ONE MODEL, THREE LEVELS: MULTI-TASK LEARNING ON HYPERGRAPHS

Anonymous authors
Paper under double-blind review

ABSTRACT

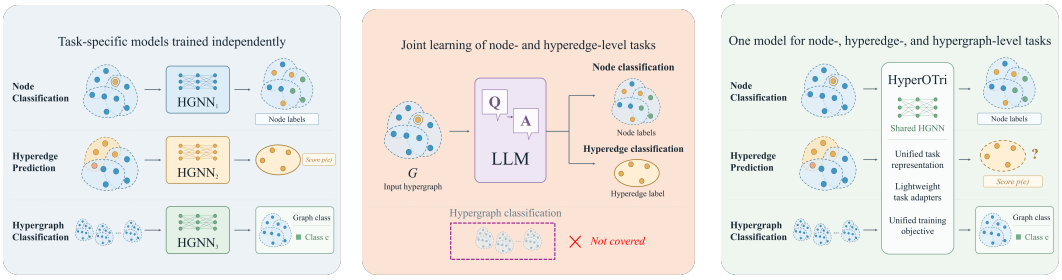
Developing a single model for multiple tasks has long been a goal in artificial intelligence. In hypergraph learning, existing work has explored using a single model to handle node-level and hyperedge-level tasks, but has not incorporated hypergraph-level tasks into the same model. Therefore, we consider the problem of handling node-level, hyperedge-level, and hypergraph-level tasks within a single model, which poses two challenges. First, these tasks involve prediction targets at different structural levels and differ in their input and prediction formats. Second, their optimization objectives may interfere with one another, so improving performance on one task may degrade performance on another. To address these challenges, we propose a model for multi-task learning on hypergraphs. To unify the input and prediction formats across the three tasks, the proposed model uses query-marked subhypergraphs as a common input representation and applies a common query readout followed by prototype matching to produce predictions in a common format. To mitigate cross-task interference, it combines a shared hypergraph encoder with task-conditioned adapters, enabling the model to capture shared higher-order structural patterns while preserving task-specific information. Experimental results demonstrate that the proposed model performs comparably to the baselines across node-, hyperedge-, and hypergraph-level tasks while mitigating cross-task interference. The code is available at: `https://anonymous.4open.science/r/HyperOTri-7EFB`.

1 INTRODUCTION

Hypergraphs allow each hyperedge to connect an arbitrary number of nodes, making them better suited than traditional graphs to modeling real-world higher-order interactions Benson et al. (2018); Bretto (2013). To capture the higher-order structural information in hypergraphs, hypergraph neural networks (HGNNs) encode structure and node attributes into low-dimensional representations through message passing between nodes and hyperedges Feng et al. (2019); Yadati et al. (2019); Dong et al. (2020); Chien et al. (2022). HGNNs have been widely applied to downstream tasks such as node classification Li et al. (2026b), hyperedge prediction Hwang et al. (2022), and hypergraph classification Zhang et al. (2025). Most HGNNs are typically designed and optimized for individual tasks. Moving beyond learning individual tasks in isolation, a long-standing goal in artificial intelligence is to develop a single general-purpose model capable of handling multiple tasks Liu et al. (2024). For hypergraph learning, this goal motivates the development of a single model capable of handling multiple hypergraph learning tasks.

Recent research has begun to explore multitask learning for hypergraphs Cao et al. (2022); Liu et al. (2022). As illustrated in Fig. 1, conventional hypergraph learning typically trains separate task-specific models for node classification, hyperedge prediction, and hypergraph classification (Fig. 1(a)). Hyper-Align Lei et al. (2026) takes a step toward unified hypergraph learning by jointly handling node and hyperedge classification within a shared question-answering framework, but does not cover hypergraph-level tasks (Fig. 1(b)). Although previous studies have explored training a single model to handle node-, edge-, and graph-level tasks on traditional graphs Sun et al. (2023); Jiao et al. (2025), these methods are designed for pairwise relations and cannot be directly applied to hypergraph learning, which involves higher-order relationships. In this work, we aim to jointly handle node-, hyperedge-, and hypergraph-level tasks within a single hypergraph model, as illustrated in Fig. 1(c). This setting raises the following two challenges.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) Conventional single-task learning (b) Hyper-Align (c) HyperOTri

Figure 1: Comparison of hypergraph learning approaches. Conventional single-task learning uses separate HGNNs for individual tasks. Hyper-Align handles node and hyperedge classification tasks within a single model but does not cover hypergraph-level tasks. HyperOTri handles node classification, hyperedge prediction, and hypergraph classification within a single model.

Challenge I: How to formulate different structural-level tasks in a unified manner? Node-, hyperedge-, and hypergraph-level tasks focus on individual nodes, sets of nodes, and entire hypergraphs, respectively. These tasks differ in their inputs, representation granularities, loss, and prediction targets.

Challenge II: How to mitigate interference among different tasks? Different hypergraph tasks place different emphasis on node attributes and higher-order structural information. Improving performance on one task may therefore degrade performance on another Zhang & Yang (2022); Wu et al. (2020); Sener & Koltun (2018).

To address these challenges, we propose HyperOTri (One model for Tri-level tasks on Hypergraphs). HyperOTri unifies task inputs by organizing nodes, hyperedges, and hypergraphs into formatted subhypergraph instances, and unifies prediction targets by identifying the prediction target in each instance with query markers. A common query readout produces target representations of the same form, while label-prototype matching reformulates all three tasks as classification problems sharing a common loss formulation. To address Challenge II, HyperOTri combines a shared hypergraph encoder with task-conditioned adapters. Guided by task identifiers, the adapters transform the shared node representations differently for each task, allowing the model to share encoder parameters while preserving task-specific information, thereby reducing cross-task interference.

The main contributions of this work are summarized as follows: 1) We propose HyperOTri, a hypergraph multi-task learning model that handles tri-level tasks within a single model. 2) We design query-marked subhypergraph representations, a common query readout, and prototype-based prediction to unify the input format, prediction format, and loss formulation across all three tasks. 3) Experiments on seven datasets show that HyperOTri performs comparably to the baselines on node-, hyperedge-, and hypergraph-level tasks, while reducing inter-task interference compared with Naive MTL.

2 PRELIMINARIES

Hypergraphs. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote a hypergraph, where $\mathcal{V} = \{v_1, \ldots, v_N\}$ is the set of N nodes and $\mathcal{E} = \{e_1, \ldots, e_M\}$ is the set of M hyperedges. Each hyperedge $e_j \in \mathcal{E}$ is a non-empty subset of $\mathcal{V}$. The node features are represented by $\mathbf{X} \in \mathbb{R}^{N \times F}$, where F is the feature dimension. The hypergraph structure is represented by an incidence matrix $H \in \{0,1\}^{N \times M}$, where $H_{ij} = 1$ if $v_i \in e_j$ and $H_{ij} = 0$ otherwise. The degree of node v_i and the degree of hyperedge e_j are defined as $d(v_i) = \sum_{j=1}^{M} H_{ij}$ and $\delta(e_j) = \sum_{i=1}^{N} H_{ij}$, respectively. Their corresponding diagonal degree matrices are $D_v = \mathrm{diag}(d(v_1), \ldots, d(v_N))$ and $D_e = \mathrm{diag}(\delta(e_1), \ldots, \delta(e_M))$.

Problem statement. Given a set of hypergraph learning tasks $\mathcal{T} = \{T_m\}_{m=1}^{M}$, each task T_m is associated with a set of hypergraphs $\mathcal{G}_m = \{\mathcal{G}_i^m\}_{i=1}^{n_m}$ in a shared hypergraph space $\mathbb{G}$ and their labels $\mathbf{Y}_m = \{y_i^m\}_{i=1}^{n_m}$ in its own label space $\mathbb{Y}_m$. We denote the union of all label spaces by $\mathbb{Y} = \bigcup_{m=1}^{M} \mathbb{Y}_m$. The goal of multitask hypergraph learning is to learn a unified model $F : \mathbb{G} \to \mathbb{Y}$

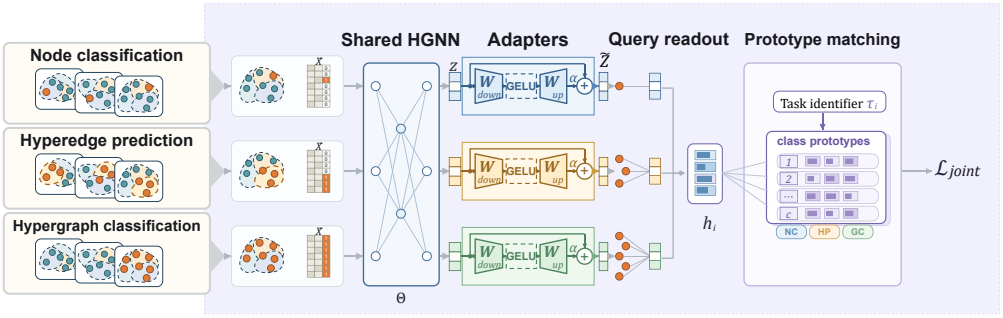


Figure 2: Overview of HyperOTri. It consists of three stages: (1) Unified task input formulation represents each prediction target together with its structural context as a query-marked subhypergraph; (2) Task-aware representation learning uses a shared hypergraph encoder with task-conditioned adapters to learn task-aware node representations; and (3) Unified prediction and joint optimization applies a common query readout followed by prototype matching against task-specific class prototypes and jointly trains the model on mixed-task mini-batches using a common classification loss formulation.

while minimizing the loss function, i.e.,

$$\arg\min_{F} \mathcal{L}\left(\{F(\mathcal{G}_m), \mathbf{Y}_m\}_{m=1}^{M}\right), \tag{1}$$

where $\mathcal{L}$ denotes the multitask loss function.

In this work, we consider three hypergraph learning tasks: node classification(NC), hyperedge prediction(HP), and hypergraph classification(HC), corresponding to node-level, hyperedge-level, and hypergraph-level tasks, respectively.

3 HyperOTri: One model for tri-level tasks on hypergraphs

3.1 Overview

Figure 2 illustrates the overall pipeline of HyperOTri. First, we represent prediction targets in node-, hyperedge-, and hypergraph-level tasks as query-marked subhypergraphs, providing a common input format and explicit identification of prediction targets (Section 3.2). Second, we use a shared hypergraph encoder to capture transferable higher-order structural patterns and task-conditioned adapters to adjust the shared node representations for each task (Section 3.3). Third, we apply a common query readout and a prototype matching mechanism across the three tasks, and jointly optimize the model over mixed-task mini-batches using a common negative log-likelihood objective (Section 3.4).

3.2 Unified Task Input Formulation via Query-Marked Subhypergraphs

To obtain a common input format across structural levels, we represent each prediction target together with its structural context as a subhypergraph, drawing inspiration from All in One Sun et al. (2023). The subhypergraph construction for the three tasks is illustrated in Figure 3. Specifically, for the tasks of node classification and hyperedge prediction, we construct K-hop sub-hypergraphs centered on the target node and the target hyperedge, respectively. For a target node v_i, its K-hop subhypergraph $\mathcal{G}_{\text{sub},v_i}$ consists of all nodes within K hops of v_i and all hyperedges whose nodes lie entirely in the resulting node set. Two nodes are one hop apart if they belong to a common hyperedge. For a target hyperedge e_i, its K-hop subhypergraph $\mathcal{G}_{\text{sub},e_i}$ consists of all hyperedges within K hops of e_i, together with all nodes contained in those hyperedges. Two hyperedges are one hop apart if they share at least one node. To avoid excessive neighborhood expansion, we set the maximum number of sampled hyperedges in each subhypergraph to L. After the aforementioned processing, multiple tasks have the same input format.

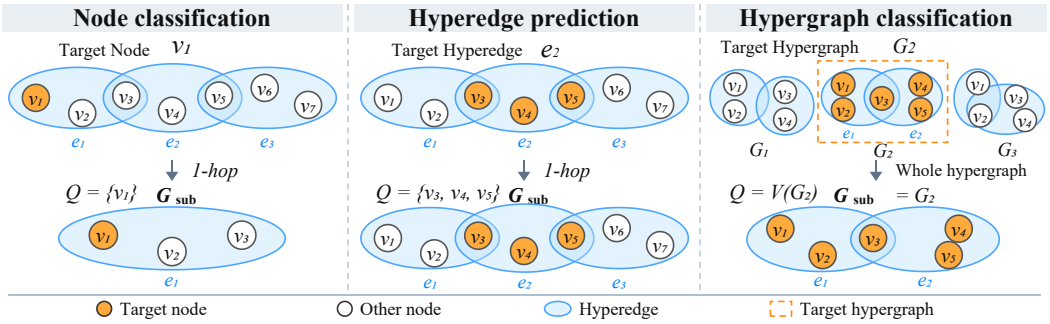


Figure 3: Query-marked subhypergraph construction for the three tasks. For node classification, $\mathcal{Q}$ contains the target node; for hyperedge prediction, it contains the nodes in the hyperedge; and for hypergraph classification, it contains all nodes of the target hypergraph. For hypergraph classification, each hypergraph in the dataset is directly used as an input instance.

To distinguish nodes associated with the prediction target from other nodes in the subhypergraph and to specify the task type, we introduce a binary indicator vector and a task identifier, respectively. We denote the set of nodes associated with the prediction target as the query set $\mathcal{Q}$. Each row is associated with a binary indicator equal to 1 if the corresponding node belongs to $\mathcal{Q}$, and 0 otherwise. For node classification, only the entry corresponding to the target node is set to 1, while all other entries are set to 0. For hyperedge prediction, entries corresponding to all nodes in the target hyperedge are set to 1, while the remaining entries are set to 0. For hypergraph classification, all entries are set to 1. We append this vector to the subhypergraph node feature matrix $\mathbf{X}'$ as an additional column, obtaining the augmented node feature matrix $\widetilde{\mathbf{X}}$. The task identifier τ specifies the task associated with the current sample. The i-th training sample is thus represented in a unified form as

$$\mathcal{D}_i = \left(\mathcal{G}_{\text{sub},i}, \widetilde{\mathbf{X}}_i, \tau_i, y_i\right), \tag{2}$$

where $\mathcal{G}_{\text{sub},i}$ is the subhypergraph associated with the i-th sample, $\widetilde{\mathbf{X}}_i$ is the node feature matrix augmented with the binary indicator column, τ_i is the task identifier, and y_i is the supervision label for the corresponding task. For node classification and hypergraph classification, y_i denotes the class label of the target node and the hypergraph, respectively. For hyperedge prediction, y_i is a binary label indicating whether the hyperedge exists.

3.3 Task-aware representation learning with Adapters

After unifying task inputs, a shared hypergraph encoder performs message passing on subhypergraphs from all three tasks to obtain node representations. Although the tasks share a common input format, their distinct prediction objectives impose different requirements on these representations, potentially leading to task interference during joint learning. To accommodate these differences, we introduce a lightweight adapter for each task. Each adapter applies a residual transformation to the shared node representations, incorporating task-specific information for subsequent query readout.

Hypergraph encoding. Given the unified subhypergraph structure $\mathcal{G}_{\text{sub}} = (\mathcal{V}', \mathcal{E}')$ and its query-augmented node feature matrix $\widetilde{X}$, we feed both into a shared HGNN-style encoder Feng et al. (2019). Each encoding layer sequentially performs node-to-hyperedge and hyperedge-to-node aggregation as follows:

$$\mathbf{Z}^{(l)} = \phi\left(\mathbf{D}_v'^{-\frac{1}{2}}\mathbf{H}'\mathbf{D}_e'^{-1}\mathbf{H}'^{\top}\mathbf{D}_v'^{-\frac{1}{2}}\mathbf{Z}^{(l-1)}\mathbf{\Theta}^{(l)}\right), \tag{3}$$

where $\mathbf{Z}^{(0)} = \widetilde{\mathbf{X}}$ and $\mathbf{Z}^{(l)} \in \mathbb{R}^{|\mathcal{V}'|\times d_l}$ denotes the node representations at layer l. Here, $\mathbf{H}' \in \{0,1\}^{|\mathcal{V}'|\times|\mathcal{E}'|}$ is the incidence matrix of the input subhypergraph, and $\mathbf{D}_v'$ and $\mathbf{D}_e'$ are the corresponding node- and hyperedge-degree matrices computed from $\mathbf{H}'$, respectively. The learnable transformation matrix $\mathbf{\Theta}^{(l)} \in \mathbb{R}^{d_{l-1}\times d_l}$ maps the representations from layer $l-1$ to layer l, and $\phi(\cdot)$ denotes a nonlinear activation function.

Task-conditioned residual adaptation. To accommodate task-specific representation requirements, following Houlsby et al. (2019); Chen et al. (2022), we adopt a lightweight residual adapter for each task. Let $Z = Z^{(L)} \in \mathbb{R}^{|\mathcal{V}'| \times D}$ denote the node representations from the final layer of the shared encoder. The adapter A_τ associated with the task identifier τ projects these representations into a lower-dimensional space, applies a nonlinear activation, and projects them back to the original dimension:

$$A_\tau(Z) = W_{\text{up}}^{(\tau)}\Big(\text{GELU}\Big(W_{\text{down}}^{(\tau)}(Z)\Big)\Big), \tag{4}$$

where $W_{\text{down}}^{(\tau)} : \mathbb{R}^D \to \mathbb{R}^{d_a}$ and $W_{\text{up}}^{(\tau)} : \mathbb{R}^{d_a} \to \mathbb{R}^D$ denote task-specific affine projection layers with bias terms, applied to each node representation. Here, d_a is the bottleneck dimension, satisfying $d_a < D$.

For a task identified by τ, the adapted node representations are obtained by selecting A_τ and adding its output to the shared representations:

$$\widetilde{Z} = Z + \alpha A_\tau(Z), \tag{5}$$

where α controls the strength of the task-conditioned adaptation. The residual connection combines the shared representation Z with the task-specific adjustment $\alpha A_\tau(Z)$.

3.4 Unified prediction and joint optimization

Building on the task-adapted node representations obtained in Section 3.3 and the unified query formulation introduced in Section 3.2, we derive the prediction target representations, compute class prediction probabilities through prototype matching, and formulate the joint training objective.

Under the unified query formulation introduced in Section 3.2, the prediction target representation for each task is obtained by aggregating the node representations within the query set $\mathcal{Q}$. Specifically, for sample i, we aggregate the representations of nodes in its query set $\mathcal{Q}_i$ as follows:

$$h_i = \text{Readout}\left(\{\widetilde{z}_{i,v} : v \in \mathcal{Q}_i\}\right), \tag{6}$$

where $\widetilde{z}_{i,v} \in \mathbb{R}^D$ denotes the task-adapted representation of node v in sample i, and $\text{Readout}(\cdot)$ is a pooling function.

Given the prediction target representation h_i, we use a unified prototype matching mechanism to perform prediction for all three tasks. For task τ_i, the class prediction probability is defined as

$$p(y_i = c \mid h_i, \tau_i) = \frac{\exp\left(\text{sim}(h_i, p_{\tau_i,c})/T\right)}{\sum_{c' \in Y_{\tau_i}} \exp\left(\text{sim}(h_i, p_{\tau_i,c'})/T\right)}, \tag{7}$$

where Y_{τ_i} denotes the class set for task identifier τ_i, $p_{\tau_i,c} \in \mathbb{R}^D$ is the learnable prototype for class c, initialized randomly, $\text{sim}(\cdot,\cdot)$ denotes cosine similarity, and $T > 0$ is a temperature parameter. The model computes the similarity between the prediction target representation and each class prototype, then applies a temperature-scaled softmax function to obtain class prediction probabilities. All three tasks share the same prediction formulation but use their own class sets and corresponding learnable prototypes. Node classification and hypergraph classification use their respective multiclass label sets, whereas hyperedge prediction uses a binary label set consisting of positive and negative classes.

Joint multitask objective. Based on the class probabilities in equation 7, we formulate the joint training objective as

$$\mathcal{L}_{\text{joint}} = -\frac{1}{|\mathcal{T}|}\sum_{\tau \in \mathcal{T}} \frac{1}{|\mathcal{B}_\tau|}\sum_{i \in \mathcal{B}_\tau} \log p(y_i \mid h_i, \tau_i), \tag{8}$$

where $\mathcal{T}$ denotes the set of tasks and $\mathcal{B}_\tau$ denotes the mini-batch sampled from task τ. The loss is first averaged over the samples within each task and then equally averaged across the three tasks. The variables y_i and τ_i denote the label and task identifier of sample i, respectively. During training, this objective jointly updates the shared hypergraph encoder, the task-conditioned adapters, and the class prototypes. The overall training procedure of HyperOTri is provided in Appendix A.

4 EXPERIMENTS

In this section, we evaluate HyperOTri by answering the following questions: **Q1:** Does HyperOTri outperform the baselines across the three tasks? **Q2:** How do different task combinations affect the performance of individual tasks? **Q3:** How does each component of HyperOTri contribute to its overall performance? **Q4:** How sensitive is HyperOTri to its key hyperparameters?

4.1 EXPERIMENTAL SETTINGS

Datasets. We conduct experiments on seven datasets: PubMed Sen et al. (2008), Citeseer Sen et al. (2008), Cora-CA Rossi & Ahmed (2015), NTU2012 Chen et al. (2003), RHG-3 Feng et al. (2024), RHG-10 Feng et al. (2024), and IMDB-Wri-Form Feng et al. (2024). PubMed, Citeseer, Cora-CA, and NTU2012 each contain a single hypergraph: Citeseer and Cora-CA are used for both node classification and hyperedge prediction, PubMed only for node classification, and NTU2012 only for hyperedge prediction. RHG-3, RHG-10, and IMDB-Wri-Form contain multiple hypergraphs and are used for hypergraph classification. Dataset statistics are summarized in Appendix B.1.

Baselines. We compare HyperOTri against five baselines from three categories. STL trains each task independently on hypergraph as a single-task baseline. Naive MTL and Hyper-FM Gao et al. (2026) are multi-task methods on hypergraphs, where Naive MTL trains a shared encoder with a joint objective, and Hyper-FM is a foundation model pretrained on hypergraphs. All in One Sun et al. (2023) and OFA Liu et al. (2024) are multi-task methods originally designed for traditional graphs, so we convert each hypergraph into a pairwise graph via clique expansion. This transformation makes the original hyperedge prediction task no longer directly comparable. We do not report hyperedge prediction results for these baselines. Detailed descriptions and configurations are provided in Appendix B.2.

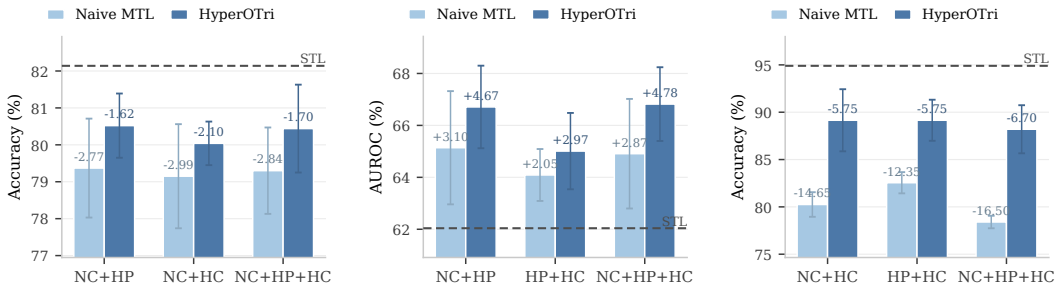
Downstream tasks and metrics. We consider three downstream tasks: node classification, hyperedge prediction, and hypergraph classification, where we use accuracy (Acc) Kipf & Welling (2017) for node and hypergraph classification and AUROC (area under the ROC curve) Fawcett (2006) for hyperedge prediction. The detailed evaluation protocol is provided in Appendix B.3.

4.2 MAIN RESULTS

To evaluate HyperOTri's multitask learning performance, we conduct experiments using three dataset combinations. For each combination, a single model is jointly trained on all three tasks, and each task is evaluated on its corresponding dataset. Specifically, each combination lists the datasets for node classification, hyperedge prediction, and hypergraph classification in that order. The three combinations are (PubMed, Citeseer, RHG-3), (Cora-CA, Cora-CA, RHG-10), and (Citeseer, NTU2012, IMDB-Wri-Form), denoted as Combination 1, Combination 2, and Combination 3, respectively. Table 1 summarizes the results of HyperOTri and baselines. Overall, HyperOTri outperforms the baseline method in the majority of tasks across Combination 1, Combination 2, and Combination 3. HyperOTri matches or outperforms STL in five of the nine cases, indicating positive knowledge transfer across tasks, while the four remaining cases reflect cross-task interference that task-conditioned adaptation mitigates but does not fully eliminate. Prior studies Huang et al. (2023); Navon et al. (2022); Standley et al. (2020); Yang et al. (2025) have reported similar findings, showing that jointly trained models often underperform their independently trained single-task counterparts, with single-task performance commonly regarded as an upper bound in multi-task evaluation. HyperOTri outperforms All in One and OFA by a clear margin. A possible reason is that All in One and OFA are designed for traditional graphs, which model only pairwise relations and therefore cannot fully capture the high-order interactions in hypergraphs. In contrast, HyperOTri directly preserves hyperedge-based interactions and does not require clique expansion to convert a hypergraph into a pairwise graph, which may discard higher-order dependencies. Moreover, HyperOTri outperforms Naive MTL and Hyper-FM. Both baselines lack task-specific adaptation to address potentially conflicting demands on the shared encoder, whereas HyperOTri uses task-conditioned adapters to accommodate such demands, reducing interference during joint training. Hyper-FM performs slightly better than HyperOTri on NTU2012. This can be attributed to Hyper-FM's pretraining stage, whose objective is closely related to hyperedge prediction.

Table 1: Multitask performance comparison across three dataset combinations. Results are reported as mean ± standard deviation (%) over five independent runs. Bold values indicate the best result in each column. "–" denotes unavailable results.

	Combination 1			Combination 2			Combination 3		
Dataset	PubMed	Citeseer	RHG-3	Cora-CA	Cora-CA	RHG-10	Citeseer	NTU2012	IMDB-Wri-Form
Task	Node	Hyperedge	Hypergraph	Node	Hyperedge	Hypergraph	Node	Hyperedge	Hypergraph
Metric	Acc.↑	AUROC↑	Acc.↑	Acc.↑	AUROC↑	Acc.↑	Acc.↑	AUROC↑	Acc.↑
STL	85.28±0.40	**64.68±1.31**	**96.67±1.90**	**82.14±1.28**	62.04±0.85	**94.90±1.19**	69.05±0.85	66.17±1.13	48.95±4.95
Naive MTL	75.61±0.95	59.91±2.86	84.13±1.09	79.08±0.89	63.42±1.75	78.15±0.51	68.60±0.55	62.90±1.17	47.63±4.88
All in One	43.50±0.81	–	71.60±0.88	52.93±0.61	–	32.10±1.08	32.73±0.27	–	43.68±0.53
OFA	81.32±0.22	–	90.00±0.47	68.43±1.45	–	82.20±0.93	47.30±1.06	–	42.89±0.64
Hyper-FM	71.91±1.00	62.41±1.55	89.00±0.21	76.39±0.89	62.82±2.50	82.60±3.76	69.02±1.12	**66.42±2.20**	49.47±3.78
HyperOTri	**85.99±0.34**	62.95±0.88	**96.67±0.82**	80.44±1.19	**66.82±1.42**	88.20±2.54	**69.95±0.77**	65.79±1.71	**50.79±4.53**



(a) Node classification (b) Hyperedge prediction (c) Hypergraph Classification

Figure 4: Effects of joint training on individual task performance. Panels (a), (b), and (c) show performance on node classification (NC), hyperedge prediction (HP), and hypergraph classification (HC), respectively. "+" in the x-axis labels denotes joint training on the listed tasks. The dashed line in each panel shows the corresponding single-task learning (STL) performance. Numbers above the bars indicate changes relative to STL in percentage points; negative values indicate negative transfer.

4.3 Effects of Joint Training on Individual Task Performance

We evaluate how joint training affects individual task performance by comparing HyperOTri with Naive MTL. NC, HP, and HC denote node classification, hyperedge prediction, and hypergraph classification, respectively. Task abbreviations joined by "+" indicate joint training on the corresponding tasks. For each target task, we evaluate both methods under two-task and three-task joint training settings and report only the target task's performance. For example, when NC is the target task, the settings are NC+HP, NC+HC, and NC+HP+HC. Both methods are compared with the same STL baseline for each target task; a performance decrease relative to STL is treated as negative transfer. These experiments use Cora-CA for NC and HP and RHG-10 for HC.

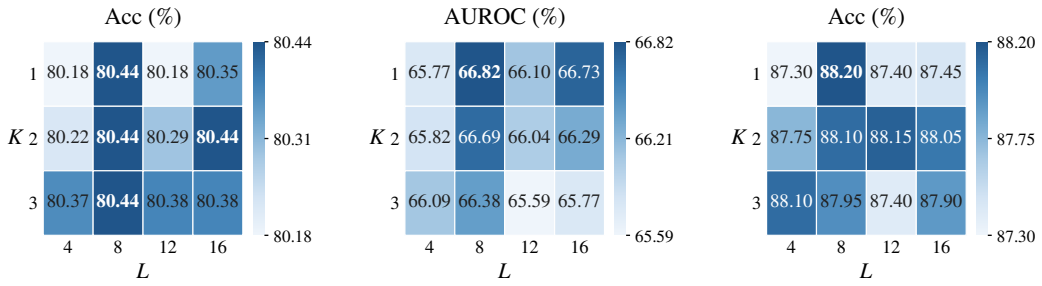
Figure 4 compares the performance changes of each target task under different task combinations relative to the STL baseline. For node classification and hypergraph classification, adding other tasks generally reduces performance relative to STL. This indicates conflicts between these tasks during joint training. In these cases, HyperOTri shows a smaller performance drop than Naive MTL; a possible reason may be task-conditioned adaptation helps reduce such interference. For hyperedge prediction, adding other tasks instead improves performance over STL under several task combinations. This suggests that hyperedge prediction can benefit from complementary structural information learned jointly with the other tasks.

4.4 Ablation Study

We evaluate four variants of HyperOTri: 1) w/o SubHG, which uses the complete hypergraph instead of the K-hop subhypergraph for node classification and hyperedge prediction; 2) w/o Unified Pred, which replaces prototype matching with separate MLP classifiers and task-specific losses; 3) w/o Adapter, which removes the task-conditioned adapters; and 4) Shared Adapter, which replaces the task-conditioned adapters with a single adapter shared across all tasks. Table 2 reports the values for each task.

Table 2: Component ablation of HyperOTri. Node classification and hypergraph classification report accuracy, and hyperedge prediction reports AUROC (mean ± standard deviation, %, over five runs). The best mean in each column is bolded.

	Node Classification			Hyperedge Prediction			Hypergraph Classification		
Variant	PubMed	Citeseer	Cora-CA	Citeseer	Cora-CA	NTU2012	RHG-3	RHG-10	IMDB-W-F
w/o SubHG	49.16 ±0.41	41.75 ±1.66	75.32 ±0.92	59.83 ±1.75	**67.77** ±1.78	63.41 ±1.11	96.44 ±1.40	**88.45** ±2.37	48.68 ±3.90
w/o Unified Pred	85.50 ±0.40	69.50 ±0.43	79.78 ±1.26	57.91 ±1.59	62.10 ±1.74	62.55 ±2.04	89.80 ±2.58	81.75 ±2.38	48.95 ±3.16
w/o Adapter	84.35 ±0.34	69.68 ±1.25	78.71 ±0.82	60.67 ±1.41	65.22 ±1.93	63.34 ±1.01	94.33 ±3.88	79.70 ±1.68	47.37 ±6.28
Shared Adapter	84.52 ±0.18	69.68 ±0.71	79.52 ±1.23	60.01 ±2.29	67.03 ±1.35	63.90 ±1.20	95.27 ±2.83	82.45 ±2.78	46.05 ±6.30
HyperOTri	**85.99** ±0.34	**69.95** ±0.77	**80.44** ±1.19	**62.95** ±0.88	66.82 ±1.42	**65.79** ±1.71	**96.67** ±0.82	88.20 ±2.54	**50.79** ±4.53



(a) Node Classification (b) Hyperedge Prediction (c) Hypergraph Classification

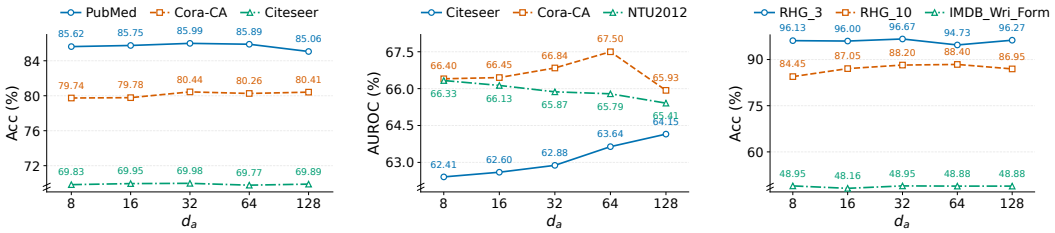
Figure 5: Effect of the subhypergraph construction parameters K and L. Node-level and hypergraph-level results are measured by accuracy, and hyperedge-level results by AUROC.

In general, HyperOTri performs best in most columns of Table 2, suggesting that each component is critical to the overall performance. Compared with HyperOTri, w/o SubHG incurs a performance drop on node classification, indicating the importance of unifying task inputs. In contrast, HyperOTri underperforms in two cases. The first is the hyperedge prediction on Cora-CA; a possible reason is that this task benefits from a broader context than the sampled subgraph preserves. The second is the hypergraph classification on RHG-10; a possible reason is that although this task takes the same input in both methods, the inputs of the other two tasks change during joint training, and their gradients can affect the shared encoder and thereby benefit this task. w/o Unified Pred underperforms HyperOTri across all evaluated tasks, indicating the benefit of the proposed unified prediction formulation for joint multi-task learning. Both w/o Adapter and Shared Adapter result in low performance, indicating that task-conditioned adaptation is necessary to preserve task-specific information and reduce interference among tasks.

4.5 Parameter Sensitivity Analysis

Effect of the subhypergraph construction parameters K and L. We jointly vary the number of hops $K \in \{1, 2, 3\}$ and the maximum number of sampled hyperedges $L \in \{4, 8, 12, 16\}$. Figure 5 reports the results on Cora-CA for node classification and hyperedge prediction and on RHG-10 for hypergraph classification. HyperOTri remains relatively stable on node classification and hyperedge prediction across different choices of K and L, while moderate values of K and L generally yield better performance. Although hypergraph classification does not require subhypergraph construction, its performance is still affected by K and L because changing the inputs of the other two tasks during joint training alters the optimization of the shared encoder. Overall, $(K, L) = (1, 8)$ provides the most favorable performance across the three task levels, and we therefore use it as the default setting.

Effect of the adapter dimension d_a. We vary the dimension of the task-conditioned adapters as $d_a \in \{8, 16, 32, 64, 128\}$. Figure 6 reports the all tasks' results. For node classification, HyperOTri achieves the highest accuracy on all three datasets when $d_a = 32$; the performance variation across different dimensions is relatively small. For hyperedge prediction, the optimal adapter dimension varies across datasets: HyperOTri achieves its best performance at $d_a = 128$ on Citeseer, $d_a = 64$



(a) Node Classification (b) Hyperedge Prediction (c) Hypergraph Classification

Figure 6: Effect of the adapter dimension d_a. Node-level and hypergraph-level results are measured by accuracy, and hyperedge-level results by AUROC.

on Cora-CA, and $d_a = 8$ on NTU2012, with performance on NTU2012 gradually decreasing as d_a increases. For hypergraph classification, HyperOTri achieves its best performance at $d_a = 32$ on RHG-3 and $d_a = 64$ on RHG-10, while its performance on IMDB-Wri-Form remains nearly unchanged across the tested dimensions. Overall, HyperOTri achieves the best node-classification performance at $d_a = 32$, while remaining competitive on hyperedge prediction and hypergraph classification; we therefore use $d_a = 32$ as the default setting.

5 Related Work

Multi-task Learning on Graphs. Multitask learning (MTL) Zhang & Yang (2022); Yue et al. (2023) on graphs aims to exploit similarities between tasks in GNNs to benefit the generalization performance Yue et al. (2023); Liu et al. (2022); Cao et al. (2022); Qin et al. (2023). Existing studies have explored the joint learning of graph tasks at different structural levels. For example, Sun et al. (2023) unify node-level, edge-level, and graph-level tasks through subgraph reformulation and optimize graph prompts via meta-learning. With advances in large language models, researchers have begun incorporating them into graph multitask learning. OFA Liu et al. (2024) unifies diverse graph tasks by grounding representation learning in text-attributed graphs. GIT Wang et al. (2025) introduces task-trees as a unified abstraction for heterogeneous graph tasks, aligning node-level, edge-level, and graph-level problems within a single representation space. These methods address multi-task learning on traditional graphs with pairwise relations, whereas our method jointly learns multiple tasks on hypergraphs over high-order interactions.

Multi-task learning on hypergraphs. Previous studies Liu et al. (2025); Jiao et al. (2024) have leveraged hypergraphs to model high-order relationships and support the joint learning of multiple tasks. Some studies primarily organize multitask learning around different modalities Shao et al. (2020), observation variables Pirvu et al. (2023), semantic attributes Sun et al. (2024), Rather than the joint learning of node classification, hyperedge prediction, and hypergraph classification considered in this work.Hyper-Align Lei et al. (2026) compiles query-object-centered hypergraph context into hypergraph tokens and supports vertex-level and hyperedge-level tasks under a unified question-answering paradigm. Hyper-Align does not integrate hypergraph-level tasks with vertex- and hyperedge-level tasks within a single model, whereas our framework integrates vertex-, hyperedge-, and hypergraph-level tasks within a single model.

6 Conclusion

We introduced HyperOTri, a single model for node classification, hyperedge prediction, and hypergraph classification on hypergraphs. HyperOTri represents prediction targets from these tasks as query-marked subhypergraphs and learns task-aware representations using a shared hypergraph encoder with task-conditioned adapters. A common query readout followed by prototype matching produces class probabilities for each task, enabling joint training with classification losses of the same form. Experiments on seven datasets show that HyperOTri performs comparably to the baselines while reducing cross-task interference compared with Naive MTL.

AI USE STATEMENT

Generative AI tools were used to assist with language editing and reference verification. All AI-assisted suggestions were manually reviewed, and the authors verified the citations, experimental results, and scientific claims. The authors take full responsibility for the final content of this work.

REFERENCES

Austin R. Benson, Rediet Abebe, Michael T. Schaub, Ali Jadbabaie, and Jon Kleinberg. Simplicial closure and higher-order link prediction. *Proceedings of the National Academy of Sciences*, 115 (48):E11221–E11230, 2018.

Alain Bretto. *Hypergraph Theory: An Introduction*. Mathematical Engineering. Springer, Cham, Switzerland, 2013. ISBN 978-3-319-00079-4. doi: 10.1007/978-3-319-00080-0.

Kaidi Cao, Jiaxuan You, and Jure Leskovec. Relational multi-task learning: Modeling relations between data and tasks. In *International Conference on Learning Representations*, 2022.

Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

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

Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. Adaptformer: Adapting vision transformers for scalable visual recognition. In *Advances in Neural Information Processing Systems*, pp. 16664–16678, 2022.

Eli Chien, Chao Pan, Jianhao Peng, and Olgica Milenkovic. You are allset: A multiset function framework for hypergraph neural networks. In *International Conference on Learning Representations*, pp. 1–24, 2022.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

Yihe Dong, Will Sawin, and Yoshua Bengio. Hnhn: Hypergraph networks with hyperedge neurons. *ICML Graph Representation Learning and Beyond Workshop*, pp. 1–11, 2020.

Tom Fawcett. An introduction to roc analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006.

Yifan Feng, Zizhao Zhang, Xibin Zhao, Rongrong Ji, and Yue Gao. 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.

Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph Neural Networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 3558–3565, 2019.

Yifan Feng, Jiashu Han, Shihui Ying, and Yue Gao. Hypergraph isomorphism computation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(5):3880–3896, 2024.

Yue Gao, Yifan Feng, Shiquan Liu, Xiangmin Han, Shaoyi Du, Zongze Wu, and Han Hu. Hypergraph foundation model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 48 (4):4063–4080, 2026.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 2790–2799, 2019.

Zhijian Huang, Sihao Lin, Guiyu Liu, Mukun Luo, Chaoqiang Ye, Hang Xu, Xiaojun Chang, and Xiaodan Liang. FULLER: Unified multi-modality multi-task 3d perception via multi-level gradient calibration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3502–3511, 2023.

Hyunjin Hwang, Seungwoo Lee, Chanyoung Park, and Kijung Shin. Ahp: Learning to negative sample for hyperedge prediction. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 2237–2242, 2022.

Pengfei Jiao, Hongqian Chen, Qing Bao, Wang Zhang, and Huaming Wu. Enhancing multi-scale diffusion prediction via sequential hypergraphs and adversarial learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 8571–8581, 2024.

Pengfei Jiao, Jialong Ni, Di Jin, Xuan Guo, Huan Liu, Hongjiang Chen, and Yanxian Bi. HGMP: Heterogeneous graph multi-task prompt learning. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence*, pp. 2982–2990, 2025.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, pp. 1–14, 2017.

Mengqi Lei, Guohuan Xie, Shihui Ying, Shaoyi Du, Jun-Hai Yong, Chuan Shi, Ling Tian, Siqi Li, and Yue Gao. Hypergraph as language. *arXiv preprint arXiv:2605.21858*, 2026.

Fan Li, Xiaoyang Wang, Wenjie Zhang, Ying Zhang, and Xuemin Lin. Dhg-bench: A comprehensive benchmark for deep hypergraph learning. In *International Conference on Learning Representations*, 2026a.

Ming Li, Yujie Fang, Dongrui Shen, Han Feng, Xiaosheng Zhuang, Kelin Xia, and Pietro Lio. High-pass matters: Theoretical insights and sheaflet-based design for hypergraph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 23039–23046, 2026b.

Ming Li, Huiting Wang, Yuting Chen, Lu Bai, Lixin Cui, Feilong Cao, and Ke Lv. Self-supervised hypergraph learning with substructure awareness for hyperedge prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 23055–23062, 2026c.

Hao Liu, Jiarui Feng, Lecheng Kong, Ningyue Liang, Dacheng Tao, Yixin Chen, and Muhan Zhang. One for all: Towards training one graph model for all classification tasks. In *International Conference on Learning Representations*, pp. 20188–20210, 2024.

Shengchao Liu, Meng Qu, Zuobai Zhang, Huiyu Cai, and Jian Tang. Structured multi-task learning for molecular property prediction. In *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics*, pp. 8906–8920, 2022.

Yanchao Liu, Qing Song, Wenchao Song, Pengzhou Zhang, and Chi Zhang. GTHNN: Graph transformer and sequential hypergraph neural network with dynamic aggregation mechanism for multi-task prediction. *Information Processing & Management*, 62(5):104180, 2025.

Anqi Mao, Mehryar Mohri, and Yutao Zhong. Cross-entropy loss functions: Theoretical analysis and applications. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 23803–23828, 2023.

Aviv Navon, Aviv Shamsian, Idan Achituve, Haggai Maron, Kenji Kawaguchi, Gal Chechik, and Ethan Fetaya. Multi-task learning as a bargaining game. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 16428–16446, 2022.

Mihai Pirvu, Alina Marcu, Maria Alexandra Dobrescu, Ahmed Nabil Belbachir, and Marius Leordeanu. Multi-task hypergraphs for semi-supervised learning using earth observations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pp. 3404–3414, 2023.

Yijian Qin, Xin Wang, Ziwei Zhang, Hong Chen, and Wenwu Zhu. Multi-task graph neural architecture search with task-aware collaboration and curriculum. In *Advances in Neural Information Processing Systems*, pp. 24879–24891, 2023.

Ryan A. Rossi and Nesreen K. Ahmed. The Network Data Repository with Interactive Graph Analytics and Visualization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 4292–4293, 2015.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective Classification in Network Data. *AI Magazine*, 29(3):93–106, 2008.

Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems*, pp. 527–538, 2018.

Wei Shao, Yao Peng, Chen Zu, Mingliang Wang, Daoqiang Zhang, and Alzheimer's Disease Neuroimaging Initiative. Hypergraph based multi-task feature selection for multimodal classification of alzheimer's disease. *Computerized Medical Imaging and Graphics*, 80(10):101663, 2020.

Trevor Standley, Amir Zamir, Dawn Chen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. Which tasks should be learned together in multi-task learning? In *Proceedings of the 37th International Conference on Machine Learning*, pp. 9120–9132, 2020.

Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 945–953, 2015.

Xian Sun, Fanglong Yao, and Chibiao Ding. Modeling high-order relationships: Brain-inspired hypergraph-induced multimodal-multitask framework for semantic comprehension. *IEEE Transactions on Neural Networks and Learning Systems*, 35(9):12142–12156, 2024.

Xiangguo Sun, Hong Cheng, Jia Li, Bo Liu, and Jihong Guan. All in one: Multi-task prompting for graph neural networks. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2120–2131, 2023.

Zehong Wang, Zheyuan Zhang, Tianyi Ma, Nitesh V. Chawla, Chuxu Zhang, and Yanfang Ye. Towards graph foundation models: Learning generalities across graphs via task-trees. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 65518–65555, 2025.

Sen Wu, Hongyang R. Zhang, and Christopher Ré. Understanding and improving information transfer in multi-task learning. In *International Conference on Learning Representations*, 2020.

Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, and Partha Talukdar. Hypergcn: A new method for training graph convolutional networks on hypergraphs. In *Advances in Neural Information Processing Systems*, pp. 1509–1520, 2019.

Yuqi Yang, Peng-Tao Jiang, Qibin Hou, Hao Zhang, Jinwei Chen, and Bo Li. Multi-task dense predictions via unleashing the power of diffusion. In *The Thirteenth International Conference on Learning Representations*, 2025.

Zhixiong Yue, Yu Zhang, and Jie Liang. Learning conflict-noticed architecture for multi-task learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 11078–11086, 2023.

Detian Zhang, Chengqiang Zhang, Yanghui Rao, Li Qing, and Chunjiang Zhu. Improved expressivity of hypergraph neural networks through high-dimensional generalized weisfeiler–leman algorithms. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 76880–76908, 2025.

Yin Zhang, Rong Jin, and Zhi-Hua Zhou. Understanding bag-of-words model: A statistical framework. *International Journal of Machine Learning and Cybernetics*, 1(1):43–52, 2010.

Yu Zhang and Qiang Yang. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 34(12):5586–5609, 2022.

Haihong Zhao, Aochuan Chen, Xiangguo Sun, Hong Cheng, and Jia Li. All in one and one for all: A simple yet effective method towards cross-domain graph pretraining. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 4443–4454, 2024.

Algorithm 1 Training Procedure of HyperOTri

Require: Hypergraph data for tasks $\mathcal{T}$; learning rate η; number of training epochs N
Ensure: Trained HyperOTri model

```
Initialize the shared hypergraph encoder, the task-conditioned adapters {A_τ}_{τ∈T}, and the learnable class prototypes {p_{τ,c}}
Construct the query-marked samples D_i = (G_{sub,i}, X̃_i, τ_i, y_i) and their query sets Q_i
for n = 1, ..., N do
    Sample a minibatch B_τ from each task τ ∈ T, and let B ← ⋃_{τ∈T} B_τ
    for each sample D_i ∈ B do
        Obtain the shared node representations Z_i using equation 3
        Obtain the task-adapted node representations Z̃_i ← Z_i + αA_{τ_i}(Z_i) using equation 5
        Compute the prediction target representation h_i ← Readout({z̃_{i,v} : v ∈ Q_i}) using equation 6
        Compute p(y_i = c | h_i, τ_i), c ∈ Y_{τ_i}, using equation 7
    end for
    Compute L_joint using equation 8
    Update the shared hypergraph encoder, task-conditioned adapters, and class prototypes by backpropagating L_joint with learning rate η
end for
```

APPENDIX

In this appendix, we provide supplementary materials for HyperOTri. Appendix A describes the overall training procedure of the proposed framework. Appendix B provides further details of the experimental setup, including datasets B.1, baselines B.2, implementation details B.4, and evaluation protocols B.3. Appendix C presents additional experimental results and analyses to complement the empirical findings in the main paper.

A OVERALL LEARNING PROCESS

Algorithm 1 presents the training procedure of HyperOTri. We first construct the query-marked samples for node classification, hyperedge prediction, and hypergraph classification, and initialize the shared hypergraph encoder, task-conditioned adapters, and learnable class prototypes. During each training epoch, mini-batches sampled from the three tasks are combined into a mixed-task batch. Each sample is first encoded by the shared hypergraph encoder to obtain shared node representations, which are then refined by the adapter associated with its task type. The adapted representations of the query nodes are aggregated into a prediction target representation and subsequently compared with the corresponding class prototypes to obtain the predictive distribution. The losses of all samples in the mixed-task batch are finally combined according to the joint multi-task objective, and all trainable components are optimized jointly through backpropagation.

B EXPERIMENTAL SETUP

B.1 DATASETS

We use seven datasets following the dataset collection and preprocessing of DHG-Bench Li et al. (2026a). These datasets include co-citation networks Sen et al. (2008)(Citeseer, and PubMed); co-authorship network Rossi & Ahmed (2015)(Cora-CA);computer vision network Chen et al. (2003)(NTU2012); synthetic structural hypergraph collections Feng et al. (2024)(RHG-3 and RHG-10), and a movie-domain hypergraph collection Feng et al. (2024) (IMDB-Wri-Form). Table 3 summarizes their domains, supported tasks, and sizes. For the first four datasets, node and hyperedge counts describe the entire hypergraph; for the hypergraph classification datasets, we report the average numbers of nodes and hyperedges per hypergraph. Detailed descriptions of these datasets are provided below.

Table 3: Statistics of the seven hypergraph datasets. NC, HP, and HC denote node classification, hyperedge prediction, and hypergraph classification, respectively. Node and hyperedge counts are reported for the entire hypergraph in single-hypergraph datasets and averaged per hypergraph otherwise.

Dataset	Domain	Task	#Hypergraphs	#Nodes	#Hyperedges
Citeseer	Citation	NC/HC	1	3,312	1,079
Pubmed	Citation	NC/HC	1	19,717	7,963
NTU2012	Vision	NC/HC	1	2,012	2,012
Cora-CA	Coauthor	NC/HC	1	2,708	1,072
RHG-3	Synthetic	HC	1,500	35.5	17.9
RHG-10	Synthetic	HC	2,000	31.3	29.8
IMDB-Wri-Form	Movie	HC	374	10.1	3.7

Citeseer, PubMed, and Cora-CA. Citeseer and PubMed are co-citation networks in which nodes represent papers and hyperedges connect papers that are cited together. Cora-CA is a co-authorship hypergraph, where nodes represent papers and each hyperedge connects papers co-authored by the same author. Node features are represented using Bag-of-Words (BoW) Zhang et al. (2010), and the labels correspond to paper categories.

NTU2012. NTU2012 is a computer vision dataset containing 2,012 objects from 67 categories. Each object is represented as a node, with visual features extracted using GVCNN Feng et al. (2018) and MVCNN Su et al. (2015). Following the standard HGNN Feng et al. (2019) construction, nearest neighbors under Euclidean distance are grouped to form hyperedges in each feature space. The object categories are used as node labels.

RHG-10 and RHG-3. RHG-10 contains ten distinct synthetic hypergraph structures. RHG-3 is constructed by randomly generating hypergraphs from three representative structural patterns, namely Hyper Pyramid, Hyper Checked Table, and Hyper Wheel, to evaluate the ability of an algorithm to recognize different high-order structures.

IMDB-Wri-Form. IMDB-Wri-Form is derived from IMDB based on co-writer relationships. The suffix "Wri" denotes the writer-based hypergraph construction, whereas "Form" indicates that the classification labels correspond to movie forms, such as animation, documentary, and drama.

B.2 Baseline Details

We provide details of the baselines used in the main experiments below. All baselines are evaluated using the same data splits, evaluation metrics, and evaluation protocol as HyperOTri.

STL. STL trains and tests a model on each task independently: node classification, hyperedge prediction, and hypergraph classification are each trained with a separate HGNN, without sharing parameters or training signals across tasks. It provides the single-task performance of each task, and the gap between STL and a multi-task learning method quantifies the cross-task interaction for that task.

Naive MTL. Naive MTL adopts a hard parameter-sharing architecture Caruana (1997), in which all tasks share a hypergraph encoder, and each task keeps its own prediction head. At each training step, we sample one batch from every task and compute its loss, which is optimized with one backward pass and one update of all model parameters. Node and hypergraph classification are trained with the standard cross-entropy loss Mao et al. (2023), while hyperedge prediction is trained with Binary Cross-Entropy loss.

Hyper-FM Gao et al. (2026). Hyper-FM is a Hypergraph Foundation Model for multi-domain knowledge extraction. It fine-tunes a language model with a self-supervised neighborhood prediction task to extract structure-aware vertex features. It then pretrains an HGNN backbone on a hierarchical multi-domain hypergraph, formed by sampling subhypergraphs from different domain hypergraphs, clustering to construct a macro-hypergraph, and connecting different domain hypergraphs via bond vertices. The pretraining adopts structural losses. Following its pre-training and fine-tuning process, we pretrain the encoder on all datasets used in our experiments. The encoder

is then jointly fine-tuned on the three tasks with a prediction head per task. We evaluate HyperFM under the same protocol as all baselines and adopt the official implementation released by the authors.

All in One Sun et al. (2023). All in One is a multi-task prompting method originally designed for traditional graphs. It unifies the format of graph prompts with language prompts through prompt tokens, token structures, and insertion patterns: the graph prompt is an additional subgraph whose tokens have the same dimension as node representations, and it is inserted into the original graph by cross-links, so that the combined graph can be fed into a pretrained GNN. To adapt to heterogeneous tasks, it first reformulates node-level, edge-level, and graph-level problems into a unified graph-level task using induced subgraphs, and then learns the prompt with meta-learning to obtain a robust initialization across tasks. To compare it on hypergraphs, we only adapt the input data: each hypergraph is converted into a pairwise graph by clique expansion Feng et al. (2019), and the resulting graphs are fed into the official implementation of All in One with its default hyperparameters.

OFA Liu et al. (2024). OFA unifies diverse graph tasks by grounding representation learning in text-attributed graphs. Nodes and edges from different domains are described using natural language and encoded into a shared embedding space, enabling graphs from citation networks, molecules, and knowledge bases to be processed by a common encoder. Task heterogeneity is handled through the notion of nodes of interest, which define task-specific subgraphs and introduce a prompt node that aggregates relevant information via message passing. This design converts node-level, link-level, and graph-level problems into a unified classification interface without task-specific readout heads. OFA encodes both structural context and task semantics directly into the graph, allowing generalization when domains, task definitions, or label spaces change. To compare it on hypergraphs, we only adapt the input data, as for All in One: each hypergraph is converted into a pairwise graph by clique expansion, and the resulting graphs are fed into the official implementation of OFA with its default hyperparameters.

B.3 Evaluation protocol

For node classification and hypergraph classification, we randomly split the nodes (respectively, the hypergraphs) into training, validation, and test sets with a ratio of 60%/20%/20%. In node classification, the hypergraph structure remains fully observed, and supervision is drawn only from the training split, whereas in hypergraph classification, the training, validation, and test hypergraphs are disjoint. We report the mean and standard deviation of the test accuracy Kipf & Welling (2017) over five runs with different random seeds.

For hyperedge prediction, we follow the evaluation protocol of Hwang et al. (2022); Li et al. (2026c). We split the hyperedges into training (60%), validation (20%), and test (20%) sets. The validation and test hyperedges are unobserved and thus are not used for node embedding. We form four test sets, each of which pairs the positive test hyperedges with an equal number of negative hyperedges generated by three sampling schemes: (i) sized negative sampling (SNS), which fills a set with k random nodes; (ii) motif negative sampling (MNS), which fills a set of size k by repeatedly merging it with a random incident edge in the clique expansion; and (iii) clique negative sampling (CNS), which picks a random hyperedge and replaces a random constituent node with a random node adjacent to all other constituent nodes. In all three schemes, k is drawn i.i.d. from the hyperedge size distribution of the input hypergraph. The four test sets correspond to SNS, MNS, CNS, and a mixture of the three (Mixed), respectively, and we report the mean AUROC Fawcett (2006) over the four test sets, averaged over five runs. All metrics are reported such that higher values indicate better performance.

B.4 Implementation details

All experiments are performed on a system equipped with an NVIDIA GeForce RTX 4090 GPU (48GB memory), 128GB of RAM, and an AMD EPYC 7502 CPU, running Ubuntu 22.04. HyperOTri is implemented using PyTorch 2.5.1(CUDA 12.4) and PyTorch Geometric 2.6.1.

HyperOTri is trained with the Adam optimizer Kingma & Ba (2015). Unless otherwise specified, we use the same hyperparameter settings across all experiments, as summarized in Table 4. The three task losses are equally weighted during joint training. The task-conditioned adapters are im-

plemented as residual bottleneck MLPs, and prediction is performed by comparing the query representation with task-specific label prototypes using cosine similarity. Since the datasets have raw node features of different dimensions, we follow Zhao et al. (2024) and apply SVD Deerwester et al. (1990) to project all node features to a common dimension of 128. We further analyze the effects of K, L, and d_a in Section 4.5.

During the subhypergraph construction described in Section 3.2, the nodes in the query set $\mathcal{Q}$ are used as the starting nodes, and the local subhypergraph is expanded for K hops through observed hyperedges incident to the currently reached nodes. At most L observed hyperedges are retained in each subhypergraph. For hyperedge prediction, the candidate hyperedge itself is not included in the subhypergraph used for message passing. For a positive candidate, the observed hyperedge whose node set exactly matches the candidate is removed before the local neighborhood is constructed. For a negative candidate, no such removal is required because the candidate hyperedge does not exist in the observed hypergraph. In both cases, the nodes of the candidate hyperedge form the query set, and their surrounding structure is collected only from the remaining observed hyperedges. This ensures that the existence of the candidate hyperedge is not directly available to the encoder.

Table 4: Hyperparameter settings used in the main experiments.

Hyperparameter	**Value**
Learning rate	10^{-3}
Training epochs	100
Encoder layers	2
Hidden dimension	128
Encoder dropout	0.2
Adapter dimension d_a	32
Adapter dropout	0.1
Residual scale α	0.1
Temperature T	0.5
Context hops K	1
Max sampled hyperedges L	8

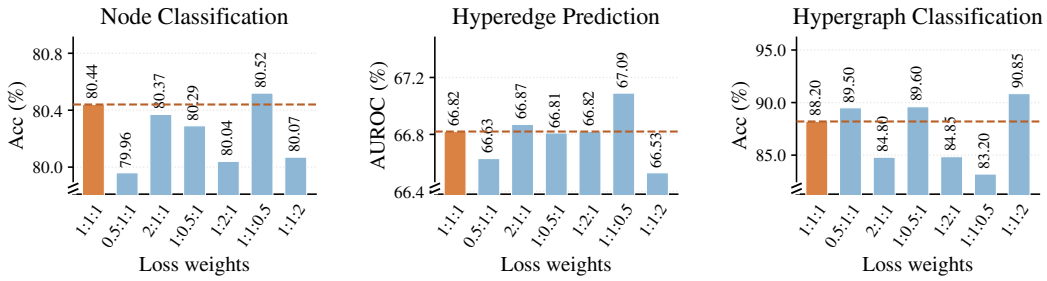
Since the two graph-based baselines, All in One and OFA, cannot be applied to hypergraphs directly, we follow the common practice of clique expansion Feng et al. (2019) to convert each hypergraph into a pairwise graph when adapting them to our setting.

C Additional results and analysis

Analysis of the task-loss weights. We vary the loss weights assigned to node classification, hyperedge prediction, and hypergraph classification to study their effects on joint optimization and the performance balance among tasks. Starting from the equal-weight setting, we set one task's weight to either 0.5 or 2 while keeping the other two at 1. Figure 7 shows that node classification and hyperedge prediction are relatively stable across the tested configurations, with performance ranges of only 0.56 percentage points for both tasks, whereas hypergraph classification is more sensitive to the loss weights. In particular, decreasing the hypergraph-classification loss weight from 1 to 0.5 yields the best node classification accuracy (80.52%) and hyperedge prediction AUROC (67.09%), but reduces hypergraph classification accuracy to 83.20%. Conversely, increasing this weight to 2 improves hypergraph-classification accuracy to 90.85%, while decreasing the other two metrics. Thus, changing the loss weights primarily shifts the performance balance among tasks rather than improving all three simultaneously. The equal-weight setting achieves competitive performance on every task without favoring a particular structural level, supporting its use as a robust default in our main experiments.

Efficiency analysis. Table 5 compares the training efficiency of HyperOTri with STL and Naive MTL under identical settings: the same data splits, the same encoder backbone, and the same hyperparameters. The reported metrics are parameter count, training time, and peak GPU memory. STL replicates the full model for every task, so its parameter count grows linearly with the number of tasks. Naive MTL shares one encoder but relies on plain task-specific heads. HyperOTri keeps a single shared hypergraph encoder and adds only the task-conditioned adapters and a unified prediction module, which use about 60% of the parameters of STL while providing task-specific parameters that a shared encoder alone cannot offer. HyperOTri incurs comparable training time to Naive MTL and similar peak GPU memory usage to both baselines across all three experiments. Overall, HyperOTri attains the best trade-off between parameter cost and task-specific modeling capacity among the three methods.

Visualization of representations. In this section, we provide a visualization of the task-level representations learned by HyperOTri. Concretely, we project the query representations of all nine dataset–task combinations to two dimensions using t-SNE, as shown in Figure 8. We can see that the



(a) Cora-CA (b) Cora-CA (c) RHG-10

Figure 7: Sensitivity of HyperOTri to the task loss weights. The x-axis labels give the relative weights of the node-classification, hyperedge-prediction, and hypergraph-classification losses; 1:1:1 is the default setting (orange). (a) Node classification and (b) hyperedge prediction (AUROC) are evaluated on Cora-CA, and (c) hypergraph classification on RHG-10. Bars show the mean over five runs.

Method	Params. (K)	Time (s)	Mem. (MB)
PubMed:NC / Citeseer:HP / RHG-3:HC			
STL	100.0	80.3	518.2
Naive MTL	33.9	69.3	518.1
HyperOTri	59.9	70.8	518.4
Cora-CA:NC / Cora-CA:HP / RHG-10:HC			
STL	101.4	20.6	275.7
Naive MTL	35.3	20.3	275.4
HyperOTri	61.3	19.0	275.8
Citeseer:NC / NTU2012:HP / IMDB-Wri-Form:HC			
STL	100.5	19.5	52.1
Naive MTL	34.4	18.0	52.1
HyperOTri	60.4	18.5	52.4

Table 5: Efficiency comparison on the three settings, averaged over five runs. For STL, which trains three independent models, the time is the sum of the three runs and the memory is the maximum over the three runs; for the multi-task methods, both are measured for the single joint training run.

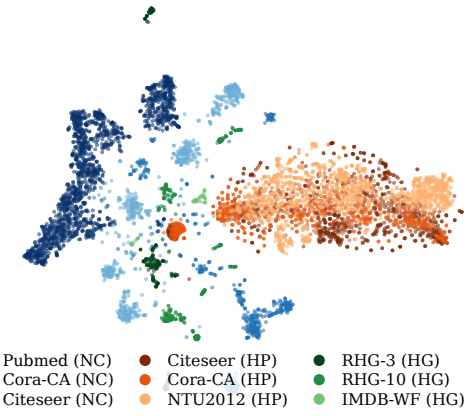


Figure 8: Visualization of representations. NC: node classification; HP: hyperedge prediction; HG: hypergraph classification.

representations are organized by task: node-level, hyperedge-level, and hypergraph-level queries occupy distinct regions of the space, and each dataset–task combination forms a compact local cluster. Moreover, representations of the same task from different datasets are close to each other; for hyperedge prediction, the embeddings of Citeseer and NTU2012 nearly overlap. This result reveals one of the key rationales behind HyperOTri: the shared encoder captures task-level commonality across datasets, while task-conditioned residual adaptation maps different tasks into their own subspaces, allowing one model to serve all tasks without interference.