PROPAGATE, THEN SHARPEN: POST-HOC REFINEMENT OF FROZEN NODE CLASSIFIERS

Anonymous authors
Paper under double-blind review

ABSTRACT

We study post-hoc refinement of frozen node classifiers: given only the graph G and class distributions Q predicted by a frozen model, can we improve accuracy without access to node features, model parameters, or gradients? APPNP answers this by propagating logits with a restart towards the initial predictions, minimizing the anchored Dirichlet energy. Instead, we consider the Potts energy, and decompose it into a Dirichlet term, which penalizes disagreement between neighbouring nodes, and a Gini term, which penalizes indecision within each node. This decomposition motivates *Propagate, Then Sharpen* (PtS), which alternates between propagation of class probabilities and node-wise, mass-preserving sharpening, with only one additional hyperparameter selected using labelled validation nodes. Across nine homophilic graphs, with a frozen MLP backbone, PtS improves mean test accuracy over independently tuned APPNP by 1.71 percentage points on clean inputs and 3.90 under severe Gaussian feature corruption. Gains over APPNP become smaller, but remain positive with frozen GCN and GraphSAGE backbones. Sharpening also removes most of the accuracy loss of deep propagation: on clean inputs without restart, accuracy falls by 2.2 points between 2 and 100 propagation steps under PtS, compared with 33.8 for APPNP.

1 INTRODUCTION

Graph-based node classification uses node attributes (features) and relationships between nodes (structure). These two inputs are not equally reliable. Features can be fragile: sensors drift or fail, attributes go missing or stale for part of the graph, or features are perturbed to protect privacy, while the graph structure usually stays intact. When this happens, correcting the model is often not an option, because retraining may be costly or access to parameters or training data might be restricted. These constraints motivate post-hoc methods that reuse the deployed model as a frozen predictor (Boudiaf et al., 2022). We ask how much prediction accuracy can be recovered using only the frozen class distributions Q and the graph G, without access to node features, model parameters, gradients, or training data. The refinement should apply to any backbone, so that another predictor can easily be swapped in. Validation labels should only be used for hyperparameter selection, and refinement should tolerate a mismatch between validation and test conditions. We therefore evaluate refinement both on clean inputs, and under Gaussian feature corruption on datasets with continuous-valued node features for which additive Gaussian noise provides a meaningful controlled perturbation.

A natural way to recover accuracy on G is to propagate predictions along its edges, and APPNP is the standard way to do so: personalized PageRank propagation of the logits, with a restart towards the original prediction at every step (Gasteiger et al., 2019). Propagation exploits informative neighbours, but repeated propagation also removes distinctions between nodes, which can result in oversmoothing (Oono & Suzuki, 2020), i.e. predictions in a connected component converge towards a common value as the number of propagation steps grows. Restart is what prevents this collapse in APPNP, but it only helps if the reference prediction is reliable. If features are missing or corrupted at test time, the restart repeatedly re-injects corrupted predictions, and propagation spreads these errors through the graph. APPNP is therefore sensitive to its depth and restart hyperparameters, and without restart it collapses at large depth (Figure 2).

The space in which predictions are propagated matters. APPNP propagates logits, so large class margins exert great influence, which helps when margins are reliable but amplifies confident errors.

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.

Propagating probabilities instead bounds the initial scores in $[0, 1]$, limiting the influence of extreme logits. The trade-off is that averaging neighbouring distributions can weaken class preferences when neighbours disagree. This motivates a refinement method that acts in two directions: smoothing predictions across neighbouring nodes, and sharpening the class distribution within each node. We start from the relaxed Potts interaction used in variational image segmentation (Potts, 1952; Liu et al., 2022; Tai et al., 2024; Liu et al., 2024). On the probability simplex, the interaction splits exactly into a Dirichlet term, which penalizes disagreement between neighbouring nodes, and a Gini term, which penalizes indecision within each node. This decomposition pairs graph propagation with a local sharpening step that reduces indecision within each node.

We call the resulting method Propagate, Then Sharpen (PtS). It alternates between a propagation and a sharpening step. Sharpening is controlled by a single parameter η: at $\eta = 0$ we recover propagation without sharpening exactly, so validation can switch sharpening off, as $\eta \to \infty$ sharpening approaches a one-hot assignment when the most probable class is unique, and in between it acts as a soft thresholding. Sharpening never changes a node's own predicted class, it changes the distribution used by the next propagation step. PtS uses only Q and G, labels enter only through hyperparameter selection, any backbone that outputs class probabilities can be used, and its gain over APPNP persists at large propagation depth (Figure 2), as well as when hyperparameters are selected at a different corruption severity than the one evaluated at (Figure 4).

Contributions

- We introduce Propagate, Then Sharpen (PtS), which alternates probability-space graph propagation with a mass-preserving categorical sharpening step derived from the Dirichlet–Gini decomposition of the Potts interaction. Sharpening adds one hyperparameter η, is off at $\eta = 0$, provably preserves every node's predicted class, and decreases a local KL–Gini energy at each step (Proposition 1).
- We show that sharpening reduces accuracy degradation with propagation depth. On a constructed two-community graph, restart alone fails while PtS keeps every node correct at every depth (Proposition 2). Across nine homophilic graphs on clean inputs without restart, increasing depth from $K = 2$ to $K = 100$ reduces mean accuracy by 5.2 percentage points for PtS and 22.0 for Logit-Sharp (APPNP with sharpening in logit-space) at the same sharpening parameter $\eta = 16$. Increasing PtS's sharpening parameter to $\eta = 200$ reduces its drop to 2.2 points (Figure 2).
- On nine homophilic graphs, PtS gains 3.90 points over independently tuned APPNP under severe Gaussian corruption and 1.71 on clean inputs (Table 1), and improves on average over a graph-adapted LAME and a post-hoc Graph-TV baseline (Tables 8 and 9).

Related Work The sharpening step of PtS comes from variational image segmentation, where the Potts model penalizes the boundaries between regions of a soft assignment field. PottsMGNet and Double-well Net connect this formulation with trainable neural architectures by deriving their updates from Potts energies and operator-splitting schemes (Tai et al., 2024; Liu et al., 2024), and soft threshold dynamics adds spatial regularization and shape priors to the output of a segmentation network (Liu et al., 2022). We transfer the same splitting idea from the image domain to a graph, where the assignment field becomes the class distributions of the nodes and the split separates smoothing from local sharpening. The combination already appears on graphs in diffuse-interface methods, which pair smoothing with phase separation, and in Merriman–Bence–Osher (MBO) schemes, which alternate diffusion and thresholding (Bertozzi & Flenner, 2012; Merkurjev et al., 2013; Garcia-Cardona et al., 2014). PtS is a soft, mass-preserving counterpart applied to frozen predictions rather than labels, approaching hard thresholding as $\eta \to \infty$ when the largest class probability is unique. Allen–Cahn Message Passing (ACMP) and GREAD apply reaction–diffusion dynamics to learned node representations inside trained graph neural networks (Wang et al., 2023; Choi et al., 2023), PtS acts on the outputs of an already trained model and has no learnable parameters. Yang et al. (2025) incorporate a graph-based non-local TV-regularized softmax into GNN training. We evaluate its objective as a post-hoc baseline on frozen predictions, denoted Graph-TV (Appendix I).

Post-hoc propagation methods refine predictions by spreading information through the graph structure. PPNP and APPNP combine neural prediction with personalized PageRank propagation in logit space, with APPNP using an iterative approximation with restart (Gasteiger et al., 2019). Correct & Smooth applies label propagation after training, using known training labels to correct prediction errors and smooth the corrected predictions (Huang et al., 2021). PtS uses no labels at inference, but its sharpening step can be inserted into the smoothing stage of Correct & Smooth (Appendix L). Lapla-

cian Adjusted Maximum-Likelihood Estimation (LAME) refines a frozen classifier's output without updating its parameters. Its objective combines KL fidelity to the original class probabilities with a pairwise agreement term weighted by affinities computed from pretrained feature representations (Boudiaf et al., 2022). Replacing that affinity with the graph operator turns its update into a fixed-point iteration of the full KL-plus-Potts energy (Appendix C), which we evaluate as a further baseline (LAME-Graph). PtS alternates anchored propagation with local, mass-preserving sharpening, using the propagated distribution as the reference for each sharpening step.

2 METHOD

Let $G = (\mathcal{V}, \mathcal{E})$ be a graph with $N = |\mathcal{V}|$ nodes and let C be the number of classes. We symmetrize directed source graphs, remove duplicate edges and existing self-loops. Let $A \in \{0,1\}^{N\times N}$ denote the resulting binary adjacency without self-loops, $\widetilde{A} = A + I$ the adjacency with added self-loops, $\widetilde{D}$ the diagonal degree matrix $\widetilde{D}_{ii} = \sum_j \widetilde{A}_{ij}$, and S the normalised adjacency

$$S = \widetilde{D}^{-1/2}\widetilde{A}\,\widetilde{D}^{-1/2}.$$

The operator S is symmetric, element-wise non-negative, and $\mathrm{spec}(S) \subseteq [-1, 1]$ with 1 an eigenvalue.

A frozen classifier f_θ produces logits $Z \in \mathbb{R}^{N\times C}$ and class distributions $Q = \mathrm{softmax}(Z) \in \mathbb{R}^{N\times C}$, with each row on the probability simplex $\Delta^{C-1} = \{p \in \mathbb{R}^C : p \geq 0,\ \mathbf{1}^\top p = 1\}$.

Refinement produces a sequence of states $U^{(0)}, \ldots, U^{(K)} \in \mathbb{R}^{N\times C}$. We write $U_i^{(k)} \in \mathbb{R}^C$ for row i of $U^{(k)}$, treated as a column vector.

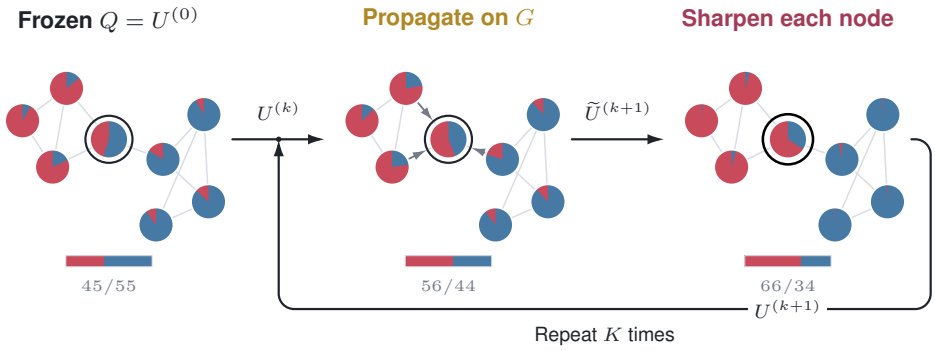

Figure 1: **Propagate, Then Sharpen.** Node colours show two-class proportions in an illustrative first iteration ($\alpha = 0.1$, $\eta = 3.5$), initialized at Q. Propagation and sharpening update all nodes, and the sharpened state becomes the current state for the next iteration. Propagation includes self-loops and a restart towards the fixed Q. Pie charts show normalised class proportions.

2.1 PROPAGATE

We start from the personalized PageRank recursion of APPNP (Gasteiger et al., 2019), but apply it to Q instead of logits:

$$U^{(0)} = Q, \qquad U^{(k+1)} = \mathcal{T}\big(U^{(k)}\big), \qquad \mathcal{T}(U) = \alpha Q + (1-\alpha)\,SU \tag{1}$$

with restart probability $\alpha \in [0, 1]$ and K propagation steps. The only difference from APPNP is the state being propagated: APPNP propagates the logits Z and applies a softmax at the end, whereas this recursion propagates the class distributions Q. We call the variant *PPR-Prob* and use it to isolate the effect of the propagation space. Since S does not preserve unit row sums, we retain the unnormalized states during propagation and row-normalize $U^{(K)}$ to obtain the final class distributions. Normalising after every step instead is the PPR-rn variant compared in Appendix N.

2.2 PROPAGATE, THEN SHARPEN

PtS extends this propagation with a node-wise categorical sharpening step. In a reaction–diffusion interpretation, this local sharpening plays the role of the reaction step. As illustrated in Figure 1, from

$U^{(0)} = Q$, we alternate between

$$\widetilde{U}^{(k+1)} = \alpha Q + (1-\alpha) S U^{(k)} \qquad \text{propagate} \tag{2a}$$

$$U_i^{(k+1)} = R_\eta\left(\widetilde{U}_i^{(k+1)}\right) \quad i = 1, \ldots, N \qquad \text{sharpen} \tag{2b}$$

where the reaction R_η is defined in (4) below. Symmetric normalization does not preserve unit row sums. We therefore define the sharpening map on the simplex and then lift it to unnormalised rows. For $p \in \Delta^{C-1}$ with $p > 0$ and a reaction strength $\eta \geq 0$, let

$$r_\eta(p) \;=\; \mathrm{softmax}(\log p + \eta\, p)\,, \tag{3}$$

which maps Δ^{C-1} into itself. For a propagated row $h \in \mathbb{R}^C_{>0}$ with total mass $m(h) = \mathbf{1}^\top h$, the reaction is

$$R_\eta(h) \;=\; m(h)\; r_\eta\!\left(\frac{h}{m(h)}\right). \tag{4}$$

At $\eta = 0$ we recover $R_0(h) = h$, so PtS reduces exactly to PPR-Prob with the same (α, K). Every row of $\widetilde{U}^{(k+1)}$ is strictly positive, so (4) is well defined: $Q = \mathrm{softmax}(Z) > 0$ elementwise, and the added self-loops give S a strictly positive diagonal, so positivity is preserved for every $\alpha \in [0, 1]$.

Writing the reaction in the two stages of (3)–(4) separates what it does. The map r_η redistributes probability mass within a node, and the factor $m(h)$ restores the row total,

$$\mathbf{1}^\top R_\eta(h) \;=\; m(h)\, \mathbf{1}^\top r_\eta(h/m(h)) \;=\; m(h) \;=\; \mathbf{1}^\top h.$$

The reaction therefore changes a node's class distribution while preserving the mass the row carries into the next propagation step, which is what determines the node's total contribution to its neighbours. We compare this choice with normalizing each row's mass after every iteration in Table 16. Since η multiplies the normalised rows, the sharpening strength is invariant to the row total.

For every $\eta \geq 0$, the reaction keeps the relative ranking of the classes within each row (Appendix D). It therefore never changes a node's own predicted class, and influences later predictions only through the state that enters the next propagation step.

Without restart ($\alpha = 0$) and without sharpening, propagation collapses: on a connected graph, every node converges to the same class distribution, and the predictions of APPNP converge to a single class shared by all nodes (Appendix D.1). The experiments in Figure 2 show this collapse and how sharpening counteracts it.

2.3 Energy interpretation and local descent

Motivated by the entropy-regularized Potts formulations in Tai et al. (2024), consider normalised class distributions $u_i \in \Delta^{C-1}$ and a symmetric affinity matrix W with $W_{ij} \geq 0$ (for PtS, $W = S$). The relaxed Potts interaction is

$$E(U) = \frac{1}{2} \sum_{i,j} W_{ij} \left(1 - u_i^\top u_j\right).$$

For one-hot assignments, this is a weighted penalty for disagreement between connected nodes. For soft assignments, it also penalizes indecision within each node. Writing $d_i = \sum_j W_{ij}$ gives

$$E(U) = \underbrace{\frac{1}{4} \sum_{i,j} W_{ij} \|u_i - u_j\|^2}_{E_{\mathrm{D}}(U)} + \underbrace{\frac{1}{2} \sum_i d_i \left(1 - \|u_i\|^2\right)}_{E_{\mathrm{G}}(U)}.$$

The Dirichlet term $E_{\mathrm{D}}(U)$ penalizes differences between neighbouring distributions, while the Gini term $E_{\mathrm{G}}(U)$ penalizes indecision within each node. This decomposition motivates the two operations of PtS. Propagation is a gradient step on an anchored, degree-normalised Dirichlet objective, while sharpening decreases a local KL–Gini energy. This does not establish descent of the joint Potts objective. Figure 6 illustrates how these components interact during propagation on WikiCS.

Propagation descends an anchored, degree-normalised Dirichlet objective. Following the optimization view of APPNP, consider $J(U) = \frac{\alpha}{2}\|U - Q\|_F^2 + \frac{1-\alpha}{2}\,\mathrm{tr}\big(U^\top (I - S) U\big)$, whose first term anchors

the state near Q and whose second term is a degree-normalised Dirichlet energy, so that α balances closeness to the original prediction against smoothness over the graph. Since $\nabla J(U) = \alpha(U - Q) + (1-\alpha)(I - S)U$, a gradient step with unit step size gives $U - \nabla J(U) = \alpha Q + (1-\alpha)SU = \mathcal{T}(U)$: one propagation step is exactly one unit-step gradient step on J (Zhu et al., 2021).

Sharpening descends the Gini term of a single node. For a propagated class distribution p_i, define

$$E_\eta(u; p_i) = \mathrm{KL}(u \| p_i) + \frac{\eta}{2}\left(1 - \|u\|^2\right), \qquad u \in \Delta^{C-1} \tag{5}$$

where the KL term keeps the update close to p_i and the Gini term favours more decisive class distributions. Stationarity gives the fixed-point condition $u = \mathrm{softmax}(\log p_i + \eta u)$, and one fixed-point step initialized at $u = p_i$, as in the limited fixed-point iterations of PottsMGNet (Tai et al., 2024), returns exactly the sharpening map (3). Appendix B gives the full derivation, and that single step already decreases the energy it comes from.

Proposition 1 (Sharpening descends the local energy)**.** *Let $p \in \Delta^{C-1}$ with $p > 0$, let $\eta \geq 0$ and let $p^* = r_\eta(p)$. Then $E_\eta(p^*; p) \leq E_\eta(p; p)$, and consequently $\frac{\eta}{2}\left(\|p^*\|^2 - \|p\|^2\right) \geq \mathrm{KL}(p^* \| p) \geq 0$, so one sharpening step does not increase the Gini term. Moreover, p^* preserves the ordering of the classes of p.*

The proof (Appendix D) uses the concavity of the Gini term: sharpening minimizes a linearized upper bound of $E_\eta(\cdot\,; p)$ that is tight at p, so iterating it is a concave–convex procedure for (5) (Yuille & Rangarajan, 2003).

Restart is not enough Sharpening also changes the behaviour of deep propagation. Restart does not by itself protect a community from a confident neighbouring one: on a two-community graph, APPNP and PPR-Prob with $\alpha = 0.1$ misclassify one entire community for every depth $K \geq 3$, while PtS with $\eta = 4$ keeps every node correct at every depth (Proposition 2 and Figure 5 in Appendix D.2). The mechanism there is that propagation cannot push a node's correct-class probability below 0.546 while it is at least 0.6 everywhere, and one sharpening step lifts 0.546 back above 0.6.

The Potts decomposition motivates combining agreement between neighbours with decisiveness within each node. PtS propagates class probabilities, then applies one local KL–Gini sharpening step while preserving the propagated row mass. Sharpening preserves the current predicted class, so it affects labels only through subsequent propagation. Setting $\eta = 0$ recovers PPR-Prob exactly.

3 Experimental protocol

We use nine homophilic graphs: WikiCS (Mernyei & Cangea, 2022), Cora-TAPE, PubMed-TAPE and TAPE-Arxiv23 (He et al., 2024), ogbn-arxiv and ogbn-products (Hu et al., 2020), and Ele-Photo, Ele-Computers and Books-History (Yan et al., 2023). We use Roman-Empire and Amazon-Ratings (Platonov et al., 2023) as heterophilic controls and report them separately. WikiCS uses its first ten supplied training/validation splits and its shared test mask, the OGB graphs use their official split, and all other graphs use ten unstratified random 60/20/20 splits. We use three backbone seeds per split, each (graph, split, backbone seed) defines one *unit*. Dataset statistics are given in Appendix E.

For each unit we train a graph-blind MLP on clean features, minimizing cross-entropy on the training labels, and freeze the checkpoint with the highest clean validation accuracy, architecture and optimizer settings are given in Appendix F. The GCN (Kipf & Welling, 2017) and GraphSAGE (Hamilton et al., 2017) backbones follow the same training and corruption protocol, model sizes are given in Table 5.

At inference, we corrupt all nodes as

$$X_{ij}^{(\sigma,r)} = X_{ij} + \sigma s_j \xi_{ij}^{(r)}, \qquad \xi_{ij}^{(r)} \sim \mathcal{N}(0,1),$$

where s_j is the standard deviation of feature j over the training nodes and $\sigma \in \{0, 0.5, 1, 1.5, 2\}$. We draw three independent Gaussian fields $\xi^{(r)}$ per graph and split, shared across backbone seeds and methods, and reuse them across severities, so that the severities form a nested noise ladder. Cora-TAPE and PubMed-TAPE use RoBERTa-base (Liu et al., 2019) CLS embeddings computed without fine-tuning, and the three CS-TAG graphs (Ele-Photo, Ele-Computers, Books-History) use the published RoBERTa-base CLS embeddings. For each corruption, we compute $Z = f_\theta(X^{(\sigma,r)})$

and $Q = \mathrm{softmax}(Z)$ once and use these across methods and experiments. APPNP refers to the propagation stage of Gasteiger et al. (2019), applied post hoc to frozen logits Z, rather than the full model trained end to end.

APPNP, PPR-Prob and PtS are each tuned independently with 250 Optuna trials (Akiba et al., 2019) per unit, severity and noise draw, maximizing validation accuracy. APPNP and PPR-Prob tune (α, K), and PtS additionally tunes η, including $\eta = 0$, full search spaces are in Appendix G and the selected values in Appendix H. Unless stated otherwise, hyperparameters are selected at the same severity at which they are evaluated, transfer across severities is reported in Figure 4. For each severity, we average test accuracies over draws, then over seeds, within each split, and report mean $\pm$ standard deviation across splits. For the OGB graphs, which have a single official split, standard deviations instead describe variation across seed$\times$draw repeats. Differences between methods are computed within (split, seed, draw) before aggregation.

We compare against two external post-hoc baselines using the same frozen predictions, graph and validation data. LAME-Graph is the graph-adapted LAME of Boudiaf et al. (2022), which replaces the feature affinity by S (Appendix C) and tunes the interaction strength and the number of iterations using 250 Optuna trials. Graph-TV applies the non-local total-variation objective of Yang et al. (2025) post hoc to the frozen predictions, with $\varepsilon = 1$ and the regularization strength selected by validation accuracy over a grid of 26 candidates. Graph-TV is evaluated at $\sigma \in \{0, 2\}$ only and excludes ogbn-products because of its memory requirements. Both baselines are detailed in Appendix I.

4 Results

4.1 Accuracy on clean and corrupted inputs

Across the nine main graphs with a frozen MLP, PtS improves mean test accuracy over independently tuned PPR-Prob by 0.57 percentage points on clean inputs and 1.09 at $\sigma = 2$. The corresponding gains over post-hoc APPNP are 1.71 and 3.90 points (Table 1). With the frozen MLP backbone at $\sigma = 2$, mean accuracy rises from 41.95% for the unrefined predictions Q to 56.34% for APPNP and 60.24% for PtS.

Table 1: Test accuracy (%) with a frozen MLP backbone. Each method is tuned independently at the evaluation severity. The mean excludes the heterophilic controls.

	$\sigma = 0$				$\sigma = 2$			
Dataset	Q(MLP)	APPNP	PtS	PtS − APPNP	Q(MLP)	APPNP	PtS	PtS − APPNP
WikiCS	72.67	77.78	**78.42**	+0.64 ± 0.50	46.36	67.97	**72.15**	+4.18 ± 1.75
Cora-TAPE	60.51	76.16	**79.74**	+3.58 ± 1.23	43.82	68.16	**72.30**	+4.13 ± 1.99
PubMed-TAPE	81.59	84.35	**85.18**	+0.83 ± 0.64	64.56	78.87	**80.73**	+1.86 ± 1.44
TAPE-Arxiv23	67.59	69.53	**69.72**	+0.19 ± 0.10	28.56	35.63	**38.91**	+3.29 ± 0.26
ogbn-arxiv	55.95	66.10	**66.56**	+0.46 ± 0.51	22.65	33.86	**40.62**	+6.76 ± 1.73
ogbn-products	59.58	70.58	**72.17**	+1.58 ± 0.24	22.46	**28.54**	27.33	-1.21 ± 0.17
Ele-Photo	66.83	71.87	**74.91**	+3.04 ± 0.63	44.15	55.97	**61.43**	+5.46 ± 1.04
Ele-Computers	60.96	75.37	**80.11**	+4.74 ± 0.96	38.05	59.94	**69.56**	+9.63 ± 1.00
Books-History	81.38	82.59	**82.88**	+0.29 ± 0.23	66.91	78.11	**79.13**	+1.02 ± 0.61
heterophilic controls								
Roman-Empire	65.54	**65.65**	65.54	-0.11 ± 0.13	**19.34**	19.30	19.30	-0.01 ± 0.02
Amazon-Ratings	49.52	51.63	**52.74**	+1.12 ± 0.35	34.17	36.74	**36.78**	+0.04 ± 0.08
Mean (9 main)	67.45	74.93	**76.63**	+1.71	41.95	56.34	**60.24**	+3.90

PtS beats APPNP on all nine main graphs on clean inputs and on eight out of nine at $\sigma = 2$, where the positive gains range from 1.02 to 9.63 points. The exception is ogbn-products, where APPNP is better on noisy data while PtS beats it on clean data. The advantage over APPNP is negligible on the two heterophilic controls at $\sigma = 2$. For the other baselines, at $\sigma = 2$, PtS also improves mean accuracy over LAME-Graph by 3.75 points across nine graphs. LAME-Graph uses the coupled Potts update, placing the graph interaction inside the softmax while retaining Q as its reference, PtS propagates first and sharpens relative to the propagated distribution (Appendix C). Against Graph-TV, the improvement is 4.24 points on the eight graphs evaluated by both methods (Table 8).

4.2 SHARPENING AND PROPAGATION DEPTH

At $\sigma = 2$, independently tuned PPR-Prob improves mean accuracy over APPNP by 2.81 points, and PtS improves over PPR-Prob by an additional 1.09 points. On clean inputs, the PtS–PPR-Prob gap is 0.57 points. Per-dataset comparisons are given in Table 10, and the full corruption ladder is reported in Table 14. Additionally, to test if the selected PtS configurations depend on sharpening, we keep their (α, K) fixed and set $\eta = 0$ without retuning (Reaction OFF). Mean accuracy at $\sigma = 2$ then falls from 60.24% to 54.93%, a drop of 5.31 points (Table 2). Figure 3 visualizes how the two contributions change across the full corruption ladder, and Table 14 gives the dataset-level values.

Table 2: Mean accuracy (%) over the nine main graphs at $\sigma = 2$ with the frozen MLP. PPR-Prob is independently tuned, Reaction OFF reuses the PtS-selected propagation settings without retuning.

Method	Sharpening	Accuracy
PPR-Prob	$\eta = 0$	59.15
PtS	Selected η	**60.24**
Reaction OFF	$\eta = 0$	54.93

Next, we vary the propagation depth K, while holding each curve's restart α and sharpening strength η fixed (Figure 2). Logit-Sharp applies the sharpening step on top of logit-space propagation using APPNP.

Without restart, increasing K from 2 to 100 reduces clean accuracy by 33.8 points for APPNP and 31.2 for PPR-Prob, compared with 2.2 points for PtS with $\eta = 200$. With noise $\sigma = 2$, APPNP and PPR-Prob lose 21.1 and 20.5 points, while PtS with $\eta = 200$ reaches 57.7% at both depths with a maximum in between. With restart $\alpha = 0.1$ on clean inputs, accuracy decreases from 74.5% to 70.4% for APPNP and from 75.7% to 74.8% for PtS with $\eta = 200$ between $K = 2$ and $K = 100$: losses of approximately 4.1 and 0.9 percentage points. At $K = 100$, both PtS curves remain above APPNP and PPR-Prob in every restart and corruption condition shown; complete depth results are reported in Table 17.

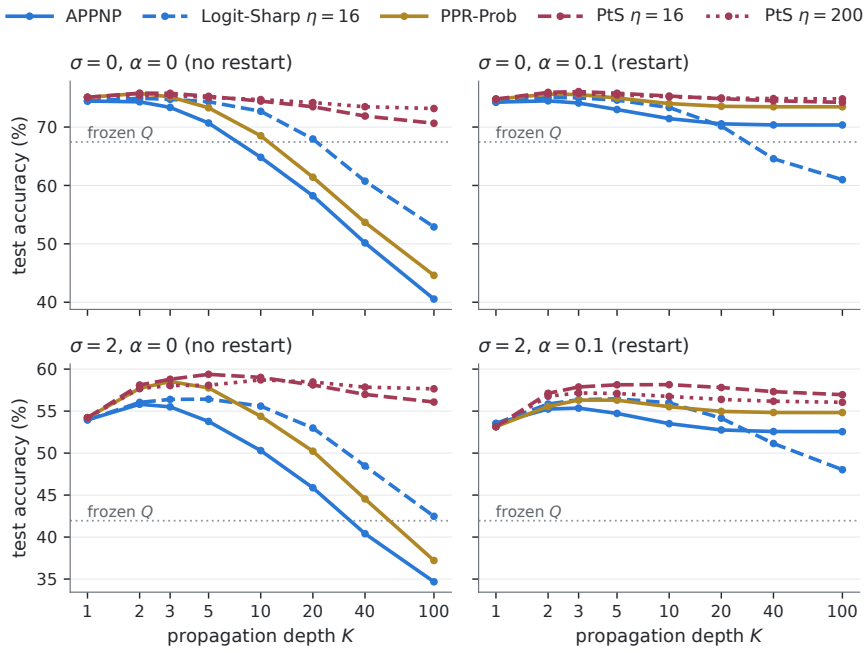

Figure 2: Mean accuracy over the nine main graphs at fixed (α, η). Columns vary the restart, rows vary the corruption severity.

4.3 CORRUPTION SEVERITY

Figure 3 shows positive mean gains for PtS over both APPNP and PPR-Prob at every tested corruption severity. With hyperparameters selected independently for each method at the evaluation severity, the gain over APPNP is 1.71 percentage points on clean inputs and 3.90 at $\sigma = 2$, while the gains over PPR-Prob are 0.57 and 1.09 points. The main comparisons select hyperparameters at the severity used for evaluation. Figure 4 examines how improvements transfer across selection and evaluation severities. The mean PtS-APPNP gain remains positive across all 25 combinations. When both methods are selected using clean validation data, PtS keeps a 2.28 point advantage at $\sigma = 2$, compared with 3.90 points under severity-matched selection.

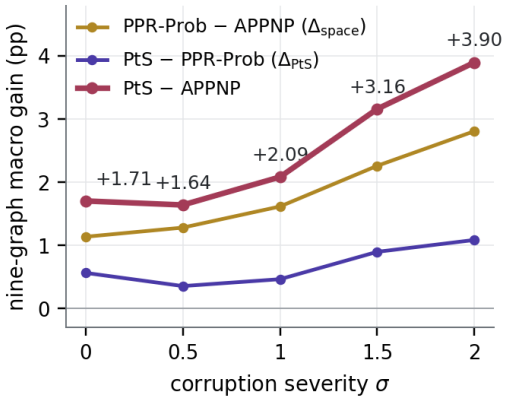

Figure 3: Mean test-accuracy gains in percentage points over the nine main homophilic graphs with a frozen MLP. The curves compare PPR-Prob with APPNP, PtS with PPR-Prob, and PtS with APPNP. Each method is independently selected at the evaluation severity.

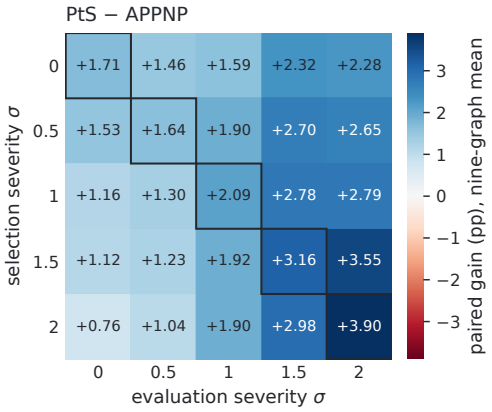

Figure 4: Mean PtS−APPNP test-accuracy gains in percentage points over the nine homophilic graphs with a frozen MLP. Rows give the severity used for hyperparameter selection and columns the evaluation severity. The first row shows clean-selection transfer.

4.4 BACKBONE

Under the same-severity selection protocol used for the main comparisons, we apply PtS to frozen GCN and GraphSAGE predictions, this still gives positive gains of 0.88 and 1.23 points at $\sigma = 2$ (Table 3). Both backbones show positive gains over APPNP on all nine main graphs at $\sigma = 2$, and on seven (GCN) and nine (GraphSAGE) on clean inputs (Appendix K). Their smaller gains compared to MLPs are consistent with graph-aware backbones profiting less from post-hoc graph propagation. The Correct & Smooth extension is evaluated separately in Appendix L.

Table 3: Mean test-accuracy gain of PtS over independently tuned APPNP across the nine main datasets, in percentage points.

Frozen backbone	$\sigma = 0$	$\sigma = 2$
MLP	+1.71	+3.90
GCN	+0.41	+0.88
GraphSAGE	+0.47	+1.23

4.5 CALIBRATION AND RUNTIME

With an MLP at $\sigma = 2$, PtS has a higher raw negative log-likelihood (NLL) and expected calibration error (ECE) than APPNP, so its uncalibrated confidences are less trustworthy. Temperature scaling fitted on validation nodes (Guo et al., 2017) reduces the mean NLL of PtS below that of APPNP without changing accuracy, although its ECE stays higher (Table 20). PtS costs between 1.3 and 54 ms per propagation step (Table 19).

PtS improves mean accuracy on clean and corrupted inputs and substantially reduces the accuracy loss from deep propagation. It keeps its advantage under clean-only selection and with graph-aware backbones. Sharpening worsens raw calibration, which temperature scaling largely mitigates.

5 Discussion

Findings Sharpening never changes a node's own predicted class, so its accuracy gain comes from subsequent propagation steps. Turning the sharpening off at PtS-selected (α, K) lowers accuracy by 5.31 points, showing that these selected propagation settings depend on sharpening (Table 10). Gains over APPNP are generally larger in higher local-homophily bins and in lower prediction-confidence quintiles (Table 18). The largest gain is in the 0.6–0.8 local-homophily bin rather than at 0.8–1. One possible explanation is that, when neighbours are confident in their predictions, propagation alone already produces a decisive and mostly correct distribution, so sharpening has less to add, while in more uncertain local regimes, averaging can leave the node with a correct but small margin and one sharpening step restores it (Lemma 1) before the next propagation. The two local-homophily bins below 0.4 have negative PtS-APPNP gains. Post-hoc refinement also recovers a substantial part of the corruption damage: with the MLP backbone the frozen prediction loses 25.5 points between $\sigma = 0$ and $\sigma = 2$, and PtS cuts that loss to 16.4 points, a 36% reduction against 27% for APPNP (Table 1). The graph-aware backbones lose less to begin with (8 and 14 points) and refinement still recovers 25% and 22% of it (Appendix K). Their smaller headroom might be due to their propagation operators acting as a form of Laplacian smoothing that reduces Dirichlet disagreement (Li et al., 2018).

Limitations The main results select hyperparameters at the severity at which they are evaluated, so they assume validation nodes that reflect test-time conditions. Figure 4 shows that the gain survives clean-only validation, but the gain shrinks. We study Gaussian corruption of node features only, leaving missing features and perturbations of the graph itself to future work. Additionally, sharpening worsens raw calibration, but this is mitigated by temperature scaling (Table 20). All nodes share a single η, and propagation favours agreement between connected nodes so that neither component can adapt to the reliability of a local neighbourhood, and misleading neighbours can reinforce errors rather than correct them. The negligible gains on the heterophilic controls and the loss against APPNP on ogbn-products (Table 1) show that the benefit is not universal. Node- or edge-adaptive parameters are therefore a natural extension, ACMP provides a related example of learned attractive and repulsive interactions (Wang et al., 2023). Finally, PtS does not minimize the full Potts energy on the graph. Propagation is normalised PPR, and we use the propagated distribution as the reference for the local sharpening step, so Proposition 1 is only a local descent guarantee at a single node. Empirically, the two PtS steps act on complementary parts of the energy. In the example on WikiCS, propagation reduces the Dirichlet term, while sharpening increases it but strongly reduces the Gini term (Figure 6). After a few iterations, both energy terms stabilize and validation accuracy reaches a plateau, while the propagation without sharpening continues to deteriorate. The energy formulation also provides a natural route for extending PtS with additional regularizers or constraints. Variational and Potts-based image segmentation methods have incorporated geometric and topological priors such as convexity, connectivity and volume constraints (Liu et al., 2020; 2022; Li et al., 2026). Similar constraints could be carried over to graphs that carry geometric or topological information that can be exploited.

Conclusion Propagation over the graph can recover accuracy from degraded features, but its standard safeguard against oversmoothing, restart, reinjects the predictions that have become unreliable. Splitting the relaxed Potts interaction into Dirichlet and Gini terms adds an extra safeguard: propagate class probabilities, then sharpen each node. PtS adds one parameter to APPNP-style propagation, needs only the frozen class distribution and the graph and costs a few milliseconds per propagation step. With an MLP backbone, it improves mean test accuracy over independently tuned APPNP by 3.90 points at $\sigma = 2$ and 1.71 on clean inputs across nine graphs, and removes most of the accuracy drop of deep propagation. Gains are smaller with graph-aware backbones and negligible on heterophilic graphs. Smoothing across neighbours and decisiveness within nodes can be treated as separate operations, and keeping nodes decisive enables deeper propagation.

AI USE STATEMENT

Generative AI tools assisted with editing and polishing the text, including checking the consistency of notation and formula definitions and identifying awkward phrasing and spelling errors. AI tools also assisted with literature search, including identifying LAME as potentially relevant prior work, the main author read and evaluated the original paper before incorporating it into the manuscript and experimental comparison. AI tools were also used for code review and for cleaning up the submitted code for readability, including removing stale experiments not included in the paper, numerical consistency checks, and generating Figures 1 and 5. They were also consulted for feedback on experiments and as discussion partners for exploratory ideas. All AI-assisted material was reviewed and checked for accuracy by the authors, who take responsibility for the final text, code, claims, and results of this work.

REPRODUCIBILITY STATEMENT

The refinement method is fully specified by (2)–(4), and derived in Appendix B; the proofs of Proposition 1 and Proposition 2 are given in Appendix D and Appendix D.2. The experimental protocol, including the backbones, corruption model, splits, and aggregation, is described in Section 3; dataset statistics are listed in Table 4, model sizes in Appendix F, search spaces in Table 6, and the selected hyperparameters in Appendix H. Baseline objectives and search settings are given in Appendix I. Code is available at: `https://anonymous.4open.science/r/Predict-then-propagate-04C0/`.

REFERENCES

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pp. 2623–2631, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362016. doi: 10.1145/3292500.3330701. URL https://doi.org/10.1145/3292500.3330701.

Andrea L. Bertozzi and Arjuna Flenner. Diffuse interface models on graphs for classification of high dimensional data. *Multiscale Modeling & Simulation*, 10(3):1090–1118, 2012. doi: 10.1137/11083109X. URL https://doi.org/10.1137/11083109X.

Malik Boudiaf, Romain Mueller, Ismail Ben Ayed, and Luca Bertinetto. Parameter-free online test-time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. URL https://openaccess.thecvf.com/content/CVPR2022/papers/Boudiaf_Parameter-Free_Online_Test-Time_Adaptation_CVPR_2022_paper.pdf.

Jeongwhan Choi, Seoyoung Hong, Noseong Park, and Sung-Bae Cho. GREAD: Graph neural reaction-diffusion networks. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 5722–5747. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/choi23a.html.

Cristina Garcia-Cardona, Ekaterina Merkurjev, Andrea L. Bertozzi, Arjuna Flenner, and Allon G. Percus. Multiclass data segmentation using diffuse interface methods on graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(8):1600–1613, 2014. doi: 10.1109/TPAMI.2014.2300478.

Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=H1gL-2A9Ym.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1321–1330. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/guo17a.html.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/5dd9db5e033da9c6fb5ba83c7a7ebea9-Paper.pdf.

Xiaoxin He, Xavier Bresson, Thomas Laurent, Adam Perold, Yann LeCun, and Bryan Hooi. Harnessing explanations: LLM-to-LM interpreter for enhanced text-attributed graph representation learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=RXFVcynVe1.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 22118–22133. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/fb60d411a5c5b72b2e7d3527cfc84fd0-Paper.pdf.

Qian Huang, Horace He, Abhay Singh, Ser-Nam Lim, and Austin Benson. Combining label propagation and simple models out-performs graph neural networks. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=8E1-f3VhX1o.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pp. 3538–3545, 2018. URL https://doi.org/10.1609/aaai.v32i1.11604.

Wenxiao Li, Xue-Cheng Tai, and Jun Liu. Topology-guaranteed image segmentation: Enforcing connectivity, genus, and width constraints. *SIAM Journal on Imaging Sciences*, 19(2):1137–1173, 2026. doi: 10.1137/25M1765870. URL https://doi.org/10.1137/25M1765870.

Hao Liu, Jun Liu, Raymond H. Chan, and Xue-Cheng Tai. Double-well net for image segmentation. *Multiscale Modeling & Simulation*, 22(4):1449–1477, 2024. doi: 10.1137/24M1632103. URL https://doi.org/10.1137/24M1632103.

Jun Liu, Xue-Cheng Tai, and Shousheng Luo. Convex shape prior for deep neural convolution network based eye fundus images segmentation, 2020. URL https://arxiv.org/abs/2005.07476.

Jun Liu, Xiangyue Wang, and Xue-Cheng Tai. Deep convolutional neural networks with spatial regularization, volume and star-shape priors for image segmentation. *Journal of Mathematical Imaging and Vision*, 64(6):625–645, 2022. doi: 10.1007/s10851-022-01087-x.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach, 2019. URL https://arxiv.org/abs/1907.11692.

Ekaterina Merkurjev, Tijana Kostić, and Andrea L. Bertozzi. An MBO scheme on graphs for classification and image processing. *SIAM Journal on Imaging Sciences*, 6(4):1903–1930, 2013. doi: 10.1137/120886935. URL https://doi.org/10.1137/120886935.

Péter Mernyei and Cătălina Cangea. Wiki-CS: A Wikipedia-based benchmark for graph neural networks, 2022. URL https://arxiv.org/abs/2007.02901.

Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1ldO2EFPr.

Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. A critical look at the evaluation of GNNs under heterophily: Are we really making progress? In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=tJbbQfw-5wv.

R. B. Potts. Some generalized order-disorder transformations. *Mathematical Proceedings of the Cambridge Philosophical Society*, 48(1):106–109, 1952. doi: 10.1017/S0305004100027419.

Xue-Cheng Tai, Hao Liu, and Raymond Chan. PottsMGNet: A mathematical explanation of encoder-decoder based neural networks. *SIAM Journal on Imaging Sciences*, 17(1):540–594, 2024. doi: 10.1137/23M1586355. URL https://doi.org/10.1137/23M1586355.

Yuelin Wang, Kai Yi, Xinliang Liu, Yu Guang Wang, and Shi Jin. ACMP: Allen-Cahn message passing with attractive and repulsive forces for graph neural networks. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=4fZc_79Lrqs.

Hao Yan, Chaozhuo Li, Ruosong Long, Chao Yan, Jianan Zhao, Wenwen Zhuang, Jun Yin, Peiyan Zhang, Weihao Han, Hao Sun, Weiwei Deng, Qi Zhang, Lichao Sun, Xing Xie, and Senzhang Wang. A comprehensive study on text-attributed graphs: Benchmarking and rethinking. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 17238–17264. Curran Associates, Inc., 2023. doi: 10.52202/075280-0755. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/37d00f567a18b478065f1a91b95622a0-Paper-Datasets_and_Benchmarks.pdf.

Yiming Yang, Jun Liu, and Wei Wan. Regularizing softmax with graph similarity for enhanced node classification in semisupervised settings. *International Journal of Intelligent Systems*, 2025(1): 8861477, 2025. doi: 10.1155/int/8861477. URL https://onlinelibrary.wiley.com/doi/abs/10.1155/int/8861477.

A. L. Yuille and Anand Rangarajan. The concave-convex procedure. *Neural Computation*, 15(4): 915–936, 2003. doi: 10.1162/08997660360581958.

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*, pp. 1215–1226, 2021. doi: 10.1145/3442381.3449953. URL https://doi.org/10.1145/3442381.3449953.

APPENDIX OVERVIEW

This appendix provides the theoretical derivations, experimental details, and additional analyses supporting the main paper.

Theory and derivations (Appendices A to D). We derive the graph energy underlying PtS, its decomposition into Dirichlet and Gini terms, and the categorical sharpening step. We then connect the construction to Potts and LAME and analyze properties of sharpening and repeated propagation, including class-order preservation, local Gini descent, propagation collapse, and the effect of sharpening at increasing depth.

Experimental setup (Appendices E to H). We provide full details on the datasets, frozen base models, hyperparameter search spaces, and selected hyperparameters used throughout the experiments.

Additional comparisons (Appendices I to M). We compare PtS with the external post-hoc baselines LAME-Graph and Graph-TV, evaluate stronger GCN and GraphSAGE backbones, study how the sharpening step can be incorporated into Correct & Smooth, and report results across the full Gaussian corruption ladder.

Ablations and mechanism (Appendices N to Q). We examine row-mass preservation, propagation depth, the evolution of the Dirichlet and Gini components, performance across nodes with different local homophily, degree, and prediction confidence

Calibration and computational cost (Appendices R to S). Finally, we report runtime and computational overhead and evaluate calibration before and after temperature scaling.

A FROM THE CONTINUOUS POTTS ENERGY TO THE GRAPH ENERGY

We start from an entropy-regularized Potts formulation and transfer it from a continuous domain to a graph. The derivations show how predictions from a frozen classifier become a KL term, and how the Potts interaction decomposes into a Dirichlet and Gini term. This decomposition motivates combining propagation between nodes with local sharpening of class distributions.

Let

$$\Delta^{C-1} = \left\{ u \in \mathbb{R}^C : u_c \geq 0,\ \sum_{c=1}^{C} u_c = 1 \right\}$$

denote the probability simplex. In the multiphase formulation, an assignment field $v(x) \in \Delta^{C-1}$ associates each position $x \in \Omega$ with a soft class distribution.

Our motivation comes from the entropy-regularized multiphase formulation in Soft Threshold Dynamics (Liu et al., 2022) and the related two-phase formulation in PottsMGNet (Tai et al., 2024). With the perimeter-scaling constants absorbed into λ, we write the continuous energy as

$$\min_{v:\Omega\to\Delta^{C-1}} \Bigg[\int_\Omega v(x)^\top g(x)\,dx + \epsilon \int_\Omega \sum_{c=1}^{C} v_c(x) \ln v_c(x)\,dx + \frac{\lambda}{2} \int_\Omega \int_\Omega G_\rho(x-y)\left(1 - v(x)^\top v(y)\right) dy\,dx \Bigg] \tag{6}$$

Here $\epsilon > 0$, $\lambda \geq 0$, and G_ρ is a Gaussian kernel. The last term is an approximation of the Potts boundary penalty. The first term assigns a cost $g(x)$ to each class at each position, and the entropy term regularizes the assignments within the simplex, with the convention $0 \log 0 = 0$.

A.1 GRAPH ANALOGUE

We replace spatial positions with graph nodes, integrals with sums, and the Gaussian kernel with a symmetric nonnegative affinity matrix.

$$W \in \mathbb{R}^{N\times N} \qquad W = W^\top \qquad W_{ij} \geq 0.$$

The graph analogue of Equation (6) is

$$\min_{\{u_i\in\Delta^{C-1}\}_{i=1}^N} \left[\sum_i u_i^\top g_i + \epsilon \sum_{i,c} u_{ic} \ln u_{ic} + \frac{\lambda}{2} \sum_{i,j} W_{ij} \left(1 - u_i^\top u_j\right) \right]. \tag{7}$$

For one-hot assignments $u_i = e_{y_i}$

$$1 - u_i^\top u_j = \mathbf{1}[y_i \neq y_j].$$

and the interaction reduces to a weighted Potts regularization that penalizes disagreement between connected nodes.

We obtain the assignment cost from the frozen classifier. Let $q_i \in \Delta^{C-1}$ denote row i of Q, the predicted class distribution at node i, with $q_{ic} > 0$ for every class, and choose $g_{ic} = -\epsilon \log q_{ic}$.

Then the first two terms in (7) become

$$\begin{aligned} \sum_i u_i^\top g_i + \epsilon \sum_{i,c} u_{ic} \log u_{ic} &= \epsilon \sum_{i,c} u_{ic} \log \frac{u_{ic}}{q_{ic}} \\ &= \epsilon \sum_i \mathrm{KL}(u_i \| q_i). \end{aligned}$$

After dividing by ϵ and defining $\gamma = \lambda/\epsilon$, we get

$$E_\gamma(U; Q) = \sum_i \mathrm{KL}(u_i \| q_i) + \frac{\gamma}{2} \sum_{i,j} W_{ij} \left(1 - u_i^\top u_j\right). \tag{8}$$

The energy therefore combines anchoring to the frozen classifier's predictions with a relaxed Potts interaction between the nodes.

Gini + Dirichlet For a symmetric W, we define $d_i = \sum_j W_{ij}$. Using the polarization identity

$$\|u_i - u_j\|^2 = \|u_i\|^2 + \|u_j\|^2 - 2u_i^\top u_j$$

and the symmetry of W, we get

$$\frac{1}{2} \sum_{i,j} W_{ij} \left(1 - u_i^\top u_j\right) = \underbrace{\frac{1}{4} \sum_{i,j} W_{ij} \|u_i - u_j\|^2}_{E_{\mathrm{D}}(U)} + \underbrace{\frac{1}{2} \sum_i d_i \left(1 - \|u_i\|^2\right)}_{E_{\mathrm{G}}(U)}.$$

The Dirichlet term E_{D} penalizes differences between the class distributions of connected nodes, and is zero when the class distributions are identical on all connected nodes. The Gini term E_{G} penalizes distributions that spread mass over several classes, and is zero when all node distributions are one-hot.

The two terms describe different properties. Two neighbouring nodes can carry identical but uncertain class distributions: their connecting edge is then not penalized by E_{D}, while both nodes still contribute to E_{G}.

B DERIVATION OF THE SHARPENING STEP

Here, we derive the sharpening step from a local energy that combines the KL distance to the propagated distribution with a Gini term. The derivation shows why one fixed-point step yields the reaction used in PtS. When $\eta = 0$, the reaction becomes an identity. The decomposition above gives

$$E_\gamma(U; Q) = \sum_i \mathrm{KL}(u_i \| q_i) + \gamma E_{\mathrm{D}}(U) + \gamma E_{\mathrm{G}}(U).$$

which motivates treating graph smoothing and local categorical sharpening as separate operations.

PtS uses the probability-space propagation defined in Section 2.1,

$$U^{(0)} = Q, \qquad \widetilde{U}^{(k+1)} = \alpha Q + (1-\alpha) S U^{(k)}.$$

For a positive propagated row $h_i = \widetilde{U}_i^{(k+1)}$, let $m_i = \mathbf{1}^\top h_i$ and $p_i = h_i / m_i$. Since the rows of S do not necessarily sum to one, this normalization is what ensures $p_i \in \Delta^{C-1}$.

We hold p fixed and use the assignment cost $g_c = -\log p_c$. Combining this cost with the entropy regularization gives

$$E(u; p) = \sum_c u_c \log \frac{u_c}{p_c} + \frac{\eta}{2}\big(1 - \|u\|^2\big), \qquad u \in \Delta^{C-1}.$$

The first term keeps the update close to the propagated distribution p and is zero when $u = p$, while the second term penalizes indecision and is zero when u is one-hot.

We use one global sharpening strength $\eta \geq 0$ for all nodes. This replaces the degree-dependent weighting in the Gini term. To handle the constraint $\sum_c u_c = 1$, we introduce a Lagrange multiplier ν:

$$L(u, \nu; p) = E(u; p) + \nu \left(\sum_c u_c - 1 \right).$$

Stationarity in the simplex gives

$$\frac{\partial \mathcal{L}}{\partial u_c} = \log \frac{u_c}{p_c} + 1 - \eta u_c + \nu = 0.$$

With p held fixed, the corresponding constrained gradient flow is

$$\dot{u}_c(t) = -\log \frac{u_c(t)}{p_c} - 1 + \eta u_c(t) - \nu(t), \qquad u(0) = p,$$

where $\dot{u}_c(t)$ denotes the time derivative of $u_c(t)$. The multiplier $\nu(t)$ is chosen so that $\sum_{c=1}^{C} \dot{u}_c(t) = 0$. Since $\sum_{c=1}^{C} p_c = 1$, this preserves $\sum_{c=1}^{C} u_c(t) = 1$.

An implicit Euler step of size $\tau > 0$ gives

$$\frac{u_c - p_c}{\tau} + \log \frac{u_c}{p_c} + 1 - \eta u_c + \nu = 0.$$

Since the new unknown distribution u appears both in the time-step term and in the gradient, we use a fixed-point iteration similar to the one in PottsMGNet (Tai et al., 2024, Section 5.2). Starting from $v^{(0)} = p$, we evaluate the non-logarithmic terms at v^k and solve for the next iterate:

$$v^{(k+1)} = \mathrm{softmax}\left(\log p + \eta v^{(k)} - \frac{v^{(k)} - p}{\tau} \right).$$

We use one inner iteration. Since $v^{(0)} = p$, the last term is zero and the update becomes

$$p_c^* = \frac{p_c \exp(\eta p_c)}{\sum_b p_b \exp(\eta p_b)}, \qquad p^* = \mathrm{softmax}(\log p + \eta p).$$

Because the fixed-point iteration is initialized at $v^{(0)} = p$, the time-step term vanishes in the first iteration, so the step size τ does not appear in the implemented reaction. We use this single iteration rather than solving to convergence, and restore the row mass afterwards:

$$R_\eta(h_i) = m_i \, \mathrm{softmax}(\log p_i + \eta p_i), \qquad \mathbf{1}^\top R_\eta(h_i) = m_i.$$

The restored mass determines the row's total contribution in the next propagation step. With $\eta = 0$ we get $R_0(h_i) = h_i$, so that PtS reduces to PPR-Prob with the same (α, K). The final class distributions are obtained by row-normalizing $U^{(K)}$.

C Relation to Potts and LAME

This part shows the connection between the graph energy, the coupled fixed-point update, and LAME-Graph. In the coupled update, the neighbourhood contribution is in the softmax, while the original predictions are kept as the reference. PtS instead uses separate steps for propagation and sharpening, with the propagated distribution as the reference in the local sharpening step.

For a symmetric W, every stationary point of the energy in Equation (8) must satisfy

$$u_i = \mathrm{softmax}\left(\log q_i + \gamma (WU)_i\right). \tag{9}$$

Here the neighbourhood interaction sits inside the softmax, while q_i is kept as the reference. PtS instead propagates first and then uses the propagated distribution p_i as the anchor for the reaction. The Potts decomposition motivates this construction, but it does not imply that the PtS iteration reduces the global energy. We evaluate the coupled iteration (9) with $B = \gamma S$ under the same protocol as everything else, reported as LAME-Graph in Tables 8 and 9.

Relationship to LAME Once the constant term $\frac{\gamma}{2}\sum_{i,j} W_{ij}$ is removed from Equation (8), the energy has the same KL-plus-affinity form as the objective of LAME (Boudiaf et al., 2022), whose update can be written as

$$u_i^{(t+1)} = \mathrm{softmax}\big(\log q_i + (Bu^{(t)})_i\big).$$

In the original LAME formulation the affinity matrix B is constructed from pretrained feature representations, using a k-nearest-neighbour affinity as well as linear and Gaussian kernels. Setting $B = \gamma W$ recovers the iteration in Equation (9), which connects the coupled update to LAME with a graph-derived affinity; this is the LAME-Graph baseline. PtS uses a different update: propagation combines the current state with the original predictions, and a separate mass-preserving reaction then sharpens the propagated distribution p_i.

D PROPERTIES OF SHARPENING

Here we examine both the properties of a single sharpening step and its behaviour under repeated propagation. We show that sharpening keeps the class ordering and decreases local Gini. The propagation analysis shows why the nodes' predictions can collapse without restart, while the example in Figure 5 shows that even restart is not always sufficient to keep the correct predictions correct.

D.1 PURE PROPAGATION COLLAPSES

Assume $\alpha = \eta = 0$ and that the graph is connected and undirected, with a self-loop at every node. Let $Q > 0$ have row sums equal to one. Propagation without restart and without sharpening is then

$$U^{(K)} = S^K Q. \tag{10}$$

From Oono & Suzuki (2020, Proposition 1), S has a single eigenvalue 1 while all other eigenvalues have absolute value less than 1, the corresponding unit eigenvector $\phi \propto \widetilde{D}^{1/2}\mathbf{1}$ is strictly positive. Write $v = \phi$, with v_j denoting its entry at node j.

Since S is real and symmetric, we can decompose it into its eigenvectors, where N is the number of nodes and $\lambda_1 = 1$,

$$S^K = \lambda_1^K \phi\phi^\top + \sum_{r=2}^{N} \lambda_r^K \phi_r \phi_r^\top.$$

Let $K \to \infty$. Since $|\lambda_r| < 1$ for $r \geq 2$ then $\lambda_r^K \to 0$ leaving us with

$$S^K \to \phi\phi^\top \tag{11}$$

Let $p_i^{(K)}$ be the final normalised prediction at node i.

$$\begin{aligned}
p_i^{(K)} &= \frac{U_i^{(K)}}{\sum_c U_{ic}^{(K)}} && \textit{By definition} \\
&= \frac{(S^K Q)_i}{\sum_c (S^K Q)_{ic}} && \textit{By (10)} \\
&\to \frac{(\phi\phi^\top Q)_i}{\sum_c (\phi\phi^\top Q)_{ic}} && \textit{By (11)} \\
&= \frac{\phi_i \sum_j v_j Q_j}{\phi_i \sum_j v_j \underbrace{\sum_c Q_{jc}}_{=1}}
\end{aligned}$$

Which result in

$$\lim_{K\to\infty} p_i^{(K)} = \frac{\sum_j v_j Q_j}{\sum_j v_j}.$$

The limit does not depend on i: it is a weighted average of the initial predictions, so every node converges to the same normalised class distribution. Pure probability propagation therefore collapses, as observed empirically in Figure 2.

The same argument applies to APPNP at $\alpha = 0$, where the propagated state is the logit matrix Z rather than Q. Here $S^K Z \to vv^\top Z$, so node i receives the logit row $\phi_i\, c$ with the common vector $c = \sum_j \phi_j Z_j$. Since $\phi_i > 0$, the softmax of $\phi_i c$ has the same maximizing class at every node: the predicted distributions need not coincide, but the predicted *labels* all collapse onto a single class.

With $\alpha \in (0, 1]$ With restart, the iteration converges to the PPNP solution instead (Gasteiger et al., 2019),

$$U^\infty = \alpha(I_n - (1-\alpha)S)^{-1} Q,$$

which depends on i through Q, so restart prevents collapse. It does so by reinjecting Q at every step, which is what makes the quality of Q decisive, Proposition 2 shows that this is not always enough.

D.2 RESTART ALONE DOES NOT PRESERVE A MINORITY COMMUNITY

Proposition 2. *Let G consist of two 9-node cliques $\mathcal{A}$ and $\mathcal{B}$ joined by a perfect matching, so that with added self-loops every node draws 90% of its propagation weight from its own community. Let the true class be 1 on $\mathcal{A}$ and 2 on $\mathcal{B}$, with frozen predictions $Q_i = (0.9, 0.1)$ for $i \in \mathcal{A}$ and $Q_j = (0.4, 0.6)$ for $j \in \mathcal{B}$, and let $\alpha = 0.1$. Then APPNP and PPR-Prob misclassify every node of $\mathcal{B}$ for every depth $K \geq 3$, while PtS with $\eta = 4$ classifies every node correctly at every depth.*

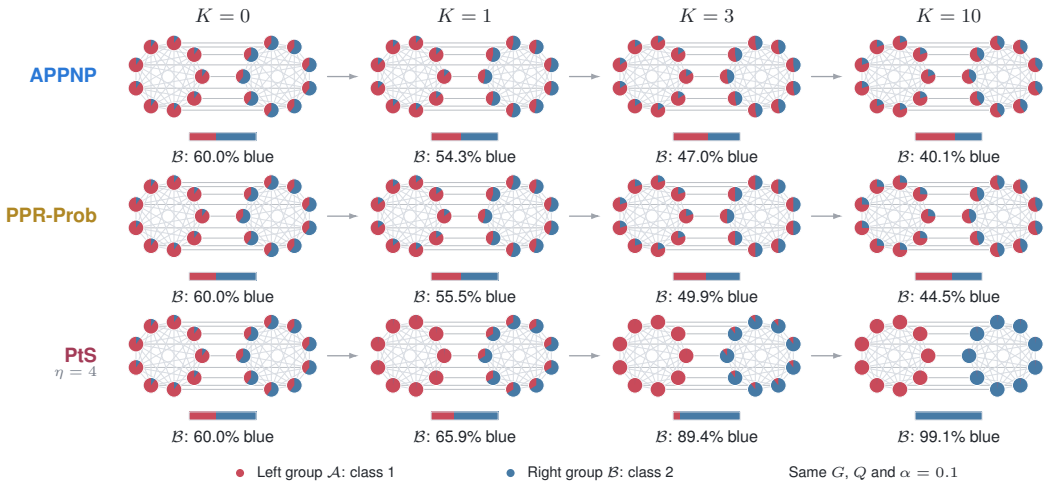

Figure 5: Class probabilities with increasing depth. All methods use the same graph, initial predictions, and $\alpha = 0.1$, PtS uses $\eta = 4$. The numbers give the correct-class probability in group $\mathcal{B}$. Self-loops are included in the updates but omitted from the drawing.

Each node is joined to one distinct node in the other group, so after adding self-loops every node has ten connections: nine within its community (eight clique neighbours and itself) and one across. All degrees are equal, so $\widetilde{D} = 10I$ and $S = \widetilde{A}/10$ is both symmetric and row-stochastic, with weight 0.9 inside the community and 0.1 across it. Propagation therefore preserves unit row sums, and so does the reaction, so all three iterations stay on the simplex. By symmetry, all nodes of a community carry the same state at every depth, and it suffices to track one node per community.

PPR-Prob Let a_k and b_k be the class-1 probabilities in $\mathcal{A}$ and $\mathcal{B}$. The update (1) gives

$$a_{k+1} = 0.09 + 0.9\,(0.9a_k + 0.1b_k), \qquad b_{k+1} = 0.04 + 0.9\,(0.1a_k + 0.9b_k).$$

The sum $\Sigma_k = a_k + b_k$ satisfies $\Sigma_{k+1} = 0.13 + 0.9\Sigma_k$ with $\Sigma_0 = 1.3$, hence $\Sigma_k = 1.3$ for all k. The difference $\Delta_k = a_k - b_k$ satisfies $\Delta_{k+1} = 0.05 + 0.72\,\Delta_k$ with $\Delta_0 = 0.5$, hence

$$\Delta_k = \tfrac{5}{28} + \left(0.5 - \tfrac{5}{28}\right) 0.72^{\,k},$$

which decreases monotonically to $5/28 \approx 0.179$. A node of $\mathcal{B}$ is misclassified exactly when $b_k = (1.3 - \Delta_k)/2 > 1/2$, that is when $\Delta_k < 0.3$. Since $\Delta_2 = 0.3452$ and $\Delta_3 = 0.29854$, the nodes of $\mathcal{B}$ are classified correctly for $K \leq 2$ and misclassified for every $K \geq 3$.

APPNP With two classes, the softmax depends only on the logit difference, so let x_k and y_k be the class-1 minus class-2 logit differences in $\mathcal{A}$ and $\mathcal{B}$, starting from $x_0 = \log 9$ and $y_0 = \log \frac{2}{3}$. APPNP applies the same affine recursion to the logits, so $\Sigma_k = x_k + y_k$ satisfies $\Sigma_{k+1} = 0.1\Sigma_0 + 0.9\Sigma_k$ and stays at $\Sigma_0 = \log 6$, while $\Delta_k = x_k - y_k$ satisfies $\Delta_{k+1} = 0.1\Delta_0 + 0.72\Delta_k$ with $\Delta_0 = \log 13.5$ and decreases monotonically to $0.1\Delta_0/0.28 \approx 0.930$. A node of $\mathcal{B}$ is misclassified exactly when $y_k = (\log 6 - \Delta_k)/2 > 0$, that is when $\Delta_k < \log 6 \approx 1.7918$. Since $\Delta_2 \approx 1.7969$ and $\Delta_3 \approx 1.5541$, the nodes of $\mathcal{B}$ are again classified correctly for $K \leq 2$ and misclassified for every $K \geq 3$.

PtS We claim that for $\eta = 4$ every node keeps at least probability 0.6 on its correct class at every depth. This holds at $k = 0$, where the correct-class probabilities are 0.9 on $\mathcal{A}$ and 0.6 on $\mathcal{B}$. Assume

it holds at step k and consider a node i with correct class y. Since the rows are distributions, the propagated value satisfies

$$\widetilde{U}_{iy}^{(k+1)} = 0.1\, Q_{iy} + 0.9 \sum_j S_{ij} U_{jy}^{(k)} \;\geq\; 0.1 \cdot 0.6 + 0.9\,(0.9 \cdot 0.6 + 0.1 \cdot 0) = 0.546,$$

using $Q_{iy} \geq 0.6$, the induction hypothesis inside the community, and non-negativity across it. In the two-class case the sharpening map satisfies

$$\frac{r_\eta(p)_y}{1 - r_\eta(p)_y} = \frac{p_y}{1 - p_y}\, e^{\eta(2p_y - 1)},$$

which is increasing in p_y, so $\widetilde{U}_{iy}^{(k+1)} \geq 0.546$ implies

$$U_{iy}^{(k+1)} \;\geq\; \frac{0.546\, e^{4(0.546)}}{0.546\, e^{4(0.546)} + 0.454\, e^{4(0.454)}} = 0.6347\ldots > 0.6.$$

The invariant is restored, so by induction every node keeps more than probability 0.6 on its correct class, and hence is classified correctly, at every depth. □

D.3 Sharpening preserves the class ordering

From the update, we define

$$p_c^* = \frac{p_c e^{\eta p_c}}{\sum_b p_b e^{\eta p_b}}, \tag{12}$$

where the entries of p^* are positive and sum to one. For any two classes c and b,

$$\frac{p_c^*}{p_b^*} = \frac{p_c}{p_b}\, e^{\eta(p_c - p_b)}. \tag{13}$$

Since $\eta \geq 0$, the ratio is at least p_c/p_b whenever $p_c \geq p_b$, so sharpening reinforces a node's existing class preference and in particular leaves its predicted class unchanged.

The same identity quantifies how much confidence a single step can restore after propagation has eroded it.

Lemma 1 (Recovering a confidence level)**.** *Let $p \in \Delta^{C-1}$ with $p > 0$ have a unique leading class c and margin $\mu = p_c - \max_{b \neq c} p_b > 0$. Then*

$$p_c^* \;\geq\; \frac{1}{1 + (C-1)e^{-\eta\mu}},$$

so for any target confidence $\beta \in (0,1)$ we have $p_c^* \geq \beta$ as soon as $\eta \;\geq\; \mu^{-1} \log\big((C-1)\beta/(1-\beta)\big)$. In particular $p_c^* \to 1$ as $\eta \to \infty$.

Proof. For $b \neq c$, (13) gives $p_b^*/p_c^* = (p_b/p_c)e^{-\eta(p_c - p_b)} \leq e^{-\eta\mu}$, since $p_b \leq p_c$ and $p_c - p_b \geq \mu$. Summing over the $C-1$ competing classes,

$$\frac{1}{p_c^*} = 1 + \sum_{b \neq c} \frac{p_b^*}{p_c^*} \leq 1 + (C-1)e^{-\eta\mu},$$

which is the stated bound. It is at least β precisely when $(C-1)e^{-\eta\mu} \leq (1-\beta)/\beta$, that is when $\eta\mu \geq \log\big((C-1)\beta/(1-\beta)\big)$. □

A node whose leading class survives propagation with margin μ can therefore be returned to any confidence level by a large enough η, which is the mechanism behind Proposition 2. The bound also shows the cost: the same η is applied to every node, including nodes whose leading class is wrong.

D.4 Proof of Proposition 1

Fix $p \in \Delta^{C-1}$ with $p > 0$ and $\eta \geq 0$, and recall the local energy (5),

$$E_\eta(u; p) = \mathrm{KL}(u\|p) + \frac{\eta}{2}\big(1 - \|u\|^2\big), \qquad u \in \Delta^{C-1}.$$

The KL term is convex in u and the Gini term is concave, so E_η is a difference of convex functions. Concavity of $u \mapsto -\frac{\eta}{2}\|u\|^2$ gives the linear upper bound

$$-\frac{\eta}{2}\|u\|^2 \;\leq\; -\frac{\eta}{2}\|p\|^2 - \eta\, p^\top (u - p),$$

with equality at $u = p$. Adding $\mathrm{KL}(u\|p) + \frac{\eta}{2}$ to both sides defines a majorizer

$$M(u) = \mathrm{KL}(u\|p) - \eta\, p^\top u + \text{const}, \qquad E_\eta(u; p) \leq M(u), \qquad E_\eta(p; p) = M(p).$$

By the Gibbs variational principle, M is minimized over the simplex at $u = \mathrm{softmax}(\log p + \eta p) = r_\eta(p) = p^*$, which is exactly the sharpening step (3). Therefore

$$E_\eta(p^*; p) \;\leq\; M(p^*) \;\leq\; M(p) \;=\; E_\eta(p; p),$$

which proves the descent claim. Writing it out, $\mathrm{KL}(p^*\|p) + \frac{\eta}{2}(1 - \|p^*\|^2) \leq \frac{\eta}{2}(1 - \|p\|^2)$, that is

$$\frac{\eta}{2}\big(\|p^*\|^2 - \|p\|^2\big) \;\geq\; \mathrm{KL}(p^*\|p) \;\geq\; 0,$$

so for $\eta > 0$ the Gini impurity $1 - \|p\|^2$ does not increase, and it strictly decreases unless $p^* = p$. The ordering claim is Appendix D.3. The same majorization argument applied at an arbitrary iterate v instead of p shows that the full fixed-point iteration $v \mapsto \mathrm{softmax}(\log p + \eta v)$ is a concave–convex procedure, so every inner iteration, not only the first, decreases $E_\eta(\cdot\,; p)$. □

E DATASETS

Table 4 summarizes the nine graphs in the main board and the heterophilic control datasets. The datasets vary in nodes, edges, homophily, and classes. The control datasets are reported separately to show how the methods act when neighbouring nodes often belong to separate classes.

Table 4: Dataset statistics. Edges are the reported undirected-pair counts, and h denotes edge homophily.

Dataset	Nodes	Edges	Features	Classes	h
WikiCS	11,701	215,603	300	10	0.65
Cora-TAPE	2,708	5,278	768	7	0.81
PubMed-TAPE	19,717	44,324	768	3	0.80
TAPE-Arxiv23	46,198	38,863	300	40	0.64
ogbn-arxiv	169,343	1,157,799	128	40	0.65
ogbn-products	2,449,029	61,859,012	100	47	0.81
Ele-Photo	48,362	436,891	768	12	0.74
Ele-Computers	87,229	628,274	768	10	0.82
Books-History	41,551	251,590	768	12	0.64
Roman-Empire	22,662	32,927	300	18	0.05
Amazon-Ratings	24,492	93,050	300	5	0.38

F FROZEN BACKBONES

Table 5 shows the architecture of the frozen base models that are used to produce the predictions. The MLP uses node features alone, while GCN and GraphSAGE use the graph structure as well. PtS is applied after these models have been trained and does not add any new trainable parameters to the base models. The MLP has architecture $d \to 256 \to 256 \to C$, with BatchNorm, ReLU, and dropout (0.5) after each hidden linear layer, and is trained with Adam (learning rate 0.01, weight decay 5×10^{-4}) for at most 500 epochs with early-stopping patience 100.

Table 5: Number of parameters of each backbone, with the feature dimension D and the number of classes C. The last column gives the depth and width of the graph backbones.

Dataset	D	C	MLP	GCN	GraphSAGE	Conv. layers $\times$ width
WikiCS	300	10	146,442	80,138	159,498	2×256
Cora-TAPE	768	7	265,479	199,175	397,575	2×256
PubMed-TAPE	768	3	264,451	198,147	395,523	2×256
TAPE-Arxiv23	300	40	154,152	87,848	174,888	2×256
ogbn-arxiv	128	40	110,120	110,120	218,664	3×256
ogbn-products	100	47	104,751	36,015	71,215	3×128
Ele-Photo	768	12	266,764	200,460	400,140	2×256
Ele-Computers	768	10	266,250	199,946	399,114	2×256
Books-History	768	12	266,764	200,460	400,140	2×256
Roman-Empire	300	18	148,498	82,194	163,602	2×256
Amazon-Ratings	300	5	145,157	78,853	156,933	2×256

G HYPERPARAMETER SEARCH SPACES

Table 6 shows the search spaces and budget for the propagation methods and variants of correct and smooth. We select hyperparameters using validation accuracy, with a separate Optuna trial per method. APPNP and PPR-Prob choose the restart weight α and number of propagation steps K, while PtS also searches over sharpening strength η. The search space also includes $\eta = 0$, so validation can choose to turn off the reaction.

Table 6: Search spaces and maximum numbers of hyperparameter evaluations per unit and severity. All methods use Optuna TPE except Graph-TV, which uses a grid search. The PtS search includes the exact sharpening off setting $\eta = 0$ alongside the logarithmic range. For hyperparameter tuning on ogbn-products, the upper limits are reduced to $K = 50$ and $T = 50$.

Method	Search	Budget
APPNP / PPR	$\alpha \in [0, 1]$, $K \in \{1, \ldots, 100\}$	250
PtS	Same, $\eta = 0$ or $\log_{10} \eta \in [-2, 2.408]$	250
LAME-Graph	$T \in \{1, \ldots, 100\}$, $\gamma = 0$ or $\log_{10} \gamma \in [-2, 2.408]$	250
Graph-TV	$\lambda = 0$ or $\log_{10} \lambda \in \{-3, \ldots, 3\}$ (25 log-spaced values), $\varepsilon = 1$ fixed	≤ 26
C&S	$\alpha_c, \alpha_s \in [0, 1]$	250
C&S-PtS	Same, plus the PtS reaction search	250

H SELECTED HYPERPARAMETERS (MLP)

Table 7: Median selected values over the units of each (graph, severity), where K is the number of propagation steps, α the restart probability, η the sharpening strength, and "off" the percentage of units for which the search selected the exact sharpening off setting $\eta = 0$.

Dataset	σ	K_{APPNP}	K_{PPR}	K_{PtS}	α_{APPNP}	α_{PtS}	η_{PtS}	off (%)
WikiCS	0	2	2	3	0.11	0.09	4.11	7
	0.5	2	2	4	0.08	0.06	6.40	8
	1	2	3	4	0.04	0.04	6.96	3
	1.5	2	2	6	0.03	0.03	12.86	0
	2	2	3	8	0.02	0.00	34.68	0
Cora-TAPE	0	3	6	54	0.06	0.13	2.24	0
	0.5	3	6	38	0.05	0.12	1.78	6
	1	4	6	34	0.03	0.10	1.42	7
	1.5	4	7	34	0.03	0.08	1.35	4
	2	5	6	38	0.02	0.06	1.46	7
PubMed-TAPE	0	30	28	46	0.32	0.31	1.51	20
	0.5	26	24	52	0.24	0.24	1.33	30
	1	4	7	15	0.11	0.15	0.82	21
	1.5	4	7	22	0.05	0.13	1.13	10
	2	5	8	24	0.03	0.10	1.61	6
TAPE-Arxiv23	0	26	4	58	0.35	0.30	0.91	27
	0.5	7	8	26	0.26	0.21	0.17	39
	1	4	8	16	0.12	0.09	0.00	53
	1.5	4	10	14	0.04	0.05	0.15	32
	2	3	9	29	0.08	0.05	0.63	8
ogbn-arxiv	0	2	2	3	0.12	0.26	6.80	0
	0.5	2	2	3	0.08	0.23	15.96	0
	1	2	3	7	0.03	0.20	6.70	0
	1.5	2	2	3	0.03	0.06	6.77	0
	2	1	2	2	0.03	0.02	0.00	56
ogbn-products	0	2	2	17	0.13	0.36	3.34	0
	0.5	2	2	4	0.04	0.31	42.74	0
	1	2	1	1	0.41	0.27	0.00	56
	1.5	2	2	2	0.48	0.33	0.00	78
	2	2	2	1	0.47	0.29	0.00	67
Ele-Photo	0	1	1	1	0.02	0.03	1.70	37
	0.5	1	1	1	0.02	0.07	6.59	22
	1	1	1	2	0.01	0.03	7.80	22
	1.5	1	2	2	0.01	0.00	9.37	19
	2	1	2	3	0.02	0.00	8.73	13
Ele-Computers	0	2	3	10	0.01	0.24	6.17	0
	0.5	2	3	7	0.00	0.10	7.27	0
	1	2	3	8	0.00	0.03	11.10	0
	1.5	3	3	9	0.00	0.01	17.22	1
	2	3	4	12	0.00	0.00	23.25	0
Books-History	0	1	1	41	0.42	0.42	3.37	20
	0.5	1	1	15	0.30	0.34	3.26	13
	1	1	2	5	0.18	0.24	3.87	7
	1.5	2	3	8	0.10	0.16	3.65	7
	2	2	3	14	0.06	0.09	4.77	2
Roman-Empire	0	45	37	60	0.98	0.97	0.00	53
	0.5	42	59	53	0.98	0.92	0.12	39
	1	46	62	52	1.00	0.93	0.09	41
	1.5	56	51	50	1.00	0.90	2.49	34
	2	52	42	63	1.00	0.84	0.11	40
Amazon-Ratings	0	40	8	36	0.55	0.26	0.00	70
	0.5	54	3	4	0.40	0.17	0.00	73
	1	32	36	45	0.35	0.18	0.00	88
	1.5	51	45	56	0.29	0.21	0.00	66
	2	47	44	47	0.18	0.20	0.00	57

Across multiple graphs, hyperparameter tuning chose more propagation steps for PtS than for PPR-Prob (Table 7). The chosen sharpening strength varies across datasets and noise levels, and in some cases it is completely turned off. This suggests that sharpening is not always helpful. The depth also needs to be interpreted with the restart weight in mind, a large K can have a limited interpretation when α is close to 1. We calculate medians separately for each hyperparameter.

I EXTERNAL POST-HOC BASELINES

Both baselines refine the same frozen predictions Q on the same observed graph and evaluation units as PtS. Validation labels are used only for hyperparameter selection. The adaptations, graph operators and selection procedures are specified below.

LAME-Graph. LAME (Boudiaf et al., 2022) refines a frozen classifier's output using a KL fidelity term and an agreement term weighted by affinities computed from pretrained feature representations. We replace these affinities by the self-looped graph operator S and initialize $U^{(0)} = Q$. The update is

$$U^{(t+1)} = \mathrm{RowSoftmax}\big(\log Q + \gamma S U^{(t)}\big).$$

The original predictions therefore remain the reference throughout the iteration, while the graph interaction enters inside the softmax. This is the coupled Potts update described in Appendix C.

We select the interaction strength γ and a fixed iteration count T by validation accuracy using 250 Optuna TPE trials per unit, severity and noise draw. The search uses $T \in \{1, \dots, 100\}$ and $\log_{10} \gamma \in [-2, 2.408]$ for positive strengths, together with an explicit identity option that returns Q exactly. We run the selected number of iterations rather than using the original energy-based stopping rule.

Graph-TV. We adapt the TV-regularized softmax of Yang et al. (2025) to frozen predictions by minimizing

$$\sum_i \mathrm{KL}(u_i \| q_i) + \lambda \, \mathrm{TV}_G(U), \qquad u_i \in \Delta^{C-1}.$$

The KL term keeps predictions close to Q, while TV penalizes disagreement across the graph. We use their classwise neighbourhood TV with loopless, degree-normalised graph weights, setting the input to $\log Q$ and $\varepsilon = 1$. Setting $\lambda = 0$ returns Q.

We select λ by validation accuracy from at most 26 grid candidates (Table 6). Each dual solve uses at most 3000 iterations, and only candidates with relative primal–dual gap at most 10^{-3} are eligible for selection. Graph-TV is evaluated at $\sigma \in \{0, 2\}$ on eight graphs, excluding ogbn-products because of memory requirements.

Table 8: Comparison of post-hoc graph refinement methods at $\sigma = 2$, including the LAME-Graph and the Graph-TV methods.

Dataset	Q	APPNP	PPR-Prob	LAME-Graph	Graph-TV	PtS
WikiCS	46.36	67.97	68.98	67.80	65.10	**72.15**
Cora-TAPE	43.82	68.16	70.47	63.94	68.72	**72.30**
PubMed-TAPE	64.56	78.87	80.22	73.87	79.64	**80.73**
TAPE-Arxiv23	28.56	35.63	38.26	35.26	36.54	**38.91**
ogbn-arxiv	22.65	33.86	40.43	40.38	35.25	**40.62**
ogbn-products	22.46	**28.54**	27.24	27.47	–	27.33
Ele-Photo	44.15	55.97	61.00	59.58	57.06	**61.43**
Ele-Computers	38.05	59.94	66.85	62.45	59.64	**69.56**
Books-History	66.91	78.11	78.90	77.66	78.90	**79.13**
heterophilic controls						
Roman-Empire	19.34	19.30	19.30	19.29	**19.31**	19.30
Amazon-Ratings	34.17	36.74	**36.80**	36.73	36.73	36.78
Mean (9 main)	41.95	56.34	59.15	56.49	–	60.24
Mean (8 main, no products)	44.38	59.81	63.14	60.12	60.11	64.35

Table 9: External $Q{+}G$ baselines on clean features ($\sigma = 0$), including the LAME-Graph and the Graph-TV methods.

Dataset	Q	APPNP	PPR-Prob	LAME-Graph	Graph-TV	PtS
WikiCS	72.67	77.78	78.24	78.18	78.18	**78.42**
Cora-TAPE	60.51	76.16	78.42	76.78	76.82	**79.74**
PubMed-TAPE	81.59	84.35	84.96	85.05	85.11	**85.18**
TAPE-Arxiv23	67.59	69.53	69.68	**69.82**	69.81	69.72
ogbn-arxiv	55.95	66.10	65.96	**67.53**	65.80	66.56
ogbn-products	59.58	70.58	71.32	71.40	–	**72.17**
Ele-Photo	66.83	71.87	74.43	74.84	72.03	**74.91**
Ele-Computers	60.96	75.37	78.71	78.32	76.47	**80.11**
Books-History	81.38	82.59	82.87	82.95	**82.97**	82.88
heterophilic controls						
Roman-Empire	65.54	**65.65**	65.54	65.55	65.57	65.54
Amazon-Ratings	49.52	51.63	**52.77**	52.05	52.36	52.74
Mean (9 main)	67.45	74.93	76.06	76.10	–	76.63
Mean (8 main, no products)	68.43	75.47	76.66	76.68	75.90	**77.19**

Tables 8 and 9 shows the results at $\sigma = 2$ and $\sigma = 0$. The comparison with Graph-TV averages over 8 graphs without products, as it requires too much memory. PtS has a higher average accuracy in these comparisons, but is not best on all datasets. On clean data, LAME-Graph is better on the ogbn-arxiv, TAPE-Arxiv23, and Books-History datasets.

J Full decomposition

Table 10 gives the per-dataset values for the decomposition summarised in Table 2, at $\sigma = 2$ with the frozen MLP

Table 10: Decomposition of PtS$-$APPNP at $\sigma = 2$ into the change from moving propagation into probability space, Δ_{space} and the change from adding the reaction, Δ_{PtS}. APPNP, PPR-Prob, and PtS are tuned independently. Reaction OFF sets $\eta = 0$ at the (α, K) selected for PtS, without retuning

Dataset	APPNP	PPR-Prob	PtS	Reaction OFF	Δ_{space} PPR$-$APPNP	Δ_{PtS} PtS$-$PPR
WikiCS	67.97	68.98	72.15	53.81	+1.01 ± 1.59	+3.17 ± 0.91
Cora-TAPE	68.16	70.47	72.30	64.83	+2.31 ± 1.90	+1.83 ± 0.65
PubMed-TAPE	78.87	80.22	80.73	76.80	+1.35 ± 1.25	+0.51 ± 0.27
TAPE-Arxiv23	35.63	38.26	38.91	36.90	+2.64 ± 0.28	+0.65 ± 0.37
ogbn-arxiv	33.86	40.43	40.62	39.95	+6.57 ± 1.71	+0.18 ± 0.22
ogbn-products	28.54	27.24	27.33	27.33	-1.30 ± 0.22	+0.09 ± 0.15
Ele-Photo	55.97	61.00	61.43	59.55	+5.03 ± 0.75	+0.42 ± 0.41
Ele-Computers	59.94	66.85	69.56	58.13	+6.91 ± 0.84	+2.72 ± 0.64
Books-History	78.11	78.90	79.13	77.11	+0.79 ± 0.47	+0.23 ± 0.19
Mean (9 main)	**56.34**	**59.15**	**60.24**	**54.93**	**+2.81**	**+1.09**

Under independent validation selection and severe corruption $\sigma = 2$, moving propagation from logit space to probability space accounts for 2.81 percentage points of the mean gain, and adding sharpening for a further 1.09 (Table 10). The split between the two varies by graph: on WikiCS it is $+1.01$ and $+3.17$, while on ogbn-arxiv it is $+6.57$ and $+0.18$. In the Reaction OFF column, we keep the (α, K) selected for PtS and set η to zero without new tuning. Mean accuracy then falls to 54.93%, below APPNP itself, so the propagation configurations selected for PtS are only viable in combination with sharpening: PtS generally selects deeper propagation than PPR-Prob (Appendix H). Figure 3 shows how the mean gains change with corruption severity, with dataset-level results in Table 14. When hyperparameters are selected on clean data and evaluated at $\sigma = 2$, PtS retains mean gains of 2.28 percentage points over APPNP in the nine main graphs (Figure 4).

K STRONGER BACKBONES

We also examine whether PtS improves when frozen predictions come from base models that already use the graph structure. We apply the same method to predictions from GCN and GraphSAGE, with model sizes given in Table 5. The comparison explores how much additional improvement we can get when the base model has already exploited neighbourhood information.

K.1 GCN BACKBONE

Table 11: Accuracy (%) with the frozen GCN backbone. Δ is PtS minus APPNP in percentage points. Bold marks the highest displayed accuracy in each condition.

	Clean ($\sigma = 0$)				Noisy ($\sigma = 2$)			
Dataset	$Q(GCN)$	APPNP	PtS	Δ	$Q(GCN)$	APPNP	PtS	Δ
WikiCS	79.08	79.05	**79.40**	+0.35 ± 0.21	76.73	76.91	**77.68**	+0.77 ± 0.30
Cora-TAPE	81.96	82.37	**83.10**	+0.73 ± 0.49	76.20	78.34	**79.94**	+1.60 ± 0.61
PubMed-TAPE	85.77	85.74	**85.81**	+0.06 ± 0.05	80.41	82.27	**82.73**	+0.46 ± 0.17
TAPE-Arxiv23	**69.28**	69.26	69.26	-0.00 ± 0.02	43.43	44.04	**44.33**	+0.29 ± 0.11
ogbn-arxiv	71.83	71.90	**72.48**	+0.58 ± 0.28	62.73	64.66	**64.78**	+0.12 ± 0.33
ogbn-products	76.55	77.24	**77.73**	+0.49 ± 0.17	63.81	69.16	**71.01**	+1.86 ± 1.10
Ele-Photo	81.33	81.42	**81.87**	+0.45 ± 0.18	78.47	78.94	**79.67**	+0.73 ± 0.22
Ele-Computers	84.03	84.61	**85.60**	+0.99 ± 0.14	77.19	80.58	**82.49**	+1.91 ± 0.41
Books-History	**82.68**	82.66	82.66	-0.00 ± 0.03	80.04	80.27	**80.44**	+0.16 ± 0.12
Roman-Empire	**45.72**	45.70	45.70	+0.00 ± 0.02	**20.38**	20.36	20.35	-0.01 ± 0.02
Amazon-Ratings	**50.02**	50.00	50.00	-0.01 ± 0.05	37.22	**37.93**	37.82	-0.11 ± 0.24
Mean (9 main)	79.17	79.36	79.77	+0.41	71.00	72.80	73.67	+0.88

With a GCN backbone, performance on clean data improves incrementally on many graphs, but PtS achieves higher accuracy than APPNP on all nine main graphs at $\sigma = 2$ (Table 11) and on seven out of nine graphs on clean graphs.

K.2 GRAPHSAGE BACKBONE

Table 12: Accuracy (%) with the frozen GraphSAGE backbone. Δ is PtS minus APPNP in percentage points. Bold marks the highest displayed accuracy in each condition.

	Clean ($\sigma = 0$)				Noisy ($\sigma = 2$)			
Dataset	$Q(SAGE)$	APPNP	PtS	Δ	$Q(SAGE)$	APPNP	PtS	Δ
WikiCS	78.72	79.08	**79.41**	+0.33 ± 0.18	71.06	75.12	**76.38**	+1.27 ± 0.43
Cora-TAPE	73.54	74.76	**75.96**	+1.20 ± 1.17	68.18	70.89	**72.69**	+1.80 ± 0.95
PubMed-TAPE	84.10	84.47	**84.80**	+0.33 ± 0.18	72.26	78.63	**80.04**	+1.41 ± 0.77
TAPE-Arxiv23	70.04	70.20	**70.35**	+0.15 ± 0.09	39.67	41.59	**43.09**	+1.50 ± 0.44
ogbn-arxiv	71.85	72.17	**72.43**	+0.26 ± 0.06	50.00	55.30	**57.44**	+2.14 ± 0.94
ogbn-products	78.18	78.34	**78.37**	+0.04 ± 0.11	39.56	39.61	**39.69**	+0.07 ± 0.13
Ele-Photo	79.44	79.53	**80.16**	+0.63 ± 0.26	76.38	76.81	**77.43**	+0.62 ± 0.22
Ele-Computers	79.71	80.96	**82.23**	+1.28 ± 0.30	73.70	77.80	**79.92**	+2.12 ± 0.36
Books-History	81.86	81.85	**81.90**	+0.05 ± 0.05	79.68	80.14	**80.28**	+0.14 ± 0.14
Roman-Empire	**80.00**	79.99	79.97	-0.03 ± 0.05	**21.05**	21.01	21.01	-0.00 ± 0.05
Amazon-Ratings	55.07	55.38	**55.49**	+0.11 ± 0.15	33.29	**37.84**	37.64	-0.20 ± 0.17
Mean (9 main)	77.49	77.93	78.40	+0.47	63.39	66.21	67.44	+1.23

With GraphSAGE, PtS achieves higher accuracy than APPNP on the nine main homophilic graphs at the selected noise levels Table 12. The gain varies across backbones and increases with noise level.

For example, the improvement under noise is more prominent on ogbn-arxiv and Ele-Computers, while it is smaller on ogbn-products.

L CORRECT AND SMOOTH COMPARISON

Correct & Smooth uses known labels in its correction and smoothing stages. We evaluate sharpening within this label-assisted pipeline, leaving correction unchanged and modifying only smoothing. The C&S-PtS variant tests whether the introduced sharpening adds gain over other methods that use APPNP propagation.

Table 13: Comparison of Correct and Smooth (C&S) with C&S-PtS, where the categorical sharpening step is applied after each smoothing step. All methods use an MLP backbone. Subscripts give the corruption severity, and Gain is C&S-PtS minus C&S at $\sigma = 2$ in percentage points.

Dataset	$C\&S_0$	$C\&S\text{-}PtS_0$	$C\&S_2$	$C\&S\text{-}PtS_2$	Gain (pp)
WikiCS	76.90	78.82	70.06	75.43	+5.38 ± 0.85
Cora-TAPE	87.38	87.23	86.18	86.00	-0.17 ± 0.20
PubMed-TAPE	86.97	86.98	84.44	84.44	0.00 ± 0.05
TAPE-Arxiv23	70.32	70.31	45.54	45.55	+0.02 ± 0.04
ogbn-arxiv	70.91	71.00	68.35	68.61	+0.26 ± 0.30
ogbn-products	77.97	78.67	75.08	76.73	+1.65 ± 0.53
Ele-Photo	87.26	87.31	85.80	86.13	+0.33 ± 0.09
Ele-Computers	90.68	90.73	89.87	90.08	+0.21 ± 0.08
Books-History	84.53	84.53	82.74	82.81	+0.07 ± 0.04
Roman-Empire	64.63	64.67	19.50	19.51	+0.01 ± 0.07
Amazon-Ratings	53.57	53.53	45.76	46.20	+0.44 ± 0.29

Table 13 shows the results on clean data and under noise, while Table 15 shows the entire noise ladder. The improvement is especially large on WikiCS and ogbn-products, while many other datasets show small differences. Sharpening can therefore also be effective here, but with dataset-dependent effects.

M FULL GAUSSIAN LADDER

Table 14 shows the results at all the five levels of gaussian noise. Each method chooses hyperparameters on the same noise levels that they are evaluated on.

Table 14: Accuracy across the full Gaussian corruption ladder. APPNP propagates logits, PPR-Prob isolates probability-space propagation, and PtS additionally applies the categorical sharpening step.

Dataset	Method	$\sigma = 0$	$\sigma = 0.5$	$\sigma = 1$	$\sigma = 1.5$	$\sigma = 2$
WikiCS	APPNP	77.78	77.12	75.33	72.19	67.97
	PPR-Prob	78.24	77.48	75.69	73.09	68.98
	PtS	**78.42**	**77.61**	**76.14**	**74.38**	**72.15**
Cora-TAPE	APPNP	76.16	75.60	73.87	71.43	68.16
	PPR-Prob	78.42	77.92	76.28	73.52	70.47
	PtS	**79.74**	**78.95**	**77.32**	**75.21**	**72.30**
PubMed-TAPE	APPNP	84.35	83.59	81.98	80.35	78.87
	PPR-Prob	84.96	84.17	82.76	81.46	80.22
	PtS	**85.18**	**84.39**	**83.05**	**81.84**	**80.73**
TAPE-Arxiv23	APPNP	69.53	**65.29**	**55.35**	44.87	35.63
	PPR-Prob	69.68	65.27	55.32	45.87	38.26
	PtS	**69.72**	65.28	55.33	**45.90**	**38.91**
ogbn-arxiv	APPNP	66.10	64.53	58.09	45.50	33.86
	PPR-Prob	65.96	65.03	62.72	52.24	40.43
	PtS	**66.56**	**65.92**	**63.04**	**53.93**	**40.62**
ogbn-products	APPNP	70.58	56.83	**38.51**	**31.64**	**28.54**
	PPR-Prob	71.32	**57.66**	37.26	30.27	27.24
	PtS	**72.17**	57.12	37.28	30.34	27.33
Ele-Photo	APPNP	71.87	70.39	66.70	61.25	55.97
	PPR-Prob	74.43	73.31	70.02	65.39	61.00
	PtS	**74.91**	**73.46**	**70.18**	**65.97**	**61.43**
Ele-Computers	APPNP	75.37	74.14	71.17	66.25	59.94
	PPR-Prob	78.71	77.95	75.21	71.46	66.85
	PtS	**80.11**	**79.26**	**77.07**	**73.74**	**69.56**
Books-History	APPNP	82.59	82.20	81.11	79.67	78.11
	PPR-Prob	82.87	82.47	81.41	80.21	78.90
	PtS	**82.88**	**82.49**	**81.47**	**80.31**	**79.13**
Heterophilic controls						
Roman-Empire	APPNP	**65.65**	53.84	**36.19**	**25.11**	**19.30**
	PPR-Prob	65.54	**53.88**	**36.19**	25.10	**19.30**
	PtS	65.54	53.87	36.18	25.10	**19.30**
Amazon-Ratings	APPNP	51.63	44.20	38.70	**37.09**	36.74
	PPR-Prob	**52.77**	**45.27**	**38.81**	37.06	**36.80**
	PtS	52.74	45.20	**38.81**	37.05	36.78

Table 15: Correct & Smooth across the full Gaussian corruption ladder. C&S-PtS applies the same categorical sharpening step during the smoothing stage.

Dataset	Method	$\sigma = 0$	$\sigma = 0.5$	$\sigma = 1$	$\sigma = 1.5$	$\sigma = 2$
WikiCS	C&S	76.90	75.64	73.14	71.19	70.06
	C&S-PtS	**78.82**	**78.27**	**77.32**	**76.31**	**75.43**
Cora-TAPE	C&S	**87.38**	**87.14**	**86.81**	**86.54**	**86.18**
	C&S-PtS	87.23	87.05	86.66	86.42	86.00
PubMed-TAPE	C&S	86.97	**86.39**	**85.48**	**84.89**	**84.44**
	C&S-PtS	**86.98**	86.38	**85.48**	84.88	**84.44**
TAPE-Arxiv23	C&S	**70.32**	66.31	57.91	**50.56**	45.54
	C&S-PtS	70.31	**66.32**	**57.92**	**50.56**	**45.55**
ogbn-arxiv	C&S	70.91	69.99	68.98	68.49	68.35
	C&S-PtS	**71.00**	**70.07**	**69.16**	**68.71**	**68.61**
ogbn-products	C&S	77.97	76.55	75.84	75.26	75.08
	C&S-PtS	**78.67**	**78.07**	**77.25**	**77.04**	**76.73**
Ele-Photo	C&S	87.26	87.04	86.62	86.20	85.80
	C&S-PtS	**87.31**	**87.11**	**86.73**	**86.40**	**86.13**
Ele-Computers	C&S	90.68	90.57	90.35	90.08	89.87
	C&S-PtS	**90.73**	**90.65**	**90.47**	**90.26**	**90.08**
Books-History	C&S	**84.53**	**84.33**	83.78	83.24	82.74
	C&S-PtS	**84.53**	84.32	**83.84**	**83.27**	**82.81**
Heterophilic controls						
Roman-Empire	C&S	64.63	**53.00**	**35.98**	**25.22**	19.50
	C&S-PtS	**64.67**	52.96	35.95	**25.22**	**19.51**
Amazon-Ratings	C&S	**53.57**	**48.18**	46.47	45.98	45.76
	C&S-PtS	53.53	**48.18**	**46.74**	**46.33**	**46.20**

N PRESERVING MASS

The row mass chooses how strongly a node's class distribution influences the next propagation step. In Table 16, we test whether rescaling this mass helps after sharpening, compared with normalizing each row sum to one. The comparison also includes propagation without sharpening so that we can examine the effect of row normalization in both cases.

We compare PtS with a variant that resets each row's mass to one after every iteration, $R_\eta^{\mathrm{rn}}(h) = r_\eta(h/m(h))$. Its sharpening-off counterpart, PPR-rn, applies row normalisation after every probability propagation step.

Table 16: Effect of preserving the propagated row mass. Δ_{PtS} is PtS minus PtS-rn and Δ_{PPR} is PPR-Prob minus PPR-rn, both computed within (split, seed, draw). Bold marks the higher of the two accuracies in each row.

Dataset	PtS-rn	PtS	Δ_{PtS}	Δ_{PPR}
Clean ($\sigma = 0$)				
WikiCS	78.24	**78.42**	$+0.18 \pm 0.27$	$+0.15 \pm 0.27$
Cora-TAPE	79.15	**79.74**	$+0.59 \pm 0.66$	$+0.17 \pm 0.58$
PubMed-TAPE	85.12	**85.18**	$+0.07 \pm 0.16$	$+0.03 \pm 0.08$
TAPE-Arxiv23	**69.75**	69.72	-0.03 ± 0.06	-0.08 ± 0.04
ogbn-arxiv	**67.45**	66.56	-0.89 ± 0.34	-1.11 ± 0.32
ogbn-products	71.93	**72.17**	$+0.24 \pm 0.11$	$+0.60 \pm 0.05$
Ele-Photo	**74.92**	74.91	-0.01 ± 0.30	-0.34 ± 0.47
Ele-Computers	79.67	**80.11**	$+0.45 \pm 0.15$	$+0.01 \pm 0.33$
Books-History	**82.90**	82.88	-0.02 ± 0.17	$+0.02 \pm 0.07$
Mean (9 main)	76.57	**76.63**	$\mathbf{+0.06}$	$\mathbf{-0.06}$
Heterophilic controls				
Roman-Empire	**65.54**	**65.54**	$+0.00 \pm 0.02$	$+0.00 \pm 0.01$
Amazon-Ratings	52.69	**52.74**	$+0.05 \pm 0.14$	$+0.02 \pm 0.15$
Corrupted ($\sigma = 2$)				
WikiCS	69.66	**72.15**	$+2.48 \pm 0.42$	$+1.01 \pm 0.42$
Cora-TAPE	71.24	**72.30**	$+1.06 \pm 0.60$	$+0.50 \pm 0.46$
PubMed-TAPE	80.63	**80.73**	$+0.10 \pm 0.13$	$+0.14 \pm 0.17$
TAPE-Arxiv23	38.69	**38.91**	$+0.22 \pm 0.16$	$+0.19 \pm 0.10$
ogbn-arxiv	**41.68**	40.62	-1.07 ± 0.29	-0.17 ± 1.11
ogbn-products	**27.44**	27.33	-0.12 ± 0.14	-0.26 ± 0.26
Ele-Photo	60.73	**61.43**	$+0.70 \pm 0.55$	$+0.40 \pm 0.32$
Ele-Computers	67.42	**69.56**	$+2.14 \pm 0.37$	$+1.10 \pm 0.49$
Books-History	79.09	**79.13**	$+0.04 \pm 0.09$	$+0.04 \pm 0.06$
Mean (9 main)	59.62	**60.24**	$\mathbf{+0.61}$	$\mathbf{+0.33}$
Heterophilic controls				
Roman-Empire	19.29	**19.30**	$+0.00 \pm 0.04$	$+0.00 \pm 0.03$
Amazon-Ratings	**36.79**	36.78	-0.02 ± 0.07	-0.01 ± 0.06

The differences are small on many noise-free datasets while mass preservation gives larger improvements under noise, especially on WikiCS and Ele-Computers. The effect is still not positive everywhere, on ogbn-arxiv, the variant with row normalization is better in both. This means mass preservation influences the results but does not guarantee higher accuracy.

O DEPTH SENSITIVITY

Table 17 gives the complete values for the depth analysis in Figure 2. Logit-Sharp updates the logits as $L \leftarrow L + \eta\ \mathrm{softmax}(L)$ after each APPNP propagation step, with final predictions $\mathrm{softmax}(L)$. We vary propagation steps K, while keeping α and η fixed. Without restart, APPNP and PPR-Prob are pure diffusion methods, so they oversmooth and lose substantial accuracy at greater depths, while strong sharpening reduces this loss substantially. Restart dampens this decrease, but PtS keeps its

advantage at greater depths here. A larger η does not always help: $\eta = 200$ is best at large depth without restart, while $\eta = 16$ is better with restart at $\sigma = 2$.

Table 17: Depth sensitivity under clean and corrupted inputs.

Curve	K=1	K=2	K=3	K=5	K=10	K=20	K=40	K=100
$\sigma = 0$, $\alpha = 0$ *(no restart)*								
frozen Q	67.5	67.5	67.5	67.5	67.5	67.5	67.5	67.5
APPNP	74.5	74.3	73.4	70.7	64.8	58.2	50.2	40.5
Logit-Sharp η=16	74.5	74.9	74.7	74.3	72.7	68.0	60.7	52.9
PPR-Prob	75.1	75.8	75.2	73.3	68.5	61.4	53.7	44.6
PtS η=16	75.1	75.8	75.8	75.3	74.4	73.5	71.9	70.6
PtS η=200	75.1	75.4	75.4	75.1	74.7	74.2	73.5	73.2
$\sigma = 0$, $\alpha = 0.1$ *(restart)*								
frozen Q	67.5	67.5	67.5	67.5	67.5	67.5	67.5	67.5
APPNP	74.3	74.5	74.1	73.0	71.4	70.6	70.4	70.4
Logit-Sharp η=16	74.3	75.1	75.0	74.6	73.3	70.2	64.6	61.0
PPR-Prob	74.8	75.6	75.6	75.0	74.0	73.6	73.5	73.5
PtS η=16	74.8	75.9	76.1	75.8	75.3	74.9	74.5	74.2
PtS η=200	74.8	75.7	75.7	75.5	75.2	75.0	74.9	74.8
$\sigma = 2$, $\alpha = 0$ *(no restart)*								
frozen Q	41.9	41.9	41.9	41.9	41.9	41.9	41.9	41.9
APPNP	54.0	55.8	55.5	53.8	50.3	45.9	40.4	34.7
Logit-Sharp η=16	54.0	56.0	56.4	56.4	55.6	53.0	48.5	42.5
PPR-Prob	54.2	57.7	58.5	57.8	54.4	50.2	44.5	37.2
PtS η=16	54.2	58.1	58.8	59.4	59.0	58.1	57.0	56.1
PtS η=200	54.2	57.7	58.0	58.1	58.7	58.5	57.9	57.7
$\sigma = 2$, $\alpha = 0.1$ *(restart)*								
frozen Q	41.9	41.9	41.9	41.9	41.9	41.9	41.9	41.9
APPNP	53.5	55.2	55.4	54.7	53.5	52.8	52.6	52.6
Logit-Sharp η=16	53.5	55.8	56.4	56.5	56.0	54.2	51.1	48.0
PPR-Prob	53.1	55.6	56.3	56.3	55.5	55.0	54.8	54.8
PtS η=16	53.1	57.1	57.9	58.1	58.2	57.8	57.3	57.0
PtS η=200	53.1	56.8	57.2	57.1	56.7	56.4	56.2	56.1

P DIRICHLET AND GINI

Figure 6 shows how the energy components and validation accuracy evolve through the PtS iteration on WikiCS. We use the configuration $\alpha = 0.1$, $\eta = 16$, and $\sigma = 2$ to illustrate what happens before and after propagation and sharpening.

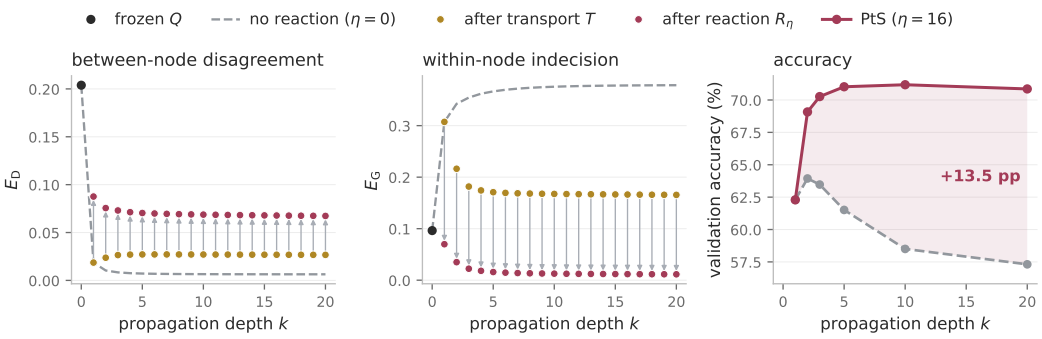

Figure 6: Evolution of the Dirichlet energy E_{D} (left), Gini energy E_{G} (middle), and validation accuracy (right) under gaussian feature corruption at $\sigma = 2$. Gold and red markers show values after propagation and sharpening. Black markers denote the frozen predictions Q. Propagation reduces disagreement between nodes, but increases indecision within nodes, while sharpening has the opposite effect

In the visualization, we can see that propagation reduces the Dirichlet term and increases the Gini term, while sharpening has the opposite effect. Still, PtS maintains higher validation accuracy at greater depth than the variant without reaction. Since sharpening preserves each node's predicted class, its effect must occur through subsequent propagation steps. Proposition 1 guarantees local energy descent under sharpening.

Q Per-node performance

We start from a strong homophily prior: neighbouring nodes should be similar. We group nodes by local homophily, degree quintile, and by the confidence $\max_c Q_{ic}$. Table 18 shows the test-accuracy difference between PtS and APPNP in each group, using backbone seed 0 for each split and noise draw 0 at $\sigma = 2$, with the hyperparameters selected in the main experiment.

We can see that PtS gives the greatest improvement in groups with higher local homophily. Most interestingly, PtS helps most in uncertain areas, the accuracy increase is also higher at nodes with lower prediction confidence. This suggests that PtS is most useful when the original predictions are uncertain, and the neighbourhood provides relevant class information.

Table 18: PtS−APPNP accuracy difference in percentage points, with nodes stratified by local homophily, by degree quintile, and by the confidence of the frozen prediction.

Node stratum	Bucket	PtS−APPNP (σ=0)	PtS−APPNP (σ=2)
Local homophily	0-.2	-1.29	-1.75
	.2-.4	-0.51	-0.35
	.4-.6	+1.88	+4.83
	.6-.8	+3.69	+9.12
	.8-1	+1.90	+5.76
	isolated	+0.00	+0.00
Degree quintile	Q1	+1.27	+3.64
	Q2	+1.63	+3.56
	Q3	+1.86	+3.77
	Q4	+1.95	+4.41
	Q5	+2.04	+5.66
Anchor confidence quintile	Q1	+3.41	+5.12
	Q2	+2.30	+4.66
	Q3	+1.60	+4.39
	Q4	+0.95	+3.87
	Q5	+0.48	+3.00

R RUNTIME AND OVERHEAD

Table 19 shows the runtime for the base models forward pass, as well as the post hoc processing at both 1 and 100 steps. We can see that, generally, for all methods, the runtime is very low.

Table 19: Runtime and overhead: MLP forward-pass and post-hoc call times in milliseconds at $\sigma = 2$

		MLP	Per step (ms)			$K = 100$ (ms)			PtS/APPNP
Dataset	Device	fwd (ms)	APPNP	PPR-Prob	PtS	APPNP	PPR-Prob	PtS	per step
WikiCS	CPU	23.0	1.00	2.02	4.72	100	202	472	4.7
Cora-TAPE	CPU	5.4	0.09	0.44	1.31	8.8	43.6	131	14.9
PubMed-TAPE	CPU	59.0	0.18	0.90	3.08	17.7	89.8	308	17.4
TAPE-Arxiv23	CPU	87.6	5.45	12.1	30.7	545	1,211	3,068	5.6
ogbn-arxiv	A100	9.9	0.90	1.69	3.85	90.0	169	385	4.3
ogbn-products	A100	161	30.7	36.7	54.3	3,072	3,669	5,428	1.8
Ele-Photo	CPU	169	1.98	3.87	9.73	198	387	973	4.9
Ele-Computers	CPU	217	4.03	6.55	15.6	403	655	1,564	3.9
Books-History	CPU	138	1.98	3.29	9.19	198	329	919	4.6
Roman-Empire	CPU	41.7	0.41	1.64	5.61	40.8	164	561	13.8
Amazon-Ratings	CPU	45.2	0.28	0.93	3.10	27.8	92.7	310	11.2

S CALIBRATION

Sharpening can make class distributions more concentrated, but directly influencing probabilities can harm the model's calibration. We therefore examine the negative log-likelihood (NLL) and expected calibration error (ECE) before and after temperature scaling using validation nodes.

Table 20: Calibration of the frozen prediction, APPNP and PtS with an MLP backbone, before and after temperature scaling fitted on validation nodes. T is the fitted temperature and accuracy drift is the change in accuracy caused by scaling. Values are means over WikiCS, Cora-TAPE, PubMed-TAPE, TAPE-arxiv23, ogbn-arxiv and ogbn-products

σ	Method	T	raw NLL	scaled NLL	raw ECE	scaled ECE	acc. drift (pp)
0	Frozen Q	1.34	1.226	1.132	0.088	0.040	0
	APPNP	0.84	0.983	0.907	0.129	0.037	0
	PtS	2.11	1.591	0.932	0.158	0.050	0
2	Frozen Q	3.55	3.514	2.112	0.302	0.047	0
	APPNP	1.47	2.108	1.752	0.178	0.075	0
	PtS	2.92	3.088	1.692	0.254	0.093	0

PtS has higher uncorrected NLL and ECE than APPNP, both with and without noise. Temperature scaling mitigates this without affecting accuracy.