + CF	**90.34 ± 1.12**	85.58 ± 1.59	95.51 ± 1.07	40.80 ± 2.05	**50.92 ± 1.96**	83.29 ± 0.64	90.25 ± 0.65	83.69 ± 0.82	73.96 ± 1.81
GPRGNN	88.88 ± 0.37	85.70 ± 0.31	94.51 ± 0.39	36.58 ± 1.04	48.03 ± 1.38	69.89 ± 0.37	84.63 ± 1.02	75.68 ± 0.79	70.11 ± 1.05
TriAxis-GPRGNN (task only)	88.66 ± 1.00	84.57 ± 0.58	93.00 ± 0.85	41.17 ± 1.40	50.46 ± 1.56	72.49 ± 3.22	90.20 ± 0.62	82.70 ± 0.97	74.96 ± 1.28
TriAxis-GPRGNN + CF	89.74 ± 1.05	85.07 ± 0.48	93.99 ± 1.06	**41.29 ± 1.32**	50.72 ± 1.28	75.86 ± 2.25	91.18 ± 0.57	82.62 ± 0.67	75.22 ± 1.21

†FAGCN uses a receiver-level grouped-edge adaptation: all incoming edge scores of a selected receiver are clamped jointly, and the mean raw score is supervised. Unlike Mowst, this is not an individual scalar-control instantiation of Definition 1 or Proposition 3.

Node Classification The results are in Table 1. Counterfactual supervision improves performance across most TriAxis comparisons on both homophilic and heterophilic datasets. TriAxis's task-only variants already form a strong adaptive baseline. Counterfactual supervision therefore refines an already effective policy rather than adding the underlying adaptive capacity, and therefore gains are sometimes modest. It still enhances almost all configurations, helping them achieve the best average performance on many datasets, aside from a small number of exceptions such as TriAxis-GCN on Tolokers. The single-axis ablations in Appendix C.4 show the gains do not rely solely on interactions

among the three controls, improving in 25 of 27 dataset axis comparisons, with statistically significant improvements in Appendix C.5. The Mowst results provide a direct external instantiation, showing finite endpoint supervision also improves a structurally different node-wise routing mechanism, while the FAGCN rows offer complementary empirical evidence for a grouped-edge adaptation. TriAxis with counterfactual supervision also competes effectively with other adaptive GNNs, such as H2GCN, OrderedGNN, ACM-GCN, NODE-MOE, and GMoE, in many instances.

Behavior Between Control Endpoints We evaluate receiver loss in TriAxis-GCN for values of c in the set $\{0, 0.25, 0.5, 0.75, 1\}$ across different controls, and we summarize the resulting five-point loss curves in Figure 3. Panel (a) illustrates that the majority of the curves are approximately monotonically increasing or decreasing, which indicates that the ordering of the endpoints typically reflects the trend toward lower loss. Panel (b) reveals that an interior value rarely outperforms both endpoints. Furthermore, panel (c) shows that these rare instances of improvement are significant on PubMed but negligible on Film. Overall, endpoint comparisons offer valuable supervision, while continuous controls help leverage beneficial interior values when available.

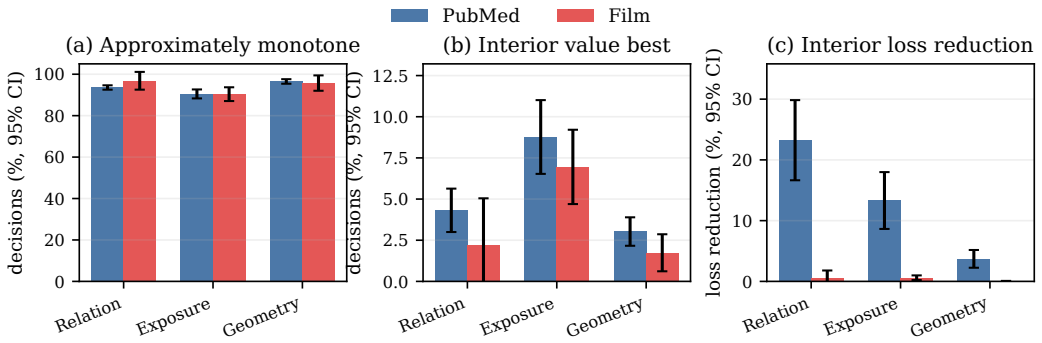

Figure 3: Receiver-loss behavior between control endpoints. Bars summarize the proportion of approximately monotone five-point loss curves, the cases where an interior value outperforms both endpoints, and the corresponding relative loss reduction.

Comparison with Alternative Router Supervision Signals To evaluate the impact of finite endpoint utilities, we apply several auxiliary supervision signals to the Mowst router. All variants utilize the same architecture, data splits, training schedule, and router-only update steps. The difference lies solely in the target used for the auxiliary loss. Specifically, *Local* employs a first-order derivative signal, *Random* adopts random soft targets, *Neutral* uses a constant target of 0.5, and *CF* represents our proposed finite endpoint utility. As shown in Table 2, the CF variant achieves the highest mean test performance across all four datasets. While the non-CF auxiliary targets improve upon the baseline Mowst performance, indicating that additional router updates are beneficial, the consistent advantage of CF over these other methods suggests a more effective supervision signal. This is because decision-specific finite endpoint utilities provide clearer guidance.

Table 2: Router-supervision ablation (test accuracy and standard deviation over ten seeds %).

Method	PubMed	Amazon Photo	Amazon-Ratings	Film
Mowst	87.41±1.24	92.33±1.73	45.60±1.75	39.92±1.33
+Local	89.10±0.50	93.86±0.39	49.98±1.02	39.52±1.32
+Random	89.05±0.45	93.79±0.40	49.28±1.51	39.47±1.34
+Neutral	88.92±0.53	94.03±0.34	49.23±0.78	39.33±1.84
+CF	**89.32±0.53**	**94.85±0.60**	**50.24±0.77**	**40.65±1.40**

Table 3: Classification accuracy and scalability of locality packing. Left: test accuracy. Right: speedups over individual-CF evaluation, overhead versus task-only training, and maximum utility error. Scalability metrics use two-layer models and packs of 16 distance-separated interventions.

	ogbn-arxiv	ogbn-products
GCN	69.53 ± 0.28	74.69 ± 0.25
TriAxis-GCN (task only)	70.02 ± 0.32	74.80 ± 0.24
TriAxis-GCN + CF	**70.51 ± 0.23**	**75.21 ± 0.20**

(a) Node classification accuracy (%) over 10 seeds for random 60/20/20 train/validation/test splits.

	ogbn-arxiv	ogbn-products
Utility speedup	12.74–14.04×	15.56–15.77×
Training speedup	1.54×	1.52×
Individual-CF overhead	80.8%	77.7%
Packed-CF overhead	**17.1%**	**16.8%**
Maximum utility error	0	0

(b) Scalability of locality packing

Node classification and scalability of locality packing on large graphs. Table 3(a) shows that TriAxis-GCN outperforms GCN on the ogbn-arxiv and ogbn-products datasets, with counterfactual supervision providing further improvements. Table 3(b) evaluates locality packing using two-layer models, where the remaining propagation radius R_ℓ is set to $\{0, 1\}$, and packs consist of 16 distance-separated interventions. In this context, packed evaluation achieves a utility speedup of 12 to 16 times, with no observed utility errors. Packed-CF training is about 1.5 times faster than individual-CF training, reducing the overhead compared to task-only training from around 80% to 20%. These results demonstrate practical efficiency of the proposed batching mechanism on large graphs.

Additional Experiments We provide additional experiments in the Appendix C, which include receiver versus aggregate utility alignment and supervision; learned control distributions and their relationship to local graph structure; numerical accuracy and runtime advantages of locality packing; ablation studies on TriAxis controls and descriptor; and sensitivity analysis of the CF hyperparameters.

7 Related Work

Adaptive graph propagation. Standard GNNs share one propagation rule across nodes, although useful propagation can vary with local structure and depth (Zhu et al., 2020; Luan et al., 2022). Existing methods address this through signed filtering (Bo et al., 2021), adaptive depth (Song et al., 2023; Hevapathige et al., 2026), directional propagation (Rossi et al., 2024), or node-wise mixtures of experts (Wang et al., 2023a; Han et al., 2024; Zeng et al., 2024). These controls are typically learned only from task loss. TriAxis-GNN factorizes propagation into node- and layer-wise relation, exposure, and geometry controls, while our CF objective supervises their alternative behaviors.

Counterfactual and intervention-based supervision. Counterfactual methods for GNNs have mainly been studied as post-hoc explanations, where features or edges are modified to identify small changes that alter a prediction (Lucic et al., 2022; Prado-Romero et al., 2024). A related line enforces counterfactual fairness by intervening on sensitive attributes to test prediction invariance (Guo et al., 2023), differing from us in both the intervened object and the purpose of intervention. Interventions have also been used during training. CSA-GNN estimates the effect of removing neighbour information and uses it to supervise graph attention (Wang et al., 2023b), while counterfactual routing analysis studies whether examples are assigned to suitable experts in mixture-of-experts models (Yoon et al., 2026), and knowledge graph completion augments training with generated counterfactual relations (Chang et al., 2023). Beyond graph learning, counterfactual comparisons are used for per-agent credit assignment in multi-agent reinforcement learning (Wolpert and Tumer, 2001; Foerster et al., 2018) and to reduce reliance on spurious features in natural language processing (Kaushik et al., 2020). Our setting differs in the object being supervised. For each node-local control, we fix its two endpoints and recalibrate the downstream network. The receiver-loss difference supervises the controller, while standard task optimization allows for intermediate control selections.

8 Conclusion, Limitations, and Future Work

In this work, we introduced counterfactual supervision for adaptive GNNs. The main idea is to compare two meaningful settings of a propagation control after their effects have been processed by the rest of the network. The resulting difference in loss provides a training signal for optimizing the control. TriAxis-GNN embodies this concept by using node- and layer-specific controls related to relation, exposure, and geometry. Across the datasets we evaluated, incorporating counterfactual supervision typically enhanced the performance of the adaptive models. Additionally, we demonstrate that distance-based batching makes these extra counterfactual evaluations computationally feasible.

Our approach has several limitations. It assumes clear endpoint settings and finite neighbourhoods, so exact batching guarantees do not hold for globally coupled architectures. Node-level utility may also diverge from total loss change across affected nodes, and endpoint comparisons cannot reveal whether intermediate control values are better. Counterfactual evaluation increases training time, and local effects may be minimal when propagation channels are similar or cancel out. Future work could extend the method to globally coupled architectures, multi-valued controls, and more efficient approximations of aggregate utility, as well as adaptive intervention schedules, conflict resolution between task and counterfactual gradients, and applications to diverse downstream tasks.

AI Disclosure Statement

Generative AI tools were used to provide feedback on the research methodology and experimental design, assist with code debugging and improving readability, support the presentation of mathematical arguments, and assist with language editing and manuscript organisation. All AI-assisted content was reviewed and verified by the authors, who take full responsibility for the final content of this work.

Ethics Statement

This work uses publicly available graph benchmark datasets and does not involve human participants, private information, or the collection of new data. We are not aware of any direct harmful applications or substantial ethical risks arising from the proposed method. The authors have reviewed the study and confirm that it adheres to the ICLR Code of Ethics.

Reproducibility Statement

We support reproducibility by providing the complete method, assumptions, and training procedure in the main paper. Appendix A contains proofs of the theoretical results. Appendix B reports dataset statistics, hyperparameter settings, backbone adaptations, and implementation details. Appendix C provides additional experiments and ablation studies, while Appendix D presents the complete training algorithm. A link to anonymized source code is provided in the abstract.

References

Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *international conference on machine learning*, pages 21–29. PMLR, 2019.

Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. Beyond low-frequency information in graph convolutional networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 3950–3957, 2021.

Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *International Conference on Learning Representations*, 2018.

Giorgos Bouritsas, Fabrizio Frasca, Stefanos Zafeiriou, and Michael M Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):657–668, 2022.

Heng Chang, Jie Cai, and Jia Li. Knowledge graph completion with counterfactual augmentation. In *Proceedings of the ACM web conference 2023*, pages 2611–2620, 2023.

Jialong Chen, Bowen Deng, Zhen Wang, Chuan Chen, and Zibin Zheng. Graph neural ricci flow: Evolving feature from a curvature perspective. In *International Conference on Learning Representations*, volume 2025, pages 31083–31111, 2025.

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *International conference on machine learning*, pages 1725–1735. PMLR, 2020.

Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network. In *International Conference on Learning Representations*, 2021.

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30, 2006.

Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

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

Assefaw Hadish Gebremedhin, Fredrik Manne, and Alex Pothen. What color is your jacobian? graph coloring for computing derivatives. *SIAM review*, 47(4):629–705, 2005.

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*, pages 1263–1272. PMLR, 2017.

Cheng-Hua Gong, Yao Cheng, Jian-Xiang Yu, Can Xu, Cai-Hua Shan, Si-Qiang Luo, and Xiang Li. A survey on learning from graphs with heterophily: Recent advances and future directions. *Frontiers of Computer Science*, 20(2):2002314, 2026.

Zhimeng Guo, Jialiang Li, Teng Xiao, Yao Ma, and Suhang Wang. Towards fair graph neural networks via graph counterfactual. In *Proceedings of the 32nd ACM international conference on information and knowledge management*, pages 669–678, 2023.

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

Haoyu Han, Juanhui Li, Wei Huang, Xianfeng Tang, Hanqing Lu, Chen Luo, Hui Liu, and Jiliang Tang. Node-wise filtering in graph neural networks: A mixture of experts approach. *arXiv preprint arXiv:2406.03464*, 2024.

Asela Hevapathige, Asiri Wijesinghe, and Ahad N Zehmakan. Beyond fixed depth: Adaptive graph neural networks for node classification under varying homophily. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 21717–21725, 2026.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

Divyansh Kaushik, Eduard Hovy, and Zachary Lipton. Learning the difference that makes a difference with counterfactually-augmented data. In *International Conference on Learning Representations*, 2020.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

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

Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. Revisiting heterophily for graph neural networks. *Advances in neural information processing systems*, 35:1362–1375, 2022.

Ana Lucic, Maartje A Ter Hoeve, Gabriele Tolomei, Maarten De Rijke, and Fabrizio Silvestri. Cf-gnnexplainer: Counterfactual explanations for graph neural networks. In *International conference on artificial intelligence and statistics*, pages 4499–4511. PMLR, 2022.

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

Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. A critical look at the evaluation of gnns under heterophily: Are we really making progress? In *The Eleventh International Conference on Learning Representations*, 2023.

Mario Alfonso Prado-Romero, Bardh Prenkaj, Giovanni Stilo, and Fosca Giannotti. A survey on graph counterfactual explanations: definitions, methods, evaluation, and research challenges. *ACM Computing Surveys*, 56(7):1–37, 2024.

Emanuele Rossi, Bertrand Charpentier, Francesco Di Giovanni, Fabrizio Frasca, Stephan Günnemann, and Michael M Bronstein. Edge directionality improves learning on heterophilic graphs. In *Learning on graphs conference*, pages 25–1. PMLR, 2024.

T Konstantin Rusch, Michael M Bronstein, and Siddhartha Mishra. A survey on oversmoothing in graph neural networks. *arXiv preprint arXiv:2303.10993*, 2023.

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

Yunchong Song, Chenghu Zhou, Xinbing Wang, and Zhouhan Lin. Ordered gnn: Ordering message passing to deal with heterophily and over-smoothing. *International Conference on Learning Representations*, 2023.

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

Haotao Wang, Ziyu Jiang, Yuning You, Yan Han, Gaowen Liu, Jayanth Srinivasa, Ramana Rao Kompella, and Zhangyang Wang. Graph mixture of experts: Learning on large-scale graphs with explicit diversity modeling. *Advances in Neural Information Processing Systems*, 36:50825–50837, 2023a.

Hongjun Wang, Jiyuan Chen, Lun Du, Qiang Fu, Shi Han, and Xuan Song. Causal-based supervision of attention in graph neural network: a better and simpler choice towards powerful attention. *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, 2023b.

David H Wolpert and Kagan Tumer. Optimal payoff functions for members of collectives. *Advances in Complex Systems*, 4(02n03):265–279, 2001.

Youngsik Yoon, Siwei Wang, Wei Chen, and Jungseul Ok. When are experts misrouted? counterfactual routing analysis in mixture-of-experts language models. *arXiv preprint arXiv:2605.07260*, 2026.

Zichao Yue and Zhiru Zhang. Gate the filter, not the message: Node-channel mixtures for pre-propagation gnns. *arXiv preprint arXiv:2606.01660*, 2026.

Hanqing Zeng, Hanjia Lyu, Diyi Hu, Yinglong Xia, and Jiebo Luo. Mixture of weak and strong experts on graphs. In *International Conference on Learning Representations*, volume 2024, pages 24462–24502, 2024.

Xu Zhang, Yonghui Xu, Wei He, Wei Guo, and Lizhen Cui. A comprehensive review of the oversmoothing in graph neural networks. In *CCF Conference on Computer Supported Cooperative Work and Social Computing*, pages 451–465. Springer, 2023.

Xin Zheng, Yi Wang, Yixin Liu, Ming Li, Miao Zhang, Di Jin, Philip S Yu, and Shirui Pan. Graph neural networks for graphs with heterophily: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 2026.

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.

APPENDIX

A PROOFS

In this section, we provide proofs for the propositions introduced in the main paper.

Proposition 1 (restated). *For $q = (i, \ell, a)$, let $\mathcal{J}_q(c)$ be the receiver loss when only $c^a_{i\ell}$ is clamped to $c \in [0, 1]$ and the complete suffix is recomputed, and let $u_q = \mathcal{J}_q(0) - \mathcal{J}_q(1)$. If $\mathcal{J}_q$ is continuously differentiable on $[0, 1]$, then*

$$u_q = -\int_0^1 \mathcal{J}'_q(t)\, dt.$$

If $\mathcal{J}'_q$ is also L_q-Lipschitz, then, for every $c_q \in [0, 1]$,

$$\left|u_q + \mathcal{J}'_q(c_q)\right| \le \frac{L_q}{2}\left(c_q^2 + (1 - c_q)^2\right).$$

Hence, if $|u_q|$ exceeds the right-hand side, then $\operatorname{sign}(u_q) = -\operatorname{sign}(\mathcal{J}'_q(c_q))$.

Proof. The fundamental theorem of calculus gives

$$u_q = -\int_0^1 \mathcal{J}'_q(t)\, dt.$$

If $\mathcal{J}'_q$ is L_q-Lipschitz, then, for any $c_q \in [0, 1]$,

$$\begin{aligned}\left|u_q + \mathcal{J}'_q(c_q)\right| &= \left|\int_0^1 \left(\mathcal{J}'_q(c_q) - \mathcal{J}'_q(t)\right) dt\right| \\ &\le L_q \int_0^1 |c_q - t|\, dt = \frac{L_q}{2}\left(c_q^2 + (1 - c_q)^2\right).\end{aligned}$$

Let the right-hand side be B_q. If $u_q > B_q$, then $-\mathcal{J}'_q(c_q) > u_q - B_q > 0$, so $\mathcal{J}'_q(c_q) < 0$. If $u_q < -B_q$, then $-\mathcal{J}'_q(c_q) < u_q + B_q < 0$, so $\mathcal{J}'_q(c_q) > 0$. Thus, $|u_q| > B_q$ implies $\operatorname{sign}(u_q) = -\operatorname{sign}(\mathcal{J}'_q(c_q))$. □

Proposition 2 (restated). *Let (U, R) be the utility and descriptor obtained by sampling a decision from the target decision population, and for fixed $\tau > 0$ define*

$$\widetilde{Y} = \frac{1}{2}\left(1 + \frac{U}{\tau + |U|}\right), \qquad \mathcal{R}_{\mathrm{CF}}(c) = \mathbb{E}\Big[\mathrm{BCE}(\widetilde{Y}, c(R))\Big],$$

where $c(R)$ is the controller output. For every descriptor value r, the pointwise minimizer is

$$c^*(r) = \mathbb{E}[\widetilde{Y} \mid R = r] = \frac{1}{2}\left(1 + \mathbb{E}\left[\frac{U}{\tau + |U|}\,\middle|\, R = r\right]\right).$$

Thus $c^*(r) \ge 1/2$ exactly when $\mathbb{E}[U/(\tau + |U|) \mid R = r] \ge 0$.

Proof. Treat the positive scale τ as fixed at its population value and define

$$\widetilde{Y} = \frac{1}{2}\left(1 + \frac{U}{\tau + |U|}\right).$$

Condition on $R = r$ and write

$$\mu(r) = \mathbb{E}[\widetilde{Y} \mid R = r].$$

Because binary cross-entropy is affine in its target, the conditional risk is

$$-\mu(r) \log c(r) - (1 - \mu(r)) \log(1 - c(r)).$$

Its pointwise minimizer is $c^*(r) = \mu(r)$, hence

$$c^*(r) = \mathbb{E}[\widetilde{Y} \mid R = r] = \frac{1}{2}\left(1 + \mathbb{E}\left[\frac{U}{\tau + |U|}\,\middle|\, R = r\right]\right).$$

Subtracting $1/2$ from both sides shows that $c^*(r) \geq 1/2$ if and only if

$$\mathbb{E}\left[\frac{U}{\tau + |U|} \middle| R = r \right] \geq 0,$$

which proves the proposition. □

Proposition 3 (restated). *Let $\mathcal{I}_\ell \subseteq \mathcal{V}_{\mathrm{tr}}$ satisfy*

$$\mathrm{dist}_G(i,j) > R_\ell \qquad \textit{for all distinct } i,j \in \mathcal{I}_\ell,$$

where

$$R_\ell = \sum_{t=\ell+1}^{K-1} r_t.$$

For a fixed control a, jointly clamp all selected nodes to endpoint zero and then jointly clamp them to endpoint one, leaving all other controls at their deployed values. Under Assumption 1, the resulting loss difference at every $i \in \mathcal{I}_\ell$ equals the individual counterfactual utility $u^a_{i\ell}$ under the same stochastic realization.

Proof. Fix a selected node $i \in \mathcal{I}_\ell$, a control a, and an endpoint $b \in \{0,1\}$. Once the stochastic realization and model state are fixed, the suffix is a deterministic function of the states immediately after the intervention, $H^{(\ell+1)}$.

Consider the backward dependency set of the final prediction at node i. Under Assumption 1, passing backward through a layer of radius r_t can expand this set by at most r_t graph hops. Before tracing through any suffix layer, the loss at i depends only on the final state of i. Passing backward through the last layer introduces dependencies at most r_{K-1} hops away. Passing through the preceding layer adds at most r_{K-2} hops. Repeating this argument through all remaining layers shows that the prediction at i can depend only on states $H_v^{(\ell+1)}$ satisfying

$$\mathrm{dist}_G(i,v) \leq \sum_{t=\ell+1}^{K-1} r_t = R_\ell. \tag{6}$$

Now compare two endpoint-b executions. In the individual execution, only $c^a_{i\ell}$ is clamped to b. In the joint execution, $c^a_{j\ell}$ is clamped to b for every $j \in \mathcal{I}_\ell$. The intervention at i is identical in both executions. Their only additional differences occur at nodes $j \in \mathcal{I}_\ell \setminus \{i\}$.

For every such j, the separation condition gives $\mathrm{dist}_G(i,j) > R_\ell$. Thus no additional jointly intervened node lies in the backward dependency ball of i. The joint and individual executions agree on every layer-$\ell+1$ state that can affect the final prediction at i. Because they also share the same parameters, graph sample, normalization state, controller state, and stochastic realization, they produce exactly the same final output and node loss at i. Therefore, for each $b \in \{0,1\}$,

$$L^{(b)}_{i,\mathcal{I}_\ell} = L^{(b)}_{i,\{i\}}.$$

Subtracting the equality at $b=1$ from that at $b=0$ gives

$$\begin{aligned} L^{(0)}_{i,\mathcal{I}_\ell} - L^{(1)}_{i,\mathcal{I}_\ell} &= L^{(0)}_{i,\{i\}} - L^{(1)}_{i,\{i\}} \\ &= u^a_{i\ell}. \end{aligned}$$

Since i was arbitrary, the equality holds for every selected node.

The radius-R_ℓ forward influence regions of different interventions need not be disjoint. They may overlap at nodes whose losses are not used for the corresponding receiver-specific utility. Exactness requires only that another selected intervention not enter the backward dependency region of a selected node's own prediction, which is guaranteed by $\mathrm{dist}_G(i,j) > R_\ell$. □

Corollary 4 (restated). *If $\mathrm{dist}_G(i,j) > 2R_\ell$, summing each jointly evaluated loss difference over $\mathcal{V}_{\mathrm{tr}} \cap B_{R_\ell}(i)$ and dividing by $|\mathcal{V}_{\mathrm{tr}}|$ recovers the aggregate utility $u^{a,\mathrm{agg}}_{i\ell}$ exactly.*

Proof. Fix $i \in \mathcal{I}_\ell$ and a labeled receiver $v \in B_{R_\ell}(i)$. An intervention at another selected node $j \in \mathcal{I}_\ell \setminus \{i\}$ can affect the final loss at v only if $\mathrm{dist}_G(j, v) \le R_\ell$. By the triangle inequality,

$$\mathrm{dist}_G(i, j) \le \mathrm{dist}_G(i, v) + \mathrm{dist}_G(v, j) \le 2R_\ell$$

would then be necessary. This contradicts the assumed separation $\mathrm{dist}_G(i, j) > 2R_\ell$. Therefore, for every $v \in B_{R_\ell}(i)$, the endpoint-b loss produced by the joint intervention is identical to that produced by intervening on i alone. Summing the equality over labeled receivers in the ball for $b = 0$ and $b = 1$, then taking their difference and dividing by $|\mathcal{V}_{\mathrm{tr}}|$, yields exactly $u_{i\ell}^{a,\mathrm{agg}}$. Since i was arbitrary, the result holds for every selected intervention. □

B Experimental Details

B.1 Dataset Statistics

Table 4 provides statistical information on the benchmark datasets.

Table 4: Benchmark dataset statistics

Setting	Dataset	Nodes	Edges	Features	Classes	Metric
Homophily	PubMed	19,717	44,324	500	3	Accuracy
	DBLP	17,716	105,734	1,639	4	Accuracy
	Amazon Photo	7,650	119,081	745	8	Accuracy
Heterophily	Film	7,600	30,019	932	5	Accuracy
	Amazon-Ratings	24,492	93,050	300	5	Accuracy
	Roman-Empire	22,662	32,927	300	18	Accuracy
	Minesweeper	10,000	39,402	7	2	ROC–AUC
	Tolokers	11,758	519,000	10	2	ROC–AUC
	Questions	48,921	153,540	301	2	ROC–AUC
Large-scale	ogbn-arxiv	169,343	1,166,243	128	40	Accuracy
	ogbn-products	2,449,029	61,859,140	100	47	Accuracy

B.2 Counterfactual Supervision Implementation Across Backbones

This section describes how counterfactual supervision is applied to the TriAxis variants and the two external backbones in Table 1.

All TriAxis variants share the controller, descriptor, and Equation (4). They differ only in how the neighbour channel N and triangle channel M are built. Proposition 3 still applies, since it allows any finite-radius aggregation. Because $\mathrm{supp}(A_\triangle) \subseteq \mathrm{supp}(A)$, both channels have a one-hop receptive field, so $R_\ell = K - 1 - \ell$ for every variant.

TriAxis-GCN: E is a self-loop-free linear transformation. The ordinary and triangle channels share one self-loop-free graph convolution, applied to the ordinary adjacency for N and the triangle-weighted adjacency for M. The channels therefore differ only in their propagated edges and weights.

TriAxis-GPRGNN: E is a self-loop-free linear transformation. Both N and M use the shared signed first-order filter $\gamma_0 E + \gamma_1(\cdot)$, applied to the ordinary and triangle adjacencies respectively. The first-order restriction keeps a one-hop receptive field per layer.

TriAxis-DirGNN: E is a self-loop-free linear transformation. Both channels share one learned mixture of incoming and outgoing propagation. Unlike the other variants, the original edge directions are kept.

For every variant, CF supervision computes the shared prefix once and recomputes only the suffix under the two clamped endpoints, as in the training algorithm of Appendix D. The backbone therefore changes how channels are computed, but not the locality-packing argument.

Mowst: For our experiments, we implement Mowst using a weak MLP expert and a strong GCN expert with a differentiable mixture of their output logits. A router computes a logit z_i from the maximum class probability, entropy, and top-two probability margin of the weak expert. The final prediction is

$$c_i = \sigma(z_i/T), \qquad \hat{\mathbf{p}}_i = \mathrm{softmax}\big(c_i \mathbf{h}_i^{\mathrm{weak}} + (1-c_i)\mathbf{h}_i^{\mathrm{strong}}\big),$$

where T is the router temperature and $\mathbf{h}_i^{\mathrm{weak}}$ and $\mathbf{h}_i^{\mathrm{strong}}$ are the class-logit vectors produced by the two experts. Thus, $c_i = 0$ selects the strong expert, whereas $c_i = 1$ selects the weak expert.

Because mixing occurs only at the output of node i, intervening on c_i cannot affect another node's prediction. The intervention radius is therefore $R_\ell = 0$, allowing all distinct selected training nodes to be clamped simultaneously. Two endpoint evaluations with shared randomness recover their exact receiver utilities. These utilities define the soft targets in Equation 1. During each CF update, the expert parameters are frozen and the confidence statistics are detached, so only the router is optimized.

FAGCN: For each oriented message $j \to i$, FAGCN assigns a signed coefficient

$$e_{ij} = \tanh(s_{ij}), \qquad s_{ij} = \mathbf{g}^\top [\mathbf{h}_i \| \mathbf{h}_j],$$

where s_{ij} is the pre-$\tanh$ edge score. Although FAGCN is originally formulated on an undirected graph, its message-passing computation associates a coefficient with each oriented edge. Its controls are therefore edge-level rather than node-level and do not constitute an exact instance of the node-wise framework in Section 3.

For each locality-packed set of receiver nodes, we jointly clamp the pre-$\tanh$ scores of all incoming edges of each receiver to -8 or $+8$. This produces coefficients numerically close to -1 or $+1$. Let R_ℓ denote the number of propagation layers following the intervened layer. We select receivers whose pairwise graph distances are greater than R_ℓ. Under Assumption 1, an intervention associated with one selected receiver then cannot influence the final loss of another selected receiver.

To obtain one supervision signal per receiver, we use the mean pre-$\tanh$ score over its incoming edges as a surrogate controller logit. The resulting utility measures the joint effect of intervening on multiple incoming-edge coefficients. It is therefore a receiver-level aggregate rather than the exact per-decision utility in Definition 1. Accordingly, we report FAGCN+CF as a generalizability experiment rather than as an exact application of Proposition 3.

B.3 Model Hyperparameters

The TriAxis variants, their paired base models, and all models with counterfactual supervision are tuned by grid search over an identical search space, so that performance differences reflect the propagation mechanism and the supervision signal rather than tuning effort. For the remaining baselines, we present results from Chen et al. (2025) and Hevapathige et al. (2026), which adhere to the same split protocol, optimizer, and use validation-loss early stopping as in our experiments. In cases where results are not available, we reproduce the baseline and tune it within the same search space that we used for our models, starting from the hyperparameters specified in the original paper. This provides a consistent evaluation protocol and comparable tuning effort across methods.

We search for the learning rate from $\{0.005, 0.01\}$, the weight decay from $\{1 \times 10^{-4}, 5 \times 10^{-4}\}$, the number of propagation layers from $\{1, 2\}$, the hidden dimension from $\{64, 256\}$, and the dropout from $\{0.1, 0.2, 0.5\}$. CF supervision introduces two additional hyperparameters: the controller temperature $T \in \{0.1, 0.2\}$ and the counterfactual update interval $T_{\mathrm{CF}} \in \{1, 2\}$. The counterfactual loss weight λ_{CF} is fixed at 1. At each CF update, we sample up to 256 valid, sufficiently separated training receivers for joint endpoint evaluation in the main experiments. Each model is trained for a maximum of 1000 epochs, with early stopping applied after 200 epochs without improvement in validation loss, and the parameters with the best validation loss are restored for testing. The final configuration for each model is selected by validation accuracy.

B.4 Implementation Details

All of our models are implemented in PyTorch 2.9.1 with PyTorch Geometric 2.7.0 under Python 3.11. Experiments are conducted on a Linux server equipped with a single NVIDIA A100 GPU, and every model is trained using the Adam optimiser (Kingma and Ba, 2015).

C Additional Experiments

C.1 Receiver-Specific and Aggregate Utility

An early-layer intervention can impact the losses of nodes beyond just its intended receiver. Therefore, we assess how closely the receiver-specific utility $u_{i\ell}^{a}$ aligns with the corresponding aggregate task utility $u_{i\ell}^{a,\mathrm{agg}}$. We evaluate the trained TriAxis-GCN+CF models on the DBLP and Roman-Empire datasets using ten different seeds. For each layer and control, we select up to 256 labeled receivers that have a pairwise distance greater than $2R_\ell$. This ensures that their influence balls are disjoint, allowing both utilities to be accurately derived from a single, jointly clamped endpoint pair.

We report sign agreement and conflict among decisions for which both utilities are nonzero, their Spearman correlation, and the mean absolute task-scale difference

$$\Delta_{\mathrm{abs}} = \left| u_{i\ell}^{a,\mathrm{agg}} - \frac{u_{i\ell}^{a}}{|\mathcal{V}_{\mathrm{tr}}|} \right|.$$

Writing δ_v for the bracketed term of Eq. (5) at node v, we also report how often the sufficient agreement condition

$$|\delta_i| > \sum_{v \in (\mathcal{V}_{\mathrm{tr}} \cap B_{R_\ell}(i)) \setminus \{i\}} |\delta_v|$$

holds. Geometry results are conditional on triangle-supported receivers, which comprise 43.8% of DBLP and 71.5% of Roman-Empire training nodes.

Table 5: Agreement between receiver-specific and aggregate task utilities at the first layer. Values are means with 95% confidence intervals over ten seeds. The final layer is omitted because $R_{K-1} = 0$, making the receiver and aggregate utilities identical up to task-loss normalization.

Dataset	Control	Sign agreement (%)	Conflict (%)	Spearman ρ	Mean Δ_{abs}	Sufficient condition (%)
DBLP	Relation	97.8 ± 1.4	2.2 ± 1.4	0.94 ± 0.04	$(1.03 \pm 0.74) \times 10^{-5}$	92.0 ± 6.7
	Exposure	96.6 ± 1.6	3.4 ± 1.6	0.93 ± 0.04	$(5.16 \pm 3.68) \times 10^{-6}$	87.1 ± 5.7
	Geometry	92.9 ± 3.7	7.1 ± 3.7	0.87 ± 0.07	$(2.09 \pm 1.62) \times 10^{-6}$	69.2 ± 8.7
Roman-Empire	Relation	97.9 ± 0.9	2.1 ± 0.9	0.94 ± 0.02	$(4.38 \pm 0.70) \times 10^{-5}$	89.4 ± 2.2
	Exposure	85.8 ± 1.1	14.2 ± 1.1	0.85 ± 0.03	$(1.34 \pm 0.13) \times 10^{-5}$	64.3 ± 2.3
	Geometry	82.0 ± 1.8	18.0 ± 1.8	0.80 ± 0.02	$(4.45 \pm 0.84) \times 10^{-6}$	60.3 ± 1.8

Table 5 indicates that receiver utility aligns well with aggregate task preference, showing over 92% sign agreement across three controls on DBLP, and around 98% for the relation control on both datasets. However, the Roman-Empire dataset shows a more complex behaviour, where exposure and geometry axes show conflict rates of 14.2% and 18.0%. Yet they still have strong Spearman correlations, showing that stronger receiver utilities generally correspond to stronger aggregate utilities. The sufficient condition is less frequent than the sign agreement observed. This shows that receiver utility usually provides a good approximation to the overall task preference, but it cannot guarantee that helping one receiver will improve the total loss across all affected nodes.

C.2 Receiver versus Aggregate Counterfactual Supervision

Receiver utility measures the change in loss at the controlled node, while aggregate utility sums the changes in loss across all affected labeled nodes. We compare the two signals using receivers that are separated by more than $2R_\ell$, which allows for precise evaluation of aggregate utility, as established in Corollary 4. Table 6 compares task-only training with receiver-CF and aggregate-CF on the DBLP and Roman-Empire datasets, using the same model configuration and random seeds but differing in utility targets.

Receiver-CF achieves the highest mean accuracy on both datasets, while aggregate-CF does not improve over task-only training. This suggests that the receiver-specific utility provides a clearer and more effective training signal. We believe this is because the aggregate target reflects influences from other nodes rather than the receiver's preferences, therefore introducing training noise without a useful signal during testing. Additionally, summing weakly correlated per-node effects increases the variance of the aggregate target compared to the receiver target. We therefore use receiver utility for training and aggregate utility mainly for analysis.

Table 6: Receiver- versus aggregate-utility supervision. Test accuracy (%) of TriAxis-GCN over ten seeds.

Training objective	DBLP	Roman-Empire
Task only	86.57 ± 1.99	82.76 ± 1.12
Receiver-CF	$\mathbf{87.06 \pm 1.19}$	$\mathbf{83.36 \pm 0.47}$
Aggregate-CF	86.17 ± 2.31	82.39 ± 0.77

C.3 Interpretability of Learned Propagation Controls

We examine the learned relation, exposure, and geometry controls of CF-trained TriAxis models on Minesweeper and Questions. We characterize their distributions across layers and associations with local graph structure.

The three axes learn distinct propagation regimes. Table 7 summarizes the learned controls, and Figure 4 shows their distributions across nodes. The three axes occupy clearly different regions of their respective behaviour spaces rather than converging to a common neutral configuration. In Minesweeper, the relation aspect strongly favors self-referenced information, increasing from 0.80 to 0.91 across layers. On the other hand, exposure relies more on neighbor information, with lower values around 0.12 and 0.09. Geometry is more balanced, with similar values for ordinary and triangle-weighted neighborhoods. For Questions, relation stays close to 0.90 across layers, while geometry shifts to favor triangle-weighted neighborhoods, increasing slightly. Exposure sits in between, decreasing from about 0.43 to 0.38. This contrast between Minesweeper and Questions shows that the model learns different settings for each dataset, using various combinations of the three propagation methods.

Table 7: Learned TriAxis propagation controls. The reported values are mean node-wise control values across runs.

Dataset	Layer	Relation e	Exposure w	Geometry s
Minesweeper	1	0.80	0.12	0.53
	2	0.91	0.09	0.52
Questions	1	0.90	0.43	0.68
	2	0.90	0.38	0.71

The learned policies are locally adaptive. The distributions in Figure 4 additionally show that the controllers can produce node-dependent decisions rather than merely learning three global dataset-level coefficients. To examine whether this variation corresponds to meaningful local graph context, we relate the learned controls to node-level structural properties. The relationship shown in Figure 5 illustrates how control in Minesweeper increases with local homophily, while exposure shows a contrary pattern. Nodes in different local neighborhoods experience distinct propagation configurations. Notably, local label homophily is used only for interpretation and isn't part of the controller's decision-making, suggesting that the learned propagation policy shapes these relationships. This suggests that the controller adjusts propagation differently based on local context rather than just applying a uniform change.

C.4 Counterfactual Supervision on Single Control Axes

The complete TriAxis model incorporates three controls, which allows for different combinations to create similar propagation behaviors. To evaluate whether CF supervision remains effective when propagation is managed by a single control, we trained the TriAxis-GPRGNN with only one active axis at a time. In this configuration, the active control dictates the propagation choice at each node and layer.

Table 8 presents the results. CF supervision enhances the average performance in 25 out of 27 dataset-axis comparisons. It improves all nine datasets for the relation axis, eight for the exposure axis, and eight for the geometry axis.

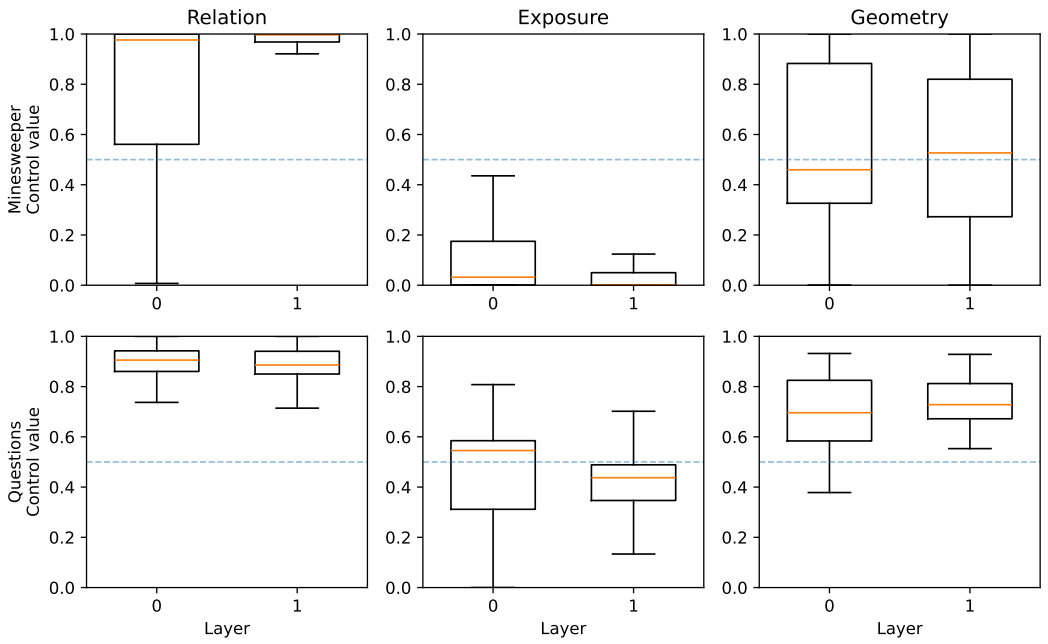

Figure 4: Distribution of learned propagation controls. Node-wise relation, exposure, and geometry controls for Minesweeper (top) and Questions (bottom) across the two propagation layers.

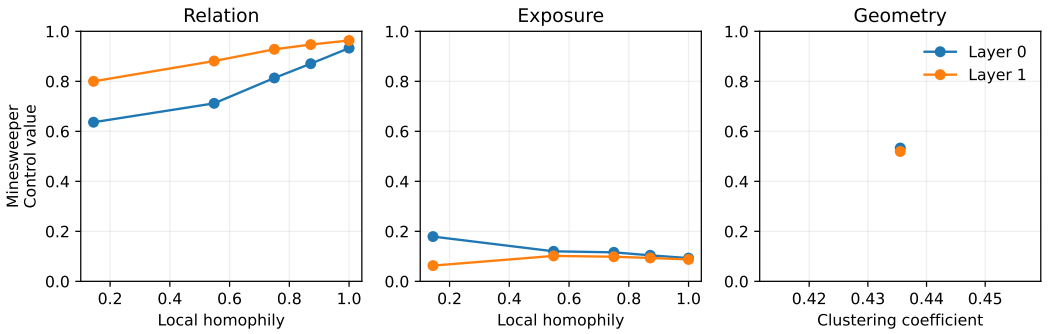

Figure 5: Node-wise adaptation to local graph context. Learned relation and exposure controls as functions of local homophily on Minesweeper, shown separately for the two propagation layers.

Table 8: Single-axis TriAxis-GPRGNN with and without counterfactual supervision.

Dataset	Metric	Axis	No CF	CF	Relative improvement (%)
PubMed	Accuracy	Exposure	87.70 ± 0.43	88.59 ± 0.40	$+1.01$
		Geometry	87.78 ± 0.38	88.49 ± 0.38	$+0.81$
		Relation	88.98 ± 0.62	89.21 ± 0.49	$+0.26$
DBLP	Accuracy	Exposure	84.54 ± 1.01	84.71 ± 1.03	$+0.20$
		Geometry	84.76 ± 0.97	85.74 ± 0.98	$+1.16$
		Relation	84.93 ± 0.98	85.84 ± 0.97	$+1.07$
Photo	Accuracy	Exposure	92.66 ± 0.42	93.59 ± 0.53	$+1.00$
		Geometry	92.47 ± 0.58	93.47 ± 0.39	$+1.08$
		Relation	93.44 ± 0.47	93.93 ± 0.51	$+0.52$
Film	Accuracy	Exposure	40.75 ± 1.99	41.18 ± 2.08	$+1.06$
		Geometry	41.13 ± 2.18	41.24 ± 2.20	$+0.27$
		Relation	40.38 ± 1.68	41.10 ± 1.95	$+1.78$
Amazon-Ratings	Accuracy	Exposure	48.90 ± 1.05	50.27 ± 0.96	$+2.80$
		Geometry	49.08 ± 1.19	50.18 ± 1.40	$+2.24$
		Relation	49.60 ± 1.46	50.39 ± 1.11	$+1.59$
Roman-Empire	Accuracy	Exposure	69.20 ± 0.85	70.98 ± 1.84	$+2.57$
		Geometry	72.34 ± 0.61	74.27 ± 0.64	$+2.67$
		Relation	73.05 ± 0.40	74.50 ± 0.30	$+1.98$
Minesweeper	ROC-AUC	Exposure	89.29 ± 0.77	89.38 ± 0.82	$+0.10$
		Geometry	85.81 ± 1.67	87.58 ± 2.05	$+2.06$
		Relation	91.00 ± 0.60	91.01 ± 0.65	$+0.01$
Tolokers	ROC-AUC	Exposure	82.34 ± 0.56	82.33 ± 0.49	-0.01
		Geometry	74.90 ± 0.58	74.91 ± 0.58	$+0.01$
		Relation	82.88 ± 0.84	83.64 ± 0.62	$+0.92$
Questions	ROC-AUC	Exposure	74.68 ± 2.52	76.07 ± 1.05	$+1.86$
		Geometry	71.56 ± 1.12	71.56 ± 1.15	$+0.00$
		Relation	75.81 ± 1.28	75.90 ± 1.29	$+0.12$

C.5 Statistical Significance Across Datasets

We evaluate cross-dataset consistency using a two-sided Wilcoxon signed-rank test, as described by Demšar (2006). In this analysis, each dataset serves as a single observation. For each model or axis, we compare its mean score with and without counterfactual on the same dataset. We apply Holm's correction across the seven planned comparisons. The results are summarized in Table 9; all seven comparisons remain significant after the correction. These findings support the effectiveness of CF supervision across the tested models and in the single-axis setting.

Table 9: Two-sided Wilcoxon signed-rank tests comparing mean performance with and without counterfactual supervision across nine datasets. Holm adjustment is applied across the seven comparisons. Wins count datasets on which the CF mean is higher; ties count as non-wins.

Setting	Model or axis	Wins	Raw p	Holm-adjusted p
Full models (Table 1)	TriAxis-GCN	$8/9$	0.039	0.039
	TriAxis-DirGNN	$9/9$	0.004	0.027
	TriAxis-GPRGNN	$8/9$	0.008	0.031
	Mowst	$9/9$	0.004	0.027
Single-axis GPRGNN (Table 8)	Exposure	$8/9$	0.008	0.031
	Geometry	$8/9$	0.008	0.031
	Relation	$9/9$	0.004	0.027

C.6 Empirical Characterization of Learned Controls

We characterize the learned controls on Amazon Photo, a homophilous, triangle-dense co-purchase graph. Figure 6 shows their node-wise distributions at both propagation layers. Table 10 reports higher mean relation and exposure values in the first layer than in the second, whereas mean geometry increases between layers. The learned propagation configuration therefore differs across layers.

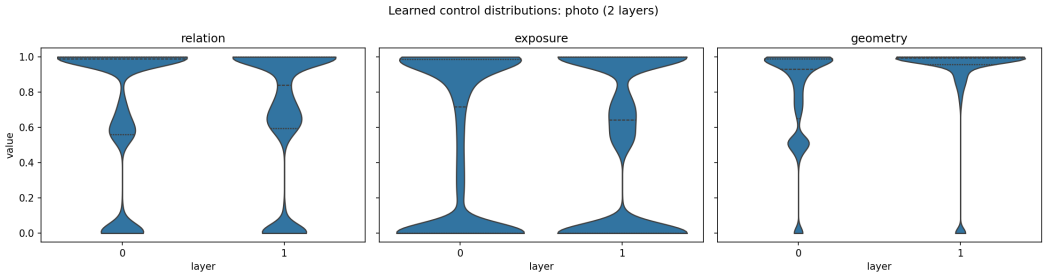

Figure 6: Per node distributions of the controls on the Amazon Photo dataset.

Table 10 also reports cross-seed stability, measured by the mean pairwise Spearman correlation between node-wise control vectors from independently trained models. Relation and exposure are more stable in layer 1 than in layer 2, whereas geometry shows the opposite pattern. The limited agreement indicates variability in individual node-wise controls across runs.

Table 10: Per node control statistics on Amazon Photo. Spearman is the mean pairwise cross seed correlation of the per node control vector.

Control	Layer	Mean	Std	Spearman
Relation	1	0.845	0.079	0.409
Relation	2	0.552	0.012	0.062
Exposure	1	0.929	0.049	0.603
Exposure	2	0.422	0.186	0.084
Geometry	1	0.415	0.158	0.095
Geometry	2	0.729	0.183	0.305

C.7 Detailed Exactness and Scalability Results

We look at locality packing using two-layer models on the ogbn-arxiv and ogbn-products datasets. In these models, the first layer has a remaining radius of 1, and the second layer has a radius of 0. Utilities for receivers depend on nodes that are farther apart than the radius, while aggregate utilities require nodes to be more than twice the radius apart. Our experiments evaluate both relation and exposure for each layer, using packs of 16 interventions to measure utility efficiency.

Numerical exactness. The comparison in Table 11 shows that both packed utilities and those from individual interventions are the same. Across all datasets, layers, and controls, the receiver and aggregate utilities match perfectly, with no errors observed.

Table 11: Exactness of locality-packed counterfactual evaluation. Errors compare packed utilities with utilities obtained by evaluating the same interventions individually.

Dataset	Control	Layer	R_ℓ	Receiver decisions	Aggregate decisions	Receiver mean error	Receiver max. error	Aggregate mean error	Aggregate max. error
ogbn-arxiv	Relation	1	1	48	24	0	0	0	0
	Exposure	1	1	48	24	0	0	0	0
	Relation	2	0	48	24	0	0	0	0
	Exposure	2	0	48	24	0	0	0	0
ogbn-products	Relation	1	1	48	24	0	0	0	0
	Exposure	1	1	48	24	0	0	0	0
	Relation	2	0	48	24	0	0	0	0
	Exposure	2	0	48	24	0	0	0	0

Utility-evaluation efficiency. Table 12 reports the computational cost of evaluating packs containing 16 decisions. Packed evaluation is significantly faster, being about 12.74 to 14.04 times quicker on ogbn-arxiv and 15.56 to 15.77 times quicker on ogbn-products. It also requires less peak memory while maintaining accuracy in evaluations.

Table 12: Efficiency of locality-packed counterfactual evaluation. Times are in seconds, and memory is peak resident memory in MiB. Speedup is individual evaluation time divided by packed evaluation time.

Dataset	Control	Layer	R_ℓ	Packed nodes	Individual time	Packed time	Speedup	Individual memory	Packed memory	Max. utility error
ogbn-arxiv	Relation	1	1	16	0.4039	0.0288	**14.04**×	592.17	566.32	0
	Exposure	1	1	16	0.4034	0.0288	**14.02**×	592.17	566.32	0
	Relation	2	0	16	0.2125	0.0166	**12.77**×	590.88	565.03	0
	Exposure	2	0	16	0.2123	0.0167	**12.74**×	590.88	565.03	0
ogbn-products	Relation	1	1	16	7.6537	0.4852	**15.77**×	9,899.71	9,460.70	0
	Exposure	1	1	16	7.6534	0.4855	**15.76**×	9,899.71	9,460.70	0
	Relation	2	0	16	3.8406	0.2468	**15.56**×	9,881.02	9,441.92	0
	Exposure	2	0	16	3.8396	0.2467	**15.56**×	9,881.02	9,441.92	0

End-to-end training efficiency. We compare the runtime of task-only training, counterfactual training with individual interventions, and counterfactual training with locality packing while keeping the matched initializations, intervention schedules, and selected decisions. As shown in Table 13, packing reduces mean counterfactual-training time per epoch by 1.54 times on ogbn-arxiv and 1.52 times on ogbn-products. Its overhead relative to task-only training is 17.1% and 16.8%, respectively, compared with 80.8% and 77.7% for individual-CF.

Table 13: End-to-end training efficiency. Time per training epoch is reported as mean $\pm$ 95% confidence interval over three matched seeds. Runtime overhead is measured relative to task-only training.

Dataset	Training regime	Time per epoch (s)	Speedup	Runtime overhead
ogbn-arxiv	Task only	1.283 ± 0.011	–	–
	Individual-CF	2.320 ± 0.038	$1.00\times$	80.8%
	Packed-CF	1.502 ± 0.000	$\mathbf{1.54}\times$	$\mathbf{17.1}\%$
ogbn-products	Task only	24.762 ± 0.001	–	–
	Individual-CF	43.993 ± 0.010	$1.00\times$	77.7%
	Packed-CF	28.920 ± 0.004	$\mathbf{1.52}\times$	$\mathbf{16.8}\%$

Effect of network depth and remaining radius. We evaluate TriAxis-GCN up to 8 layers on the DBLP and Tolokers datasets using locality packing with a maximum pack size of 16. Figure 7 shows that locality packing reduces counterfactual training overhead at all depths, achieving speedups of 1.77 to 2.10 times on DBLP and 1.40 to 2.26 times on Tolokers compared to individual evaluations. DBLP reaches the pack cap of 16 at every tested radius, while the denser Tolokers graph yields smaller packs at larger radii, which reduces the speedup for deeper models.

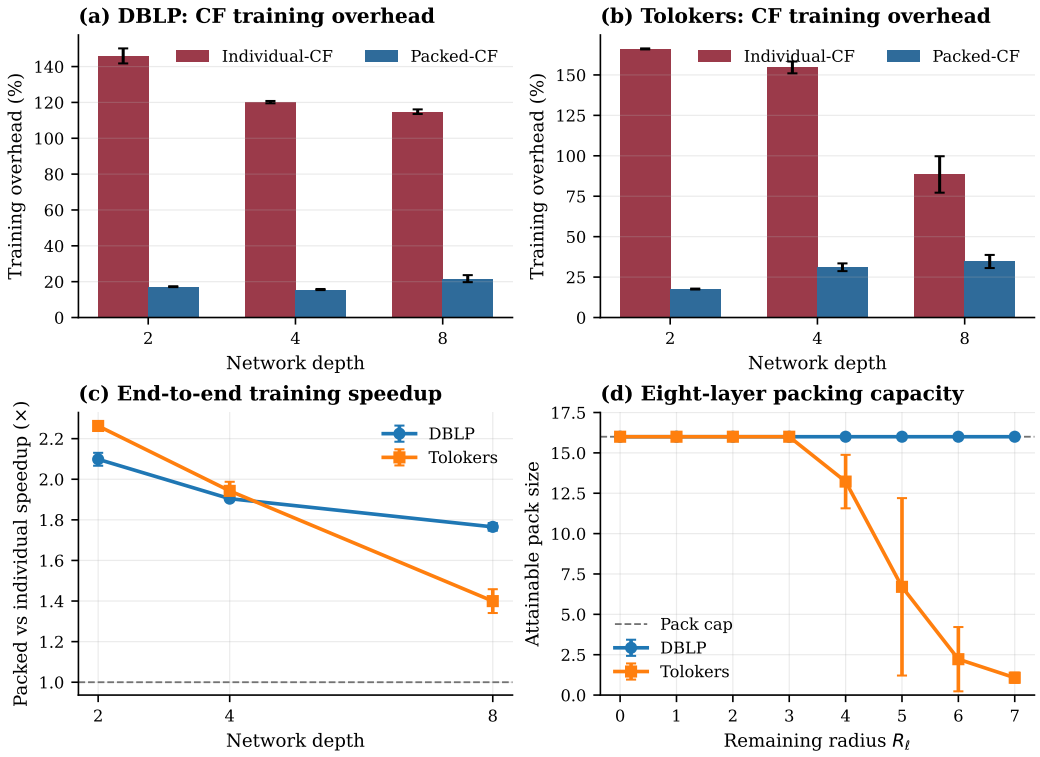

Figure 7: Scalability across network depth.In panel (d), pack sizes are capped at 16. The reported value is the achieved pack size under this cap, so a value of 16 indicates that the cap was reached rather than the maximum attainable pack size.

C.8 Ablation Study on Axis Components in TriAxis-GNN

Figure 8 shows TriAxis-GCN accuracy across all axis subsets on four datasets. Accuracy rises with the number of active axes on every dataset, with the full model always performing best. This indicates the three controls are largely complementary. Relation is the strongest individual axis on the heterophilous datasets, and gains from combining all three axes are much larger there than on the homophilous datasets, consistent with adaptive propagation mattering most where useful neighborhood behavior varies across nodes.

C.9 Ablation Study on Descriptor Components

We ablate the channel-content, channel-alignment, and local-structure components of the TriAxis controller descriptor. Figure 9 compares the full descriptor with variants that remove or retain only one component. The full descriptor achieves the highest mean accuracy on all four datasets, while the importance of individual components varies across datasets. This indicates that the three components provide complementary information.

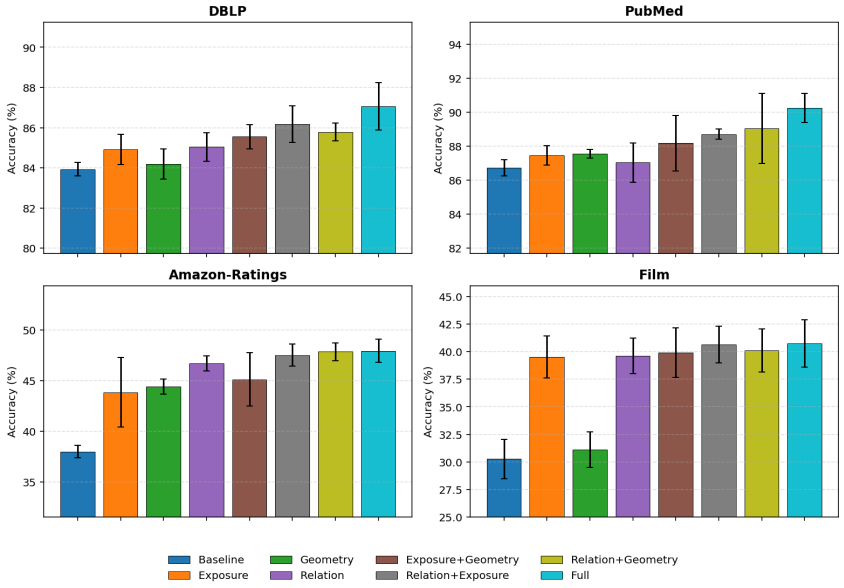

Figure 8: TriAxis-GCN axis-subset ablation. Test accuracy (%, mean with 95% CI over ten seeds) on DBLP, PubMed, Amazon-Ratings, and Film for the zero-axis baseline, each single axis, each pair of axes, and the full model.

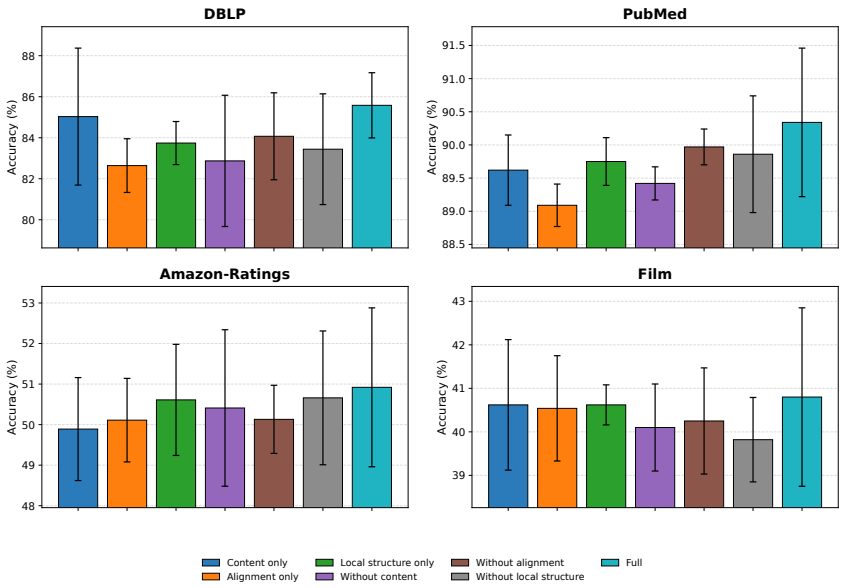

Figure 9: TriAxis-DirGNN descriptor-component ablation. Test accuracy (%, mean with 95% CI over ten seeds) on DBLP, PubMed, Amazon-Ratings, and Film for the full descriptor, each individual component, and variants with one component removed.

C.10 HYPERPARAMETER SENSITIVITY ANALYSIS

Figure 10 examines sensitivity to the controller temperature T, CF update interval T_{CF}, and CF-loss weight λ_{CF} on PubMed and Film datasets on TriAxis-GCN. Accuracy varies moderately across the tested settings, with temperature showing more noticeable effects than the CF-loss weight. Performance remains largely unchanged across the tested values of the CF-loss weight λ_{CF}. Moderately small temperatures, which yield sharper control decisions, tend to perform best. Additionally, frequent counterfactual supervision with update intervals between 1 and 5 consistently performs well.

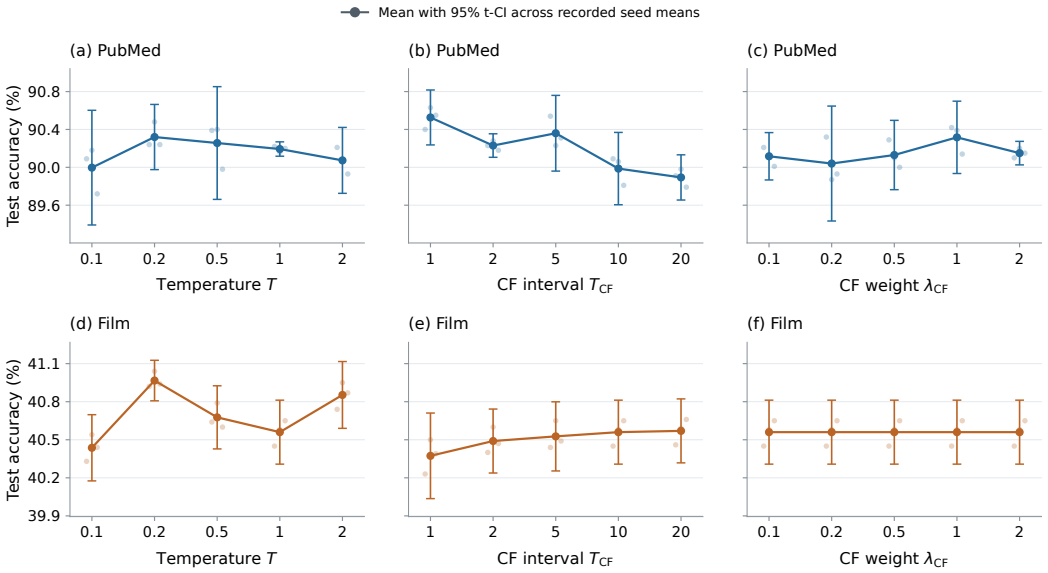

Figure 10: Sensitivity to counterfactual-supervision hyperparameters.

We further examine the sensitivity of counterfactual supervision to the maximum number of receivers per counterfactual update on Mowst. Figure 11 reports the resulting test accuracy on Amazon Photo and PubMed. Across all tested values, mean accuracy remains nearly constant on both datasets, and the confidence intervals overlap throughout. These results indicate that performance is not sensitive to this hyperparameter, so the number of receivers per update can be chosen according to the available computational budget.

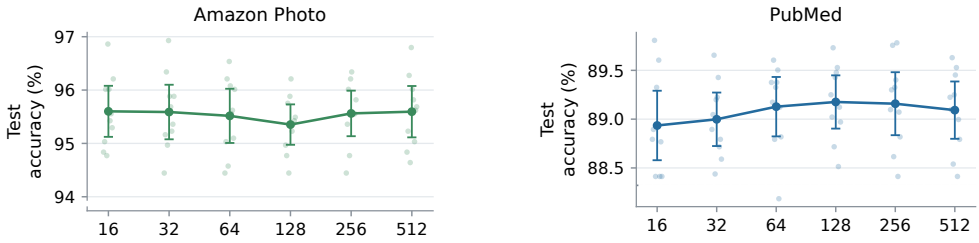

Figure 11: Sensitivity to the number of receivers per counterfactual update.

D TRAINING ALGORITHM

Algorithm 1 summarises the training procedure of the proposed counterfactual supervision.

Algorithm 1 Counterfactual propagation training

Require: Graph G, labels $\{y_i\}_{i\in\mathcal{V}_{\text{tr}}}$, temperature T, CF interval T_{CF}, training steps T_{train}

Initialize θ and ϕ
for $t = 1, \ldots, T_{\text{train}}$ **do**
  Update (θ, ϕ) using $\mathcal{L}_{\text{task}}$ ▷ Standard task update
  if $t \bmod T_{\text{CF}} = 0$ **then** ▷ Perform a periodic CF update
    Select (ℓ, a) and compute representations through layer ℓ
    Form valid candidates $\mathcal{C}$
    Greedily construct $\mathcal{I}_\ell \subseteq \mathcal{C}$ with $\text{dist}_G(i, j) > R_\ell$ ▷ Select non-interfering receivers
    for $b \in \{0, 1\}$ **do**
      Jointly set $c^a_{i\ell} = b$ for all $i \in \mathcal{I}_\ell$
      Recompute the suffix with shared randomness ▷ Compute losses under endpoint b
      Record $L_i^{(b)}$ for all $i \in \mathcal{I}_\ell$
    end for
    Set $u^a_{i\ell} = L_i^{(0)} - L_i^{(1)}$ ▷ Measure the endpoint preference
    if valid decisions remain **then**
      Update only ϕ using $\mathcal{L}_{\text{CF}}$ ▷ Supervise the controller
    end if
  end if
end for
return θ, ϕ