Collective Node Classification through Neuro-Symbolic Integration

Raffaele Pojer[1], Andrea Passerini[2], Kim G. Larsen[1], and Manfred Jaeger[1]

[1] Aalborg University, Aalborg, Denmark
{rafpoj,kgl,jaeger}@cs.aau.dk
[2] University of Trento, Trento, Italy
andrea.passerini@unitn.it

Abstract. Graph neural networks (GNNs) predict node labels independently conditioned on the node attributes and graph structure. This limits their ability to exploit homophilic or heterophilic label distribution patterns. We use a recently introduced neuro-symbolic framework [13], to embed a standard GNN into a Relational Bayesian Network (RBN), enabling maximum a-posteriori (MAP) inference that jointly optimises node labels to match the observed local homophily structure of a graph. Crucially, this is achieved without any task-specific implementation: the local-homophily constraint is expressed as a symbolic RBN probability formula, and MAP inference is carried out by the toolbox's generic, off-the-shelf inference engine. We evaluate the approach on a new class of synthetic benchmarks based on the Ising model, where homophily level, feature informativeness, and neighbourhood informativeness can be independently controlled, and on nine real-world node classification datasets. Our results show consistent accuracy improvements over the base GNN across nearly all tested settings, outperforming the specialised collective classification method GMNN in most conditions. On real-world benchmarks, this approach improves node classification accuracy on 8 out of 9 datasets for both GGCN and GCN, with the largest gains on heterophilic graphs.

Keywords: graph neural networks · relational Bayesian networks · collective classification · homophily · neuro-symbolic integration

1 Introduction

Node classification is one of the central tasks in graph machine learning, and graph neural networks (GNNs) have become the de facto standard approach for it. Through iterative message passing, GNNs aggregate neighbourhood information to produce node representations that are subsequently mapped to class probabilities. Despite their success, GNNs share a fundamental limitation: predictions for different nodes are made *independently*, given the node features and the graph structure. This means that the predicted label distribution cannot directly account for structured dependencies among labels, most notably, the tendency of neighbouring nodes to share (homophily) or differ in (heterophily)

their labels. Common beliefs about GNNs concerns the effect of homophily on their performance, particularly in node classification. When a node's neighbourhood is homophilic, i.e., when neighbours predominantly share its label, message passing, by updating a node's representation towards an aggregate (typically an average) of its neighbours' representations, reinforces the correct signal and tends to improve classification. To the contrary, if the neighbourhood is heterophilic, then the same aggregation instead blends together representations from different classes and can degrade performance, *provided that* the classes were distinguishable in feature space to begin with. This behaviour of message passing is often referred to as a smoothing operation. The story, however, is more subtle than this simple picture suggests, as recently argued by [1]. The authors states that homophily and heterophily are properties *induced by* the task and by the observed labelling, rather than intrinsic properties of the graph that by themselves determine predictive difficulty. The impact of homophily on GNN performance has been widely discussed in the literature [8,7]. Instead of proposing another countermeasure for the smoothing behaviour itself, we take a different view: treating the level of homophily or heterophily of the observed labelling as a signal in its own right, and show how it can be exploited, on top of a fixed, already-trained GNN, through symbolic MAP inference.

Collective classification methods address the independence limitation by modelling label dependencies across nodes jointly. A representative GNN-based approach is GMNN [14], which combines two GNNs through an EM procedure: one for predicting node labels, and one for modelling their local dependencies. While effective in homophilic settings, GMNN is tailored to a specific structural assumption and does not generalise naturally beyond it. Rather than designing a new, task-specific architecture for node classification, in this paper we show that an existing, general-purpose neuro-symbolic framework [13] already provides everything that is needed: a pre-trained GNN can be lifted into a collective classifier simply by embedding it in a neuro-symbolic GNN–RBN model and performing *maximum a-posteriori (MAP) inference*. The key idea is to embed the GNN inside a Relational Bayesian Network (RBN) that additionally encodes soft constraints on the local homophily of the predicted labelling. MAP inference then simultaneously maximises the GNN's confidence in its predictions and the degree to which the predicted labelling matches the estimated homophily structure of the graph. Crucially, neither the GNN's parameters nor its training procedure need to be modified.

We evaluate the approach on a new family of synthetic benchmarks derived from the Ising model, which provides independent control over all three relevant factors, and on nine standard real-world datasets covering a wide range of homophily levels. Our results consistently show accuracy improvements over the base GNN, and superiority over GMNN in most conditions. Both data and code for the applications are publicly available at `https://github.com/raffaelepojer/NeSy-for-graph-data/tree/main/hetero-hom-experiments`. Primula, the RBN framework used in this paper, is available at `https://github.com/manfred-jaeger-aalborg/primula3`.

2 Background

2.1 Relational Bayesian Networks

Relational Bayesian Networks (RBNs) [4], were introduced for modeling probability distributions over relational structures rather than fixed sets of random variables. The motivation is that many domains involve an unbounded number of entities and probabilistic relations between them, making conventional Bayesian networks designed for tabular data inadequate. RBNs address this limitation by defining probability distributions directly over relations among objects in a domain. An RBN consists of a set of relations together with probabilistic dependency specifications. Similar to Bayesian networks, dependencies are represented by a directed acyclic structure, but the nodes correspond to relations rather than individual random variables. RBNs define generative probability distributions for graphs over a specific relational signature, by specifying dependencies between relations and attributes, and defining conditional probability distributions through expressive *probability formulas*.

The RBN language defines probability formulas through four syntactic constructs, where $F, F_1, \ldots, F_t$ denote probability formulas:

- **Constants.** A real value $q \in [0, 1]$ is a probability formula evaluating to q unconditionally, serving as an unconditional prior (e.g., `SOFTMAX` scores for categorical relations).
- **Atoms.** A relation symbol applied to variables evaluates to 1 if the ground atom is true in the current graph, 0 otherwise.
- **WIF-THEN-ELSE.** The *weighted-if* construct `WIF` F_1 `THEN` F_2 `ELSE` F_3 computes the mixture $F_1 \cdot F_2 + (1 - F_1) \cdot F_3$.
- **COMBINE.** The core expressive construct aggregates sub-formula values over a set of domain tuples selected by a `WHERE` clause:

  `COMBINE` $F_1, \ldots, F_t$ `WITH` ⟨`comb`⟩ `FORALL` ⟨`vars`⟩ `WHERE` ⟨`cond`⟩

  Built-in combination functions include `noisy-or`, `mean`, `sum`, and `log-reg` (logistic regression).

The RBN language with probability formulas, can be as expressive as full first-order logic with equality [4].

To make this concrete, consider a small citation network where nodes represent research papers, each carrying a topic label (three in this case) and a Boolean `highlyCited` attribute indicating whether the paper is highly cited, and where edges represent citation links between papers. An RBN for this domain might specify: a prior distribution over research topics, a citation probability that depends on whether two papers share the same topic, a derived co-citation relation, and finally a model for whether a paper is highly cited as a function of both its in-degree and how often it is co-cited with papers of the same topic. Listing 1.1 shows how these dependencies are expressed as RBN probability formulas.

Listing 1.1. A simple RBN for a citation network. Nodes are papers with a research *topic* and a Boolean *cited* attribute (whether the paper is highly cited). Edges represent citations.

```
% Prior distribution over research topic
topic(v) := SOFTMAX 4.2, 3.1, 2.8 ;

% Citation probability depends on whether
% the two papers share the same topic
cite(v, w) := WIF topic(v) & topic(w)
              THEN 0.25
              ELSE 0.05;

% Both cited by some common paper u
cocited(v, w) := COMBINE
                   cite(u, v) & cite(u, w)
                   WITH noisy-or
                   FORALL u;

% Number of distinct papers that cite v
in_degree(v) := COMBINE 1
                  WITH sum
                  FORALL u
                  WHERE cite(u, v);

% Combination of number of citations and shared
highlyCited(v) :=
    COMBINE
        0.8 * @in_degree(v),
        0.5 * COMBINE topic(v) & topic(w) & cocited(v, w)
              WITH  mean
              FORALL  w
    WITH  log-reg;
```

2.2 Graph Neural Networks

Graph Neural Networks (GNNs) are deep learning models that operate on graph-structured data through an iterative *message passing* paradigm. At each layer, every node aggregates feature vectors from its immediate neighbourhood and combines them with its own representation via a learnable transformation. Formally, the update at layer k is

$$\mathbf{h}^{k+1}(v) = \sigma\left(\sum_{u \in \mathcal{N}_v} W\mathbf{h}^{k}(u)\right), \tag{1}$$

where $\mathbf{h}^k(u) \in \mathbb{R}^d$, $W \in \mathbb{R}^{m \times d}$ is a learnable weight matrix, $\mathcal{N}_v$ denotes the neighbours of v, and σ is a non-linear activation. The final layer representation is passed through a `SOFTMAX` to produce a class-probability vector for node

v. In most architectures, the update also includes a self-loop, i.e., a separate dependence of $\mathbf{h}^{k+1}(v)$ on $\mathbf{h}^k(v)$ itself.

A key property of GNNs, and a limitation in the context of collective classification, is that predictions for different nodes are computed *independently*: the class probability of v depends on the features of v and its k-hop neighbourhood, but not on the predicted labels of other nodes. This means a standard GNN cannot directly incorporate knowledge about the label distribution structure, such as homophily or heterophily.

2.3 From GNN to RBNs and vice-versa

Despite their very different origins, GNNs and RBNs share a fundamental common ground: both ultimately define *conditional probability distributions* over node attributes given the graph structure and observed input features. A GNN for node classification produces, for each node v, a probability vector

$$P_{\mathrm{GNN}_{\mathcal{N}}}(\mathit{target}(v) \mid \boldsymbol{A}, E)$$

over class labels, conditioned on the node attributes $\boldsymbol{A}$ and the edge relation E. An RBN, on the other hand, defines a full *generative* probability model over all relations in its signature, including a conditional distribution $P_{\mathrm{RBN}}(\mathit{target}(v) \mid \boldsymbol{A}, E)$ for each node attribute. The key insight is that these two conditional distributions are semantically identical, and that a GNN function can be expressed exactly as an RBN probability formula, so that the GNN becomes one component of the larger generative model defined by the RBN [5]. This alignment is what makes the integration seamless: once a GNN is embedded into an RBN, its predictions are not approximated or replaced, they are preserved exactly, and the RBN can add further probabilistic structure on top.

Concretely, the connection is established by observing that the update equation (1) can be written coordinate-wise as a weighted sum over individual scalar feature channels [5], which maps directly onto an RBN probability formula.

Using such auxiliary formulas as building blocks, the full coordinate-wise encoding of [5] extends to a complete GNN for node classification. For a GNN $\mathcal{N}$ that predicts a categorical node attribute *target* with d possible values, the final encoding takes the form

$$\mathit{target}(v) := \texttt{SOFTMAX}\ \ F_{K,1}(v), \ldots, F_{K,d}(v), \tag{2}$$

where $F_{K,i}(v)$ are the d components of the final GNN layer before applying the softmax. The resulting RBN formula defines the same conditional distribution over *target* as the GNN itself, making the two representations semantically equivalent.

This compilation, called *GNN-to-RBN encoding*, covers all GNN models in the aggregate-combine-readout (ACR) class, as defined in [2]. It satisfies three important consistency properties:

- **Semantic equivalence.** The conditional distribution $P_{\mathrm{RBN}}(\mathit{target}(v) \mid \boldsymbol{A}, E)$ in the RBN exactly equals the class-probability distribution $P_{\mathrm{GNN}_{\mathcal{N}}}(\mathit{target}(v) \mid \boldsymbol{A}, E)$ produced by the GNN. Both models define the same function from inputs to label probabilities.
- **Training equivalence.** Minimising the cross-entropy loss of the GNN is equivalent to maximising the log-likelihood of the enclosing RBN model. A GNN trained conventionally can therefore be directly embedded into an RBN without any retraining or modification of its weights.
- **Computational equivalence.** Forward and backward passes execute the same sequence of arithmetic operations in both representations.

As an alternative to compiling the GNN into native RBN syntax, an *interface* approach is also supported, in which the RBN contains a declaration of the form

$$\mathit{target(v)} := \langle \text{Link to PyTorch GNN model} \rangle, \tag{3}$$

linking to an external PyTorch implementation. This avoids the overhead of re-implementing GNN architectures in native RBN syntax, and the three consistency properties above also hold for this version. Both integration methods are implemented in the Primula tool [12], which provides the RBN modelling and inference engine used in this work.

In both cases, once the GNN is embedded, its conditional distribution over *target* becomes one factor in the joint generative model defined by the RBN. The RBN can then define additional relations and attributes, capturing domain knowledge or structural constraints, whose distributions may depend on *target*. General probabilistic inference, including MAP inference, can then be performed over the joint model, combining the discriminative power of the GNN with the symbolic expressivity of the RBN.

3 Collective Classification via MAP Inference

3.1 Towards Collective Classification

Beyond the basic relation between homophily and message passing discussed above, three further complications arise in practice that motivate our approach.

First, as pointed out in [8], many widely used heterophilic benchmarks contain node features that are sufficiently discriminative on their own, so that strong classification results can be achieved even without exploiting the graph structure at all. This masks the true difficulty that arises when the label distribution is structured (homophilic or heterophilic) but features are weak, and makes it hard to tell whether a method genuinely exploits graph structure or merely relies on feature information. This calls for controlled synthetic benchmarks in which the three relevant factors, homo/heterophily level, feature informativeness, and neighbourhood utility, can be varied independently.

Second, homophily is typically measured *globally*, as the fraction of edges connecting same-class nodes, yet *local* homophily can vary substantially across

the graph even at fixed global level [7]. A model that only accounts for the global homophily level may therefore still fail on nodes whose local neighbourhood structure is atypical.

Third, and independently of both feature strength and local variation, GNNs predict labels for different nodes *independently* once conditioned on attributes and structure [14]. Even a GNN that has learned a good feature-to-class mapping cannot enforce consistency between neighbouring predictions: two adjacent nodes may each receive a high-confidence label that is jointly inconsistent with the graph's local homophily structure. It is this last limitation that our MAP inference procedure is designed to directly address.

3.2 MAP Inference for Collective Node Classification

We address these limitations by embedding a pre-trained GNN $\mathcal{N}$ into a GNN–RBN model that additionally encodes soft constraints on the *local homophily* of the predicted labelling. For a given labelling $\boldsymbol{y}$ the local homophily at node v is defined as the proportion of edges incident to v that connect nodes of the same label:

$$LH_{\boldsymbol{y}}(v) = \frac{|\{u \in \mathcal{N}_v : \boldsymbol{y}(u) = \boldsymbol{y}(v)\}|}{|\mathcal{N}_v|} \in [0, 1].$$

When a given predicted labelling $\hat{\boldsymbol{y}}$ is an accurate approximation of a true (but partially unknown) labelling $\boldsymbol{y}^*$ ($\hat{\boldsymbol{y}} \approx \boldsymbol{y}^*$) then $|LH_{\boldsymbol{y}^*}(v) - LH_{\hat{\boldsymbol{y}}}(v)| \approx 0$ should also hold for all v. To enforce this matching objective, we introduce a Boolean auxiliary attribute $\overline{LH}(v)$, defined purely at the symbolic level as an RBN probability formula, whose probability is a smooth function of the difference $LH_{\boldsymbol{y}^*}(v) - LH_{\hat{\boldsymbol{y}}}(v)$, shown in Figure 1. These task-specific function definitions are directly supported by the symbolic modeling tools in the RBN-GNN framework.

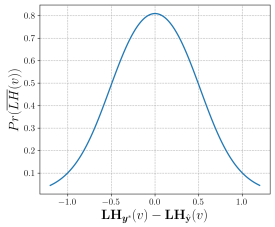


Fig. 1. Probability model for $\overline{LH}$ as a function of the homophily difference.

This function is the product of two logistic regression functions and is constructed so that $P(\overline{LH}(v) = \mathit{true})$ is maximised when the difference is zero and decreases smoothly towards the extreme values ± 1. The full GNN–RBN model defines the joint conditional distribution,

$$P(\hat{\boldsymbol{Y}}, \overline{\boldsymbol{LH}} | \boldsymbol{LH}_{\boldsymbol{y}^*}, \boldsymbol{A}) = P_{\mathcal{N}}(\hat{\boldsymbol{Y}} | \boldsymbol{A})\, P(\overline{\boldsymbol{LH}} | \hat{\boldsymbol{Y}}, \boldsymbol{LH}_{\boldsymbol{y}^*}). \tag{4}$$

After training $\mathcal{N}$ conventionally with cross-entropy loss, we condition on $\overline{\boldsymbol{LH}} = \boldsymbol{true}$ and perform MAP inference for $\hat{\boldsymbol{Y}}$:

$$\hat{\boldsymbol{y}} = \underset{\boldsymbol{y}\in\mathcal{Y}}{arg\,max}\ P_{\mathcal{N}}(\hat{\boldsymbol{Y}} = \boldsymbol{y}|\boldsymbol{A})\, P(\overline{\boldsymbol{LH}} = \boldsymbol{true}|\boldsymbol{y}, \hat{\boldsymbol{LH}}). \tag{5}$$

Since $\boldsymbol{LH}_{\boldsymbol{y}^*}$ cannot be computed exactly (for the unlabelled nodes) we substitute the estimate $\hat{\boldsymbol{LH}}$, obtained by propagating the local homophily values of the labelled nodes across the graph via a procedure inspired by label propagation (Algorithm 1 in Appendix A). As in label propagation, this estimation step relies on an assumption of its own: that local homophily values themselves vary smoothly between adjacent nodes, so that the (known) local homophily at labelled nodes is informative about the (unknown) local homophily at their unlabelled neighbours. Although the variables $\hat{Y}(v)$ are independent under $P_{\mathcal{N}}$, conditioning on $\overline{\boldsymbol{LH}} = \boldsymbol{true}$ introduces dependencies between them, turning the problem into a genuine *collective* classification. The MAP objective simultaneously maximises the GNN's confidence and penalises labellings that deviate from the estimated local homophily structure.

Training nodes whose neighbourhood contains at least one other labelled node are seeded with their locally computed homophily; all remaining nodes are initialised to μ, the mean homophily over seeded training nodes. The process iterates until convergence, treating nodes differently according to their neighbourhood composition, and returns estimated homophily values for all nodes.

4 Ising Benchmark Data

In many existing homo/heterophilic benchmark datasets, node features alone often suffice for classification, which obscures the structural role of label distributions [8]. This makes it difficult to assess whether a method genuinely exploits the graph structure or merely relies on feature information. To move beyond this limitation, we introduce graphs derived from the *Ising model* as a controllable synthetic testbed. The Ising model has also been used to construct GNN benchmarks targeting a different question, long-range dependence rather than homophily, in the LRIM benchmark [9].

Unlike most current datasets, where feature-level and structural heterophily are entangled, Ising graphs allow us to isolate each contributing factor as a clean experimental variable. In particular, we can independently control:

- the **homo/heterophily level** of the label distribution;
- the **feature informativeness**, i.e., how strongly node features predict the label;
- the **neighbourhood informativeness**, i.e., whether aggregating neighbour features provides additional predictive signal beyond the node's own feature.

4.1 Data Generation

We generate random node labellings according to the Ising model from statistical physics. The graph structure consists of a fixed regular $n \times n$ grid, where each

node is connected to its up to four immediate neighbours (up, down, left, right). Nodes have binary class labels $+1$ or -1 (representing positive and negative electromagnetic spin). The probability of a labelling $\boldsymbol{y} \in \{-1, +1\}^{n \times n}$ is

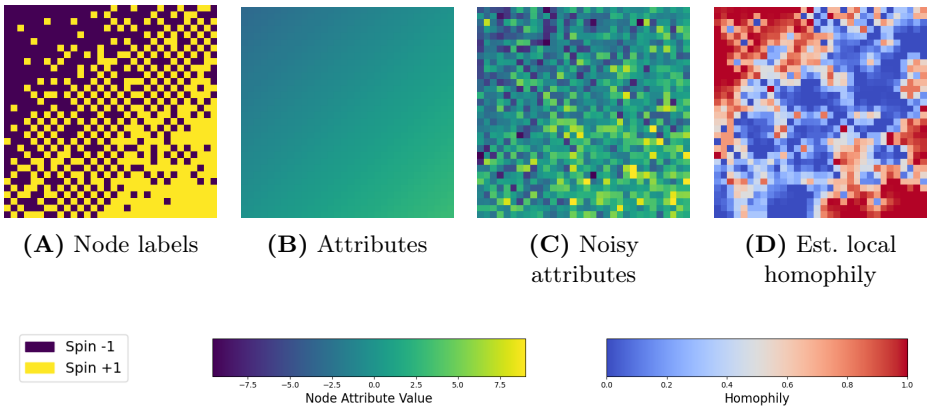
$$P(\boldsymbol{y}) = \frac{1}{Z} \exp\big(\phi(\boldsymbol{y})\big),$$

with

$$\phi(\boldsymbol{y}) = \sum_{v} \boldsymbol{y}(v) \left(F \cdot f(v) + H \cdot \sum_{u \in \mathcal{N}_v} \boldsymbol{y}(u) \right), \tag{6}$$

where $f(v) \in \mathbb{R}$ is a scalar node feature (a linear gradient over the grid, representing the external magnetic field at v), and F, H are real parameters. The parameter H controls the bias towards homophilic ($H > 0$) or heterophilic ($H < 0$) labellings; the parameter F controls the influence of the node feature on the label. With $F = 0$, label probabilities depend only on the number of neighbours with the same label, and node features are entirely uninformative. For each sampled labelling $\boldsymbol{y}$, two versions of the data are constructed. In the **noise-free** version, each node v receives the unperturbed feature value $A(v) = F \cdot f(v)$. When the underlying feature signal varies smoothly across the grid, a node's own attribute value already closely approximates the values held by its neighbours, so averaging over the neighbourhood adds essentially no information beyond what the node's own feature already encodes. Injecting independent noise into each node's observed attribute breaks this correspondence: since the noise at different nodes is uncorrelated while the underlying signal is not, averaging over the neighbourhood cancels out much of the noise and yields a better estimate of the true signal than any single noisy observation alone.

We generate data on 32×32 grids with a fixed 48/32/20% train/validation/test split.



(A) Node labels **(B)** Attributes **(C)** Noisy attributes **(D)** Est. local homophily

Fig. 2. Ising data for $H = -0.4$, $F = 0.1$.

To make the data generation concrete, Figure 2 shows one specific configuration, $H = -0.4$, $F = 0.1$, which represents a moderately heterophilic graph ($h = 0.39$) with non-zero feature signal. The remaining four configurations in the benchmark vary H and F to cover strongly homophilic graphs (e.g. $H = 0.9$, $h = 0.97$), strongly heterophilic graphs with no feature signal ($H = -0.5$, $F = 0$, $h = 0.07$), and an intermediate heterophilic case with a stronger feature ($H = -0.7$, $F = 0.3$).

4.2 Ising: Results

Table 1 reports classification accuracies on the five configurations. We compare the state-of-the-art GGCN [15], which is specifically designed for both homophilic and heterophilic data, a standard GCN [6], and a baseline MLP (node features only, no aggregation). All models use 2 message-passing layers and a hidden dimension of 16 (32 for MLP). The number of layers was selected via validation performance and did not require further tuning across configurations. For each model we report accuracy with and without the MAP inference step. MAP inference uses 3 random restarts; results are means and standard deviations over 5 runs. For comparison we include GMNN [14] on the same split. When nodes have no informative attributes ($F = 0$), a GNN alone cannot do much better than predicting the majority class; adding MAP inference enables a very significant jump in performance. In the $F > 0$, noise-free settings the MLP performs almost as well as the proper GNN models, yet MAP inference still yields a marked improvement in all cases. The configuration $H = -0.4$, $F = 0.1$, which has the most fragmented distribution of local homophily values, poses the greatest challenge. With noisy attributes, performance drops across all models. The base GNNs still benefit from neighbourhood aggregation over the MLP, whose noisy-feature accuracy collapses towards majority-class level; after MAP inference, however, this gap narrows sharply, with MLP+MAP matching or exceeding both GNN+MAP variants in two of the three noisy configurations. This suggests that once the estimated local-homophily signal is available, it can compensate for a weak base classifier almost as effectively as for a strong one.

5 Real-World Data

We evaluate on the nine benchmark datasets used by GGCN [15], as summarised in Table 2. Datasets are loaded via PyTorch Geometric [3] using the Geom-GCN splits [10], except for Chameleon and Squirrel, where we use the filtered splits of [11]: the original versions contain many duplicate nodes with identical labels and neighbourhoods, which distorts local homophily estimates and lets models inflate accuracy by memorising duplicated test nodes. We retrain GCN [6] and GGCN [15] from scratch using their authors' original settings, and evaluate each with and without our MAP inference step (3 restarts), using the estimated local homophily. As in the Ising experiments, we also compare against GMNN [14] on the same splits.

Table 1. Node classification accuracy (%) on Ising benchmarks (mean ± std over 5 runs). Gray cells show the better result within each base/+MAP pair; bold indicates the best per column within each noise condition.

	Model	(H, F) $(0.5, 0.0)$	$(-0.5, 0.0)$	$(-0.4, 0.1)$	$(-0.7, 0.3)$	$(0.9, 0.05)$
No noise	GGCN	52.20±0.00	50.63±0.95	61.17±0.66	76.68±1.17	94.63±0.00
	GGCN+MAP	**91.01±0.49**	95.56±0.47	79.52±0.39	92.27±0.53	96.23±0.71
	GCN	52.20±0.00	48.78±0.00	61.56±0.37	71.90±0.39	94.63±0.00
	GCN+MAP	90.82±0.61	96.04±0.47	80.39±0.39	**93.33±0.64**	95.65±0.00
	MLP	52.20±0.00	51.22±0.00	62.05±0.57	77.07±0.62	94.63±0.00
	MLP+MAP	90.43±0.99	**96.33±0.72**	**80.48±0.95**	92.75±0.43	**96.91±0.39**
	GMNN	84.00±2.37	57.22±2.67	60.59±0.89	76.93±1.46	92.78±2.61
With noise	GGCN	–	–	60.39±4.10	62.15±7.15	76.10±0.44
	GGCN+MAP	–	–	78.26±1.10	**90.24±2.76**	94.01±0.24
	GCN	–	–	62.34±0.57	71.41±0.79	76.59±0.69
	GCN+MAP	–	–	79.13±0.19	89.08±0.58	93.53±0.58
	MLP	–	–	54.34±0.90	52.68±0.82	59.32±0.90
	MLP+MAP	–	–	**80.00±0.66**	87.15±0.39	**97.87±0.39**
	GMNN	–	–	60.93±1.56	75.66±2.04	90.44±1.24

5.1 Results

Table 2 shows results ordered by increasing homophily. MAP inference improves over the base GNN on 8 of 9 datasets for both GGCN and GCN, with Pubmed the sole exception for both. Gains are largest on the heterophilic datasets, especially for the weaker GCN baseline, and smaller but still positive on the strongly homophilic citation networks. Compared to GMNN, which is competitive mainly on homophilic data as expected from its design, MAP inference generally matches or outperforms it without any specialised architecture or joint training. This mirrors the Ising findings: MAP inference helps most when local label structure is informative but not exploited by independent GNN predictions.

6 Scalability and Inference Overhead

To assess the computational overhead introduced by the MAP inference step, we measure total inference time on Ising grids of increasing size. Table 3 reports the results. Inference time, for the Ising experiment, scales roughly linearly with the number of nodes, increasing by a factor of approximately 4× at each step as graph size quadruples. This is expected since each node contributes a MAP atom whose value has to be scored involving the neighbor MAP atoms, so the size of the search space, grows directly with the number of edges in the graph. For the Ising used here, since it is a grid, each node has a fixed number of neighbours (up to four), the number of edges itself grows linearly with the number of nodes, which is why the observed scaling in Table 3 is linear in the number of nodes as well.

Table 2. Node classification accuracy (%) on real-world benchmarks, ordered by increasing homophily (h), reported as mean ± std. Gray cells highlight the better result within each base/+MAP pair; **bold** indicates the best result overall across all three models.

		Texas	Wisconsin	Squirrel	Actor	Chameleon	Cornell	Citeseer	Pubmed	Cora
Homophily (h)		.13	.20	.21	.22	.24	.31	.74	.80	.81
# Nodes		183	251	2,223	7,600	890	183	3,327	19,717	2,708
# Edges		325	515	93,996	26,752	17,708	298	4,676	44,327	5,278
# Classes		5	5	5	5	5	5	6	3	7
GGCN	Base	82.97±4.99	86.67±3.70	31.71±2.19	34.97±0.94	33.96±2.74	75.95±4.09	76.87±1.46	**89.13**±0.33	86.02±1.53
	+MAP	**85.95**±5.64	**87.65**±4.39	36.99±2.57	**43.32**±3.78	42.17±7.44	**80.27**±3.21	**81.80**±2.62	80.83±3.71	**89.40**±1.69
GCN	Base	57.03±4.90	48.24±6.86	35.35±1.39	28.14±0.65	39.55±2.94	40.27±4.90	74.85±1.79	86.75±0.50	86.52±1.05
	+MAP	62.43±8.50	68.24±11.05	**59.28**±8.06	42.23±4.60	**45.52**±3.67	55.41±9.15	79.60±3.22	82.19±3.07	89.22±1.81
GMNN	Base	58.11±4.23	58.43±9.44	34.74±1.30	28.74±1.51	36.80±2.91	44.32±5.82	74.65±1.38	84.91±0.51	81.37±1.60

Grid ($n \times n$)	**# nodes**	**Total time**
16	256	0.482 s
32	1,024	1.485 s
64	4,096	5.558 s
128	16,384	26.465 s

Table 3. Total inference time (seconds) by graph size.

7 Discussion and Conclusion

We have presented an approach to collective node classification based on embedding a standard GNN into a Relational Bayesian Network, and performing MAP inference to jointly assign labels that are consistent with both the GNN's predictions and the estimated local homophily structure of the graph. The approach is deliberately generic: it reuses an existing neuro-symbolic toolbox [13] without any task-specific inference code, any pre-trained GNN can be used as the base model without modification, and the MAP inference step is applied as a post-processing step after standard training. The experimental results demonstrate consistent improvements over the base GNN across a wide range of homophily conditions, on both controlled synthetic benchmarks and standard real-world datasets. Compared to GMNN, our approach is more broadly applicable: it does not require retraining or a specialised architecture, and it performs well under heterophily as well as homophily. A current limitation is the dependence on the homophily estimation step, which may be less accurate on graphs where neighbouring nodes have very different local homophily values. Future work will investigate more robust homophily estimation methods and the scalability of the MAP inference algorithm to larger graphs.

Acknowledgments. RP was supported by funding from the Villum Investigator Grant S4OS (37819) from Villum Foundation. AP was partly supported by the European Union (Grant Agreement no. 101120763 – TANGO).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Arnaiz-Rodriguez, A., Errica, F.: Oversmoothing, oversquashing, heterophily, long-range, and more: Demystifying common beliefs in graph machine learning (2026). https://doi.org/10.48550/arXiv.2505.15547, https://arxiv.org/abs/2505.15547
2. Barceló, P., Kostylev, E.V., Monet, M., Pérez, J., Reutter, J., Silva, J.P.: The logical expressiveness of graph neural networks. In: 8th International Conference on Learning Representations (ICLR 2020) (2020)
3. Fey, M., Lenssen, J.E.: Fast graph representation learning with pytorch geometric (2019). https://doi.org/10.48550/arXiv.1903.02428, https://arxiv.org/abs/1903.02428
4. Jaeger, M.: Relational Bayesian networks. In: Proceedings of UAI (1997)
5. Jaeger, M.: Learning and Reasoning with Graph Data: Neural and Statistical-Relational Approaches. In: International Research School in Artificial Intelligence in Bergen. Open Access Series in Informatics (OASIcs) (2022). https://doi.org/10.4230/OASIcs.AIB.2022.5
6. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks (2017). https://doi.org/10.48550/arXiv.1609.02907, https://arxiv.org/abs/1609.02907
7. Loveland, D., Zhu, J., Heimann, M., Fish, B., Schaub, M.T., Koutra, D.: On performance discrepancies across local homophily levels in graph neural networks. In: Learning on Graphs Conference (2024)
8. Luan, S., Hua, C., Lu, Q., Ma, L., Wu, L., Wang, X., Xu, M., Chang, X.W., Precup, D., Ying, R., et al.: The heterophilic graph learning handbook: Benchmarks, models, theoretical analysis, applications and challenges. arXiv preprint arXiv:2407.09618 (2024). https://doi.org/10.48550/arXiv.2407.09618
9. Mathys, J., Christiansen, H., Errica, F., Maruyama, T., Alesiani, F.: LRIM: a physics-based benchmark for provably evaluating long-range capabilities in graph learning. In: The Fourteenth International Conference on Learning Representations (2026), https://openreview.net/forum?id=IAZXEX1dVV
10. Pei, H., Wei, B., Chang, K.C.C., Lei, Y., Yang, B.: Geom-gcn: Geometric graph convolutional networks (2020). https://doi.org/10.48550/arXiv.2002.05287, https://arxiv.org/abs/2002.05287
11. Platonov, O., Kuznedelev, D., Diskin, M., Babenko, A., Prokhorenkova, L.: A critical look at the evaluation of gnns under heterophily: Are we really making progress? (2024). https://doi.org/10.48550/arXiv.2302.11640, https://arxiv.org/abs/2302.11640
12. Pojer, R., Jaeger, M.: Primula-3 for probabilistic modeling and reasoning on graph data. In: ECML PKDD Conference, Demo track (2025). https://doi.org/10.1007/978-3-032-06129-4_35

13. Pojer, R., Passerini, A., Jaeger, M.: Generalized reasoning with graph neural networks by relational bayesian network encodings. In: Learning on Graphs Conference (2024)
14. Qu, M., Bengio, Y., Tang, J.: Gmnn: Graph Markov neural networks. In: ICML (2019)
15. Yan, Y., Hashemi, M., Swersky, K., Yang, Y., Koutra, D.: Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks. In: ICDM (2022). https://doi.org/10.1109/ICDM54844.2022.00169

A Homophily Propagation Algorithm

Algorithm 1 estimates the unknown true local homophily values $\boldsymbol{LH}_{\boldsymbol{y}^*}$ via a label-propagation-style procedure on an undirected copy of the input graph.

Training nodes with at least one labelled train neighbour are seeded with their locally computed homophily; all remaining nodes are initialised to μ, the mean over seeded training nodes. At each iteration, estimates are updated by weighted averages over neighbours, with weights $W_{\text{train}} = 2$ and $W_{\text{test}} = 1$ giving greater influence to observed training values. Training nodes with only labelled neighbours are never updated. The process repeats until no estimate changes by more than ϵ, or a maximum number of iterations is reached.

B Ising Dataset Configurations

Figure 3 shows the remaining four Ising benchmark configurations in a single panel, using the same visualization format as Figure 2: (A) node labels, (B) unperturbed attribute, (C) noisy attribute, and (D) estimated local homophily. The shared legend at the bottom applies to all rows.

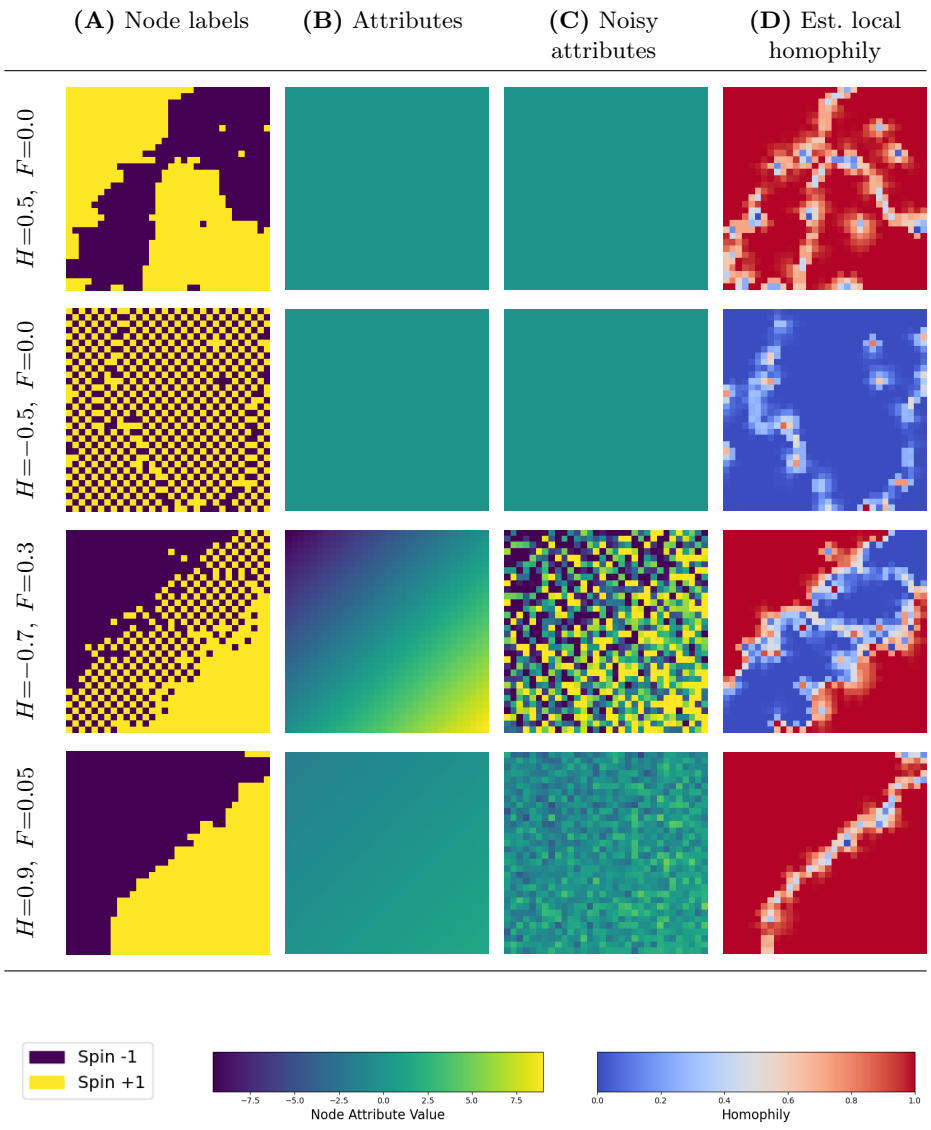


Fig. 3. Remaining four Ising benchmark configurations. Each row corresponds to one (H, F) setting; columns (A)–(D) follow the same format as Figure 2 in the main text.

Algorithm 1 Propagate Homophily

procedure PROPAGATEHOMOPHILY(graph(V, E), iterations, tolerance)
  Convert **graph** to an undirected graph
  $W_{train} \leftarrow 2.0, \quad W_{test} \leftarrow 1.0$
  Initialize: `stabilized[`v`]` $\leftarrow$ `false`, `train_hom[`v`]` $\leftarrow 0$ for all $v \in V$
  $sumTrainHom \leftarrow 0$
  $countValidTrain \leftarrow 0$
  for each training node $v \in V$ with non-empty $trainNbrs$ **do**
    `train_hom[`v`]` $\leftarrow$ COMPUTELOCALHOMOPHILY($trainNbrs$)
    `hom[`v`]` $\leftarrow$ `train_hom[`v`]`
  end for
  $initValue \leftarrow$ AVERAGE($\{$`train_hom[`v`]` $\mid v \in V_{\text{train}}$ and $trainNbrs \neq \emptyset\}$)
  for each test node v, or training node v with empty $trainNbrs$ **do**
    `hom[`v`]` $\leftarrow initValue$
  end for
  $converged \leftarrow$ `false`
  for $iter \leftarrow 1$ **to** iterations **while not** converged **do**
    $converged \leftarrow$ `true`
    for each node $v \in V$ **do**
      $nbrs \leftarrow$ GETALLNEIGHBORS(v, graph)
      $trainNbrs \leftarrow$ GETTRAINNEIGHBORS(v, graph)
      $testNbrs \leftarrow$ GETTESTNEIGHBORS(v, graph)
      if v is a test node **or** (v is training and $trainNbrs$ is empty) **then**
        if $nbrs$ is empty **then**
          `hom[`v`]` $\leftarrow initValue$
        else
          `hom[`v`]` $\leftarrow$ WEIGHTEDAVERAGE($trainNbrs, testNbrs, W_{train}, W_{test}$)
        end if
      else if v is a training node and $testNbrs$ is not empty **then**
        $avgTest \leftarrow$ AVERAGE(`[hom[`u`] for each` $u \in testNbrs$`]`)
        `hom[`v`]` $\leftarrow$ WEIGHTEDAVERAGE(`train_hom[`v`]`$, avgTest, W_{train}, W_{test}$)
      end if
      if change in `hom[`v`]` $>$ tolerance **then** $converged \leftarrow$ `false`
    end for
  end for
  return `hom`
end procedure