LEARNED EDGE TARGETS FOR GRAPH-COUPLED NEURAL LAYERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph regularization commonly encourages neighboring representations to agree. Endpoint differences can also inform prediction. We introduce **L**earned **E**dge targets for **G**raph-c**O**upled neural layers (LEGO), which learn preferred differences and compute coupled node representations through a fixed number of primal-dual updates. Learning the targets improves fixed total variation (TV) blocks on five low-label transductive benchmarks and on deduplicated Chameleon and Squirrel, and also benefits GCN and GAT. We also learn the response shape with a radial potential (LP). To explain these effects, we decompose the targets into integrable and cycle components. Integrable targets translate node coordinates at every update; cycle components can affect the output through heterogeneous edge responses. Anchor injection and integrable-target controls recover much of the predictive gain. Interventions at frozen checkpoints also indicate that predictions depend on cycle components and edge correspondence. We derive a perturbation bound relating radial optimality residuals to error in the unrolled output. Code is available at `https://anonymous.4open.science/r/LEGO-977D`.

1 INTRODUCTION

A graph neural network (GNN) combines neighborhood propagation with nonlinear feature transformations (Gilmer et al., 2017; Kipf and Welling, 2017; Hamilton et al., 2017). Graph regularization often encourages neighboring representations to agree (Li et al., 2018; Oono and Suzuki, 2020). On heterophilic graphs, adjacent nodes frequently have different labels, making their differences relevant to prediction (Zhu et al., 2020; Platonov et al., 2023). We study how a graph-coupled nonlinear block can learn preferred endpoint differences for low-label transductive node prediction.

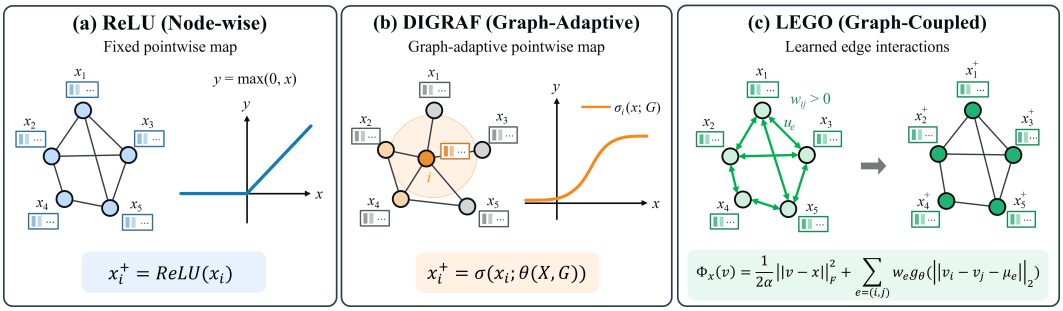


Figure 1: Nonlinear computation on graphs. **(a)** ReLU applies a fixed node-wise map, $x_i^+ = \mathrm{ReLU}(x_i)$. **(b)** DIGRAF (Mantri et al., 2024) applies a graph-adaptive pointwise map, $x_i^+ = \sigma(x_i; \theta(X, G))$. **(c)** LEGO couples node outputs through learned edge interactions. Each edge $e = (i, j)$ has a positive weight w_e and a learned target μ_e (green arrows), the preferred value of the endpoint difference $v_i - v_j$. The outputs x^+ are computed by a fixed number of primal-dual updates on the anchored objective $\Phi_x(v)$.

LEGO places a learned target at the center of each edge penalty. Figure 1 compares its coupled outputs with fixed and graph-adaptive pointwise maps. We predict positive weights and preferred

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.

differences from endpoint features and use a radial potential to penalize deviations from the targets. We then adjust the representations around a propagated node anchor by alternating edge responses and node updates.

Our anchored objective is related to Elastic GNNs, which combine ℓ_1 and ℓ_2 graph penalties (Liu et al., 2021b), and TWIRLS, which unfolds robust graph energies through iteratively reweighted least squares (Yang et al., 2021). Our contributions are as follows:

- We identify two roles of learned edge targets: integrable components shift node coordinates, while cycle components capture closed-path discrepancies.
- We introduce LEGO, which learns preferred endpoint differences with TV or learned radial potentials, and establish finite-depth identities and approximation-error bounds.
- We show target-learning gains across benchmarks and GCN/GAT, and use controlled experiments to study target placement, cycle geometry, and edge correspondence.

2 METHODOLOGY

2.1 LAYER INPUTS AND TARGET-CENTERED OBJECTIVE

Let $G = (V, E)$ be an undirected graph with $n = |V|$ nodes and $m = |E|$ edges. The set $\vec{E}$ contains one oriented representative of each unordered pair of distinct adjacent nodes. Its incidence matrix $K \in \mathbb{R}^{m\times n}$ maps node features $U \in \mathbb{R}^{n\times d}$ to differences $(KU)_e = U_i - U_j$ for $e = (i, j) \in \vec{E}$, where d is the feature width. At layer ℓ, incoming states $H^{(\ell)}$ determine a node anchor $Z^{(\ell)}$, positive edge weights $W^{(\ell)}$, and edge targets $M^{(\ell)}$:

$$Z^{(\ell)} = \mathrm{PreAct}(H^{(\ell)}), \quad W^{(\ell)} = \mathrm{WeightNet}(H^{(\ell)}, \vec{E}), \quad M^{(\ell)} = \mathrm{OffsetNet}(H^{(\ell)}, \vec{E}). \tag{1}$$

The input and output widths are d_ℓ and $d_{\ell+1}$, respectively, with $Z^{(\ell)} \in \mathbb{R}^{n\times d_{\ell+1}}$, $W^{(\ell)} \in \mathbb{R}^m_{>0}$, and $M^{(\ell)} \in \mathbb{R}^{m\times d_{\ell+1}}$. We write $h_i^{(\ell)}$ for an incoming node state and $\mu_e^{(\ell)}$ for an edge target. Figure 2 shows how the graph block uses these fields. With anchor scale $\alpha > 0$, they define the convex reference objective

$$U_\star^{(\ell+1)} = \arg\min_U \frac{1}{2\alpha}\|U - Z^{(\ell)}\|_F^2 + \sum_{e\in\vec{E}} w_e^{(\ell)} g_{\theta_\ell}\big(\|(KU)_e - \mu_e^{(\ell)}\|_2\big). \tag{2}$$

The potential g_{θ_ℓ} is convex and increasing, with trainable parameters θ_ℓ in LP. The quadratic anchor gives a unique minimizer, denoted by $P_{\theta_\ell, W^{(\ell)}, M^{(\ell)}}(Z^{(\ell)})$. The implemented layer returns the finite map $\widehat{P}^{K_{\mathrm{iter}}, T_{\mathrm{newton}}}$, where K_{iter} counts node-edge updates and T_{newton} counts scalar LP iterations. We differentiate through this unrolled computation during supervised training and select its depth by validation.

A two-node example illustrates the target's role. For one edge with weight w, target μ, and TV potential $g_{\mathrm{TV}}(t) = t$, define the anchor mean $z_c = (Z_1 + Z_2)/2$, residual $x = Z_1 - Z_2 - \mu$, and radius $r = \|x\|_2$. At $r > 0$, the exact endpoint difference $v_\star$ and node outputs are

$$v_\star = \mu + \frac{\max\{r - 2\alpha w, 0\}}{r} x, \qquad U_{\star,1} = z_c + v_\star/2, \quad U_{\star,2} = z_c - v_\star/2. \tag{3}$$

When $r \le 2\alpha w$, including $r = 0$, the output difference equals μ and the node mean stays at z_c. The target is the preferred separation; setting $\mu = 0$ gives equal endpoints in this regime. We give the general radial derivation in Appendix A.10.

2.2 TRAINABLE WEIGHTS, TARGETS, AND POTENTIALS

The shared backbone forms its anchor from self, one-hop, high-pass, and two-hop paths. Let $\mathrm{Mean}(X)_i$ average the rows of a node-feature matrix X over the symmetrized neighbors of node i, (set to zero for an empty neighborhood). The one-hop mean $N_1^{(\ell)}$, two-hop mean $N_2^{(\ell)}$, and high-pass field $H_{\mathrm{hp}}^{(\ell)}$ are

$$N_1^{(\ell)} = \mathrm{Mean}(H^{(\ell)}), \quad N_2^{(\ell)} = \mathrm{Mean}(N_1^{(\ell)}), \quad H_{\mathrm{hp}}^{(\ell)} = H^{(\ell)} - N_1^{(\ell)}. \tag{4}$$

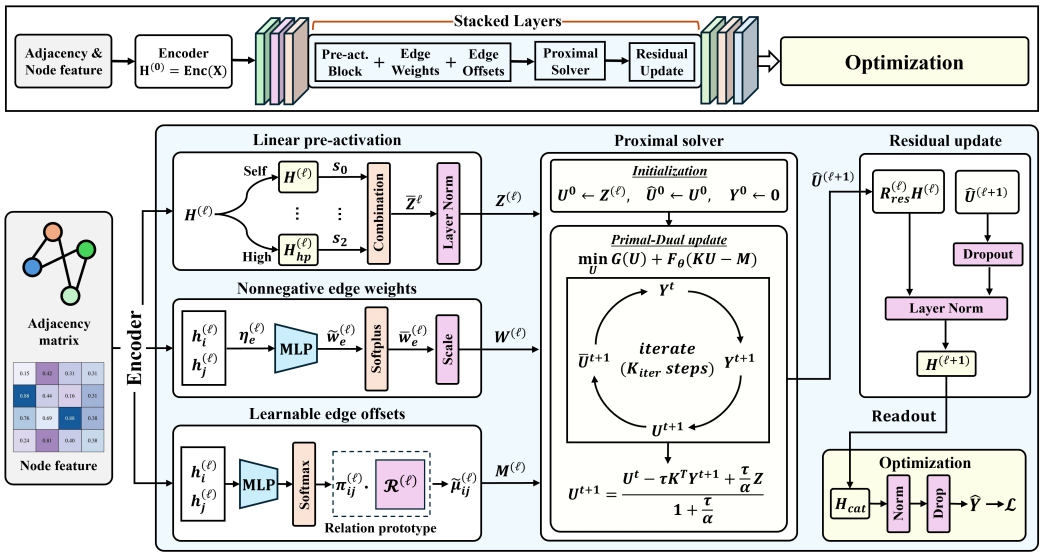


Figure 2: LEGO inside a stacked graph network. The upper strip shows the encoder and repeated graph blocks. The expanded view routes a node anchor, nonnegative edge weights, and learned offsets into the primal-dual computation. The self and high-pass paths illustrate anchor construction, and the relation prototypes illustrate dictionary mixing. The right panels depict the residual block and the concatenated readout used for prediction.

We combine these fields to form the propagated anchor:

$$Z^{(\ell)} = \mathrm{LayerNorm}\big(s_0^{(\ell)} H^{(\ell)} W_{\mathrm{self}}^{(\ell)} + s_1^{(\ell)} N_1^{(\ell)} W_{\mathrm{1hop}}^{(\ell)} + s_2^{(\ell)} H_{\mathrm{hp}}^{(\ell)} W_{\mathrm{hp}}^{(\ell)} + s_3^{(\ell)} N_2^{(\ell)} W_{\mathrm{2hop}}^{(\ell)} + b_z^{(\ell)}\big). \tag{5}$$

Each branch matrix maps d_ℓ input features to $d_{\ell+1}$ output features. The learned scalars $s_0^{(\ell)}, \ldots, s_3^{(\ell)}$ combine the branches, and $b_z^{(\ell)}$ is a node-shared bias. WeightNet uses the symmetric endpoint features $\eta_e^{(\ell)} = [h_i^{(\ell)} + h_j^{(\ell)} \| |h_i^{(\ell)} - h_j^{(\ell)}| \| h_i^{(\ell)} \odot h_j^{(\ell)}]$. The symbols $\|$ and $\odot$ denote concatenation and elementwise multiplication. For $m > 0$, WeightNet converts these features into positive scores and normalized weights:

$$\bar{w}_e^{(\ell)} = \mathrm{softplus}(\mathrm{MLP}_w^{(\ell)}(\eta_e^{(\ell)})) + \varepsilon_w, \qquad w_e^{(\ell)} = s_w^{(\ell)} \frac{\bar{w}_e^{(\ell)}}{m^{-1} \sum_{e'} \bar{w}_{e'}^{(\ell)}}. \tag{6}$$

The scalar-output multilayer perceptron $\mathrm{MLP}_w^{(\ell)}$ predicts an edge score, $\varepsilon_w > 0$ ensures positivity, and $s_w^{(\ell)} = \mathrm{softplus}(\tilde{s}_w^{(\ell)})$ sets the learned mean weight. The parameter $\tilde{s}_w^{(\ell)}$ is unconstrained.

OffsetNet combines a dictionary mixture with a linearly projected endpoint difference. For K_μ learned dictionary vectors $r_k^{(\ell)} \in \mathbb{R}^{d_{\ell+1}}$, the ordered mixture coefficients $\pi_{ij}^{(\ell)}$, dictionary output $\tilde{\mu}_{ij}^{(\ell)}$, and linear difference $\Delta_{ij}^{(\ell)}$ are

$$\pi_{ij}^{(\ell)} = \mathrm{softmax}\left(\mathrm{MLP}_\pi^{(\ell)}([h_i^{(\ell)} + h_j^{(\ell)} \| h_i^{(\ell)} - h_j^{(\ell)} \| h_i^{(\ell)} \odot h_j^{(\ell)}])\right),$$
$$\tilde{\mu}_{ij}^{(\ell)} = \sum_{k=1}^{K_\mu} \pi_{ij,k}^{(\ell)} r_k^{(\ell)}, \qquad \Delta_{ij}^{(\ell)} = (h_i^{(\ell)} - h_j^{(\ell)}) D_\mu^{(\ell)}. \tag{7}$$

The network $\mathrm{MLP}_\pi^{(\ell)}$ outputs dictionary logits, and $D_\mu^{(\ell)} \in \mathbb{R}^{d_\ell \times d_{\ell+1}}$ is a bias-free linear projection. A learned scale $s_\mu^{(\ell)} = \mathrm{sigmoid}(\tilde{s}_\mu^{(\ell)})$, parameterized by an unconstrained scalar $\tilde{s}_\mu^{(\ell)}$, gives the raw

and clipped targets

$$\mu^{(\ell)}_{\mathrm{raw},e} = s^{(\ell)}_{\mu}\left[\frac{\tilde{\mu}^{(\ell)}_{ij} - \tilde{\mu}^{(\ell)}_{ji}}{2} + \Delta^{(\ell)}_{ij}\right], \qquad \mu^{(\ell)}_{e} = \mu_{\max}\tanh\left(\mu^{(\ell)}_{\mathrm{raw},e}/\mu_{\max}\right). \tag{8}$$

The bound $\mu_{\max} > 0$ applies to each target coordinate. Antisymmetrization and odd clipping reverse the target when the endpoints are exchanged; the symmetric weight remains unchanged. The resulting node output is invariant to edge reversal. Before clipping, the linear branch is a difference of node potentials. The decomposition of the full target into integrable and cycle components also depends on dictionary mixing and clipping (Section 3).

The potential sets the response to a residual of a given magnitude. For $t \geq 0$, the learned-potential variant (LP) uses a sigmoid-mixture derivative and its integral:

$$\psi_\theta(t) = b_0 + \sum_{k=1}^{K_\psi} a_k \mathrm{sigmoid}(\beta_k t + c_k), \qquad g_\theta(t) = \int_0^t \psi_\theta(s)\, ds. \tag{9}$$

The K_ψ bases have learned shifts c_k, positive amplitudes a_k, positive slopes β_k, and a positive base b_0, with positivity enforced by softplus. TV uses $g_{\mathrm{TV}}(t) = t$ and $\psi_{\mathrm{TV}}(t) = 1$. Both potentials are convex and increasing, with derivative bound $L_\theta = b_0 + \sum_k a_k$ for LP and $L_\theta = 1$ for TV. Both have positive slopes at the origin, so the vector penalty has a kink at its target. We denote the resulting models by LEGO-TV and LEGO-LP, with Unshifted-TV and Unshifted-LP as their zero-target counterparts.

2.3 Finite Primal-Dual Computation

Chambolle-Pock splitting (Chambolle and Pock, 2011) alternates edge responses and node updates. We select $K_{\mathrm{iter}} = 12$ outer updates and $T_{\mathrm{newton}} = 8$ LP iterations by validation (Appendix B.6). Within a layer call, the anchor, weights, targets, and potential parameters remain fixed. We omit the layer index and write the anchor and edge terms as

$$G_Z(U) = \frac{1}{2\alpha}\|U - Z\|_F^2, \qquad F_\theta(S) = \sum_e w_e g_\theta(\|S_e\|_2). \tag{10}$$

The matrix $S \in \mathbb{R}^{m\times d}$ contains edge residuals, with d denoting the layer output width. Starting from $U^0 = \bar{U}^0 = Z$ and a zero dual edge field Y^0, the computation applies

$$\begin{aligned} \tilde{Y}^{t+1} &= Y^t + \sigma(K\bar{U}^t - M), & Y^{t+1} &= \tilde{Y}^{t+1} - \sigma\widehat{S}^{t+1}, \\ U^{t+1} &= \frac{U^t - \tau K^\top Y^{t+1} + (\tau/\alpha)Z}{1 + \tau/\alpha}, & \bar{U}^{t+1} &= U^{t+1} + (U^{t+1} - U^t). \end{aligned} \tag{11}$$

The extrapolated node state is $\bar{U}^t$, and $\widehat{S}^{t+1}$ is the computed radial edge response. For each edge, the response uses an input q_e, radius r_e, and scaled weight λ_e:

$$q_e = \tilde{Y}_e^{t+1}/\sigma, \qquad r_e = \|q_e\|_2, \qquad \lambda_e = w_e/\sigma. \tag{12}$$

TV returns radius $\hat{s}_e = \max\{r_e - \lambda_e, 0\}$. LP returns zero when $r_e \leq \lambda_e\psi_\theta(0)$. On its positive-root branch, the optimality residual Γ_e and implemented radius updates are

$$\begin{aligned} \Gamma_e(s) &= s + \lambda_e\psi_\theta(s) - r_e, & s^{(0)} &= r_e, \\ s^{(q+1)} &= \Pi_{[0,r_e]}\left(s^{(q)} - \frac{\Gamma_e(s^{(q)})}{1 + \lambda_e\psi'_\theta(s^{(q)})}\right), & q &= 0, \ldots, T_{\mathrm{newton}} - 1. \end{aligned} \tag{13}$$

Projection onto $[0, r_e]$, denoted by $\Pi_{[0,r_e]}$, keeps each iterate in the feasible radial interval. LP returns $\hat{s}_e = s^{(8)}$. For $r_e > 0$, the edge response is $\widehat{S}_e^{t+1} = (\hat{s}_e/r_e)q_e$, and a zero input returns zero. The radial response scales the input through closed-form TV shrinkage or the unrolled LP recurrence, preserving its direction.

The primal and dual step sizes are $\tau = \sigma = \kappa/\sqrt{2d_{\max}}$, where $d_{\max} > 0$ is the maximum incidence degree and $\kappa = 0.9$. Since $\|K\|^2 \leq 2d_{\max}$, these steps satisfy $\tau\sigma\|K\|^2 < 1$, the convergence

condition for exact edge responses. These step sizes also control how quickly graph interaction accumulates over the unrolled updates.

The returned node field is

$$\widehat{U}^{(\ell+1)} = \widehat{P}^{K_{\text{iter}},T_{\text{newton}}}_{\theta_\ell,W^{(\ell)},M^{(\ell)}}(Z^{(\ell)}). \tag{14}$$

An edgeless layer returns $Z^{(\ell)}$. On a connected component, each update preserves the anchor mean because the rows of $K^\top Y$ sum to zero. Let $\chi = (1+\tau/\alpha)^{-1}$ denote the coefficient of the previous primal state. Expanding the node recurrence expresses the finite correction as

$$\widehat{U} - Z = -\alpha(1-\chi)\sum_{t=1}^{K_{\text{iter}}} \chi^{K_{\text{iter}}-t} K^\top Y^t, \qquad \|Y_e^t\|_2 \le w_e L_\theta. \tag{15}$$

The coefficients $(1-\chi)\chi^{K_{\text{iter}}-t}$ sum to $1-\chi^{K_{\text{iter}}}$. Their accumulation depends on degree through the step size; the magnitude and direction of the correction depend on $K^\top Y^t$. We derive nodewise and global bounds and relate this correction to the optimal dual force in Appendix A.3.

2.4 Residual Integration and Training

We use the node output directly in the residual block:

$$H^{(\ell+1)} = \text{LayerNorm}\left(R^{(\ell)}_{\text{res}}(H^{(\ell)}) + \text{Dropout}(\widehat{U}^{(\ell+1)})\right). \tag{16}$$

The residual map $R^{(\ell)}_{\text{res}}$ is the identity or a dimension-matching linear projection. The mean-preservation identity applies to $\widehat{U}$ before residual integration and normalization. The classifier uses the concatenated encoder and layer outputs, with GELU in both the encoder and classifier (Hendrycks and Gimpel, 2016).

Training objective. For a network with L layers on a graph with $m > 0$, we minimize supervised cross-entropy with edge-weight and target regularization:

$$\mathcal{L}_{\text{train}} = \mathcal{L}_{\text{sup}} + \lambda_w \sum_{\ell=0}^{L-1} s_w^{(\ell)} + \lambda_\mu \sum_{\ell=0}^{L-1} \frac{\|M^{(\ell)}\|_F^2}{m d_{\ell+1}}. \tag{17}$$

Supervised cross-entropy is $\mathcal{L}_{\text{sup}}$, and $\lambda_w, \lambda_\mu \ge 0$ weight the mean edge-scale and squared-target penalties. For $m = 0$, we set both edge regularizers to zero. We analyze the effective coupling scale in Appendix B.1.

3 Target Geometry in the Finite Layer

3.1 Node Translation and the Integrable Component

An integrable target field has the form $M = KB$ for a node field $B \in \mathbb{R}^{n\times d}$. Its edge targets are realizable as node differences and sum to zero around every cycle. The incidence decomposition (Lim, 2020) expresses an arbitrary target as an integrable component and an orthogonal cycle component:

$$B^\dagger \in \arg\min_B \|M - KB\|_F^2, \qquad M = KB^\dagger + M_\perp, \qquad K^\top M_\perp = 0. \tag{18}$$

We fix $B^\dagger$ by requiring zero mean on each connected component. Write $\Pi_{\mathcal{I}}$ for projection onto $\mathcal{I} = \text{Im}(K)$ and $\Pi_\perp = I - \Pi_{\mathcal{I}}$ for projection onto $\ker K^\top$, which contains $M_\perp$. With c connected components, the cycle-field space has dimension $(m-n+c)d$.

For the finite-depth analysis, set $T = K_{\text{iter}}$ and $Q = T_{\text{newton}}$, with $Q = 0$ for closed-form TV. Substituting $U = B^\dagger + V$, where V is the translated node field, makes the two edge residuals identical. The anchor-based initialization then gives

$$\widehat{P}^{T,Q}_{\theta,W,M}(Z) = B^\dagger + \widehat{P}^{T,Q}_{\theta,W,M_\perp}(Z - B^\dagger). \tag{19}$$

For $M = KB$, Eq. 19 reduces to $B + \widehat{P}^{T,Q}_{\theta,W,0}(Z-B)$. An integrable target amounts to applying a zero-target graph block in learned node coordinates and restoring the translation at the output. The identity holds at finite depth and motivates Integrable-train, which learns the node field directly. OffsetNet's raw linear branch corresponds to $B = s_\mu H D_\mu$, while $B^\dagger$ describes the full clipped target, including dictionary mixing. Appendix A.2 proves the identity by induction.

3.2 How Cycle Targets Reach Node Outputs

The cycle target is used in the edge-response calculation before $K^\top$ aggregates them. This ordering matters because radial transformations can change the relative magnitudes of edge rows whose initial divergence is zero. Define $a_e(q) = 1 - \hat{s}_e(\|q\|_2)/\|q\|_2$ for $q \neq 0$, with $a_e(0) = 0$, and let $D(X)$ be the diagonal matrix of these multipliers for an edge field X. Set $X = K(Z - B^\dagger)$, $D_M = D(X - M_\perp)$, and $D_{\text{int}} = D(X)$. The first primal updates from the common anchor satisfy

$$U_M^1 - U_{\text{int}}^1 = \frac{\tau\sigma}{1+\tau/\alpha} K^\top \left[D_M M_\perp + (D_{\text{int}} - D_M) X \right]. \tag{20}$$

The two terms describe reweighting of the cycle field and the target-induced change in the response to the integrable residual. With heterogeneous multipliers, the reweighted cycle field can have nonzero divergence and affect the node update. Lemma 1 extends this response through all finite updates, including the different threshold branches of TV.

3.3 Target Placement and the Exact Map

The positive slopes of TV and LP at the origin also matter for target placement in the exact map. When two such maps agree over all anchors up to a node translation, their targets differ by the corresponding node-potential field (Theorem 2, Appendix A.7). For comparison, completing the square for $g_{\text{quad}}(t) = t^2/2$ gives

$$P_{\text{quad},W,M}(Z) = P_{\text{quad},W,0}\big(Z + \alpha K^\top D_W M\big), \qquad D_W = \operatorname{diag}(W). \tag{21}$$

The diagonal matrix D_W contains the edge weights. For a quadratic potential, the target changes the exact solution through an additive anchor adjustment. The weighted divergence can include a contribution from Euclidean cycle fields when weights vary across edges. With uniform weights, those fields cancel. The anchor-injection control uses the same transformation before an Unshifted-LP block and keeps the full OffsetNet parameterization (Appendix C.3).

3.4 Error Propagation Through the Finite Computation

We distinguish two sources of error relative to the variational reference: approximate radial responses and finite outer depth. Let U_{ex}^T denote the same T-step recurrence with exact edge responses. Comparing the implemented trajectory with this intermediate output yields the following bound.

Theorem 1 (Finite-unroll perturbation bound)**.** *Fix Z, W, M, θ, α, anchor-based initialization, zero dual initialization, and the fixed extrapolation $\bar{U}^{t+1} = 2U^{t+1} - U^t$. Assume $\tau\sigma\|K\|^2 < 1$ and put $\chi = (1+\tau/\alpha)^{-1}$. At update t of the implemented trajectory, define*

$$\begin{aligned} R_t &= \left(\sum_e (\epsilon_e^t)^2 \right)^{1/2}, \\ \epsilon_e^t &= \begin{cases} |\hat{s}_e^t + (w_e/\sigma)\psi_\theta(\hat{s}_e^t) - r_e^t|, & \textit{LP positive-root call}, \\ 0, & \textit{TV or exact zero/identity branch}. \end{cases} \end{aligned} \tag{22}$$

The input radius at update t is r_e^t, ϵ_e^t is its radial optimality residual, and R_t aggregates these residuals across edges. In exact arithmetic,

$$\|\widehat{P}_{\theta,W,M}^{T,Q}(Z) - U_{\text{ex}}^T\|_F \le \chi\sqrt{\tau\sigma} \sum_{t=1}^{T} R_t. \tag{23}$$

Consequently,

$$\|\widehat{P}_{\theta,W,M}^{T,Q}(Z) - P_{\theta,W,M}(Z)\|_F \le \underbrace{\|U_{\text{ex}}^T - P_{\theta,W,M}(Z)\|_F}_{\textit{outer truncation}} + \underbrace{\chi\sqrt{\tau\sigma} \sum_{t=1}^{T} R_t}_{\textit{inner error bound}}. \tag{24}$$

The proof uses Chambolle-Pock nonexpansiveness (Chambolle and Pock, 2011) and the quadratic anchor (Appendix A.5). For TV, the radial response is exact. For LP, the residual sum controls the inner approximation error along the implemented trajectory. The outer term measures finite-depth truncation. Complementary gap and stationarity bounds appear in Appendix A.9.

4 EXPERIMENTS

We evaluate low-label transductive node prediction on RomanEmpire, Minesweeper, AmazonRatings, Chameleon, and Squirrel using ten shared random $10\%/25\%/65\%$ train/validation/test partitions. We select configurations and checkpoints on the validation set and pair common-module initializations across training comparisons. Table 1(a) uses the full-size graphs, including the duplicate-containing Chameleon and Squirrel instances (Platonov et al., 2023). Panel (b) reports their deduplicated versions. We report accuracy except on Minesweeper, where we use ROC-AUC.

Within each potential, LEGO and Unshifted share the backbone, WeightNet architecture, and twelve outer updates. TV uses closed-form radial responses, and LP uses eight projected Newton updates. Target learning adds OffsetNet and its regularizer. We train each model independently. The target-gain intervals use paired resampling of split identifiers. Appendix C.1 gives the full protocol, including the layerwise order of checkpoint interventions.

4.1 TARGET LEARNING WITH FIXED AND LEARNED POTENTIALS

With the TV potential fixed, learning the target improves performance on every main graph, and all paired intervals are positive (Table 1(a)). These results support learning the penalty center even when the radial potential is fixed. We observe the same target-learning benefit with LP. LP also achieves higher mean scores than TV; the paired interval is positive on Squirrel. We extend the analysis to quadratic penalties and report paired contrasts between potential families in Appendix C.14.

Table 1: Target learning with a shared backbone. Entries are mean ± standard deviation. **(a)** Accuracy or Minesweeper ROC-AUC over ten paired splits. The interval rows give paired 95% intervals for LEGO minus Unshifted within each potential. **(b)** Accuracy on deduplicated graphs.

(a) Five main graphs

Model / contrast	Roman.	Mine.	Amazon.	Cham.	Squirrel
GELU	73.5 ± 1.4	89.4 ± 1.1	45.1 ± 0.9	57.7 ± 1.5	34.9 ± 0.7
Unshifted-TV	80.3 ± 1.6	90.7 ± 1.5	50.5 ± 0.7	58.2 ± 1.8	37.4 ± 0.6
LEGO-TV	82.7 ± 0.9	91.8 ± 1.1	52.1 ± 0.8	59.9 ± 1.7	38.8 ± 0.5
TV target-gain interval	$[1.7, 3.1]$	$[0.5, 1.7]$	$[1.1, 2.2]$	$[1.2, 2.2]$	$[1.2, 1.7]$
Unshifted-LP	80.5 ± 1.0	90.9 ± 0.8	50.7 ± 0.9	59.3 ± 1.2	37.6 ± 0.6
LEGO-LP	$\mathbf{83.2 \pm 1.1}$	$\mathbf{92.1 \pm 0.9}$	$\mathbf{52.5 \pm 0.9}$	$\mathbf{60.2 \pm 1.3}$	$\mathbf{39.2 \pm 0.4}$
LP target-gain interval	$[2.0, 3.4]$	$[0.6, 1.8]$	$[1.2, 2.4]$	$[0.1, 1.7]$	$[1.3, 1.9]$

(b) Deduplicated Chameleon and Squirrel

Dataset	GELU	Unshifted-TV	**LEGO-TV**
Chameleon	41.24 ± 3.63	41.71 ± 2.79	$\mathbf{43.02 \pm 1.95}$
Squirrel	39.80 ± 3.15	40.54 ± 2.60	$\mathbf{43.11 \pm 3.03}$

We also observe higher mean accuracy from target learning on deduplicated Chameleon and Squirrel. LEGO-TV improves over Unshifted-TV and GELU in Table 1(b). Appendix C.17 includes architecture comparisons for these graph versions.

4.2 TARGET PLACEMENT, NODE COORDINATES, AND CYCLE COMPONENTS

Anchor injection uses $Z + \alpha K^\top D_W M$ as the input to Unshifted-LP; Projected-OffsetNet uses $\Pi_{\mathcal{I}} M$ as the target of the graph block. Both keep OffsetNet and match LEGO-LP's parameter count. Integrable-train learns a node field B and uses KB. These controls recover much of the target-learning improvement, and LEGO-LP achieves higher mean scores (Table 2). Increasing the pointwise width also improves GELU; the width and anchor-injection results appear in Appendix C.3. We measure the cycle-norm fraction $r_\perp = \|M_\perp\|_F / (\|M\|_F + \varepsilon_{\mathrm{diag}})$ and average normalized circulation δ_{cyc} over fundamental cycles, with $\varepsilon_{\mathrm{diag}} > 0$ stabilizing the denominator. Most target energy lies in the integrable component (Appendix C.11). Removing the smaller cycle component reduces performance more on average than the norm-controlled random and integrable perturbations. This suggests that the trained predictor uses the learned cycle direction.

Table 2: Target placement, integrability, and cycle removal. Training rows report mean prediction scores. Anchor injection and Projected-OffsetNet match LEGO-LP's parameter count. Frozen rows give original minus intervened metrics over ten checkpoints, with paired 95% intervals for cycle removal.

Comparison	Roman.	Mine.	Amazon.	Cham.	Squirrel
Training from initialization					
GELU-wide	75.4	90.1	46.0	58.5	35.8
Anchor injection	81.2	91.7	52.0	60.0	38.0
Integrable-train	82.1	92.0	51.6	59.9	38.6
Projected-OffsetNet	83.0	91.7	51.5	60.0	38.7
LEGO-LP	**83.2**	**92.1**	**52.5**	**60.2**	**39.2**
Frozen target interventions					
Cycle removal	$+1.4$ $[0.8, 2.0]$	$+0.6$ $[0.2, 1.0]$	$+1.2$ $[0.7, 1.7]$	$+0.4$ $[0.1, 0.7]$	$+0.8$ $[0.5, 1.1]$
Random direction	$+0.17$	$+0.05$	$+0.11$	$+0.06$	$+0.10$
Integrable direction	$+0.30$	$+0.19$	$+0.32$	$+0.16$	$+0.23$

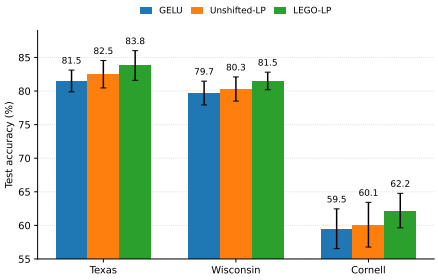


Figure 3: WebKB mean accuracy and standard deviation for GELU, Unshifted-LP, and LEGO-LP.

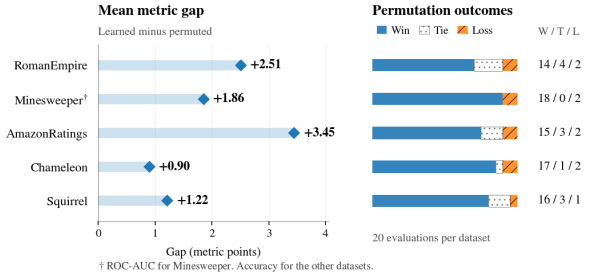


Figure 4: Frozen target-row reassignment. **Left:** mean original-minus-permuted metric. **Right:** original-model win, tie, and loss counts.

4.3 Graph-Block Integration in GCN and GAT

We insert the TV blocks at the hidden activation positions of GCN and GAT, preserving their propagation, normalization, residual paths, and readout (Appendix C.19). In both backbones, LEGO-TV achieves higher mean scores than Unshifted-TV and Original on every graph (Table 3). These consistent gains suggest that the benefit of learned edge targets transfers across convolutional and attention-based propagation rules and is not specific to the shared backbone design. On WebKB (Figure 3), mean accuracy increases from GELU to Unshifted-LP and again to LEGO-LP in the shared backbone.

Table 3: GCN and GAT with graph blocks at their hidden activation positions. Entries are mean $\pm$ standard deviation of accuracy or Minesweeper ROC-AUC. Original retains the pointwise activation. Unshifted-TV and LEGO-TV use the TV computation defined in the experimental protocol.

Backbone	Block	Roman.	Mine.	Amazon.	Cham.	Squirrel
GCN	Original	66.5 ± 0.67	86.2 ± 0.99	44.1 ± 0.48	55.1 ± 0.63	31.0 ± 0.72
	Unshifted-TV	69.3 ± 0.39	87.2 ± 1.05	48.2 ± 0.46	54.2 ± 0.71	32.9 ± 0.74
	LEGO-TV	$\mathbf{70.8 \pm 0.34}$	$\mathbf{88.0 \pm 0.82}$	$\mathbf{49.5 \pm 0.43}$	$\mathbf{56.9 \pm 0.58}$	$\mathbf{34.1 \pm 0.69}$
GAT	Original	66.9 ± 0.41	85.1 ± 1.12	43.4 ± 0.49	54.2 ± 0.85	31.2 ± 0.93
	Unshifted-TV	67.1 ± 0.43	85.8 ± 1.05	46.1 ± 0.51	54.9 ± 0.82	32.5 ± 0.90
	LEGO-TV	$\mathbf{69.7 \pm 0.38}$	$\mathbf{87.0 \pm 1.36}$	$\mathbf{47.2 \pm 0.45}$	$\mathbf{56.0 \pm 0.77}$	$\mathbf{34.1 \pm 0.88}$

4.4 Edge Correspondence and the Tree Boundary

At frozen LEGO-LP checkpoints, we reassign target rows to change their endpoint correspondence. The target multiset and marginal magnitudes remain unchanged. The original assignment achieves higher mean performance and wins most paired evaluations (Figure 4, Appendix C.7).

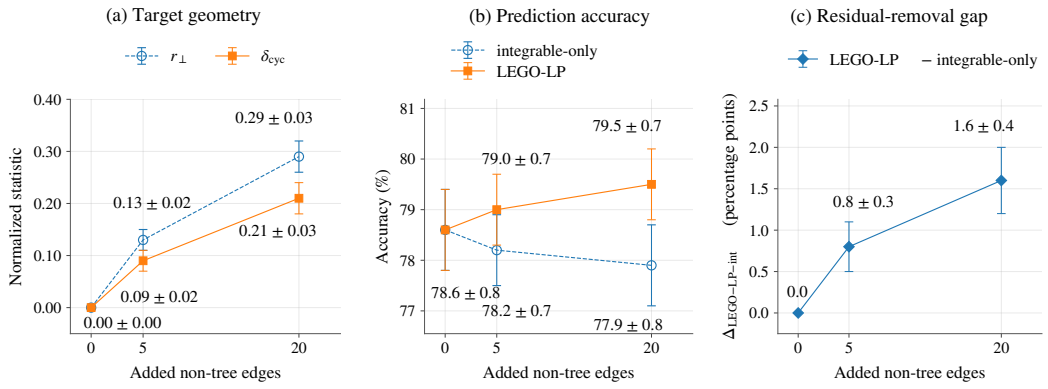


Figure 5: Tree and cycle study over 30 matched replicates. **Left:** projection ratio and normalized circulation. **Middle:** full-target and frozen integrable-only accuracy. **Right:** their paired accuracy difference. Points and error bars show means and standard deviations.

In Figure 5, we compare thirty matched trees and their augmented graphs using shared features, labels, and masks. On trees, $\Pi_{\mathcal{I}} M = M$, so projection preserves the targets and prediction accuracy. Integrable targets still change the node coordinates the graph block uses. Added edges create cycle directions, and the augmented graphs exhibit larger cycle residuals and frozen removal gaps. Because labels are generated independently of graph construction, the comparison examines learned target use under controlled edge addition (Appendix C.15).

Table 4: Full-model cost at the selected LP depth. Training-step time and incremental peak memory are averages of five dataset means, each measured over ten seeds. Inference p95 (the 95th latency percentile) uses a geometric mean. Memory increments are measured above the pre-pass allocation.

Measurement	GELU	Unshifted-LP	LEGO-LP
Training step (ms)	22.37	41.49	42.75
Inference p95 (ms)	5.34	17.53	17.66
Training incremental peak (GiB)	0.63	2.02	2.15
Inference incremental peak (MiB)	84.5	202.6	301.4

4.5 Computational Cost of Target Learning

We profile the full GELU, Unshifted-LP, and LEGO-LP networks on one NVIDIA A100 with two layers of width 128 (Table 4). Adding targets has a smaller effect on latency than introducing graph coupling. Measurement details appear in Appendix B.3.

5 Discussion and Conclusion

We introduced LEGO, a graph-coupled neural layer that learns preferred endpoint differences. The fixed-TV results, deduplicated graph experiments, and GCN/GAT integration support the value of target learning. Our incidence decomposition explains node-coordinate translations and cycle responses at finite depth. Anchor and integrable-target controls recover much of the improvement. Checkpoint interventions further indicate that trained predictors use cycle geometry and edge correspondence. We discuss the limitations of our method in Appendix E.

AI USE STATEMENT

Generative AI tools were used to assist with language editing and to help verify selected mathematical arguments. The authors developed the key proof ideas. The authors independently reviewed and verified all mathematical claims and proof steps and take full responsibility for the final manuscript.

ETHICS STATEMENT

LEGO learns predictive edge preferences on the benchmark instances described in Appendix C. In applications involving people or sensitive attributes, consider predictive performance alongside privacy, fairness, data provenance, and the consequences of downstream decisions.

REPRODUCIBILITY STATEMENT

The assumptions and proofs appear in Appendix A. Appendix B specifies the architecture, finite computation, and training procedure, and Appendix C gives the experimental protocols. Scalar routines and numerical behavior are described in Appendix B.2, memory accounting in Appendix B.4, and model selection in Appendix B.6.

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 *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 21–29. PMLR, 2019.

Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J Zico Kolter. Differentiable convex optimization layers. In *Advances in Neural Information Processing Systems*, volume 32, pages 9562–9574, 2019.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andrew Ballard, Justin Gilmer, George Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Maya Bechler-Speicher, Amir Globerson, and Ran Gilad-Bachrach. The intelligible and effective graph neural additive network. In *Advances in Neural Information Processing Systems*, volume 37, pages 90552–90578, 2024. doi: 10.52202/079017-2874.

Mitchell Black, Zhengchao Wan, Amir Nayyeri, and Yusu Wang. Understanding oversquashing in GNNs through the lens of effective resistance. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 2528–2547. PMLR, 2023.

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. doi: 10.1609/aaai.v35i5.16514.

Cristian Bodnar, Francesco Di Giovanni, Benjamin Chamberlain, Pietro Liò, and Michael Bronstein. Neural sheaf diffusion: A topological perspective on heterophily and oversmoothing in GNNs. In *Advances in Neural Information Processing Systems*, volume 35, pages 18527–18541, 2022. doi: 10.52202/068431-1346.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26, 2013.

Saverio Caminiti, Irene Finocchi, and Rossella Petreschi. On coding labeled trees. *Theoretical computer science*, 382(2):97–108, 2007.

Antonin Chambolle and Thomas Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145, 2011. doi: 10.1007/s10851-010-0251-1.

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, 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.

Yoonhyuk Choi, Taewook Ko, Jiho Choi, and Chong-Kwon Kim. Beyond binary: Improving signed message passing in graph neural networks for multi-class graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(11):9535–9549, 2025a. doi: 10.1109/TPAMI.2025.3581218.

Yoonhyuk Choi, Taewook Ko, Jiho Choi, and Chong-Kwon Kim. Selective blocking for message-passing neural networks on heterophilic graphs. In *Proceedings of the Forty-first Conference on Uncertainty in Artificial Intelligence*, volume 286 of *Proceedings of Machine Learning Research*, pages 737–751. PMLR, 2025b.

Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. In *Advances in Neural Information Processing Systems*, volume 33, pages 13260–13271, 2020.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, volume 29, pages 3844–3852, 2016.

Tyler Derr, Yao Ma, and Jiliang Tang. Signed graph convolutional networks. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 929–934, 2018. doi: 10.1109/ICDM.2018.00113.

Lun Du, Xiaozhou Shi, Qiang Fu, Xiaojun Ma, Hengyu Liu, Shi Han, and Dongmei Zhang. GBK-GNN: Gated bi-kernel graph neural networks for modeling both homophily and heterophily. In *Proceedings of the ACM Web Conference 2022*, pages 1550–1558, 2022. doi: 10.1145/3485447.3512201.

Rui Duan, Mingjian Guang, Junli Wang, Chungang Yan, Hongda Qi, Wenkang Su, Can Tian, and Haoran Yang. Unifying homophily and heterophily for spectral graph neural networks via triple filter ensembles. In *Advances in Neural Information Processing Systems*, volume 37, pages 93540–93567, 2024. doi: 10.52202/079017-2966.

Wenqi Fan, Xiaorui Liu, Wei Jin, Xiangyu Zhao, Jiliang Tang, and Qing Li. Graph trend filtering networks for recommendation. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*, pages 112–121, 2022.

Thomas Frerix, Thomas Möllenhoff, Michael Moeller, and Daniel Cremers. Proximal backpropagation. *arXiv preprint arXiv:1706.04638*, 2017.

Guoji Fu, Peilin Zhao, and Yatao Bian. p-laplacian based graph neural networks. In *International conference on machine learning*, pages 6878–6917. PMLR, 2022.

Johannes Gasteiger, Stefan Weißenberger, and Stephan Günnemann. Diffusion improves graph learning. In *Advances in Neural Information Processing Systems*, volume 32, pages 13354–13366, 2019.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1263–1272. PMLR, 2017.

Stephen Gould, Richard Hartley, and Dylan Campbell. Deep declarative networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(8):3988–4004, 2022. doi: 10.1109/TPAMI.2021.3059462.

Fangda Gu, Heng Chang, Wenwu Zhu, Somayeh Sojoudi, and Laurent El Ghaoui. Implicit graph neural networks. In *Advances in Neural Information Processing Systems*, volume 33, pages 11984–11995, 2020.

Jingwei Guo, Kaizhu Huang, Rui Zhang, and Xinping Yi. ES-GNN: Generalizing graph neural networks beyond homophily with edge splitting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):11345–11360, 2024. doi: 10.1109/TPAMI.2024.3459932.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, volume 30, pages 1024–1034, 2017.

Mingguo He, Zhewei Wei, Zengfeng Huang, and Hongteng Xu. BernNet: Learning arbitrary graph spectral filters via Bernstein approximation. In *Advances in Neural Information Processing Systems*, volume 34, pages 14239–14251, 2021.

Yixuan He, Michael Perlmutter, Gesine Reinert, and Mihai Cucuringu. MSGNN: A spectral graph neural network based on a novel magnetic signed laplacian. In *Proceedings of the First Learning on Graphs Conference*, volume 198 of *Proceedings of Machine Learning Research*, pages 40:1–40:39. PMLR, 2022.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Junjie Huang, Huawei Shen, Liang Hou, and Xueqi Cheng. Signed graph attention networks. In *Artificial Neural Networks and Machine Learning – ICANN 2019: Workshop and Special Sessions*, volume 11731 of *Lecture Notes in Computer Science*, pages 566–577. Springer, 2019. doi: 10.1007/978-3-030-30493-5_53.

Bianca Iancu, Luana Ruiz, Alejandro Ribeiro, and Elvin Isufi. Graph-adaptive activation functions for graph neural networks. In *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2020. doi: 10.1109/MLSP49062.2020.9231732.

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

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

Taewook Ko, Yoonhyuk Choi, and Chong-Kwon Kim. A spectral graph convolution for signed directed graphs via magnetic laplacian. *Neural Networks*, 164:562–574, 2023. doi: 10.1016/j.neunet.2023.05.009.

Runlin Lei, Zhen Wang, Yaliang Li, Bolin Ding, and Zhewei Wei. EvenNet: Ignoring odd-hop neighbors improves robustness of graph neural networks. In *Advances in Neural Information Processing Systems*, volume 35, pages 4694–4706, 2022. doi: 10.52202/068431-0339.

Bingheng Li, Erlin Pan, and Zhao Kang. PC-Conv: Unifying homophily and heterophily with two-fold filtering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 13437–13445, 2024. doi: 10.1609/aaai.v38i12.29246.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. doi: 10.1609/aaai.v32i1.11604.

Xiang Li, Renyu Zhu, Yao Cheng, Caihua Shan, Siqiang Luo, Dongsheng Li, and Weining Qian. Finding global homophily in graph neural networks when meeting heterophily. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 13242–13256. PMLR, 2022.

Derek Lim, Felix Hohne, Xiuyu Li, Sijia Linda Huang, Vaishnavi Gupta, Omkar Bhalerao, and Ser-Nam Lim. Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. In *Advances in Neural Information Processing Systems*, volume 34, pages 20887–20902, 2021.

Lek-Heng Lim. Hodge laplacians on graphs. *Siam Review*, 62(3):685–715, 2020.

Juncheng Liu, Bryan Hooi, Kenji Kawaguchi, Yiwei Wang, Chaosheng Dong, and Xiaokui Xiao. Scalable and effective implicit graph neural networks on large graphs. In *International Conference on Learning Representations*, 2024.

Xiaorui Liu, Jiayuan Ding, Wei Jin, Han Xu, Yao Ma, Zitao Liu, and Jiliang Tang. Graph neural networks with adaptive residual. *Advances in Neural Information Processing Systems*, 34:9720–9733, 2021a.

Xiaorui Liu, Wei Jin, Yao Ma, Yaxin Li, Hua Liu, Yiqi Wang, Ming Yan, and Jiliang Tang. Elastic graph neural networks. In *International conference on machine learning*, pages 6837–6849. PMLR, 2021b.

Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. Revisiting heterophily for graph neural networks. In *Advances in Neural Information Processing Systems*, volume 35, pages 1362–1375, 2022. doi: 10.52202/068431-0100.

Krishna Sri Ipsit Mantri, Xinzhi (Aurora) Wang, Carola-Bibiane Schönlieb, Bruno Ribeiro, Beatrice Bevilacqua, and Moshe Eliasof. DiGRAF: Diffeomorphic graph-adaptive activation function. In *Advances in Neural Information Processing Systems*, volume 37, pages 3649–3681, 2024. doi: 10.52202/079017-0121.

Antonio G Marques, Santiago Segarra, and Gonzalo Mateos. Signal processing on directed graphs: The role of edge directionality when processing and learning from network data. *IEEE Signal Processing Magazine*, 37(6):99–116, 2020.

Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and Leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4602–4609, 2019. doi: 10.1609/aaai.v33i01.33014602.

Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. 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? *arXiv preprint arXiv:2302.11640*, 2023.

Michael Poli, Stefano Massaroli, Junyoung Park, Atsushi Yamashita, Hajime Asama, and Jinkyoo Park. Graph neural ordinary differential equations. *arXiv preprint arXiv:1911.07532*, 2019.

Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. In *Advances in Neural Information Processing Systems*, volume 35, pages 14501–14515, 2022. doi: 10.52202/068431-1054.

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. DropEdge: Towards deep graph convolutional networks on node classification. *arXiv preprint arXiv:1907.10903*, 2019.

Emanuele Rossi, Bertrand Charpentier, Francesco Di Giovanni, Fabrizio Frasca, Stephan Günnemann, and Michael M Bronstein. Edge directionality improves learning on heterophilic graphs. In *Proceedings of the Second Learning on Graphs Conference*, volume 231 of *Proceedings of Machine Learning Research*, pages 25:1–25:27. PMLR, 2024.

Luana Ruiz, Fernando Gama, Antonio G Marques, and Alejandro Ribeiro. Median activation functions for graph neural networks. In *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7440–7444. IEEE, 2019. doi: 10.1109/ICASSP.2019.8683472.

Luana Ruiz, Fernando Gama, Antonio García Marques, and Alejandro Ribeiro. Invariance-preserving localized activation functions for graph neural networks. *IEEE Transactions on Signal Processing*, 68:127–141, 2020. doi: 10.1109/TSP.2019.2955832.

Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3693–3702, 2017.

Arseny Skryagin, Felix Divo, Mohammad Amin Ali, Devendra Singh Dhami, and Kristian Kersting. Graph neural networks need cluster-normalize-activate modules. In *Advances in Neural Information Processing Systems*, volume 37, pages 79819–79842, 2024. doi: 10.52202/079017-2535.

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

Junfu Wang, Yuanfang Guo, Liang Yang, and Yunhong Wang. Understanding heterophily for graph neural networks. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 50489–50529. PMLR, 2024.

Xiyuan Wang and Muhan Zhang. How powerful are spectral graph neural networks. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 23341–23362. PMLR, 2022.

Yifei Wang, Yisen Wang, Jiansheng Yang, and Zhouchen Lin. Dissecting the diffusion process in linear graph convolutional networks. In *Advances in Neural Information Processing Systems*, volume 34, pages 5758–5769, 2021.

Yu-Xiang Wang, James Sharpnack, Alexander J Smola, and Ryan J Tibshirani. Trend filtering on graphs. *Journal of Machine Learning Research*, 17(105):1–41, 2016.

Ezra Winston and J Zico Kolter. Monotone operator equilibrium networks. In *Advances in Neural Information Processing Systems*, volume 33, pages 10718–10728, 2020.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6861–6871. PMLR, 2019.

Qitian Wu, Wentao Zhao, Chenxiao Yang, Hengrui Zhang, Fan Nie, Haitian Jiang, Yatao Bian, and Junchi Yan. SGFormer: Simplifying and empowering transformers for large-graph representations. In *Advances in Neural Information Processing Systems*, volume 36, pages 64753–64773, 2023. doi: 10.52202/075280-2826.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2021. doi: 10.1109/TNNLS.2020.2978386.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5453–5462. PMLR, 2018.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

Yujun Yan, Milad Hashemi, Kevin Swersky, Yaoqing Yang, and Danai Koutra. Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks. In *2022 IEEE International Conference on Data Mining (ICDM)*, pages 1287–1292. IEEE, 2022.

Yongyi Yang, Tang Liu, Yangkun Wang, Jinjing Zhou, Quan Gan, Zhewei Wei, Zheng Zhang, Zengfeng Huang, and David Wipf. Graph neural networks inspired by classical iterative algorithms. In *International Conference on Machine Learning*, pages 11773–11783. PMLR, 2021.

Yongyi Yang, Tang Liu, Yangkun Wang, Zengfeng Huang, and David Wipf. Implicit vs unfolded graph neural networks. *Journal of Machine Learning Research*, 26(82):1–46, 2025.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? In *Advances in Neural Information Processing Systems*, volume 34, pages 28877–28888, 2021.

Jingxiao Zhang, Shifei Ding, Jian Jun Zhang, Lili Guo, and Xuan Li. L2DGCN: Learnable enhancement and label selection dynamic graph convolutional networks for mitigating degree bias. In *Advances in Neural Information Processing Systems*, volume 38, pages 193457–193476, 2025. doi: 10.52202/085713-5809.

Si Zhang, Hanghang Tong, Jiejun Xu, and Ross Maciejewski. Graph convolutional networks: a comprehensive review. *Computational Social Networks*, 6(1):11, 2019. doi: 10.1186/s40649-019-0069-y.

Xitong Zhang, Yixuan He, Nathan Brugnone, Michael Perlmutter, and Matthew Hirn. MagNet: A neural network for directed graphs. In *Advances in Neural Information Processing Systems*, volume 34, pages 27003–27015, 2021.

Yifei Zhang, Hao Zhu, Ziqiao Meng, Piotr Koniusz, and Irwin King. Graph-adaptive rectified linear unit for graph neural networks. In *Proceedings of the ACM Web Conference 2022*, pages 1331–1339, 2022. doi: 10.1145/3485447.3512159.

Xin Zheng, Yixin Liu, Shirui Pan, Miao Zhang, Di Jin, and Philip S. Yu. Graph neural networks for graphs with heterophily: A survey. *arXiv preprint arXiv:2202.07082v1*, 2022.

Xin Zheng, Miao Zhang, Chunyang Chen, Qin Zhang, Chuan Zhou, and Shirui Pan. Auto-HeG: Automated graph neural network on heterophilic graphs. In *Proceedings of the ACM Web Conference 2023*, pages 611–620, 2023. doi: 10.1145/3543507.3583498.

Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. In *Advances in Neural Information Processing Systems*, volume 33, pages 7793–7804, 2020.

Jiong Zhu, Ryan A Rossi, Anup Rao, Tung Mai, Nedim Lipka, Nesreen K Ahmed, and Danai Koutra. Graph neural networks with heterophily. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11168–11176, 2021a. doi: 10.1609/aaai.v35i12.17332.

Meiqi Zhu, Xiao Wang, Chuan Shi, Houye Ji, and Peng Cui. Interpreting and unifying graph neural networks with an optimization framework. In *Proceedings of the Web Conference 2021*, pages 1215–1226, 2021b. doi: 10.1145/3442381.3449953.

Wei Zhuo, Han Yu, Guang Tan, and Xiaoxiao Li. Commute graph neural networks. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 80612–80628. PMLR, 13–19 Jul 2025.

Appendix Table of Contents

A Theoretical Analysis **17**
A.1 Notation and Tensor Conventions 17
A.2 Standard Convex and Incidence Facts 17
A.3 Bounds on the Implemented Response 18
A.4 Radial Reduction and Scalar Error 18
A.5 Proof of the Finite-Unroll Perturbation Bound 19
A.6 Finite Cycle Responses and Their Derivatives 19
A.7 Exact-Map Identification 20
A.8 Exact Stability and Outer Convergence 21
A.9 A Posteriori Output Bounds 22
A.10 Two-Node Interpretation 23
A.11 Dictionary and Clipping Contributions 23

B Implementation Details **23**
B.1 Architecture and Trainable Parameterization 23
B.2 Learned-Potential and Closed-Form TV Routines 25
B.3 Full-Model Runtime and Memory Profile 28
B.4 Computational Cost and Memory Measurement 28
B.5 Differentiation and Gradient Measurements 29
B.6 Hyperparameter and Model Selection 29
B.7 Parameter Accounting for Matched Controls 29
B.8 Gauge-Fixed Computation of the Projected Controls 30
B.9 Global Projection Distance and Cycle-Normalized Circulation 31

C Experimental Details **32**
C.1 Evaluation Protocol 33
C.2 Baseline Architectures and Shared-Backbone Controls 33
C.3 Width and Anchor-Injection Controls 36
C.4 Paired WebKB Comparisons 37
C.5 Node-Potential Training and Integrable Targets 37
C.6 Projected-OffsetNet Training Protocol 38
C.7 Frozen Edge-Assignment Permutations 38
C.8 Macro-F1 at Selected Checkpoints 39
C.9 Graph Structure, Target Geometry, and Correlations 39
C.10 Frozen-Checkpoint Computation and Depth Sensitivity 40
C.11 Projection Controls and Paired Uncertainty 41
C.12 Sensitivity to Finite Computational Depth 42
C.13 Target-Parameter Sensitivity 42
C.14 Target Learning across Fixed and Learned Potentials 43
C.15 Tree and Cycle Comparisons 44
C.16 Node Classification on Heterophilic Benchmarks 45
C.17 Node Prediction on Deduplicated Chameleon and Squirrel 45
C.18 Analysis on Large Graphs 45
C.19 Graph-Coupled Block Transfer to GCN and GAT 47

D Related Work **48**
D.1 Translational Relation Geometry 48
D.2 Graph-Adaptive Nonlinear Operations 48
D.3 Anchored Optimization and Target Placement 48
D.4 Graph Differences and Learned Transport 49
D.5 Relations Encoded in Propagation 49
D.6 Propagation and the Shared Backbone 49

E Limitations **50**

TECHNICAL APPENDICES AND SUPPLEMENTARY MATERIAL

A THEORETICAL ANALYSIS

We first establish convexity, orientation invariance, and node-translation identities. We then bound the unrolled responses and derive their derivatives with respect to cycle perturbations. The exact-map identification theorem and a posteriori bounds characterize its relation to the variational reference.

A.1 NOTATION AND TENSOR CONVENTIONS

Node and edge features are stored as rows. Feature projections act from the right, and incidence operators act from the left. For a layer with feature width d, the node fields are $Z, U \in \mathbb{R}^{n\times d}$, the edge fields are $M, Y \in \mathbb{R}^{m\times d}$, and the weights and incidence matrix are $W \in \mathbb{R}^m_{\geq 0}$ and $K \in \mathbb{R}^{m\times n}$. We use $P_{\theta,W,M}(Z)$ for the exact minimizer, $\widehat{P}^{T,Q}_{\theta,W,M}(Z)$ for the implemented layer, and U^T_{ex} for the T-step recurrence with exact edge responses. The depths T, Q count outer and projected Newton updates, with $Q = 0$ for TV. The notation suppresses K, α, τ, σ. A subscript $+$ restricts a tensor to the positive-weight support, whose m_+ edges connect c_+ components, including isolated nodes. Since WeightNet assigns positive weights to all incidence edges, LEGO uses the full incidence graph as its support.

A.2 STANDARD CONVEX AND INCIDENCE FACTS

Convex reference and orientation convention. Define

$$R_{\theta,W,M}(U) = \sum_e w_e g_\theta(\|(KU)_e - \mu_e\|_2), \qquad \Phi_Z(U, M) = \frac{\|U - Z\|^2_F}{2\alpha} + R_{\theta,W,M}(U). \quad (25)$$

The functions $R_{\theta,W,M}$ and Φ_Z denote the edge regularizer and anchored objective, respectively. The LP derivative and TV alternative are

$$\psi_\theta(s) = b_0 + \sum_{k=1}^{K_\psi} a_k \,\mathrm{sigmoid}(\beta_k s + c_k), \quad g_\theta(s) = \int_0^s \psi_\theta(v)\, dv, \qquad g_{\mathrm{TV}}(s) = s. \quad (26)$$

For LP, $b_0 > 0$, $a_k \geq 0$, and $\beta_k > 0$ imply $g_\theta(s) \geq b_0 s$, $\psi'_\theta \geq 0$, and $\psi_\theta \leq L_\theta = b_0 + \sum_k a_k$. TV has $L_\theta = 1$. A convex nondecreasing potential composed with an affine residual norm gives a finite, continuous, convex regularizer. Adding the $1/\alpha$-strongly convex, coercive anchor makes

$$P_{\theta,W,M}(Z) = \mathrm{prox}_{\alpha R_{\theta,W,M}}(Z) \quad (27)$$

unique. Reversing an incidence row and its target preserves the residual norm, so symmetric weights and antisymmetric targets give orientation-independent exact outputs. The finite updates share this property: a row-sign matrix D sends (K, M, Y) to (DK, DM, DY) while preserving $K^\top Y$.

Translation, integrability, and projection. For any node field B, put $N_+ = M_+ - K_+B$, extending N arbitrarily on inactive edges. The single residual equality $K_+U - M_+ = K_+(U - B) - N_+$ implies

$$\Phi_Z(U, M) = \Phi_{Z-B}(U - B, N). \quad (28)$$

$$P_{\theta,W,M}(Z) = B + P_{\theta,W,N}(Z - B). \quad (29)$$

Conversely, equality of the target-centered and translated zero-target objectives for every U, Z forces $M_+ = K_+B$. To see this, evaluate at $U = Z = B$ and use the positive support weights and the potential's zero set. Integrability is also equivalent to zero circulation. Every cycle sum of KB telescopes, and zero circulation makes path integration independent of the path, so B is defined by a root in each support component.

The orthogonal split $M_+ = K_+B^\dagger + M_{\perp,+}$ obeys $K_+^\top M_{\perp,+} = 0$. Pythagoras gives

$$\|M_+ - K_+B\|^2_F = \|M_{\perp,+}\|^2_F + \|K_+(B^\dagger - B)\|^2_F. \quad (30)$$

The zero component means fixing the gauge of $B^\dagger$. Because $\ker K_+$ consists of componentwise constants, $\operatorname{rank}(K_+) = n - c_+$ and the target quotient has dimension $(m_+ - n + c_+)d$. The unweighted diagnostic measures Euclidean distance to the incidence subspace, independent of edge-weight magnitudes.

Finite translation and component means. For $M = KB + N$, initialize the two calls at Z and $Z - B$ with zero dual fields. If their primal states differ by B and their dual states agree, their next dual inputs coincide. The same deterministic radial routine then returns equal dual fields, and both the primal update and extrapolation preserve the translation. Induction proves Eq. 19; setting $N = 0$ gives the integrable-only implementation. The argument also holds for a computed node field $\widetilde{B}$ and its residual $\widetilde{N} = M - K\widetilde{B}$. Appendix B.8 relates the normal-equation residual to projection accuracy.

Restoring B before residual integration recovers the original node output, so the translation identity applies layer by layer in evaluation mode. For each connected component, summing the primal recurrence preserves the anchor mean because $K^\top Y$ sums to zero. Minimization over componentwise translations gives the same mean identity for the exact map. Residual integration and normalization subsequently act on these restored outputs.

A.3 Bounds on the Implemented Response

With the threshold branches and radius-initialized projected Newton routine defined in the main text, every finite inner depth satisfies

$$\max\{0, r - \lambda_e L_\theta\} \le \hat{s} \le r, \qquad \|Y_e^t\|_2 \le w_e L_\theta. \tag{31}$$

To verify this, put $d_q = r - s^{(q)}$. On the LP positive-root branch, the projected update gives

$$d_{q+1} = \min\left\{ r, \frac{\lambda_e \psi_\theta(s^{(q)}) + \lambda_e \psi'_\theta(s^{(q)}) d_q}{1 + \lambda_e \psi'_\theta(s^{(q)})} \right\}. \tag{32}$$

Starting from $d_0 = 0$, induction gives $0 \le d_q \le \min(r, \lambda_e L_\theta)$. The threshold and TV branches obey the same bound. Since $\|Y_e\| = \sigma(r - \hat{s})$ and $\sigma\lambda_e = w_e$, the dual bound follows even on a Newton two-cycle.

For $E^t = U^t - Z$ and $\chi = (1 + \tau/\alpha)^{-1}$, the exact finite recursion is

$$E^t = -\alpha(1 - \chi) \sum_{j=1}^{t} \chi^{t-j} K^\top Y^j. \tag{33}$$

Consequently,

$$\|E_i^t\|_2 \le \alpha(1 - \chi^t) L_\theta \sum_{e \ni i} w_e, \qquad \|E^t\|_F \le \alpha(1 - \chi^t) \|K\| L_\theta \|W\|_2. \tag{34}$$

The inequalities bound the unrolled correction at each node and over the full graph. For an optimal dual $Y^\star$ and $E^\star = -\alpha K^\top Y^\star$, the correction can be decomposed further:

$$E^t = (1 - \chi^t) E^\star - \alpha(1 - \chi) \sum_{j=1}^{t} \chi^{t-j} K^\top (Y^j - Y^\star). \tag{35}$$

The first term is the accumulated optimal incidence force; the second accounts for dual-trajectory error. The coefficient sum $1 - \chi^t$ determines the accumulation at depth t, and the dual-trajectory term determines its relation to optimization error. Both identities depend on the degree-based step size through χ, as in Eq. 15.

A.4 Radial Reduction and Scalar Error

For input $x \neq 0$, $r = \|x\|_2$, and $\lambda_e \ge 0$, alignment with x minimizes the distance at every fixed radius. The exact edge response is $(s^\star/r)x$, where

$$s^\star = \arg\min_{s \ge 0} \tfrac{1}{2}(s - r)^2 + \lambda_e g_\theta(s). \tag{36}$$

It is zero when $\Gamma_e(0) \geq 0$ and otherwise solves $\Gamma_e(s) = s + \lambda_e \psi_\theta(s) - r = 0$. The derivative satisfies $\Gamma_e' \geq 1$. Therefore, on a positive-root call,

$$|\hat{s} - s^\star| \leq |\Gamma_e(\hat{s})|, \qquad \left\| \frac{\hat{s}}{r} x - \frac{s^\star}{r} x \right\|_2 = |\hat{s} - s^\star|. \tag{37}$$

On the exact zero branch, $\Gamma_e(0) \geq 0$ expresses the active nonnegativity constraint and gives zero error in Eq. 22. At a common dual input, the Moreau-form update differs from the exact dual proximal by at most σR_t in Frobenius norm.

A.5 Proof of the Finite-Unroll Perturbation Bound

Proof of Theorem 1. Put $F(S) = \sum_e w_e g_\theta(\|S_e\|_2)$ and $G = G_Z$, and write F^* for the convex conjugate of F. For node and edge fields $x \in \mathbb{R}^{n \times d}$ and $y \in \mathbb{R}^{m \times d}$, write $z = (x, y)$ and define the positive-definite operator

$$\mathsf{V} = \begin{pmatrix} \tau^{-1} I & -K^\top \\ -K & \sigma^{-1} I \end{pmatrix}, \qquad \|z\|_{\mathsf{V}}^2 = \langle z, \mathsf{V} z \rangle. \tag{38}$$

Positive definiteness follows from $\tau\sigma\|K\|^2 < 1$. An exact primal-first step is

$$x' = \mathrm{prox}_{\tau G}(x - \tau K^\top y), \qquad y' = \mathrm{prox}_{\sigma F^*}(y + \sigma[K(2x' - x) - M]). \tag{39}$$

Writing $\mathcal{B}(x, y) = (\partial G(x) + K^\top y, \partial F^*(y) - Kx + M)$, the two optimality conditions give $\mathsf{V}(z - z') \in \mathcal{B}(z')$. The operator $\mathcal{B}$ is monotone: its off-diagonal incidence terms cancel in the monotonicity inner product. For two such steps, monotonicity and Cauchy-Schwarz imply

$$\|z_1' - z_2'\|_{\mathsf{V}}^2 \leq \langle z_1' - z_2', z_1 - z_2 \rangle_{\mathsf{V}}, \tag{40}$$

so the exact step is nonexpansive in this metric.

Reindex the dual-first recurrence by $z^t = (U^t, Y^{t+1})$. At initialization, $U^0 = Z$. Replacing the first edge response by its exact value changes z^0 by $(0, e^1)$ with $\|e^1\|_F \leq \sigma R_1$, giving initial metric error at most $\sqrt{\sigma} R_1$. Each later update is an exact primal-first step at the current state followed by a dual perturbation $(0, e^{t+1})$ of metric norm at most $\sqrt{\sigma} R_{t+1}$. Nonexpansiveness then gives

$$\|\widehat{z}^{T-1} - z_{\mathrm{ex}}^{T-1}\|_{\mathsf{V}} \leq \sqrt{\sigma} \sum_{t=1}^{T} R_t. \tag{41}$$

To extract the last primal update, let $L = [I, -\tau K^\top]$. The quadratic proximal is χ-Lipschitz, and $L\mathsf{V}^{-1}L^\top = \tau I$ because $L = \tau[I, 0]\mathsf{V}$. Therefore,

$$\|\widehat{U}^T - U_{\mathrm{ex}}^T\|_F \leq \chi \|L(\widehat{z}^{T-1} - z_{\mathrm{ex}}^{T-1})\|_F \leq \chi\sqrt{\tau}\, \|\widehat{z}^{T-1} - z_{\mathrm{ex}}^{T-1}\|_{\mathsf{V}}. \tag{42}$$

This proves Eq. 23. The triangle inequality gives Eq. 24. The case $T = 0$ is immediate. □

A.6 Finite Cycle Responses and Their Derivatives

Lemma 1 (Finite-depth response to a cycle perturbation)**.** *Let $N \in \ker K^\top$ and perturb only the target, $M(\varepsilon) = M + \varepsilon N$, holding the current layer's other tensors fixed. On a differentiable branch of the T, Q computation, define*

$$\mathcal{A}_e(q) = a_e(q) q, \qquad X^t = Y^t/\sigma + K\bar{U}^t - M, \qquad J^t = \mathrm{blockdiag}_e\big(D\mathcal{A}_e(X_e^t)\big). \tag{43}$$

The map $\mathcal{A}_e$ is the edgewise dual response divided by σ, X^t is its input, and J^t collects the edge Jacobians. With dots denoting derivatives at $\varepsilon = 0$, the finite computation obeys

$$\begin{aligned} \dot{Y}^{t+1} &= J^t(\dot{Y}^t + \sigma K \dot{\bar{U}}^t - \sigma N), \\ \dot{U}^{t+1} &= \chi \dot{U}^t - \tau\chi K^\top \dot{Y}^{t+1}, \qquad \dot{\bar{U}}^{t+1} = 2\dot{U}^{t+1} - \dot{U}^t, \end{aligned} \tag{44}$$

starting from zero dotted states. If every edge remains on a zero-radius branch through update T, then $\dot{Y}^t = -t\sigma N$ and $\dot{U}^t = 0$ for all $t \leq T$.

For full and integrable targets, denote trajectory differences by $\Delta U^t, \Delta \bar{U}^t, \Delta Y^t$, and let $N = M - KB$. At step t, put $X_A^t = Y_A^t/\sigma + K\bar{U}_A^t - M_A$ and $D_A^t = D(X_A^t)$ for $A \in \{M, \mathrm{int}\}$. Direct subtraction gives

$$\begin{aligned} \Delta Y^{t+1} &= D_M^t \Delta Y^t + \sigma D_M^t K \Delta \bar{U}^t - \sigma D_M^t N + \sigma (D_M^t - D_{\mathrm{int}}^t) X_{\mathrm{int}}^t, \\ \Delta U^{t+1} &= \chi \Delta U^t - \tau \chi K^\top \Delta Y^{t+1}, \qquad \Delta \bar{U}^{t+1} = 2\Delta U^{t+1} - \Delta U^t. \end{aligned} \tag{45}$$

At initialization, this reduces to Eq. 20. The two target-dependent terms reweight the cycle field and change the radial multipliers.

Proof of Lemma 1. The implemented dual map is $Y^{t+1} = \sigma \mathcal{A}(X^t)$. Differentiating this expression and the affine primal/extrapolation updates gives Eq. 44. At a nonzero input, put $r = \|q\|_2$ and $u = q/r$. Differentiating the radial multiplier gives

$$D\mathcal{A}_e(q) = a_e(q)(I - uu^\top) + (1 - \hat{s}_e'(r))uu^\top. \tag{46}$$

The derivative $\hat{s}_e'(r)$ follows the executed Newton and projection branches, including $s^{(0)} = r$ and active interval endpoints. On a zero-radius branch, $\mathcal{A}_e(q) = q$ and its Jacobian is I. This includes the origin when the branch holds in a neighborhood. At switching points, the formula applies to differentiable one-sided branches with well-defined limits.

On zero-radius branches, $J^t = I$. Substitution into Eq. 44 and induction give $\dot{Y}^t = -t\sigma N$ and $\dot{U}^t = \dot{\bar{U}}^t = 0$, since $K^\top N = 0$. □

An exact finite-depth cancellation regime. Suppose the two trajectories at targets M and $M + N$, $K^\top N = 0$, remain on zero-radius branches for every edge at every update through T. Then $Y^{t+1} = Y^t + \sigma(K\bar{U}^t - M)$ on both trajectories. Induction gives $\Delta Y^t = -t\sigma N$ and $\Delta U^t = \Delta \bar{U}^t = 0$. The two node trajectories agree exactly. A concrete instance uses a componentwise constant anchor, $M = 0$, and

$$T\sigma \|N_e\|_2 \le w_e \psi_\theta(0) \quad \text{for every edge.} \tag{47}$$

At update t, the perturbed radial input is $-tN$. The condition keeps TV and LP on their exact zero branches, so both finite maps return Z despite the nonzero cycle field.

Response on heterogeneous branches. On a scalar triangle with representatives $(1,2),(2,3),(3,1)$, unit TV weights, and $N = (1,1,1)^\top$, take $M = 0$ and $Z = (a,0,0)^\top$ with $a > 1/\sigma$. The first and third inputs lie outside the TV dual threshold and the second inside it, so $J^0 = \mathrm{diag}(0,1,0)$. Thus,

$$\dot{U}^1 = \tau\sigma\chi K^\top J^0 N = \tau\sigma\chi(0,1,-1)^\top \neq 0. \tag{48}$$

Equation 44 propagates this first-update response to the chosen depth. Taking its inner product with the downstream gradient gives the change in the loss.

Loss and OffsetNet parameter derivatives. For a differentiable downstream loss ℓ, including later-layer recomputation in evaluation mode, the chain rule gives

$$D_M \ell[N] = \langle \nabla_{U^T} \ell, \dot{U}^T \rangle = \langle \Pi_\perp \nabla_M \ell, N \rangle. \tag{49}$$

The second equality follows from $N = \Pi_\perp N$ and the self-adjointness of the orthogonal projector. It expresses the loss derivative in terms of the unrolled layer response.

For OffsetNet parameters φ at fixed hidden states, a perturbation $\delta\varphi$ changes the target by $D_\varphi M[\delta\varphi]$ and the loss by $\langle \nabla_M \ell, D_\varphi M[\delta\varphi] \rangle$. A pure-cycle direction N is locally attainable when $D_\varphi M[\delta\varphi] = N$ for some $\delta\varphi$. In a stacked network, Eq. 49 holds the intervened layer's anchor and weights fixed and includes downstream recomputation through ℓ. Appendix B.5 measures full parameter gradients.

A.7 Exact-Map Identification

Theorem 2 (Identification up to node translation for the exact family)**.** *Fix a simple loop-free positive-weight support graph, its weights W, feature width $d \ge 1$, and $\alpha > 0$. Let g_θ be finite, convex and*

nondecreasing on $[0,\infty)$*, with* $g_\theta(0)=0$*, differentiability at positive radii, and finite positive right derivative* $g'_\theta(0)>0$*. For target fields* M, N *and a node field* B*,*

$$P_{\theta,W,M}(Z)=B+P_{\theta,W,N}(Z-B) \quad \textit{for every } Z \quad \Longleftrightarrow \quad M_+=N_++K_+B. \tag{50}$$

TV and LP satisfy these assumptions.

Proof. Sufficiency is the coordinate substitution in Appendix A.2. For necessity, equality of the proximal maps implies equality of the corresponding convex regularizers up to a constant. Indeed, $U=\operatorname{prox}_f(U+S)$ exactly when $S\in\partial f(U)$. Therefore, equality of proximal maps gives equality of subdifferentials everywhere. The restrictions of finite convex functions to any segment are absolutely continuous with the same one-sided directional derivatives, so their increments agree. Applied to the scaled regularizers, this gives

$$R_{\theta,W,M}(U)=R_{\theta,W,N}(U-B)+c. \tag{51}$$

For edge $e=(i,j)$, put $A_eU=U_i-U_j$ and $\mathcal{H}_e(M)=\{U : A_eU=\mu_e\}$, an affine space of codimension d. Let $q_0=g'_\theta(0)>0$ be the origin slope and $\mathbb{B}_d$ the closed unit ball in $\mathbb{R}^d$. At zero residual, the edge summand has the multivalued subdifferential $w_eq_0A_e^\top\mathbb{B}_d$. At positive residual it is differentiable. The convex sum rule preserves this nondifferentiability, giving the set $\bigcup_{e\in E_+}\mathcal{H}_e(M)$.

Equality up to a constant makes this union equal to $\bigcup_{f\in E_+}\mathcal{H}_f(N+KB)$. A finite union of proper affine subspaces has empty relative interior. Therefore, for each e, some f has $\mathcal{H}_e(M)=\mathcal{H}_f(N+KB)$. Equal affine spaces have the same direction kernel. Distinct unordered endpoint pairs have different kernels, and the graph is simple, so $e=f$. Equality of the target-equality sets then gives $\mu_e=\nu_e+(K_+B)_e$ on every active edge. □

Quadratic contrast. For $g(s)=s^2/2$ and unit weights, $M=KB+N$ with $K^\top N=0$ gives

$$R_{W,M}(U)=\tfrac{1}{2}\|K(U-B)\|_F^2+\tfrac{1}{2}\|N\|_F^2. \tag{52}$$

The cycle term is constant in U, so it leaves the translated exact map unchanged. With nonuniform positive weights, take $B_W\in\arg\min_B\sum_e w_e\|\mu_e-(KB)_e\|_2^2$ and $N_W=M-KB_W$. Then $K^\top\operatorname{diag}(W)N_W=0$, and the expansion holds in the weighted norm. Its zero origin slope distinguishes the quadratic potential from TV and LP.

One cycle field at two anchors. On the scalar unit-weight TV triangle with representatives $(1,2),(2,3),(3,1)$, let $\alpha=1$ and $M=(1,1,1)^\top$. The penalty is at least 3 because endpoint differences sum to zero. At $Z=0$, the learned-target and zero-target exact outputs both equal zero. At $Z=(1/2,0,0)^\top$, the target-centered penalty attains its lower bound at the anchor and its minimizer equals Z. The zero-target output is $(1/6,1/6,1/6)^\top$, which satisfies the optimality condition with dual field $(1/6,0,-1/6)^\top$. For these two target fields, the outputs agree at one anchor and differ at another.

A.8 Exact Stability and Outer Convergence

For fixed W, M, θ, α, exact proximal optimality gives $Z_i-P(Z_i)\in\alpha\partial R(P(Z_i))$. Monotonicity yields

$$\|P(Z_1)-P(Z_2)\|_F^2\le\langle P(Z_1)-P(Z_2),Z_1-Z_2\rangle. \tag{53}$$

This is firm nonexpansiveness of the exact map with W, M, θ, α fixed. Theorem 1 treats the finite recurrence.

With exact edge responses and extrapolation $\bar U^{t+1}=2U^{t+1}-U^t$, Chambolle-Pock converges as the outer depth grows under $\tau\sigma\|K\|^2<1$ (Chambolle and Pock, 2011). The shifted conjugate $[F(\cdot-M)]^*(Y)=F^*(Y)+\langle Y,M\rangle$ gives the target term in the dual update. Continuity of F, existence of the strongly convex primal minimizer, and finite-dimensionality give the saddle-point conditions. Our step sizes satisfy the required inequality through $\|K\|^2\le 2d_{\max}$. The radial residuals account for approximate LP responses at finite depth. On an edgeless graph, the layer returns Z directly.

A.9 A POSTERIORI OUTPUT BOUNDS

We use convex duality and strong convexity to bound the distance from an approximate candidate to the exact output, using either a primal-dual gap or a stationarity residual.

Primal-dual gap. For $F^*(Y) < \infty$, define

$$\mathcal{D}_Z(Y, M) = \langle Y, KZ - M\rangle - \tfrac{\alpha}{2}\|K^\top Y\|_F^2 - F^*(Y), \qquad \mathcal{G}_Z(U, Y, M) = \Phi_Z(U, M) - \mathcal{D}_Z(Y, M). \tag{54}$$

Here, $\mathcal{D}_Z$ is the dual objective and $\mathcal{G}_Z$ is the primal-dual gap. Weak duality and strong convexity give

$$\|U - P_{\theta,W,M}(Z)\|_F \le \sqrt{2\alpha \mathcal{G}_Z(U, Y, M)}. \tag{55}$$

TV has $F^*(Y) = 0$ on the balls $\|Y_e\|_2 \le w_e$, and $+\infty$ outside. For LP, define the scalar conjugate γ_θ on nonnegative arguments by

$$\gamma_\theta(p) = \sup_{s \ge 0}\{ps - g_\theta(s)\}, \qquad C_\theta = \sum_{k=1}^{K_\psi} \frac{a_k}{\beta_k}\,\mathrm{softplus}(-c_k). \tag{56}$$

Here, $p \ge 0$ is the normalized dual radius, and C_θ is the limiting conjugate value at L_θ. For $\psi_\theta(0) < p < L_\theta$, let $s_p > 0$ solve $\psi_\theta(s_p) = p$. The sigmoid mixture gives

$$\gamma_\theta(p) = \begin{cases} 0, & 0 \le p \le \psi_\theta(0), \\ ps_p - g_\theta(s_p), & \psi_\theta(0) < p < L_\theta, \\ C_\theta, & p = L_\theta, \\ +\infty, & p > L_\theta. \end{cases} \tag{57}$$

The first branch follows from the derivative sign at the origin, and the interior branch from strict monotonicity of ψ_θ. At the upper endpoint, the explicit potential satisfies $L_\theta s - g_\theta(s) \to C_\theta$ as $s \to \infty$. Values above L_θ give an unbounded supremum. Consequently,

$$F^*(Y) = \sum_{e \in E_+} w_e \gamma_\theta(\|Y_e\|_2 / w_e), \qquad Y_e = 0 \quad (e \notin E_+). \tag{58}$$

The dual-response bound in Appendix A.3 places the implemented edge field in this finite conjugate domain in exact arithmetic.

For an interior dual radius p, let $[l, u]$ satisfy $0 \le l \le u$ and $\psi_\theta(l) \le p \le \psi_\theta(u)$, and define $h_p(s) = ps - g_\theta(s)$. This concave function attains its maximum at $s_p \in [l, u]$. Its endpoint values and supporting tangents give

$$\begin{aligned} \max\{h_p(l), h_p(u)\} &\le \gamma_\theta(p) \\ &\le \min_{s \in \{l,u\}} \big\{h_p(s) + |p - \psi_\theta(s)|(u - l)\big\}. \end{aligned} \tag{59}$$

The upper bound uses the distance $|s - s_p| \le u - l$ at either endpoint. Substituting these scalar upper bounds into F^* gives a lower dual objective and a conservative output bound. Evaluating $h_p(s)$ at a single radius gives a lower bound on the conjugate.

A direct stationarity bound. A direct stationarity calculation gives a simpler, sometimes looser bound. Set $S = KU - M$ and choose the edge field V as

$$V_e = \begin{cases} w_e \psi_\theta(\|S_e\|_2) S_e / \|S_e\|_2, & S_e \ne 0, \\ \text{any vector with } \|V_e\|_2 \le w_e \psi_\theta(0), & S_e = 0. \end{cases} \qquad b(U, V) = \frac{U - Z}{\alpha} + K^\top V. \tag{60}$$

Then $b(U, V) \in \partial \Phi_Z(U, M)$. Strong monotonicity against $0 \in \partial \Phi_Z(P(Z), M)$ gives

$$\|U - P_{\theta,W,M}(Z)\|_F \le \alpha \|b(U, V)\|_F. \tag{61}$$

At zero residual, projecting a recorded dual row onto the subgradient ball gives a valid choice of V_e. At nonzero residual, the radial formula determines it.

A.10 Two-Node Interpretation

For a single edge of weight w and target μ, define $z_c = (Z_1 + Z_2)/2$, $u_c = (U_1 + U_2)/2$, $z_\Delta = Z_1 - Z_2$, and $v = U_1 - U_2$. Then

$$\Phi_Z(U, \mu) = \frac{1}{\alpha}\|u_c - z_c\|_2^2 + \frac{1}{4\alpha}\|v - z_\Delta\|_2^2 + w g_\theta(\|v - \mu\|_2). \tag{62}$$

The mean is $u_c = z_c$. Setting $x = z_\Delta - \mu$ and $r = \|x\|_2$, radial reduction with scale $2\alpha w$ gives $v^\star = \mu$ when $r \le 2\alpha w \psi_\theta(0)$ and $v^\star = \mu + (s^\star/r)x$ otherwise. The endpoints are $U_1^\star = z_c + v^\star/2$ and $U_2^\star = z_c - v^\star/2$. In TV, $s^\star = \max(r - 2\alpha w, 0)$. Every single-edge target is integrable and gives the preferred endpoint difference at a fixed anchor.

A.11 Dictionary and Clipping Contributions

Write $M_{\mathrm{raw}} = M_{\mathrm{dic}} + M_{\mathrm{lin}}$, where M_{dic} is the scaled antisymmetric dictionary contribution and $M_{\mathrm{lin}} = s_\mu K H D_\mu$ is the linear node-difference contribution. Clipping acts through $T_\mu(X) = \mu_{\max} \tanh(X/\mu_{\max})$. The orthogonal projectors are $\Pi_{\mathcal{I}} = K(K^\top K)^\dagger K^\top$ and $\Pi_\perp = I - \Pi_{\mathcal{I}}$, with the dagger denoting the Moore-Penrose pseudoinverse. Since $\Pi_\perp M_{\mathrm{lin}} = 0$,

$$M_\perp = \underbrace{\Pi_\perp M_{\mathrm{dic}}}_{N_{\mathrm{dic}}} + \underbrace{\Pi_\perp \big(T_\mu(M_{\mathrm{raw}}) - M_{\mathrm{raw}}\big)}_{N_{\mathrm{clip}}}. \tag{63}$$

The fields N_{dic} and N_{clip} are the dictionary and clipping contributions to the cycle component. Their interaction determines the total cycle energy:

$$\|M_\perp\|_F^2 = \|N_{\mathrm{dic}}\|_F^2 + \|N_{\mathrm{clip}}\|_F^2 + 2\langle N_{\mathrm{dic}}, N_{\mathrm{clip}}\rangle. \tag{64}$$

For $u \ge 0$, $u - \tanh u = \int_0^u \tanh^2 v \, dv \le u^3/3$. Oddness extends the absolute inequality to all real u. Entrywise application and contraction of orthogonal projection give

$$\|N_{\mathrm{clip}}\|_F \le \frac{(\sum_{e,j} |(M_{\mathrm{raw}})_{ej}|^6)^{1/2}}{3\mu_{\max}^2} \le \frac{\|M_{\mathrm{raw}}\|_\infty^2 \|M_{\mathrm{raw}}\|_F}{3\mu_{\max}^2}. \tag{65}$$

On a cyclically oriented scalar triangle, take $B = (2a, a, 0)^\top$, $a > 0$, and $M_{\mathrm{raw}} = KB = (a, a, -2a)^\top$ with zero dictionary contribution. Writing $u = \tanh(a/\mu_{\max})$, the clipped circulation is $2\mu_{\max} u^3/(1 + u^2) > 0$. Clipping alone can generate a cycle component from the linear branch.

B Implementation Details

B.1 Architecture and Trainable Parameterization

Tensor shapes and edge networks. The anchor branches in Section 2.2 map d_ℓ incoming features to $d_{\ell+1}$ output features. WeightNet and OffsetNet each receive endpoint features of width $3d_\ell$. Their multilayer perceptrons contain three affine maps, with GELU after each of the first two maps and hidden width $d_{\mathrm{edge}} = 128$:

$$\begin{aligned} \mathrm{MLP}_w &: 3d_\ell \longrightarrow d_{\mathrm{edge}} \longrightarrow d_{\mathrm{edge}} \longrightarrow 1, \\ \mathrm{MLP}_\pi &: 3d_\ell \longrightarrow d_{\mathrm{edge}} \longrightarrow d_{\mathrm{edge}} \longrightarrow K_\mu. \end{aligned} \tag{66}$$

Each affine map includes a bias. WeightNet produces a scalar score, and OffsetNet produces dictionary logits. The dictionary has shape $K_\mu \times d_{\ell+1}$, and the bias-free difference projection $D_\mu^{(\ell)}$ has shape $d_\ell \times d_{\ell+1}$. We antisymmetrize the ordered dictionary outputs, combine them with the projected endpoint difference, scale, and clip them as in Eq. 8.

Edge-network initialization. We initialize the final affine weights and biases of both edge networks to zero. WeightNet starts with $s_w = 0.08$ and uses $\varepsilon_w = 10^{-8}$, so all edges have equal initial weights after mean normalization. OffsetNet starts with $s_\mu = 0.12$ and $D_\mu = 0$, and initializes every dictionary coordinate independently from $\mathcal{N}(0, 0.02^2)$. The zero logits give uniform dictionary coefficients in both orientations, so antisymmetrization yields $M = 0$ at initialization. The target model starts at the zero-target interaction, and the distinct sampled dictionary vectors remain trainable.

Anchor parameterization. Each branch coefficient is parameterized by $s_j = 2\,\mathrm{sigmoid}(\tilde{s}_j)$ for $j \in \{0,1,2,3\}$, where $\tilde{s}_j$ is an unconstrained scalar initialized to zero. The four coefficients start at one and remain in $(0,2)$. The branch matrices use bias-free affine maps, and the shared bias b_z starts at zero. The high-pass field is $H_{\mathrm{hp}} = H - N_1$, so it reparameterizes self and one-hop information. For one layer, suppressing its index, the expression before layer normalization is

$$H A_{\mathrm{self}} + N_1 A_{1\mathrm{hop}} + N_2 A_{2\mathrm{hop}} + b_z, \tag{67}$$

where the effective branch matrices are $A_{\mathrm{self}} = s_0 W_{\mathrm{self}} + s_2 W_{\mathrm{hp}}$, $A_{1\mathrm{hop}} = s_1 W_{1\mathrm{hop}} - s_2 W_{\mathrm{hp}}$, and $A_{2\mathrm{hop}} = s_3 W_{2\mathrm{hop}}$. The implemented four-branch parameterization assigns separate trainable matrices and scales to these paths. All shared-backbone controls use that same parameterization.

Layerwise tensor evaluation. We compute the anchor, weights, and targets from the incoming state once per layer call. We use the same tensors throughout the updates, and each weight multiplies every feature coordinate in its edge row. Differentiation through the recurrence reaches the edge networks and potential parameters. Appendix C.1 specifies the layerwise order for checkpoint interventions.

Positive potential parameters. The LP amplitudes, slopes, and base use

$$a_k = \mathrm{softplus}(\tilde{a}_k), \qquad \beta_k = \mathrm{softplus}(\tilde{\beta}_k), \qquad b_0 = \mathrm{softplus}(\tilde{b}_0). \tag{68}$$

The tilded scalars are unconstrained trainable parameters, and the shifts c_k are learned directly. This parameterization gives a convex increasing g_θ. Integrating the sigmoid bases yields

$$g_\theta(t) = b_0 t + \sum_{k=1}^{K_\psi} \frac{a_k}{\beta_k} \left[\mathrm{softplus}(\beta_k t + c_k) - \mathrm{softplus}(c_k)\right]. \tag{69}$$

This expression satisfies $g_\theta(0) = 0$ and $g'_\theta = \psi_\theta$. Each LP layer has $3K_\psi + 1$ potential parameters: the three parameters of every sigmoid basis and the base slope. With the specified two layers and eight bases, the potential contributes 50 trainable scalars. TV uses the parameter-free potential $g_{\mathrm{TV}}(t) = t$.

Potential initialization. The initial unconstrained LP parameters are $\tilde{b}_0 = -2.25$, $\tilde{a}_k = -5$, and $\tilde{\beta}_k = -1$. For the eight-basis configuration, the initial shifts are evenly spaced:

$$c_k = -3 + \frac{6(k-1)}{K_\psi - 1}, \qquad k = 1, \ldots, K_\psi. \tag{70}$$

We use these values to initialize the sigmoid mixture before supervised optimization. Each layer has its own potential parameters; the TV and quadratic variants use their stated closed-form potentials.

Finite computation used in training. Projected Newton supplies the radial responses in LP training, evaluation, checkpoint interventions, gradient measurements, and profiling. The depth studies vary its update count. The bracketed routine with residual-controlled refinement is evaluated on the scalar grid in Appendix B.2, where each input has a directly evaluated optimality residual.

Hardware. Graph experiments run on NVIDIA A100 GPUs, with additional A100 devices for the larger graphs. The scalar study uses binary64 CPU arithmetic. Appendix B.3 reports single-device measurements, and Figure 6 compares training time across computational depths.

Shared backbone and reference training settings. Table 5 collects the architecture and reference training configuration. The network has two LEGO layers and uses the concatenated encoder and layer outputs for classification. We apply dropout in the encoder, residual graph blocks, and classifier. Layer normalization follows the encoder, anchor block, and residual graph blocks, and the encoder and classifier use GELU. Node features are ℓ_2-normalized before training.

The extrapolation is $\bar{U}^{t+1} = 2U^{t+1} - U^t$. Each positive-root LP call starts at its input radius and applies the selected number of projected Newton updates. TV uses closed-form responses within the same outer recurrence. The complete layer computation is given in Algorithm 1.

Table 5: Shared-backbone architecture and reference training configuration. Appendix B.6 gives the validation candidates.

Quantity	Setting
Hidden and edge-network widths d, d_{edge}	128
LEGO layers L	2
Target dictionary size K_μ	8
LP basis count K_ψ	8
Anchor scale α	1.0
Step-size factor κ	0.9
Outer updates K_{iter}	12
LP projected Newton updates T_{newton}	8
Target clipping bound $\mu_{\max}$	0.75
Dropout probability	0.6
Optimizer	AdamW
Learning rate	2×10^{-3}
Weight decay	5×10^{-4}
Mean-weight penalty λ_w	10^{-4}
Target penalty λ_μ	5×10^{-5}
Global gradient-norm clipping threshold	1.0
Label smoothing	0.0
Maximum training epochs	2000
Early-stopping patience	200
Plateau schedule factor and patience	0.5, 50
Minimum learning rate	10^{-5}

Readout and supervised loss. After L layers, concatenate the encoded input and layer outputs into $H_{\mathrm{cat}} = [H^{(0)} \| \cdots \| H^{(L)}]$. The classifier produces logits $\Lambda \in \mathbb{R}^{n \times c_{\mathrm{cls}}}$, where c_{cls} is the number of classes:

$$\Lambda = \mathrm{Dropout}\left(\phi(\mathrm{LayerNorm}(H_{\mathrm{cat}})W_1 + b_1)\right) W_2 + b_2. \tag{71}$$

The activation ϕ is GELU, and W_1, W_2, b_1, b_2 are trainable classifier parameters. For training nodes V_{lab} with labels y_i, supervised cross-entropy is $\mathcal{L}_{\mathrm{sup}} = -|V_{\mathrm{lab}}|^{-1} \sum_{i \in V_{\mathrm{lab}}} \log \mathrm{softmax}(\Lambda_i)_{y_i}$.

We stack L copies of Algorithm 1 and differentiate $\mathcal{L}_{\mathrm{train}}$ through the executed arithmetic branches of the backbone, WeightNet, OffsetNet, and LP parameters (Appendix B.5).

Effective coupling and amplitude rescaling. On a nonempty edge set, write $w_e = s_w \hat{w}_e$ with $m^{-1} \sum_e \hat{w}_e = 1$. Let

$$\bar{g}_\theta = g_\theta / L_\theta, \qquad v_e = w_e L_\theta, \qquad \rho = s_w L_\theta. \tag{72}$$

The normalized potential is $\bar{g}_\theta$, v_e is the edgewise response bound, and ρ is its mean. Thus, $w_e g_\theta = v_e \bar{g}_\theta$ and $\|Y_e^t\|_2 \le v_e$, with $m^{-1} \sum_e v_e = \rho$. For TV, $L_\theta = 1$.

In the LP parameterization, the transformation

$$w'_e = w_e / c, \qquad b'_0 = c b_0, \qquad a'_k = c a_k, \qquad \beta'_k = \beta_k, \quad c'_k = c_k \quad (c > 0), \tag{73}$$

preserves $w_e g_\theta$, the threshold decisions, the projected Newton updates, and the effective coupling ρ. The explicit mean-weight penalty becomes $\lambda_w s_w / c$ and acts specifically on the weight factor in the product $s_w L_\theta$. Mean normalization fixes the sum of relative weights, whose supervised learning signal changes their distribution across edges. AdamW applies weight decay to the raw trainable parameters, which also change under this amplitude rescaling.

B.2 Learned-Potential and Closed-Form TV Routines

TV and LP share the anchored outer recurrence. Their edge-response calculations are given below, with LP using the projected Newton routine specified for training.

Closed-form primal edge proximal. For edge input $x \in \mathbb{R}^d$, weight $w_e \ge 0$, and dual step size $\sigma > 0$, define the input radius r, scaled weight λ_e, and TV output radius $s^\star_{\mathrm{TV},e}$ by

$$r = \|x\|_2, \quad \lambda_e = \frac{w_e}{\sigma}, \quad s^\star_{\mathrm{TV},e} = \max\{r - \lambda_e, 0\}. \tag{74}$$

Algorithm 1 LEGO layer with a learned potential or total variation

Require: Hidden features $H^{(\ell)}$, graph E, potential choice θ, parameters $\alpha, \kappa, K_{\mathrm{iter}}, T_{\mathrm{newton}}, \mu_{\max}$
Ensure: $H^{(\ell+1)}$

Form $Z^{(\ell)}$ from the self, one-hop, high-pass, and two-hop branches, then apply layer normalization
if $|E| = 0$ **then**
    $\widehat{U}^{(\ell+1)} \leftarrow Z^{(\ell)}$
else
    Compute edge weights $W^{(\ell)}$ from symmetric edge features using MLP_w, softplus, edge-mean normalization, and a learnable scale
    Compute edge offsets $M^{(\ell)}$ from a dictionary mixture, local difference projection, antisymmetrization, and elementwise $\tanh$ clipping
    Set step sizes $\tau = \sigma = \kappa/\sqrt{2d_{\max}}$
    Initialize $U^0 = Z^{(\ell)}$, $\bar{U}^0 = U^0$, $Y^0 = 0$
    for $t = 0, \ldots, K_{\mathrm{iter}} - 1$ **do**
        $\widetilde{Y}^{t+1} \leftarrow Y^t + \sigma(K\bar{U}^t - M^{(\ell)})$
        if the potential choice is TV **then**
            Compute $\widehat{S}^{t+1}$ by the closed-form radial shrinkage in the main text
        else
            Compute $\widehat{S}^{t+1}$ by radial reduction and T_{newton} projected Newton steps initialized at the input radius
        $Y^{t+1} \leftarrow \widetilde{Y}^{t+1} - \sigma\widehat{S}^{t+1}$
        $U^{t+1} \leftarrow \big(U^t - \tau K^\top Y^{t+1} + (\tau/\alpha)Z^{(\ell)}\big)/(1 + \tau/\alpha)$
        $\bar{U}^{t+1} \leftarrow 2U^{t+1} - U^t$
    $\widehat{U}^{(\ell+1)} \leftarrow U^{K_{\mathrm{iter}}}$
$H^{(\ell+1)} \leftarrow \mathrm{LayerNorm}\left(R^{(\ell)}_{\mathrm{res}}(H^{(\ell)}) + \mathrm{Dropout}(\widehat{U}^{(\ell+1)})\right)$
return $H^{(\ell+1)}$

Minimizing $\frac{1}{2}(s-r)^2 + \lambda_e s$ over $s \geq 0$ gives the thresholded radius above and the radial edge response

$$\mathrm{prox}_{\lambda_e\|\cdot\|_2}(x) = \begin{cases} (s^\star_{\mathrm{TV},e}/r)x, & r > 0, \\ 0, & r = 0. \end{cases} \tag{75}$$

We evaluate the positive-radius branch after testing r and insert its output into the Moreau update. With a zero weight, the response equals the input; a positive weight sets the shrinkage threshold.

Equivalent direct dual projection. The conjugate of the TV edge penalty is the indicator of a product of Euclidean balls, one of radius w_e for each edge. Consequently, the same dual update can be evaluated directly from the intermediate field $\widetilde{Y}^{t+1}$:

$$Y_e^{t+1} = \begin{cases} \widetilde{Y}_e^{t+1}, & \|\widetilde{Y}_e^{t+1}\|_2 \leq w_e, \\ \dfrac{w_e}{\|\widetilde{Y}_e^{t+1}\|_2}\widetilde{Y}_e^{t+1}, & \|\widetilde{Y}_e^{t+1}\|_2 > w_e. \end{cases} \tag{76}$$

This radial projection preserves inputs inside the ball, including the origin, and maps exterior inputs to its boundary. This projection is equivalent to the Moreau expression.

Learned-potential radial routine. For LP, let $r = \|x\|_2$ and $\lambda_e = w_e/\sigma$. The response is zero when $r = 0$ or $\lambda_e\psi_\theta(0) \geq r$, and equals x when $\lambda_e = 0$. On the positive-root branch, initialize at the input radius and define

$$\Gamma_e(s) = s + \lambda_e\psi_\theta(s) - r, \quad s^{(0)} = r, \tag{77}$$

where Γ_e is the scalar optimality residual. We apply

$$s^{(q+1)} = \Pi_{[0,r]}\left(s^{(q)} - \frac{\Gamma_e(s^{(q)})}{1 + \lambda_e\psi'_\theta(s^{(q)})}\right), \quad q = 0, \ldots, T_{\mathrm{newton}} - 1. \tag{78}$$

The projection $\Pi_{[0,r]}$ restricts each iterate to the feasible interval. The selected depth $T_{\mathrm{newton}} = 8$ returns $\hat{s} = s^{(T_{\mathrm{newton}})}$ and edge output $(\hat{s}/r)x$, with gradients following the executed branches. The bound in Appendix A.3 controls dual-response magnitude, and Appendix A.4 controls radial error through $|\Gamma_e(\hat{s})|$. Table 6 evaluates this residual on the deterministic scalar grid.

Safeguarded scalar comparison. Our scalar study compares the training routine with bracketed Newton and residual-controlled refinement. Both use the same threshold branches. On the positive-root branch, initialize $l_0 = 0$, $h_0 = r$, and $s_0 = r$, so $\Gamma_e(l_0) < 0 < \Gamma_e(h_0)$. At update q, the interval $[l_q, h_q]$ brackets the root and s_q is the current radius. Define the Newton proposal n_q and bracket width d_q as

$$n_q = s_q - \frac{\Gamma_e(s_q)}{1 + \lambda_e \psi'_\theta(s_q)}, \quad d_q = h_q - l_q. \tag{79}$$

We accept a finite proposal in $[l_q + d_q/4, h_q - d_q/4]$ and otherwise use the midpoint. The residual sign selects the endpoint to replace, with zero residual collapsing the bracket to the root. The routine permits $T_{\mathrm{sg}} = 8$ safeguarded updates and then at most $J_{\mathrm{bis}} = 64$ midpoint bisections. Either phase stops at

$$|\Gamma_e(s_q)| \le \varepsilon_e, \quad \varepsilon_e = 10^{-6}(1 + r). \tag{80}$$

Here, ε_e includes both absolute and radius-scaled tolerance terms. The exceedance count includes terminal residuals above this tolerance and nonfinite evaluations, with all tested inputs retained in the denominator.

The central-half acceptance rule contracts bracket width by at least a factor of $3/4$, and each bisection halves it. Define

$$C_e = 1 + \frac{\lambda_e}{4} \sum_k a_k \beta_k. \tag{81}$$

The constant C_e bounds Γ'_e from above. After T_{sg} safeguarded updates and j bisections, the exact-arithmetic residual is at most $C_e(3/4)^{T_{\mathrm{sg}}} 2^{-j} r$. The binary64 experiment reports the evaluated residual.

Projected Newton scalar behavior. A steep sigmoid mixture gives an explicit projected Newton two-cycle. Set $K_\psi = 8$, $b_0 = 0.1$, $a_k = 0.25$, $\beta_k = 100$, and $c_k = -50$ for every basis, with $r = \lambda_e = 1$. Then

$$\psi_\theta(s) = 0.1 + 2\,\mathrm{sigmoid}(100s - 50), \quad s^\star \approx 0.4865521296837064. \tag{82}$$

We computed the reference root to 80 decimal digits and evaluated both finite routines in binary64. Projected Newton follows

$$1 \longrightarrow 0 \longrightarrow 0.9 \longrightarrow 0 \longrightarrow 0.9 \longrightarrow \cdots, \quad s^{(8)} = s^{(16)} = 0.9, \quad |\Gamma_e(0.9)| = 2. \tag{83}$$

The safeguarded routine reaches $\hat{s} \approx 0.4865521320656336$ after eight safeguarded steps and fourteen bisections, with $|\Gamma_e(\hat{s})| \approx 8.05 \times 10^{-8}$. Projected Newton alternates between two values and returns the same output at both even depths, with a nonzero optimality residual.

Deterministic scalar stress grid. We next evaluate a grid in the same eight-basis family. Let $A > 0$ be the total sigmoid amplitude and t_0 its common transition center. Set $a_k = A/8$, $\beta_k = \beta$, and $c_k = -\beta t_0$, and evaluate every combination of

$$\begin{gathered} b_0 \in \{0.01, 0.1, 1\}, \quad A \in \{0.2, 2, 20\}, \quad \beta \in \{1, 10, 100, 1000\}, \\ t_0/r \in \{0.25, 0.5, 0.75\}, \quad r \in \{0.1, 1, 10\}, \quad \lambda_e \in \{0.01, 0.1, 1, 10\}. \end{gathered} \tag{84}$$

The shared slope is β, and t_0/r locates the transition relative to the input radius. This Cartesian grid contains 1,296 cases: 879 positive-root cases and 417 exact-zero cases. We evaluate the positive-root cases in binary64 and report their maximum residual, upper-tail quantiles, and tolerance-exceedance counts in Table 6.

Residual-controlled refinement satisfies the radius-scaled tolerance across the grid. The maximum number of additional bisections is 24. The quantiles are computed from the inverse empirical distribution over all positive-root cases, including tolerance exceedances.

Table 6: Radial optimality on 879 positive-root cases from the deterministic stress grid. Residual columns summarize $|\Gamma_e(\hat{s})|$ in binary64. The final column counts inputs with residual above $10^{-6}(1+r)$ or a nonfinite evaluation. All routines use the same inputs.

Scalar routine	Maximum	95th percentile	99th percentile	Tolerance-exceedance count
Projected Newton, 8 steps	2.00×10^{2}	2.00×10^{1}	2.00×10^{2}	$182/879$
Safeguarded phase, 8 steps	7.53	1.32	6.54	$468/879$
Safeguarded phase with refinement	1.09×10^{-5}	7.75×10^{-6}	1.01×10^{-5}	$0/879$

Table 7: Full-model computational profile on one NVIDIA A100. Training-step and incremental-memory entries average the five dataset means, each measured over ten seeds. Inference p95 uses a geometric mean. Ratios compare the corresponding aggregate measurements.

Profile	GELU	Unshifted-LP	LEGO-LP	LEGO-LP/GELU	LEGO-LP/Unshifted-LP
Training step (ms)	22.37	41.49	42.75	$1.91\times$	$1.03\times$
Inference p95 (ms)	5.34	17.53	17.66	$3.31\times$	$1.01\times$
Training incremental peak (GiB)	0.63	2.02	2.15	$3.41\times$	$1.06\times$
Inference incremental peak (MiB)	84.5	202.6	301.4	$3.57\times$	$1.49\times$

Arithmetic and active parameters. Each outer update applies K and $K^\top$ and evaluates the radial responses. Outside the edge networks, TV requires $O(K_{\mathrm{iter}}(n+m)d)$ arithmetic. LP adds $O(K_{\mathrm{iter}} m T_{\mathrm{newton}} K_\psi)$ basis evaluations. The safeguarded comparator uses at most $O((T_{\mathrm{sg}} + J_{\mathrm{bis}} + 1)K_\psi)$ per positive-root input. Both variants maintain node and edge states, storing or recomputing intermediates for reverse-mode differentiation. TV learns weights and targets under a parameter-free potential. Appendix B.4 reports end-to-end LP costs.

B.3 Full-Model Runtime and Memory Profile

We profile the full GELU, Unshifted-LP, and LEGO-LP networks on one NVIDIA A100. The width, layer count, and selected LP depth are those in Table 5. A training step includes the forward pass, backpropagation, and optimizer update, and inference consists of one evaluation-mode pass. Each dataset uses the same graph and measurement procedure across variants.

Table 7 reports training-step time, the 95th percentile of inference latency (p95), and peak allocated memory above the pre-pass level. We measure each dataset and variant over ten seeds. We aggregate training time and memory as the arithmetic mean of the five dataset means, and inference p95 as the geometric mean. Appendix B.4 defines the allocation measurements.

We report training memory in GiB and inference memory in MiB.

OffsetNet mainly increases edge-state memory; its latency impact is smaller. Appendix C.18 describes the edge partitioning used for larger graphs.

B.4 Computational Cost and Memory Measurement

Table 7 uses the allocation measurements below. Figure 6 varies computational depth.

Incremental and total memory. Let B_{alloc} be the allocated bytes immediately before a measured pass and P_{alloc} the largest allocation during that pass, including its starting state. The reported increment is

$$I_{\mathrm{alloc}} = P_{\mathrm{alloc}} - B_{\mathrm{alloc}}. \tag{85}$$

The total peak $B_{\mathrm{alloc}} + I_{\mathrm{alloc}}$ includes parameters, graph tensors, buffers, and optimizer state. Reserved allocator memory is measured separately. The units in the training and inference rows reflect their respective allocation scales.

State storage across graph sizes. With b bytes per scalar, one width-d edge field occupies bmd bytes and a node field bnd bytes. Retaining outer-iteration states for reverse-mode differentiation requires storage of order $bK_{\mathrm{iter}}(n+m)d$, plus edge-network and LP intermediates. Recomputing states trades computation for storage. Edge chunking bounds temporary edge computations. Backward

Table 8: Gradient measurements during finite-unroll training, sampled every twenty steps. The clipping rate counts sampled steps with global pre-clipping norm above 1.0. The final column counts nonfinite gradients in the target, weight, and potential modules.

Dataset	Clipping rate (%)	Nonfinite count
RomanEmpire	0.24	0
Minesweeper	0.22	0
AmazonRatings	5.32	0
Chameleon	4.88	0
Squirrel	8.50	0

storage also depends on the states retained across chunks and iterations. The measured five-graph allocations appear in Table 7.

B.5 Differentiation and Gradient Measurements

LP training differentiates $\mathcal{L}_{\text{train}}$ through the twelve outer updates and eight projected Newton updates to the backbone, WeightNet, OffsetNet, and potential parameters.

We sample gradients every twenty training steps. A clipping event occurs when the global pre-clipping norm exceeds 1.0. Across all datasets, the sampled target, weight, and potential gradients are finite, and each module's mean gradient is nonzero. We report the clipping frequency and nonfinite-gradient counts in the table.

B.6 Hyperparameter and Model Selection

Validation candidates include learning rates $\{10^{-3}, 2\times10^{-3}, 5\times10^{-3}\}$, weight decay $\{0, 5\times10^{-4}, 10^{-3}\}$, widths $\{64, 128\}$, dropout probabilities $\{0.4, 0.5, 0.6\}$, and early-stopping patience $\{100, 200\}$. LP candidates also include $\alpha \in \{0.5, 1.0, 2.0\}$, $K_{\text{iter}} \in \{8, 12, 16\}$, $T_{\text{newton}} \in \{6, 8, 10\}$, $\lambda_w \in \{0, 10^{-4}\}$, and $\lambda_\mu \in \{10^{-5}, 5\times10^{-5}, 10^{-4}\}$. We allocate the same hyperparameter search resources to every method within a comparison.

We select $(K_{\text{iter}}, T_{\text{newton}}) = (12, 8)$ by validation for the shared-backbone study. The final comparisons use width 128, $\alpha = 1$, and dropout 0.6. Table 5 gives the reference configuration, and each sensitivity study varies the indicated parameter around it.

Table 22 uses the TV and LP runs from Table 1 and independently trained quadratic models. We also train the integrable-target and anchor-injection controls from initialization. For each candidate, we select the checkpoint and configuration by validation before test evaluation. Checkpoint interventions act on these selected models.

Checkpoint selection in the graph-coupled models. After each optimizer update, an evaluation-mode pass computes the validation metric. The learning-rate scheduler maximizes this metric, reducing the learning rate by the factor and patience in Table 5. We replace the selected checkpoint only when the validation score is strictly larger; ties keep the earlier checkpoint. Early stopping counts epochs since the last strict improvement. We restore the selected state before the final test evaluation. Minesweeper uses the positive-class softmax probability for ROC-AUC, and the other datasets use the class with the largest logit for accuracy.

B.7 Parameter Accounting for Matched Controls

LEGO adds OffsetNet to the corresponding Unshifted model. Anchor injection and Projected-OffsetNet preserve that module and the total parameter count. GELU-wide increases pointwise width, and Integrable-train learns node potentials. The controls are defined in Appendices C.3 and C.6. For a constant-width layer, the number of trainable WeightNet and OffsetNet scalars are

$$\begin{aligned} N_w &= 3d\, d_{\text{edge}} + d_{\text{edge}}^2 + 3d_{\text{edge}} + 2, \\ N_\mu &= 3d\, d_{\text{edge}} + d_{\text{edge}}^2 + 2d_{\text{edge}} + d_{\text{edge}}K_\mu + K_\mu + K_\mu d + d^2 + 1. \end{aligned} \tag{86}$$

The counts N_w and N_μ include all affine biases and their learned scale parameters. The terms $K_\mu d$ and d^2 count the target dictionary and difference projection. At $d = d_{\text{edge}} = 128$ and $K_\mu = 8$, one layer contributes $65{,}922$ WeightNet and $84{,}233$ OffsetNet parameters. Across two layers, OffsetNet contributes $168{,}466$ parameters, and the LP potentials contribute 50. Layer normalization contributes its trainable affine parameters to the network totals. Residual integration uses parameter-free mixing.

Shared-backbone parameter totals. For input width p, hidden width d, and c_{cls} classes, the two-layer GELU backbone has

$$N_{\text{base}}(p, d, c_{\text{cls}}) = 11d^2 + (p + 20)d + (d + 1)c_{\text{cls}} + 8. \tag{87}$$

The count includes the encoder, four anchor branches per layer, affine normalization parameters, branch scales, and concatenated classifier. The quadratic term combines the eight anchor matrices with the classifier's $3d$-to-d map. At the shared hidden width, Unshifted-LP adds $2N_w + 2(3K_\psi + 1)$ parameters and LEGO-LP adds a further $2N_\mu$. Table 9 gives these totals. Anchor injection and Projected-OffsetNet have the LEGO-LP totals because their additional operations add no parameters. GELU-wide applies the same backbone count at its wider feature dimension.

Table 9: Trainable parameter counts for the shared two-layer, width-128 architecture. The LEGO-LP column also applies to anchor injection and Projected-OffsetNet. Counts follow the architecture and edge-network formulas.

Dataset	GELU	Unshifted-LP	LEGO-LP
RomanEmpire	223,514	355,408	523,874
Minesweeper	183,946	315,840	484,306
AmazonRatings	221,837	353,731	522,197
Chameleon	481,037	612,931	781,397
Squirrel	450,829	582,723	751,189

B.8 Gauge-Fixed Computation of the Projected Controls

The projected controls use the minimum-norm node field $B^\dagger$ from

$$\min_{B \in \mathbb{R}^{n \times d}} \|M_+ - K_+ B\|_F^2, \tag{88}$$

with zero mean on every support component. For one feature column $\boldsymbol{\mu}_+$ of M_+, write $b^\dagger \in \mathbb{R}^n$ for its node-potential coordinate. The normal equations are

$$L_+ b^\dagger = K_+^\top \boldsymbol{\mu}_+, \qquad L_+ = K_+^\top K_+. \tag{89}$$

The support Laplacian is L_+. Its null space consists of componentwise constants, so the zero-mean constraints select the unique node-potential solution

$$b^\dagger = L_+^\dagger K_+^\top \boldsymbol{\mu}_+. \tag{90}$$

The Moore-Penrose pseudoinverse is denoted by $L_+^\dagger$. Applying Eq. 89 to all feature columns gives $B^\dagger$ and the integrable target $K_+ B^\dagger$. The Laplacian and component partition depend only on the incidence graph. The right-hand side changes with the target generated at each layer. We use a Euclidean projection and apply the learned weight magnitudes in the subsequent graph block.

Nonsingular gauge formulation. Let $C_0 \in \mathbb{R}^{n \times c_+}$ have one normalized indicator column for each support component, with value $|V_j|^{-1/2}$ on component V_j and zero elsewhere. Its columns are orthonormal and span $\ker L_+$. The component-mean projector is $Q_0 = C_0 C_0^\top$, and the zero-mean solution satisfies

$$(L_+ + Q_0) B^\dagger = K_+^\top M_+, \qquad C_0^\top B^\dagger = 0. \tag{91}$$

The matrix $L_+ + Q_0$ is positive definite: L_+ acts positively on componentwise zero-mean fields, and Q_0 acts as the identity on componentwise constants. Multiplication by $C_0^\top$ gives the zero-mean condition because $C_0^\top K_+^\top = 0$. Consequently, the exact integrable projector has the equivalent form

$$\Pi_{\mathcal{I}} = K_+ (L_+ + Q_0)^{-1} K_+^\top. \tag{92}$$

This expression equals the pseudoinverse formulation on the support. For an isolated node, the corresponding row of $B^\dagger$ is zero. $\Pi_{\mathcal{I}}$ acts as the identity on integrable edge fields.

Stacking the coordinate solutions gives the zero-mean node field and residual

$$M_{\perp,+} = M_+ - K_+ B^\dagger. \tag{93}$$

A depth-first spanning defines one fundamental cycle for each non-tree edge in δ_{cyc}. The controls use 0, $K_+B^\dagger$, or $M_{\perp,+}$ according to their specified intervention.

Projection accuracy from the normal-equation residual. For a computed zero-mean node field $\widetilde{B}$, define

$$E_L = K_+^\top (M_+ - K_+\widetilde{B}), \qquad \widetilde{M_\perp} = M_+ - K_+\widetilde{B}. \tag{94}$$

The matrix E_L is the normal-equation residual, and $\widetilde{M_\perp}$ is the edge residual associated with $\widetilde{B}$. For a support with at least one edge, let $\lambda_+(L_+) > 0$ be the smallest positive eigenvalue of L_+. Then

$$\begin{aligned} \|\widetilde{M_\perp} - M_{\perp,+}\|_F^2 &= \|K_+(\widetilde{B} - B^\dagger)\|_F^2 \\ &= \langle E_L, L_+^\dagger E_L\rangle \le \frac{\|E_L\|_F^2}{\lambda_+(L_+)}. \end{aligned} \tag{95}$$

The inner product is the Frobenius inner product. On the zero-mean subspace, $E_L = -L_+(\widetilde{B} - B^\dagger)$, and the largest eigenvalue of $L_+^\dagger$ is $1/\lambda_+(L_+)$. These identities give the equality and upper bound. Reconstruction holds for every $\widetilde{B}$ by Appendix A.2. The displayed residual bound quantifies the distance from the orthogonal projection.

Adjoint of the projection. For a scalar loss ℓ and the incoming target gradient $G_M = \nabla_{\Pi_{\mathcal{I}} M}\ell$, differentiation through the exact projection gives

$$\nabla_M \ell = \Pi_{\mathcal{I}} G_M = K_+(L_+ + Q_0)^{-1} K_+^\top G_M. \tag{96}$$

The matrix G_M has the same shape as the projected target. Symmetry of $L_+ + Q_0$ gives the same linear-system operator in the forward projection and its adjoint, with their respective right-hand sides. Appendix C.6 applies this operation to the clipped OffsetNet output.

Projection timing in a stacked network. Projection follows the layerwise order in Appendix C.1. In a translated reconstruction, restoring the node field recovers the original evaluation-mode states and logits by induction.

B.9 Global Projection Distance and Cycle-Normalized Circulation

The two diagnostics in Eq. 123 aggregate target geometry at different scales. Let $\mathcal{B}_{\mathrm{cyc}}$ be the fundamental cycle basis. For $C \in \mathcal{B}_{\mathrm{cyc}}$, let $c_C \in \mathbb{R}^m$ be its signed edge vector, with traversal sign $\sigma_e(C) \in \{-1, 1\}$ on each cycle edge and zero elsewhere. The cycle condition gives $c_C^\top K = 0$, so

$$c_C^\top M = c_C^\top M_\perp, \qquad \delta_{\mathrm{cyc}} = \frac{1}{|\mathcal{B}_{\mathrm{cyc}}|} \sum_{C \in \mathcal{B}_{\mathrm{cyc}}} \frac{\|c_C^\top M_\perp\|_2}{\sum_{e \in C} \|\mu_e\|_2 + \varepsilon_{\mathrm{diag}}}. \tag{97}$$

The numerator is the cycle circulation carried by $M_\perp$. Each fundamental cycle receives equal weight after normalization by its own target-norm sum, and shared edges can contribute to several terms. The global ratio $r_\perp$ instead compares Frobenius norms and counts each edge once.

For the restriction $M_{\perp,C}$ of the residual to the edges in C, Cauchy-Schwarz gives

$$\frac{\|c_C^\top M_\perp\|_2}{\sum_{e \in C} \|\mu_e\|_2 + \varepsilon_{\mathrm{diag}}} \le \frac{\sqrt{|C|}\, \|M_{\perp,C}\|_F}{\sum_{e \in C} \|\mu_e\|_2 + \varepsilon_{\mathrm{diag}}}. \tag{98}$$

The scale of this average depends on cycle length $|C|$, overlap, and the cycle-specific denominators. Normalized circulation depends on the chosen fundamental basis; the projection ratio is basis independent.

Table 10: Benchmark graph sizes. Adjacency entries a count the stored representation used in these experiments.

Statistic	RomanEmpire	Minesweeper	AmazonRatings	Chameleon	Squirrel	Actor	Cornell	Texas	Wisconsin
Nodes	22,662	10,000	24,492	2,277	5,201	7,600	183	183	251
Adjacency entries a	32,927	39,402	93,050	33,824	211,872	33,391	295	309	499
Features	300	7	300	2,325	2,089	932	1,703	1,703	1,703
Classes	18	2	5	5	5	5	5	5	5

An elementary example. Consider a scalar triangle with one pendant edge. Orient the triangle cyclically, list the pendant edge last, and choose the target $M = (a, a, a, b)^\top$ for positive scalars a, b. Its decomposition has $M_\perp = (a, a, a, 0)^\top$ and integrable component $(0, 0, 0, b)^\top$. Consequently,

$$r_\perp = \frac{\sqrt{3}\, a}{\sqrt{3a^2 + b^2} + \varepsilon_{\mathrm{diag}}}, \qquad \delta_{\mathrm{cyc}} = \frac{3a}{3a + \varepsilon_{\mathrm{diag}}}. \tag{99}$$

Increasing the integrable pendant target changes the global norm ratio and leaves the triangle circulation unchanged. The same distinction holds for bounded targets with $0 < a \ll b < \mu_{\max}$ and $\varepsilon_{\mathrm{diag}} \ll a$.

Cycle dimension and target energy. For $m > 0$, cycle-rank density $\rho_{\mathrm{cyc}} = (m - n + c)/m$ gives the proportion of edge-field dimensions in the cycle subspace. All block variants on that graph share it. Strictly positive weights keep LEGO's support equal to the full incidence graph. The ratio $r_\perp$ measures how the learned target occupies that space. For a nonzero field, omitting the diagnostic stabilizer, orthogonality gives

$$r_\perp^2 = \frac{\|M_\perp\|_F^2}{\|KB^\dagger\|_F^2 + \|M_\perp\|_F^2}. \tag{100}$$

On a graph with a cycle, any nonzero $N \in \ker K^\top$ gives a field $M = N$ with $r_\perp = 1$, even at small cycle dimension. On a graph with at least one edge, a nonzero $M = KB$ gives $r_\perp = 0$ for any ρ_{cyc}. Mixtures of the orthogonal components give intermediate ratios.

For comparison, suppose the row-stacked vector $\mathrm{vec}(M)/\|M\|_F$ is uniform on the unit sphere in $\mathbb{R}^{md}$. Its second moment is $I/(md)$, giving

$$\mathbb{E}\left[\frac{\|\Pi_\perp M\|_F^2}{\|M\|_F^2}\right] = \frac{\mathrm{tr}(\Pi_\perp \otimes I_d)}{md} = \frac{m - n + c}{m}. \tag{101}$$

OffsetNet generates targets from endpoint features, with an integrable raw linear branch and cycle contributions from dictionary mixing and clipping. The projection ratio measures how the resulting target is distributed between the two subspaces. Appendix C defines the incidence graph used for $m - n + c$.

C Experimental Details

Graph versions. Table 10 identifies the evaluated graph instances by their node, adjacency, and feature counts. Chameleon and Squirrel use the full-size, duplicate-containing versions discussed in the main experimental setup (Platonov et al., 2023). Appendix C.17 reports the deduplicated graph experiments. RomanEmpire, Minesweeper, and AmazonRatings provide the remaining main comparisons.

Adjacency and incidence conventions. The count a in Table 10 describes the stored adjacency, including any reciprocal or self entries. For its binary adjacency matrix A, the layer constructs the incidence edge set

$$E_{\mathrm{inc}} = \{\{i, j\} : i < j,\ A_{ij} + A_{ji} > 0\}, \qquad m = |E_{\mathrm{inc}}| = \mathrm{rows}(K). \tag{102}$$

The set E_{inc} contains distinct adjacent non-self pairs, each represented by one oriented row of K. If this incidence graph has c connected components, including isolated nodes, then

$$\mathrm{rank}(K) = n - c, \qquad \rho_{\mathrm{cyc}} = \frac{m - n + c}{m} \quad (m > 0). \tag{103}$$

The statistic ρ_{cyc} is the cycle rank normalized by the incidence-edge count. The layer updates, target projections, and cycle calculations use this same incidence graph. Table 17 reports the preprocessed incidence-edge counts for the five main graphs alongside their cycle-rank densities.

C.1 Evaluation Protocol

Partitions and selection. The main five-graph, nine-graph architecture, and WebKB studies use random train/validation/test partitions with nominal proportions $10\%/25\%/65\%$. Cornell, Texas, and Wisconsin constitute WebKB. We generate partitions before training and share them across methods. We use graph structure and node features transductively, training labels for supervision, and validation to select the configuration and checkpoint. The benchmark protocol uses $50\%/25\%/25\%$ partitions (Platonov et al., 2023). Our experiments use the low-label setting specified above.

Training comparisons, checkpoint interventions, and numerical diagnostics. Training comparisons optimize each variant from initialization, pairing split identifiers and common-module initialization seeds and sharing the applicable optimization and early-stopping settings. Within each potential, the LEGO and Unshifted models use the same graph-update depth. Checkpoint interventions retain the selected parameters and alter the stated target field or computation. Numerical diagnostics report finite-depth sensitivity or scalar optimality residuals. We distinguish comparisons of independently trained models from interventions on a trained predictor.

Layerwise intervention order. At each controlled layer, the current hidden state determines the anchor, weights, and target. Projection or reassignment then replaces that target before the graph-coupled computation. Translated evaluations also shift the anchor and restore the node field before residual integration. Later layers recompute their tensors from the resulting hidden states. This order applies to all target interventions.

Randomization and aggregation. The main five-graph study, nine-graph architecture comparison, and WebKB study use ten runs per method. The structural analysis in Appendix C.9 uses five-seed reruns, and the synthetic study uses thirty matched replicates. Reassignment comprises twenty original/permuted evaluation pairs per dataset. We average repeated permutation effects within checkpoints before aggregating at the checkpoint level. Win/tie/loss counts summarize the individual evaluations.

Metrics and uncertainty. We use accuracy for RomanEmpire, AmazonRatings, Chameleon, Squirrel, Actor, Cornell, Texas, and Wisconsin. Minesweeper uses ROC-AUC. Both metrics are expressed on a 0 to 100 scale. Tables 1 and 24 report sample means and standard deviations, with overlapping entries computed from the same runs. The five-graph target-learning contrasts and frozen cycle-removal contrasts include paired intervals. Integrable-training, anchor-injection, direction-control, and WebKB contrasts report differences of sample means. Permutation ties use scores rounded to the displayed precision. Figure 7 averages dataset-level target gains on the common reporting scale.

Shared-backbone and architecture comparisons. GELU uses the shared encoder, four-branch anchor, residual path, normalization, and concatenated readout. Unshifted-TV adds zero-target coupling with positive weights and a fixed TV potential. Unshifted-LP learns the radial potential as well. LEGO-TV and LEGO-LP add OffsetNet and target regularization to their respective zero-target models. We train and evaluate architecture baselines, including DIGRAF and Dir-GNN, with the same partitions, preprocessing, validation rule, and tuning resources. Appendix C.2 specifies their operators.

C.2 Baseline Architectures and Shared-Backbone Controls

We evaluate shared-backbone controls and architecture baselines under Appendix C.1. H_2GCN-RW is our row-normalized H_2GCN variant, and Three-channel is our gated identity, low-pass, and high-pass control. Their definitions below specify the operators used in the experiments and their relation to H_2GCN and TFE-GNN.

Propagation operators. For the baseline descriptions, let $\widetilde{A}$ be the binary, symmetrized adjacency with one self-loop per node. Define

$$\begin{aligned} D_{\mathrm{deg}} &= \mathrm{diag}(\widetilde{A}\mathbf{1}), \\ S_{\mathrm{sym}} &= D_{\mathrm{deg}}^{-1/2}\widetilde{A}D_{\mathrm{deg}}^{-1/2}, \qquad S_{\mathrm{rw}} = D_{\mathrm{deg}}^{-1}\widetilde{A}. \end{aligned} \tag{104}$$

The degree matrix is D_{deg}, $\mathbf{1}$ is the all-ones node vector, and S_{sym} and S_{rw} are the symmetric and row-normalized propagation matrices. Sparse edge aggregation evaluates their products with node features. These baseline operators include self-loops; LEGO's incidence matrix contains one row per distinct non-self pair. $\mathrm{H_2}$GCN-RW, ACM-GCN, and GloGNN additionally normalize each input feature row by its ℓ_1 norm with a denominator floor of 10^{-12} before their feature transformations.

GCN and GAT. The GCN implementation applies an affine feature transformation followed by S_{sym}, then ReLU and dropout at each hidden layer (Kipf and Welling, 2017). Its optional residual projection is added before the configured normalization and activation, and a final node-wise affine map produces class logits. GAT transforms features into attention heads, computes additive source and destination scores, and applies LeakyReLU with slope 0.2 (Veličković et al., 2018). Attention is normalized over edges entering the same destination and receives dropout before message aggregation. Concatenating the head outputs gives the configured hidden width. The hidden layers use ReLU and dropout, followed by an affine classifier.

$\mathrm{H_2}$GCN-RW and GCNII. $\mathrm{H_2}$GCN separates ego features, distinct one-hop and strict two-hop neighborhoods, and intermediate representations (Zhu et al., 2020). The original model uses symmetric degree normalization within each neighborhood. Our $\mathrm{H_2}$GCN-RW variant uses row normalization. For neighborhood distance $q \in \{1, 2\}$, let A_q be the binary adjacency connecting nodes at shortest-path distance exactly q, and let $D_q = \mathrm{diag}(A_q\mathbf{1})$ be its degree matrix. The original operator is $D_q^{-1/2}A_qD_q^{-1/2}$. We evaluate the variant $D_q^{-1}A_q$, setting inverse-degree entries to zero at empty neighborhoods. A nodewise affine map, ReLU, and dropout produce the initial embedding. Each propagation stage concatenates the one-hop and two-hop outputs, and the classifier receives the initial embedding and all subsequent states. With embedding width d_{H2} and J_{H2} stages, the resulting classifier input width is $d_{\mathrm{H2}}(2^{J_{\mathrm{H2}}+1} - 1)$. GCNII combines symmetrically normalized propagation with the initial hidden state and an identity mapping (Chen et al., 2020). Its hidden update is

$$\begin{aligned} H^{(j)} &= \mathrm{ReLU}\left(\left[(1-\alpha_{\mathrm{II}})S_{\mathrm{sym}}\widetilde{H}^{(j-1)} + \alpha_{\mathrm{II}}H^{(0)}\right]\left[(1-\vartheta_j)I + \vartheta_j W_j\right]\right), \\ \vartheta_j &= \log(1 + \lambda_{\mathrm{II}}/j). \end{aligned} \tag{105}$$

The field $\widetilde{H}^{(j-1)}$ is the previous hidden state after dropout, and the layer index starts at $j = 1$. The coefficient α_{II} weights the initial representation, λ_{II} sets the identity-mapping schedule, and W_j is the layer's learned matrix. An affine classifier acts on the final hidden state.

GPRGNN and FAGCN. GPRGNN produces initial class logits with a two-layer perceptron and learns a polynomial in S_{sym} (Chien et al., 2021):

$$\Lambda = \sum_{j=0}^{J_{\mathrm{GPR}}} \omega_j S_{\mathrm{sym}}^j \Lambda_0. \tag{106}$$

The matrices Λ_0 and Λ are the initial and propagated class logits, J_{GPR} is the propagation order, and ω_j are trainable coefficients. We initialize them with personalized PageRank weights, placing the remaining tail mass in the final coefficient. The propagation coefficients form an optimizer group with zero weight decay, and dropout applies to the perceptron and its output before propagation. FAGCN computes a signed edge gate by applying $\tanh$ to an affine map of the concatenated endpoint states (Bo et al., 2021). Symmetric inverse-degree factors scale these gates, and gate dropout precedes aggregation. Each stage adds a scaled copy of the initial hidden representation before the final affine classifier.

ACM-GCN and the three-channel control. ACM-GCN uses separately learned low-pass, high-pass, and nodewise feature maps (Luan et al., 2022). The corresponding operators are S_{rw}, $I - S_{\mathrm{rw}}$,

and I, with ReLU applied to each channel. Scalar projections of the channel outputs pass through a sigmoid and a learned three-by-three mixing matrix. A rowwise softmax gives the channel weights, and the output is three times their weighted sum. The implementation stacks two such channel-mixing layers with intervening dropout.

TFE-GNN forms trainable polynomial ensembles of low-pass and high-pass filters, followed by a learned combination of the two ensembles (Duan et al., 2024). Our Three-channel control has an explicit three-channel parameterization. It encodes the input features and applies

$$\mathcal{T}_0(H) = H, \qquad \mathcal{T}_1(H) = \tfrac{1}{2}(S_{\mathrm{sym}}H + S_{\mathrm{sym}}^2 H), \qquad \mathcal{T}_2(H) = H - S_{\mathrm{sym}}H. \tag{107}$$

The field H is the encoded representation, and $\mathcal{T}_0, \mathcal{T}_1, \mathcal{T}_2$ denote the identity, averaged one-hop and two-hop, and high-pass channels. Each channel has an affine map and ReLU. A two-layer gate takes the concatenated channel outputs and returns three softmax weights per node. Their weighted sum passes through dropout and the classifier.

GloGNN. GloGNN combines global interactions in class space with local propagation (Li et al., 2022). Starting from nodewise class logits, each normalization stage forms a class-by-class Gram matrix, solves the associated regularized class-space system, and evaluates global products through this low-dimensional representation. Learned node-dependent coefficients combine successive powers of S_{rw}. We include an initial-logit residual at every stage, and these operations use double precision.

CGNN. CGNN computes an approximate commute-based proximity before training its message-passing layers (Zhuo et al., 2025). Its preprocessing orders nodes by cosine similarity to the mean feature vector and adds a bidirectional chain for the proximity calculation. A stationary distribution and a rank-truncated singular-value decomposition produce the commute approximation. Positive edge proximities are obtained by exponentiating the negative of the clipped commute estimates. Propagation uses the supplied graph edges, with distinct learned incoming and outgoing feature maps. Each layer averages the current node state and the two degree-averaged message fields, then applies ReLU and dropout. A two-layer perceptron produces the class logits.

L2DGCN. L2DGCN combines graph enhancement, restart propagation, and label selection (Zhang et al., 2025). Graph enhancement retains edges incident to the selected higher-degree nodes and constructs cosine nearest-neighbor edges among the remaining nodes. A label-propagation teacher starts from one-hot training labels on the enhanced graph. The prediction model propagates features with restart before a two-layer perceptron, and a second restart propagation supplies the confidence used for pseudo-label selection. Confident non-training nodes receive teacher-generated labels. The auxiliary objective includes pseudo-label cross-entropy and a contrastive term formed from normalized hidden representations and class prototypes. The teacher and graph updates use detached representations. Supervised and auxiliary losses train the feature transformations.

Other architecture comparisons. GraphSAGE aggregates neighborhood features (Hamilton et al., 2017), and GAT-sep separates ego and aggregated neighbor representations (Platonov et al., 2023). MagNet uses a Hermitian magnetic Laplacian (Zhang et al., 2021), and Dir-GNN separates incoming and outgoing neighborhoods (Rossi et al., 2024). DIGRAF learns a graph-adaptive diffeomorphic activation (Mantri et al., 2024). LINKX combines node features and adjacency information in the large-graph comparison (Lim et al., 2021).

Shared-backbone computation. The pointwise and graph-coupled variants use the same encoder, four-branch anchor, residual integration, layer normalization, dropout placement, and concatenated readout. At layer ℓ, let $\mathcal{B}^{(\ell)}$ denote the nonlinear block that maps the anchor and edge fields to the node output before residual integration. Its principal variants are

$$\begin{aligned} \mathcal{B}^{(\ell)}_{\mathrm{GELU}} &= \mathrm{GELU}(Z^{(\ell)}), \\ \mathcal{B}^{(\ell)}_{\mathrm{Unshifted},g} &= \widehat{P}^{T,Q_g}_{g,W^{(\ell)},0}(Z^{(\ell)}), \\ \mathcal{B}^{(\ell)}_{\mathrm{LEGO},g} &= \widehat{P}^{T,Q_g}_{g,W^{(\ell)},M^{(\ell)}}(Z^{(\ell)}). \end{aligned} \tag{108}$$

The potential choice is $g \in \{g_{\mathrm{TV}}, g_\theta\}$, $T = 12$ is the outer depth, and Q_g is zero for TV and eight for LP. Each graph-coupled model learns its own WeightNet parameters, and each LP model learns

its own potential parameters. The paired constructions share an architecture and initialization, and we optimize their parameter values independently.

Paired potential and target variants. Variant construction duplicates a common initialized graph-coupled network before applying the potential and target choices. For the TV and quadratic variants, we replace the LP module with the corresponding parameter-free potential. For zero-target variants, we remove OffsetNet and use a zero edge field. Consequently, the encoder, anchor, WeightNet, residual path, and classifier share their initial values, and each variant is then optimized independently.

Target placement and parameterization. Anchor injection retains the full target module and uses its weighted divergence to modify the anchor. Integrable-train obtains the target from node potentials. Projected-OffsetNet retains the full clipped target module and applies the orthogonal projection before the graph block and target penalty. Their layer outputs are

$$\begin{aligned} \mathcal{B}_{\text{inj}} &= \widehat{P}^{12,8}_{\theta,W,0}(Z + \alpha K^\top D_W M), \\ \mathcal{B}_{\text{node}} &= \widehat{P}^{12,8}_{\theta,W,KB}(Z), \\ \mathcal{B}_{\text{proj}} &= \widehat{P}^{12,8}_{\theta,W,\Pi_{\mathcal{I}}M}(Z). \end{aligned} \tag{109}$$

The layer index is suppressed, $D_W = \operatorname{diag}(W)$ contains the edge weights, and B is the learned node-potential field. The following appendices detail the width, anchor-injection, node-potential, and projected-target controls.

C.3 Width and Anchor-Injection Controls

GELU-wide increases the pointwise width of the shared backbone. Anchor injection keeps OffsetNet and LEGO-LP's total parameter count, supplying $(Z + \alpha K^\top \operatorname{diag}(W)M, W, 0)$ to the LP block. Table 11 reports both controls after training and validation selection.

Table 11: Width and target-placement controls over ten training seeds. Scores are mean ± standard deviation. Minesweeper uses ROC-AUC. Anchor injection and LEGO-LP have identical OffsetNet architectures and total parameter counts. The final columns show differences between the corresponding LEGO-LP and control means.

Dataset	GELU-wide	Anchor injection	LEGO-LP − GELU-wide	LEGO-LP − anchor injection
RomanEmpire	75.4 ± 1.3	81.2 ± 1.6	$+7.8$	$+2.0$
Minesweeper	90.1 ± 1.0	91.7 ± 1.5	$+2.0$	$+0.4$
AmazonRatings	46.0 ± 0.9	52.0 ± 1.0	$+6.5$	$+0.5$
Chameleon	58.5 ± 1.4	60.0 ± 1.0	$+1.7$	$+0.2$
Squirrel	35.8 ± 0.7	38.0 ± 0.8	$+3.4$	$+1.2$

Increasing width improves the pointwise model; anchor injection recovers more of the gain. LEGO-LP achieves higher mean scores than both controls. Anchor injection approaches its performance on several graphs.

Quadratic anchor identity and the placement control. For the quadratic potential, let D_W be the diagonal edge-weight matrix, Z_M the modified anchor, and $\|M\|_W$ the weighted target norm:

$$D_W = \operatorname{diag}(W), \quad Z_M = Z + \alpha K^\top D_W M, \quad \|M\|_W^2 = \sum_{e \in \vec{E}} w_e \|\mu_e\|_2^2. \tag{110}$$

With $g_{\text{quad}}(t) = t^2/2$, expanding the edge term and completing the anchor square yields

$$\frac{1}{2\alpha}\|U - Z\|_F^2 + \frac{1}{2}\|KU - M\|_W^2 = \frac{1}{2\alpha}\|U - Z_M\|_F^2 + \frac{1}{2}\|KU\|_W^2 + c_M, \tag{111}$$

where $c_M = \frac{1}{2}\|M\|_W^2 + (\|Z\|_F^2 - \|Z_M\|_F^2)/(2\alpha)$ is constant in U. The learned-target quadratic minimizer equals the zero-target minimizer at Z_M for any nonnegative weights, including zero

weights. The weighted quadratic specialization in Appendix A gives the corresponding node-coordinate translation on the active support.

Anchor injection applies this transformation before the zero-target LP recurrence. A target perturbation $N \in \mathbb{R}^{m\times d}$ preserves Z_M whenever $K^\top D_W N = 0$, which also leaves the exact quadratic map unchanged. For nonuniform weights, this weighted null space differs from the Euclidean cycle space $\ker K^\top$ used in the projection diagnostics.

The target-centered quadratic call starts at Z, and the anchor injection starts at Z_M. They share an exact minimizer and follow trajectories from their respective initializations at finite depth. LP additionally transforms target-centered residuals through nonlinear edge responses before aggregation.

C.4 Paired WebKB Comparisons

We compare GELU, Unshifted-LP, and LEGO-LP on WebKB. Figure 3 displays their mean accuracies and standard deviations, and Table 12 reports the consecutive improvements from graph coupling and target learning.

Table 12: WebKB mean accuracy contrasts over $N = 10$ paired runs per dataset. Splits, common-module initialization seeds, and early stopping are shared. The score ordering is displayed in Figure 3.

Dataset	N	Unshifted-LP − GELU	LEGO-LP − Unshifted-LP
Texas	10	+1.0	+1.3
Wisconsin	10	+0.6	+1.2
Cornell	10	+0.6	+2.1

C.5 Node-Potential Training and Integrable Targets

Integrable-train learns node potentials and restricts every target to their endpoint difference:

$$B^{(\ell)} = \mathrm{PotentialNet}(H^{(\ell)}), \quad M^{(\ell)} = KB^{(\ell)}. \tag{112}$$

The nodewise network $\mathrm{PotentialNet}$ maps incoming rows of width d_ℓ to potential rows of width $d_{\ell+1}$. The incidence operator acts on its output, giving $\mu_{ij}^{(\ell)} = B_i^{(\ell)} - B_j^{(\ell)}$ and placing the target in $\mathrm{Im}(K)$. The graph block receives this difference field directly, and the target penalty is evaluated on $KB^{(\ell)}$. Suppressing the layer index, a target gradient G_M induces

$$\nabla_B \ell = K^\top G_M, \qquad \nabla_B \frac{\lambda_\mu \|KB\|_F^2}{md} = \frac{2\lambda_\mu}{md} K^\top KB. \tag{113}$$

The first expression applies to a scalar loss ℓ with target gradient G_M, and $d = d_{\ell+1}$ is the potential width. Adding a componentwise constant to B preserves both its target and target penalty. The backbone, weight network, potential, and training settings follow LEGO-LP. Appendix C.6 describes the full OffsetNet integrable parameterization.

Table 13 compares node-potential training with integrable-only evaluation at selected LEGO-LP checkpoints.

Table 13: LP integrable-target controls on the five main datasets. Integrable-train learns node potentials, while integrable-only projects the targets of the trained LEGO-LP checkpoint. Entries are accuracy except for Minesweeper (ROC-AUC).

Control	Roman.	Mine.	Amazon.	Cham.	Squirrel
Integrable-train	82.1	92.0	51.6	59.9	38.6
Integrable-only	81.8	91.5	51.3	59.8	38.4

C.6 PROJECTED-OFFSETNET TRAINING PROTOCOL

Projected-OffsetNet retains OffsetNet's architecture and parameter count and projects its targets onto node-potential differences. If $M^{(\ell)}_{\mathrm{clip}}$ is the antisymmetric field after elementwise clipping, each training and evaluation pass uses

$$M^{(\ell)}_{\mathrm{PI}} = \Pi_{\mathcal{I}} M^{(\ell)}_{\mathrm{clip}}, \quad \Pi_{\mathcal{I}} = K(K^\top K)^\dagger K^\top. \tag{114}$$

Projection acts independently on each feature coordinate using the zero-mean gauge in Appendix B.8. The resulting $M^{(\ell)}_{\mathrm{PI}}$ enters both the graph block and the target penalty.

Self-adjointness of $\Pi_{\mathcal{I}}$ maps target gradients through the same projection before differentiation through clipping and OffsetNet. Training uses the full target-network parameterization with its output restricted to the incidence subspace.

Table 14 gives the Projected-OffsetNet training results and the corresponding LEGO-LP scores from Table 2. The control follows the training protocol in Appendix C.1 and has the LEGO-LP parameter totals in Table 9.

Projection geometry and effective capacity. Orthogonal projection contracts the Frobenius norm and can increase individual coordinates. The coordinatewise clipping bound applies to $M^{(\ell)}_{\mathrm{clip}}$; projection makes $M^{(\ell)}_{\mathrm{PI}}$ integrable. The squared norm of the regularized target decreases by exactly the squared cycle norm:

$$\|M^{(\ell)}_{\mathrm{clip}}\|_F^2 = \|M^{(\ell)}_{\mathrm{PI}}\|_F^2 + \|\Pi_\perp M^{(\ell)}_{\mathrm{clip}}\|_F^2. \tag{115}$$

The control keeps the parameter count and restricts both target directions and their gradients.

Table 14: Mean scores for Projected-OffsetNet and its paired LEGO-LP training comparison. Entries use accuracy except for Minesweeper (ROC-AUC).

Model	Roman.	Mine.	Amazon.	Cham.	Squirrel
Projected-OffsetNet	83.0	91.7	51.5	60.0	38.7
LEGO-LP	**83.2**	**92.1**	**52.5**	**60.2**	**39.2**

C.7 FROZEN EDGE-ASSIGNMENT PERMUTATIONS

At frozen LEGO-LP checkpoints, we permute the target rows using the representative system $e = (i, j)$ with $i < j$. For a permutation ρ of these edge rows, the target in each controlled layer is

$$(M^\rho)_e = M_{\rho(e)}. \tag{116}$$

The permutation index $\rho(e)$ identifies the target row assigned to edge e. This operation changes endpoint correspondence and cycle geometry. The target-row multiset and marginal magnitudes remain unchanged in the selected orientation.

Table 15 gives the twenty evaluation pairs per dataset summarized in Figure 4. Checkpoint-level means and individual win/tie/loss counts follow Appendix C.1.

Relation to cycle-residual removal. At a controlled layer, let P_ρ be the row-permutation matrix implementing ρ. The perturbations applied by projection and reassignment are

$$\Delta M_{\mathrm{proj}} = -M_\perp, \qquad \Delta M_{\mathrm{perm}} = (P_\rho - I)M. \tag{117}$$

The field ΔM_{proj} removes the learned cycle component; ΔM_{perm} reassigns the targets among endpoint pairs.

Coordinate representation. Let $S_\pm$ be a diagonal matrix with entries in $\{-1, 1\}$, representing edge reversals. The same target intervention in the reoriented coordinates is

$$\widetilde{K} = S_\pm K, \quad \widetilde{M} = S_\pm M, \quad \widetilde{P}_\rho = S_\pm P_\rho S_\pm, \quad \widetilde{P}_\rho \widetilde{M} = S_\pm P_\rho M. \tag{118}$$

Tildes denote quantities in the reoriented coordinates. Conjugation gives a signed permutation that represents the same intervention and preserves its node output. We define the perturbation distribution by sampling ordinary row permutations in the $i < j$ representative system.

Table 15: Frozen edge-reassignment results over twenty evaluation pairs per dataset. Mean gaps are original minus permuted performance, with repeated permutations averaged within checkpoints before checkpoint-level aggregation. Gaps use accuracy points except for Minesweeper, which uses ROC-AUC points. Win/tie/loss counts summarize individual evaluation pairs. The final row averages dataset-level gaps and sums the counts.

Dataset	Evaluation pairs	Mean metric gap	Win/Tie/Loss
RomanEmpire	20	$+2.51$	14/4/2
Minesweeper	20	$+1.86$	18/0/2
AmazonRatings	20	$+3.45$	15/3/2
Chameleon	20	$+0.90$	17/1/2
Squirrel	20	$+1.22$	16/3/1
Mean / total	100	$+1.99$	80/11/9

Table 16: Macro-F1 over ten seeds at checkpoints selected by the original validation metric. The class-balanced metric is evaluated after checkpoint selection.

Dataset	GELU	Unshifted-LP	**LEGO-LP**
RomanEmpire	69.09	73.28	**78.73**
Minesweeper	79.61	80.43	**81.54**
AmazonRatings	37.52	41.64	**43.00**
Chameleon	51.25	54.70	**56.75**
Squirrel	31.64	35.64	**36.02**

C.8 Macro-F1 at Selected Checkpoints

We evaluate macro-F1, the unweighted mean of classwise F1 scores, at the original validation-selected checkpoints. We select the Minesweeper model using ROC-AUC. LEGO-LP has higher macro-F1 than both GELU and Unshifted-LP on every graph in Table 16.

C.9 Graph Structure, Target Geometry, and Correlations

We examine graph structure and target geometry in five-seed diagnostic reruns. These runs complement the ten-run prediction results in Table 1(a) and target measurements in Table 21. At each validation-selected checkpoint, we average geometry over layers and then over checkpoints.

Graph-level quantities and retrained gains. Edge heterophily is the fraction of incidence edges whose endpoints have different labels. Cycle-rank density is $\rho_{\mathrm{cyc}} = (m - n + c)/m$, using the preprocessed incidence-edge count m, node count n, and component count c. Both quantities are graph-dependent and are shared across block variants. Equation 6 assigns a positive weight to every incidence edge, so learned weight magnitudes preserve this support. For a dataset, the retrained shift gain is

$$\Delta_{\mathrm{shift}} = \overline{A}_{\text{LEGO-LP}} - \overline{A}_{\text{Unshifted-LP}}. \tag{119}$$

The quantity $\overline{A}$ is the mean primary evaluation metric over the five diagnostic runs. Table 17 pairs this gain with graph structure. We observe positive gains across all five graphs, consistent with the main experiment.

Checkpoint geometry and frozen removal. For each selected LEGO-LP model, we compute $r_\perp$ and δ_{cyc} from its full clipped targets and evaluate the same model after layerwise integrable projection. The frozen removal gap is

$$\Delta_{\mathrm{rem}} = \frac{1}{5}\sum_{s=1}^{5}\big(A^{(s)}_{\mathrm{full}} - A^{(s)}_{\mathrm{int}}\big). \tag{120}$$

The index s identifies a diagnostic run, and $A^{(s)}_{\mathrm{full}}$ and $A^{(s)}_{\mathrm{int}}$ are its original and integrable-only scores. Target replacement follows the layerwise evaluation order in Appendix C.1, with model parameters held constant. Table 18 pairs the target statistics with this behavioral response.

Table 17: Graph structure and retrained shift gains from five-seed diagnostic reruns. The edge column gives preprocessed incidence counts. Gains use accuracy points except for Minesweeper, which uses ROC-AUC points.

Dataset	Edges m	Edge heterophily	ρ_{cyc}	Δ_{shift}
RomanEmpire	32,927	0.953	0.312	$+2.2$
Minesweeper	39,402	0.317	0.746	$+0.7$
AmazonRatings	93,050	0.620	0.737	$+1.4$
Chameleon	33,824	0.770	0.933	$+1.2$
Squirrel	211,872	0.778	0.975	$+1.2$

Table 18: Checkpoint-level target geometry and frozen cycle-removal gaps from the same five-seed diagnostic reruns. Geometry is measured before intervention. The removal gap uses each dataset's primary-metric points.

Dataset	$r_\perp$	δ_{cyc}	Δ_{rem}
RomanEmpire	0.13	0.069	$+1.4$
Minesweeper	0.10	0.010	$+0.5$
AmazonRatings	0.08	0.286	$+1.2$
Chameleon	0.09	0.009	$+0.5$
Squirrel	0.14	0.023	$+1.0$

Cross-dataset associations. Table 19 gives Pearson correlations calculated from the displayed dataset means in Tables 17 and 18. Each coefficient uses five graph-level observations, with accuracy for four graphs and ROC-AUC for Minesweeper. The first two rows compare graph structure with retrained shift gains, and the next two compare learned target geometry with frozen removal gaps. The final two rows relate the same target diagnostics to retrained gains.

Table 19: Pearson correlations across the five displayed dataset means. The final two rows combine checkpoint diagnostics with retrained model contrasts.

Quantity	Performance contrast	Pearson correlation
Edge heterophily	Retrained shift gain	$+0.84$
Cycle-rank density ρ_{cyc}	Retrained shift gain	-0.75
Projection ratio $r_\perp$	Frozen removal gap	$+0.38$
Cycle circulation δ_{cyc}	Frozen removal gap	$+0.56$
Projection ratio $r_\perp$	Retrained shift gain	$+0.36$
Cycle circulation δ_{cyc}	Retrained shift gain	$+0.26$

The associations are positive for edge heterophily and negative for cycle-rank density when paired with retrained target gains. Projection ratio and cycle circulation are positively associated with frozen removal gaps. These coefficients summarize the five graph-level means. The direct target and removal comparisons appear in Tables 1 and 2.

C.10 Frozen-Checkpoint Computation and Depth Sensitivity

We compare outputs $\widehat{U}$ and $\widehat{U}_{\mathrm{deep}}$ at the same layer tensors Z, W, M and step sizes. The subscript deep denotes the $(32, 16)$ projected Newton computation. Figure 6 uses

$$\Delta_U = \frac{\|\widehat{U} - \widehat{U}_{\mathrm{deep}}\|_F}{\|\widehat{U}_{\mathrm{deep}}\|_F + \varepsilon_{\mathrm{diag}}}. \tag{121}$$

The discrepancy Δ_U measures change relative to the full representation, including the anchor. Define the graph corrections $C = \widehat{U} - Z$ and $C_{\mathrm{deep}} = \widehat{U}_{\mathrm{deep}} - Z$. For $C_{\mathrm{deep}} \neq 0$, the discrepancy factors as

$$\Delta_U = \frac{\|C - C_{\mathrm{deep}}\|_F}{\|C_{\mathrm{deep}}\|_F} \frac{\|C_{\mathrm{deep}}\|_F}{\|Z + C_{\mathrm{deep}}\|_F + \varepsilon_{\mathrm{diag}}}. \tag{122}$$

The first factor is the relative change in the graph correction, and the second is the correction's size relative to the full output. An anchor-dominated representation makes the second factor small, so Δ_U describes full-output sensitivity at the two finite depths.

Prediction sensitivity at frozen checkpoints. For each validation-selected LP checkpoint, we evaluate the complete network at depths $(12, 8)$ and $(32, 16)$, preserving the trained parameters, projected Newton rule, and step sizes. Let A_{default} and A_{deep} denote the corresponding network-level prediction metrics. Table 20 reports $\Delta_{\mathrm{metric}} = A_{\mathrm{default}} - A_{\mathrm{deep}}$, averaged over seeds. Training and checkpoint selection use $(12, 8)$, and the deeper evaluation uses the selected parameters directly.

Table 20: Frozen-checkpoint depth comparison. Entries give the mean prediction metric at $(12, 8)$ minus that at $(32, 16)$. Parameters and step sizes are unchanged, and both depths use projected Newton. Displayed differences are rounded to one decimal place.

Model	Roman.	Mine.	Amazon.	Cham.	Squirrel	Cornell	Texas	Wisconsin
Unshifted-LP	0.0	0.0	0.0	0.0	-0.1	0.0	0.0	0.0
LEGO-LP	-0.1	-0.1	0.0	0.0	-0.1	0.0	0.0	0.0

At the displayed precision, the prediction metrics change by at most 0.1 points when the frozen computation is increased from $(12, 8)$ to $(32, 16)$. This comparison characterizes empirical sensitivity to additional finite updates at the selected checkpoints. Radial optimality is assessed separately through the residual Γ_e in Appendix B.2, where finite projected-Newton iterates can exhibit periodic behavior. Figure 6 extends the finite-depth comparison to layer outputs and objective values.

C.11 Projection Controls and Paired Uncertainty

We quantify the full learned targets through the projection ratio $r_\perp$ and normalized cycle circulation δ_{cyc}:

$$r_\perp = \frac{\|M_\perp\|_F}{\|M\|_F + \varepsilon_{\mathrm{diag}}}, \qquad \delta_{\mathrm{cyc}} = \frac{1}{|\mathcal{B}_{\mathrm{cyc}}|} \sum_{C \in \mathcal{B}_{\mathrm{cyc}}} \frac{\|\sum_{e \in C} \sigma_e(C)\mu_e\|_2}{\sum_{e \in C} \|\mu_e\|_2 + \varepsilon_{\mathrm{diag}}}. \tag{123}$$

The fundamental cycle basis is $\mathcal{B}_{\mathrm{cyc}}$, $\sigma_e(C) \in \{-1, 1\}$ is the traversal sign, and $\varepsilon_{\mathrm{diag}} > 0$ stabilizes the denominators. We set $\delta_{\mathrm{cyc}} = 0$ when $\mathcal{B}_{\mathrm{cyc}}$ is empty, and average the statistic over layers and then checkpoints.

Table 21 reports these measurements for the ten-checkpoint main study. Most target energy is integrable. Appendix A.11 decomposes the cycle field into dictionary and clipping contributions.

Table 21: Target geometry over the ten validation-selected LEGO-LP checkpoints used in the main study. Each statistic is averaged over layers and then checkpoints.

Statistic	Roman.	Mine.	Amazon.	Cham.	Squirrel
$r_\perp$	0.13	0.10	0.08	0.09	0.14
δ_{cyc}	0.07	0.01	0.28	0.01	0.03

Layerwise norm-controlled target perturbations. Index the cycle-removal, random-direction, and integrable-direction controls by $j \in \{\mathrm{cyc}, \mathrm{rnd}, \mathrm{int}\}$. At layer ℓ, let $H_j^{(\ell)}$ be the hidden state produced by control j and $M_j^{(\ell)} = \mathrm{OffsetNet}(H_j^{(\ell)}, \vec{E})$ its current target. The additive intervention $\Delta M_j^{(\ell)}$ obeys

$$\begin{aligned} \Delta M_{\mathrm{cyc}}^{(\ell)} &= -\Pi_\perp M_{\mathrm{cyc}}^{(\ell)}, \\ \|\Delta M_j^{(\ell)}\|_F &= \|\Pi_\perp M_j^{(\ell)}\|_F \quad (j \in \{\mathrm{rnd}, \mathrm{int}\}), \\ \Delta M_{\mathrm{int}}^{(\ell)} &\in \mathrm{Im}(K). \end{aligned} \tag{124}$$

Each layer receives $M_j^{(\ell)} + \Delta M_j^{(\ell)}$, with anchor and weights computed from $H_j^{(\ell)}$. First-layer norms agree because the encoded input is shared. Later layers normalize using the cycle norm of their own

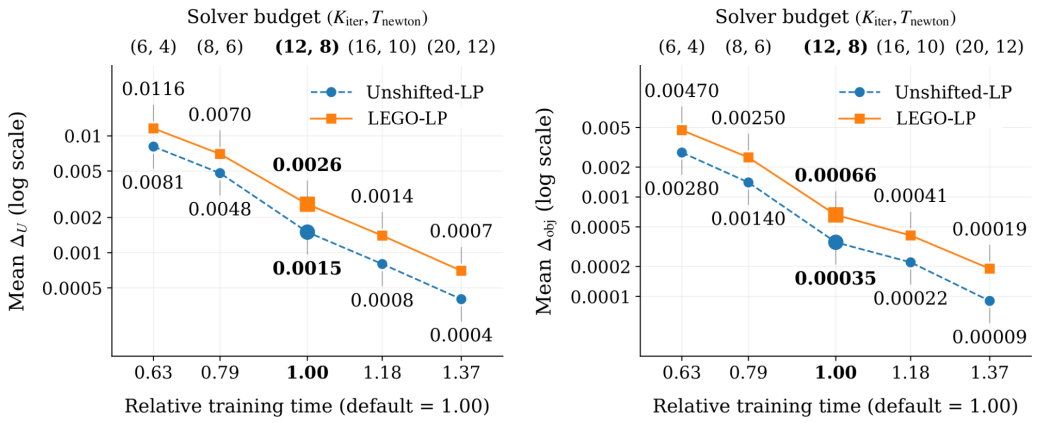


Figure 6: Sensitivity to finite LP depth. **Left:** full-output-normalized discrepancy Δ_U. **Right:** normalized signed objective difference Δ_{obj}. We train each depth independently and compare it at its selected checkpoint with a $(32, 16)$ projected Newton computation using the same layer tensors and step sizes. Results aggregate eight datasets, and training time is relative to the selected depth.

current target. Removal subtracts the learned cycle field; the other controls add directions at the corresponding norm.

For an isotropic edge-feature direction D, the expected fraction of energy in the cycle subspace is $\mathbb{E}[\|\Pi_\perp D\|_F^2/\|D\|_F^2] = (m - n + c)/m$ (Appendix B.9). The removal intervention uses the specific direction opposite to the model's learned cycle field.

The geometry measurements use the full positive-weight incidence graph and its depth-first fundamental cycle basis.

Frozen integrable-only evaluation replaces the target with $KB^\dagger$ at each layer. Table 2 reports checkpoint-level paired differences between the original and projected scores.

C.12 Sensitivity to Finite Computational Depth

We train alternative LP depths with the same backbone, protocol, and selection rule. At each selected checkpoint, we compare the trained depth with a $(32, 16)$ projected Newton computation. Figure 6 plots Δ_U from Appendix C.10 and the normalized objective difference

$$\Delta_{\text{obj}} = \frac{\Phi_Z(\widehat{U}, M) - \Phi_Z(\widehat{U}_{\text{deep}}, M)}{\max\{1, \Phi_Z(\widehat{U}_{\text{deep}}, M)\}}. \tag{125}$$

The sign of Δ_{obj} identifies the lower of the two finite objective values. The positive values in Figure 6 decrease across the evaluated computational budgets, alongside the output discrepancies. These quantities characterize the sensitivity of the prescribed finite computation to its update budget. Appendix A.9 gives output bounds based on primal-dual gaps and stationarity, and Appendix B.3 reports end-to-end cost at the selected depth.

C.13 Target-Parameter Sensitivity

Figure 7 varies K_μ, $\mu_{\max}$, and λ_μ individually around the reference configuration in Appendix B.1. We observe higher mean scores with target learning across the displayed settings. Intermediate dictionary sizes and clipping bounds perform best in their respective sweeps, and stronger target regularization lowers the average cycle ratio.

For a fixed raw target, increasing $\mu_{\max}$ lowers the bound on the clipping-induced cycle component (Appendix A.11). Retraining at each clipping bound also changes the raw target and dictionary contribution, so the projection ratio can vary nonmonotonically.

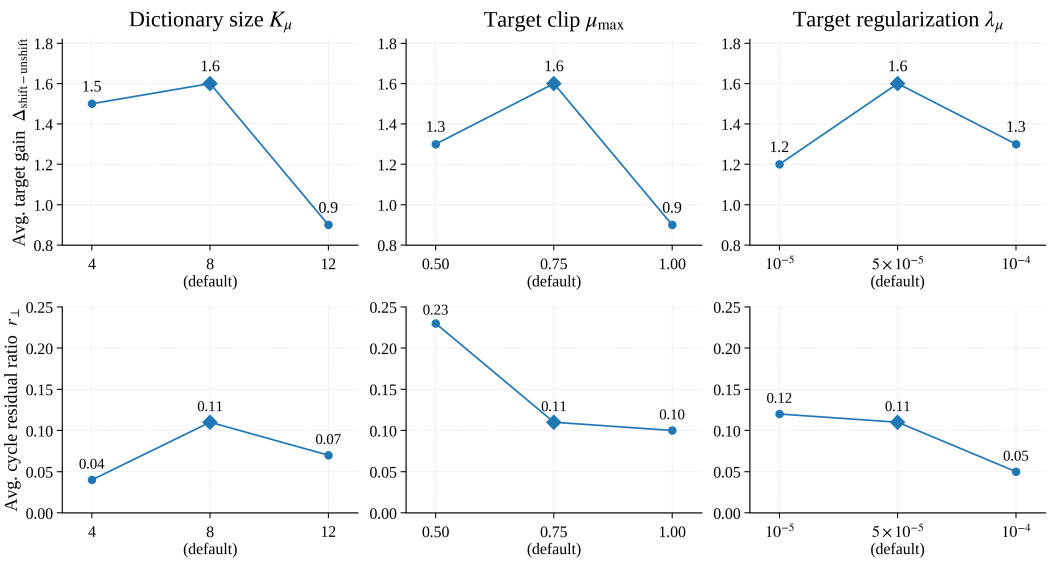


Figure 7: Sensitivity around the selected LEGO-LP target configuration. Each column varies dictionary size K_μ, clipping bound $\mu_{\max}$, or target regularization λ_μ. The top row averages LEGO-LP minus Unshifted-LP over the five main datasets. The bottom row averages the cycle-residual ratio over the projection-diagnostic datasets. We train each setting independently with the same backbone, solver depth, and validation-selection rule. The marked reference values are those in Appendix B.1.

C.14 Target Learning across Fixed and Learned Potentials

Potential families. We compare TV g_{TV}, quadratic g_{quad}, and learned LP g_{learn} to distinguish target learning from potential flexibility:

$$g_{\mathrm{TV}}(t) = t, \qquad g_{\mathrm{quad}}(t) = \frac{1}{2}t^2, \qquad g_{\mathrm{learn}}(t) = g_\theta(t). \tag{126}$$

The LP potential g_θ is defined in Eq. 26. We train an Unshifted and a learned-target model for each potential. Learned-target models share OffsetNet, antisymmetrization, and clipping, and all six models learn WeightNet and its scale s_w. The TV and quadratic potentials are parameter-free.

Matched training. The TV and LP comparisons use the runs from Table 1. The quadratic pair uses the same split identifiers, paired common-module initializations, and validation rule. All six variants have width 128, two layers, $\alpha = 1$, and $K_{\mathrm{iter}} = 12$. Each LP model learns its own potential with $T_{\mathrm{newton}} = 8$.

Solver implementation. All variants use the same outer primal-dual update and step-size rule. LP performs $T_{\mathrm{newton}} = 8$ projected Newton steps per radial response. The fixed potentials use the closed forms

$$s^\star_{\mathrm{TV}} = \max\{r - \lambda_e, 0\}, \qquad s^\star_{\mathrm{quad}} = \frac{r}{1+\lambda_e}, \qquad \lambda_e = \frac{w_e}{\sigma}, \tag{127}$$

where r and λ_e are the input radius and scaled edge weight from Eq. 12. These radii determine the edge outputs. TV's linear radial penalty produces nonlinear shrinkage through its threshold.

Results and contrasts. For paired run s and potential g, let $A^{(s)}_{g,\mathrm{shift}}$ and $A^{(s)}_{g,0}$ denote the primary metrics of the learned-target and Unshifted models. Their target gain is

$$\Delta^{(s)}_g = A^{(s)}_{g,\mathrm{shift}} - A^{(s)}_{g,0}. \tag{128}$$

Table 22 also reports quadratic scores and compares LEGO-LP with each learned-target fixed-potential model. Accuracy and ROC-AUC use the 0 to 100 scale. We compute 95% intervals with 10,000 paired replicates, resampling split identifiers while retaining their paired seeds.

Table 22: Potential-family comparisons over ten paired, validation-selected split and seed instances. The TV and LP contrasts use the same runs as Table 1. Independently trained quadratic models extend the comparison under the same selection protocol. Target gains compare learned targets with zero targets, and brackets give paired 95% intervals.

Score / contrast	Roman.	Mine.	Amazon.	Cham.	Squirrel
Quadratic: zero targets	79.7 ± 1.2	90.1 ± 1.0	49.9 ± 0.9	58.5 ± 1.6	36.8 ± 0.8
Quadratic: learned targets	81.6 ± 1.1	90.9 ± 0.8	51.2 ± 0.9	59.1 ± 1.3	37.9 ± 0.5
Δ_{TV} [95% CI]	$+2.4$ $[1.7, 3.1]$	$+1.1$ $[0.5, 1.7]$	$+1.6$ $[1.1, 2.2]$	$+1.7$ $[1.2, 2.2]$	$+1.4$ $[1.2, 1.7]$
Δ_{quad} [95% CI]	$+1.9$ $[1.2, 2.6]$	$+0.8$ $[0.2, 1.4]$	$+1.3$ $[0.7, 1.8]$	$+0.6$ $[-0.2, 1.4]$	$+1.1$ $[0.8, 1.4]$
Δ_{LP} [95% CI]	$+2.7$ $[2.0, 3.4]$	$+1.2$ $[0.6, 1.8]$	$+1.8$ $[1.2, 2.4]$	$+0.9$ $[0.1, 1.7]$	$+1.6$ $[1.3, 1.9]$
LEGO-LP − LEGO-TV	$+0.5$ $[-0.2, 1.2]$	$+0.3$ $[-0.2, 0.9]$	$+0.4$ $[-0.2, 0.9]$	$+0.3$ $[-0.5, 1.1]$	$+0.4$ $[0.1, 0.6]$
LEGO-LP − learned-target quadratic	$+1.6$ $[0.9, 2.3]$	$+1.2$ $[0.6, 1.7]$	$+1.3$ $[0.8, 1.9]$	$+1.1$ $[0.3, 1.9]$	$+1.3$ $[1.0, 1.5]$

Target learning improves the mean score for all three potentials. The TV and LP target-gain intervals are positive on every graph, and the quadratic intervals are positive on four graphs and span zero on Chameleon. LP also achieves higher mean scores than the learned-target fixed-potential models. Its paired intervals against TV are positive on Squirrel, and those against the quadratic model are positive on every graph.

C.15 Tree and Cycle Comparisons

Each replicate has a node set V_s with $n_s = 1{,}000$ nodes, 100-dimensional node features, and five classes. We sample node labels independently and uniformly from $\{1, \dots, 5\}$, independently of graph construction. We then draw each node's feature vector from the Gaussian distribution associated with its class, producing class-conditional Gaussian features.

For each replicate, the base edge set $E_{s,0}$ is sampled uniformly from labeled trees by drawing a length-$(n_s - 2)$ Prüfer sequence with independent uniform entries in V_s and decoding the sequence (Caminiti et al., 2007). We sample twenty distinct unordered non-self pairs uniformly without replacement from the complement of $E_{s,0}$. Let $F_{s,5}$ contain the first five sampled pairs and $F_{s,20}$ contain all twenty. The feature matrix X_s, label vector y_s, and train/validation/test masks are shared across the three variants. For $k \in \{5, 20\}$, the augmented edge set is

$$E_{s,k} = E_{s,0} \cup F_{s,k}. \tag{129}$$

The incidence matrix $K_{s,k}$ represents this graph with one row per edge. Since edge addition preserves connectivity,

$$|E_{s,k}| = n_s - 1 + k, \qquad \operatorname{rank}(K_{s,k}) = n_s - 1, \qquad \rho_{\mathrm{cyc}}(G_{s,k}) = \frac{k}{n_s - 1 + k}. \tag{130}$$

At layer width d, each added edge contributes one independent cycle direction per feature coordinate, so the cycle field has dimension kd. Under the independent uniform label distribution, each pair of distinct nodes shares a class with probability $1/5$. Thus, the tree and both augmented variants have expected edge homophily 0.2, where edge homophily is the fraction of unordered edges whose endpoints share a class.

We train LEGO-LP on each graph with the same optimizer settings and validation-selection rule. At each selected checkpoint, we measure full-target geometry and compare original and integrable-only prediction using the intervention order in Appendix C.1. The removal gap is their paired accuracy difference, computed within each replicate and then summarized across the thirty replicates. Table 23 reports the means and standard deviations in Figure 5.

On a tree, the edge projector is the identity and every target has a node-potential representation. For any such representation $M = KB$, the shared deterministic solver obeys

$$\widehat{P}^{T,Q}_{\theta,W,KB}(Z) = B + \widehat{P}^{T,Q}_{\theta,W,0}(Z - B). \tag{131}$$

Table 23: Synthetic tree-to-cycle study over thirty matched replicates. Geometry uses the full targets. Integrable-only and LEGO-LP evaluate one trained checkpoint, and the final column summarizes their paired accuracy difference. Entries are means ± standard deviations, with accuracy on the 0 to 100 scale.

Graph	$r_\perp$	δ_{cyc}	Integrable-only	LEGO-LP	Removal gap
Tree	0.00 ± 0.00	0.00 ± 0.00	78.6 ± 0.8	78.6 ± 0.8	0.0
Tree + 5 edges	0.13 ± 0.02	0.09 ± 0.02	78.2 ± 0.7	79.0 ± 0.7	0.8 ± 0.3
Tree + 20 edges	0.29 ± 0.03	0.21 ± 0.03	77.9 ± 0.8	79.5 ± 0.7	1.6 ± 0.4

The equal tree accuracies follow from projection preserving every target. The translation identity, proved in Appendix A.2, describes the equivalent evaluation in node coordinates. Learned integrable targets affect prediction through the translated anchor and restored node field, as examined by the training comparisons in Tables 1 and 13.

The augmented graphs have larger projection ratios and cycle circulation, and full-target evaluation improves over integrable-only evaluation at their trained checkpoints. With labels generated independently of graph construction, we examine learned target use under controlled edge addition. Each paired removal contrast evaluates the cycle component on one graph, sharing neighborhoods and degree-based step sizes between the original and intervened evaluations. Appendix B.9 relates cycle dimension to target-energy allocation.

C.16 Node Classification on Heterophilic Benchmarks

Table 24 compares the shared-backbone models with the architectures defined in Appendix C.2. We train and evaluate every method using the nine-graph protocol in Appendix C.1. H_2GCN-RW uses row-normalized neighborhoods, and Three-channel uses the specified gated identity, low-pass, and high-pass branches.

LEGO-LP has the highest mean scores on AmazonRatings, Squirrel, Actor, Cornell, Texas, and Wisconsin. CGNN leads on RomanEmpire, and the Three-channel control leads on Minesweeper and Chameleon. The shared-backbone comparisons show consecutive improvements from graph coupling and target learning.

C.17 Node Prediction on Deduplicated Chameleon and Squirrel

We train GELU, Unshifted-TV, and LEGO-TV on deduplicated Chameleon and Squirrel. Both TV variants use the same backbone and WeightNet architecture, with twelve outer updates and closed-form TV responses. LEGO-TV adds OffsetNet and target regularization. We also train GCN, GAT, GPRGNN, and FAGCN as architecture baselines. The GCN and GAT implementations include post-aggregation multilayer perceptrons, skip connections, and layer normalization (Platonov et al., 2023). Table 25 reports test accuracy, and Table 1(b) shows the shared-backbone rows.

LEGO-TV has the highest mean accuracy in the full comparison on both deduplicated graphs. Among the architecture baselines GCN, GAT, GPRGNN, and FAGCN, FAGCN has the highest mean accuracy. Comparing LEGO-TV with Unshifted-TV supports the benefit of target learning after duplicate removal.

C.18 Analysis on Large Graphs

Table 26 extends our architecture comparison to the Penn94, arXiv-year, and snap-patents benchmarks (Lim et al., 2021).

We train every method on the same partitions and evaluate it at its selected validation checkpoint. LEGO-LP uses the architecture, optimizer family, and solver depth in Appendix B. On the largest graph, we distribute edge computations through edge chunking and additional A100 GPUs, keeping the layer objective unchanged.

Table 24: Architecture and implementation-variant comparisons over ten shared splits. Entries are mean ± standard deviation of accuracy, with ROC-AUC for Minesweeper. H_2GCN-RW and Three-channel use the operators specified in Appendix C.2. The final block uses the shared backbone; WebKB results appear in Figure 3 and Table 12.

(a) Five main graphs

Method	Roman.	Mine.	Amazon.	Chameleon	Squirrel
GCN	$66.5_{\pm 0.67}$	$86.2_{\pm 0.99}$	$44.1_{\pm 0.48}$	$55.1_{\pm 0.63}$	$31.0_{\pm 0.72}$
SAGE	$77.6_{\pm 1.02}$	$88.5_{\pm 0.62}$	$48.7_{\pm 0.41}$	$56.2_{\pm 1.07}$	$33.3_{\pm 0.85}$
GAT	$66.9_{\pm 0.41}$	$85.1_{\pm 1.12}$	$43.4_{\pm 0.49}$	$54.2_{\pm 0.85}$	$31.2_{\pm 0.93}$
H_2GCN-RW	$70.4_{\pm 0.58}$	$90.1_{\pm 1.27}$	$46.3_{\pm 0.64}$	$53.6_{\pm 0.88}$	$31.3_{\pm 0.71}$
GCNII	$72.8_{\pm 0.54}$	$89.7_{\pm 1.38}$	$46.5_{\pm 0.56}$	$53.8_{\pm 0.80}$	$30.9_{\pm 0.87}$
MagNet	$75.0_{\pm 0.67}$	$90.8_{\pm 1.43}$	$46.9_{\pm 0.70}$	$57.0_{\pm 1.31}$	$32.2_{\pm 1.18}$
GPRGNN	$73.9_{\pm 0.57}$	$90.2_{\pm 1.23}$	$47.2_{\pm 0.66}$	$55.6_{\pm 0.81}$	$30.8_{\pm 0.62}$
FAGCN	$71.0_{\pm 0.85}$	$88.7_{\pm 1.26}$	$46.2_{\pm 0.58}$	$54.9_{\pm 0.78}$	$31.1_{\pm 0.90}$
ACM-GCN	$75.2_{\pm 0.64}$	$91.0_{\pm 1.37}$	$47.5_{\pm 0.62}$	$56.4_{\pm 1.43}$	$31.9_{\pm 1.08}$
GloGNN	$74.4_{\pm 0.63}$	$90.5_{\pm 1.28}$	$46.3_{\pm 0.71}$	$54.1_{\pm 0.67}$	$31.2_{\pm 0.81}$
GAT-sep	$79.1_{\pm 0.39}$	$91.3_{\pm 0.47}$	$49.2_{\pm 0.80}$	$58.1_{\pm 1.14}$	$34.6_{\pm 0.83}$
DIGRAF	$78.0_{\pm 1.14}$	$90.7_{\pm 1.07}$	$49.3_{\pm 0.82}$	$59.1_{\pm 1.44}$	$37.1_{\pm 0.70}$
Dir-GNN	$82.3_{\pm 0.68}$	$91.3_{\pm 1.42}$	$48.3_{\pm 0.72}$	$59.7_{\pm 1.48}$	$35.0_{\pm 1.16}$
Three-channel	$78.8_{\pm 0.71}$	$92.5_{\pm 1.47}$	$50.2_{\pm 0.75}$	$60.4_{\pm 1.60}$	$36.2_{\pm 0.58}$
CGNN	$83.5_{\pm 0.78}$	$91.7_{\pm 1.52}$	$48.9_{\pm 0.74}$	$59.3_{\pm 0.75}$	$34.6_{\pm 0.94}$
L2DGCN	$83.0_{\pm 0.68}$	$90.5_{\pm 1.40}$	$51.7_{\pm 0.66}$	$53.0_{\pm 0.59}$	$35.3_{\pm 0.55}$
Shared four-branch backbone					
GELU	$73.5_{\pm 1.37}$	$89.4_{\pm 1.06}$	$45.1_{\pm 0.9}$	$57.7_{\pm 1.53}$	$34.9_{\pm 0.67}$
Unshifted-LP	$80.5_{\pm 1.04}$	$90.9_{\pm 0.82}$	$50.7_{\pm 0.91}$	$59.3_{\pm 1.24}$	$37.6_{\pm 0.59}$
LEGO-LP (ours)	$83.2_{\pm 1.12}$	$92.1_{\pm 0.93}$	$52.5_{\pm 0.90}$	$60.2_{\pm 1.32}$	$39.2_{\pm 0.44}$

(b) Actor and WebKB

Method	Actor	Cornell	Texas	Wisconsin
GCN	$22.1_{\pm 0.54}$	$40.1_{\pm 3.22}$	$61.6_{\pm 3.94}$	$63.1_{\pm 0.80}$
SAGE	$27.6_{\pm 0.85}$	$54.2_{\pm 2.95}$	$72.5_{\pm 1.67}$	$74.3_{\pm 1.65}$
GAT	$24.9_{\pm 0.63}$	$42.4_{\pm 2.64}$	$63.8_{\pm 0.97}$	$65.2_{\pm 1.01}$
H_2GCN-RW	$25.8_{\pm 1.08}$	$55.1_{\pm 3.09}$	$71.0_{\pm 1.30}$	$73.6_{\pm 1.28}$
GCNII	$26.3_{\pm 1.02}$	$56.2_{\pm 2.88}$	$74.2_{\pm 1.33}$	$78.2_{\pm 2.91}$
MagNet	$26.5_{\pm 0.94}$	$54.9_{\pm 4.27}$	$70.3_{\pm 1.47}$	$77.8_{\pm 1.53}$
GPRGNN	$25.1_{\pm 0.90}$	$51.6_{\pm 4.03}$	$65.6_{\pm 1.31}$	$78.0_{\pm 1.20}$
FAGCN	$26.9_{\pm 1.21}$	$57.0_{\pm 2.90}$	$74.3_{\pm 1.41}$	$79.1_{\pm 1.26}$
ACM-GCN	$26.0_{\pm 0.99}$	$55.2_{\pm 3.01}$	$71.1_{\pm 1.52}$	$76.1_{\pm 1.46}$
GloGNN	$27.2_{\pm 0.72}$	$48.7_{\pm 2.99}$	$67.7_{\pm 1.17}$	$75.4_{\pm 1.11}$
GAT-sep	$27.7_{\pm 0.89}$	$55.2_{\pm 1.54}$	$73.1_{\pm 1.55}$	$70.6_{\pm 2.82}$
Dir-GNN	$27.6_{\pm 0.94}$	$58.1_{\pm 2.79}$	$73.8_{\pm 2.39}$	$78.3_{\pm 1.33}$
Three-channel	$28.0_{\pm 0.82}$	$53.9_{\pm 3.02}$	$68.7_{\pm 1.14}$	$77.3_{\pm 1.20}$
CGNN	$26.6_{\pm 1.16}$	$57.2_{\pm 2.82}$	$75.3_{\pm 3.20}$	$79.9_{\pm 1.25}$
L2DGCN	$31.5_{\pm 0.34}$	$51.7_{\pm 3.24}$	$81.3_{\pm 2.81}$	$80.4_{\pm 3.14}$
Shared four-branch backbone				
GELU	$27.4_{\pm 1.50}$	$59.5_{\pm 2.96}$	$81.5_{\pm 1.62}$	$79.7_{\pm 1.77}$
Unshifted-LP	$30.8_{\pm 0.74}$	$60.1_{\pm 3.33}$	$82.5_{\pm 2.04}$	$80.3_{\pm 1.80}$
LEGO-LP (ours)	$32.0_{\pm 0.50}$	$62.2_{\pm 2.57}$	$83.8_{\pm 2.21}$	$81.5_{\pm 1.31}$

Table 25: Test accuracy on deduplicated Chameleon and Squirrel. Entries are means with standard deviations. The upper block contains the architecture baselines, and the lower block compares GELU, Unshifted-TV, and LEGO-TV in the shared backbone.

Method	Chameleon	Squirrel
Architecture baselines		
GCN (Kipf and Welling, 2017)	$41.52_{\pm 3.87}$	$40.35_{\pm 1.61}$
GAT (Veličković et al., 2018)	$39.69_{\pm 3.52}$	$33.12_{\pm 2.68}$
GPRGNN (Chien et al., 2021)	$40.14_{\pm 3.70}$	$40.03_{\pm 1.55}$
FAGCN (Bo et al., 2021)	$42.31_{\pm 2.56}$	$42.06_{\pm 3.44}$
Shared four-branch backbone		
GELU	$41.24_{\pm 3.63}$	$39.80_{\pm 3.15}$
Unshifted-TV	$41.71_{\pm 2.79}$	$40.54_{\pm 2.60}$
LEGO-TV (ours)	$43.02_{\pm 1.95}$	$43.11_{\pm 3.03}$

Table 26: Node classification on Penn94, arXiv-year, and snap-patents. Entries are mean accuracy with standard deviation for the models trained in the large-graph study.

Datasets	**Penn94**	**arXiv-year**	**snap-patents**
Homophily	0.470	0.222	0.073
GCN (Kipf and Welling, 2017)	$81.0_{\pm 0.6}$	$45.0_{\pm 0.4}$	$43.8_{\pm 0.4}$
GAT (Veličković et al., 2018)	$80.8_{\pm 1.0}$	$45.0_{\pm 0.5}$	$45.1_{\pm 0.4}$
H_2GCN-RW (Zhu et al., 2020)	$80.4_{\pm 0.9}$	$47.6_{\pm 0.4}$	$48.1_{\pm 0.7}$
GPRGNN (Chien et al., 2021)	$81.4_{\pm 0.7}$	$43.6_{\pm 0.8}$	$41.5_{\pm 0.5}$
LINKX (Lim et al., 2021)	$82.2_{\pm 0.6}$	$49.3_{\pm 1.3}$	$52.6_{\pm 0.3}$
LEGO-LP (ours)	$84.3_{\pm 0.5}$	$48.7_{\pm 0.4}$	$49.4_{\pm 0.6}$

Partitioning the incidence rows gives blocks $K_1, \dots, K_J$, where J is the number of edge blocks. With the dual field partitioned into matching blocks $Y_1, \dots, Y_J$, the node aggregation is

$$K^\top Y = \sum_{j=1}^{J} K_j^\top Y_j. \tag{132}$$

The edge penalty is likewise a sum over the blocks. Each block evaluates its edge responses from the same node iterate, and sums their incidence contributions before the next primal update. For the edge partition $E_1, \dots, E_J$, the weight normalizer remains

$$\overline{w} = \frac{1}{m} \sum_{j=1}^{J} \sum_{e \in E_j} \overline{w}_e, \qquad w_e = s_w \frac{\overline{w}_e}{\overline{w}}. \tag{133}$$

The scalar $\overline{w}$ is the graph-wide mean of WeightNet's positive scores $\overline{w}_e$. Targets and dual states retain their incidence-row correspondence across blocks, and all blocks use the step sizes determined by the full graph's maximum degree. Partitioning distributes the existing edge calculation and leaves the mathematical update unchanged.

LEGO-LP has the highest mean accuracy on Penn94; LINKX leads on arXiv-year and snap-patents. Appendix B.4 describes how graph size affects storage of edge fields and differentiation intermediates.

C.19 Graph-Coupled Block Transfer to GCN and GAT

Table 3 compares Original, Unshifted-TV, and LEGO-TV within GCN (Kipf and Welling, 2017) and GAT (Veličković et al., 2018). The graph-coupled blocks replace the pointwise map at each hidden activation position.

Backbone-preserving integration. Each comparison preserves the backbone's aggregation, self-feature treatment, normalization, residual connections, dropout placement, encoder, and classifier. For the coupled blocks, we use the backbone pre-activation as $Z^{(\ell)}$ and compute the positive weights

from the incoming states $H^{(\ell)}$. LEGO-TV also predicts the antisymmetric clipped target through OffsetNet. We pass the unrolled output in Eq. 14 to the backbone's post-activation operations.

Matched variants and training. Unshifted-TV sets $M = 0$, and LEGO-TV generates antisymmetric clipped targets with OffsetNet. The two models share the WeightNet architecture, anchor scale, and TV update rule. Original retains its pointwise activation. We train all three variants independently with paired common-module initializations.

Each backbone uses width 128 and depth two. The TV variants use $\alpha = 1$, $K_{\text{iter}} = 12$, and $T_{\text{newton}} = 0$ for closed-form responses. All variants share the optimizer, learning-rate schedule, preprocessing, partitions, and validation-based early stopping, with test evaluation at the selected checkpoints.

Evaluation follows Appendix C.1, with the same incidence representatives in both coupled variants.

Results. LEGO-TV achieves higher mean scores than Unshifted-TV on every graph in both GCN and GAT. Relative to Original, zero-target coupling improves GAT on every graph and GCN on four graphs. LEGO-TV improves both backbones across the five graphs.

D Related Work

D.1 Translational Relation Geometry

TransE represents a relation as a displacement between entity embeddings (Bordes et al., 2013). With head $h = U_j$, tail $t = U_i$, and relation vector $r = \mu_e$, its residual norm is $\|U_i - U_j - \mu_e\|_2$. LEGO places this displacement inside an anchored neural layer. OffsetNet predicts the target from endpoint features, and node-classification supervision trains the representation obtained by the ensuing graph updates. The displacement is thus learned as part of computing node states.

The composition of these translations around an oriented cycle C gives

$$\mathrm{hol}_C(M) = \sum_{e \in C} \sigma_e(C)\mu_e, \qquad \mathrm{hol}_C(M - KB) = \mathrm{hol}_C(M). \tag{134}$$

The additive holonomy $\mathrm{hol}_C(M)$ is the cycle sum, with traversal signs $\sigma_e(C)$. Node-potential differences telescope around a cycle. Therefore, the incidence decomposition separates the target into realizable node differences and a cycle component that preserves these closed-path discrepancies (Lim, 2020). The finite translation identity shows how this decomposition acts on the implemented layer.

D.2 Graph-Adaptive Nonlinear Operations

Graph-adaptive activations make the nonlinear stage responsive to neighborhood information. Median and localized invariant activations use local graph structure (Ruiz et al., 2019; 2020); learned kernel combinations and featurewise rectification adapt the response to the graph (Iancu et al., 2020; Zhang et al., 2022). DIGRAF learns a graph-adaptive diffeomorphic map (Mantri et al., 2024), and cluster-normalize-activate modules use groups of representations to define their transformations (Skryagin et al., 2024). We follow these approaches to incorporate graph information into nonlinear computation.

LEGO implements that stage through edge responses and node updates. The target gives the preferred difference; the edge response depends on its discrepancy from the current endpoint difference. The output depends on interactions among node states throughout the unrolled computation. The GCN and GAT experiments examine this placement directly, and the TV comparison tests learned centers with a fixed radial response shape.

D.3 Anchored Optimization and Target Placement

Elastic GNNs combine ℓ_1 and ℓ_2 graph penalties, linking local adaptivity to an anchored optimization problem (Liu et al., 2021b). TWIRLS unfolds robust graph energies through iteratively reweighted

least squares (Yang et al., 2021), and p-Laplacian regularization gives another route to nonlinear graph propagation (Fu et al., 2022). LEGO shares the anchored formulation and learns the center of each edge penalty. Positive weights set interaction strengths, and additive targets provide preferred differences on the standard incidence graph.

The difference lies in where the learned adjustment is applied. AirGNN applies an ℓ_{21} penalty to node deviations from the input (Liu et al., 2021a); LEGO-TV applies Euclidean norms to target-centered edge residuals. For a quadratic edge penalty, the learned target enters the exact solution through the anchor change $Z + \alpha K^\top D_W M$. Anchor injection applies this transformation before Unshifted-LP with the full OffsetNet architecture. TV and LP additionally respond to the target-centered edge residuals before incidence aggregation, as described by Eq. 20.

Optimization-based graph models (Zhu et al., 2021b), differentiable convex layers (Agrawal et al., 2019), and deep declarative networks (Gould et al., 2022) place structured computations within trainable networks. The computation and its differentiation can occur at several levels: proximal backpropagation acts on parameter updates (Frerix et al., 2017), implicit graph models and monotone equilibrium networks define representations through equilibrium equations (Gu et al., 2020; Winston and Kolter, 2020; Liu et al., 2024), and graph neural ordinary differential equations use continuous-depth dynamics (Poli et al., 2019). (Yang et al., 2025) studies the relation between implicit and unfolded graph models. LEGO differentiates through its prescribed Chambolle-Pock recurrence (Chambolle and Pock, 2011). Its translation identity applies at finite depth, and the perturbation bound connects radial residuals to the implemented output.

D.4 Graph Differences and Learned Transport

Graph trend filtering uses difference penalties for adaptive estimation and recommendation (Wang et al., 2016; Fan et al., 2022). The operator defining a difference can itself encode relational structure. MagNet uses phase-dependent comparisons (Zhang et al., 2021), and Neural Sheaf Diffusion learns linear maps from node spaces to edge spaces (Bodnar et al., 2022). In these constructions, the learned or phase-dependent operators define the comparison between endpoint representations.

LEGO uses the standard incidence difference and learns an additive target for each edge. Its geometric analysis therefore uses the graph's incidence subspaces: integrable targets are node-potential differences, and cycle targets carry closed-path discrepancies. Projected-OffsetNet keeps the full target-network architecture and projects its output into the integrable subspace. With this control, we relate the additive parameterization directly to the decomposition used in the analysis.

D.5 Relations Encoded in Propagation

Message-passing and relational frameworks describe interactions through node and edge computations (Gilmer et al., 2017; Battaglia et al., 2018). Edge-conditioned filters learn aggregation from edge information (Simonovsky and Komodakis, 2017). Signed convolution and attention encode positive and negative relations (Derr et al., 2018; Huang et al., 2019), and multi-class signed propagation and selective blocking adapt interactions to heterogeneous neighborhoods (Choi et al., 2025a;b). Directed and magnetic signed operators use observed edge direction or sign (Rossi et al., 2024; He et al., 2022; Ko et al., 2023; Marques et al., 2020).

LEGO learns the preferred endpoint difference in the nonlinear block following propagation. Its incidence orientation specifies coordinates for this difference. Antisymmetric targets and symmetric weights make the node output invariant to reversal of those coordinates. We use edge reassignment to test the learned correspondence between a target and its endpoint pair.

D.6 Propagation and the Shared Backbone

The anchor combines information across neighborhoods, drawing on spectral filters, hop mixing, and diffusion (Defferrard et al., 2016; Abu-El-Haija et al., 2019; Gasteiger et al., 2019; Klicpera et al., 2019; Wu et al., 2019). Intermediate representations remain available through depth-wise readout and residual paths (Xu et al., 2018; Chen et al., 2020). Graph perturbations and neighborhood statistics offer additional architectural choices (Rong et al., 2019; Corso et al., 2020). Broader reviews organize these components (Wu et al., 2021; Zhang et al., 2019).

Heterophily-oriented filters adapt frequencies and propagation ranges (Bo et al., 2021; Chien et al., 2021; He et al., 2021; Lei et al., 2022). Gating, global interactions, and filter ensembles vary in their combination (Du et al., 2022; Li et al., 2022; 2024; Duan et al., 2024). Edge and label-compatibility models adapt endpoint interactions (Guo et al., 2024; Zhu et al., 2021a; Luan et al., 2022; Wang et al., 2024). Diffusion analyses explain the resulting representation dynamics (Wang et al., 2021; Yan et al., 2022), and architecture search and surveys organize the design space (Zheng et al., 2023; 2022). We draw on these architectural choices in forming the distinct feature paths of LEGO's anchor.

Structural expressivity (Xu et al., 2019; Morris et al., 2019; Wang and Zhang, 2022) and effective resistance (Black et al., 2023) characterize discrimination and information flow. Transformers extend interactions globally (Ying et al., 2021; Rampášek et al., 2022; Wu et al., 2023), and additive graph networks structure prediction contributions (Bechler-Speicher et al., 2024). In our comparisons, we keep the surrounding architecture fixed and examine the preferred differences learned inside its nonlinear block.

E LIMITATIONS

Finite computation and parameter scaling. LEGO implements a prescribed-depth nonlinear map. Its distance from the variational reference depends on outer truncation and radial residuals, as quantified in Theorem 1. The scalar study shows that finite projected-Newton iterates can exhibit periodic behavior for steep admissible potentials, so agreement between two iteration budgets is not by itself an optimality certificate. The checkpoint depth study therefore characterizes empirical prediction sensitivity to additional finite updates, while the scalar study assesses radial optimality directly through the residual Γ_e. For LP, the mean-weight regularizer acts on s_w, and the effective interaction scale is the product $s_w L_\theta$ described in Appendix B.1.

Graph setting and computational cost. The experiments address transductive node prediction with shared feature preprocessing and train/validation/test partitions. Chameleon and Squirrel use the full-size graph versions described in Appendix C, including their duplicate nodes. Graph-coupled computation maintains both node and edge states, which increases latency and memory use relative to pointwise activation. The large-graph experiments use edge chunking and additional devices to accommodate these states.