From Hypergraph Energy Functions to Hypergraph Neural Networks

Yuxin Wang [1 2 †] **Quan Gan** [3] **Xipeng Qiu** [1 4] **Xuanjing Huang** [1 5] **David Wipf** [3]

Abstract

Hypergraphs are a powerful abstraction for representing higher-order interactions between entities of interest. To exploit these relationships in making downstream predictions, a variety of hypergraph neural network architectures have recently been proposed, in large part building upon precursors from the more traditional graph neural network (GNN) literature. Somewhat differently, in this paper we begin by presenting an expressive family of parameterized, hypergraph-regularized energy functions. We then demonstrate how minimizers of these energies effectively serve as node embeddings that, when paired with a parameterized classifier, can be trained end-to-end via a supervised bilevel optimization process. Later, we draw parallels between the implicit architecture of the predictive models emerging from the proposed bilevel hypergraph optimization, and existing GNN architectures in common use. Empirically, we demonstrate state-of-the-art results on various hypergraph node classification benchmarks. Code is available at https://github.com/yxzwang/PhenomNN.

1. Introduction

Hypergraphs represent a natural extension of graphs, whereby each hyperedge can link an arbitrary number of hypernodes (or nodes for short). This flexibility more directly facilitates the modeling of higher-order relationships between entities (Chien et al., 2022; Benson et al., 2016; 2017) leading to strong performance in diverse real-world situations (Agarwal et al., 2005; Li & Milenkovic, 2017; Feng et al., 2019; Huang & Yang, 2021). Currently, hypergraph-graph-based modeling techniques frequently rely, either implicitly or explicitly, on some type of expansion (e.g., clique, star), which effectively converts the hypergraph into a regular graph with a new edge set and possibly additional nodes as well. For example, one approach is to first extract a particular expansion graph and then build a graph neural network (GNN) model on top of it (Zhang et al., 2022).

We instead adopt a different starting point that both allows us to incorporate multiple expansions if needed, but also transparently explore the integrated role of each expansion within a unified framework. To accomplish this, our high-level strategy is to first define a family of parameterized hypergraph energy functions, with regularization factors that we later show closely align with popular existing expansions. We then demonstrate how the minimizers of such energy functions can be treated as learnable node embeddings and trained end-to-end via a bilevel optimization process. Namely, the lower-level minimization process produces optimal features contigent on a given set of parameters, while the higher-level process trains these parameters (and hence the features they influence) w.r.t. downstream node classification tasks.

To actualize this goal, after presenting related work in Section 2, we provide relevant background and notation w.r.t. hypergraphs in Section 3. The remainder of the paper then presents our primary contributions, which can be summarized as follows:

- We present a general class of hypergraph-regularized energy functions in Section 4 and elucidate their relationship with traditional hypergraph expansions that have been previously derived from spectral graph theory.
- We demonstrate how minimizers of these energy functions can serve as principled, trainable features for hypergraph prediction tasks in Sections 5 and 6. And by approximating the energy minimizers using provably-convergence proximal gradient steps, the resulting architecture borrows the same basic structure as certain graph neural network layers that: (i) have been fine-tuned to accommodate hypergraphs, and (ii) maintain

[†]Work completed during an internship at the AWS Shanghai AI Lab. [1]School of Computer Science, Fudan University [2]Institute of Modern Languages and Linguistics, Fudan University [3]Amazon [4]Peng Cheng Laboratory [5]Shanghai Collaborative Innovation Center of Intelligent Visual Computing. Correspondence to: Yuxin Wang <wangyuxin21@m.fudan.edu.cn>, Quan Gan <quagan@amazon.com>, Xipeng Qiu <xpqiu@fudan.edu.cn>, Xuanjing Huang <xjhuang@fudan.edu.cn>, David Wipf <davidwipf@gmail.com>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

the inductive bias infused by the original energy function.

- The resulting framework, which we name **PhenomNN** for *Purposeful Hyper-Edges iN Optimization Motivated Neural Networks*, is applied to a multitude of hypergraph node classification benchmarks in Section 7, achieving competitive or SOTA performance in each case.

2. Related Work

Hypergraph Expansions/Neural Networks. Hypergraphs are frequently transformed into graphs by expansion methods including the clique and star expansions. An extensive spectral analysis study of of different hypergraph expansions is provided in (Agarwal et al., 2006), but not from the vantage point of energy functions as is our focus. An alternative line expansion (Yang et al., 2020) has also been proposed that can be viewed in some sense as a hybrid combination of clique and star expansions, although this involves the creation of additional nodes, and there may be scalability issues. In terms of predictive models, previous spectral-based hypergraph neural networks are analogous to applying GNNs on clique expansions, including HGNN (Feng et al., 2019), HCHA (Bai et al., 2021), H-GNNs (Zhang et al., 2022). Meanwhile, FastHyperGCN (Yadati et al., 2019) and HyperGCN (Yadati et al., 2019) reduce a hyperedge into a subgraph using Laplacian operators (Chan & Liang, 2020), which can be viewed as a modified form of clique expansion. HGAT (Ding et al., 2020), HNHN (Dong et al., 2020), HyperSAGE (Arya et al., 2020), UniGNN (Huang & Yang, 2021), (Srinivasan et al., 2021), Set-based models (Chien et al., 2022), (Heydari & Livi, 2022), (Aponte et al., 2022), HEAT (Georgiev et al., 2022) take into account hyperedge features and use a message-passing framework, which can be interpreted as GNNs applied to the star expansion graph. And finally, (Wang et al., 2023) use gradient diffusion processes to motivate a broad class of hypergraph neural networks, although in the end there is not actually any specific energy function that is being minimized by the proposed model layers.

Graph Neural Networks from Unfolded Optimization. A variety of recent work has demonstrated that robust GNN architectures can be formed via graph propagation layers that mirror the unfolded descent iterations of a graph-regularized energy function (Chen & Eldar, 2021; Liu et al., 2021; Ma et al., 2020; Pan et al., 2021; Yang et al., 2021; Zhang et al., 2020; Zhu et al., 2021; Ahn et al., 2022). In doing so, the node embeddings at each layer can be viewed as increasingly refined approximations of an interpretable energy minimizer, that may be designed, for example, to mitigate GNN oversmoothing or perhaps inject robustness to spurious edges. Furthermore, these learnable embeddings can be integrated within a bilevel optimization framework (Wang et al., 2016) for supervised training. While at a high level we adopt a similar conceptual starting point, we nonetheless introduce nontrivial adaptations that are particular to the hypergraph domain, where this framework has not yet been extensively explored, and provide hypergraph-specific insights along the way.

3. Hypergraph Background and Notation

A hypergraph can be viewed as a higher-order form of graph whereby edges can encompass more than two nodes. Specifically, let $\mathcal{G}(\mathcal{V}, \mathcal{E})$ denote a hypergraph, where $\mathcal{V}$ is a set of $n = |\mathcal{V}|$ vertices and $\mathcal{E}$ is a set of $m = |\mathcal{E}|$ hyperedges. In contrast to a traditional graph, each hyperedge $e_k \in \mathcal{E}$, can link an arbitrary number of nodes. The corresponding hypergraph connectivity structure is conveniently represented in a binary incidence matrix $B \in \mathbb{R}^{n\times m}$, where $B_{ik} = 1$ if node $v_i \in e_k$, otherwise $B_{ik} = 0$. We also use $D_H \in \mathbb{R}^{m\times m}$ to denote the degree matrix of the hypergraph, where $m_{e_k} \triangleq D_H[k,k] = \sum_i B_{ik}$.

And finally, we define input features and embeddings for both nodes and hyperedges. In this regard, $X \in \mathbb{R}^{n\times d_x}$ represents a matrix of d_x-dimensional initial/given node features, while $Y \in \mathbb{R}^{n\times d_y}$ refers to the corresponding node embeddings of size d_y we seek to learn. Analogously, $U \in \mathbb{R}^{n\times d_u}$ and $Z \in \mathbb{R}^{m\times d_z}$ are the initial edge features and learnable embeddings respectively. While here we have presented the most general form, we henceforth just assume $d = d_x = d_y = d_z = d_u$ for simplicity.

4. A Family of Hypergraph Energy Functions

Our goal is to pursue hypergraph-based energy functions whose minima produce embeddings that will ultimately be useful for downstream predictive tasks. In this section, we first present an initial design of these functions followed by adaptations for handling the situation where no edge features U are available. We then show how in certain circumstances the proposed energy functions reduce to special cases that align with hypergraph star and clique expansions, before concluding with revised, simplified energy expressions informed by these considerations.

4.1. Initial Energy Function Design and Motivation

We begin with the general form

$$\ell(Y, Z; \psi) = g_1(Y, X; \psi) + g_2(Z, U; \psi) + g_3(Y, Z, \mathcal{G}; \psi) \tag{1}$$

where $g_1(Y, X; \psi)$ and $g_2(Z, U; \psi)$ are non-structural regularization factors over node and edge representations respectively, while $g_3(Y, Z, \mathcal{G}; \psi)$ explicitly incorporates hypergraph structure. In all cases ψ represents parameters that

control the shape of the energy, with particular choices that should be clear from the context (note that these parameters need not all be shared across terms; however, we nonetheless lump them together for notational convenience).

For the non-structural terms in (1), a plausible design criteria is to adopt functions that favor embeddings (either node or edge) that are similar to the corresponding input features or some transformation thereof. Hence we select

$$\begin{aligned} g_1(Y, X; \psi) &= \sum_{i=1}^{n} \|y_i - f(x_i; W_x)\|_2^2 \\ g_2(Z, U; \psi) &= \sum_{k=1}^{m} \|z_k - f(u_k; W_u)\|_2^2, \end{aligned} \tag{2}$$

noting that both cases favor embeddings with minimal ℓ_2 distance from the trainable base predictor, and by extension, the initial features $\{X, U\}$. In practice, the function f can be implemented as an MLP with node/edge weights W_x and W_u respectively.

Turning to $g_3(Y, Z, \mathcal{G}; \psi)$, our design is guided by the notion that:

(i) Both node and edge embeddings should be individually constrained to a shared subset of $\mathbb{R}^d$, e.g., consistent with most GNN architectures we may enforce non-negative embeddings;

(ii) Nodes sharing an edge should be similar when projected into an appropriate space, and;

(iii) Nodes within an edge set should have similar embeddings to the edge embedding, again, when suitably projected.

With these desiderata in mind, we adopt

$$g_3(Y, Z, \mathcal{G}; \psi) = \sum_{i=1}^{n} \phi(y_i) + \sum_{k=1}^{m} \phi(z_i) + \lambda_0 \overbrace{\sum_{e_k \in \mathcal{E}} \sum_{i \in e_k} \sum_{j \in e_k} ||y_i H_0 - y_j||_2^2}^{(a)} + \lambda_1 \overbrace{\sum_{e_k \in \mathcal{E}} \sum_{i \in e_k} ||y_i H_1 - z_k||_2^2}^{(b)} \tag{3}$$

For the first terms we choose $\phi : \mathbb{R}^d \to \mathbb{R}_+^d$ defined as $\phi(p) \triangleq \sum_{j=1}^{d} \mathcal{I}_\infty[p_i < 0]$, where $\mathcal{I}_\infty$ is an indicator function that assigns an infinite penalty to any $p_i < 0$. This ensures that all node and edge embedding must be non-negative to achieve finite energy. Next, the term labeled (a) in (3) directly addresses criteria (ii). We note that the summation is over both indices i and j so that the symmetric counterpart, where the roles of nodes v_i and v_j are switched, is effectively included in the summation. And finally, criteria (iii) is handled by the last term, labeled (b). Here the node and edge embeddings play different roles and exhibit a natural asymmetry.[1] Incidentally, the projections H_0 and H_1 can be viewed as compatibility matrices, initially introduced for label or belief propagation (Eswaran et al., 2017; Yamaguchi et al., 2016; Zhou et al., 2003) to provide additional flexibility to the metric in which entities are compared; for term (a) H_0 facilitates the handling of nodes with potentially heterophily relationships, while for term (b) H_1 accommodates the comparison of fundamentally different embedding types.

4.2. Handling a Lack of Edge Features

In some practical situations there may not be any initial hyperedge features U. In such cases we could potentially modify $\ell(Y, Z; \psi)$ accordingly in multiple different ways. First, and perhaps simplest, we can simply remove $g_2(Z, U; \psi)$ from (1). We will explore the consequences of this option further in Section 4.3. But for tasks more related to hyperedge classification, it may be desirable to maintain this term for additional flexibility. Hence as a second option, we could instead create pseudo features $\widetilde{U}$ with $\widetilde{u}_k = \text{AGG}\left[\{x_i | i \in e_k\}\right]$ for all $e_k \in \mathcal{E}$ for some aggregation function AGG. Or in a similar spirit, we could adopt $f(u_k; W_u) \equiv \text{AGG}\left[\{f(x_i; W_x) | i \in e_k\}\right]$ such that aggregation now takes place after the initial feature transformations.

4.3. Analysis of Simplified Special Cases

Because most hypergraph benchmarks for node classification, and many real-world use cases, involve data devoid of hyperedge features, in this section we more closely examine simplifications of (1) that arise when $g_2(Z, U; \psi)$ is removed. For analysis purposes, it is useful to first introduce two representative hypergraph expansions, both of which can be viewed as converting the original hypergraph to a regular graph, which is tantamount to the assumption that edges in these expanded graphs involve only pairs of nodes.

Clique Expansion. For the *clique expansion* (Zien et al., 1999), we form the regular graph $\mathcal{G}_C(\mathcal{V}, \mathcal{E}_C)$, where the node set $\mathcal{V}$ remains unchanged while the edge set $\mathcal{E}_C$ is such that, for all $e_k \in \mathcal{E}$, we have that $\{v_i | i \in e_k\}$ forms a complete subgraph of $\mathcal{G}_C$. We define L_C, A_C, and D_C as the corresponding Laplacian, adjacency matrix, and degree matrix of $\mathcal{G}_C$ respectively.

Star Expansion. In contrast, the *star expansion* (Zien et al., 1999) involves creating the bipartite graph $\mathcal{G}_S(\mathcal{V}_S, \mathcal{E}_S)$, with revised node set $\mathcal{V}_S = \{v_1, \ldots, v_{n+m}\}$ and edge set $\mathcal{E}_S$ defined such that $\{v_i, v_{n+k}\} \in \mathcal{E}_S$ iff

[1] While we could consider adding an additional factor $||y_i - z_k H_2||_2^2$ to this term, we found that in practice it was not necessary.

$B_{ik} = 1$. Conceptually, the resulting graph is formed with a new node associated with each hyperedge (from the original hypergraph), and an edge connecting every such new node to the original nodes within the corresponding hyperedges. Additionally, $L_S = D_S - A_S$ is the revised Laplacian matrix, with D_S and A_S the degree and adjacent matrices of the star expansion graph.

Unification. We now introduce simplifying assumptions to link the proposed energy with the Laplacians of clique and star expansions as follows:

Proposition 4.1. *Suppose $g_2(Z, U; \psi)$ is removed from (1), $H_0 = H_1 = I$, and define $Z^* \triangleq D_H^{-T} B^T Y$. It then follows that*

$$\begin{aligned}\min_Z \ \ell(Y, Z; \psi) \ &= \ g_1(Y, X; \psi) + \sum_{i=1}^{n} \phi(y_i) \qquad (4)\\ &+ \ 2\lambda_0 tr[Y^T L_C Y] + \lambda_1 tr\left(\begin{bmatrix} Y \\ Z^* \end{bmatrix}^T L_S \begin{bmatrix} Y \\ Z^* \end{bmatrix}\right)\\ &= \ g_1(Y, X; \psi) + \sum_{i=1}^{n} \phi(y_i)\\ &+ \ 2\lambda_0 tr[Y^T L_C Y] + \lambda_1 tr[Y^T \bar{L}_S Y],\end{aligned}$$

where $\bar{L}_S \triangleq \bar{D}_S - \bar{A}_S$, with $\bar{A}_S \triangleq BD_H^{-1}B^T$ and $\bar{D}_S$ a diagonal matrix with nonzero elements formed as the corresponding row-sums of $\bar{A}_S$. Moreover, if $\mathcal{G}$ is m_e-uniform,[2] then under the same assumptions

$$\min_Z \ \ell(Y, Z; \psi) = g_1(Y, X; \psi) + \sum_{i=1}^{n} \phi(y_i) + \beta tr[Y^T L_C Y], \qquad (5)$$

where $\beta \triangleq 2\lambda_0 + \frac{\lambda_1}{m_e}$.

All proofs are deferred to Appendix C. This last result demonstrates that, under the stated assumptions, the graph-dependent portion of the original hypergraph energy, after optimizing away the influence of Z, can be reduced to a weighted quadratic penalty involving the graph Laplacian of the clique expansion. Moreover, this factor further resolves as

$$\text{tr}[Y^T L_C Y] = \frac{1}{2} \sum_{e_k \in \mathcal{E}} \sum_{i \in e_k} \sum_{j \in e_k} ||y_i - y_j||_2^2. \qquad (6)$$

Of course in more general settings, for example when $H_0 \neq H_1 \neq I$, or when $\phi(p) \neq \sum_{j=1}^{d} \mathcal{I}_\infty[p_i < 0]$, this equivalence will *not* generally hold.

4.4. Revised Hypergraph Energy Functions

The analysis from the previous sections motivates two practical, revised forms of our original energy from (1), which we will later use for all of our empirical evaluations. For convenience, we define

$$\ell(Y; \psi) \triangleq \ell(Y, Z = Z^*; \psi). \qquad (7)$$

Then the first, more general variant, we adopt is

$$\begin{aligned}&\ell(Y; \psi = \{W, H_0, H_1\})\\ &= \ ||Y - f(X; W)||_{\mathcal{F}}^2 + \sum_i \phi(y_i) +\\ &\overbrace{\lambda_0 \ \text{tr}\left[(YH_0)^T D_C Y H_0 - 2(YH_0)^T A_C Y + Y^T D_C Y\right]}^{(a)} +\\ &\overbrace{\lambda_1 \ \text{tr}\left[(YH_1)^T \bar{D}_S Y H_1 - 2(YH_1)^T B Z^* + Z^{*T} D_H Z^*\right]}^{(b)},\end{aligned} \qquad (8)$$

where $\bar{D}_S$ is defined as in Proposition 4.1. Moreover, to ease later exposition, we have overloaded the definition of f such that $\|Y - f(X; W)\|_{\mathcal{F}}^2 \equiv \sum_{i=1}^{n} \|y_i - f(x_i; W)\|_2^2$. And secondly, as a less complex alternative we have

$$\begin{aligned}&\ell(Y; \psi = \{W, I, I\}) = \qquad (9)\\ &||Y - f(X; W)||_{\mathcal{F}}^2 + \sum_i \phi(y_i) + \text{tr}[Y^T(\lambda_0 L_C + \lambda_1 \bar{L}_S)Y].\end{aligned}$$

5. Hypergraph Node Classification via Bilevel Optimization

We now demonstrate how the optimal embeddings obtained by minimizing the energy functions from the previous section can be applied to our ultimate goal of hypergraph node classification. For this purpose, define

$$Y^*(\psi) = \arg\min_Y \ell(Y; \psi), \qquad (10)$$

noting that the solution depends explicitly on the parameters ψ governing the shape of the energy. We may then consider treating $Y^*(\psi)$, which is obtainable from the above optimization process, as features to be applied to a discriminative node classification loss $\mathcal{D}$ that can be subsequently minimized via a second, meta-level optimization step.[3] In aggregate we arrive at the *bilevel* optimization problem

$$\ell(\theta, \psi) \triangleq \sum_{i=1}^{n'} \mathcal{D}(h[y_i^*(\psi); \theta], \tau_i), \qquad (11)$$

[2] An m_e-uniform hypergraph is such that every hyperedge joins exactly m_e nodes. Hence a regular graph is by default a 2-uniform hypergraph.

[3] Because our emphasis is hypergraph node classification, we will not explicitly use any analogous hyperedge embeddings for the meta-level optimization; however, they nonetheless still play a vital role given that they are co-adapted with the node embeddings during the lower-level optimization per the discussion from the previous section.

where $\mathcal{D}$ is chosen as an classification-friendly cross-entropy function, $y_i^*(\psi)$ is the i-th row of $Y^*(\psi)$, and $\tau_i \in \mathbb{R}^c$ is the ground-truth label of node i to be approximated by some differentiable node-wise function $h : \mathbb{R}^d \to \mathbb{R}^c$ with trainable parameters θ. We have also implicitly assumed that the first n' nodes of $\mathcal{G}$ are labeled. Intuitively, (11) involves training a classifier h, with input features $y_i^*(\psi)$, to predict labels τ_i.

At this point, assuming $\partial Y^*(\psi)/\partial\psi$ is somehow computable, then $\ell(\psi, \theta)$ can be efficiently trained over *all* parameters, including ψ from the lower level optimization. However, directly computing $\partial Y^*(\psi)/\partial\psi$ is not generally feasible. Instead, in the remainder of this section we will derive approximate embeddings $\hat{Y}(\psi) \approx Y^*(\psi)$ whereby $\partial\hat{Y}(\psi)/\partial\psi$ can be computed efficiently. And as will be assessed in greater detail later, the computational steps we derive to produce $\hat{Y}(\psi)$ will mirror the layers of canonical graph neural network architectures. It is because of this association that we refer to our overall model as **PhenomNN**, for *Purposeful Hyper-Edges iN Optimization Motivated Neural Networks* as mentioned in the introduction.

5.1. Deriving Proximal Gradient Descent Steps

To efficiently deploy proximal gradient descent (PGD) (Parikh et al., 2014), we first must split our loss into a smooth, differentiable part, and a non-smooth but separable part. Hence we adopt the decomposition

$$\ell(Y;\psi) = \bar{\ell}(Y;\psi) + \sum_i \phi(y_i), \tag{12}$$

where $\bar{\ell}(Y;\psi)$ is defined by exclusion upon examining the original form of $\ell(Y;\psi)$. The relevant proximal operator is

$$\begin{aligned}\textbf{prox}_{\phi}(V) &\triangleq \arg\min_Y \frac{1}{2}||V-Y||_{\mathcal{F}}^2 + \sum_i \phi(y_i) \\ &= \max(0, V),\end{aligned} \tag{13}$$

where the max operator is assumed to apply elementwise. Subsequent PGD iterations for minimizing (12) are then computed as

$$\bar{Y}^{(t+1)} = Y^{(t)} - \alpha\Omega\nabla_{Y^{(t)}}\bar{\ell}(Y^{(t)};\psi) \tag{14}$$

$$Y^{(t+1)} = \max(0, \bar{Y}^{(t+1)}), \tag{15}$$

where α is a step-size parameter and Ω is a positive-definite pre-conditioner to be defined later. Incidentally, as will become apparent shortly, (14) will occupy the role of a pre-activation hypergraph neural network layer, while (15) provides a ReLU nonlinearity. A related association was previously noted within the context of traditional GNNs (Yang et al., 2021). We now examine two different choices for Ω and ψ that correspond with the general form from (8) and the simplified alternative from (9).

General Form. To compute (14), we consider term (a) and (b) from (8) separately. Beginning with (a), the corresponding gradient is

$$2D_CY - 2\tilde{Y}_C, \tag{16}$$

where $\tilde{Y}_C \triangleq A_CY(H_0 + H_0^T) - D_CYH_0H_0^T$. Similarly, for ($b$) the gradient is given by

$$2BD_H^{-1}D_H(BD_H^{-1})^TY - 2\tilde{Y}_S, \tag{17}$$

where $\tilde{Y}_S \triangleq (B(BD_H^{-1})^TYH_1^T + BD_H^{-1}B^TYH_1) - \bar{D}_SYH_1H_1^T$. Additionally, given that $BD_H^{-1}D_H(BD_H^{-1})^T = B(BD_H^{-1})^T = BD_H^{-1}B^T = \bar{A}_S$, we can reduce (17) to

$$2\bar{A}_SY - 2\tilde{Y}_S, \tag{18}$$

since now $\tilde{Y}_S = \bar{A}_SY(H_1 + H_1^T) - \bar{D}_SYH_1H_1^T$. Combining terms, the gradient for $\bar{\ell}(Y;\psi)$ is

$$\begin{aligned}\frac{\bar{\ell}(Y;\psi)}{\partial Y} = 2\lambda_0(D_CY - \tilde{Y}_C) + 2\lambda_1(\bar{A}_SY - \tilde{Y}_S) \\ +2Y - 2f(X;W),\end{aligned} \tag{19}$$

and (14) becomes

$$\begin{aligned}\bar{Y}^{(t+1)} = Y^{(t)} - \alpha\Big[\lambda_0(D_CY^{(t)} - \tilde{Y}_C^{(t)}) \\ +\lambda_1(\bar{A}_SY^{(t)} - \tilde{Y}_S^{(t)}) + Y^{(t)} - f(X;W)\Big],\end{aligned} \tag{20}$$

where $\alpha/2$ is the step size. The coefficient $\bar{\Omega}$ before $Y^{(t)}$ is

$$\bar{\Omega} \triangleq \lambda_0D_C + \lambda_1\bar{A}_S + I. \tag{21}$$

Applying Jacobi preconditioning (Axelsson, 1996) often aids convergence by helping to normalize the scales across different dimensions. One natural candidate for the preconditioner is $\left(\text{diag}[\bar{\Omega}]\right)^{-1}$; however, we use the more spartan $\Omega = \tilde{D}^{-1}$ where $\tilde{D} \triangleq \lambda_0D_C + \lambda_1\bar{D}_S + I$. After rescaling and applying (15), the composite PhenomNN update is given by

$$\begin{aligned}Y^{(t+1)} = \text{ReLU}\Bigg((1-\alpha)Y^{(t)} + \alpha\tilde{D}^{-1}\Big[f(X;W) \\ + \lambda_0\tilde{Y}_C^{(t)} + \lambda_1(\bar{L}_SY^{(t)} + \tilde{Y}_S^{(t)})\Big]\Bigg),\end{aligned} \tag{22}$$

where $\bar{L}_S = \bar{D}_S - \bar{A}_S$ as in Proposition 4.1. This respresents the general form of PhenomNN.

Simplified Alternative. Regarding the simplified energy from (9), the relevant gradient is

$$\begin{aligned}\frac{\partial\bar{\ell}(Y;\psi = W, I, I)}{\partial Y} = 2(\lambda_0L_C + \lambda_1\bar{L}_S)Y + \\ 2Y - 2f(X;W),\end{aligned} \tag{23}$$

leading to the revised update

$$\bar{Y}^{(t+1)} = Y^{(t)} - \alpha\left[\tilde{\Omega}Y^{(t)} - f\left(X;W\right)\right], \quad \text{with } \tilde{\Omega} \triangleq \lambda_0 L_C + \lambda_1 \bar{L}_S + I \tag{24}$$

and step size $\alpha/2$ as before. And again, we can apply preconditioning, in this case rescaling each gradient step by $\Omega = \left(\text{diag}[\tilde{\Omega}]\right)^{-1} = \left(\lambda_0 D_C + \lambda_1 \bar{D}_S + I\right)^{-1} = \tilde{D}^{-1}$. So the final/composite update formula, including (15), becomes

$$Y^{(t+1)} = \text{ReLU}\Big((1-\alpha)Y^{(t)} + \alpha\tilde{D}^{-1}\left[(\lambda_0 A_C + \lambda_1 \bar{A}_S)Y^{(t)} + f\left(X;W\right)\right]\Big). \tag{25}$$

We henceforth refer to this variant as **PhenomNN$_{\text{simple}}$**.

5.2. Overall Algorithm

The overall algorithm for PhenomNN is demonstrated in Algorithm 1.

Algorithm 1 PhenomNN Algorithm for Hypergraph Node Classification.

Input: Hypergraph incidence matrix B, node features X, number of layers T, training epochs E, and node labels $\tau = \{\tau_i\}$.
for $e = 0$ to $E-1$ **do**
  Set initial projection $Y^{(0)} = f(X;W)$, where f is the trainable base model.
  for $t = 0$ to $T-1$ **do**
    $Y^{(t+1)} = Update(Y^{(t)})$, where $Update$ is computed via (22) for PhenomNN or (25) for PhenomNN$_{\text{simple}}$.
  end for
  Compute loss $\ell(\theta,\psi) = \sum_i \mathcal{D}(h[y_i^{(T)};\theta],\tau_i)$ from (11), where $\psi = \{W, H_0, H_1\}$ for PhenomNN and $\psi = \{W, I, I\}$ for PhenomNN$_{\text{simple}}$, noting that each $y_i^{(T)}$ is a trainable function of ψ by design.
  Backpropagate over all parameters ψ, θ using optimizer (Adam, SGD, etc.)
end for

5.3. Convergence Analysis

We now consider the convergence of the iterations (22) and (25) introduced in the previous section. First, for the more general form we have the following:

Proposition 5.1. *The PhenomNN updates from (22) are guaranteed to monotonically converge to the unique global minimum of $\ell(Y;\psi)$ on the condition that*

$$\alpha < \frac{1 + \lambda_0 d_{Cmin} + \lambda_1 d_{Smin}}{1 + \lambda_0 d_{Cmin} + \sigma_{max}}, \tag{26}$$

where d_{Cmin} is the minimum diagonal element of $I \otimes D_C$, d_{Smin} is the minimum diagonal element of $I \otimes \bar{D}_S$ and σ_{max} is the max eigenvalue of $(Q - P + \lambda_1 I \otimes \bar{A}_S)$ with

$$Q \triangleq \lambda_0 H_0^T H_0 \otimes D_C + \lambda_1 H_1^T H_1 \otimes \bar{D}_S, \tag{27}$$

$$P \triangleq \lambda_0 (H_0 + H_0^T) \otimes A_C + \lambda_1 (H_1 + H_1^T) \otimes \bar{A}_S. \tag{28}$$

And for the restricted case where $\psi = \{W, I, I\}$, the convergence conditions simplify as follows:

Corollary 5.2. *The PhenomNN$_{simple}$ updates from (25) are guaranteed to monotonically converge to the unique global minimum of $\ell(Y;\psi = \{W,I,I\})$ on the condition that*

$$\alpha < \frac{1 + \lambda_0 d_{Cmin} + \lambda_1 d_{Smin}}{1 + \lambda_0 d_{Cmin} + \lambda_1 d_{Smin} - \sigma_{min}}, \tag{29}$$

where σ_{min} is the min eigenvalue of $(\lambda_0 A_C + \lambda_1 \bar{A}_S)$.

5.4. Complexity Analysis

Analytically, PhenomNN$_{\text{simple}}$ has a time complexity given by $O(|\mathcal{E}|Td + |\mathcal{V}|Pd^2)$, where $|\mathcal{E}|$ is edge number, $|\mathcal{V}|$ is the node number, T is the number of layers/iterations, d is the hidden size, and P is the number of MLP layers in $f(\cdot;W)$. In contrast, for PhenomNN this complexity increases to $O(|\mathcal{E}|Td + |\mathcal{V}|(T+P)d^2)$, which is roughly the same as a standard GCN model. In fact, the widely-used graph convolution networks (GCN) (Kipf & Welling, 2016) have equivalent complexity to PhenomNN up to the factor of P which is generally small (e.g., $P = 1$ for PhenomNN in our experiments, while for a GCN $P = 0$). In this way then, PhenomNN$_{\text{simple}}$ is actually somewhat cheaper than a GCN when $T > P$. Additionally, we include complementary empirical results related to time and space complexity in Section 7.

6. Connections with Existing GNN Layers

As mentioned in Section 4.3, the clique and star expansions can be invoked to transform hypergraphs into homogeneous and bipartite graphs respectively (where the latter is a special case of a heterogeneous graph). In this section we examine how the layer-wise structure of two of the most popular GNN models, namely GCN (Kipf & Welling, 2016) mentioned previously, and relational graph convolution networks (RGCN) (Schlichtkrull et al., 2018), relate to PhenomNN and simplifications thereof.

6.1. Homogeneous Graphs and GCN

Using the so-called message-passing form of expression, the embedding update for the i-th node of the t-th GCN

layer can be written as

$$y_i^{(t+1)} = \sigma\left(\sum_{j\in\mathcal{N}_i}\frac{1}{c_{ij}}W^{(t)}y_j^{(t)}\right) \tag{30}$$

where σ is an activation function like ReLU, $W^{(t)}$ are weights, $c_{ij} \triangleq \sqrt{|\mathcal{N}_i||\mathcal{N}_j|}$ and $\mathcal{N}_i$ refers to the set of neighboring nodes in some input graph (note also that the graph could have self-loops in which case $i \in \mathcal{N}_i$). Interestingly, follow-up work (Ma et al., 2020; Pan et al., 2021; Yang et al., 2021; Zhang et al., 2020; Zhu et al., 2021) has demonstrated that this same basic layer-wise structure can be closely linked to iterative steps designed to minimize the energy

$$\ell(Y) = ||Y - f(X;W)||_{\mathcal{F}}^2 + \lambda \text{tr}[Y^T L Y], \tag{31}$$

where f is defined as before and L is the assumed graph Laplacian matrix. One way to see this is to examine a preconditioned gradient step along (31), which can be expressed as

$$Y^{(t+1)} = (1-\alpha)Y^{(t)} + \alpha\widetilde{D}_0^{-1}\left[\lambda A Y^{(t)} + f\left(X;W\right)\right], \tag{32}$$

with preconditioner $\tilde{D}_0^{-1} = (\lambda D + I)^{-1}$, step-size parameter α, graph adjacency matrix A, and corresponding degree matrix D. Moreover, for a single node i, (32) can be reduced to

$$y_i^{(t+1)} = \left(\sum_{j\in\mathcal{N}_i}\frac{1}{\tilde{c}_i}y_j^{(t)}\right) + \tilde{f}_i(x_i;W), \tag{33}$$

where $\tilde{c}_i$ is a scaling constant dependent on λ, the gradient step-size, and the preconditioner, while $\tilde{f}_i$ is merely f similarly rescaled. If we add an additional penalty ϕ and subsequent proximal operator step to introduce a non-linearity, then this result is very similar to (30), although without the weight matrix directly on each $y_j^{(t)}$ but with an added skip connection to the input layer.

Importantly for our purposes though, if the input graph is chosen to be a hypergraph clique expansion, and we set $D = D_C$, $A = A_C$, $\lambda = \lambda_0$, and $\lambda_1 = 0$, then we arrive at a special case of PhenomNN$_{\text{simple}}$ from (25). Of course one might not naturally conceive of the more generalized form that leads to PhenomNN$_{\text{simple}}$, and by extension PhenomNN, without the interpretable grounding of the underlying hypergraph energy functions involved.

6.2. Heterogeneous Graphs and RGCN

For heterogeneous graphs applied to RGCN, the analogous message-passing update for the i-th node in the t-th layer is given by

$$y_i^{(t+1)} = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_i^r}\frac{1}{c_{i,r}}y_j^{(t)}W_r^{(t)} + y_i^{(t)}W_0^{(t)}\right), \tag{34}$$

where $\mathcal{R}$ is the set of edge types in a heterogeneous input graph, $\mathcal{N}_i^r$ is the set of neighbors with edge type r, $c_{i,r} \triangleq |\mathcal{N}_i^r|$, and $W_r^{(t)}$ and $W_0^{(t)}$ are weight/projection matrices. In this context, the RGCN input could conceivably be chosen as the bipartite graph produced by a given star expansion (e.g., such a graph could be assigned the edge types "hypergraph node belongs to hyperedge" and "hyperedge belongs to hypergraph node").

For comparison purposes, we can also re-express our general PhenomNN model from (22), in the node-wise message-passing form

$$y_i^{(t+1)} = \sigma\left(\sum_{j\in\mathcal{N}_i^C} y_j^{(t)}W_{ij}^{(t)} + y_i^{(t)}W_i^{(t)} + \alpha\tilde{D}_{ii}^{-1}f\left(x_i;W\right)\right), \tag{35}$$

where $\mathcal{N}_i^C$ are neighbors in the clique (not star) expansion graph (more on this below) and the weight matrices are characterized by the special energy-function-dependent forms

$$W_{ij}^{(t)} \triangleq \alpha\tilde{D}_{ii}^{-1}\Big[\lambda_0 A_C[i,j](H_0 + H_0^T) + \lambda_1\bar{A}_S[i,j](H_1 + H_1^T - I)\Big], \tag{36}$$

$$W_i^{(t)} \triangleq (1-\alpha)I - \alpha\tilde{D}_{ii}^{-1}\Big[\lambda_0 D_C[i,i]H_0H_0^T + \lambda_1\bar{D}_S[i,i](H_1H_1^T - I)\Big]. \tag{37}$$

While the basic structures of (34) and (35) are similar, there are several key differences:

- When RGCN is applied to the star expansion, neighbors are defined by the resulting bipartite graph, and nodes in the original hypergraph do not directly pass messages to each other. In contrast, because within PhenomNN we have optimized away the hyperedge embeddings, the implicit graph that dictates neighborhood structure is actually the *clique expansion graph* as reflected in (35).
- The PhenomNN projection matrices have special structure infused from the energy function and optimization over the edge embeddings. As such, unlike RGCN node i receives messages from its connected neighbors and itself, with projection matrices $W_{ij}^{(t)}$ and $W_i^{(t)}$ that can vary from node to node and edge to edge. In contrast, RGCN has layer-wise (or analogously iteration-wise) dependent weights.

Table 1. Results on datasets from (Zhang et al., 2022): Mean accuracy (%) ± standard deviation results over 10 train-test splits. Boldfaced letters are used to indicate the best mean accuracy and underline for the second. "-" means not reported in their paper so in average ranking we just average over the ones that are available. OOM indicates out-of-memory.

	Cora (co-authorship)	DBLP (co-authorship)	Cora (co-citation)	Pubmed (co-citation)	Citeseer (co-citation)	NTU2012 (both features)	ModelNet40 (both features)	Avg Ranking
MLP+HLR	59.8 ± 4.7	63.6 ± 4.7	61.0 ± 4.1	64.7 ± 3.1	56.1 ± 2.6	-	-	13.6
FastHyperGCN	61.1 ± 8.2	68.1 ± 9.6	61.3 ± 10.3	65.7 ± 11.1	56.2 ± 8.1	-	-	12.4
HyperGCN	63.9 ± 7.3	70.9 ± 8.3	62.5 ± 9.7	68.3 ± 9.5	57.3 ± 7.3	-	-	10.8
HGNN	63.2 ± 3.1	68.1 ± 9.6	70.9 ± 2.9	66.8 ± 3.7	56.7 ± 3.8	83.54 ± 0.50	97.15 ± 0.14	9.4
HNHN	64.0 ± 2.4	84.4 ± 0.3	41.6 ± 3.1	41.9 ± 4.7	33.6 ± 2.1	-	-	13.0
HGAT	65.4 ± 1.5	OOM	52.2 ± 3.5	46.3 ± 0.5	38.3 ± 1.5	84.05 ± 0.36	96.44 ± 0.15	12.0
HyperSAGE	72.4 ± 1.6	77.4 ± 3.8	69.3 ± 2.7	72.9 ± 1.3	61.8 ± 2.3	-	-	8.6
UniGNN	75.3 ± 1.2	88.8 ± 0.2	70.1 ± 1.4	74.4 ± 1.0	63.6 ± 1.3	84.45 ± 0.40	96.69 ± 0.07	6.0
H-ChebNet	70.6 ± 2.1	87.9 ± 0.24	69.7 ± 2.0	74.3 ± 1.5	63.5 ± 1.3	83.16 ± 0.46	96.95 ± 0.09	8.0
H-APPNP	76.4 ± 0.8	89.4 ± 0.18	70.9 ± 0.7	75.3 ± 1.1	64.5 ± 1.4	83.57 ± 0.42	97.20 ± 0.14	4.6
H-SSGC	72.0 ± 1.2	88.6 ± 0.16	68.8 ± 2.1	74.5 ± 1.3	60.5 ± 1.7	84.13 ± 0.34	97.07 ± 0.07	7.6
H-GCN	74.8 ± 0.9	89.0 ± 0.19	69.5 ± 2.0	75.4 ± 1.2	62.7 ± 1.2	84.45 ± 0.40	97.28 ± 0.15	5.4
H-GCNII	76.2 ± 1.0	89.8 ± 0.20	72.5 ± 1.2	75.8 ± 1.1	64.5 ± 1.0	85.17 ± 0.36	97.75 ± 0.07	3.0
PhenomNN$_{\text{simple}}$	**77.62 ± 1.30**	89.74 ± 0.16	72.81 ± 1.67	76.20 ± 1.41	65.07 ± 1.08	85.39 ± 0.40	**97.83 ± 0.09**	1.9
PhenomNN	77.11 ± 0.45	**89.81 ± 0.05**	**73.09 ± 0.65**	**78.12 ± 0.24**	**65.77 ± 0.45**	**85.40 ± 0.42**	97.77 ± 0.11	**1.3**

- PhenomNN has an additional weighted skip connection from the input base model $f(x_i; W)$. While of course RGCN could also be equipped with a similar term, this would be accomplished in a post-hoc fashion, and not tethered to an underlying energy function.

7. Hypernode Classification Experiments

In this section we evaluate PhenomNN$_{\text{simple}}$ and PhenomNN on various hypergraph benchmarks focusing on hypernode classification and compare against previous SOTA approaches.

Datasets. Existing hypergraph benchmarks mainly focus on hypernode classification. We adopt five public citation network datasets from (Zhang et al., 2022): Co-authorship/Cora,Co-authorship/DBLP, Co-citaion/Cora, Co-citaion/Pubmed, Co-citaion/Citeseer. These datasets and splits are constructed by (Yadati et al., 2019) (https://github.com/malllabiisc/HyperGCN). We also adopt two other public visual object classification datasets: Princeton ModelNet40 (Wu et al., 2015) and the National Taiwan University (NTU) 3D model dataset (Chen et al., 2003). We follow HGNN (Feng et al., 2019) to preprocess the data by MVCNN (Su et al., 2015) and GVCNN (Feng et al., 2018) and obtain the hypergraphs. Additionally, we use the datasets provided by the public code (https://github.com/iMoonLab/HGNN) associated with (Feng et al., 2019). Finally, (Chien et al., 2022) construct a public hypergraph benchmark for hypernode classification which includes ModelNet40*, NTU2012*, Yelp (Yelp), House (Chodrow et al., 2021), Walmart (Amburg et al., 2020), and 20News (Dua & Graff, 2017). ModelNet40* and NTU2012* have the same raw data as ModelNet40 and NTU2012 mentioned before in (Zhang et al., 2022) but different splits. All datasets from (Chien et al., 2022) are downloaded from their code site (https://github.com/jianhao2016/AllSet).[4]

Baselines. For datasets from (Zhang et al., 2022), we adopt the baselines from their paper which includes a multi-layer perceptron with explicit hypergraph Laplacian regularization (MLP+HLR), FastHyperGCN (Yadati et al., 2019), HyperGCN (Yadati et al., 2019), HGNN (Feng et al., 2019), HNHN (Dong et al., 2020), HGAT (Ding et al., 2020), HyperSAGE (Arya et al., 2020), UniGNN (Huang & Yang, 2021), and various hypergraph GNNs (H-GNNs) (Zhang et al., 2022) proposed by them. For datasets from (Chien et al., 2022), we also select baselines from their paper including an MLP, CE (Clique Expansion)+GCN, CE+GAT, HNHN, HGNN, HCHA (Bai et al., 2021), HyperGCN, UniGCNII (Huang & Yang, 2021), HAN (Wang et al., 2019b) with full batch and mini-batch settings, and AllsetTransformer and AllDeepSets (Chien et al., 2022).

Implementations. We use a one-layer MLP for $f(X; W)$. Also, in practice we found that only using ReLU at the end of propagation steps works well. Detailed hyperparameter settings are deferred to Appendix D. We choose the hidden dimension of our models to be the same or less than the baselines in previous work. For results in Table 1, we conduct experiments on 10 different train-test splits and report average accuracy of test samples following (Zhang et al., 2022). For results in Table 2, we randomly split the data into training/validation/test samples using (50%/25%/25%) splitting percentages as in (Chien et al., 2022) and report

[4] Note that we excluded a few datasets for the following reasons: The Zoo dataset is very small; the Mushroom dataset is too easy; the Citation datasets are similar to (Zhang et al., 2022), and since we have ModelNet40* and NTU2012* for comparison of different baselines from both papers, we did not select them.

Table 2. Results using the benchmarks from (Chien et al., 2022): Mean accuracy (%) ± standard deviation. The number behind Walmart and House is the feature noise standard deviation for each dataset, and for HAN*, additional preprocessing of each dataset is required (see (Chien et al., 2022) for more details). Boldfaced letters are used to indicate the best mean accuracy and underline is for the second. OOM indicates out-of-memory.

	NTU2012*	ModelNet40*	Yelp	House(1)	Walmart(1)	House(0.6)	Walmart(0.6)	20Newsgroups	Avg Ranking
MLP	85.52 ± 1.49	96.14 ± 0.36	31.96 ± 0.44	67.93 ± 2.33	45.51 ± 0.24	81.53 ± 2.26	63.28 ± 0.37	81.42 ± 0.49	6.9
CEGCN	81.52 ± 1.43	89.92 ± 0.46	OOM	62.80 ± 2.61	54.44 ± 0.24	64.36 ± 2.41	59.78 ± 0.32	OOM	11.5
CEGAT	82.21 ± 1.23	92.52 ± 0.39	OOM	69.09 ± 3.00	51.14 ± 0.56	77.25 ± 2.53	59.47 ± 1.05	OOM	10.4
HNHN	89.11 ± 1.44	97.84 ± 0.25	31.65 ± 0.44	67.80 ± 2.59	47.18 ± 0.35	78.78 ± 1.88	65.80 ± 0.39	81.35 ± 0.61	7.1
HGNN	87.72 ± 1.35	95.44 ± 0.33	33.04 ± 0.62	61.39 ± 2.96	62.00 ± 0.24	66.16 ± 1.80	77.72 ± 0.21	80.33 ± 0.42	7.8
HCHA	87.48 ± 1.87	94.48 ± 0.28	30.99 ± 0.72	61.36 ± 2.53	62.45 ± 0.26	67.91 ± 2.26	77.12 ± 0.26	80.33 ± 0.80	8.8
HyperGCN	56.36 ± 4.86	75.89 ± 5.26	29.42 ± 1.54	48.31 ± 2.93	44.74 ± 2.81	78.22 ± 2.46	55.31 ± 0.30	81.05 ± 0.59	12
UniGCNII	89.30 ± 1.33	98.07 ± 0.23	31.70 ± 0.52	67.25 ± 2.57	54.45 ± 0.37	80.65 ± 1.96	72.08 ± 0.28	81.12 ± 0.67	6.2
HAN (full batch)*	83.58 ± 1.46	94.04 ± 0.41	OOM	71.05 ± 2.26	OOM	83.27 ± 1.62	OOM	OOM	9.6
HAN (mini batch)*	80.77 ± 2.36	91.52 ± 0.96	26.05 ± 1.37	62.00 ± 9.06	48.57 ± 1.04	82.04 ± 2.68	63.10 ± 0.96	79.72 ± 0.62	10.4
AllDeepSets	88.09 ± 1.52	96.98 ± 0.26	30.36 ± 1.57	67.82 ± 2.40	64.55 ± 0.33	80.70 ± 1.59	**78.46 ± 0.26**	81.06 ± 0.54	5.6
AllSetTransformer	88.69 ± 1.24	98.20 ± 0.20	**36.89 ± 0.51**	69.33 ± 2.20	**65.46 ± 0.25**	83.14 ± 1.92	**78.46 ± 0.40**	81.38± 0.58	3.1
PhenomNN$_{\text{simple}}$	**91.03 ± 1.04**	**98.66 ± 0.20**	32.26 ± 0.40	**71.77 ± 1.68**	**64.11 ± 0.49**	**86.96 ± 1.33**	**78.46 ± 0.32**	**81.74 ± 0.52**	**1.6**
PhenomNN	90.62 ± 1.88	98.61 ± 0.17	31.92 ± 0.36	70.71 ± 2.35	62.98 ± 1.36	85.28 ± 2.30	78.26 ± 0.26	81.41 ± 0.49	3.1

the average accuracy over ten random splits. All experiments are implemented on RTX 3090 with Pytorch and DGL (Wang et al., 2019a).

Results. As shown in Table 1, our models achieve the best performance and top ranking on all datasets from (Zhang et al., 2022) compared to previous baselines. And in Table 2, our models achieve the first (PhenomNN$_{\text{simple}}$) and tied-for-second (PhenomNN) overall performance ranking on the benchmarks from (Chien et al., 2022).

Empirical evaluation of time and space complexity. In practice, we find that PhenomNN is roughly $2\times$ to $3\times$ slower than a GCN given the integration of two expansions based on H_0 and H_1, which implies that the constant multiplying the theoretical complexity from above is at least doubled as expected. Of course timing results will still vary based on hardware and implementation details. As an example, we measure the training time of GCN and our models on the same hardware on Coauthorship-DBLP data with hidden size 64 and 8 layers. We observe 0.047s/epoch for GCN and 0.045s/epoch for PhenomNN$_{\text{simple}}$ and 0.143s/epoch for PhenomNN under these conditions. In terms of the space efficiency, our models are also analytically similar to common GNNs. And under the same settings as above, the memory consumption is 1665MB for GCN, 1895MB for PhenomNN$_{\text{simple}}$, and 2424MB for PhenomNN.

Ablations. For space considerations, we defer ablations to Appendix B; however, we nonetheless highlight some of our findings here. For example, in Table 5 (Appendix B) we demonstrate the effect of different hypergraph energy function terms, which are associated with different hypergraph expansions per Proposition 4.1. In brief here, we explore different selections of $\{\lambda_0, \lambda_1\} \in \{\{0,1\},\{1,0\},\{1,1\},\}$ which in effect modulate the inclusion of clique- and star-like expansion factors. Results demonstrate that on most datasets, the combination of both expansions, with their complementary roles, is beneficial.

We also explore the tolerance of our model to different hidden dimensions in Table 6. In brief, we fix other hyperparameters and obtain results across different hidden dimensions with PhenomNN$_{\text{simple}}$ for simplicity; results for PhenomNN are similar. Overall, this ablation demonstrates the stability of our approach across hidden dimension.

Additional comparisons and discussion. As suggested by reviewers, we include additional discussion and comparison with existing work in Appendix A due to the page limit. This includes side-by-side evaluations with RGCN and the model from (Wang et al., 2023) which was not yet published at the time of our original submission.

8. Conclusion

While hypergraphs introduce compelling modeling flexibility, they still remain relatively under-explored in the GNN literature. With the potential to better understand hypergraph properties and expand their utility, we have introduced an expressive family of hypergraph energy functions and fleshed out their connection with previous hypergraph expansions. We then leverage this perspective to design what can be interpreted as hypergraph neural network layers that are in one-to-one correspondence with proximal gradient steps descending these energies. We also characterize the similarities and differences of these layers w.r.t. popular existing GNN architectures. In the end, the proposed framework achieves competitive or SOTA performance on key hypergraph node classification benchmarks.

9. Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 62236004 and No. 62022027)

and the Major Key Project of PCL (PCL2021A12).

References

Agarwal, S., Lim, J., Zelnik-Manor, L., Perona, P., Kriegman, D., and Belongie, S. Beyond pairwise clustering. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pp. 838–845. IEEE, 2005.

Agarwal, S., Branson, K., and Belongie, S. Higher order learning with graphs. In *Proceedings of the 23rd international conference on Machine learning*, pp. 17–24, 2006.

Ahn, H., Yang, Y., Gan, Q., Moon, T., and Wipf, D. P. Descent steps of a relation-aware energy produce heterogeneous graph neural networks. *Advances in Neural Information Processing Systems*, 35:38436–38448, 2022.

Amburg, I., Veldt, N., and Benson, A. Clustering in graphs and hypergraphs with categorical edge labels. In *Proceedings of The Web Conference 2020*, WWW '20, pp. 706–717, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450370233. doi: 10.1145/3366423.3380152. URL https://doi.org/10.1145/3366423.3380152.

Aponte, R., Rossi, R. A., Guo, S., Hoffswell, J., Lipka, N., Xiao, C., Chan, G. Y.-Y., Koh, E., and Ahmed, N. K. A hypergraph neural network framework for learning hyperedge-dependent node embeddings. *ArXiv*, abs/2212.14077, 2022.

Arya, D., Gupta, D. K., Rudinac, S., and Worring, M. Hypersage: Generalizing inductive representation learning on hypergraphs. *arXiv preprint arXiv:2010.04558*, 2020.

Axelsson, O. *Iterative solution methods*. Cambridge university press, 1996.

Bai, S., Zhang, F., and Torr, P. H. Hypergraph convolution and hypergraph attention. *Pattern Recognition*, 110: 107637, 2021.

Benson, A. R., Gleich, D. F., and Leskovec, J. Higher-order organization of complex networks. *Science*, 353(6295): 163–166, 2016.

Benson, A. R., Gleich, D. F., and Lim, L.-H. The spacey random walk: A stochastic process for higher-order data. *SIAM Review*, 59(2):321–345, 2017.

Chan, T.-H. H. and Liang, Z. Generalizing the hypergraph laplacian via a diffusion process with mediators. *Theoretical Computer Science*, 806:416–428, 2020.

Chen, D.-Y., Tian, X.-P., Shen, Y.-T., and Ouhyoung, M. On visual similarity based 3d model retrieval. In *Computer graphics forum*, volume 22, pp. 223–232. Wiley Online Library, 2003.

Chen, S. and Eldar, Y. C. Graph signal denoising via unrolling networks. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5290–5294, 2021.

Chien, E., Pan, C., Peng, J., and Milenkovic, O. 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.

Chodrow, P. S., Veldt, N., and Benson, A. R. Hypergraph clustering: from blockmodels to modularity. *arXiv preprint arXiv:2101.09611*, 2021.

Ding, K., Wang, J., Li, J., Li, D., and Liu, H. Be more with less: Hypergraph attention networks for inductive text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 4927–4936, 2020.

Dong, Y., Sawin, W., and Bengio, Y. Hnhn: Hypergraph networks with hyperedge neurons. *arXiv preprint arXiv:2006.12278*, 2020.

Dua, D. and Graff, C. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Eswaran, D., Günnemann, S., Faloutsos, C., Makhija, D., and Kumar, M. ZooBP: Belief propagation for heterogeneous networks. In *International Conference on Very Large Databases*, 2017.

Feng, Y., Zhang, Z., Zhao, X., Ji, R., and Gao, Y. Gvcnn: Group-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 264–272, 2018.

Feng, Y., You, H., Zhang, Z., Ji, R., and Gao, Y. Hypergraph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3558–3565, 2019.

Georgiev, D., Brockschmidt, M., and Allamanis, M. Heat: Hyperedge attention networks. *ArXiv*, abs/2201.12113, 2022.

Henderson, H. V. and Searle, S. R. The vec-permutation matrix, the vec operator and kronecker products: a review. *Linear and Multilinear Algebra*, 1981.

Heydari, S. and Livi, L. F. Message passing neural networks for hypergraphs. In *ICANN*, 2022.

Huang, J. and Yang, J. Unignn: a unified framework for graph and hypergraph neural networks. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, 2021.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Li, P. and Milenkovic, O. Inhomogeneous hypergraph clustering with applications. *Advances in Neural Information Processing Systems*, 2017:2309–2319, 2017.

Liu, X., Jin, W., Ma, Y., Li, Y., Liu, H., Wang, Y., Yan, M., and Tang, J. Elastic graph neural networks. In *International Conference on Machine Learning*, 2021.

Ma, Y., Liu, X., Zhao, T., Liu, Y., Tang, J., and Shah, N. A unified view on graph neural networks as graph signal denoising. *arXiv preprint arXiv:2010.01777*, 2020.

Pan, X., Song, S., and Huang, G. A unified framework for convolution-based graph neural networks, 2021. URL https://openreview.net/forum?id=zUMD--Fb9Bt.

Parikh, N., Boyd, S., et al. Proximal algorithms. *Foundations and Trends® in Optimization*, 2014.

Schlichtkrull, M., Kipf, T. N., Bloem, P., Berg, R. v. d., Titov, I., and Welling, M. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pp. 593–607. Springer, 2018.

Srinivasan, B., Zheng, D., and Karypis, G. Learning over families of sets-hypergraph representation learning for higher order tasks. In *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, pp. 756–764. SIAM, 2021.

Su, H., Maji, S., Kalogerakis, E., and Learned-Miller, E. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pp. 945–953, 2015.

Wang, M., Zheng, D., Ye, Z., Gan, Q., Li, M., Song, X., Zhou, J., Ma, C., Yu, L., Gai, Y., Xiao, T., He, T., Karypis, G., Li, J., and Zhang, Z. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*, 2019a.

Wang, P., Yang, S., Liu, Y., Wang, Z., and Li, P. Equivariant hypergraph diffusion neural operators. In *International Conference on Learning Representations (ICLR)*, 2023.

Wang, X., Ji, H., Shi, C., Wang, B., Ye, Y., Cui, P., and Yu, P. S. Heterogeneous graph attention network. In *The World Wide Web Conference*, pp. 2022–2032, 2019b.

Wang, Z., Ling, Q., and Huang, T. Learning deep ℓ_0 encoders. In *AAAI Conference on Artificial Intelligence*, volume 30, 2016.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1912–1920, 2015.

Yadati, N., Nimishakavi, M., Yadav, P., Nitin, V., Louis, A., and Talukdar, P. Hypergcn: A new method for training graph convolutional networks on hypergraphs. In *Advances in Neural Information Processing Systems*, pp. 1511–1522, 2019.

Yamaguchi, Y., Faloutsos, C., and Kitagawa, H. CAMLP: Confidence-aware modulated label propagation. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, 2016.

Yang, C., Wang, R., Yao, S., and Abdelzaher, T. F. Hypergraph learning with line expansion. *ArXiv*, abs/2005.04843, 2020.

Yang, Y., Liu, T., Wang, Y., Zhou, J., Gan, Q., Wei, Z., Zhang, Z., Huang, Z., and Wipf, D. Graph neural networks inspired by classical iterative algorithms. In *International Conference on Machine Learning*, pp. 11773–11783. PMLR, 2021.

Yelp. Yelp business dataset. URL https://www.yelp.com/dataset.

Zhang, H., Yan, T., Xie, Z., Xia, Y., and Zhang, Y. Revisiting graph convolutional network on semi-supervised node classification from an optimization perspective. *CoRR*, 2020.

Zhang, J., Li, F., Xiao, X., Xu, T., Rong, Y., Huang, J., and Bian, Y. Hypergraph convolutional networks via equivalency between hypergraphs and undirected graphs. *ArXiv*, abs/2203.16939, 2022.

Zhou, D., Bousquet, O., Lal, T., Weston, J., and Schölkopf, B. Learning with local and global consistency. In *Advances in Neural Information Processing Systems*, 2003.

Zhu, M., Wang, X., Shi, C., Ji, H., and Cui, P. Interpreting and unifying graph neural networks with an optimization framework. *arXiv preprint arXiv:2101.11859*, 2021.

Zien, J., Schlag, M., and Chan, P. Multilevel spectral hypergraph partitioning with arbitrary vertex sizes. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 18(9):1389–1399, 1999. doi: 10.1109/43.784130.

A. Additional Comparisons with Existing Work

A.1. Further Discussion

In terms of expressiveness and generalizability, the primary difference between the AllSet from (Chien et al., 2022) and PhenomNN can be loosely distilled as follows: AllSet is explicitly designed for expanding per-layer expressive power as much as possible by combining principles from Deep Sets and SetTransformers. But this complexity may reduce the feasibility of exploring deeper models that spread information longer distances across a hypergraph, e.g., only a single layer model is used for the experiments in (Chien et al., 2022). In contrast, PhenomNN is motivated by harnessing the interpretable inductive biases that come from descending an explicit lower-level energy function, whose minima are trainable by a higher-level downstream classification task. In this way, PhenomNN can in principle include an arbitrary number of layers to pass information across the hypergraph, since additional layers merely iterate the embeddings closer to the energy function minimum. Given these considerations, both AllSet and PhenomNN both have merits, and neither model fully encompasses the other as a special case.

Another recently published work raised by reviewers (Wang et al., 2023) proposes a quite interesting model called ED-HNN (for equivariant diffusion hypergraph neural network). This approach is complementary to our submission and different in at least three key respects: First, although (Wang et al., 2023) use gradient diffusion processes to motivate a broad class of GNN models, that are in some ways similar to AllSets from (Chien et al., 2022), in the end there is not actually any specific energy function that is being minimized by their proposed Algorithm 1. Indeed there is no guarantee provided that each layer of their method is reducing any specific graph-regularized quantity of interest, which is our primary focus.

Secondly, their incorporation of proximal operators is fundamentally different than ours. In ED-HNN, MLPs are used to implicitly model the effect of arbitrary proximal operators within an iterative ADMM optimization scheme. However, although conceptually understandable, a general MLP can model any function while proximal operators must obey very stringent properties (e.g., in 1D they must be nondecreasing functions of the input argument). In contrast, we apply proximal gradient descent to an explicit energy function with strict convergence guarantees. And last but not least, we consider an energy function dependent on both node and hyperedge embeddings, while ED-HNN only considers node-wise embeddings (that are regrouped within a penalty for each hyperedge).

A.2. Extra Empirical Results

ED-HNN comparisons. We include five new benchmarks for comparison from ED-HNN (Wang et al., 2023) suggested by reviewers in Table 3 , where we observe that both models perform well relative to a wide variety of baselines.

Table 3. Extension datasets of Table 2 to be compared with ED-HNN. Results for ED-HNN are from (Wang et al., 2023), while results for other baselines are from (Chien et al., 2022).

	Cora	Citeseer	Pubmed	Cora-CA	DBLP-CA
MLP	75.17 ± 1.21	72.67 ± 1.56	87.47 ± 0.51	74.31 ± 1.89	84.83 ± 0.22
CECGN	76.17 ± 1.39	70.16 ± 1.31	86.45 ± 0.43	77.05 ± 1.26	88.00 ± 0.26
CEGAT	76.41 ± 1.53	70.63 ± 1.30	86.81 ± 0.42	76.16 ± 1.19	88.59 ± 0.29
HNHN	76.36 ± 1.92	72.64 ± 1.57	86.90 ± 0.30	77.19 ± 1.49	86.78 ± 0.29
HGNN	79.39 ± 1.36	72.45 ± 1.16	86.44 ± 0.44	82.64 ± 1.65	91.03 ± 0.20
HCHA	79.14 ± 1.02	72.42 ± 1.42	86.41 ± 0.36	82.55 ± 0.97	90.92 ± 0.22
HyperGCN	78.45 ± 1.26	71.28 ± 0.82	82.84 ± 8.67	79.48 ± 2.08	89.38 ± 0.25
UniGCNII	78.81 ± 1.05	73.05 ± 2.21	88.25 ± 0.40	83.60 ± 1.14	91.69 ± 0.19
HAN (full batch)*	80.18 ± 1.15	74.05 ± 1.43	86.21 ± 0.48	84.04 ± 1.02	90.89 ± 0.23
HAN (mini batch)*	79.70 ± 1.77	74.12 ± 1.52	85.32 ± 2.25	81.71 ± 1.73	90.17 ± 0.65
AllDeepSets	76.88 ± 1.80	70.83 ± 1.63	88.75 ± 0.33	81.97 ± 1.50	91.27 ± 0.27
AllSetTransformer	78.58 ± 1.47	73.08 ± 1.20	88.72 ± 0.37	83.63 ± 1.47	91.53 ± 0.23
ED-HNN	80.31 ± 1.35	73.70 ± 1.38	**89.03 ± 0.53**	83.97 ± 1.55	91.90 ± 0.19
$\text{PhenomNN}_{\text{simple}}$	81.98 ± 1.58	75.00 ± 0.58	88.25 ± 0.42	85.18 ± 0.97	91.91 ± 0.24
PhenomNN	**82.29 ± 1.42**	**75.10 ± 1.59**	88.07 ± 0.48	**85.81 ± 0.90**	**91.91 ± 0.21**

RGCN comparisons. Although we already provide comparisons with the heterogeneous GNN model HAN in Table 2, given the connection between PhenomNN and RGCN models detailed in Section 6.2, it also makes sense to provide further evaluations with the latter. To this end, Table 4 summarizes results comparing PhenomNN to heterogeneous graphs applied to the star expansion. HAN results are reproduced from the main paper. Meanwhile, for the RGCN for hypergraphs implementation, we use code from (Chien et al., 2022) which executes full-batch training that produces OOM for some datasets. In any event, from the available results we observe that our models can outperform both RGCN and the related heterogeneous GNN HAN models alike.

Table 4. Additional comparisons with heterogeneous GNN models applied to star expansions.

	NTU2012*	ModelNet40*	Yelp	House(1)	Walmart(1)	House(0.6)	Walmart(0.6)	20Newsgroups
HAN (full batch)*	83.58 ± 1.46	94.04 ± 0.41	OOM	71.05 ± 2.26	OOM	83.27 ± 1.62	OOM	OOM
HAN (mini batch)*	80.77 ± 2.36	91.52 ± 0.96	26.05 ± 1.37	62.00 ± 9.06	48.57 ± 1.04	82.04 ± 2.68	63.10 ± 0.96	79.72 ± 0.62
RGCN (full batch)*	86.74 ± 1.69	97.62 ± 0.32	OOM	66.38 ± 3.69	OOM	78.17 ± 2.74	OOM	OOM
$\text{PhenomNN}_{\text{simple}}$	**91.03 ± 1.04**	**98.66 ± 0.20**	**32.26 ± 0.40**	**71.77 ± 1.68**	**64.11 ± 0.49**	**86.96 ± 1.33**	**78.46 ± 0.32**	**81.74 ± 0.52**
PhenomNN	90.62 ± 1.88	98.61 ± 0.17	31.92 ± 0.36	70.71 ± 2.35	62.98 ± 1.36	85.28 ± 2.30	78.26 ± 0.26	81.41 ± 0.49

B. Ablation Tables

Table 5. Results for ablations of hypergraph expansion combinations under the same settings as applied in the main paper. Boldfaced letters indicate the best expansion compared with the same model.

	Cora (co-authorship)	DBLP (co-authorship)	Cora (co-citation)	Pubmed (co-citation)	Citeseer (co-citation)	NTU2012 (both features)	ModelNet40 (both features)	
$\text{PhenomNN}_{\text{simple}}$-**clique**	77.06 ± 1.27	89.54 ± 0.05	72.37 ± 1.49	75.71 ± 1.04	64.92 ± 1.56	85.36 ± 0.36	97.81 ± 0.09	
$\text{PhenomNN}_{\text{simple}}$-**star**	77.28 ± 1.27	89.54 ± 0.18	**72.81 ± 1.67**	**76.20 ± 1.41**	64.96 ± 1.13	85.31 ± 0.23	97.81 ± 0.08	
$\text{PhenomNN}_{\text{simple}}$	**77.62 ± 1.30**	**89.74 ± 0.16**	**72.81 ± 1.67**	**76.20 ± 1.41**	**65.07 ± 1.08**	**85.39 ± 0.40**	**97.83 ± 0.09**	
PhenomNN-**clique**	76.74 ± 0.41	89.56 ± 0.08	72.68 ± 0.63	77.94 ± 0.20	65.65 ± 0.34	85.15 ± 0.40	97.71 ± 0.15	
PhenomNN-**star**	76.83 ± 0.52	89.52 ± 0.05	**73.09 ± 0.65**	77.52 ± 0.34	65.46 ± 0.46	85.25 ± 0.38	**97.77 ± 0.11**	
PhenomNN	**77.11 ± 0.45**	**89.81 ± 0.05**	**73.09 ± 0.65**	**78.12 ± 0.24**	**65.77 ± 0.45**	**85.40 ± 0.42**	**97.77 ± 0.11**	
	NTU2012*	ModelNet40*	Yelp	House(1)	Walmart(1)	House(0.6)	Walmart(0.6)	20Newsgroups
$\text{PhenomNN}_{\text{simple}}$-**clique**	90.36 ± 1.80	98.64 ± 0.23	31.76 ± 0.42	70.93 ± 2.25	61.84 ± 0.66	86.40 ± 1.60	77.38 ± 0.17	**81.74 ± 0.52**
$\text{PhenomNN}_{\text{simple}}$-**star**	90.68 ± 1.38	98.50 ± 0.13	32.18 ± 0.41	71.21 ± 2.19	**64.11 ± 0.49**	86.26 ± 1.51	78.38 ± 0.21	81.47 ± 0.38
$\text{PhenomNN}_{\text{simple}}$	**91.03 ± 1.04**	**98.66 ± 0.20**	**32.26 ± 0.40**	**71.77 ± 1.68**	**64.11 ± 0.49**	**86.96 ± 1.33**	**78.46 ± 0.32**	**81.74 ± 0.52**
PhenomNN-**clique**	90.14 ± 1.26	98.55 ± 0.16	31.58 ± 0.53	70.37 ± 2.66	60.96 ± 0.37	85.00 ± 1.82	77.19 ± 0.25	81.07 ± 0.54
PhenomNN-**star**	90.38 ± 1.78	**98.61 ± 0.17**	**31.92 ± 0.36**	69.50 ± 2.34	**63.82 ± 0.49**	**85.22 ± 1.67**	**78.26 ± 0.26**	81.11 ± 0.36
PhenomNN	**90.62 ± 1.88**	**98.61 ± 0.17**	**31.92 ± 0.36**	**70.71 ± 2.35**	**63.82 ± 0.49**	**85.22 ± 1.67**	**78.26 ± 0.26**	**81.41 ± 0.49**

Table 6. Results with different hidden sizes of $\text{PhenomNN}_{\text{simple}}$. NTU2012*, ModelNet40*, House(1), and House(0.6) are four representative datasets from Table 2 in the main paper. The '/' symbol indicates that the result was not computed because this dimension is higher than what is used in our paper and previous works.

Model	Hidden	NTU2012*	ModelNet40*	House(1)	House(0.6)
$\text{PhenomNN}_{\text{simple}}$	512	/	98.66 ± 0.20	71.77 ± 1.68	86.96 ± 1.33
	256	91.03 ± 1.04	98.57 ± 0.14	69.38 ± 2.47	86.12 ± 2.11
	128	89.46 ± 1.39	98.42 ± 0.15	68.60 ± 1.96	84.56 ± 1.42
	64	89.96 ± 1.26	98.51 ± 0.21	68.66 ± 2.10	85.44 ± 1.46

C. Proofs

C.1. Proof of Proposition 4.1

We first reproduce the proposition here for ease of comparison. Suppose $g_2(Z, U; \psi)$ is removed from (1), $H_0 = H_1 = I$, and define $Z^* \triangleq D_H^{-T} B^T Y$. It then follows that

$$\begin{aligned} \min_Z \ \ell(Y, Z; \psi) &= g_1(Y, X; \psi) + \textstyle\sum_{i=1}^n \phi(y_i) + 2\lambda_0 \text{tr}[Y^T L_C Y] + \lambda_1 \text{tr}\left(\begin{bmatrix} Y \\ Z^* \end{bmatrix}^T L_S \begin{bmatrix} Y \\ Z^* \end{bmatrix} \right) \quad (38) \\ &= g_1(Y, X; \psi) + \textstyle\sum_{i=1}^n \phi(y_i) + 2\lambda_0 \text{tr}[Y^T L_C Y] + \lambda_1 \text{tr}[Y^T \bar{L}_S Y] \quad (39) \end{aligned}$$

Moreover, if $\mathcal{G}$ is m_e-uniform, then under the same assumptions

$$\min_Z \ \ell(Y, Z; \psi) = g_1(Y, X; \psi) + \sum_{i=1}^n \phi(y_i) + \beta \text{tr}[Y^T L_C Y], \quad (40)$$

where $\beta \triangleq 2\lambda_0 + \frac{\lambda_1}{m_e}$.

Proof. After removing $g_2(Z, U; \psi)$ from $\ell(Y, Z; \psi)$ and setting $H_0 = H_1 = I$, we have

$$\ell(Y, Z; \psi) = g_1(Y, X; \psi) + \sum_{i=1}^n \phi(y_i) + \sum_{k=1}^m \phi(z_i) + \lambda_0 \sum_{e_k \in \mathcal{E}} \sum_{i \in e_k} \sum_{j \in e_k} ||y_i - y_j||_2^2 + \lambda_1 \sum_{e_k \in \mathcal{E}} \sum_{i \in e_k} ||y_i - z_k||_2^2 \quad (41)$$

First we know

$$\sum_{e_k \in \mathcal{E}} \sum_{i \in e_k} \sum_{j \in e_k} ||y_i - y_j||_2^2 = 2\text{tr}[Y^T L_C Y] \quad (42)$$

from the definition of Laplacian L_C. Then we solve Z^* for minimizing (41). If there were no ϕ term, then it follows that $z_k^* = \text{MEAN}(y_i | i \in e_k)$, because the mean function minimizes the sum of squared errors. However, because each y_i is forced to be positive by ϕ, the resulting mean will also be positive and therefore feasible as well. Therefore, the mean estimator will remain optimal even if we include the ϕ term.

It can also be shown that the aforementioned mean estimator satisfies

$$Z^* = D_H^{-T} B^T Y, \quad (43)$$

and hence

$$\sum_{e_k \in \mathcal{E}} \sum_{i \in e_k} ||y_i - z_k^*||_2^2 = \text{tr}\left(\begin{bmatrix} Y \\ Z^* \end{bmatrix}^T L_S \begin{bmatrix} Y \\ Z^* \end{bmatrix} \right) \quad (44)$$

from the definition of Laplacian L_S. This expression then allows us to reproduce (38). Processing further, we have

$$\begin{aligned} \sum_{e_k \in \mathcal{E}} \sum_{i \in e_k} ||y_i - z_k^*||_2^2 &= \sum_{e_k \in \mathcal{E}} \sum_{i \in e_k} ||y_i - \frac{\sum_{j \in e_k} y_j}{m_{e_k}}||_2^2 \\ &= \sum_{e_k \in \mathcal{E}} \frac{1}{m_{e_k}} \sum_{i \in e_k} \sum_{j \in e_k} ||y_i - y_j||_2^2 \\ &= \text{tr}[Y^T \bar{L}_S Y], \end{aligned} \quad (45)$$

which leads to (39).

Recall the definition of $A_C = BB^T$ and $\bar{A}_S = BD_H^{-1}B^T$, so if $\mathcal{G}$ is m_e-uniform, it means all diagonal elements in D_H is m_e, so we have

$$\bar{A}_S = BD_H^{-1}B^T = \frac{1}{m_e} BB^T = \frac{1}{m_e} A_C. \quad (46)$$

From the definition of $\bar{L}_S$, we get

$$\begin{aligned}\bar{L}_S &= \bar{D}_S - \bar{A}_S \\ &= \frac{1}{m_e} D_C - \frac{1}{m_e} A_C \\ &= \frac{1}{m_e} L_C,\end{aligned} \tag{47}$$

which leads to (40). □

C.2. Proof of Proposition 5.1

It is notable that the updating consists of two parts (14) and (15) using the proximal gradient descent we discussed before. So the main point here is to prove the descent of (14) for $\psi = \{W, H_0, H_1\}$ for Proposition 5.1 and $\psi = \{W, I, I\}$ for Corollary 5.2 respectively.

We first provide a basic mathematical result.

Lemma C.1. *(Roth's Column Lemma (Henderson & Searle, 1981)). For any three matrices* $\mathbf{X}, \mathbf{Y}$ *and* $\mathbf{Z}$*,*

$$vec(\mathbf{XYZ}) = (\mathbf{Z}^\top \otimes \mathbf{X}) vec(\mathbf{Y}) \tag{48}$$

We now proceed with the proof of our result.

Proof. The gradient of $\bar{\ell}(Y; \psi)$ is as follows:

$$\nabla_Y \bar{\ell}(Y; \psi) = 2\left(\lambda_0 (D_C Y - \tilde{Y}_C) + \lambda_1 (\bar{A}_S Y - \tilde{Y}_S) + Y - f(X; W)\right), \tag{49}$$

where $\tilde{Y}_C = A_C Y (H_0 + H_0^T) - D_C Y H_0 H_0^T$ and $\tilde{Y}_S = \bar{A}_S Y (H_0 + H_0^T) - \bar{D}_S Y H_0 H_0^T$. We rewrite the equation in

$$\begin{aligned}\frac{\nabla_Y \bar{\ell}(Y; \psi)}{2} = (\mathbf{I} + \lambda_0 D_C + \lambda_1 \bar{A}_S) Y - f(X; W) \\ -\lambda_0 (A_C Y (H_0 + H_0^T) - D_C Y H_0 H_0^T) \\ -\lambda_1 (\bar{A}_S Y (H_0 + H_0^T) - \bar{D}_S Y H_0 H_0^T),\end{aligned} \tag{50}$$

We do vectorization on both sides of (50) to obtain:

$$\begin{aligned}\frac{vec(\nabla_Y \bar{\ell}(Y; \psi))}{2} = (\mathbf{I} + \lambda_0 I \otimes D_C + \lambda_1 I \otimes \bar{A}_S) vec(Y) - vec(f(X; W)) \\ -vec\left(\lambda_0 (A_C Y (H_0 + H_0^T) - D_C Y H_0 H_0^T)\right) \\ -vec\left(\lambda_1 (\bar{A}_S Y (H_0 + H_0^T) - \bar{D}_S Y H_0 H_0^T)\right),\end{aligned} \tag{51}$$

Here, using Roth's Column Lemma C.1 to rewrite equation (51)

$$\begin{aligned}\frac{vec(\nabla_Y \bar{\ell}(Y; \psi))}{2} = (\mathbf{I} + \lambda_0 I \otimes D_C + \lambda_1 I \otimes \bar{A}_S) vec(Y) - vec(f(X; W)) \\ -vec\left(\lambda_0 (H_0 + H_0^T) \otimes A_C Y\right) + vec\left(\lambda_0 H_0^T H_0 \otimes D_C Y\right) \\ -vec\left(\lambda_1 (H_0 + H_0^T) \otimes \bar{A}_S Y\right) + vec\left(\lambda_1 H_0^T H_0 \otimes \bar{D}_S Y\right),\end{aligned} \tag{52}$$

This is needed behind but for now we just leave it. Then we write the updating after pre-conditioning in

$$\bar{Y}^{(t+1)} = Y^{(t)} - \alpha \tilde{D}^{-1} \nabla_{Y^{(t)}} \bar{\ell}(Y; \psi), \tag{53}$$

where $\tilde{D} = \lambda_0 D_C + \lambda_1 \bar{D}_S + I$.

Do vetorization on both sides turns (53) to :

$$vec(\bar{Y}^{(t+1)}) = vec(Y^{(t)}) - \alpha vec(\tilde{D}^{-1}\nabla_{Y^{(t)}}\bar{\ell}(Y;\psi)) \tag{54}$$
$$= vec(Y^{(t)}) - \alpha\hat{D}^{-1}vec(\nabla_{Y^{(t)}}\bar{\ell}(Y;\psi)), \tag{55}$$

where $\hat{D}^{-1} = I \otimes \tilde{D}^{-1}$. Note that we apply Roth's column lemma to (54) to derive (55).

From the property of strongly convex function $\ell(Y;\psi)$, We know the following inequality holds for any $\bar{Y}^{(t+1)}$ and $Y^{(t)}$:

$$\ell(\bar{Y}^{(t+1)};\psi) \leq \ell(Y^{(t)};\psi) + vec(\nabla_{Y^{(t)}}\bar{\ell}(Y;\psi))^\top vec(\bar{Y}^{(t+1)} - Y^{(t)}) + \frac{1}{2}vec(\bar{Y}^{(t+1)} - Y^{(t)})^\top \nabla^2_{Y^{(t)}}\bar{\ell}(Y;\psi)vec(\bar{Y}^{(t+1)} - Y^{(t)}), \tag{56}$$

where $\nabla^2_{Y^{(t)}}\bar{\ell}(Y;\psi)$ is a Hessian matrix whose elements are $\nabla^2_{Y^{(t)}}\bar{\ell}(Y;\psi)_{ij} = \frac{\partial \ell_Y(Y)}{\partial vec(Y)_i \partial vec(Y)_j}|_{Y=Y^{(t)}}$.

Applying the gradient descent update $vec(\bar{Y}^{(t+1)} - Y^{(t)}) = -\alpha\hat{D}^{-1}vec(\nabla_{Y^{(t)}}\bar{\ell}(Y;\psi))$, we get:

$$\ell(\bar{Y}^{(t+1)};\psi) \leq \ell(Y^{(t)};\psi) - (\hat{D}^{-1}vec(\nabla_{Y^{(t)}}\bar{\ell}(Y;\psi)))^\top(\alpha\hat{D})(\hat{D}^{-1}vec(\nabla_{Y^{(t)}}\bar{\ell}(Y;\psi))) + (\hat{D}^{-1}vec(\nabla_{Y^{(t)}}\bar{\ell}(Y;\psi)))^\top(\frac{\alpha^2}{2}\nabla^2_{Y^{(t)}}\bar{\ell}(Y;\psi))(\hat{D}^{-1}vec(\nabla_{Y^{(t)}}\bar{\ell}(Y;\psi))). \tag{57}$$

If $\alpha\hat{D} - \frac{\alpha^2}{2}\nabla^2_{Y^{(t)}}\bar{\ell}(Y;\psi) \succ 0$ holds, then gradient descent will always decrease the loss, and furthermore, since $\ell(Y;\psi)$ is strongly convex, with proximal descent, it will monotonically decrease the loss until the unique global minimum. To compute $\nabla^2_{Y^{(t)}}\bar{\ell}(Y;\psi)$, we differentiate (52) and arrive at:

$$\nabla^2_{Y^{(t)}}\bar{\ell}(Y;\psi) = 2(\mathbf{I} + Q - P + \mathbf{D}). \tag{58}$$

where $\mathbf{D} = \lambda_0 I \otimes D_C + \lambda_1 I \otimes \bar{A}_S$ and Q and P is in Proposition 5.1. Returning to the above inequality, we can then proceed as follows:

$$\begin{aligned}\alpha\hat{D} - \frac{\alpha^2}{2}(\mathbf{I} + Q - P + \mathbf{D}) &= \alpha(\mathbf{I} + \lambda_0 I \otimes D_C + \lambda_1 I \otimes \bar{D}_S) - \alpha^2(\mathbf{I} + Q - P + \lambda_0 I \otimes D_C + \lambda_1 I \otimes \bar{A}_S) \\ &= (\alpha - \alpha^2)(\mathbf{I} + \lambda_0 I \otimes D_C) + \lambda_1 I \otimes \alpha\bar{D}_S - \alpha^2(Q - P + \lambda_1 I \otimes \bar{A}_S) \\ &\succ [(\alpha - \alpha^2)(1 + \lambda_0 d_{C\min}) + \alpha\lambda_1 d_{S\min}]\mathbf{I} - \alpha^2(Q - P + \lambda_1 I \otimes \bar{A}_S).\end{aligned} \tag{59}$$

If α satisfies $[(\alpha - \alpha^2)(1 + \lambda_0 d_{C\min}) + \alpha\lambda_1 d_{S\min}]\mathbf{I} - \alpha^2(Q - P + \lambda_1 I \otimes \bar{A}_S) \succ 0$, then $\alpha\hat{D} - \frac{\alpha^2}{2}\nabla^2_{Y^{(t)}}\bar{\ell}(Y;\psi) \succ 0$ holds. Therefore, a sufficient condition for convergence to the unique global optimum is:

$$(\alpha - \alpha^2)(1 + \lambda_0 d_{C\min}) + \alpha\lambda_1 d_{S\min} - \alpha^2\sigma_{\max} > 0. \tag{60}$$

where $\sigma_{\max}$ is the max eigenvalue of $(Q - P + \lambda_1 I \otimes \bar{A}_S)$ Consequently, to guarantee the aforementioned convergence we arrive at the final inequality:

$$\alpha < \frac{1 + \lambda_0 d_{C\min} + \lambda_1 d_{S\min}}{1 + \lambda_0 d_{C\min} + \sigma_{\max}}. \tag{61}$$

□

C.3. Proof for Corollary 5.2

This proof is more simpler because without compatibility matrix we don't need vectorization here. For $\psi = \{W, I, I\}$, the gradient becomes

$$\nabla_Y \bar{\ell}(Y;\psi) = 2(\lambda_0 L_C + \lambda_1 \bar{L}_S)Y + 2Y - 2f(X;W), \tag{62}$$

The Hessian matrix is

$$\nabla^2_{Y^{(t)}}\bar{\ell}(Y;\psi) = 2(\lambda_0 L_C + \lambda_1 L_s + I). \tag{63}$$

While all other conditions are similar, we rewrite (59) in

$$\begin{aligned}\alpha\hat{D} - \frac{\alpha^2}{2}\nabla^2_{Y^{(t)}}\bar{\ell}(Y;\psi) &= \alpha(\lambda_0 D_C + \lambda_1 \bar{D}_S + I) - \alpha^2(\lambda_0 L_C + \lambda_1 \bar{L}_S + I) \\ &= (\alpha-\alpha^2)(\lambda_0 D_C + \lambda_1 \bar{D}_S + I) + \alpha^2(\lambda_0 A_C + \lambda_1 \bar{A}_S) \\ &\succ (\alpha-\alpha^2)(\lambda_0 d_{Cmin} + \lambda_1 d_{Smin} + I) + \alpha^2(\lambda_0 A_C + \lambda_1 \bar{A}_S)\end{aligned} \tag{64}$$

To make $(\alpha-\alpha^2)(\lambda_0 d_{Cmin} + \lambda_1 d_{Smin} + I) + \alpha^2(\lambda_0 A_C + \lambda_1 \bar{A}_S) \succ 0$ a sufficient condition is that

$$(\alpha-\alpha^2)(\lambda_0 d_{Cmin} + \lambda_1 d_{Smin} + 1) + \alpha^2\sigma_{min} > 0 \tag{65}$$

where σ_{min} is the min eigenvalue of $(\lambda_0 A_C + \lambda_1 \bar{A}_S)$. That comes to

$$\alpha < \frac{\lambda_0 d_{Cmin} + \lambda_1 d_{Smin} + 1}{\lambda_0 d_{Cmin} + \lambda_1 d_{Smin} + 1 - \sigma_{min}}. \tag{66}$$

D. Hyperparameters

Here we present hyperparameters for reproducing results in Table 1, Table 2 in Table 7 and 8 . And for Table 5 the hyperparameters are in Table 9 and 10. Note that in the ablation for combination coefficients, we re-searched for hyperparameters for each combination.

Table 7. PhenomNN$_{\text{simple}}$ hyperparameters for Table 1 and 2.

Dataset	lr	dropout	hidden	λ_0	λ_1	α	prop step
Coauthorship/Cora	0.01	0.7	64	20	80	0.1	16
Coauthorship/DBLP	0.005	0.6	64	100	100	0.1	16
Cocitation/Cora	0.005	0.7	64	0	20	1	16
Cocitation/PubMed	0.02	0.7	64	0	20	0.1	16
Cocitation/Citeseer	0.005	0.7	64	1	20	1	16
NTU2012	0.001	0.2	128	1	1	0.1	16
ModelNet40	0.0005	0.4	128	1	1	0.05	16
NTU2012*	0.01	0.2	256	50	20	0.05	16
ModelNet40*	0.01	0	512	50	1	0.05	16
Yelp	0.01	0.1	64	1	100	0.1	4
House(1)	0.1	0	512	50	20	$\frac{1}{70}$ or $(\lambda_0+\lambda_1)^{-1}$	16
House(0.6)	0.1	0	512	1	1	0.05	16
Walmart(1)	0.01	0	256	0	50	1	16
Walmart(0.6)	0.1	0	256	1	20	1	16
20Newsgroups	0.01	0.2	64	0.1	0	1	7

Table 8. PhenomNN hyperparameters for Table 1 and 2.

Dataset	lr	dropout	hidden	λ_0	λ_1	α	prop step
Coauthorship/Cora	0.001	0.8	64	20	100	0.1	16
Coauthorship/DBLP	0.001	0.6	64	1	1	1	16
Cocitation/Cora	0.01	0.6	64	0	20	1	16
Cocitation/PubMed	0.01	0.6	64	1	1	1	16
Cocitation/Citeseer	0.001	0.8	64	50	50	0.1	16
NTU2012	0.001	0.2	64	20	80	0.05	16
ModelNet40	0.0005	0.2	64	0	20	0.05	16
NTU2012*	0.01	0.2	256	100	20	0.05	16
ModelNet40*	0.001	0.2	512	0	20	0.05	16
Yelp	0.01	0.2	64	0	1	0.01	4
House(1)	0.01	0.2	64	50	100	0.05	16
House(0.6)	0.01	0.2	512	0	1	0.05	16
Walmart(1)	0.001	0	256	0	50	1	16
Walmart(0.6)	0.01	0	256	0	50	1	16
20Newsgroups	0.01	0	64	0.1	0.1	1	8

Table 9. PhenomNN$_{\text{simple}}$ hyperparameters for combination ablation. For every dataset, the first row is PhenomNN$_{\text{simple}}$-clique, the second is PhenomNN$_{\text{simple}}$-star.

Dataset	lr	dropout	hidden	λ_0	λ_1	α	prop step
Coauthorship/Cora	0.01	0.7	64	20	0	0.1	16
Coauthorship/Cora	0.01	0.7	64	0	50	0.1	16
Coauthorship/DBLP	0.005	0.6	64	20	0	0.1	16
Coauthorship/DBLP	0.01	0.6	64	0	100	0.1	16
Cocitation/Cora	0.005	0.6	64	20	0	1	16
Cocitation/Cora	0.005	0.7	64	0	20	1	16
Cocitation/PubMed	0.1	0.5	64	20	0	1	16
Cocitation/PubMed	0.02	0.7	64	0	20	0.1	16
Cocitation/Citeseer	0.01	0.7	64	20	0	1	16
Cocitation/Citeseer	0.005	0.7	64	0	20	1	16
NTU2012	0.001	0.2	128	20	0	0.05	16
NTU2012	0.001	0.2	128	0	100	0.1	16
ModelNet40	0.0005	0.4	128	20	0	0.05	16
ModelNet40	0.0005	0.4	128	0	20	0.05	16
NTU2012*	0.001	0	256	80	0	0.05	16
NTU2012*	0.01	0	256	0	1	0.1	16
ModelNet40*	0.01	0.2	512	100	0	0.05	16
ModelNet40*	0.01	0	512	0	80	0.05	16
Yelp	0.01	0	64	50	0	0.01	4
Yelp	0.01	0.1	64	0	100	0.1	4
House(1)	0.01	0	512	80	0	0.05	16
House(1)	0.01	0	512	0	1	0.05	16
House(0.6)	0.1	0	512	1	0	0.05	16
House(0.6)	0.1	0	512	0	80	0.05	16
Walmart(1)	0.01	0	256	50	0	1	16
Walmart(1)	0.01	0	256	0	50	1	16
Walmart(0.6)	0.1	0	256	20	0	1	16
Walmart(0.6)	0.01	0	256	0	20	1	16
20Newsgroups	0.01	0.2	64	0.1	0	1	7
20Newsgroups	0.01	0.2	64	0	0.1	1	7

Table 10. PhenomNN hyperparameters for combination ablation. For every dataset, the first row is PhenomNN-clique, the second is PhenomNN-star.

Dataset	lr	dropout	hidden	λ_0	λ_1	α	prop step
Coauthorship/Cora	0.001	0.8	64	1	0	0.1	16
Coauthorship/Cora	0.001	0.8	64	0	80	0.1	16
Coauthorship/DBLP	0.01	0.6	64	1	0	1	16
Coauthorship/DBLP	0.01	0.6	64	0	20	1	16
Cocitation/Cora	0.01	0.6	64	50	0	0.1	16
Cocitation/Cora	0.01	0.6	64	0	20	1	16
Cocitation/PubMed	0.01	0.6	64	1	0	1	16
Cocitation/PubMed	0.001	0.8	64	0	1	1	16
Cocitation/Citeseer	0.001	0.8	64	80	0	0.05	16
Cocitation/Citeseer	0.001	0.8	64	0	20	1	16
NTU2012	0.001	0.2	64	1	0	0.05	16
NTU2012	0.001	0.2	64	0	100	0.1	16
ModelNet40	0.001	0.4	64	1	0	0.05	16
ModelNet40	0.0005	0.2	64	0	20	0.05	16
NTU2012*	0.01	0.2	256	1	0	0.05	16
NTU2012*	0.01	0.2	256	0	20	0.05	16
ModelNet40*	0.001	0.2	512	1	0	0.05	16
ModelNet40*	0.001	0.2	512	0	20	0.05	16
Yelp	0.01	0.2	64	1	0	0.01	4
Yelp	0.01	0.2	64	0	1	0.01	4
House(1)	0.01	0	512	1	0	0.05	16
House(1)	0.01	0.2	64	0	1	1	16
House(0.6)	0.01	0	64	1	0	0.05	16
House(0.6)	0.01	0.2	512	0	1	0.05	16
Walmart(1)	0.01	0	256	80	0	0.05	16
Walmart(1)	0.001	0	256	0	50	1	16
Walmart(0.6)	0.01	0	256	20	0	0.05	16
Walmart(0.6)	0.001	0	256	0	50	1	16
20Newsgroups	0.01	0	64	0.1	0	0.05	8
20Newsgroups	0.01	0	64	0	0.1	1	8