Local Evidence and Geometric Readout Repair in Trained GNNs

Nadi Tomeh and Hugo Attali

Université Sorbonne Paris Nord, CNRS,
Laboratoire d'Informatique de Paris Nord, LIPN,
F-93430 Villetaneuse, France
{tomeh,attali}@lipn.fr

Abstract. Many node-classification GNNs apply a linear classifier to a nonnegative mixture of local messages. An error can reflect either poor mixture weights or a reachable logit set poorly positioned for the classifier. We separate these causes with an exact-mass linear program and two learned post-hoc repairs. Every reweighted prediction has an equivalent centered logit translation, but only translations in a message-induced displacement set are realizable by reweighting. Across eight datasets, eight GNN backbones, and ten splits, mean accuracy rises from 62.6% for the frozen models to 63.8% with reweighting and 65.3% with set-conditioned translation. A parameter-matched node-only translator reaches 64.6%, showing that translation explains most of the gain while the message set supplies a smaller additional benefit. Although oracle reweighting can correct many errors, label-free reweighting captures little of this potential: local evidence is often present but hard to select, and relaxing the evidence constraint is more effective than learning within it.

Keywords: Graph neural networks · Readout geometry · Post-hoc repair

1 Introduction

We study trained message-passing graph neural networks (GNNs) for transductive node classification [27, 8, 16, 30, 11]. Many end with a nonnegative weighted sum of final messages followed by a linear classifier. A local error can therefore arise because the coefficients select an unhelpful mixture or because no admissible mixture of the final messages supports the true class. End-to-end accuracy conflates these causes.

If exact-mass reweighting makes the true class win, the fixed messages already contain sufficient local evidence and only its selection needs repair. If no admissible reweighting succeeds, correction requires changing the messages or another part of the model. This distinction reveals whether a lightweight readout repair can reuse existing evidence or must move beyond the fixed-message polytope, a useful diagnosis when helpful and conflicting messages coexist in heterophilous or noisy graphs [36, 6, 18, 26].

We study the logit set induced by nonnegative reweighting with fixed branch masses. A linear program maximizes a proposed class's worst-rival margin; a positive optimum certifies fixed-message support, and linear-program geometry gives a compact solution. With the GNN frozen, we train two label-free, set-conditioned adapters from the same local information: one reweights within the reachable set, while the other predicts a free centered logit translation. The oracle measures evidence availability; comparing the nested repairs tests label-free selection and the effect of relaxing the evidence constraint. A refitted head and a parameter-matched node-only translator separate these effects from generic readout capacity and from translation that ignores the message set.

2 Fixed-Message Reachability

2.1 Reachable logits and class margins

Fix a node v classified into one of C classes. Partition the adjustable terms in its final aggregation into B_v groups and write its frozen logits as

$$z_v^0 = \beta_v + \sum_{g=1}^{B_v} \sum_{a \in \mathcal{A}_{v,g}} \alpha_{v,a} \ell_{v,a}. \tag{1}$$

Here $\mathcal{A}_{v,g}$ is group g, $\mathcal{A}_v = \bigcup_g \mathcal{A}_{v,g}$, and $K_v = |\mathcal{A}_v|$. Message $h_{v,a} \in \mathbb{R}^{d_g}$ defines the neighbor-logit contribution $\ell_{v,a} = W_g h_{v,a} \in \mathbb{R}^C$, where $W_g \in \mathbb{R}^{C \times d_g}$ is the frozen branch transform and classifier. The fixed term $\beta_v \in \mathbb{R}^C$ collects the classifier bias and any non-adjustable root, residual, or branch-bias contribution, while $\alpha_v \in \mathbb{R}^{K_v}_{\geq 0}$ is the frozen coefficient vector and

$$m_{v,g} = \sum_{a \in \mathcal{A}_{v,g}} \alpha_{v,a} > 0. \tag{2}$$

In GCN and SGC, a indexes an incoming source and $\alpha_{v,a}$ is its normalized propagation coefficient [16, 31]. In GAT and GATv2, a also identifies the head and $\alpha_{v,a}$ is the final attention coefficient [30, 5]. GraphSAGE and GraphConv expose their neighbor branch while keeping the root transform in β_v [11, 23]; MixHop and H2GCN use one group per propagation stream [1, 36]. Appendix B gives every backbone's exact exposure.

Admissible coefficients preserve each group's frozen mass:

$$\begin{aligned} \Delta_v &= \left\{ \lambda \in \mathbb{R}^{K_v}_{\geq 0} : \sum_{a \in \mathcal{A}_{v,g}} \lambda_a = m_{v,g},\ g = 1, \ldots, B_v \right\}, \\ z_v(\lambda) &= \beta_v + \sum_{g=1}^{B_v} \sum_{a \in \mathcal{A}_{v,g}} \lambda_a \ell_{v,a}. \end{aligned} \tag{3}$$

Every feasible λ selects one logit vector; their set is the evidence polytope $\mathcal{L}_v = \{z_v(\lambda) : \lambda \in \Delta_v\}$. The frozen vector α_v is feasible and selects the original output

$z_v^0 = z_v(\alpha_v)$. Exact mass therefore isolates redistribution: it cannot change a branch's scale or transfer mass across attention heads or propagation streams. For one group, $\mathcal{L}_v = \beta_v + m_{v,1}\,\mathrm{conv}\{\ell_{v,a} : a \in \mathcal{A}_v\}$, so it is the scaled convex hull of the individual neighbor-logit contributions. Multiple groups add one point from each such hull. Reweighting moves the selected output within $\mathcal{L}_v$ but cannot change the polytope itself.

The margin of class c against rival r is

$$M_v^\lambda(c,r) = z_{v,c}(\lambda) - z_{v,r}(\lambda) = \beta_{v,c} - \beta_{v,r} + \sum_{g=1}^{B_v} \sum_{a \in \mathcal{A}_{v,g}} \lambda_a(\ell_{v,a,c} - \ell_{v,a,r}), \quad r \neq c, \tag{4}$$

Class c wins exactly when $\min_{r \neq c} M_v^\lambda(c,r) \geq 0$; its region is $K_c = \{z \in \mathbb{R}^C : z_c \geq z_r \ \forall r \neq c\}$. The minimum selects the strongest rival at the chosen point, so a positive value means that c beats all alternatives, not only the frozen model's current prediction.

2.2 Exact-mass oracle and compact solutions

To test whether the fixed messages can support class c, we maximize its worst-rival margin with the linear program (LP)

$$\begin{aligned} \gamma_v(c) = \max_{\lambda,\gamma} \quad & \gamma \\ \text{s.t.} \quad & \gamma \leq M_v^\lambda(c,r) \quad \forall r \neq c, \\ & \lambda \in \Delta_v. \end{aligned} \tag{5}$$

For fixed λ, the largest feasible γ is its worst-rival margin, so the outer maximization finds the best exact-mass weighting. Positive γ gives a strict win, zero reaches only a decision boundary, and negative γ means every weighting loses to some rival. For $c = y_v$, positivity certifies coefficient-only correction; the label is used only in this oracle analysis.

Proposition 1 (Compact exact-mass optimum). *The evidence polytope $\mathcal{L}_v$ intersects the interior of K_c if and only if $\gamma_v(c) > 0$. An optimal basic solution can be chosen with at most $C + B_v - 2$ active terms in total and at most $C - 1$ active terms in any group.*

The proof is in Appendix A. On a misclassified node, $\gamma_v(y_v) > 0$ identifies a selection failure; otherwise the true class requires changing the messages or model. The support bound is independent of neighborhood size: a single-group readout needs at most $C - 1$ terms, while each additional positive-mass group must retain one. A basic optimum thus gives a small correcting set to inspect, but not necessarily a unique explanation or a claim that the original GNN should aggregate only those terms. Compactness is expected from LP geometry [3, 2]; the substantive observation is positive true-class margin, not sparsity alone. Because the tested class is supplied and the optimizer may be nonunique, the LP is an oracle diagnostic rather than a prediction rule. Support counts head- or stream-specific terms separately.

3 Learning Constrained and Free Readout Corrections

We freeze the GNN and learn two adapters from the same local information. Reweighting predicts coefficients and must select a point in $\mathcal{L}_v$; translation predicts a logit displacement that may leave this set. Neither changes the graph, messages, or classifier, so their comparison isolates the cost of restricting a correction to combinations of the frozen messages.

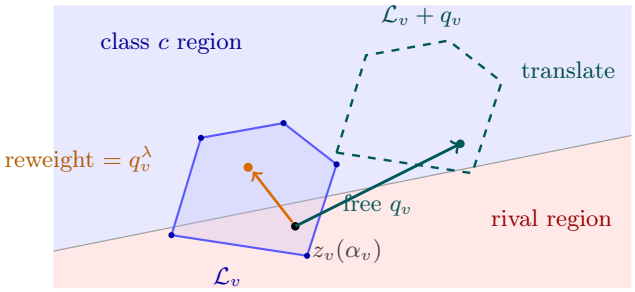

Fig. 1. Reweighting remains in $\mathcal{L}_v$ and has an equivalent centered translation q_v^λ. A free q_v may lie outside the feasible displacements and shifts the whole set (dashed) without changing its shape.

3.1 Shared local encoding

Both adapters receive the same final-stage set $\mathcal{S}_v$: the adjustable messages in $\mathcal{A}_v$ plus exposed fixed terms such as a separate root contribution. Thus $\mathcal{A}_v \subseteq \mathcal{S}_v$. The full set describes the alternatives around v; fixed terms provide context but are not reweighted.

For each $a \in \mathcal{S}_v$, we use the feature vector

$$x_{v,a} = [h_v, h_{v,a}, h_{v,a} - h_v, z_v^0, z_a^0, z_a^0 - z_v^0, \cos(h_v, h_{v,a}), \|h_{v,a} - h_v\|_2] \in \mathbb{R}^{3d+3C+2}, \tag{6}$$

where $z_a^0 \in \mathbb{R}^C$ is the frozen source-node logit vector. These features compare recipient and message in both spaces: relative representation features describe compatibility with v, while frozen logits expose class evidence.

Each term retains its fixed readout weight $\omega_{v,a} \geq 0$: $\omega_{v,a} = \alpha_{v,a}$ for adjustable terms, while an exposed root has weight one. These inputs weight the context summary, not the predicted coefficients. With $\bar{m}_v = \sum_{a \in \mathcal{S}_v} \omega_{v,a}$ and branch index $k \in \{\mathrm{R}, \mathrm{T}\}$, where R denotes reweighting and T translation, each adapter uses

$$p_v^k = \frac{1}{\bar{m}_v} \sum_{a \in \mathcal{S}_v} \omega_{v,a} \psi_k(x_{v,a}), \qquad c_v^k = [h_v, z_v^0, p_v^k, \bar{m}_v]. \tag{7}$$

where $p_v^k \in \mathbb{R}^{d_p}$ and $c_v^k \in \mathbb{R}^{d+C+d_p+1}$. The separately parameterized encoders are permutation invariant and handle variable set sizes; h_v, z_v^0, and $\bar{m}_v$ retain recipient state and scale. Thus p_v^k summarizes the weighted set, and c_v^k augments it with the recipient.

3.2 Readout repair variants

Evidence-constrained correction. The reweighting branch scores each adjustable message in context and tilts its frozen coefficient; groupwise normalization preserves the original mass:

$$\begin{aligned} s_{v,a} &= s_\theta([x_{v,a}, c_v^{\mathrm{R}}]) \in \mathbb{R}, \\ \lambda_{v,a}^\theta &= m_{v,g} \frac{\alpha_{v,a} \exp s_{v,a}}{\sum_{b \in \mathcal{A}_{v,g}} \alpha_{v,b} \exp s_{v,b}}, \qquad a \in \mathcal{A}_{v,g}. \end{aligned} \tag{8}$$

The exponential makes every coefficient nonnegative, and the denominator enforces $\sum_{a \in \mathcal{A}_{v,g}} \lambda_{v,a}^\theta = m_{v,g}$ exactly. Thus $\lambda_v^\theta \in \Delta_v$: the adapter changes the selected point but not the messages, support, or polytope. Zero initialization gives $\lambda_v^\theta = \alpha_v$, and $z_v^{\mathrm{R}} = z_v(\lambda_v^\theta)$. Because the update is multiplicative, a term with zero frozen coefficient remains unavailable; the adapter neither adds edges nor enlarges the original support.

Free logit correction. The translation branch maps c_v^{T} to a node-specific shift. We remove the common-logit component, which does not affect classification:

$$\bar{q}_v = \rho_\phi(c_v^{\mathrm{T}}) \in \mathbb{R}^C, \qquad q_v = \bar{q}_v - \frac{\mathbf{1}^\top \bar{q}_v}{C} \mathbf{1}, \qquad z_v^{\mathrm{T}} = z_v^0 + q_v. \tag{9}$$

Centering removes the unidentifiable all-ones direction of softmax logits. Adding the same q_v to every $z_v(\lambda)$ then translates $\mathcal{L}_v$ rigidly, without changing its shape or the relative position selected by α_v. "Free" means that q_v need not be realizable by reweighting; its magnitude remains bounded in the implementation. This is a residual correction at the frozen readout, not a modification of individual messages or graph edges.

Why reweighting is contained in translation. Let $P = I - \frac{1}{C}\mathbf{1}\mathbf{1}^\top \in \mathbb{R}^{C \times C}$ center a logit vector and define the centered reweighting-displacement set

$$\mathcal{D}_v = P(\mathcal{L}_v - z_v^0) = \{P(z_v(\lambda) - z_v^0) : \lambda \in \Delta_v\}. \tag{10}$$

Proposition 2 (Reweighting is contained in translation). *For every $\lambda \in \Delta_v$, the centered translation $q_v^\lambda = P(z_v(\lambda) - z_v^0)$ makes $z_v^0 + q_v^\lambda$ produce the same softmax probabilities and class prediction as $z_v(\lambda)$. Conversely, a centered translation q_v is equivalent to an exact-mass reweighting if and only if $q_v \in \mathcal{D}_v$.*

For a chosen λ, adding q_v^λ to z_v^0 recovers $z_v(\lambda)$ up to a common offset, hence gives the same softmax. Conversely, $\mathcal{D}_v$ is exactly the set of such feasible displacements. The proposition is pointwise, not an equivalence of learned adapters: reweighting must infer an output in $\mathcal{D}_v$, whereas translation may predict any bounded centered q_v. Since their inputs are the same, an accuracy gap isolates this output constraint rather than a difference in observed information.

3.3 Training the adapters

Each adapter minimizes mean training-node cross-entropy, updating only θ or ϕ while the graph, backbone, messages, support, and classifier remain fixed. "Label-free" refers to prediction: labels supervise training, but the target class is not an input, and test-time repair uses only $\mathcal{S}_v$. Both start as exact no-ops, so validation selection may retain the frozen rule when a repair does not help. Appendix B gives optimization details.

3.4 Relation to existing methods

Graph attention and adaptive propagation learn coefficients jointly with the representations [30, 5, 6], while sparse attention constrains their geometry [22, 24]. We instead freeze the GNN and train small post-hoc modules, following adapter and side-network paradigms [12, 35]. Our adapters use a weighted Deep Sets encoder [34] at the final readout. Unlike calibration [9, 13], the objective is error correction; unlike edge-conditioned or feature-wise message modulation [29, 4], earlier GNN layers remain unchanged.

Compact oracle supports are related to GNN and counterfactual explanations [33, 21, 20], but our LP tests class reachability under fractional exact-mass reweighting rather than explaining a fixed prediction or editing the graph. Graph diffusion and Correct-and-Smooth act through graph-wide prediction propagation [17, 7, 14]; our repairs remain local to the trained final message set.

4 Experiments

We evaluate eight datasets and eight backbones over ten transductive node-classification splits. Validation accuracy selects each backbone checkpoint; the graph, messages, coefficients, and classifier are then frozen for all diagnostics and repairs. Table cells are ten-split means, and macro averages weight the 64 dataset–backbone cases equally. Appendix B gives the datasets, architectures, optimization, and batching details.

4.1 Oracle reachability of frozen errors

For every validation and test node, we solve Eq. (5) for the true class with Gurobi [10], preserving every group mass. Let $\widehat{y}_v^0 = \arg\max_c z_{v,c}^0$. Table 1 reports on test nodes

$$\begin{aligned} \text{TCReach} &= \Pr[\gamma_v(y_v) > 0], \\ \text{ErrReach} &= \Pr[\gamma_v(y_v) > 0 \mid \widehat{y}_v^0 \neq y_v], \\ \text{Support} &= \mathbb{E}[\|\lambda_v^\star\|_0 \mid \gamma_v(y_v) > 0]. \end{aligned} \tag{11}$$

TC reach covers all nodes, Err. reach conditions on frozen-model mistakes, and Support counts the active coefficients in one positive-margin basic optimum. The test label selects the LP target, so these are oracle diagnostics rather than predictions. A nonpositive score rules out correction within this frozen final readout;

it does not rule out useful information elsewhere in the graph. Appendix B gives numerical thresholds and solver details.

Table 1. Exact-mass frozen-message oracle diagnostics over $n = 10$ splits, with backbones in rows and datasets in columns. TC reach and Err. reach are percentages; Support is the mean number of active coefficients in positive true-class solutions.

(a) True-class reach (%)

Model	Actor	Amazon	Cham.	Cite.	Cora	PubMed	Roman	Squir.	Avg.
GCN	62.2	69.4	94.0	80.8	95.3	95.6	78.8	94.6	83.8
GAT	59.3	62.2	92.1	81.1	94.5	94.1	70.1	91.0	80.5
SAGE	52.2	68.4	78.4	79.7	94.1	93.2	92.8	66.2	78.1
GATv2	62.7	66.8	92.9	80.7	94.6	94.1	83.3	92.2	83.4
GraphConv	56.4	65.5	72.4	80.7	93.9	94.0	93.8	46.5	75.4
SGC	94.0	87.8	99.6	89.2	99.1	99.6	96.6	100.0	95.7
MixHop	72.2	87.9	91.0	86.5	97.9	97.6	97.7	68.8	87.4
H2GCN	79.7	88.5	96.2	83.3	96.4	96.8	96.1	91.0	91.0
Avg.	67.3	74.6	89.6	82.7	95.7	95.6	88.7	81.3	84.4

(b) Error reach (%)

Model	Actor	Amazon	Cham.	Cite.	Cora	PubMed	Roman	Squir.	Avg.
GCN	45.8	43.8	80.8	20.4	60.4	59.2	54.1	88.6	56.6
GAT	42.2	31.2	77.5	22.8	55.4	53.0	43.2	82.0	50.9
SAGE	26.1	39.7	45.0	17.8	51.7	32.3	65.6	41.2	39.9
GATv2	46.8	37.2	78.8	22.0	56.3	51.3	53.7	83.9	53.7
GraphConv	37.7	38.9	33.6	23.2	53.9	43.8	69.7	25.7	40.8
SGC	91.5	78.8	98.9	54.7	92.8	96.3	93.6	99.9	88.3
MixHop	57.5	79.1	79.7	42.3	84.4	77.7	90.0	51.6	70.3
H2GCN	68.7	80.0	90.5	32.6	72.9	69.5	83.6	84.7	72.8
Avg.	52.0	53.6	73.1	29.5	66.0	60.4	69.2	69.7	59.2

(c) Active support

Model	Actor	Amazon	Cham.	Cite.	Cora	PubMed	Roman	Squir.	Avg.
GCN	1.40	1.22	1.65	1.07	1.12	1.11	1.95	1.79	1.41
GAT	1.40	1.22	1.56	1.11	1.19	1.10	1.82	1.80	1.40
SAGE	1.25	1.30	1.33	1.08	1.12	1.05	1.61	1.33	1.26
GATv2	1.43	1.25	1.52	1.11	1.18	1.09	1.69	1.71	1.37
GraphConv	1.50	1.27	1.38	1.09	1.13	1.13	1.81	1.29	1.32
SGC	2.20	1.60	2.24	1.48	1.79	1.26	2.31	2.83	1.96
MixHop	3.52	3.64	3.65	3.27	3.52	3.10	4.41	3.60	3.59
H2GCN	3.68	3.71	3.90	3.21	3.43	3.11	4.15	3.86	3.63
Avg.	2.05	1.90	2.15	1.68	1.81	1.62	2.47	2.27	1.99

Across the 64 dataset–backbone cases, macro true-class reach is 84.4% and error reach is 59.2%. Thus many errors select the wrong point from an adequate local set, while about two fifths are not strictly correctable under the exact-mass constraints. Error reach ranges from 39.9% for GraphSAGE to 88.3% for SGC and from 29.5% on Citeseer to 73.1% on Chameleon. A positive solution uses 1.99 active terms on average; MixHop and H2GCN use more because every stream retains a term. This compact support aids inspection, but reachability is a label-aware evidence diagnostic, not a deployable predictor.

4.2 Accuracy and controls for the learned repairs

Table 2 compares the frozen model with learned reweighting and set-conditioned translation for every dataset and backbone.

Table 2. Mean test accuracy (%) over ten splits. Rows are backbones and columns are datasets; bold marks the best learned repair when it improves on Frozen.

(a) Frozen									
Model	Actor	Amazon	Cham.	Cite.	Cora	PubMed	Roman	Squir.	Avg.
GCN	30.2	45.6	68.4	75.9	88.1	89.2	53.8	52.7	63.0
GAT	29.5	45.0	64.9	75.4	87.8	87.5	47.4	50.3	61.0
SAGE	35.3	47.6	60.8	75.3	87.8	89.9	79.2	42.4	64.8
GATv2	29.8	47.1	66.6	75.2	87.7	87.8	64.0	51.7	63.7
GraphConv	30.2	43.7	58.4	74.9	86.9	89.3	79.6	28.4	61.4
SGC	29.6	42.4	67.5	76.2	88.6	88.1	46.5	50.0	61.1
MixHop	34.5	42.0	55.4	76.7	86.6	89.4	76.9	35.7	62.1
H2GCN	35.2	42.5	60.3	75.2	86.8	89.4	76.3	41.0	63.3
Avg.	31.8	44.5	62.8	75.6	87.5	88.8	65.5	44.0	62.6
(b) Reweight									
Model	Actor	Amazon	Cham.	Cite.	Cora	PubMed	Roman	Squir.	Avg.
GCN	30.5	45.6	**68.5**	**76.0**	**88.1**	89.3	59.6	54.2	64.0
GAT	**30.0**	45.1	64.9	75.6	87.8	88.5	49.9	52.3	61.8
SAGE	**35.4**	47.6	60.7	**75.5**	87.9	90.0	**79.8**	**42.6**	64.9
GATv2	30.0	47.0	66.8	**75.3**	**87.8**	88.4	69.6	52.1	64.6
GraphConv	30.2	43.7	58.4	74.8	86.9	89.3	**80.5**	29.1	61.6
SGC	33.1	48.9	**67.8**	76.1	**88.6**	89.6	67.5	**51.1**	65.3
MixHop	34.4	46.5	**56.8**	76.2	**87.4**	**89.8**	79.5	**37.2**	63.5
H2GCN	35.0	47.3	**61.9**	75.2	86.8	**89.5**	78.1	**42.6**	64.5
Avg.	32.3	46.5	63.2	75.6	87.7	89.3	70.6	45.2	63.8
(c) Set translation									
Model	Actor	Amazon	Cham.	Cite.	Cora	PubMed	Roman	Squir.	Avg.
GCN	**30.8**	**45.8**	68.4	75.9	88.1	**89.4**	**76.3**	**57.3**	66.5
GAT	30.0	**46.4**	**67.3**	**75.6**	**88.0**	**88.8**	**76.2**	**54.7**	65.9
SAGE	35.4	**47.7**	**61.1**	75.3	**88.0**	**90.0**	79.1	42.4	64.9
GATv2	**30.2**	**47.1**	**67.1**	75.0	87.4	**88.7**	**80.7**	**53.3**	66.2
GraphConv	**30.5**	**43.8**	**62.1**	74.8	**87.1**	**89.3**	79.8	**38.0**	63.2
SGC	**35.3**	**49.4**	67.8	76.1	88.3	**89.7**	**78.7**	50.3	67.0
MixHop	**34.8**	**49.2**	56.0	**76.8**	86.5	89.8	**81.0**	36.0	63.7
H2GCN	**35.5**	**50.0**	60.9	**75.3**	**86.9**	89.3	**79.0**	41.2	64.8
Avg.	32.8	47.4	63.8	75.6	87.5	89.4	78.9	46.6	**65.3**

To test whether these gains only reflect additional post-hoc capacity, Table 3 adds two controls. Refit head learns a new affine classifier on the frozen final embeddings and tests whether moving the linear decision boundary is sufficient. Node translation keeps the original classifier and predicts the same bounded centered-logit correction as set translation, but from $[h_v, z_v^0]$ alone. Its hidden width is chosen separately in every case to match the set translator's parameter count within 1%. The comparison between the two translators therefore tests the value of the message-set input at matched output space and capacity. Appendix C gives the architectures, optimization protocol, and complete 64-case results.

Table 3. Capacity and information controls. Test accuracy is the macro mean over all 64 dataset–backbone cases; every case averages ten splits. The node translator is parameter matched to the set translator within 1% in every case.

Method	Learned output	Message set	Accuracy	Δ Frozen
Frozen	none	no	62.6	–
Refit head	linear head	no	62.1	-0.5
Node translation	centered shift	no	64.6	+2.1
Set reweight	coefficients	yes	63.8	+1.2
Set translation	centered shift	yes	**65.3**	**+2.7**

The 64-case mean rises from 62.6% to 63.8% with reweighting, far below the oracle opportunity in Table 1: the oracle sees the target class, whereas the adapter must infer weights without it.

Set translation reaches 65.3% and improves on Frozen in 53 cases. Gains are largest on Roman-empire (+13.4 points), Amazon-ratings (+2.9), and Squirrel (+2.6), but negligible on the citation graphs. Its 1.5-point advantage over reweighting reflects freedom to leave $\mathcal{D}_v$; inside $\mathcal{D}_v$, the two corrections are pointwise equivalent (Proposition 2).

The controls refine this conclusion. Refitting the head decreases the macro mean by 0.5 points, while node translation gains 2.1 points despite having no message-set input. Translation itself therefore explains most of the improvement. Set conditioning adds another 0.63 points and wins in 40 of 64 cases under parameter matching. This increment is concentrated but not exclusive to Roman-empire: it is +3.38 points there, +0.96 on Squirrel, and +0.24 over the 56 non-Roman cases. Thus the full local set can help choose the correction direction, but it is not required for every graph or backbone.

5 Conclusion

The exact-mass oracle shows that many errors have a compact true-class combination, but learned reweighting captures little of this opportunity. Free translation performs better because it can express corrections outside the fixed-message polytope. The controls show that a node-only translation already captures most of this benefit, while the full message set provides a smaller additional gain at matched capacity. Local evidence therefore has two roles: it defines the reweighting diagnostic and can improve how a free correction direction is selected. Future work may learn stable near-optimal sets rather than one LP solution and test larger structured relaxations.

References

1. Abu-El-Haija, S., Perozzi, B., Kapoor, A., Alipourfard, N., Lerman, K., Harutyunyan, H., Ver Steeg, G., Galstyan, A.: MixHop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In: Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97, pp. 21–29 (2019)
2. Bertsimas, D., Tsitsiklis, J.N.: Introduction to Linear Optimization. Athena Scientific, Belmont, MA (1997)
3. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press (2004)
4. Brockschmidt, M.: GNN-FiLM: Graph neural networks with feature-wise linear modulation. In: International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 119, pp. 1144–1153 (2020)
5. Brody, S., Alon, U., Yahav, E.: How attentive are graph attention networks? In: International Conference on Learning Representations (2022), https://openreview.net/forum?id=F72ximsx7C1

6. Chien, E., Peng, J., Li, P., Milenkovic, O.: Adaptive universal generalized PageRank graph neural network. In: International Conference on Learning Representations (2021)
7. Gasteiger, J., Weißenberger, S., Günnemann, S.: Diffusion improves graph learning. In: Advances in Neural Information Processing Systems 32. pp. 13333–13345 (2019)
8. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: Proceedings of the 34th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 70, pp. 1263–1272 (2017)
9. Guo, C., Pleiss, G., Sun, Y., Weinberger, K.Q.: On calibration of modern neural networks. In: Proceedings of the 34th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 70, pp. 1321–1330 (2017)
10. Gurobi Optimization, LLC: Gurobi Optimizer Reference Manual (2026), https://www.gurobi.com
11. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems 30 (2017)
12. Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., Gelly, S.: Parameter-efficient transfer learning for NLP. In: Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97, pp. 2790–2799 (2019)
13. Hsu, H.H.H., Shen, Y., Tomani, C., Cremers, D.: What makes graph neural networks miscalibrated? In: Advances in Neural Information Processing Systems 35 (2022)
14. Huang, Q., He, H., Singh, A., Lim, S.N., Benson, A.R.: Combining label propagation and simple models out-performs graph neural networks. In: International Conference on Learning Representations (2021)
15. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: International Conference on Learning Representations (2015)
16. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: International Conference on Learning Representations (2017)
17. Klicpera, J., Bojchevski, A., Günnemann, S.: Predict then propagate: Graph neural networks meet personalized PageRank. In: International Conference on Learning Representations (2019)
18. Lim, D., Hohne, F., Li, X., Huang, S.L., Gupta, V., Bhalerao, O., Lim, S.N.: Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. In: Advances in Neural Information Processing Systems 34 (2021)
19. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: International Conference on Learning Representations (2019)
20. Lucic, A., ter Hoeve, M.A., Tolomei, G., de Rijke, M., Silvestri, F.: Cf-gnnexplainer: Counterfactual explanations for graph neural networks. In: Proceedings of The 25th International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 151, pp. 4499–4511 (2022)
21. Luo, D., Cheng, W., Xu, D., Yu, W., Zong, B., Chen, H., Zhang, X.: Parameterized explainer for graph neural network. In: Advances in Neural Information Processing Systems 33 (2020)
22. Martins, A.F.T., Astudillo, R.: From softmax to sparsemax: A sparse model of attention and multi-label classification. In: Proceedings of the 33rd International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 48, pp. 1614–1623 (2016)

23. Morris, C., Ritzert, M., Fey, M., Hamilton, W.L., Lenssen, J.E., Rattan, G., Grohe, M.: Weisfeiler and leman go neural: Higher-order graph neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 4602–4609 (2019)
24. Niculae, V., Martins, A.F.T., Blondel, M., Cardie, C.: A regularized framework for sparse and structured neural attention. In: Advances in Neural Information Processing Systems 30 (2017)
25. Pei, H., Wei, B., Chang, K.C.C., Lei, Y., Yang, B.: Geom-GCN: Geometric graph convolutional networks. In: International Conference on Learning Representations (2020), https://openreview.net/forum?id=S1e2agrFvS
26. Platonov, O., Kuznedelev, D., Diskin, M., Babenko, A., Prokhorenkova, L.: A critical look at the evaluation of GNNs under heterophily: Are we really making progress? In: International Conference on Learning Representations (2023), https://openreview.net/forum?id=tJbbQfw-5wv
27. Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: The graph neural network model. IEEE Transactions on Neural Networks **20**(1), 61–80 (2009). https://doi.org/10.1109/TNN.2008.2005605
28. Shchur, O., Mumme, M., Bojchevski, A., Günnemann, S.: Pitfalls of graph neural network evaluation (2018), neurIPS Workshop on Relational Representation Learning
29. Simonovsky, M., Komodakis, N.: Dynamic edge-conditioned filters in convolutional neural networks on graphs. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3693–3702 (2017)
30. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: International Conference on Learning Representations (2018)
31. Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., Weinberger, K.Q.: Simplifying graph convolutional networks. In: Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97, pp. 6861–6871 (2019)
32. Yang, Z., Cohen, W.W., Salakhutdinov, R.: Revisiting semi-supervised learning with graph embeddings. In: Proceedings of the 33rd International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 48, pp. 40–48 (2016)
33. Ying, Z., Bourgeois, D., You, J., Zitnik, M., Leskovec, J.: GNNExplainer: Generating explanations for graph neural networks. In: Advances in Neural Information Processing Systems 32 (2019)
34. Zaheer, M., Kottur, S., Ravanbakhsh, S., Póczos, B., Salakhutdinov, R., Smola, A.J.: Deep sets. In: Advances in Neural Information Processing Systems 30 (2017)
35. Zhang, J.O., Sax, A., Zamir, A., Guibas, L.J., Malik, J.: Side-tuning: A baseline for network adaptation via additive side networks. In: European Conference on Computer Vision. pp. 698–714 (2020). https://doi.org/10.1007/978-3-030-58580-8_41
36. Zhu, J., Yan, Y., Zhao, L., Heimann, M., Akoglu, L., Koutra, D.: Beyond homophily in graph neural networks: Current limitations and effective designs. In: Advances in Neural Information Processing Systems 33 (2020)

A Proof of Proposition 1

For any $\lambda \in \Delta_v$, the rth class-c margin coordinate is

$$z_{v,c}(\lambda) - z_{v,r}(\lambda) = M_v^{\lambda}(c, r), \qquad r \neq c. \tag{12}$$

Consequently, $\gamma_v(c) > 0$ holds exactly when one feasible logit has every class-c margin positive, which is equivalent to intersection with the interior of K_c.

The feasible coefficient set is compact, so the LP has an optimal basic solution. The full LP has $K_v + 1$ variables: K_v coefficients and γ. If k coefficients are positive, the other $K_v - k$ nonnegativity constraints are active. The B_v exact-mass equalities are linearly independent because the groups are disjoint and have positive mass, and there are at most $C - 1$ active rival-margin constraints. A basic solution requires $K_v + 1$ linearly independent active constraints, hence

$$K_v + 1 \leq (K_v - k) + B_v + (C - 1), \qquad \text{hence} \qquad k \leq C + B_v - 2. \tag{13}$$

Every positive-mass group contains at least one active coefficient. Removing the mandatory active term from each of the other $B_v - 1$ groups yields $k_g \leq k - (B_v - 1) \leq C - 1$ for each group g. This proves both support bounds.

B Experimental Details

B.1 Datasets, preprocessing, and splits

Table 4 summarizes the eight node-classification datasets. Cora, Citeseer, and PubMed follow the Planetoid collection [32]; Actor, Chameleon, and Squirrel follow the Geom-GCN collection [25]; and Amazon-ratings and Roman-empire come from the Heterophilous Graph Benchmark [26]. E counts directed entries after preprocessing. For every dataset, we cast the provided node features to floating point without further normalization, symmetrize the graph, coalesce duplicate directed entries, and add one self-loop per node. The resulting graph and all node features are available during transductive training and evaluation.

Table 4. Dataset statistics after graph preprocessing. Split percentages are train/validation/test; "provided" denotes the ten masks distributed with the dataset.

Dataset	N	E	Features	Classes	Ten-split protocol
Actor	7,600	61,011	932	5	provided, 48/32/20
Amazon-ratings	24,492	210,592	300	5	provided, 50/25/25
Chameleon	2,277	65,069	2,325	5	provided, 48/32/20
Citeseer	3,327	12,431	3,703	6	random, 60/20/20
Cora	2,708	13,264	1,433	7	random, 60/20/20
PubMed	19,717	108,365	500	3	random, 60/20/20
Roman-empire	22,662	88,516	300	18	provided, 50/25/25
Squirrel	5,201	402,047	2,089	5	provided, 48/32/20

For Actor, Amazon-ratings, Chameleon, Roman-empire, and Squirrel, we use provided split columns 0–9. Citeseer, Cora, and PubMed provide only one standard mask in this data interface, so we create ten class-stratified splits with seeds 100–109 and ratios 60/20/20. Every reported entry averages these ten splits, which addresses the known split sensitivity of GNN evaluation [28, 26]. Model and adapter initialization uses seed 0 independently on every split. Training labels fit the backbone and adapter, validation accuracy selects checkpoints, and test labels are used only for final accuracy and the explicitly labeled oracle analysis.

B.2 Backbone architectures and training

We evaluate GCN [16], GAT [30], GraphSAGE [11], GATv2 [5], GraphConv [23], SGC [31], MixHop [1], and H2GCN [36]. All use hidden width 128. GCN, GAT, GraphSAGE, GATv2, and GraphConv have two message-passing layers. GAT and GATv2 use eight concatenated heads in the first layer and one non-concatenated output head. SGC is one linear filter with two propagation steps, MixHop is one layer with powers $\{0, 1, 2\}$, and our H2GCN-style layer concatenates root, one-hop mean, and two-hop mean channels. ReLU and dropout are applied between message-passing layers; attention dropout is also applied inside GAT and GATv2.

All backbones minimize training-node cross-entropy with Adam [15], learning rate 10^{-2}, weight decay 5×10^{-4}, and gradient clipping at 5.0. The dropout argument is 0.5. GCN, GAT, and GraphSAGE run for 200 epochs. The five added backbones run for at most 1000 epochs and stop after 100 epochs without a strict validation-accuracy improvement. In both cases, the checkpoint with the highest validation accuracy is retained. Backbone parameters, including the classifier, are subsequently frozen.

B.3 Final-stage term exposure

Table 5 specifies how Eq. (1) is constructed for each architecture. A higher-order source–recipient entry is the corresponding nonzero entry after sparse matrix multiplication; multiple paths to the same source are therefore combined into one coefficient. The translation adapter additionally receives fixed root terms for GraphSAGE and GraphConv as context with weight one, although those terms remain in β_v and are not reweighted.

Table 5. Architecture-faithful exposure of the final aggregation. Every group preserves its own frozen coefficient mass.

Backbone	Adjustable term and frozen coefficient	Groups	Fixed contribution in β_v
GCN	Normalized final-layer edge/self-loop term; $\alpha_{v,a} = \widehat{A}_{va}$	1	Convolution and classifier biases
GAT/GATv2	Final attention edge–head copy; $\alpha_{v,a}$ is its attention coefficient	1	Residual, layer bias, and classifier bias
GraphSAGE	Neighbor/self-loop term; $\alpha_{v,a} = 1/\deg(v)$	1	Separate root transform, neighbor-branch bias, and classifier bias
GraphConv	Neighbor/self-loop term; $\alpha_{v,a} = 1$	1	Separate root transform, neighbor-branch bias, and classifier bias
SGC	Nonzero entry of $\widehat{A}^2$	1	Linear and classifier biases
MixHop	Nonzero entry of $\widehat{A}^p$, $p \in \{0, 1, 2\}$	3	Concatenated-stream and classifier biases
H2GCN	Nonzero entry of I, P, or P^2, with P the no-self mean operator	3	Concatenated-stream and classifier biases

B.4 Repair adapters and optimization

The term encoder ψ_k, reweighting scorer s_θ, and translation head ρ_ϕ are two-layer MLPs with hidden width $H = 128$, ReLU, and dropout 0.2; the pooled representation has dimension $d_p = 128$. The final affine layer of each output head is initialized to zero. If $\widehat{s}_{v,a}$ and $\widehat{q}_v$ denote its unbounded outputs, the implementation uses $s_{v,a} = 4\tanh(\widehat{s}_{v,a})$ in Eq. (8) and centers $4\tanh(\widehat{q}_v)$ as in Eq. (9). Reweighting normalizes separately within every group, so its output satisfies the exact mass constraints throughout training.

The two adapters use the same width and feature definition but are not parameter matched. For pair-feature dimension $d_x = 3d + 3C + 2$, define the parameter count of a two-layer width-H MLP by $F(i, o) = iH + H + Ho + o$. Translation has $F(d_x, H) + F(d+C+H+1, C)$ parameters, whereas reweighting has $F(d_x, H) + F(d_x+d+C+H+1, 1)$. Across the reported cases these range from 101K–1.93M and 150K–3.36M parameters, respectively. We do not tune depth or width separately, so the comparison concerns these common architectures rather than capacity-matched optima.

Both adapters minimize mean training-node cross-entropy with AdamW [19], weight decay 10^{-4}, and gradient clipping at 5.0. Translation uses learning rate 10^{-3}, at most 500 epochs, and patience 80; reweighting uses learning rate 5×10^{-4}, at most 300 epochs, and patience 60. Validation accuracy selects the checkpoint, with the frozen prediction retained as epoch zero. Zero output makes

translation an exact no-op and returns the original coefficient rule for reweighting; its separate sparse accumulation may differ from the saved frozen logits only by floating-point accumulation order.

Edge terms are evaluated in chunks without truncating any local set: chunks contain 65,536 terms on V100 cases and 32,768 terms on A100 cases. This chunking only changes accumulation order. SGC, MixHop, and H2GCN additionally use target-node minibatches, with batch size 8 for Squirrel, 16 for Amazon-ratings, and 64 otherwise; every target still includes its complete incoming term set. These cases use minibatch AdamW updates to the same empirical cross-entropy objective. The remaining cases use one full-batch update per epoch.

B.5 LP implementation and numerical checks

For every validation and test node, we solve Eq. (5) on the complete exposed term set without a node or neighborhood cap. Gurobi uses one thread, dual simplex (`Method=1`), disabled presolve, and its default feasibility and optimality tolerances. This returns a basic optimum from which support is measured. We reconstruct the fixed term directly from checkpoint logits,

$$\beta_v = z_v^0 - \sum_a \alpha_{v,a}\ell_{v,a}, \tag{14}$$

which retains all non-adjustable contributions and floating-point accumulation remainders and enforces $z_v(\alpha_v) = z_v^0$ exactly.

Numerically, a node is reachable when $\gamma_v(y_v) > 10^{-7}$ and a coefficient is active when $\lambda_{v,a} > 10^{-7}$. TC reach is this event over test nodes; Err. reach conditions it on errors of the frozen model; Support averages the active coefficients of reachable solutions. Head- or stream-specific copies of one source count as different terms. No solver failure or violation of the support bound in Proposition 1 was observed.

C Full Capacity-Control Results

Every control starts from the same trained checkpoint and train/validation/test masks as the corresponding repair experiment. The graph and GNN backbone are always frozen. The controls alter only the final prediction rule and are fitted from training-node labels; validation accuracy selects the checkpoint, and test labels never affect selection and only score the selected checkpoint.

Refitted linear head. Let $r_v \in \mathbb{R}^{d_f}$ be the frozen node embedding immediately before the original affine classifier. We standardize each coordinate using its training-node mean and population standard deviation; coordinates with standard deviation below 10^{-6} are divided by one. A newly initialized affine map $W_{\mathrm{L}} \in \mathbb{R}^{C\times d_f}$, $b_{\mathrm{L}} \in \mathbb{R}^C$ is then trained on these standardized embeddings. No GNN representation, graph coefficient, or message is updated. This control asks whether the frozen representation only needs a different linear boundary; it is intentionally a small linear model, with $C(d_f + 1)$ trainable parameters, rather than a capacity-matched translator.

Parameter-matched node translation. The second control retains the original logits and predicts

$$q_v^{\mathrm{N}} = 4P\tanh\big(\rho_{\mathrm{N}}([h_v, z_v^0])\big), \qquad z_v^{\mathrm{N}} = z_v^0 + q_v^{\mathrm{N}}, \tag{15}$$

where P is the centering matrix from Eq. (10). ρ_{N} has two affine layers with ReLU and dropout 0.2 between them; its final layer is initialized to zero. Thus it has the same centered, bounded residual output and exact no-op initialization as set translation, but it does not receive the message features, their weights, the pooled set vector, or the total set mass. The only inputs are the frozen recipient representation h_v and original logits z_v^0.

For node-control width H_{N}, its parameter count is

$$N_{\mathrm{N}}(H_{\mathrm{N}}) = H_{\mathrm{N}}(d + 2C + 1) + C, \qquad H_{\mathrm{N}} = \arg\min_{h\in\mathbb{N}} |N_{\mathrm{N}}(h) - N_{\mathrm{T}}|, \tag{16}$$

where N_{T} is the actual trainable-parameter count of the set-conditioned translator in that dataset–backbone case. The resulting models contain 101K–1.93M parameters, and the largest relative mismatch in all 64 cases is 0.083%. Matching is performed independently for each case because d and C vary.

Both controls use AdamW with learning rate 10^{-3}, weight decay 10^{-4}, gradient clipping at 5.0, at most 1100 epochs, and patience 200 on strict validation-accuracy improvement. This budget is at least as generous as the set translator's 500 epochs and patience 80. Node translation retains the frozen prediction as epoch zero, so it cannot be selected below Frozen on the validation set; the refitted head is selected among its trained affine checkpoints. Node translation follows the same target-node batching schedule as set translation, while the linear head is fitted full batch.

Table 6 gives the complete matrices corresponding to the macro comparison in Table 3. Each cell averages the same ten splits as the main repair table.

Table 6. Full capacity-control results. Entries are mean test accuracy (%) over ten splits; rows are backbones and columns are datasets.

(a) Refit head

Model	Actor	Amazon	Cham.	Cite.	Cora	PubMed	Roman	Squir.	Avg.
GCN	28.3	44.9	67.7	76.2	87.1	88.6	51.8	51.3	62.0
GAT	28.7	44.7	64.3	75.4	87.1	87.3	45.7	49.5	60.3
SAGE	34.8	46.2	58.9	75.6	87.5	89.2	78.4	41.9	64.1
GATv2	28.6	46.3	65.8	75.2	87.3	87.6	63.0	52.0	63.2
GraphConv	29.7	42.4	57.9	74.1	86.5	88.9	78.7	31.5	61.2
SGC	28.4	42.0	66.7	75.9	87.7	87.7	45.4	49.7	60.5
MixHop	32.9	41.2	56.7	76.5	86.4	89.2	76.2	39.2	62.3
H2GCN	33.1	41.7	60.8	75.4	86.1	88.9	75.0	42.9	63.0
Avg.	30.6	43.7	62.3	75.5	87.0	88.4	64.3	44.7	62.1

(b) Node translation

Model	Actor	Amazon	Cham.	Cite.	Cora	PubMed	Roman	Squir.	Avg.
GCN	30.1	45.9	68.9	76.4	88.2	89.2	68.1	56.8	65.4
GAT	29.7	47.4	67.6	75.7	88.0	87.9	68.2	53.9	64.8
SAGE	35.4	47.7	61.4	75.5	87.9	89.9	79.2	42.3	64.9
GATv2	29.8	47.4	67.2	75.1	87.7	88.0	77.0	53.0	65.7
GraphConv	30.0	43.6	58.5	74.9	86.9	89.3	79.6	32.8	61.9
SGC	34.8	50.0	67.6	76.0	88.5	89.8	75.8	50.0	66.6
MixHop	34.6	49.7	55.8	76.6	86.6	89.5	78.4	35.7	63.4
H2GCN	35.1	50.1	60.1	75.2	86.8	89.5	77.5	41.0	64.4
Avg.	32.4	47.7	63.4	75.7	87.6	89.1	75.5	45.7	64.6