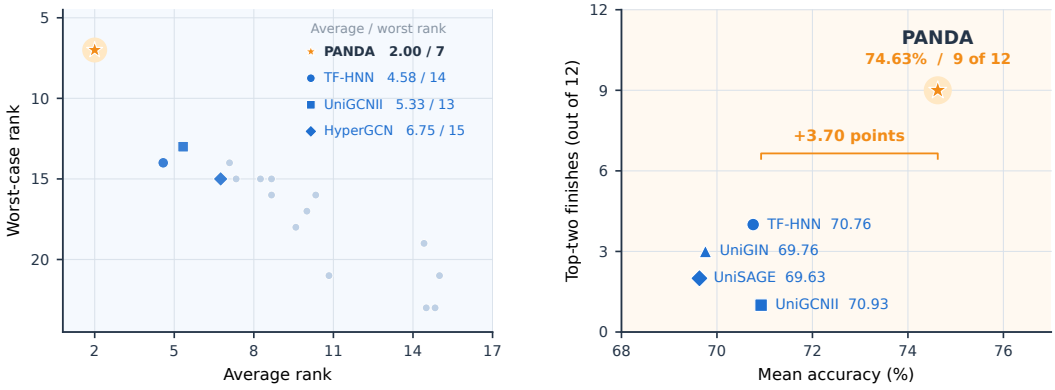
# PANDA: LEARNING THE HYPERGRAPH YOU WERE NEVER GIVEN VIA PROTOTYPE-ANCHORED DIFFUSION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Hypergraph neural networks leverage higher order interactions by propagating node features along an observed incidence. When the observed incidence misses task relevant relations, hypergraph structure learning methods infer or refine hyperedges from node features. When learned and observed structures share a propagation schedule, their diffusion depths cannot be controlled independently. We introduce Prototype Anchored Neural Diffusion for Adaptive Hypergraphs (PANDA), which learns a soft incidence between $N$ nodes and $K$ trainable feature space prototypes. Shallow diffusion over its induced operator limits repeated global coupling, while an observed view filter with independently chosen depth supports either one step smoothing or deeper restart propagation. A learned scalar gate combines the two filtered representations at each layer. The learned incidence requires $O(NK)$ storage, and neither filter needs to materialize a dense node to node operator. Across 12 node classification benchmarks, PANDA achieves the best average accuracy rank of $2.00$, compared with $4.58$ for the runner up TF-HNN. The results support allocating diffusion depth according to relational connectivity, so learning global relations need not entail deep global mixing.




(a) Accuracy ranks (b) Mean performance

Figure 1: Comparison of PANDA and the rivals in terms of average rank and mean accuracy. *(a)* Average and worst-case accuracy ranks for the 17 methods with complete results. Lower is better. *(b)* Mean accuracy and number of top-two finishes for the five methods with the highest mean accuracy. Higher is better. PANDA exceeds UniGCNII, the runner-up by mean accuracy, by $3.70$ percentage points. Top-two counts include ties, and all quantities are computed from Table 1.

## 1 INTRODUCTION

Hypergraph neural networks (HNNs) turn higher-order memberships into paths for exchanging node information (Feng et al., 2019; Bai et al., 2021; Huang & Yang, 2021). Learning useful representations therefore depends on both which relations are available and how information propagates



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.

through them. For node classification, recorded memberships can leave task-relevant relations unrepresented. Papers on the same topic, for example, need not share an author or a citation group. Hypergraph structure learning addresses this gap by refining a supplied incidence or inferring latent hyperedges from task supervision (Cai et al., 2022; Duta & Liò, 2025).

Once relations are specified, propagation determines how they shape node representations. HNDiffN formulates continuous-time diffusion on nodes and hyperedges to develop stable neural architectures (Lu et al., 2026). HyperAim separates low-pass and high-pass filtering and combines frequency-specific representations (Li et al., 2026c). Combining learned and observed relations also requires choosing where to fuse them. HERALD blends an observed operator with one derived from learned incidence (Zhang et al., 2022). HGraphormer combines global attention with a hypergraph operator before message passing (Qu et al., 2026). In these constructions, multi-step propagation acts after the relation sources have been combined, tying together propagation through the two sources.

This coupling matters because additional propagation can play different roles on the two structures. When a learned operator has global support, one application gives every node nonzero dependence on every other node, although the strength of that dependence and the mixing rate remain spectral properties. Further applications repeat global coupling rather than reveal new topological reach. Over an observed hypergraph, additional steps can instead connect nodes through successive shared hyperedges. The question is how to learn global higher-order relations while retaining independent control over these two forms of propagation.

We introduce Prototype-Anchored Neural Diffusion for Adaptive Hypergraphs (PANDA). Its dual-view construction learns a feature-induced incidence while retaining observed memberships. The $K$ trainable prototypes serve as shared reference points in feature space. Each of the $N$ nodes assigns soft membership to these prototypes, giving an $N \times K$ incidence whose columns define soft hyperedges. Nodes can then exchange information through shared prototypes even when no observed hyperedge connects them. This factorization requires $O(NK)$ storage for the learned incidence and supports propagation without constructing a dense node-to-node matrix.

PANDA then uses asymmetric dual diffusion to control repeated coupling separately in the two views. The feature induced operator is positive semidefinite, has rank at most $K$, and has a strictly positive entry for every node pair. For a fixed learned incidence, repeated application converges to global averaging, with finite step contraction governed by its nonprincipal spectrum. PANDA accordingly uses a shallow feature induced filter to limit repeated global coupling and an independently specified observed filter that supports either one step smoothing or deeper restart propagation. Both filters act on the same projected node features. Adaptive fusion combines their outputs through a learned scalar gate at each layer. The prototypes and gates are trained jointly with the classifier using the labeled nodes in the transductive setting. Global support thus remains available even when feature induced diffusion is shallow.

Across 12 node classification benchmarks, PANDA ranks first or tied first on 7 and places in the top two on 9. Its average accuracy rank is $2.00$, compared with $4.58$ for the runner up TF-HNN (Tang et al., 2025). Figure 1 summarizes this advantage in both rank and mean accuracy. Component ablations isolate the two views, their fusion, and the observed filter. Incidence removal and rewiring further measure reliance on observed memberships.

## 2 Related Work

A hypergraph represents group relations through hyperedges that can connect more than two nodes. With hyperedge memberships fixed, hypergraph neural networks can still learn how node information is exchanged. AllSet learns multiset aggregation functions (Chien et al., 2022), while ED-HNN parameterizes permutation-equivariant diffusion within hyperedges (Wang et al., 2023a). Both make information exchange more expressive over the supplied memberships. The distinction is between learning what to communicate along an existing relation and learning which relations should carry information. More flexible aggregation addresses the former, while changing the available propagation paths requires a choice about the incidence itself.

Learning incidence also involves deciding what role the observed structure should retain. Dynamic construction rebuilds hyperedges from evolving node representations, as in DHGNN and TDHNN (Jiang et al., 2019; Zhou et al., 2023). HSL revises a supplied incidence through task guided sam-

pling while maintaining consistency with the original structure (Cai et al., 2022). SPHINX instead infers a discrete latent hypergraph when none is supplied (Duta & Liò, 2025). DSA-HG induces soft latent hyperedges and augments the observed structure (Chen et al., 2027). DHKH retains original topology alongside dynamic hyperedges, allowing both observed and inferred relations to contribute to representation learning (Kang et al., 2022). These designs show that learning a hypergraph and replacing the observed hypergraph are separate decisions.

AnchorGraph constructs a scalable pairwise graph through normalized assignments to data anchors, and the anchor variant of IDGL uses this factorization for learned graph structure (Liu et al., 2010; Chen et al., 2020). SoftHGNN forms continuous visual hyperedges from similarities to learned prototypes (Lei et al., 2026). PANDA uses prototype memberships for transductive node classification while retaining the supplied hypergraph and filtering the two operators separately before fusion.

The effect of a relation also depends on how propagation transforms the signal. TF-HNN separates structural propagation from model training and exploits long range interactions through a precomputed message passing module (Tang et al., 2025). Extending propagation therefore need not add trainable message passing layers. Work on heterophilic hypergraphs and high pass filtering shows why smoothing alone can discard discriminative information (Li et al., 2025b; 2026b). HyperAim makes frequency selectivity explicit through decoupled low pass and high pass channels together with a coupled framelet channel (Li et al., 2026c). These studies treat diffusion depth, spectral response, and trainable network depth as separate design choices.

Model driven reconstruction also separates explicit iterative operators from learned parameters and studies recovery under incomplete graph observations (Zhao et al., 2024; 2025). PANDA applies this separation to relation construction and propagation in transductive hypergraph classification.

Combining relation sources adds a further choice about where propagation should interact. HERALD and HGraphormer mix learned and observed operators before message passing (Zhang et al., 2022; Qu et al., 2026), whereas S2-HGNN fuses representations from an original hypergraph branch and scale-derived auxiliary branches (Zhou et al., 2026). For multi-step linear diffusion, powers of a mixed operator include terms that switch between relation sources. Filtering each operator separately allows its depth and coefficients to be chosen before the outputs meet and prevents a diffusion step from alternating between learned and observed relations. The point of fusion therefore changes the propagation computation, not just the final weighting of representations. PANDA uses separate polynomial filters within each layer and combines their outputs through a scalar gate. The operator induced by its global prototype incidence receives a shallow filter, while observed memberships retain their own diffusion schedule. Appendix A provides further comparisons.

# 3 PANDA

We study transductive node classification on a hypergraph with node set $\mathcal{V} = \{1, \dots, N\}$. The model receives a feature matrix $\mathbf{X} \in \mathbb{R}^{N \times F}$ with $F$ features per node and an observed incidence $\mathbf{H} \in \{0, 1\}^{N \times M}$ over $M$ hyperedges. Each node $n \in \mathcal{V}$ has a label $y_n \in \{1, \dots, C\}$ among $C$ classes. During training, only a subset of node labels is observed, while $\mathbf{X}$ and $\mathbf{H}$ are available for the full hypergraph. Conventional message-passing HNNs propagate node representations along the support specified by $\mathbf{H}$ (Zhou et al., 2006; Feng et al., 2019; Huang & Yang, 2021). PANDA constructs a feature-induced view from soft memberships between nodes and $K$ learned feature prototypes while preserving $\mathbf{H}$ in an observed view. At each layer, the operators associated with the two views propagate the same projected representation for different numbers of steps. A scalar gate combines their outputs before the next layer. Figure 2 shows the resulting computation.

## 3.1 Dual-View Construction

PANDA keeps feature-induced relations and observed memberships in separate views because their operators have different support and mixing behavior.

**Feature-induced view.** Directly representing all feature-based node relations requires an $N \times N$ pairwise affinity matrix. PANDA instead uses $1 \leq K \leq N$ shared, trainable prototypes $\mathbf{P} \in \mathbb{R}^{K \times d}$ to parameterize an $N \times K$ higher-order incidence. A linear encoder without a bias term produces

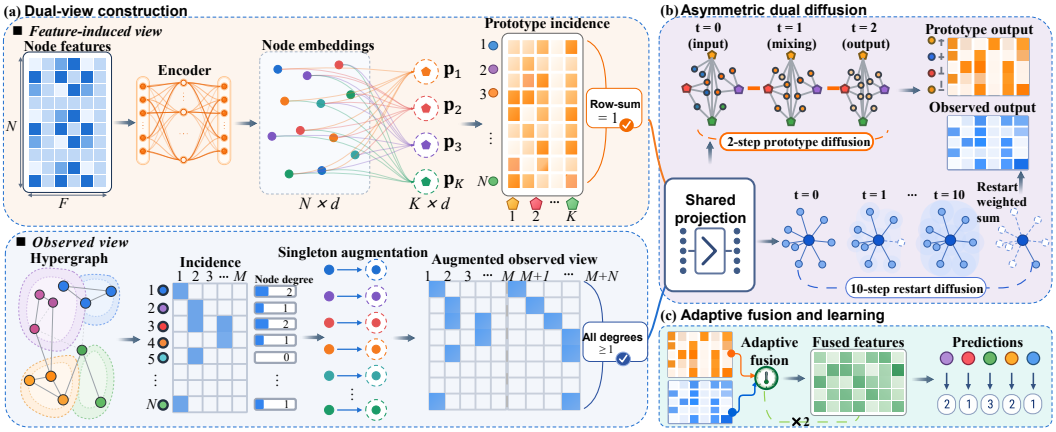



Figure 2: Overview of PANDA. *(a)* Dual-view construction. The feature-induced view constructs a row-normalized incidence from node features and $K$ trainable prototypes. The illustrated observed view appends one singleton hyperedge for each node. *(b)* Asymmetric dual diffusion. The two branches share a feature projection. The feature-induced branch applies shallow prototype diffusion. The illustrated observed branch uses restart diffusion over the augmented incidence. *(c)* Adaptive fusion and learning. Each layer uses a learned scalar gate to combine the branch outputs, and the final layer produces node logits.

$\mathbf{Z} = E(\mathbf{X}) \in \mathbb{R}^{N\times d}$, and temperature-scaled cosine similarities define

$$[\mathbf{H}_\sigma]_{nk} = \frac{\exp(\cos(\mathbf{z}_n, \mathbf{p}_k)/\tau)}{\sum_{j=1}^{K} \exp(\cos(\mathbf{z}_n, \mathbf{p}_j)/\tau)}, \qquad \mathbf{H}_\sigma \in \mathbb{R}^{N\times K}, \tag{1}$$

Here $\mathbf{z}_n$ and $\mathbf{p}_k$ denote the $n$-th and $k$-th rows of $\mathbf{Z}$ and $\mathbf{P}$, respectively. Each cosine uses $\max(\|\mathbf{v}\|_2, \varepsilon)$ in its norm denominator for a fixed $\varepsilon > 0$. The learned temperature is $\tau = \mathrm{clip}(\exp(q_\tau), \tau_{\min}, \tau_{\max})$, where $0 < \tau_{\min} < \tau_{\max}$. Spherical $K$-means initializes $\mathbf{P}$ from $\mathbf{Z}$ once before the first optimization step of each run, after which $\mathbf{P}$ is updated by gradient descent. Each row of $\mathbf{H}_\sigma$ is a distribution over prototypes, and each column defines a soft hyperedge shared by all nodes. This construction creates feature-induced paths between nodes without requiring them to share a hyperedge in $\mathbf{H}$. The prototype columns act as trainable hyperedges rather than post hoc cluster labels. Their memberships are degree-normalized into the same node-hyperedge-node operator form used for the observed view, and are learned for node classification rather than used to approximate a fixed pairwise affinity. The same incidence is shared across layers, so layer-specific projections alter the propagated signal without redefining the feature-induced support.

Define $\mathbf{D}_e^\sigma = \mathrm{diag}(\mathbf{H}_\sigma^\top \mathbf{1}_N)$. Finite assignment logits make its diagonal strictly positive, so $\mathbf{D}_e^\sigma$ is invertible. Since row stochasticity gives $\mathbf{H}_\sigma \mathbf{1}_K = \mathbf{1}_N$, every node has unit degree, and the standard hypergraph normalization (Feng et al., 2019) reduces to

$$\mathbf{S}_\sigma = \mathbf{H}_\sigma (\mathbf{D}_e^\sigma)^{-1} \mathbf{H}_\sigma^\top = \mathbf{H}_\sigma (\mathrm{diag}(\mathbf{H}_\sigma^\top \mathbf{1}_N))^{-1} \mathbf{H}_\sigma^\top. \tag{2}$$

Eq. (2) has the anchor graph factorization of Liu et al. (2010), with trainable prototype memberships replacing assignments to fixed anchors. Proposition 1 records the properties of this form that determine the feature induced filter. Let $\mathbf{Q} = \mathbf{H}_\sigma (\mathbf{D}_e^\sigma)^{-1/2} \in \mathbb{R}^{N\times K}$.

**Proposition 1** (Feature-induced operator)**.** *For a fixed $\mathbf{H}_\sigma$ defined by Eq. (1), finite assignment logits imply*

$$\begin{gathered} \mathbf{S}_\sigma = \mathbf{S}_\sigma^\top = \mathbf{Q}\mathbf{Q}^\top \succeq 0, \qquad \mathrm{rank}(\mathbf{S}_\sigma) \le K, \\ \mathbf{S}_\sigma \mathbf{1}_N = \mathbf{1}_N, \qquad [\mathbf{S}_\sigma]_{ij} > 0 \;\forall i, j, \qquad \lim_{t\to\infty} \mathbf{S}_\sigma^t = \frac{1}{N} \mathbf{1}_N \mathbf{1}_N^\top. \end{gathered} \tag{3}$$

For $N > 1$, let $\mathbf{\Pi} = N^{-1} \mathbf{1}_N \mathbf{1}_N^\top$ and $\lambda_\star = \max_{r>1} \lambda_r(\mathbf{S}_\sigma)$. The eigendecomposition in Appendix B also gives $\|\mathbf{S}_\sigma^t \mathbf{U} - \mathbf{\Pi}\mathbf{U}\|_F \le \lambda_\star^t \|\mathbf{U} - \mathbf{\Pi}\mathbf{U}\|_F$ for any node signal $\mathbf{U}$. Thus $K$ bounds

rank without restricting support, while the assignment-dependent gap $1-\lambda_\star$ controls finite-step mixing. Neither $K$ nor $\tau$ fixes this rate alone because the spectrum also depends on the encoded nodes and learned prototypes. In the experiments, PANDA uses powers through order two. This includes the identity, direct global interaction, and one repeated interaction. Appendix D.5.3 measures the resulting operator behavior on six datasets. The incidence factorization permits propagation through $\mathbf{H}_\sigma$ and $\mathbf{H}_\sigma^\top$ without materializing a dense $N \times N$ operator. A proof is given in Appendix B. Because $\mathbf{S}_\sigma$ does not encode the memberships in $\mathbf{H}$, PANDA retains them in a separate observed view.

**Observed view.** The observed view retains the higher-order memberships in $\mathbf{H}$ and may append one singleton hyperedge per node. Figure 2 illustrates the augmented incidence $\mathbf{H}_{\text{obs}} = [\mathbf{H} \mid \mathbf{I}_N]$. The appended identity provides a self-information path for every node without introducing relations between distinct nodes. Writing $M_{\text{obs}}$ for the number of columns in $\mathbf{H}_{\text{obs}}$, define $\mathbf{D}_v^{\text{obs}} = \text{diag}(\mathbf{H}_{\text{obs}}\mathbf{1}_{M_{\text{obs}}})$ and $\mathbf{D}_e^{\text{obs}} = \text{diag}(\mathbf{H}_{\text{obs}}^\top \mathbf{1}_N)$. We assume that observed hyperedges are nonempty. If singleton augmentation is disabled, inverse square-root node degrees at isolated nodes are defined as zero. The normalized propagation operator is

$$\mathbf{S}_{\text{obs}} = (\mathbf{D}_v^{\text{obs}})^{-1/2}\mathbf{H}_{\text{obs}}(\mathbf{D}_e^{\text{obs}})^{-1}\mathbf{H}_{\text{obs}}^\top(\mathbf{D}_v^{\text{obs}})^{-1/2}. \tag{4}$$

The hyperedge-degree factor averages messages within each observed hyperedge, and the symmetric node-degree factors balance nodes with different membership counts. This operator is symmetric positive semidefinite with eigenvalues in $[0, 1]$, and its powers remain confined to the connected components defined by $\mathbf{H}_{\text{obs}}$ (Zhou et al., 2006).

**Proposition 2** (Singleton augmentation)**.** *If $\mathbf{H}_{\text{obs}} = [\mathbf{H} \mid \mathbf{I}_N]$, let $d_n^{\text{obs}} = 1 + \sum_{e=1}^{M}[\mathbf{H}]_{ne}$ denote the observed degree of node $n$. Then every node $n$ satisfies*

$$\left[\mathbf{D}_v^{\text{obs}}\right]_{nn} = d_n^{\text{obs}} \geq 1, \qquad [\mathbf{S}_{\text{obs}}]_{nn} \geq (d_n^{\text{obs}})^{-1}. \tag{5}$$

*If node $n$ is isolated in $\mathbf{H}$, the $n$-th row of $\mathbf{S}_{\text{obs}}$ is the $n$-th standard basis vector.*

Thus singleton augmentation preserves an isolated node under observed-view propagation without adding paths between distinct nodes. A proof is given in Appendix B.

**Design implication.** The two constructions yield a globally supported, rank-constrained operator $\mathbf{S}_\sigma$ and an observed operator $\mathbf{S}_{\text{obs}}$ whose propagation paths are fixed by $\mathbf{H}_{\text{obs}}$. Their different support and mixing behavior motivate the asymmetric filters introduced next.

### 3.2 Asymmetric Dual Diffusion

Propagation depth has different meanings in the two views because powers of the globally supported $\mathbf{S}_\sigma$ repeat global coupling without extending its support, while powers of $\mathbf{S}_{\text{obs}}$ extend reach along supplied memberships. PANDA therefore assigns a separate truncated polynomial filter to each operator.

For a network with $D$ layers, let $d_0 = F$, $d_D = C$, and $\mathbf{X}^{(0)} = \mathbf{X}$. Let $m \geq 1$ denote the maximum power in the feature-induced filter. In the observed branch, $L = 1$ denotes one-step HGNN smoothing, whereas $L > 1$ denotes the truncation order of a restart filter. At layer $\ell \in \{1, \ldots, D\}$, a shared affine map produces $\mathbf{U}^{(\ell)} = \mathbf{X}^{(\ell-1)}\mathbf{W}_\ell + \mathbf{1}_N\mathbf{b}_\ell^\top$, where $\mathbf{W}_\ell \in \mathbb{R}^{d_{\ell-1} \times d_\ell}$ and $\mathbf{b}_\ell \in \mathbb{R}^{d_\ell}$. Using the same $\mathbf{U}^{(\ell)}$ in both branches isolates the effects of their propagation operators and diffusion filters from separate feature transformations. The feature-induced and observed branch outputs are

$$\mathbf{R}^{(\ell)} = \Psi_m[\mathbf{S}_\sigma](\mathbf{U}^{(\ell)}) = \sum_{t=0}^{m} c_t \mathbf{S}_\sigma^t \mathbf{U}^{(\ell)}, \tag{6}$$

$$\mathbf{O}^{(\ell)} = \Phi_L[\mathbf{S}_{\text{obs}}](\mathbf{U}^{(\ell)}) = \begin{cases} \mathbf{S}_{\text{obs}}\mathbf{U}^{(\ell)}, & L = 1, \\ \alpha \sum_{t=0}^{L} (1-\alpha)^t \mathbf{S}_{\text{obs}}^t \mathbf{U}^{(\ell)}, & L > 1 \end{cases}. \tag{7}$$

The feature-induced filter starts from $c_0 = \alpha_p \in (0, 1)$ and follows $c_t = c_{t-1}(1 - c_{t-1})$. Since $c_{t-1} - c_t = c_{t-1}^2 > 0$, its positive weights decrease with diffusion order. The observed branch

uses one HGNN smoothing step when $L = 1$ and an unnormalized truncation of the personalized PageRank Neumann series with restart coefficient $\alpha \in (0, 1)$ when $L > 1$ (Gasteiger et al., 2019). Accordingly, $m$ limits repeated global coupling, while the observed branch offers either one-step smoothing or multi-step propagation along observed paths. Both polynomials are left unnormalized, so their output magnitudes can also vary with $m$ and $L$. The two filters introduce no hop-specific parameters. Appendix C.2 reports the settings used for every dataset. PANDA next fuses the branch outputs $\mathbf{R}^{(\ell)}, \mathbf{O}^{(\ell)} \in \mathbb{R}^{N \times d_\ell}$ at layer $\ell$.

### 3.3 Adaptive Fusion and Learning

A learned scalar gate sets the nominal interpolation coefficients of $\mathbf{O}^{(\ell)}$ and $\mathbf{R}^{(\ell)}$ at each layer. Because both filters act on the same $\mathbf{U}^{(\ell)}$ and preserve its feature coordinates, their outputs can be interpolated directly. PANDA parameterizes this interpolation with a scalar logit $q_\ell$ and the feature-induced branch coefficient $g_\ell = \mathrm{sigmoid}(q_\ell)$,

$$\mathbf{Y}^{(\ell)} = (1 - g_\ell)\mathbf{O}^{(\ell)} + g_\ell \mathbf{R}^{(\ell)}. \tag{8}$$

Sharing $g_\ell$ across nodes and feature coordinates restricts adaptation to the layer level and adds only $D$ fusion parameters. Because the filtered outputs can differ in magnitude, $g_\ell$ is not by itself a measure of view importance. After every nonfinal layer, batch normalization, ReLU, and dropout map $\mathbf{Y}^{(\ell)}$ to $\mathbf{X}^{(\ell)}$. The final layer returns logits $\mathbf{Y}^{(D)} \in \mathbb{R}^{N \times C}$. Let $\mathcal{V}_{\mathrm{lab}} \subset \mathcal{V}$ denote the subset of nodes whose labels are observed during training. Only this labeled subset enters the classification objective,

$$\mathcal{L}_{\mathrm{cls}} = -\frac{1}{|\mathcal{V}_{\mathrm{lab}}|} \sum_{n \in \mathcal{V}_{\mathrm{lab}}} \log\Big(\mathrm{softmax}(\mathbf{Y}^{(D)}_{n,:})_{y_n}\Big). \tag{9}$$

Because $\mathbf{H}_\sigma$ depends on $E$, $\mathbf{P}$, and $\tau$, this masked objective learns the feature induced incidence jointly with the feature projections and fusion gates. PANDA keeps the two propagation operators separate and couples their outputs only through $\mathbf{Y}^{(\ell)}$. Training on $\mathcal{V}_{\mathrm{lab}}$ jointly learns $\mathbf{H}_\sigma$, the shared projections, and the layerwise gates, while $\mathbf{H}_{\mathrm{obs}}$ remains fixed.

**Complexity.** Let $d_\star = \max\{d, d_0, \ldots, d_D\}$ and $\xi = \mathrm{nnz}(\mathbf{H}_{\mathrm{obs}})$. Let $\rho_\star$ denote the largest of $d_\star$, $mK$, and $L\xi/N$. One PANDA forward pass costs $O(DNd_\star\rho_\star)$ and forms no dense node-to-node operator. Appendix B.2 gives the full derivation and memory analysis.

**Algorithm 1** PANDA forward pass

**Input:** Node features $\mathbf{X} \in \mathbb{R}^{N \times F}$, incidence $\mathbf{H} \in \{0, 1\}^{N \times M}$, singleton-augmentation choice, current model parameters $E$, $\mathbf{P}$, $q_\tau$, and $\{\mathbf{W}_\ell, \mathbf{b}_\ell, q_\ell\}_{\ell=1}^{D}$, number of classes $C$, depth $D$, diffusion settings $(m, \alpha_p)$ and $(L, \alpha)$ with $m, L \geq 1$, and temperature bounds $0 < \tau_{\min} < \tau_{\max}$

**Output:** Logits $\mathbf{Y}^{(D)} \in \mathbb{R}^{N \times C}$, used by Eq. (9) during training

$\mathbf{Z} \leftarrow E(\mathbf{X})$
$\tau \leftarrow \mathrm{clip}(\exp(q_\tau), \tau_{\min}, \tau_{\max})$
$[\mathbf{H}_\sigma]_{nk} \leftarrow \mathrm{softmax}_k\big(\cos(\mathbf{z}_n, \mathbf{p}_k)/\tau\big)$ ▷ Eq. (1)
Define $\mathbf{S}_\sigma$ from $\mathbf{H}_\sigma$ ▷ Eq. (2)
Set $\mathbf{H}_{\mathrm{obs}} \leftarrow [\mathbf{H} \mid \mathbf{I}_N]$ if augmented, otherwise $\mathbf{H}_{\mathrm{obs}} \leftarrow \mathbf{H}$ ▷ Section 3.1
Define $\mathbf{S}_{\mathrm{obs}}$ from $\mathbf{H}_{\mathrm{obs}}$ ▷ Eq. (4)
$\mathbf{X}^{(0)} \leftarrow \mathbf{X}$
**for** $\ell = 1, \ldots, D$ **do**
  $\mathbf{U}^{(\ell)} \leftarrow \mathbf{X}^{(\ell-1)}\mathbf{W}_\ell + \mathbf{1}_N \mathbf{b}_\ell^\top$ ▷ Section 3.2
  $\mathbf{R}^{(\ell)} \leftarrow \Psi_m[\mathbf{S}_\sigma](\mathbf{U}^{(\ell)})$ ▷ Eq. (6)
  $\mathbf{O}^{(\ell)} \leftarrow \Phi_L[\mathbf{S}_{\mathrm{obs}}](\mathbf{U}^{(\ell)})$ ▷ Eq. (7)
  $g_\ell \leftarrow \mathrm{sigmoid}(q_\ell)$, $\mathbf{Y}^{(\ell)} \leftarrow (1 - g_\ell)\mathbf{O}^{(\ell)} + g_\ell \mathbf{R}^{(\ell)}$ ▷ Eq. (8)
  **if** $\ell < D$ **then** $\mathbf{X}^{(\ell)} \leftarrow \mathrm{Dropout}(\mathrm{ReLU}(\mathrm{BN}(\mathbf{Y}^{(\ell)})))$ ▷ Section 3.3
  **end if**
**end for**
**return** $\mathbf{Y}^{(D)}$

# 4 EXPERIMENTS

## 4.1 EXPERIMENTAL SETUP

We evaluate PANDA on 12 hypergraph node classification benchmarks. Cora, Citeseer, Pubmed, and DBLP use the HyperGCN releases (Yadati et al., 2019) distributed by DHG (Dai & Gao, 2023). Zoo, 20news, mushroom, ntu2012, and modelnet40 use the AllSet releases (Chien et al., 2022), while house and cooking200 use DHG. Training is transductive, with the full feature matrix and observed incidence available and supervision restricted to training labels. Appendix C.1 gives dataset sources and statistics.

The 22 baselines cover fixed incidence propagation and structure learning. They include HGNN (Feng et al., 2019), HGNN$^+$ (Gao et al., 2023), HNHN (Dong et al., 2020), HyperGCN (Yadati et al., 2019), the five UniGNN variants (Huang & Yang, 2021), AllSetTransformer, AllDeepSets (Chien et al., 2022), ED-HNN (Wang et al., 2023a), PhenomNN (Wang et al., 2023b), TF-HNN (Tang et al., 2025), SheafHyperGNN (Duta et al., 2023), FrameHGNN (Li et al., 2025a), DPHGNN (Saxena et al., 2024), and HyperGT (Liu et al., 2024). The structure learning group contains DHGNN (Jiang et al., 2019), HERALD (Zhang et al., 2022), TDHNN (Zhou et al., 2023), and SHSL (Li et al., 2025c). Baseline architectures follow their published definitions, and model specific hyperparameters use the settings recommended in the original papers. Appendix C.3 records the memory adaptations used for HERALD and SHSL on the affected datasets.

For the five citation and co-authorship datasets, we retain the official DHG training masks (Dai & Gao, 2023) and split the remaining nodes into a 500-node validation set and a disjoint test set following Planetoid (Yang et al., 2016). The other datasets use the AllSet $50/25/25$ split (Chien et al., 2022). Every method uses the same splits, five seeds, training budget, and validation-accuracy checkpointing. We report test accuracy as mean$\pm$standard deviation. Appendices C.3 and C.2 give the remaining settings.

## 4.2 MAIN RESULTS

PANDA ranks first or tied first on seven datasets and among the top two on nine (Table 1). Its average accuracy rank is $2.00$, compared with $4.58$ for TF-HNN and $5.33$ for UniGCNII. The Macro-F1 comparison also places PANDA first by average rank at $2.67$ (Appendix D.1).

The leading results span citation and co-authorship data, categorical attributes, geometric descriptors, and the degree-only house and cooking200 benchmarks. PANDA ranks first on cooking200, cora_cc, dblp_ca, zoo, and house, and shares the best result on mushroom and ntu2012. It ranks second on cora_ca and modelnet40, with the latter $0.2$ points below TF-HNN, and third on citeseer_cc, $0.8$ points below PhenomNN. The strongest competitor changes across datasets, whereas PANDA maintains broad top placements. Table 2 examines which design choices support this consistency.

## 4.3 ABLATION STUDY

Table 2 compares the two relational views, singleton augmentation, the fusion gate, and the observed filter over five seeds. Its full row matches the corresponding PANDA results in Table 1.

Removing the observed view produces the largest reduction, averaging $5.7$ points and reaching $13.6$ on cora_ca. Removing the feature-induced view costs $2.4$ points on average and $5.3$ on modelnet40. Singleton augmentation adds $2.0$ points on average and $4.3$ on citeseer_cc, where $56\%$ of test nodes have no observed hyperedge. Appendix D.2.4 compares this effect across datasets with different isolated-node prevalence. The convex gate adds $1.2$ points on average. The selected multi-step restart branch improves on one-step observed propagation by $1.1$ points. The two views provide the principal gains, while singleton preservation, fusion, and the observed propagation choice refine their use.

Table 1: Node classification accuracy in percent over five seeds. Bold and underlined entries mark the best and second-best displayed means. The final column averages dense ranks over available datasets, with lower values better. PANDA configurations are listed in Appendix C.2. A dash denotes an unavailable result.

| Method | cora_ca | citeseer_cc | cooking200 | cora_cc | pubmed_cc | dblp_ca | zoo | 20news | mushroom | ntu2012 | modelnet40 | house | Avg rank↓ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *Classic HGNN* | | | | | | | | | | | | | |
| HGNN | 60.5±1.2 | 37.8±0.2 | 22.1±1.1 | 44.5±0.9 | 24.0±1.6 | 88.1±0.2 | 57.7±8.8 | 81.9±0.1 | 98.2±0.5 | 86.4±0.5 | 91.8±0.2 | 52.8±3.7 | 10.33 |
| HGNN$^+$ | 61.0±1.1 | 38.1±0.3 | 25.5±1.0 | 44.2±0.9 | 23.1±1.9 | 87.5±0.2 | 73.1±5.4 | 81.7±0.4 | 99.6±0.2 | 86.5±0.2 | 91.6±0.1 | 48.0±4.8 | 10.00 |
| HNHN | 48.9±2.8 | 29.3±2.8 | 22.4±2.3 | 38.4±2.2 | 23.2±1.9 | 74.3±7.9 | 55.4±26.5 | 73.0±6.0 | 91.4±2.1 | 86.0±0.6 | 91.4±0.1 | 46.6±4.0 | 14.42 |
| HyperGCN | 54.0±0.4 | 57.1±1.1 | 18.6±0.2 | 58.9±0.9 | 39.1±0.0 | 78.7±0.8 | 96.9±1.5 | **82.7±0.1** | **99.9±0.0** | 85.5±1.1 | 96.3±0.1 | 55.4±0.7 | 6.75 |
| *UniGNN family* | | | | | | | | | | | | | |
| UniGCN | 63.3±0.8 | 37.6±0.2 | 29.9±2.8 | 45.3±0.2 | 24.0±1.6 | 85.3±0.5 | 84.6±0.0 | 81.9±0.1 | 99.8±0.1 | 85.8±0.7 | 91.4±0.1 | 55.7±3.0 | 8.67 |
| UniGAT | 57.7±1.7 | 29.5±0.4 | 22.7±1.5 | 51.4±0.7 | 38.2±1.5 | 86.5±0.2 | 86.9±3.9 | 82.0±0.2 | 99.7±0.0 | 85.8±0.7 | 91.6±0.2 | 47.4±3.8 | 9.58 |
| UniGIN | 65.9±1.5 | 52.2±2.4 | 30.9±0.2 | 58.3±2.0 | 37.3±5.6 | 84.6±0.5 | 97.7±4.6 | 82.3±0.2 | 99.8±0.1 | 86.4±0.5 | 92.2±0.2 | 49.5±5.6 | 7.08 |
| UniSAGE | 65.6±1.0 | 51.7±1.2 | 30.8±0.6 | 58.5±1.3 | 37.1±5.5 | 85.5±0.1 | 96.9±4.5 | 82.4±0.3 | 99.8±0.1 | 86.4±0.5 | 92.2±0.2 | 48.7±5.0 | 7.33 |
| *Set-function methods* | | | | | | | | | | | | | |
| AllSetTransformer | 27.0±5.4 | 28.2±8.4 | 17.0±0.1 | 21.4±5.2 | **39.7±0.4** | 86.8±1.9 | 64.6±7.5 | 82.1±0.2 | 99.6±0.2 | 80.9±1.1 | 96.1±0.1 | 52.8±1.1 | 10.83 |
| AllDeepSets | 18.9±2.3 | 18.3±1.4 | 17.1±0.0 | 18.4±1.7 | 39.5±0.4 | 49.9±9.8 | 47.7±7.1 | 78.2±2.3 | 98.9±1.1 | 12.4±2.9 | 25.2±4.1 | 52.0±0.0 | 14.83 |
| ED-HNN | 22.1±5.8 | 28.1±5.0 | 17.1±0.3 | 23.4±2.2 | 38.7±1.3 | 32.9±9.7 | 47.7±5.8 | 72.6±10.4 | 98.3±1.3 | 6.8±2.6 | 20.7±8.1 | 53.5±2.5 | 15.00 |
| *Deep & energy-based HNNs* | | | | | | | | | | | | | |
| UniGCNII | 71.2±1.0 | 60.3±0.8 | 18.1±0.7 | 67.6±0.3 | 37.5±2.0 | 85.7±1.1 | 90.0±3.1 | 82.3±0.1 | 99.8±0.0 | 88.9±0.4 | 97.6±0.1 | 52.1±0.6 | 5.33 |
| PhenomNN | **73.8±0.9** | 61.6±0.6 | 17.1±0.2 | 68.0±1.1 | 35.5±7.3 | 84.6±1.2 | 40.8±10.8 | - | 92.4±0.3 | 82.4±1.0 | 97.2±0.2 | 53.4±2.0 | 8.64 |
| *Spectral & framelet HNNs* | | | | | | | | | | | | | |
| TF-HNN | 70.0±1.1 | 59.0±0.9 | 17.1±0.1 | 65.0±1.0 | 39.2±0.1 | 85.6±0.1 | 86.9±5.2 | 82.4±0.1 | 99.8±0.1 | **90.5±0.6** | **98.2±0.1** | 55.4±2.4 | 4.58 |
| SheafHyperGNN | 68.5±2.3 | 56.3±0.7 | 18.3±1.0 | 63.9±1.9 | 39.2±0.2 | - | 92.3±3.4 | - | - | 89.1±0.9 | - | 53.9±2.4 | 5.50 |
| FrameHGNN | 51.5±4.5 | 35.7±2.3 | - | 42.6±5.8 | - | - | 46.2±0.0 | - | - | 71.5±1.7 | - | 50.8±1.1 | 15.83 |
| *Attention & hybrid HNNs* | | | | | | | | | | | | | |
| DPHGNN | 61.6±2.5 | 42.5±4.7 | 21.8±2.0 | 59.8±2.9 | 39.2±0.1 | - | 55.4±8.3 | 80.6±0.6 | 99.4±0.3 | 69.1±11.4 | 70.6±14.7 | 50.9±2.7 | 10.64 |
| HyperGT | 51.3±1.0 | 49.6±1.5 | 17.0±0.0 | 51.2±1.6 | 39.1±0.0 | - | 75.4±3.1 | 81.5±0.1 | 98.1±1.6 | 75.5±1.4 | 95.3±0.3 | 55.4±2.8 | 10.64 |
| *Structure-learning methods* | | | | | | | | | | | | | |
| DHGNN | 67.9±0.4 | **67.9±2.2** | 17.1±0.0 | 67.9±0.4 | 20.8±0.0 | 81.8±0.2 | 76.2±4.5 | 80.9±0.1 | 99.5±0.1 | 86.8±0.5 | 95.8±0.1 | 50.3±2.0 | 8.25 |
| HERALD | 65.7±2.1 | 36.5±3.4 | 22.7±1.4 | 55.1±0.4 | 23.1±1.9 | 88.1±0.4 | 69.2±8.8 | 81.9±0.1 | 98.8±0.3 | 87.5±0.4 | 92.2±0.1 | 53.6±2.9 | 8.67 |
| TDHNN | 18.8±5.9 | 21.1±0.4 | 15.9±2.4 | 18.8±5.9 | 39.1±0.0 | 24.4±3.9 | 57.7±5.4 | 82.0±0.2 | 98.7±0.1 | 25.1±15.0 | 71.2±19.7 | 52.0±0.0 | 14.50 |
| SHSL | 49.6±4.4 | 55.1±1.9 | 23.5±0.4 | 49.6±4.4 | 20.8±0.0 | 88.3±0.2 | 72.3±1.5 | 81.7±0.2 | 99.1±0.4 | 81.0±0.7 | 92.2±0.1 | - | 10.00 |
| PANDA (ours) | 73.0±1.6 | 60.8±0.7 | **38.0±0.9** | **69.1±0.8** | 39.3±0.2 | **88.9±0.2** | **100.0±0.0** | 81.8±0.4 | **99.9±0.1** | **90.5±0.5** | 98.0±0.2 | **56.2±1.8** | **2.00** |

Table 2: Component ablation over five runs. Accuracy is mean±standard deviation, and Drop reports the mean and maximum decrease across six datasets. Removing a view retains the other. Without the gate, PANDA uses learned additive fusion $\mathbf{O}^{(\ell)} + \beta_\ell \mathbf{R}^{(\ell)}$. One-step observed sets $L = 1$ in Eq. (7).

| | Test accuracy (%) | | | | | | Drop (pp) | |
|---|---|---|---|---|---|---|---|---|
| Variant | cora_ca | citeseer_cc | cora_cc | ntu2012 | modelnet40 | house | Mean | Max |
| **PANDA (full)** | 73.0±1.6 | 60.8±0.7 | 69.1±0.8 | 90.5±0.5 | 98.0±0.2 | 56.2±1.8 | – | – |
| w/o feature-induced view | 71.2±0.2 | 58.6±0.9 | 68.3±0.4 | 87.5±0.2 | 92.7±0.1 | 55.0±2.2 | 2.4 | 5.3 |
| w/o observed view | 59.4±1.1 | 58.1±1.4 | 59.4±1.1 | 86.8±1.4 | 96.1±0.1 | 53.6±2.5 | 5.7 | 13.6 |
| w/o singleton aug. | 70.6±2.0 | 56.5±2.7 | 66.1±2.8 | 89.7±1.3 | 97.4±0.7 | 55.4±2.5 | 2.0 | 4.3 |
| w/o fusion gate | 72.4±1.0 | 58.7±1.3 | 68.3±0.3 | 89.6±0.7 | 97.3±0.9 | 54.2±3.3 | 1.2 | 2.1 |
| One-step observed | 72.1±1.1 | 59.4±1.5 | 66.8±0.4 | 88.9±0.4 | 97.4±0.9 | 56.5±0.7 | 1.1 | 2.3 |

### 4.4 Efficiency Analysis

Table 3: Inference efficiency on cora_ca with hidden width 128 on an RTX 5060 Ti, measured over 20 runs after 5 warm-ups. Bold marks the lowest structure-learning peak memory. † excludes unprofiled sparse DHG operations.

| Model | Params | Inference (ms) | Peak mem (MB) | FLOPs |
|---|---|---|---|---|
| HGNN | 92,359 | 0.3 | 27.3 | 0.25G |
| TF-HNN | 184,711 | 0.2 | 42.8 | 0.50G |
| UniGCNII | 100,423 | 1.0 | 32.8 | 0.27G |
| AllSetTransformer | 214,151 | 2.0 | 35.0 | 0.60G |
| PhenomNN | 184,455 | 18.2 | 38.0 | 0.50G |
| SheafHyperGNN | 193,666 | 13.5 | 72.1 | 0.73G |
| DHGNN | 184,576 | 1.1 | 44.4 | —† |
| HERALD | 752,008 | 3.8 | 222.7 | 1.20G |
| TDHNN | 1,238,610 | 1.5 | 50.6 | 1.57G |
| SHSL | 4,291,554 | 5.4 | 144.5 | 11.12G |
| PANDA (ours) | 555,150 | 3.3 | **39.0** | 1.49G |

Among the compared structure learning methods, PANDA records the lowest peak memory at $39.0$ MB, $12\%$ below DHGNN, and uses fewer parameters than HERALD, TDHNN, and SHSL (Table 3). Its $1.49$G FLOPs and $3.3$ ms inference time are also below those of several learned structure baselines. The factorized $N \times K$ incidence avoids storing an $N \times N$ relation matrix.

## 5 Implications Summary

The feature induced operator has global one step support with assignment dependent finite step mixing, whereas the observed operator expands reach through connected memberships. PANDA filters them separately before fusion. Component ablations, incidence removal, and rewiring test these roles in Appendices D.2.1 and D.3.

## 6 Conclusion

PANDA separates learned relation support from observed reach. Trainable prototype hyperedges provide factorized global interaction, while observed memberships follow an independent diffusion schedule. Across 12 benchmarks, PANDA attains a $2.00$ average accuracy rank with seven first or tied first results and the lowest peak memory among the compared structure learning methods. Extending prototype assignment to unseen nodes and conditioning fusion on local relation quality are natural directions beyond the transductive, layer level design. The results suggest that learned relations benefit from propagation matched to their operator behavior.

### AI USE STATEMENT

We used generative AI tools to polish the manuscript and assist with literature organization, mathematical and notation checks, and consistency checks. We did not use generative AI tools to formulate the research idea, develop the method or its implementation, design or run experiments, process datasets, or select reported results. The authors wrote the initial manuscript and manually reviewed all AI assisted revisions, cited sources, mathematical statements, and reported values. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

### ETHICS STATEMENT

PANDA is evaluated on existing public benchmarks for hypergraph node classification. Applications involving data about individuals require assessment of privacy, bias, and potential downstream harms in their intended setting.

### REPRODUCIBILITY STATEMENT

Section 3 specifies the method and forward pass, with proofs in Appendix B. Dataset descriptions and the evaluation protocol appear in Appendix C.1 and Section 4.1. Hyperparameters and random seeds are provided in Appendices C.2 and C.3. Together, these sections provide the information needed to reconstruct PANDA and evaluate it under the reported protocol.

## REFERENCES

Song Bai, Feihu Zhang, and Philip H. S. Torr. Hypergraph convolution and hypergraph attention. *Pattern Recognition*, 110:107637, 2021. doi: 10.1016/j.patcog.2020.107637.

Derun Cai, Moxian Song, Chenxi Sun, Baofeng Zhang, Shenda Hong, and Hongyan Li. Hypergraph structure learning for hypergraph neural networks. In *International Joint Conference on Artificial Intelligence*, pp. 1923–1929, 2022. doi: 10.24963/ijcai.2022/267.

Ding-Yun Chen, Xiao-Pei Tian, Yu-Te Shen, and Ming Ouhyoung. On visual similarity based 3D model retrieval. *Computer Graphics Forum*, 22(3):223–232, 2003. doi: 10.1111/1467-8659.00669.

Yu Chen, Lingfei Wu, and Mohammed J. Zaki. Iterative deep graph learning for graph neural networks: Better and robust node embeddings. In *Advances in Neural Information Processing Systems*, pp. 19314–19326, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/e05c7ba4e087beea9410929698dc41a6-Abstract.html.

Yuanyi Chen, Moxuan Zeng, and Xiang-Yan Tang. Hypergraph augmentation via latent hyperedge induction. *Information Sciences*, 758:123992, 2027. doi: 10.1016/j.ins.2026.123992.

Eli Chien, Chao Pan, Jianhao Peng, and Olgica Milenkovic. You are AllSet: A multiset function framework for hypergraph neural networks. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=hpBTIv2uy_E.

Philip S. Chodrow, Nate Veldt, and Austin R. Benson. Generative hypergraph clustering: From blockmodels to modularity. *Science Advances*, 7(28):eabh1303, 2021. doi: 10.1126/sciadv.abh1303.

Qionghai Dai and Yue Gao. The DeepHypergraph library. In *Hypergraph Computation*, Artificial Intelligence: Foundations, Theory, and Algorithms, pp. 237–240. Springer, Singapore, 2023. doi: 10.1007/978-981-99-0185-2_12.

Inderjit S. Dhillon and Dharmendra S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1):143–175, 2001. doi: 10.1023/A:1007612920971.

Yihe Dong, Will Sawin, and Yoshua Bengio. HNHN: Hypergraph networks with hyperedge neurons. In *Graph Representation Learning and Beyond Workshop at the International Conference on Machine Learning*, 2020. URL https://www.microsoft.com/en-us/research/publication/hnhn-hypergraph-networks-with-hyperedge-neurons/.

Iulia Duta and Pietro Liò. SPHINX: Structural prediction using hypergraph inference network. In *International Conference on Machine Learning*, pp. 14884–14901, 2025. URL https://proceedings.mlr.press/v267/duta25a.html.

Iulia Duta, Giulia Cassarà, Fabrizio Silvestri, and Pietro Liò. Sheaf hypergraph networks. In *Advances in Neural Information Processing Systems*, pp. 12087–12099, 2023. doi: 10.52202/075280-0529.

Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph neural networks. In *AAAI Conference on Artificial Intelligence*, pp. 3558–3565, 2019. doi: 10.1609/aaai.v33i01.33013558.

Richard Forsyth. Zoo. UCI Machine Learning Repository, 1990. URL https://doi.org/10.24432/C5R59V.

Luca Franceschi, Mathias Niepert, Massimiliano Pontil, and Xiao He. Learning discrete structures for graph neural networks. In *International Conference on Machine Learning*, pp. 1972–1982, 2019. URL https://proceedings.mlr.press/v97/franceschi19a.html.

Yue Gao, Yifan Feng, Shuyi Ji, and Rongrong Ji. HGNN$^+$: General hypergraph neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):3181–3199, 2023. doi: 10.1109/TPAMI.2022.3182052.

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.

Jing Huang and Jie Yang. UniGNN: A unified framework for graph and hypergraph neural networks. In *International Joint Conference on Artificial Intelligence*, pp. 2563–2569, 2021. doi: 10.24963/ijcai.2021/353.

Jianwen Jiang, Yuxuan Wei, Yifan Feng, Jingxuan Cao, and Yue Gao. Dynamic hypergraph neural networks. In *International Joint Conference on Artificial Intelligence*, pp. 2635–2641, 2019. doi: 10.24963/ijcai.2019/366.

Xiaojun Kang, Xinchuan Li, Hong Yao, Dan Li, Bo Jiang, Xiaoyue Peng, Tiejun Wu, Shihua Qi, and Lijun Dong. Dynamic hypergraph neural networks based on key hyperedges. *Information Sciences*, 616:37–51, 2022. doi: 10.1016/j.ins.2022.10.006.

Ken Lang. NewsWeeder: Learning to filter netnews. In *International Conference on Machine Learning*, pp. 331–339, 1995. doi: 10.1016/B978-1-55860-377-6.50048-7.

Mengqi Lei, Yihong Wu, Siqi Li, Xinhu Zheng, Juan Wang, Shaoyi Du, and Yue Gao. SoftHGNN: Soft hypergraph neural networks for general visual recognition. *International Journal of Computer Vision*, 134:244, 2026. doi: 10.1007/s11263-026-02841-z.

Fan Li, Xiaoyang Wang, Wenjie Zhang, Ying Zhang, and Xuemin Lin. DHG-Bench: A comprehensive benchmark for deep hypergraph learning. In *International Conference on Learning Representations*, 2026a. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/4f25c9511feb3d6496450d68ca21efb9-Abstract-Conference.html.

Ming Li, Yujie Fang, Yi Wang, Han Feng, Yongchun Gu, Lu Bai, and Pietro Liò. Deep hypergraph neural networks with tight framelets. In *AAAI Conference on Artificial Intelligence*, pp. 18385–18392, 2025a. doi: 10.1609/aaai.v39i17.34023.

Ming Li, Yongchun Gu, Yi Wang, Yujie Fang, Lu Bai, Xiaosheng Zhuang, and Pietro Liò. When hypergraph meets heterophily: New benchmark datasets and baseline. In *AAAI Conference on Artificial Intelligence*, pp. 18377–18384, 2025b. doi: 10.1609/aaai.v39i17.34022.

Ming Li, Yujie Fang, Dongrui Shen, Han Feng, Xiaosheng Zhuang, Kelin Xia, and Pietro Liò. High-pass matters: Theoretical insights and sheaflet-based design for hypergraph neural networks. In *AAAI Conference on Artificial Intelligence*, pp. 23039–23046, 2026b. doi: 10.1609/aaai.v40i27.39469.

Ming Li, Ruiting Zhao, Zihao Yan, Lu Bai, Lixin Cui, and Feilong Cao. HyperAim: Hypergraph contrastive learning with adaptive multi-frequency filters. In *AAAI Conference on Artificial Intelligence*, pp. 23063–23070, 2026c. doi: 10.1609/aaai.v40i27.39472.

Mingyuan Li, Yanlin Yang, Lei Meng, Lu Peng, Haixing Zhao, and Zhonglin Ye. Self-supervised hypergraph structure learning. *Artificial Intelligence Review*, 58:190, 2025c. doi: 10.1007/s10462-025-11199-6.

Jianpeng Liao, Jun Yan, and Qian Tao. DualHGNN: A dual hypergraph neural network for semi-supervised node classification based on multi-view learning and density awareness. In *International Joint Conference on Neural Networks*, pp. 1–10, 2023. doi: 10.1109/IJCNN54540.2023.10191471.

Wei Liu, Junfeng He, and Shih-Fu Chang. Large graph construction for scalable semi-supervised learning. In *International Conference on Machine Learning*, pp. 679–686, 2010. URL https://icml.cc/Conferences/2010/papers/16.pdf.

Zexi Liu, Bohan Tang, Ziyuan Ye, Xiaowen Dong, Siheng Chen, and Yanfeng Wang. Hypergraph transformer for semi-supervised classification. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 7515–7519, 2024. doi: 10.1109/ICASSP48485.2024.10446248.

Fengcheng Lu, Michael Ng, and Andy Yip. Hypergraph neural diffusion networks. *Neural Networks*, 195:108271, 2026. doi: 10.1016/j.neunet.2025.108271.

Shilin Qu, Weiqing Wang, Yuan-Fang Li, Xin Zhou, and Fajie Yuan. Hypergraph node representation learning with one-stage message passing. *Neural Networks*, 193:107973, 2026. doi: 10.1016/j.neunet.2025.107973.

Siddhant Saxena, Shounak Ghatak, Raghu Kolla, Debashis Mukherjee, and Tanmoy Chakraborty. DPHGNN: A dual perspective hypergraph neural networks. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2548–2559, 2024. doi: 10.1145/3637528.3672047.

Charles Stewart, III and Jonathan Woon. Congressional committee assignments, 103rd to 114th congresses, 1993–2017: House. Massachusetts Institute of Technology, 2017. URL https://web.mit.edu/17.251/www/data_page.html.

Bohan Tang, Zexi Liu, Keyue Jiang, Siheng Chen, and Xiaowen Dong. Training-free message passing for learning on hypergraphs. In *International Conference on Learning Representations*, pp. 69491–69519, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/ad9804eed175610302917a0c21ab9b52-Abstract-Conference.html.

*Mushroom*. UCI Machine Learning Repository, 1981. URL https://doi.org/10.24432/C5959T.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. URL https://jmlr.org/papers/v9/vandermaaten08a.html.

Peihao Wang, Shenghao Yang, Yunyu Liu, Zhangyang Wang, and Pan Li. Equivariant hypergraph diffusion neural operators. In *International Conference on Learning Representations*, 2023a. URL https://openreview.net/forum?id=RiTjKoscnNd.

Yuxin Wang, Quan Gan, Xipeng Qiu, Xuanjing Huang, and David Wipf. From hypergraph energy functions to hypergraph neural networks. In *International Conference on Machine Learning*, pp. 35605–35623, 2023b. URL https://proceedings.mlr.press/v202/wang23d.html.

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1912–1920, 2015. doi: 10.1109/CVPR.2015.7298801.

Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, and Partha Talukdar. HyperGCN: A new method for training graph convolutional networks on hypergraphs. In *Advances in Neural Information Processing Systems*, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/1efa39bcaec6f3900149160693694536-Abstract.html.

Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *International Conference on Machine Learning*, pp. 40–48, 2016. URL https://proceedings.mlr.press/v48/yanga16.html.

Donghan Yu, Ruohong Zhang, Zhengbao Jiang, Yuexin Wu, and Yiming Yang. Graph-revised convolutional network. In *European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 378–393, 2021. doi: 10.1007/978-3-030-67664-3_23.

Jiying Zhang, Yuzhao Chen, Xi Xiao, Runiu Lu, and Shu-Tao Xia. Learnable hypergraph laplacian for hypergraph learning. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 4503–4507, 2022. doi: 10.1109/ICASSP43922.2022.9747687.

Yanan Zhao, Yuelong Li, Haichuan Zhang, Vishal Monga, and Yonina C. Eldar. Deep, convergent, unrolled half-quadratic splitting for image deconvolution. *IEEE Transactions on Computational Imaging*, 10:574–588, 2024. doi: 10.1109/TCI.2024.3377132.

Yanan Zhao, Xingchao Jian, Feng Ji, Wee Peng Tay, and Antonio Ortega. Generalized graph signal reconstruction via the uncertainty principle. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1–5, 2025. doi: 10.1109/ICASSP49660.2025.10889406.

Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. Learning with hypergraphs: Clustering, classification, and embedding. In *Advances in Neural Information Processing Systems*, pp. 1601–1608, 2006. URL https://proceedings.neurips.cc/paper/2006/hash/dff8e9c2ac33381546d96deea9922999-Abstract.html.

Jiangnan Zhou, Sheng Zhang, Bing Wu, Qiuming Wang, Chennan Wu, Ziqiang Luo, Ka Sun, and Hongmei Mao. S2-HGNN: Scale-aware hypergraph node classification with spectral inductive bias. *Entropy*, 28(6):592, 2026. doi: 10.3390/e28060592.

Peng Zhou, Zongqian Wu, Xiangxiang Zeng, Guoqiu Wen, Junbo Ma, and Xiaofeng Zhu. Totally dynamic hypergraph neural networks. In *International Joint Conference on Artificial Intelligence*, pp. 2476–2483, 2023. doi: 10.24963/ijcai.2023/275.

# A Extended Related Work

## A.1 Fixed Incidence and Learned Propagation

Classical hypergraph learning starts from domain defined memberships and propagates signals through a normalized hypergraph operator (Zhou et al., 2006). HGNN and hypergraph attention instantiate this construction in neural layers (Feng et al., 2019; Bai et al., 2021), while HNHN and UniGNN generalize the node to hyperedge and hyperedge to node transformations (Dong et al., 2020; Huang & Yang, 2021). HyperGCN derives graph connections from each hyperedge (Yadati et al., 2019). AllSet and ED-HNN instead learn more expressive permutation invariant or permutation equivariant transformations over the supplied memberships (Chien et al., 2022; Wang et al., 2023a). DHG-Bench provides a recent unified evaluation across node, edge, and graph level hypergraph tasks (Li et al., 2026a). Despite their different aggregation rules and computational expansions, the relational support of these methods is derived from the supplied hypergraph rather than learned from node features.

## A.2 Learning Hypergraph Structure

Methods that learn hypergraph structure differ in where they intervene. DHGNN and TDHNN reconstruct hyperedges from evolving node representations (Jiang et al., 2019; Zhou et al., 2023). HSL revises a supplied incidence through task guided sampling, while SPHINX infers a discrete latent hypergraph when no structure is supplied (Cai et al., 2022; Duta & Liò, 2025). DSA-HG learns soft assignments to latent hyperedges and integrates them with the observed hypergraph (Chen et al., 2027). HERALD learns a hypergraph Laplacian and combines it with the observed operator before propagation (Zhang et al., 2022). Other models introduce parallel relational signals. DHKH and DualHGNN preserve original relations alongside learned ones, DPHGNN and S2-HGNN add graph derived or scale derived relations, and HGraphormer couples global attention with hypergraph propagation (Kang et al., 2022; Liao et al., 2023; Saxena et al., 2024; Zhou et al., 2026; Qu et al., 2026). These methods differ in the relations they construct and in where propagated representations interact. PANDA keeps the learned and observed incidences as separate operators and assigns their diffusion schedules before combining the filtered representations.

## A.3 Pairwise Structure Learning and Prototype Factorization

Graph structure learning provides mechanisms for inferring pairwise adjacency, including bilevel optimization and attention based edge sampling (Franceschi et al., 2019; Yu et al., 2021). AnchorGraph constructs $\mathbf{W} = \mathbf{Z}\mathbf{\Lambda}^{-1}\mathbf{Z}^{\top}$ from normalized node to anchor weights, yielding a low rank affinity with storage linear in the number of nodes for a fixed anchor count (Liu et al., 2010). The anchor variant of IDGL introduces a related factorization into iterative graph structure learning (Chen et al., 2020). SoftHGNN learns continuous prototype memberships for visual tokens and propagates through the resulting soft hyperedges (Lei et al., 2026). PANDA likewise uses an $N \times K$ incidence, but retains the supplied hypergraph as a separate operator. Its learned and observed operators receive independent filters before their representations are fused.

## A.4 Diffusion Depth and Spectral Control

APPNP separates a trainable predictor from parameter-free personalized PageRank propagation on graphs (Gasteiger et al., 2019). TF-HNN brings precomputed polynomial filtering to a fixed hypergraph incidence (Tang et al., 2025), while PhenomNN and HNDiffN derive hypergraph propagation from energy minimization and continuous-time diffusion (Wang et al., 2023b; Lu et al., 2026). Work on heterophilic hypergraphs and HyperAim further shows that useful propagation depends jointly on spectral response and diffusion depth (Li et al., 2025b; 2026b;c). PANDA makes this choice separately for learned and observed hypergraph relations. It assigns a shallow polynomial to the globally supported feature-induced operator and a longer restart polynomial to the observed operator, then combines their outputs after filtering.

# B Algorithmic Details and Operator Proofs

## B.1 Prototype Initialization and Reuse

At the beginning of each run, PANDA computes normalized encoded features using the initialized encoder and runs spherical $K$-means once. The resulting centroids initialize the trainable prototype matrix $\mathbf{P}$. This initialization precedes the repeated forward computation in Algorithm 1. Subsequent optimization updates the encoder, $\mathbf{P}$, and the temperature by gradient descent without rerunning $K$-means. Each forward pass computes $\mathbf{H}_\sigma$ once from the current parameters and reuses it in all $D$ layers. The feature-induced incidence is therefore fixed across the layers of one forward pass but evolves across optimization steps.

The observed incidence is constructed once for a dataset configuration. When singleton augmentation is enabled, $\mathbf{H}_{\mathrm{obs}} = [\mathbf{H} \mid \mathbf{I}_N]$. Both normalized operators admit evaluation through their incidence factors, so Algorithm 1 does not require either node-to-node operator to be materialized.

## B.2 Factorized Evaluation and Complexity

Let $I_{\mathrm{km}}$ denote the number of spherical $K$-means iterations, let $\xi = \mathrm{nnz}(\mathbf{H}_{\mathrm{obs}})$, and define $d_\star = \max\{d, d_0, \ldots, d_D\}$. For an incidence-factorized forward pass, the feature encoder costs $O(NFd)$ and prototype assignment costs $O(NKd)$. These costs occur once per forward pass and are absorbed into the final bound because $D, m \geq 1$ and $d \leq d_\star$. Across $D$ layers, projections cost $O(\sum_{\ell=1}^{D} Nd_{\ell-1}d_\ell)$, observed diffusion costs $O(\sum_{\ell=1}^{D} L\xi d_\ell)$, and feature-induced diffusion costs $O(\sum_{\ell=1}^{D} mNKd_\ell)$. With $\rho_\star$ equal to the largest of $d_\star$, $mK$, and $L\xi/N$, one forward pass costs $O(DNd_\star\rho_\star)$. The one-time prototype initialization costs $O(I_{\mathrm{km}}NKd)$.

The factorized algorithm stores $NK$ feature induced incidence entries and $\xi$ observed incidence entries. The timing study uses the equivalent cached normalized operator for the observed branch. If $\eta = \mathrm{nnz}(\mathbf{S}_{\mathrm{obs}})$, that branch costs $O(\sum_{\ell=1}^{D} L\eta d_\ell)$ and stores $\eta$ operator entries. The feature induced branch uses its $N \times K$ incidence factorization. Appendix D.4 measures this computation on fixed cardinality synthetic hypergraphs.

## B.3 Proofs of Operator Properties

We prove the properties used to characterize the two propagation operators in Section 3.1.

*Proof of Proposition 1.* Row normalization gives $\mathbf{H}_\sigma \mathbf{1}_K = \mathbf{1}_N$, so the feature-induced node degree matrix is $\mathbf{I}_N$. Finite softmax logits make every entry of $\mathbf{H}_\sigma$ positive, so every diagonal entry of $\mathbf{D}_e^\sigma$ is also positive. With $\mathbf{Q} = \mathbf{H}_\sigma(\mathbf{D}_e^\sigma)^{-1/2}$, the operator satisfies $\mathbf{S}_\sigma = \mathbf{Q}\mathbf{Q}^\top$. It is therefore symmetric positive semidefinite and has rank at most $K$. Since $\mathbf{H}_\sigma^\top \mathbf{1}_N = \mathbf{D}_e^\sigma \mathbf{1}_K$, it is row stochastic because

$$\mathbf{S}_\sigma \mathbf{1}_N = \mathbf{H}_\sigma(\mathbf{D}_e^\sigma)^{-1}\mathbf{H}_\sigma^\top \mathbf{1}_N = \mathbf{H}_\sigma \mathbf{1}_K = \mathbf{1}_N. \tag{10}$$

Every operator entry is positive because

$$[\mathbf{S}_\sigma]_{ij} = \sum_{k=1}^{K} \frac{[\mathbf{H}_\sigma]_{ik}[\mathbf{H}_\sigma]_{jk}}{[\mathbf{D}_e^\sigma]_{kk}} > 0. \tag{11}$$

Since $\mathbf{S}_\sigma$ itself is strictly positive, the defining condition for primitivity holds at the first power. Row stochasticity fixes its spectral radius at one, and the Perron–Frobenius theorem gives a simple eigenvalue $\lambda_1 = 1$ with $|\lambda_r| < 1$ for every $r > 1$. Symmetry and positive semidefiniteness provide an orthonormal eigenbasis with $0 \leq \lambda_r < 1$ for $r > 1$, while row stochasticity gives $\mathbf{v}_1 = \mathbf{1}_N/\sqrt{N}$. Hence

$$\mathbf{S}_\sigma^t = \frac{1}{N}\mathbf{1}_N\mathbf{1}_N^\top + \sum_{r=2}^{N} \lambda_r^t \mathbf{v}_r\mathbf{v}_r^\top \longrightarrow \frac{1}{N}\mathbf{1}_N\mathbf{1}_N^\top. \tag{12}$$

For $N > 1$, write $\mathbf{\Pi} = N^{-1}\mathbf{1}_N\mathbf{1}_N^\top$ and $\lambda_\star = \max_{r>1} \lambda_r$. The same expansion gives $\|\mathbf{S}_\sigma^t - \mathbf{\Pi}\|_2 = \lambda_\star^t$. Hence $\|\mathbf{S}_\sigma^t\mathbf{U} - \mathbf{\Pi}\mathbf{U}\|_F \leq \lambda_\star^t\|\mathbf{U} - \mathbf{\Pi}\mathbf{U}\|_F$ for any matrix-valued node signal $\mathbf{U}$. The rank bound limits the number of nonzero spectral modes, but does not determine $\lambda_\star$. Its value depends on the complete assignment matrix and therefore on the encoded nodes, prototype geometry, and temperature jointly. □

**Observed-operator spectrum and component support.** For the binary incidence $\mathbf{H}_{\mathrm{obs}}$, define $\mathbf{Q}_{\mathrm{obs}} = (\mathbf{D}_v^{\mathrm{obs}})^{-1/2}\mathbf{H}_{\mathrm{obs}}(\mathbf{D}_e^{\mathrm{obs}})^{-1/2}$ on nodes with positive observed degree, so $\mathbf{S}_{\mathrm{obs}} = \mathbf{Q}_{\mathrm{obs}}\mathbf{Q}_{\mathrm{obs}}^{\top} \succeq 0$. Let $d_i^{\mathrm{obs}} = [\mathbf{D}_v^{\mathrm{obs}}]_{ii}$, $\delta_e = [\mathbf{D}_e^{\mathrm{obs}}]_{ee}$, and $a_i = x_i/\sqrt{d_i^{\mathrm{obs}}}$. For any vector supported on these nodes,

$$\mathbf{x}^{\top}(\mathbf{I} - \mathbf{S}_{\mathrm{obs}})\mathbf{x} = \sum_e \left( \sum_{i\in e} a_i^2 - \frac{1}{\delta_e}\left(\sum_{i\in e} a_i\right)^2 \right) = \sum_e \frac{1}{2\delta_e} \sum_{i,j\in e} (a_i - a_j)^2 \geq 0. \tag{13}$$

Hence $\mathbf{I} - \mathbf{S}_{\mathrm{obs}} \succeq 0$ on the positive-degree subspace, and all eigenvalues of $\mathbf{S}_{\mathrm{obs}}$ lie in $[0, 1]$. The zero-degree convention adds zero rows and columns and therefore only zero eigenvalues. If $\mathbf{x}$ is supported on a connected component $\mathcal{C}$ of $\mathbf{H}_{\mathrm{obs}}$, then $\mathbf{S}_{\mathrm{obs}}^t\mathbf{x}$ remains supported on $\mathcal{C}$ for every $t \geq 0$.

*Proof of Proposition 2.* Let $d_n^{\mathrm{obs}} = 1 + \sum_{e=1}^{M} [\mathbf{H}]_{ne}$ be the observed degree of node $n$ after appending its singleton hyperedge. By construction, $[\mathbf{D}_v^{\mathrm{obs}}]_{nn} = d_n^{\mathrm{obs}} \geq 1$. For the augmented binary incidence,

$$[\mathbf{S}_{\mathrm{obs}}]_{nn} = \frac{1}{d_n^{\mathrm{obs}}} \sum_{e:\, [\mathbf{H}_{\mathrm{obs}}]_{ne}=1} \frac{1}{[\mathbf{D}_e^{\mathrm{obs}}]_{ee}}. \tag{14}$$

The appended singleton has edge degree one, so its term equals $(d_n^{\mathrm{obs}})^{-1}$ and all remaining terms are nonnegative. If node $n$ is isolated in $\mathbf{H}$, its singleton is its only incident edge. Its diagonal operator entry is one and every off-diagonal entry in row $n$ is zero, so that row is the $n$-th standard basis vector. □

# C Experimental Setup and Reproducibility

## C.1 Datasets and Splits

Table 4 lists the statistics and release sources of the 12 benchmarks. The five co-authorship and co-citation hypergraphs follow HyperGCN (Yadati et al., 2019) and are loaded from DHG (Dai & Gao, 2023). Zoo, 20news, mushroom, ntu2012, and modelnet40 use the processed features and incidences released with AllSet (Chien et al., 2022). House and cooking200 use DHG's HouseCommittees and Cooking200 releases, respectively.

Table 4: Hypergraph benchmark statistics. † DHG training mask, with the remaining nodes divided into 500 validation nodes and a disjoint test set. ‡ Random 50/25/25 split following the AllSet protocol. Isolated test nodes have no hyperedge in the preprocessed observed incidence before singleton augmentation. $F$ denotes feature dimension and $C$ denotes the number of classes.

| Dataset | $N$ | $M$ | $F$ | $C$ | Train / Val / Test | Isolated test | Release |
|---|---|---|---|---|---|---|---|
| *Co-authorship / co-citation* | | | | | | | |
| cora_ca† | 2,708 | 970 | 1,433 | 7 | 140 / 500 / 2,068 | 250 (12%) | DHG |
| cora_cc† | 2,708 | 1,483 | 1,433 | 7 | 140 / 500 / 2,068 | 971 (47%) | DHG |
| citeseer_cc† | 3,312 | 1,004 | 3,703 | 6 | 138 / 500 / 2,674 | 1,500 (56%) | DHG |
| pubmed_cc† | 19,717 | 7,531 | 500 | 3 | 78 / 500 / 19,139 | 15,416 (81%) | DHG |
| dblp_ca† | 41,302 | 20,865 | 1,425 | 6 | 1,740 / 500 / 39,062 | 0 | DHG |
| *Benchmarks with 50/25/25 splits* | | | | | | | |
| zoo‡ | 101 | 40 | 16 | 7 | 50 / 25 / 26 | 0 | AllSet |
| 20news‡ | 16,242 | 100 | 100 | 4 | 8,121 / 4,060 / 4,061 | 0 | AllSet |
| mushroom‡ | 8,124 | 287 | 22 | 2 | 4,062 / 2,031 / 2,031 | 0 | AllSet |
| house‡ | 1,290 | 335 | 1 | 3 | 645 / 322 / 323 | 0 | DHG |
| cooking200‡ | 7,403 | 2,750 | 1 | 20 | 3,701 / 1,850 / 1,852 | 0 | DHG |
| *3-D shapes with 50/25/25 splits* | | | | | | | |
| ntu2012‡ | 2,012 | 1,635 | 100 | 67 | 1,006 / 503 / 503 | 0 | AllSet |
| modelnet40‡ | 12,311 | 10,630 | 100 | 40 | 6,155 / 3,077 / 3,079 | 0 | AllSet |

The underlying data sources are distinct from these hypergraph releases. Zoo and Mushroom originate from the UCI repository (Forsyth, 1990; UCI, 1981), and 20news derives from the 20 Newsgroups collection (Lang, 1995). The shape benchmarks derive from the NTU collection (Chen et al., 2003) and ModelNet40 (Wu et al., 2015). House traces to congressional committee assignments collected by Stewart & Woon (2017) and organized as a hypergraph by Chodrow et al. (2021). We use DHG's three-class release. The Cooking200 release represents Yummly dishes as nodes and ingredients as hyperedges.

In the DHG pubmed_cc release used here, the $19{,}717 \times 500$ feature matrix is stored as 8 bit integers and contains only four nonzero entries. Before singleton augmentation, 15,416 of its 19,139 test nodes are also isolated in the observed incidence. These release properties leave little node specific evidence for most test nodes and provide context for the pubmed_cc results in Table 1.

For the five co-authorship and co-citation datasets, we retain the DHG training masks and partition the remaining nodes into a 500-node validation set and a disjoint test set as described in Section 4.1. For the other seven datasets, we generate a $50/25/25$ split with split seed 42 following the AllSet protocol. Our common preprocessing removes hyperedges with fewer than two distinct nodes and merges duplicate hyperedges. The reported hyperedge counts follow this preprocessing. House and cooking200 use a scalar node-degree feature. Dataset-specific learning rates and prototype counts are reported in Table 6.

**Isolated test nodes.** Citeseer_cc has 1,500 isolated nodes among its 2,674 test nodes. In the unaugmented ablation, inverse entries for zero node degrees are set to zero, so $[\mathbf{S}_{\mathrm{obs}}\mathbf{U}]_{n,:} = \mathbf{0}$ at an isolated node $n$. For $L > 1$, the $t = 0$ restart term still contributes $\alpha\mathbf{U}_{n,:}$ to the complete observed branch. With singleton augmentation, Proposition 2 instead gives $[\mathbf{S}_{\mathrm{obs}}\mathbf{U}]_{n,:} = \mathbf{U}_{n,:}$, so every propagated power preserves that node. Table 2 reports the aggregate effect, and Appendix D.2.4 compares datasets with different isolated-node prevalence.

### C.2 PANDA Hyperparameters

Table 5 gives the settings shared across datasets, and Table 6 gives every dataset-specific value. All dataset-specific choices use validation accuracy, and test labels are used only for final evaluation.

Table 5: Hyperparameters shared across the PANDA experiments.

| Hyperparameter | Value |
|---|---|
| *Architecture and diffusion* | |
| Layers $D$ | 2 |
| Default prototype rule | $K = \mathrm{clip}(4C, 8, K_{\max})$ with $K_{\max} = 64$ |
| Temperature $\tau$ | learnable, initialized at 0.5, clipped to $[0.05, 2]$ |
| Spherical $K$-means initialization | 20 iterations |
| Feature-induced diffusion depth $m$ | 2 |
| Feature-induced coefficient $\alpha_p$ | 0.5 |
| Restart coefficient $\alpha$ for $L > 1$ | 0.1 |
| Fusion | one learned scalar gate per layer |
| Gate-logit initialization $q_\ell$ | $-2$ |
| *Training* | |
| Optimizer | Adam |
| Weight decay | $5 \times 10^{-4}$ |
| Dropout | 0.5 |
| Maximum epochs | 500 |
| Early stopping | patience 100 on validation accuracy |
| Learning-rate scheduler | ReduceLROnPlateau with factor 0.5 and patience 20 |
| *Evaluation* | |
| Seeds | $\{42, 123, 456, 789, 1024\}$ |
| Reported accuracy | test accuracy at the best-validation epoch, mean$\pm$standard deviation |

The default prototype rule is used except on pubmed_cc, where validation selects $K = 32$ directly. The prototype cap is increased to 128 on ntu2012 and modelnet40. In the selected two-layer ar-

chitecture, the listed hidden width $d$ is also the prototype-encoder dimension and the intermediate width $d_1$. Before each training run, the prototype vectors are initialized from normalized encoded features using spherical $K$-means (Dhillon & Modha, 2001).

Table 6: Exact validation-selected PANDA configuration for each dataset. Singleton indicates whether one singleton hyperedge is appended for every node.

| Dataset | Hidden width $d$ | Prototypes $K$ | Observed depth $L$ | Singleton | Learning rate |
|---|---|---|---|---|---|
| cora_ca | 256 | 28 | 10 | Yes | 0.01 |
| citeseer_cc | 256 | 24 | 10 | Yes | 0.05 |
| cooking200 | 128 | 64 | 1 | No | 0.05 |
| cora_cc | 256 | 28 | 10 | Yes | 0.01 |
| pubmed_cc | 128 | 32 | 10 | Yes | 0.05 |
| dblp_ca | 128 | 24 | 10 | Yes | 0.001 |
| zoo | 256 | 28 | 10 | Yes | 0.02 |
| 20news | 128 | 16 | 10 | Yes | 0.02 |
| mushroom | 128 | 8 | 10 | Yes | 0.02 |
| ntu2012 | 256 | 128 | 10 | Yes | 0.02 |
| modelnet40 | 128 | 128 | 10 | Yes | 0.01 |
| house | 128 | 12 | 10 | Yes | 0.02 |

### C.3 Baseline Configurations and Reproducibility

#### C.3.1 Baseline Scope

Table 1 compares PANDA with HNNs that propagate over the supplied incidence and methods that construct or refine the propagation structure.

HGNN uses normalized incidence propagation (Feng et al., 2019). HGNN$^+$ and HNHN extend node to hyperedge and hyperedge to node aggregation (Gao et al., 2023; Dong et al., 2020), while HyperGCN constructs a graph approximation of hyperedges (Yadati et al., 2019). UniGCN, UniGAT, UniGIN, and UniSAGE instantiate the unified aggregation framework of Huang & Yang (2021). UniGCNII adds initial residual and identity mapping terms.

AllDeepSets and AllSetTransformer implement learned multiset functions through sum based and attention based aggregation (Chien et al., 2022). ED-HNN uses equivariant diffusion (Wang et al., 2023a), and PhenomNN unfolds hypergraph energy minimization (Wang et al., 2023b). TF-HNN separates fixed message passing from a trainable predictor (Tang et al., 2025). SheafHyperGNN learns incidence dependent restriction maps (Duta et al., 2023). DPHGNN combines spatial and spectral perspectives (Saxena et al., 2024), HyperGT uses transformer based propagation (Liu et al., 2024), and FrameHGNN uses tight framelet operators (Li et al., 2025a).

DHGNN, HERALD, TDHNN, and SHSL are evaluated with the same splits, five seeds, metrics, training budget, and validation based checkpointing used for the other baselines. Their architecture specific components follow the published formulations, and their hyperparameters use the settings recommended in the original papers. When a paper recommends several configurations, the common validation set determines the choice.

Only HERALD and SHSL require dataset-specific memory adaptations. Their dense node-pair constructions cannot complete on the largest inputs within the available memory. HERALD therefore uses observed-incidence HGNN propagation on pubmed_cc, dblp_ca, and 20news. SHSL uses the same fallback on cooking200, pubmed_cc, dblp_ca, 20news, mushroom, and modelnet40. These entries report the adapted form, while all remaining datasets use the original structure-learning computation.

Runs of SheafHyperGNN exceed the available memory on dblp_ca, 20news, mushroom, and modelnet40. FrameHGNN encounters the same constraint on cooking200, pubmed_cc, dblp_ca, 20news, mushroom, and modelnet40. PhenomNN's dense preprocessing on 20news does not complete within the available resources. The DPHGNN and HyperGT runs on dblp_ca and the SHSL run on house

also do not complete. These cells are therefore unreported and excluded from the corresponding method's average rank.

#### C.3.2 Training Protocol

Model training and GPU evaluation use one NVIDIA GeForce RTX 5060 Ti with 16 GB of memory. All experiments use full batch optimization and disjoint training, validation, and test masks. PANDA's dataset specific settings are given in Table 6. Each seed starts a fresh model, optimizer, scheduler, and prototype initialization state. Spherical $K$-means runs once on normalized encoded features at the beginning of training, after which the prototypes are updated by gradient descent.

Every method uses the same data split and seed set, at most 500 epochs, early stopping patience 100, and validation accuracy checkpointing. Table 6 reports PANDA's validation selected settings.

Algorithm 1 gives the forward computation. The selected two layer PANDA has no initial residual term. The measured configuration allocates parameters for an optional deeper residual path, although this path is inactive when $D = 2$. Table 3 includes these parameters. The reported $39.0$ MB peak memory includes the normalized propagation objects used during inference.

## D Supplementary Experiments

### D.1 Additional Performance Evidence

#### D.1.1 Macro-F1 Results

Accuracy weights every node equally and can obscure performance on rare classes. Macro-F1 instead gives equal weight to the F1 score of each class. Table 7 uses the same data splits, five seeds, and validation-selection rule as Table 1. PANDA's values come from a separate five-seed evaluation with the configurations in Table 6.

PANDA obtains the lowest average Macro-F1 rank at $2.67$, compared with $5.25$ for TF-HNN and $5.33$ for UniGCNII. It ranks first on four datasets and among the top two on nine. This top-two coverage matches the accuracy comparison, showing that PANDA's broad advantage persists when each class contributes equally.

PANDA ranks first on cora_cc and dblp_ca and second on cora_ca and citeseer_cc. It also leads on cooking200 and mushroom, while ranking second on both 3D shape datasets and house. On cooking200, PANDA reaches $12.5\%$ macro averaged F1, compared with at most $9.1\%$ for the baselines. This $3.4$ point margin extends the gain to class balanced performance when node degree is the only input feature. The absolute scores also identify cooking200 as difficult for every evaluated model.

Both performance tables use dense ranks computed from means rounded to the displayed one decimal place. Tied means receive the same rank, and the next distinct mean receives the next integer rank. Average ranks exclude unavailable results, so the number of evaluated datasets can differ across methods. Best and second-best markings use the same displayed means.

#### D.1.2 Cross-Dataset Statistical Comparison

We treat datasets as the independent experimental units and restrict the comparison to the 17 methods with accuracy results on all 12 benchmarks. A Friedman test rejects equal performance across methods ($\chi^2(16) = 88.80$, $p = 4.17 \times 10^{-12}$). We then compare PANDA with each of the 16 complete baselines using exact two-sided Wilcoxon signed-rank tests on the reported dataset means and apply Holm correction across all comparisons. PANDA remains significant against every complete baseline at the $0.05$ level. Against TF-HNN, PANDA wins, ties, and loses on $9$, $1$, and $2$ datasets, respectively, with a mean paired gain of $3.87$ points and adjusted $p = 0.0254$. Against UniGCNII, the corresponding counts are $11$, $0$, and $1$, with a mean paired gain of $3.70$ points and adjusted $p = 0.0146$. Figure 3 shows the paired effect sizes and their distribution across datasets.

Table 7: Macro-F1 in percent over five seeds. Bold and underlined entries denote the best and second-best displayed means. The final column is the mean dense rank over available datasets. PANDA uses the separate evaluation series described in Appendix D.1.

| Method | cora_ca | citeseer_cc | cooking200 | cora_cc | pubmed_cc | dblp_ca | zoo | 20news | mushroom | ntu2012 | modelnet40 | house | Avg rank↓ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *Classic HGNN* | | | | | | | | | | | | | |
| HGNN | 58.9±1.0 | 32.4±0.2 | 4.1±0.6 | 45.7±1.0 | 18.6±3.7 | 87.5±0.3 | 23.9±10.6 | 79.4±0.1 | 98.2±0.5 | 81.4±1.3 | 89.2±0.6 | 38.9±9.4 | 11.83 |
| HGNN$^+$ | 59.1±0.8 | 33.1±0.6 | 6.1±0.4 | 45.6±1.3 | 16.5±4.1 | 86.9±0.3 | 42.3±8.1 | 79.2±0.4 | 99.6±0.2 | 82.2±0.5 | 89.1±0.2 | 30.8±4.9 | 11.33 |
| HNHN | 47.0±3.9 | 23.2±2.8 | 5.1±0.9 | 37.9±2.0 | 16.6±4.2 | 71.2±10.8 | 32.9±17.5 | 68.7±9.9 | 91.3±2.1 | 82.0±1.0 | 88.9±0.3 | 33.7±6.3 | 15.08 |
| HyperGCN | 52.6±0.4 | 54.0±0.6 | 2.8±0.1 | 57.6±0.9 | 18.8±0.0 | 78.2±0.8 | 86.5±6.7 | **80.4±0.1** | 99.9±0.0 | 80.0±1.6 | 94.8±0.2 | **51.6±1.7** | 7.42 |
| *UniGNN family* | | | | | | | | | | | | | |
| UniGCN | 61.4±1.0 | 32.4±0.1 | 7.8±1.1 | 46.1±0.2 | 18.5±3.6 | 84.6±0.6 | 65.9±2.1 | 79.5±0.2 | 99.8±0.1 | 81.1±1.2 | 89.0±0.2 | 49.4±7.4 | 9.42 |
| UniGAT | 56.5±1.7 | 32.9±0.6 | 4.1±0.6 | 48.6±0.7 | 22.7±3.3 | 85.8±0.3 | 71.4±7.6 | 79.6±0.2 | 99.7±0.0 | 81.0±1.3 | 89.0±0.3 | 47.1±3.7 | 8.83 |
| UniGIN | 64.5±1.5 | 49.4±1.8 | 9.1±0.4 | 55.3±2.2 | 21.3±2.0 | 84.1±0.4 | **95.4±9.1** | 79.9±0.2 | 99.8±0.1 | 82.3±0.5 | 90.1±0.3 | 35.1±6.1 | 6.83 |
| UniSAGE | 64.3±1.1 | 49.3±1.1 | 9.1±0.4 | 56.1±1.3 | 20.7±1.9 | 84.9±0.1 | 94.4±8.9 | 80.0±0.4 | 99.8±0.1 | 82.3±0.5 | 90.1±0.3 | 36.8±7.0 | 6.75 |
| *Set-function methods* | | | | | | | | | | | | | |
| AllSetTransformer | 12.0±1.5 | 15.4±7.0 | 1.5±0.1 | 8.2±1.4 | 18.9±0.2 | 86.0±2.0 | 35.2±9.0 | 79.7±0.2 | 99.6±0.2 | 69.6±2.1 | 94.2±0.2 | 38.0±4.7 | 12.83 |
| AllDeepSets | 13.6±1.2 | 14.1±1.9 | 1.5±0.0 | 14.9±1.6 | 18.9±0.1 | 34.3±9.8 | 12.5±5.2 | 74.2±4.6 | 98.9±1.1 | 1.6±0.6 | 5.1±1.9 | 34.2±0.0 | 16.83 |
| ED-HNN | 8.2±1.5 | 24.0±6.4 | 1.6±0.4 | 15.9±4.1 | **22.9±3.5** | 16.2±8.5 | 12.0±4.1 | 64.1±17.8 | 98.3±1.3 | 0.6±0.5 | 4.4±2.8 | 43.3±7.9 | 16.00 |
| *Deep & energy-based HNNs* | | | | | | | | | | | | | |
| UniGCNII | 69.1±1.4 | 56.7±1.1 | 2.5±0.3 | 65.3±0.3 | 20.8±2.5 | 85.1±1.0 | 83.8±5.7 | 80.0±0.1 | 99.8±0.0 | 84.7±0.7 | 96.9±0.1 | 44.5±0.6 | 5.33 |
| PhenomNN | **72.8±0.6** | 57.3±1.1 | 1.7±0.4 | 67.0±0.9 | 17.6±3.1 | 83.6±1.2 | 10.9±2.2 | - | 92.3±0.3 | 73.8±1.5 | 95.9±0.3 | 45.5±6.8 | 10.09 |
| *Spectral & framelet HNNs* | | | | | | | | | | | | | |
| TF-HNN | 68.4±1.0 | 54.5±0.9 | 1.6±0.1 | 63.5±0.9 | 19.6±0.6 | 84.7±0.1 | 72.6±13.4 | 80.1±0.2 | 99.8±0.1 | **87.2±0.9** | **97.4±0.2** | 47.8±5.4 | 5.25 |
| SheafHyperGNN | 66.5±2.7 | 52.7±0.6 | 2.2±0.6 | 61.3±2.7 | 20.0±1.9 | - | 86.0±7.5 | - | - | 85.5±1.4 | - | 41.7±9.2 | 7.25 |
| FrameHGNN | 48.2±3.1 | 33.1±1.7 | - | 37.4±4.2 | - | - | 9.0±0.0 | - | - | 57.3±3.0 | - | 47.0±3.0 | 16.17 |
| *Attention & hybrid HNNs* | | | | | | | | | | | | | |
| DPHGNN | 59.6±1.7 | 39.2±6.3 | 4.3±1.5 | 57.2±4.5 | 19.4±1.3 | - | 19.4±9.2 | 77.8±1.0 | 99.4±0.3 | 57.0±16.0 | 59.6±16.0 | 45.7±3.6 | 11.73 |
| HyperGT | 49.4±0.7 | 46.6±1.9 | 1.6±0.1 | 49.2±1.1 | 18.8±0.0 | - | 50.2±5.7 | 79.2±0.2 | 98.1±1.6 | 64.7±2.4 | 93.3±0.4 | 46.0±9.8 | 11.82 |
| *Structure-learning methods* | | | | | | | | | | | | | |
| DHGNN | 66.1±0.5 | **61.8±1.2** | 1.6±0.1 | 66.1±0.5 | 11.5±0.0 | 80.9±0.2 | 52.5±6.7 | 78.2±0.1 | 99.5±0.1 | 80.7±0.6 | 94.2±0.1 | 36.1±5.1 | 9.83 |
| HERALD | 64.1±2.0 | 34.8±0.7 | 4.6±0.9 | 52.7±0.6 | 16.5±4.1 | 87.5±0.4 | 40.6±14.9 | 79.5±0.1 | 98.8±0.3 | 83.5±0.4 | 89.8±0.2 | 45.1±7.0 | 9.67 |
| TDHNN | 4.5±1.1 | 5.8±0.1 | 1.4±0.2 | 4.5±1.1 | 18.7±0.0 | 6.5±0.8 | 20.7±6.4 | 79.7±0.2 | 98.7±0.1 | 8.9±7.6 | 55.5±24.8 | 34.2±0.0 | 16.83 |
| SHSL | 47.4±4.5 | 52.0±1.6 | 5.6±0.3 | 47.4±4.5 | 11.5±0.0 | 87.7±0.2 | 43.6±2.7 | 79.3±0.2 | 99.1±0.3 | 74.2±1.2 | 89.9±0.2 | - | 10.64 |
| PANDA (ours) | 70.1±0.5 | 58.2±0.2 | **12.5±0.2** | **68.0±1.5** | 21.1±1.2 | **88.3±0.3** | 93.9±7.8 | 78.8±0.8 | **100.0±0.1** | 87.0±0.6 | 97.3±0.3 | 50.6±5.5 | **2.67** |

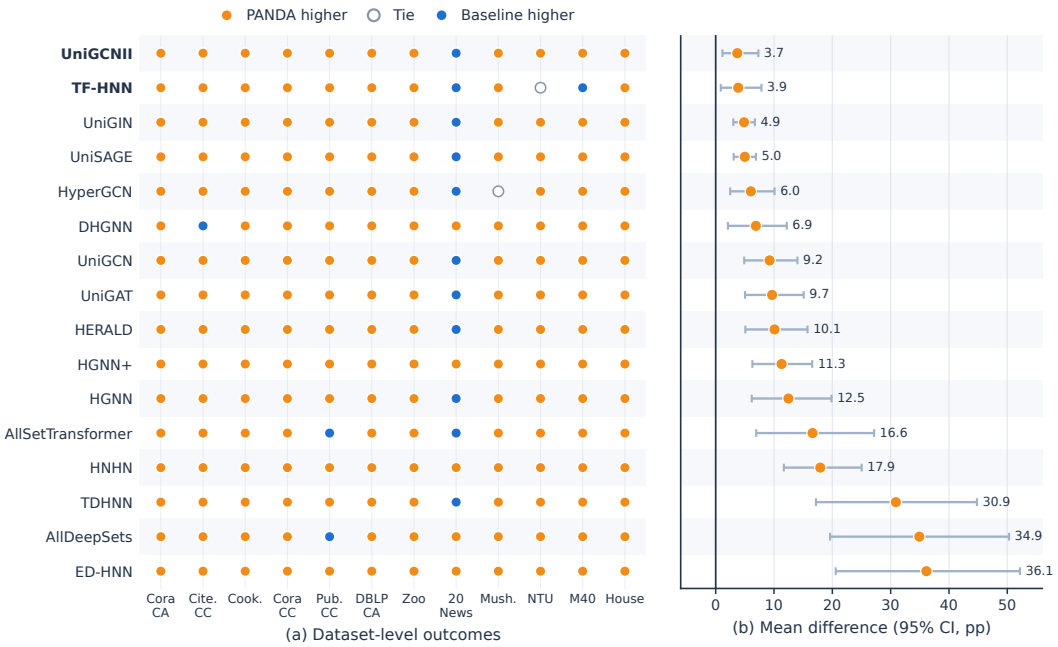



Figure 3: Paired accuracy effects of PANDA against 16 complete baselines on 12 benchmarks. (a) Circles mark datasets on which PANDA has higher, equal, or lower displayed accuracy. (b) Points give the mean paired difference, and bars show 95% bootstrap intervals from 20,000 dataset resamples. Baselines are ordered by mean difference. All Wilcoxon comparisons remain significant after Holm correction at the 0.05 level.

## D.2 Component and Hyperparameter Analysis

### D.2.1 Component Contributions Across Datasets

Table 2 establishes the contribution of both relational views in a six-dataset controlled comparison. Figure 4 extends the same five removal costs to the 11 datasets whose selected configuration includes all components. The observed view contributes 4.5 points on average and improves ten datasets, while the feature-induced view contributes 1.9 points and improves ten. Their largest gains are 13.6 on cora_ca and 6.2 on zoo. Singleton augmentation, the fusion gate, and the selected multi-step observed branch contribute 1.5, 1.1, and 0.9 points on average. The dominant component varies across benchmarks, but both relational views contribute broadly.

Each cell is the full-model accuracy in Table 1 minus the corresponding five-seed ablation mean. The one-step comparison changes both the depth and restart terms of the observed filter. Table 8 extends this analysis across the one-step baseline and several restart depths. Cooking200 is excluded because its selected configuration already uses one-step propagation without singleton augmentation.

### D.2.2 Hyperparameter Sensitivity

Figure 5 tests whether PANDA depends on a narrow prototype resolution or observed receptive field. Across the 16-fold range from $K = 8$ to $K = 128$, accuracy varies by 1.4 points on cora_ca and 1.1 points on citeseer_cc. Changing $L$ from 1 to 20 produces ranges of 0.4 and 0.8 points. Neither sweep shows a monotonic gain, and both contain broad regions with similar performance. On these datasets, PANDA is therefore stable across substantial changes in relation capacity and observed diffusion depth.

The sweeps use the five seeds $\{42, 123, 456, 789, 1024\}$, hidden width 128, and learning rate 0.02. One quantity changes at a time around a common analysis configuration. This controlled configuration is separate from the validation-selected settings used in Table 1.

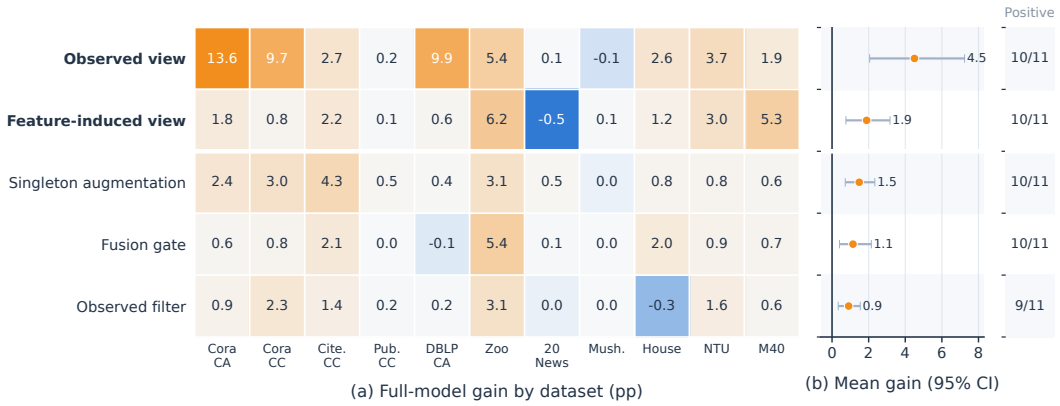



Figure 4: Component contributions across 11 datasets. Panel (a) reports full model accuracy minus each ablation, averaged over five seeds. Multi step observed compares the selected restart branch with one step observed propagation. Panel (b) reports the mean contribution with a 95% bootstrap interval from 20,000 dataset resamples. The adjacent count uses the unrounded dataset means.

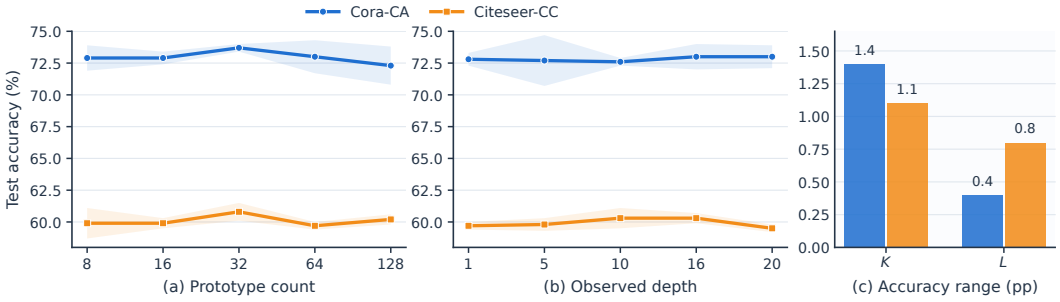



Figure 5: Hyperparameter sensitivity. (a) Prototype count and (b) observed diffusion depth. Curves and bands show mean accuracy and one standard deviation over five seeds. (c) Accuracy range across each sweep. The remaining settings are held fixed within a sweep.

### D.2.3 Observed Diffusion Depth

Table 8 compares the one-step HGNN filter at $L = 1$ with restart diffusion at $L \in \{2, 5, 10, 20\}$ while holding all remaining settings fixed within each dataset. For $L > 1$, the observed branch follows Eq. (7), $\alpha \sum_{t=0}^{L} (1-\alpha)^t \mathbf{S}_{\text{obs}}^t \mathbf{U}^{(\ell)}$. This controlled series is separate from the validation-selected runs in Table 1. Accuracy is not monotonic in $L$. The $L = 10$ setting is best on cora_ca and citeseer_cc and tied for the best result on modelnet40, showing that longer observed propagation is useful without requiring the largest tested depth.

Table 8: Observed diffusion depth. The $L = 1$ column uses one-step HGNN smoothing, while $L > 1$ uses the restart filter in Eq. (7). Bold marks the best setting per dataset, with ties marked within $0.05$ pp.

| Dataset | $L = 1$ | $L = 2$ | $L = 5$ | $L = 10$ | $L = 20$ |
|---|---|---|---|---|---|
| cora_ca | 72.5±1.1 | 69.8±2.2 | 71.6±1.0 | **73.0±1.1** | 71.5±0.8 |
| citeseer_cc | 59.8±0.8 | 59.1±1.6 | 59.2±2.1 | **61.4±0.9** | 60.7±0.5 |
| modelnet40 | 97.6±0.8 | 98.0±0.4 | 98.0±0.2 | **98.1±0.2** | **98.1±0.2** |

### D.2.4 Singleton Augmentation Across Isolation Regimes

The six-dataset ablation in Table 2 evaluates singleton augmentation under the same result series as the main comparison. Removing singleton edges costs $4.3$ points on citeseer_cc and $2.4$ on cora_ca, where $56\%$ and $12\%$ of test nodes are isolated in the observed incidence. The corresponding reduction is $0.6$ point on modelnet40, which has no isolated test nodes. The contrast is consistent with Proposition 2: singleton edges preserve isolated nodes and can also alter normalization for nodes that already belong to observed hyperedges.

## D.3 Hyperedge Removal and Rewiring

Figure 6 contrasts hyperedge removal with cardinality-preserving rewiring. Removal deletes communication paths, while rewiring replaces their memberships with distinct sampled nodes. PANDA degrades more slowly than HGNN under both perturbations. Rewiring is more damaging than removal for PANDA and UniGCNII, separating the effect of misleading memberships from missing paths.

On cora_ca, PANDA reaches $56.5\%$ at full removal and $45.3\%$ at full rewiring. UniGCNII reaches $59.1\%$ and $50.0\%$, respectively, and the same ordering appears on citeseer_cc. The variant without the observed view remains constant because its forward map is independent of the supplied incidence. At full rewiring, this variant exceeds the complete model on both datasets. The crossover marks the point at which the supplied memberships no longer improve the feature-induced route under the current layer-wise gate.

Removal deletes a fraction $p$ of hyperedges. Rewiring selects the same fraction and samples distinct replacement nodes without replacement, preserving every selected hyperedge's cardinality. At full removal, one singleton edge is retained so that the otherwise empty hypergraph can be instantiated.

The figure reports five runs with seeds $\{42, 123, 456, 789, 1024\}$. Each run samples an independent perturbed hypergraph, retrains the model, and selects its checkpoint by validation accuracy. The bands therefore reflect variation from both relation corruption and model training. These controlled runs use fixed analysis configurations separate from Table 1.

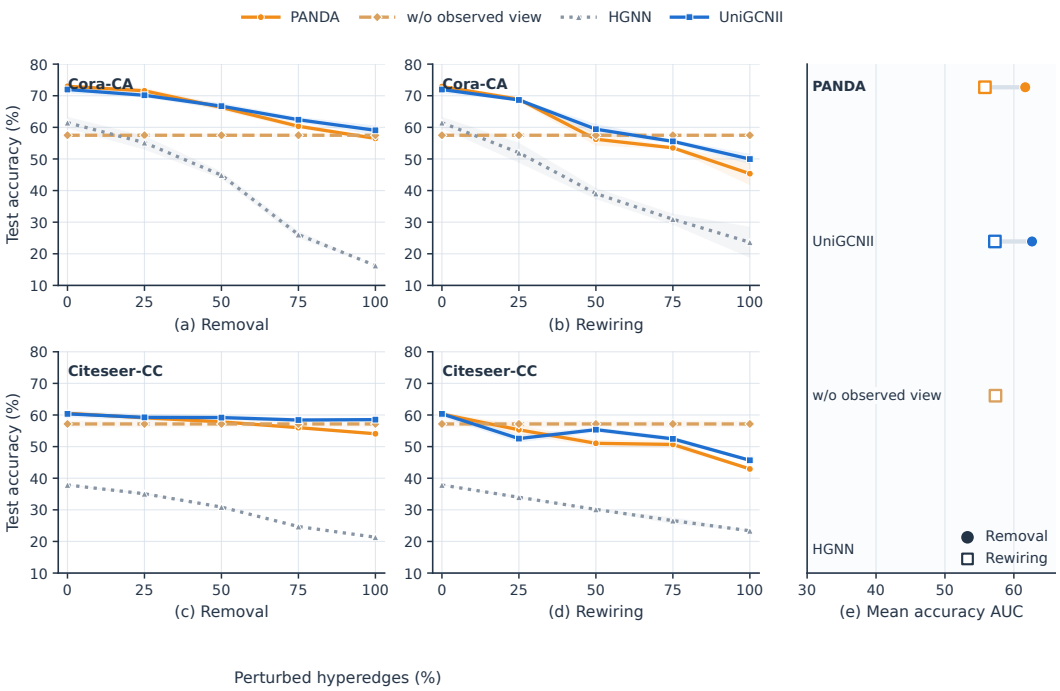



Figure 6: Accuracy under observed-hyperedge perturbations. Panels (a–d) remove hyperedges or rewire their memberships on cora_ca and citeseer_cc. Shading denotes one standard deviation over five independent perturbation and training runs. Panel (e) integrates accuracy over the corruption range and averages the area across both datasets. Filled circles denote removal and open squares rewiring.

### D.4 Controlled Inference Scaling

Figure 7 tests the computational consequence of PANDA's factorized incidence. Increasing the graph from 1,000 to 32,000 nodes multiplies the mean forward time by 34.6 as the node count grows by 32. A log-log fit gives an empirical exponent of 1.03. This controlled result supports the near-linear dependence on $N$ predicted by the complexity analysis in Appendix B.2 when feature width, prototype count, and hyperedge density are fixed.

For each $N \in \{1{,}000, 2{,}000, 4{,}000, 8{,}000, 16{,}000, 32{,}000\}$, we generate $N/2$ hyperedges containing eight distinct uniformly sampled nodes. Features and hidden representations have width 128. PANDA uses $K = 32$, two layers, $m = 2$, and $L = 10$. The synthetic inputs use seed 42.

We measure evaluation-mode CPU forward passes with four PyTorch threads after prototype initialization and five warm-up calls. Each point averages 20 calls. The experiment isolates computational scaling and is separate from the GPU comparison in Table 3.

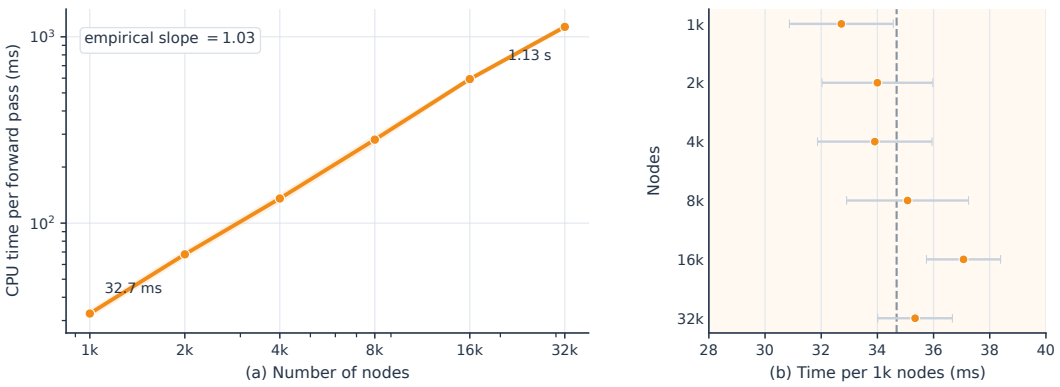



Figure 7: Controlled CPU inference scaling of PANDA with four threads. (a) Mean forward time as node count varies. (b) The same measurements normalized per 1,000 nodes. Feature width, hidden width, prototype count, and hyperedge cardinality remain fixed. Uncertainty denotes one standard deviation over 20 forward passes.

## D.5 Prediction and Propagation Visualizations

### D.5.1 Comparisons Across Datasets and Algorithms

The prediction projections show how class ambiguity appears in the output distributions. We compare PANDA with HGNN, TF-HNN, and UniGCNII on six datasets spanning citation data, text, categorical attributes, degree features, and shape descriptors. TF-HNN and UniGCNII have the next two best average accuracy ranks in the main comparison. The panels use independently trained seed 42 models selected by validation accuracy, and each reports accuracy on the full test set.

For each model, t-SNE (van der Maaten & Hinton, 2008) projects the predicted class-probability vectors of the same test nodes. We include all test nodes up to 1,500 and otherwise use a fixed subset of that size. The projection uses PCA initialization, perplexity 35, automatic learning rate, 1,000 iterations, and seed 42. Colors denote ground-truth classes and remain fixed across algorithms within a dataset. Each panel has an independent t-SNE fit, so only its internal class mixing is interpretable.

Branches dominated by one class correspond to confident output regions, while mixed central regions expose ambiguous predictions. PANDA records the highest accuracy among the four analysis runs on citeseer_cc and house and remains close to the leader on cora_ca, 20news, mushroom, and ntu2012. Table 1 provides the five seed quantitative comparison.

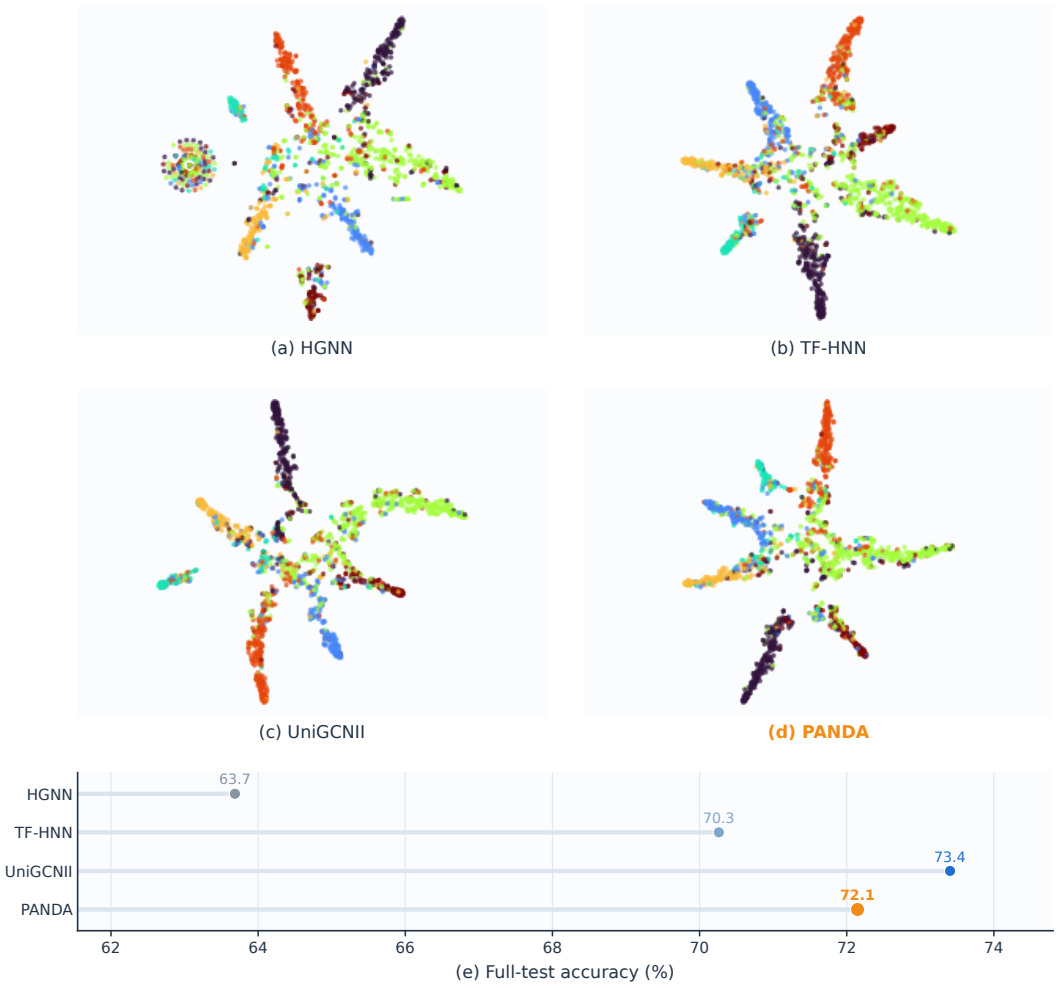



Figure 8: Test-prediction projections on cora_ca. Panels (a–d) compare HGNN, TF-HNN, UniGCNII, and PANDA using the same test nodes. Colors identify classes. Panel (e) reports full-test accuracy for the independent seed-42 analysis runs.

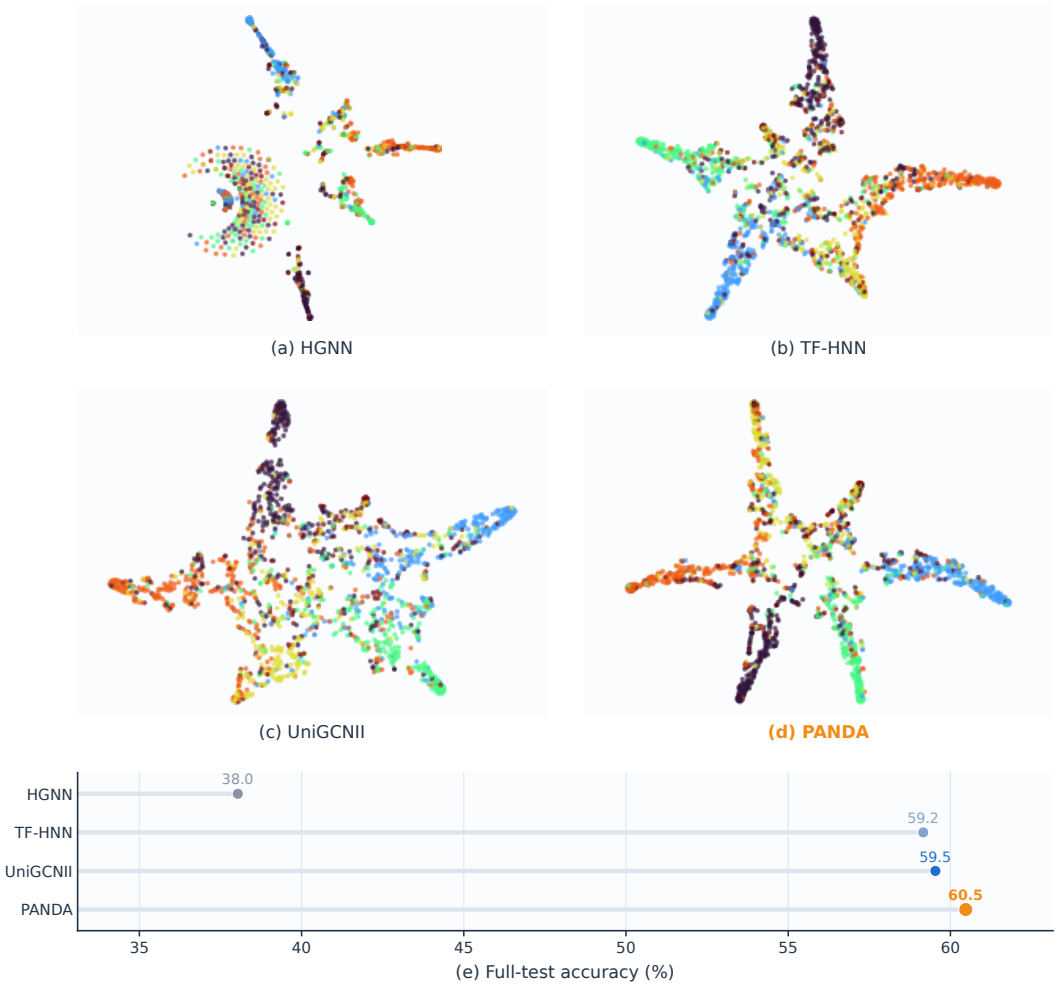



Figure 9: Test-prediction projections on citeseer_cc. Colors identify the same classes across algorithms. The models and projection settings follow Figure 8. PANDA has the highest test accuracy among these four analysis runs.

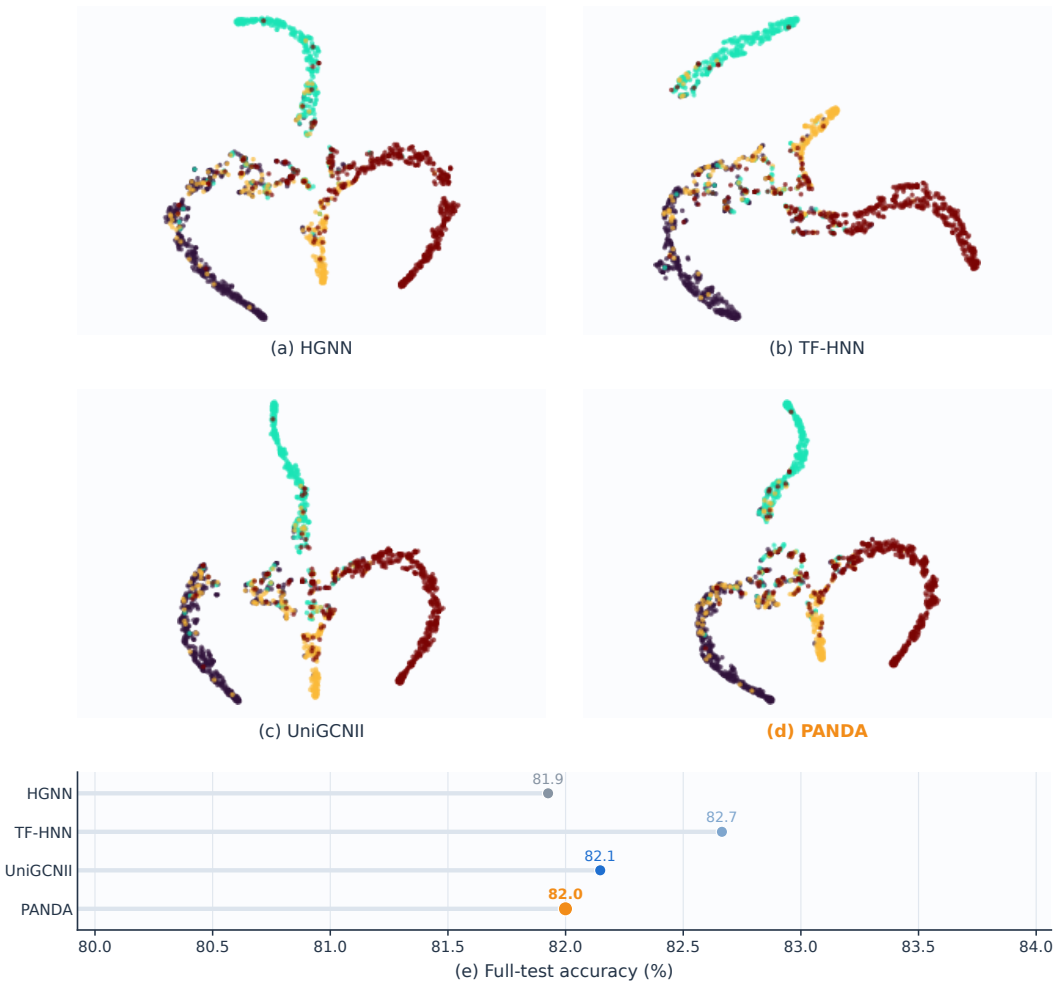



Figure 10: Test-prediction projections on 20news, using the same 1,500 sampled test nodes for each algorithm. Colors identify the four labels in this benchmark release. TF-HNN has the highest test accuracy among these four analysis runs.

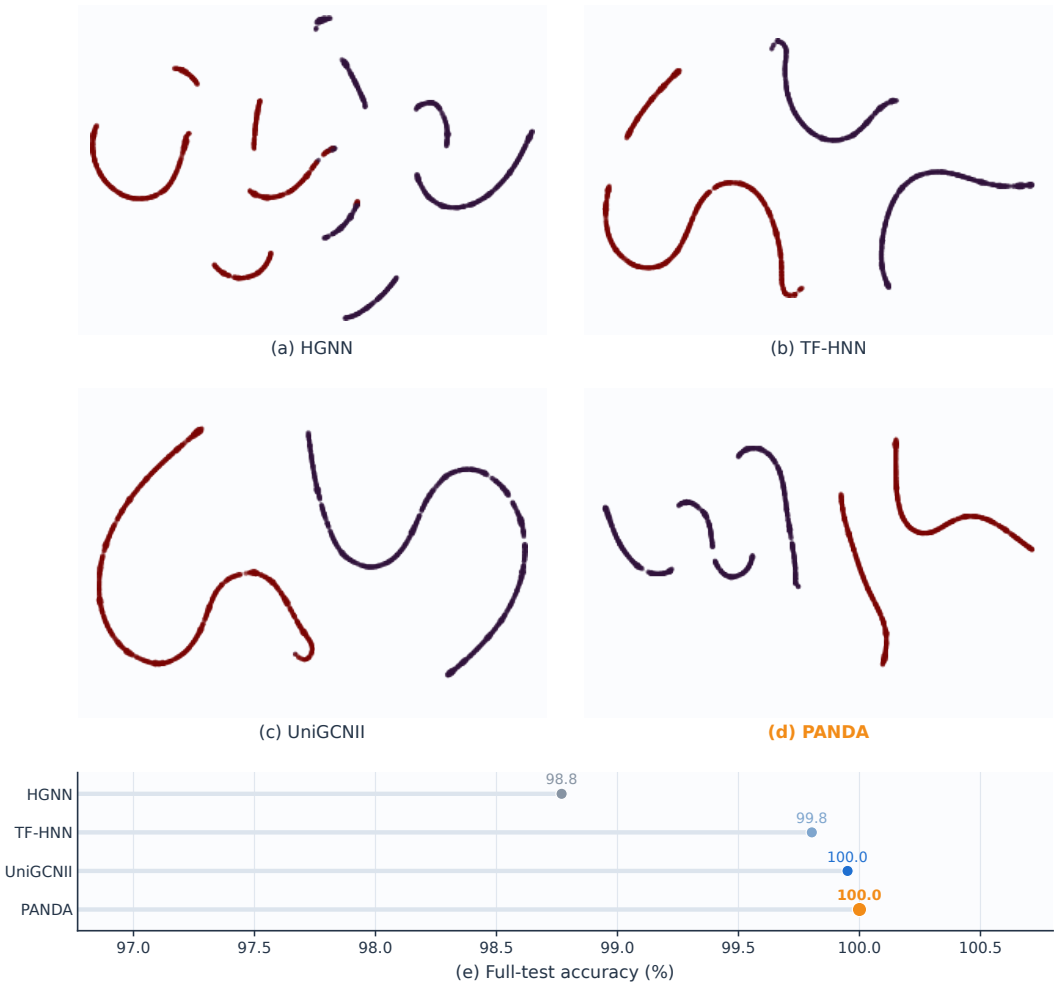



Figure 11: Test-prediction projections on mushroom, using the same 1,500 sampled test nodes for each algorithm. Colors distinguish its two classes. The projections group similar predicted class probabilities and do not measure separation in the raw feature space.

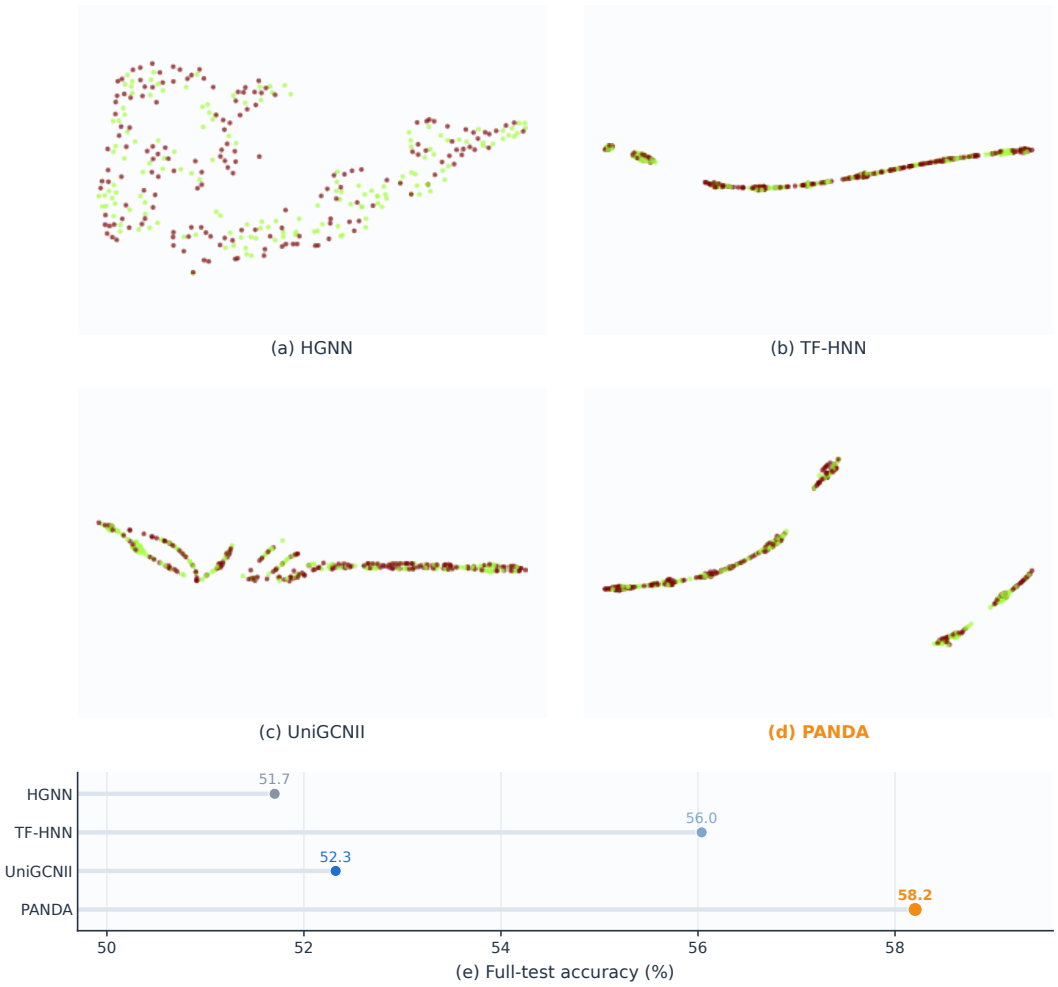



Figure 12: Test-prediction projections on house. All methods use the same scalar degree feature and test nodes. PANDA has the highest accuracy among these four analysis runs, while class overlap remains visible.

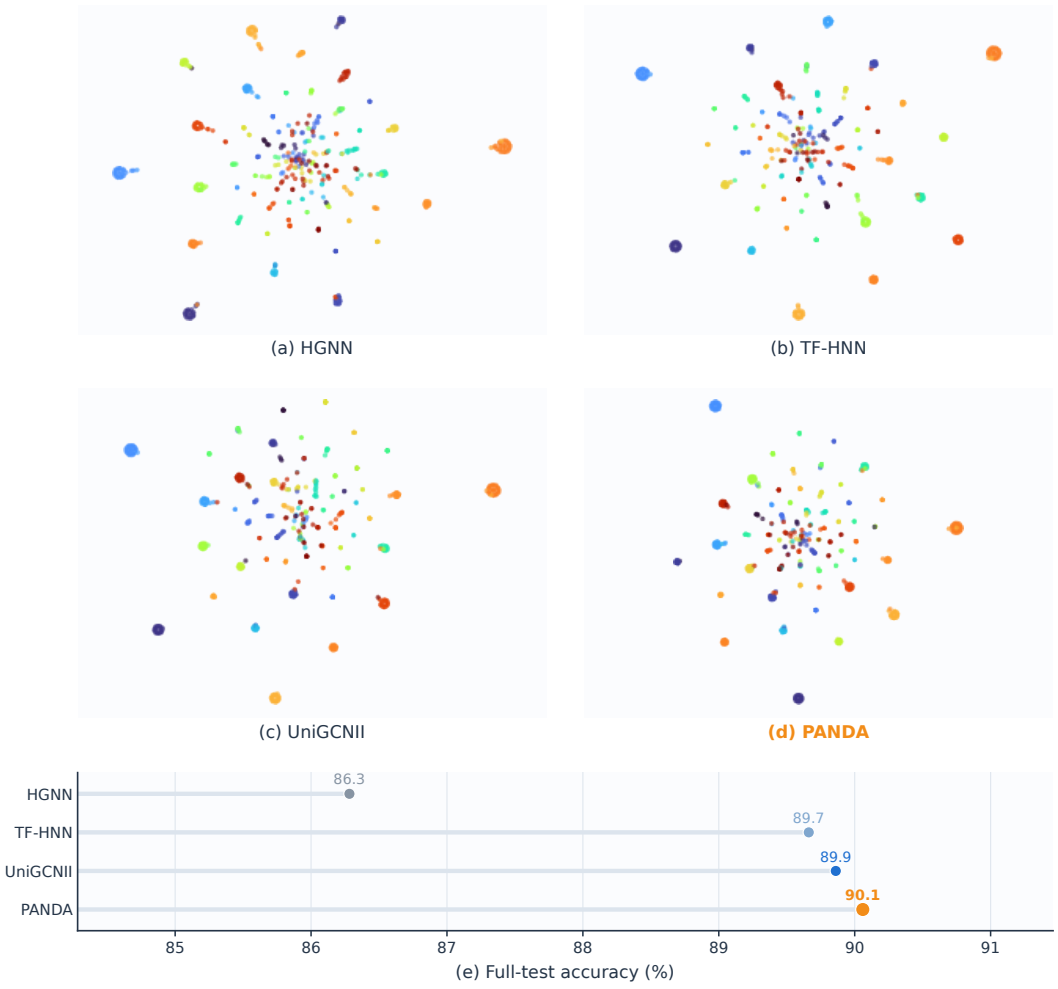



Figure 13: Test-prediction projections on ntu2012. The input features are shape descriptors. All 67 classes are retained, with identical class colors across algorithms. The quantitative comparison is given by each panel's full-test-set accuracy.

### D.5.2 What the Prototypes Represent

Figure 14 tests whether learned prototypes necessarily coincide with class centers. House retains nearly the same label composition across its highest mass prototypes, whereas mushroom forms more label selective memberships. Across all datasets, the soft label sharing score ranges from 0.13 on modelnet40 to 1.00 on pubmed_cc, cooking200, and house. On house and cooking200, the bias free linear encoder maps the positive scalar degree features to a common direction, which makes cosine assignment invariant to feature magnitude. Pubmed_cc has nearly empty input features as documented in Appendix C.1. The assignments are therefore nearly identical across nodes on these three datasets, and the resulting $\mathbf{S}_\sigma$ approaches $\mathbf{\Pi}$. On the remaining datasets, the learned incidence can align with labels when supported by the feature geometry without requiring every prototype to represent one class.

For each dataset, we form the joint distribution of test labels and prototypes from the soft node assignments. Panels (a) and (b) show up to 12 prototypes with the greatest assignment mass. Circle area gives the joint assignment mass for one prototype and label, so each column simultaneously reveals prototype mass and label composition. Panel (c) reports the mass-weighted conditional label entropy normalized by the entropy of the test labels. Zero corresponds to label-pure prototypes, while one means that prototype membership preserves the dataset's overall label diversity. Figure 16 complements this soft diagnostic with the normalized mutual information of hard prototype assignments.

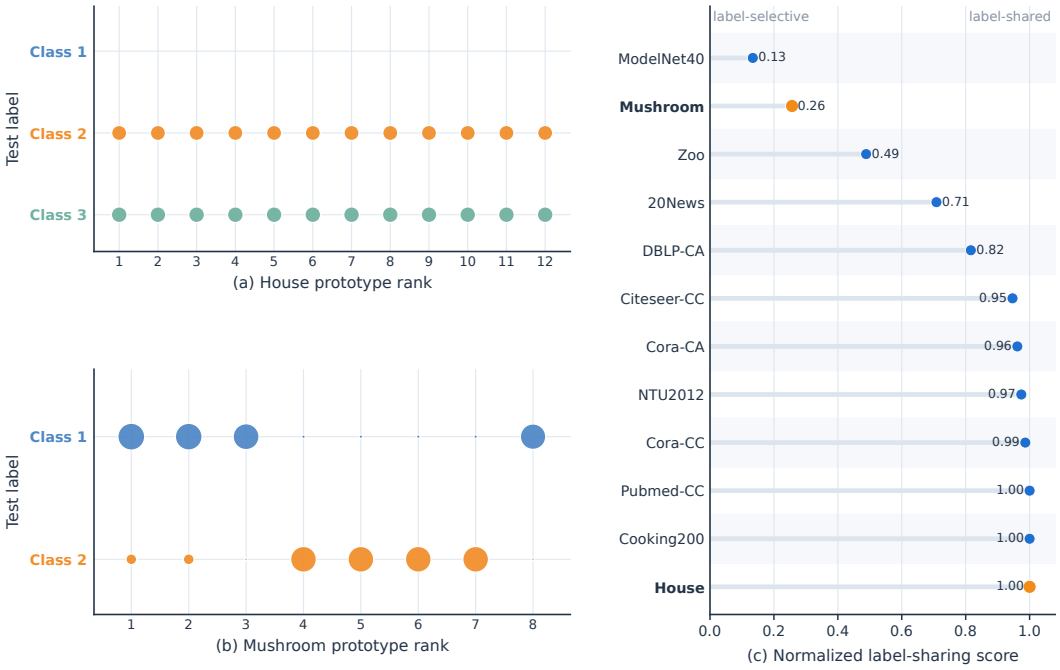



Figure 14: Prototype label composition and class sharing. Panels (a) and (b) show up to 12 prototypes with the greatest mass. Rows denote dataset classes, and circle area gives joint soft assignment mass on the test split. A class absent from the test split has zero displayed mass. Panel (c) reports assignment mass weighted conditional label entropy normalized by test label entropy across 12 datasets. Orange points identify the two datasets expanded in the composition panels.

### D.5.3 Contraction Under Repeated Propagation

Figure 15 measures how repeated applications of the two operators contract centered node variation. The feature induced curves show a sharp reduction after the first application followed by dataset dependent decay. The initial reduction reflects projection onto a subspace of rank at most $K$, while the later slope is governed by the nonprincipal spectrum in Proposition 1. House reaches the plotting floor after one application and cooking200 after two. The default $m = 2$ includes the first two powers shown in this diagnostic.

We obtain each curve by repeatedly applying a saved operator to the same matrix of 32 independent standard Gaussian probes. The vertical quantity is the Frobenius norm after removing the node-wise mean, divided by its value before propagation. Isolating the operators from restart terms, nonlinearities, feature projections, and learned gates makes the curves measurements of operator powers. PANDA uses the finite filters in Eq. (6) and Eq. (7), and Proposition 1 characterizes the limiting feature-induced behavior.

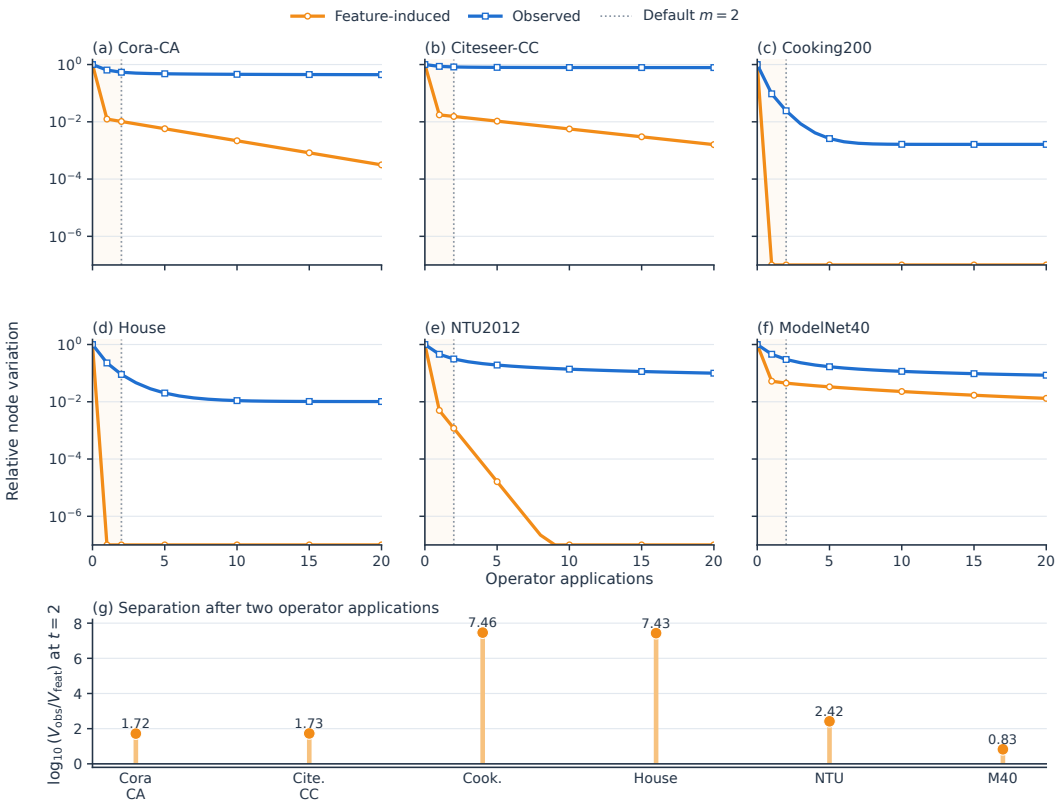



Figure 15: Contraction under repeated operator application. Panels (a) to (f) show relative centered node variation for the feature induced and observed operators using identical Gaussian probes. The dotted line marks the default feature induced depth $m = 2$, and values below $10^{-7}$ appear at the plotting floor. Panel (g) reports the base ten logarithm of the observed to feature variation ratio after two applications.

### D.5.4 Fusion Weights and Prototype Alignment

Figure 16 shows that fusion varies across both datasets and layers. The feature-induced coefficient ranges from nearly zero on cooking200 to $0.62$ in the first citeseer_cc layer. Citeseer_cc then reduces this weight to $0.27$ in the second layer, while modelnet40 keeps similar weights near $0.4$. A shared incidence can therefore support different mixtures at successive representation levels.

The coefficient $g_\ell$ weights the feature-induced output in Eq. (8) and is shared across nodes. Modelnet40 reaches $0.87$ prototype-label NMI, while several datasets obtain strong accuracy with much lower alignment. Across the 12 checkpoints, NMI and the mean gate weight have a Spearman correlation of $0.41$ with $p = 0.18$. Gate magnitude, prototype alignment, and removal cost therefore capture related but distinct properties. Figure 4 supplies the corresponding predictive contribution of each component.

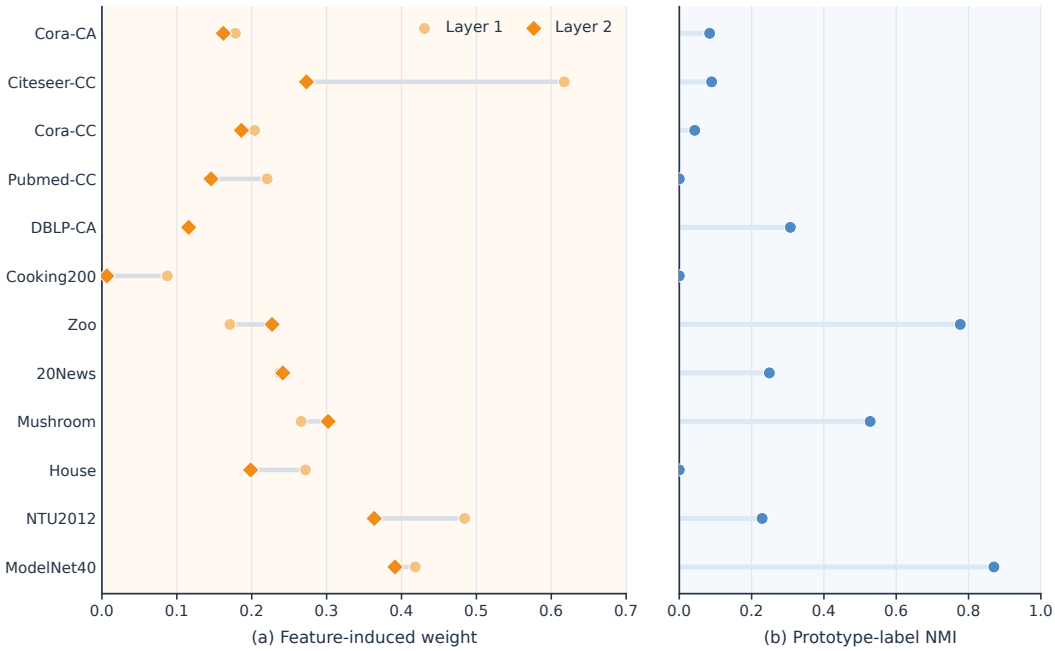



Figure 16: Fusion and assignment diagnostics across 12 datasets. (a) Paired feature-induced coefficients for the first and second layers. Lines expose the direction and magnitude of each layer transition. (b) Normalized mutual information between hard prototype assignments and test labels. Both panels use the same validation-selected seed-42 checkpoint for each dataset.