HYPER: A FOUNDATION MODEL FOR INDUCTIVE LINK PREDICTION WITH KNOWLEDGE HYPERGRAPHS

Anonymous authors
Paper under double-blind review

ABSTRACT

Inductive link prediction with knowledge hypergraphs is the task of predicting missing hyperedges involving completely *novel entities* (i.e., nodes unseen during training). Existing methods for inductive link prediction with knowledge hypergraphs assume a fixed relational vocabulary and, as a result, cannot generalize to knowledge hypergraphs with *novel relation types* (i.e., relations unseen during training). Inspired by knowledge graph foundation models, we propose HYPER as a foundation model for link prediction, which can generalize to *any knowledge hypergraph*, including novel entities and novel relations. Importantly, HYPER can learn and transfer across different relation types of *varying arities*, by encoding the entities of each hyperedge along with their respective positions in the hyperedge. To evaluate HYPER, we construct 16 new inductive datasets from existing knowledge hypergraphs, covering a diverse range of relation types of varying arities. Empirically, HYPER consistently outperforms all existing methods in both node-only and node-and-relation inductive settings, showing strong generalization to unseen, higher-arity relational structures.

1 INTRODUCTION

Generalizing knowledge graphs with relations between *any* number of nodes, knowledge hypergraphs offer flexible means of storing, processing, and managing *relational data*. Knowledge hypergraphs can encode rich relationships between entities; e.g., consider a relationship between *four* entities: "Bengio has a research project on topic ClimateAI in Montreal funded by CIFAR". This relational information can be represented in a knowledge hypergraph (see Figure 1) via an (ordered) hyperedge $\mathsf{Research}(\mathsf{Bengio}, \mathsf{ClimateAI}, \mathsf{Montreal}, \mathsf{CIFAR})$, where Research represents a relation of arity *four*.

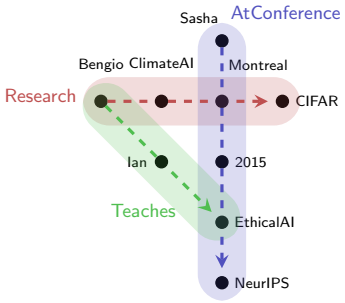

Figure 1: A knowledge hypergraph with three hyperedges over distinct relation types.

The generality knowledge hypergraphs motivated a body of work for machine learning with knowledge hypergraphs (Guan et al., 2021; Fatemi et al., 2020; Yadati, 2020; Zhou et al., 2023b; Huang et al., 2025b). One of the most prominent learning tasks is inductive link prediction with knowledge hypergraphs, where the goal is to predict missing hyperedges involving completely *novel entities* (Yadati, 2020; Zhou et al., 2023b; Huang et al., 2025b). The main shortcoming of existing methods for inductive link prediction with knowledge hypergraphs is that they cannot generalize to knowledge hypergraphs with *novel relation types*. This constitutes the main motivation of our work: *Can we design an effective model architecture for inductive link prediction with knowledge hypergraphs, where the predictions can involve both novel entities and novel relations?*

Example. Consider the knowledge hypergraphs depicted in Figure 2: The training hypergraph G_{train} is over the relations Research, Teaches, and AtConference, while the inference graph G_{inf} is over the novel relations Trading, Sells, and AtFair. The task is to predict missing links such as $\mathsf{Sells}(\mathsf{Samsung}, \mathsf{Best\ Buy}, \mathsf{Q60D\ TV})$ in G_{inf}. Ideally, the model should learn relation invariants that map Teaches $\mapsto$ Sells, Research $\mapsto$ Trading, and AtConference $\mapsto$ AtFair, as these relation types

play analogous structural roles in their respective graphs, even though their labels and entities are entirely different.

Approach. In essence, our study builds on the success of knowledge graph foundation models (KGFMs) (Galkin et al., 2024; Mao et al., 2024), which have shown remarkable performance in link prediction tasks involving both novel entities and novel relations. However, KGFMs can only perform link prediction using *binary relations*, which raises the question of how to translate the success of KGFMs to fully relational data. To this end, we propose HYPER, a class of knowledge hypergraph foundation models for inductive link prediction, which can generalize to any knowledge hypergraph. The fundamental idea behind our approach is to learn properties of relations that are transferable between different types of relations of varying arity. Consider, for example, the two hyperedges (from Figure 2):

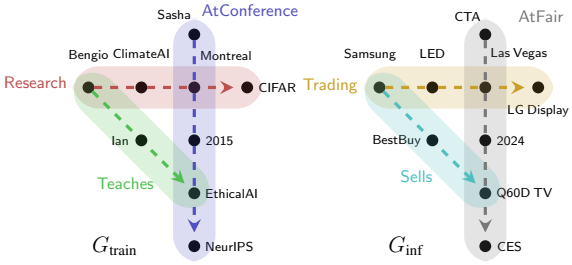

Figure 2: A model is trained on relations like Research, Teaches, and AtConference, and is expected to generalize to structurally similar relations TradingDeal, Sells, and AtBusinessFair at test time.

$$\text{AtConference}(\text{Sasha}, \underline{\text{Montreal}}, 2015, \text{EthicalAI}, \text{NeurIPS}),$$
$$\text{Research}(\text{Bengio}, \text{ClimateAI}, \underline{\text{Montreal}}, \text{CIFAR}),$$

which "intersect" with each other. The entity Montreal appears in the *second* position of the first hyperedge and in the *third* position of the second hyperedge. Such (pairwise) interactions between relations can be viewed as *fundamental relations* to learn from: any model learning from relations between relations can transfer this knowledge to novel relation types that have similar interactions.

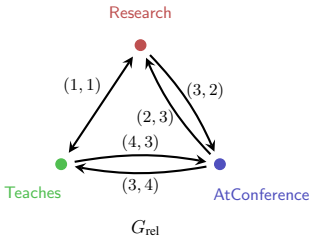

Figure 3: The relation graph G_{rel} corresponding to the knowledge hypergraph G_{train}.

Furthermore, we can encode such relations between relations in a separate relation graph, which can be used to learn from. We illustrate this on our running example in Figure 3, where the relations appear as nodes; the interactions between relations as edges; and finally, the positions of the interactions as edge weights. In our setting, a directed edge from relation r_1 to r_2 with edge label (i, j) indicates that *"The i-th position of r_1 and the j-th position of r_2 intersect in G"*, which captures a fundamental interaction between r_1 and r_2. Critically, however, there is no upper bound on the number of such possible interactions. While there are at most $m \times n$ interactions between an m-ary relation and an n-ary relation, we cannot impose any bound on the arity of the relations since then the model would not generalize to all knowledge hypergraphs.

Contributions. Our main contributions can be summarized as follows:

- To the best of our knowledge, HYPER is the first foundation model that allows zero-shot generalization to knowledge hypergraphs of *arbitrary* arity with *novel nodes* and *novel relations* at test time.
- We evaluate HYPER on 3 existing benchmark datasets and additionally on 16 new benchmark datasets with varying proportions of test-time tuples involving unseen relations. HYPER consistently outperforms existing hypergraph baselines trained end-to-end, particularly when the proportion of new relations is high.
- To assess the performance of KGFMs on hypergraphs, we reify the knowledge hypergraphs into KGs and apply KGFMs on them. Remarkably, HYPER, trained on only 2 hypergraphs and 3KGs, consistently outperforms the popular KGFM model ULTRA trained on 50 KGs.
- We conduct an empirical investigation over the positional interaction encoding scheme within HYPER, demonstrating the critical role of encoding choices.

2 Related Work

Link Prediction with Knowledge Graphs. Link prediction in knowledge graphs (KGs) has been extensively explored. Early knowledge graph embedding methods (Bordes et al., 2013; Sun et al., 2019; Trouillon et al., 2016; Balazevic et al., 2019; Abboud et al., 2020) are limited to the *transductive* setup: these methods do not generalize to unseen entities or to unseen relations. Multi-relational graph neural networks (GNNs) such as RGCN (Schlichtkrull et al., 2018) and CompGCN (Vashishth et al., 2020) similarly rely on stored entity embeddings, remaining inherently transductive. To overcome these limitations, Teru et al. (2020) introduced GraIL, a pioneering method enabling *node-inductive* link prediction, which is later shown to be a form of the labeling trick (Zhang et al., 2021). Subsequently, architectures such as NBFNet (Zhu et al., 2021), A*Net (Zhu et al., 2023), RED-GNN (Zhang & Yao, 2022), and AdaProp (Zhang et al., 2023) leveraged conditional message passing, significantly enhancing expressivity and performance (Huang et al., 2023). However, these methods are *not* inductive on relations, as they assume a fixed relational vocabulary. KGFMs are specifically tailored for inductive predictions on both unseen nodes and relations. InGram (Lee et al., 2023) and ULTRA (Galkin et al., 2024) introduced new KGFM frameworks. Following these, TRIX (Zhang et al., 2024) introduced recursive updating of entity and relation embeddings with provably improved expressiveness over ULTRA. KG-ICL (Cui et al., 2024) employed in-context learning with unified tokenization for entities and relations. Additionally, double-equivariant GNNs, like ISDEA (Gao et al., 2023) and MTDEA (Zhou et al., 2023a), emphasized relational equivariance, enhancing robustness to unseen relations. Huang et al. (2025a) proposed MOTIF as a general KGFM framework and formally studied the expressive power of KGFMs. All of these methods are confined to KGs with binary relations, and they do not naturally apply to higher-arity relations, as shown in Table 1.

Table 1: Methods' ability to handle high-arity relations (**High-arity**) and inductively generalize to unseen entities (**Ind.** e) and relations (**Ind.** r).

Methods	High-arity	Ind. e	Ind. r
HypE, BoxE	✓	✗	✗
NBFNet, A*Net	✗	✓	✗
G-MPNN, HCNet	✓	✓	✗
ULTRA, KG-ICL	✗	✓	✓
HYPER	✓	✓	✓

Link Prediction with Knowledge Hypergraphs. Knowledge hypergraphs generalize traditional KGs to handle higher-arity relational data. Initial researches such as HypE (Fatemi et al., 2020) and BoxE (Abboud et al., 2020) leveraged shallow embedding models adapted from KG embedding frameworks. Later approaches extended graph neural networks to knowledge hypergraphs. G-MPNN (Yadati, 2020) and RD-MPNNs (Zhou et al., 2023b) introduced relational message passing mechanisms explicitly designed for hypergraph settings, incorporating positional entity information critical for high-arity relations. Huang et al. (2025b) proposed HCNets as a conditional message-passing approach tailored for inductive hypergraph link prediction and conducted an expressivity analysis. While these methods can handle knowledge hypergraphs, they are *not* inductive on relations: none of these methods can generalize to unseen relations (shown in Table 1). Our work on HYPER builds on these foundations by combining the strengths of conditional message passing on knowledge hypergraphs with the powerful inductive generalization techniques explored in recent KGFMs (Galkin et al., 2024; Lee et al., 2023; Huang et al., 2025a) to effectively generalize to knowledge hypergraphs within unseen nodes and relations.

Foundation Models on Hypergraphs. Existing foundation models on hypergraphs are tailored to text-attributed hypergraphs. HyperBERT (Bazaga et al., 2024) integrates pretrained language models with hypergraph convolution for node classification, while HyperGene (Du et al., 2021) and SPHH (Abubaker et al., 2023) propose self-supervised objectives tailored to local and global hypergraph structures. More recent works such as Hyper-FM (Feng et al., 2025) and IHP (Yang et al., 2024) introduce multi-domain pretraining and instruction-guided adaptation, respectively, marking the first steps toward generalizable hypergraph models. These methods rely heavily on text attributes for generalization and are predominantly tailored to node classification tasks; they do not support link prediction over knowledge hypergraphs with unseen relations at test time.

3 Preliminaries

Knowledge Hypergraphs. A *knowledge hypergraph* $G = (V, E, R)$ consists of a set of nodes V, hyperedges E (i.e., facts) of the form $e = r(u_1, \ldots, u_k)$, where $r \in R$ is a relation type, and $u_i \in V$,

$1 \leq i \leq k$, are nodes. The arity of a relation r is given by $k = \texttt{ar}(r)$, where $\texttt{ar} : R \mapsto \mathbb{N}_{>0}$. For an hyperedge e, $\rho(e)$ denotes its relation, and $e(i)$ denotes the node at the i-th position of e. We refer to the knowledge hypergraph with all edges having arity of exactly 2 as a *knowledge graph*. The set of edge-position pairs associated with a node v is defined as:

$$E(v) = \{(e, i) \mid e(i) = v, e \in E, 1 \leq i \leq \texttt{ar}(\rho(e))\}.$$

The positional neighborhood of a hyperedge e with respect to a position i is:

$$\mathcal{N}_i(e) = \{(e(j), j) \mid j \neq i, 1 \leq j \leq \texttt{ar}(\rho(e))\}.$$

Link Prediction on Hyperedges. Given a knowledge hypergraph $G = (V, E, R)$ and a query $q(u_1, \ldots, u_{t-1}, ?, u_{t+1} \ldots, u_k)$, the *link prediction* task involves scoring all possible hyperedges formed by replacing the placeholder '?' with each node $v \in V$. We denote a k-tuple of nodes by $\mathbf{u} = (u_1, \ldots, u_k)$ and the tuple excluding position t by $\tilde{\mathbf{u}} = (u_1, \ldots, u_{t-1}, u_{t+1}, \ldots, u_k)$. Thus, we represent a query succinctly as $\mathbf{q} = (q, \tilde{\mathbf{u}}, t)$. In the *fully-inductive setting* for link prediction (i.e., node and relation-inductive link prediction), the goal is to answer queries of the form $\mathbf{q} = (q, \tilde{\mathbf{u}}, t)$ on an inference hypergraph $G_{\text{inf}} = (V_{\text{inf}}, E_{\text{inf}}, R_{\text{inf}})$, where both the entity set V_{inf} and the relation set R_{inf} are entirely disjoint from those seen during training. The model is trained on a separate training knowledge hypergraph $G_{\text{train}} = (V_{\text{train}}, E_{\text{train}}, R_{\text{train}})$, with $V_{\text{train}} \cap V_{\text{inf}} = \emptyset$ and $R_{\text{train}} \cap R_{\text{inf}} = \emptyset$, and must learn transferable representations that generalize across both novel entities and unseen relation types of arbitrary arity. At inference time, each hyperedge $e = r(u_1, \ldots, u_k) \in E_{\text{inf}}$ corresponds to a fact involving a relation $r \in R_{\text{inf}}$, and queries involve predicting a missing node at position t within such a tuple, using the surrounding nodes $\tilde{\mathbf{u}}$ and relation $q = \rho(e)$. The model must score candidate completions $q(u_1, \ldots, u_{t-1}, v, u_{t+1}, \ldots, u_k)$ for each $v \in V_{\text{inf}}$.

Reification. To apply the models designed for KGs on knowledge hypergraphs, we transform an input knowledge hypergraph $G = (V, E, R)$ into a KG via a *reification* process, similar to the one proposed in Fatemi et al. (2020). Specifically, for each hyperedge $r(u_1, \ldots, u_k) \in E$, we introduce a node $\texttt{edge\_id} \notin V$ in the KG to represent the hyperedge itself. We then generate binary edges of the form $\texttt{hasEntity}_i(\texttt{edge\_id}, u_i)$ for each $i \in [k]$ to capture the positions of the entities in the relation. Finally, we add the original relation r as a node to the KG and add an edge $\texttt{hasRelationType}(\texttt{edge\_id}, r)$. For instance, Figure 4 shows the reified KG of our running example in Figure 1. This reification procedure encodes the full higher-order structure of the original knowledge hypergraph into a KG.

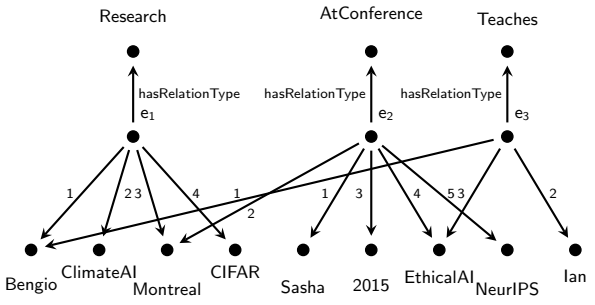

Figure 4: Reified KG corresponding to the knowledge hypergraph G_{train} from Fig 1, where i abbreviates $\mathsf{hasEntity}_i$.

Link Prediction over Reified Knowledge Hypergraphs. Given a high-arity query of the form $q(u_1, \ldots, u_{t-1}, ?, u_{t+1}, \ldots, u_k)$ over the original knowledge hypergraph, we perform link prediction in the reified KG by encoding the query as a subgraph which is used to augment the testing knowledge graph. Concretely, we add a new node $\texttt{edge\_id}$ and binary triples $\texttt{hasEntity}_i(\texttt{edge\_id}, u_i)$ for all $i \neq t$, as well as a triple $\texttt{hasRelationType}(\texttt{edge\_id}, q)$. The prediction task is then reduced to a standard tail prediction problem: ranking all candidate entities $v \in V$ for the fact $\texttt{hasEntity}_t(\texttt{edge\_id}, v)$. We evaluate the model performance using standard ranking metrics over the original entity vocabulary. We use superscript (†) to denote models evaluated under this regime.

4 HYPER: A KNOWLEDGE HYPERGRAPH FOUNDATION MODEL

We now present HYPER, a general framework for learning foundation models over knowledge hypergraphs. Given a knowledge hypergraph $G = (V, E, R)$ and a query $\boldsymbol{q} = (q, \tilde{\boldsymbol{u}}, t)$, HYPER computes link prediction scores through the following steps:

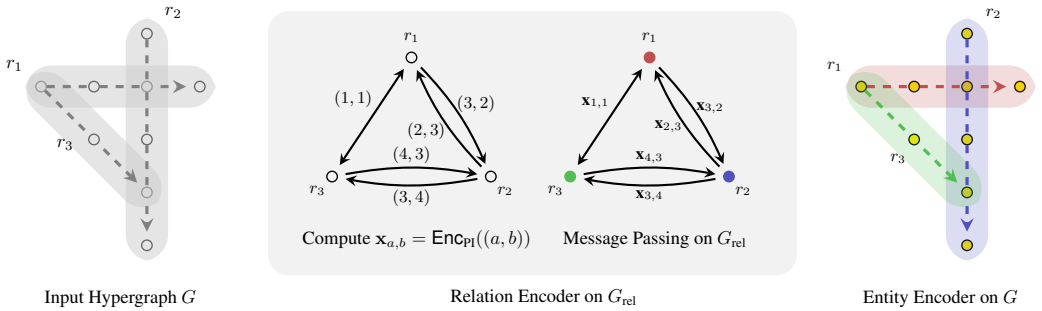

Figure 5: Overall framework of HYPER. HYPER first constructs a relation graph G_{rel} based on the observed positional interactions between the relations. $\mathsf{Enc}_{\mathrm{PI}}$ then computes embeddings for each position pair, which are refined via message passing over G_{rel}. The resulting relation representations are then used for message passing over the original knowledge hypergraph G (shown in color).

1. **Relation encoder:** Relations are encoded in three steps:
   (a) **Relation graph:** Build a relation graph G_{rel} where each node corresponds to a relation $r \in R$, and edges capture observed positional interactions between relations.
   (b) **Encoding positional interactions:** Use an encoder $\mathsf{Enc}_{\mathrm{PI}}$ to embed each interacting position pair (a, b) from G_{rel} into fundamental relation representations.
   (c) **Encoding the relations:** Perform conditional message passing over G_{rel} using fundamental relation representations to obtain relation embeddings for all $r \in R$.
2. **Entity encoder:** Use learned relation representations to conduct conditional message passing over the original knowledge hypergraph G and obtain link probability via decoder Dec.

The overall framework is illustrated in Figure 5, and the detailed architecture is presented in Appendix F. We now describe each component.

Relation graph. Given a knowledge hypergraph $G = (V, E, R)$, we construct the relation graph $G_{\mathrm{rel}} = (V_{\mathrm{rel}}, E_{\mathrm{rel}}, R_{\mathrm{rel}})$. The set of nodes is given as $V_{\mathrm{rel}} = R$, i.e., each node in G_{rel} corresponds to a relation type in the knowledge hypergraph G. The relation types R_{rel} are defined as all ordered pairs (a, b) for $\{1 \leq a, b \leq k_{\max}\}$, where $k_{\max} = \max\{\mathtt{ar}(r) \mid r \in R\}$ denotes the maximum arity among the observed relations. The edge set E_{rel} captures *positional interactions* between relation types: for each pair of hyperedges $e_1, e_2 \in E$ with relation types $r_1 = \rho(e_1)$ and $r_2 = \rho(e_2)$, if there exists a shared entity v appearing in position i in e_1 and position j in e_2, we add a directed edge (r_1, r_2) with relation type (i, j) to E_{rel}. These positional interactions can be computed efficiently via sparse matrix multiplication (see Appendix B) and are invariant over the renaming of relations.

Encoding positional interactions. Unlike knowledge graphs, where each fact involves two entities and naturally leads to four types of fundamental relations (head-to-head, head-to-tail, tail-to-head, and tail-to-tail) as introduced in Galkin et al. (2024), knowledge hypergraphs allow facts with arbitrary arity. This introduces a key challenge: *How to build a foundation model that can adapt to unseen knowledge hypergraphs with varying and arbitrarily large arities?*

The natural extension of the concept of fundamental relations from KGs to knowledge hypergraphs results in mn types of positional interactions between an hyperedge of arity m and an hyperedge of arity n. Each of such interaction is characterized by a pair (a, b), where a and b denote the entity positions involved in the relation. As a consequence, a foundation model for knowledge hypergraphs must be capable of encoding positional interactions in a way that generalizes across different arities.

A naive solution would be to associate a separate embedding to each (a, b) pair. However, such an approach does not generalize to unseen arities, as it would require pre-training embeddings for all possible (a, b) combinations. To address this, we propose a *shared, compositional position interaction encoding scheme*. Specifically, given a positional interaction labeled (a, b), we define a positional interaction encoder $\mathsf{Enc}_{\mathrm{PI}} : \mathbb{N}_{>0} \times \mathbb{N}_{>0} \to \mathbb{R}^d$, which maps a pair of argument positions to a dense vector representation of d dimensions. To be effective in inductive settings, we require the encoder $\mathsf{Enc}_{\mathrm{PI}}$ to satisfy the following requirements:

1. **Extrapolation.** The encoder should generalize to unseen positions and combinations, allowing the model to operate on arities and interaction patterns not present during training.
2. **Injectivity.** Distinct position pairs (a, b) and (a', b') should map to distinct embeddings to preserve the identifiability of positional interactions:

$$\forall a, b, a', b' \in \mathbb{N}_{>0}, (a, b) \neq (a', b') \implies \mathsf{Enc}_{\text{PI}}((a, b)) \neq \mathsf{Enc}_{\text{PI}}((a', b')).$$

In practice, we implement Enc_{PI} as a two-layer multilayer perceptron (MLP) over concatenated sinusoidal encodings of the input positions. Let $\boldsymbol{p}_a, \boldsymbol{p}_b \in \mathbb{R}^d$ denote the sinusoidal positional encodings of positions a and b, respectively. Then, the embedding corresponding to the interaction (a, b) is computed as $\mathbf{x}_{a,b} = \text{MLP}([\boldsymbol{p}_a \,\|\, \boldsymbol{p}_b])$, where MLP denotes a shared two-layer feedforward network with ReLU activations. This produces a dense embedding that captures the interaction between the two positions. Empirically, we find that this instantiation of Enc_{PI} enables strong generalization across knowledge hypergraphs with varying arities and relational structures. When applied to knowledge graphs, our method recovers standard encoding patterns employed in many KGFMs (Galkin et al., 2024; Lee et al., 2023; Zhang et al., 2024; Huang et al., 2025a). In particular, head-to-tail, head-to-head, tail-to-tail, and tail-to-head interactions correspond to $\mathsf{Enc}_{\text{PI}}((1, 2))$, $\mathsf{Enc}_{\text{PI}}((1, 1))$, $\mathsf{Enc}_{\text{PI}}((2, 2))$, and $\mathsf{Enc}_{\text{PI}}((2, 1))$, respectively.

Encoding the relations. HYPER uses *Hypergraph Conditional Networks* (HCNets) (Huang et al., 2025b) to encode relations for its strong inductive performance, support for bidirectional message passing, and easy extensibility to higher-order relational patterns (Huang et al., 2025a). HCNets produce query-conditioned representations by aggregating messages from neighboring edges with relation and position information. Here, we take $\mathsf{Enc}_{\text{PI}}((a, b))$ as the computed messages for each typed edges when message-passing over relation graph with positional encoding (a, b).

Entity encoder. Similarly to how we encode the relation, HYPER uses a variant of HCNet to encode the entities. In the context of the entity encoder, we apply a separate HCNet over the original knowledge hypergraph $G = (V, E, R)$. Each node $v \in V$ aggregates from its incident hyperedges by first taking the relation embeddings $\boldsymbol{h}^{(T)}_{r|\boldsymbol{q}}$ obtained from the relation encoder as the messages for each typed hyperedges and then transformed by a layer-specific MLP.

5 EXPERIMENTS

In this section, we aim to evaluate the generalization and effectiveness of HYPER across inductive link prediction tasks on both knowledge hypergraphs and knowledge graphs. We focus on answering the following questions:

Q1: How well does HYPER generalize to unseen entities and relation types?

Q2: How does HYPER handle varying proportions of unseen relations in the test set?

Q3: How does HYPER compare to KGFMs on reified knowledge hypergraphs?

Q4: What is the impact of different variants of pretraining mix on HYPER?

Q5: How does the encoding of positional information impact the model's ability to generalize?

Q6: How well does HYPER perform on standard knowledge graphs (see Appendix D)?

Q7: What are computational complexity and empirical scalability of HYPER (see Appendix E)?

5.1 EXPERIMENTAL SETUPS

Models. We evaluate models using the datasets summarized in Table 12. As a supervised learning baseline, we include G-MPNN (Yadati, 2020) and also HCNet, as a state-of-the-art node-inductive method on knowledge hypergraphs, which is representative of the performance methods relying on end-to-end training. These models, by design, cannot generalize to unseen relations since they explicitly store the trained relation embeddings, and thus have to assign a randomly initialized vector for the representation of the unseen relations. For a fair comparison, we evaluate HYPER(end2end), HYPER models trained directly on the corresponding train set for each dataset.

Table 2: MRR results on node and relation inductive knowledge hypergraph datasets. Superscript † means the model is applied over the reification of hypergraphs.

Method	JF				MFB				WP				WD			
	25	50	75	100	25	50	75	100	25	50	75	100	25	50	75	100
End-to-End Inference																
G-MPNN	0.006	0.003	0.001	0.002	0.002	0.004	0.007	0.003	0.005	0.002	0.001	0.000	0.001	0.001	0.001	0.001
HCNet	0.011	0.009	0.069	0.028	0.033	0.026	0.016	0.082	0.104	0.050	0.019	0.003	0.086	0.043	0.015	0.007
HYPER	**0.202**	**0.468**	**0.207**	**0.198**	**0.332**	**0.200**	**0.135**	**0.222**	**0.159**	**0.143**	**0.139**	**0.202**	**0.215**	**0.205**	**0.172**	**0.205**
Zero-shot Inference																
ULTRA†(3KG)	0.119	0.304	0.109	0.091	0.209	0.153	0.062	0.222	0.040	0.070	0.067	0.071	0.171	0.201	0.149	0.176
ULTRA†(4KG)	0.099	0.325	0.102	0.132	0.343	0.215	0.111	0.274	0.047	0.091	0.089	0.086	0.094	0.141	0.054	0.075
ULTRA†(50KG)	0.147	0.407	0.126	0.111	0.310	0.218	0.100	0.262	0.045	0.071	0.045	0.065	0.062	0.124	0.104	0.150
HYPER(3KG)	0.148	0.297	0.112	0.130	0.248	0.191	0.039	0.276	**0.143**	0.147	0.186	0.221	0.167	0.158	0.123	0.146
HYPER(4HG)	0.187	0.377	0.188	**0.181**	0.349	0.244	0.139	0.278	0.075	0.068	0.086	0.168	0.087	0.158	0.057	0.165
HYPER(3KG+2HG)	**0.216**	**0.455**	**0.213**	0.173	**0.363**	**0.250**	**0.140**	**0.299**	0.132	**0.152**	**0.192**	**0.222**	**0.223**	**0.200**	**0.154**	**0.182**
Finetuned Inference																
HYPER(3KG+2HG)	0.217	0.456	0.209	0.176	0.347	0.243	0.158	0.275	0.169	0.171	0.194	0.210	0.225	0.234	0.166	0.210

To also evaluate the pretraining paradigm for foundation models, we include ULTRA†(3KG/4KG/50KG) from Galkin et al. (2024) as baseline KGFM models[1]. They are pretrained on increasingly large KG corpora and evaluated on reified hypergraphs for tail-only link prediction, following Section 3. To assess the benefits of pretraining on different relational structures, we experimented with two HYPER variants: HYPER(3KG), trained only on three *knowledge graph* datasets (FB15k-237 (Toutanova & Chen, 2015), WN18RR (Dettmers et al., 2018), and Codex Medium (Safavi & Koutra, 2020)), and HYPER(4HG), trained on four *knowledge hypergraph* datasets (JF17K (Wen et al., 2016), Wikipeople (Guan et al., 2021), FB-AUTO (Fatemi et al., 2020), and M-FB15K (Fatemi et al., 2020)). We further include HYPER(3KG + 2HG), a HYPER model trained on a comprehensive mixture of three knowledge graph (FB15k-237, WN18RR, Codex Medium) and two knowledge hypergraph datasets (JF17K, Wikipeople), aiming to combine the advantages of both types of data, and fine-tuned this checkpoint over the training sets for each downstream task.

Evaluations. We adopt *filtered ranking protocol*: for each query $q(u_1, \cdots, u_k)$ where $k = \texttt{ar}(q)$ and for each position $t \leq k$, we replace the t-th position by all other entities such that the resulting hyperedges does not appear in training, validation, or testing knowledge hypergraphs. We report Mean Reciprocal Rank (MRR) and provide averaged results for *three* runs for the end-to-end and fine-tuned experiments. We report the standard deviation along with the full tables in Table 18 and Table 19. The codebase is provided in `https://anonymous.4open.science/r/HYPER`. See computation resources used in Appendix C and further experimental details in Appendix F.

5.2 NODE-RELATION INDUCTIVE LINK PREDICTION OVER KNOWLEDGE HYPERGRAPHS

Dataset construction and task settings. To evaluate the transferability and generalization capabilities of HYPER, we follow the methodology proposed in InGram (Lee et al., 2023) to construct new datasets with varying proportions of unseen relations. We derive these datasets from three hypergraph datasets: JF17K (Wen et al., 2016) (JF), Wikipeople (Guan et al., 2021) (WP), and M-FB15K (Fatemi et al., 2020) (MFB). We also include WD50K (WD) Galkin et al. (2020), originally a hyper-relational KG, which we convert into a knowledge hypergraph by hashing the main relation and predicates in canonical order. For each source dataset, we create four variants with different percentages of test tuples containing previously unseen relations: 25%, 50%, 75%, and 100%. For instance, JF-25 includes 25% test tuples with unseen relations, while JF-100 contains only entirely unseen relations. This setup[2] allows us to systematically evaluate how models perform under increasingly challenging inductive scenarios. We present all the details in Appendix A.

Overall performances of HYPER (Q1). We report model performance across each dataset in Table 2. Note that HYPER and its variants drastically outperform HCNet in node and relation-inductive

[1]We also include additional baseline results for other KGFM in Appendix F.2.

[2]These percentages are meaningful only in the end-to-end evaluation setting. In the zero-shot setting, all relation types are unobserved.

settings. HCNet relies on learnable embeddings for each relation type and struggles with unseen relations, leading to sharp performance drops under inductive settings. In contrast, HYPER leverages a pretrained relation encoder, enabling strong generalization and robust performance even with entirely unseen relations. Note that fine-tuning HYPER(3KG+2HG) further boosts results, often matching or surpassing HYPER trained from end-to-end. This demonstrates the strong transferability of HYPER's representations and shows that lightweight finetuning on small task-specific datasets can approach end-to-end performance without full retraining.

Impact on the ratio of known relations (Q2). We experiment with multiple relation-split settings that vary the proportion of test triplets involving unseen relations, ranging from 25% to 100%. While node-inductive baselines such as HCNet and G-MPNN already perform poorly under low relational shift (e.g., 25%), their performance degrades substantially as the proportion of unseen relations increases (e.g., 100%), reflecting the difficulty of generalizing to novel relation types. In contrast, HYPER maintains consistently strong performance across all splits, demonstrating its robustness and ability to generalize effectively under an increased proportion of unseen relations.

HYPER vs. ULTRA on reified knowledge hypergraph (Q3). Across all datasets, HYPER consistently outperforms KGFMs like ULTRA† on reified hypergraphs. While KGFMs can in principle generalize to binary relations, reified hypergraphs form atypical structures, e.g., tripartite graphs with auxiliary edge nodes, which is not commonly seen in pretraining corpora. Notably, ULTRA†(50KG), trained on 50 knowledge graphs, performs only marginally better than the version trained on just 3, and remains substantially behind HYPER(3KG + 2HG). This suggests that increasing the number of training graphs does not close the gap introduced by the lack of explicit hypergraph modeling. While reification technically enables the application of KGFMs to knowledge hypergraphs, it fails to capture the structure of entity-role interactions, resulting in significantly weaker performance.

Impact of different pretraining datasets (Q4). The composition of pretraining data has a noticeable impact on generalization. While HYPER(4HG), pretrained on hypergraph datasets, performs strongly on JF and MFB, both of which contain a large proportion of higher-arity relations, it struggles on WP, which primarily consists of binary edges. Conversely, WP benefits more from pretraining on binary relational graphs, as seen with HYPER(3KG). The best overall performance comes from HYPER(3KG + 2HG), which combines both binary and hypergraph pretraining sources. This suggests that pretraining on diverse relation structures and thus the underlying distribution improves generalization across tasks with varying arities.

5.3 NODE INDUCTIVE LINK PREDICTION OVER KNOWLEDGE HYPERGRAPHS

Settings. To further assess the applicability of node-inductive link prediction tasks with knowledge hypergraphs, we follow Yadati (2020) and Huang et al. (2025b), and experiment on three existing datasets: JF-IND, WP-IND, and MFB-IND. We compare our models with several existing approaches for inductive link prediction on knowledge hypergraphs. These include HGNN (Feng et al., 2018) and HyperGCN (Yadati et al., 2019), which were originally designed for simple hypergraphs and adapted to knowledge hypergraphs by ignoring relations (Yadati, 2020).

We also compare with G-MPNN (Yadati, 2020) and RD-MPNN (Zhou et al., 2023b), which were modified for inductive settings by replacing learned entity embeddings with a uniform vector, and HCNet (Huang et al., 2025b). We also include the zero-shot performance of standard KGFM on the reification of hypergraphs ULTRA†(3KG/4KG/50KG).

Table 3: MRR results on node-inductive datasets. Superscript † means the model is applied over the reification of hypergraphs.

Method	JF-IND	WP-IND	MFB-IND
End-to-End Inference			
HGNN	0.102	0.072	0.121
HyperGCN	0.099	0.075	0.118
G-MPNN	0.219	0.177	0.124
RD-MPNN	0.402	0.304	0.122
HCNet	**0.435**	0.414	0.368
HYPER(end2end)	0.422	**0.435**	**0.427**
Zero-shot Inference			
ULTRA†(3KG)	0.173	0.101	0.054
ULTRA†(4KG)	0.286	0.183	0.163
ULTRA†(50KG)	0.346	0.286	0.149
HYPER(3KG)	0.263	0.259	0.184
HYPER(4HG)	0.403	0.375	**0.497**
HYPER(3KG + 2HG)	**0.459**	**0.415**	0.404
Finetuned Inference			
HYPER(3KG + 2HG)	0.463	0.446	0.455

Results and discussion. Table 3 presents the performance of all models across the node-inductive datasets. We continue to observe that HYPER significantly outperforms prior node-inductive baselines such as HCNet, G-MPNN, and RD-MPNN. Among HYPER variants, even without fine-tuning, pretrained HYPER models achieve strong results. Fine-tuned HYPER further improves performance, achieving the best MRR on JF-IND and WP-IND, and competitive results on MFB-IND compared with HYPER trained end-to-end. Notably, HYPER consistently outperforms ULTRA, which struggles to generalize to the distinct structure of reified hypergraphs. These results confirm HYPER's robust generalization across a variety of datasets.

5.4 IMPACT OF POSITIONAL INTERACTION ENCODERS

To evaluate the importance of design choices in the positional interaction encoder $\mathsf{Enc}_{\mathrm{PI}}$ **(Q5)**, we compare HYPER to three alternatives $\mathsf{Enc}_{\mathrm{PI}}$ equipping with different positional encoding schemes: (i) all-one encoding ($\boldsymbol{p}_a = \mathbf{1}^d$), which collapses all positions and violates injectivity; (ii) random encoding ($\boldsymbol{p}_a \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$), which lacks structure and hinders generalization; and (iii) magnitude encoding ($\boldsymbol{p}_a = a\mathbf{1}^d$), which is unbounded and thus unsuitable for MLPs. In contrast, HYPER uses sinusoidal encoding, which is both injective and bounded, enabling effective extrapolation and robust zero-shot performance. As shown in Table 4, sinusoidal encoding yields the best overall performance across 19 hypergraphs, significantly outperforming other schemes in both MRR and Hits@3. This highlights the critical property of injectivity and extrapolation of $\mathsf{Enc}_{\mathrm{PI}}$ in achieving robust zero-shot generalization.

Table 4: Averaged zero-shot performance of HYPER(3KG + 2HG) with different positional interaction encoders.

Model	Total Avg (19 hypergraphs)	
	MRR	**Hits@3**
All-one	0.236	0.262
Random	0.213	0.239
Magnitude	0.227	0.251
Sinusoidal	**0.285**	**0.281**

5.5 CORRUPTION OVER ARGUMENT POSITION

To validate the significance of ordered information in knowledge hypergraphs (**Q5**), we conduct an ablation study to corrupt positional information. Specifically, for each of 16 newly proposed dataset, we take the most frequent relation type in each test graph and **randomly** and **inconsistently** permuted the argument positions for 50% of its hyperedges, making the semantic role of each argument position ambiguous. For instance, a hyperedge $r(a, b, c)$ might become $r(b, a, c)$. We evaluate our HYPER(3KG + 2HG) model in the zero-shot setting on these corrupted datasets. Empirically, we observe that when we permute those relations that explicitly stored ordered information, such as cvg.musical_game_song_relationship in JF-50, the performance drops dramatically, as shown in Figure 6. This is because each argument position carries a distinct semantic role (e.g., musical, game, song), and HYPER relies on implicitly learning these roles to generalize. Corrupting this positional structure prevents HYPER from inferring roles for unseen relations, leading to a dramatic decline in performance.

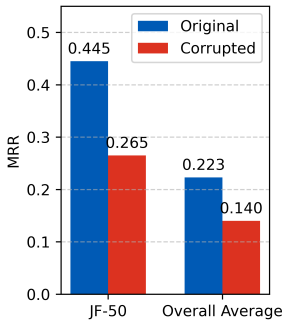

Figure 6: Zero-shot performance of HYPER(3KG + 2HG) over original and corrupted datasets.

6 CONCLUSION

In this work, we introduced HYPER, the first foundation model for inductive link prediction over knowledge hypergraphs with arbitrary arity, capable of generalizing to both unseen entities and unseen relations. Through extensive experiments on 16 newly constructed and 3 existing inductive benchmarks, we demonstrate that HYPER consistently outperforms state-of-the-art knowledge hypergraph baselines and KGFMs applied to reified hypergraphs, demonstrating its strong generalization across varied domains and relational structures. One limitation of HYPER lies in its computational complexity of relation arity: the number of positional interactions grows quadratically with the arity of each hyperedge. Future work may explore scalable approximations to mitigate the cost.

ETHICS STATEMENT

This work proposes a foundation model for inductive reasoning over knowledge hypergraphs, which may benefit applications in scientific discovery, query answering, and recommendation systems by improving generalization across relational contexts. However, the same capabilities could also be misused for generating or reinforcing biased or spurious inferences when applied to real-world knowledge bases that contain noise, imbalance, or socially sensitive information. Future applications should therefore include safeguards for interpretability and error auditing, especially in domains with fairness or safety considerations. We acknowledge and adhere to the ICLR Code of Ethics.

REPRODUCIBILITY STATEMENT

We ensure the reproducibility of our results. A complete description of dataset construction procedures, including inductive splits with varying proportions of unseen relations, is provided in Appendix A. Details of the model architecture, training objectives, and optimization are presented in Section 4 and Appendix F. We further describe efficient implementation details and computational resources in Appendix C. To facilitate verification, we release an anonymous codebase with all scripts for data preprocessing, model training, and evaluation at `https://anonymous.4open.science/r/HYPER`. A detailed method of relation graph constructions is included in Appendix B. Together, these materials provide all the necessary information to reproduce the experiments and results reported in this paper.

REFERENCES

Ralph Abboud, İsmail İlkan Ceylan, Thomas Lukasiewicz, and Tommaso Salvatori. Boxe: A box embedding model for knowledge base completion. In *NeurIPS*, 2020.

Abdalgader Abubaker, Takanori Maehara, Madhav Nimishakavi, and Vassilis Plachouras. Self-supervised pretraining for heterogeneous hypergraph neural networks. *arXiv preprint arXiv:2311.11368*, 2023.

Ivana Balazevic, Carl Allen, and Timothy Hospedales. Tucker: Tensor factorization for knowledge graph completion. In *EMNLP-IJCNLP*, 2019.

Adrián Bazaga, Pietro Liò, and Gos Micklem. Hyperbert: Mixing hypergraph-aware layers with language models for node classification on text-attributed hypergraphs. In *EMNLP*, 2024.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *NIPS*, 2013.

Yuanning Cui, Zequn Sun, and Wei Hu. A prompt-based knowledge graph foundation model for universal in-context reasoning. In *NeurIPS*, 2024.

Tim Dettmers, Minervini Pasquale, Stenetorp Pontus, and Sebastian Riedel. Convolutional 2D knowledge graph embeddings. In *AAAI*, 2018.

Boxin Du, Changhe Yuan, Robert Barton, Tal Neiman, and Hanghang Tong. Hypergraph pre-training with graph neural networks. *arXiv preprint arXiv:2105.10862*, 2021.

Bahare Fatemi, Perouz Taslakian, David Vazquez, and David Poole. Knowledge hypergraphs: Prediction beyond binary relations. In *IJCAI*, 2020.

Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph neural networks. In *AAAI*, 2018.

Yifan Feng, Shiquan Liu, Xiangmin Han, Shaoyi Du, Zongze Wu, Han Hu, and Yue Gao. Hypergraph foundation model. *arXiv preprint arXiv:2503.01203*, 2025.

Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. AMIE: Association rule mining under incomplete evidence in ontological knowledge bases. In *WWW*, 2013.

Mikhail Galkin, Priyansh Trivedi, Gaurav Maheshwari, Ricardo Usbeck, and Jens Lehmann. Message passing for hyper-relational knowledge graphs. In *EMNLP*, 2020.

Mikhail Galkin, Etienne Denis, Jiapeng Wu, and William L. Hamilton. Nodepiece: Compositional and parameter-efficient representations of large knowledge graphs. In *ICLR*, 2022.

Mikhail Galkin, Xinyu Yuan, Hesham Mostafa, Jian Tang, and Zhaocheng Zhu. Towards foundation models for knowledge graph reasoning. In *ICLR*, 2024.

Jianfei Gao, Yangze Zhou, Jincheng Zhou, and Bruno Ribeiro. Double equivariance for inductive link prediction for both new nodes and new relation types. In *arXiv*, 2023.

Saiping Guan, Xiaolong Jin, Jiafeng Guo, Yuanzhuo Wang, and Xueqi Cheng. Link prediction on n-ary relational data based on relatedness evaluation. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

Xingyue Huang, Miguel Romero Orth, İsmail İlkan Ceylan, and Pablo Barceló. A theory of link prediction via relational weisfeiler-leman on knowledge graphs. In *NeurIPS*, 2023.

Xingyue Huang, Pablo Barceló, Michael M. Bronstein, İsmail İlkan Ceylan, Mikhail Galkin, Juan L Reutter, and Miguel Romero Orth. How expressive are knowledge graph foundation models? In *ICML*, 2025a.

Xingyue Huang, Miguel A. Romero Orth, Pablo Barceló, Michael M. Bronstein, and İsmail İlkan Ceylan. Link prediction with relational hypergraphs. *TMLR*, 2025b.

Jaejun Lee, Chanyoung Chung, and Joyce Jiyoung Whang. Ingram: Inductive knowledge graph embedding via relation graphs. In *ICML*, 2023.

Shuwen Liu, Bernardo Grau, Ian Horrocks, and Egor Kostylev. Indigo: Gnn-based inductive knowledge graph completion using pair-wise encoding. In *NeurIPS*, 2021.

Haitao Mao, Zhikai Chen, Wenzhuo Tang, Jianan Zhao, Yao Ma, Tong Zhao, Neil Shah, Mikhail Galkin, and Jiliang Tang. Position: Graph foundation models are already here. In *ICML*, 2024.

Tara Safavi and Danai Koutra. CoDEx: A Comprehensive Knowledge Graph Completion Benchmark. In *EMNLP*, 2020.

Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *ESWC*, 2018.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. In *ICLR*, 2019.

Komal K. Teru, Etienne G. Denis, and William L. Hamilton. Inductive relation prediction by subgraph reasoning. In *ICML*, 2020.

Kristina Toutanova and Danqi Chen. Observed versus latent features for knowledge base and text inference. In *Workshop on Continuous Vector Space Models and their Compositionality*, 2015.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *ICML*, pp. 2071–2080. PMLR, 2016.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. Composition-based multi-relational graph convolutional networks. In *ICLR*, 2020.

Jianfeng Wen, Jianxin Li, Yongyi Mao, Shini Chen, and Richong Zhang. On the representation and embedding of knowledge bases beyond binary relations. In *IJCAI*, 2016.

Naganand Yadati. Neural message passing for multi-relational ordered and recursive hypergraphs. In *NeurIPS*, 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 *NeurIPS*, 2019.

Mingdai Yang, Zhiwei Liu, Liangwei Yang, Xiaolong Liu, Chen Wang, Hao Peng, and Philip S Yu. Instruction-based hypergraph pretraining. In *SIGIR*, 2024.

Muhan Zhang, Pan Li, Yinglong Xia, Kai Wang, and Long Jin. Labeling trick: A theory of using graph neural networks for multi-node representation learning. In *NeurIPS*, 2021.

Yongqi Zhang and Quanming Yao. Knowledge graph reasoning with relational digraph. In *WebConf*, 2022.

Yongqi Zhang, Zhanke Zhou, Quanming Yao, Xiaowen Chu, and Bo Han. Adaprop: Learning adaptive propagation for graph neural network based knowledge graph reasoning. In *KDD*, 2023.

Yucheng Zhang, Beatrice Bevilacqua, Mikhail Galkin, and Bruno Ribeiro. TRIX: A more expressive model for zero-shot domain transfer in knowledge graphs. In *LoG*, 2024.

Jincheng Zhou, Beatrice Bevilacqua, and Bruno Ribeiro. A multi-task perspective for link prediction with new relation types and nodes. In *NeurIPS GLFrontiers*, 2023a.

Xue Zhou, Bei Hui, Ilana Zeira, Hao Wu, and Ling Tian. Dynamic relation learning for link prediction in knowledge hypergraphs. In *Appl Intell*, 2023b.

Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal Xhonneux, and Jian Tang. Neural bellman-ford networks: A general graph neural network framework for link prediction. In *NeurIPS*, 2021.

Zhaocheng Zhu, Xinyu Yuan, Mikhail Galkin, Sophie Xhonneux, Ming Zhang, Maxime Gazeau, and Jian Tang. A*net: A scalable path-based reasoning approach for knowledge graphs. In *NeurIPS*, 2023.

A Dataset generation details

A.1 Generating Datasets for Node and Relation-inductive Link Prediction

To evaluate our models in an inductive setting, we created multiple dataset variants with different proportions of unseen relations. Our dataset generation process, following InGram (Lee et al., 2023), is detailed in Algorithm 1. This process creates training and inference hypergraphs with controlled percentages of unseen relations in the test set.

Algorithm 1 Generating Datasets for Node and Relation-inductive Link Prediction

Require: Source knowledge hypergraph $G = (V, E, R)$, number of training entities n_{train}, number of inference entities n_{test}, relation percentage p_{rel}, tuple percentage p_{tri}, seed value

Ensure: Training knowledge hypergraph $G_{\text{train}} = (V_{\text{train}}, E_{\text{train}}, R_{\text{train}})$ and Inference knowledge hypergraph $G_{\text{inf}} = (V_{\text{inf}}, E_{\text{inf}}, R_{\text{inf}})$

$G \leftarrow$ Giant connected component of G
Randomly split R into R_{train} and R_{inf} such that $|R_{\text{train}}| : |R_{\text{inf}}| = (1 - p_{\text{rel}}) : p_{\text{rel}}$
Uniformly sample n_{train} entities from V and form V_{train} by taking the sampled entities and their neighbors
$E_{\text{train}} := \{r(v_1, v_2, \ldots, v_n) | v_i \in V_{\text{train}}, r \in R_{\text{train}}, r(v_1, v_2, \ldots, v_n) \in E\}$
$E_{\text{train}} \leftarrow$ Hyperedges in the giant connected component of E_{train}
$V_{\text{train}} \leftarrow$ Entities involved in E_{train}
$R_{\text{train}} \leftarrow$ Relations involved in E_{train}
Let G' be the subgraph of G where the entities in V_{train} are removed
In G', uniformly sample n_{test} entities and form V_{inf} by taking the sampled entities and their neighbors
$E_{\text{inf}} := X \cup Y$ such that $|X| : |Y| = (1 - p_{\text{tri}}) : p_{\text{tri}}$ where $X := \{r(v_1, v_2, \ldots, v_n) | v_i \in V_{\text{inf}}, r \in R_{\text{train}}, r(v_1, v_2, \ldots, v_n) \in E\}$ and $Y := \{r(v_1, v_2, \ldots, v_n) | v_i \in V_{\text{inf}}, r \in R_{\text{inf}}, r(v_1, v_2, \ldots, v_n) \in E\}$
$E_{\text{inf}} \leftarrow$ Hyperedges in the giant connected component of E_{inf}
$V_{\text{inf}} \leftarrow$ Entities involved in E_{inf}
$R_{\text{inf}} \leftarrow$ Relations involved in E_{inf}
Split E_{inf} into auxiliary, validation, and test sets with a ratio of 3:1:1

The parameter p_{tri} controls the percentage of test tuples containing unseen relations. For example, when $p_{\text{tri}} = 0.25$, approximately 25% of the tuples in the inference hypergraph contain relations not seen during training. This allows us to systematically evaluate how models perform under increasingly challenging inductive scenarios.

After generating the inference hypergraph, we split it into three disjoint sets: auxiliary (for training), validation, and test sets with a ratio of 3:1:1. For a fair comparison, these sets are fixed and provided to all models.

A.2 Dataset Statistics

Table 5 and Table 6 summarize the statistics of our constructed datasets and the hyperparameters used to generate them, respectively. Additionally, Table 7 presents the arity distribution across these datasets. Together, these tables illustrate that our benchmarks vary significantly in terms of arity, density, and number of relation types, ensuring a diverse and comprehensive evaluation setting.

B Sparse Matrix Multiplication for Computing Positional Interaction

In this section, we describe the procedure to generalize sparse matrix multiplication to efficiently construct knowledge hypergraphs from hyperedges of arbitrary arity. Unlike knowledge graphs (Galkin et al., 2024), where only two positions (head and tail) exist per relation, resulting in only 4 fundamental relations (head-to-head, head-to-tail, tail-to-head, tail-to-tail), knowledge hypergraphs involve k positions per hyperedges, leading to k^2 types of possible positional interactions in total.

Table 5: Statistics of datasets for inductive hypergraph completion. Max arity is shown for training graph and inference graph, respectively.

Dataset	Train			Inference			Test			Max Arity
	$\|V\|$	$\|R\|$	$\|E\|$	$\|V\|$	$\|R\|$	$\|E\|$	$\|V\|$	$\|R\|$	$\|E\|$	
JF-25	2,616	41	3,371	1,159	36	1,056	209	15	103	5/4
JF-50	2,859	53	3,524	1,102	37	1,292	157	5	109	5
JF-75	3,129	67	4,287	1,488	38	1,697	225	11	131	5
JF-100	2,123	48	2,449	1,696	35	2,159	52	5	25	5
WP-25	6,378	128	7,453	2,784	66	4,794	830	19	959	6/4
WP-50	7,586	155	9,536	3,608	87	4,390	531	29	413	7/6
WP-75	7,787	118	9,271	4,737	101	6,221	629	27	459	6
WP-100	7,787	118	9,271	4,891	63	7,516	275	15	155	6
WD-25	4,533	239	5,482	3,008	191	3,106	250	37	148	22/5
WD-50	3,796	162	4,147	2,303	188	2,353	145	30	91	19/6
WD-75	6,518	243	6,305	5,194	244	5,831	547	57	385	22/5
WD-100	6,798	237	7,271	3,576	105	3,951	385	29	282	19/4
MFB-25	1,266	11	8,182	1,929	12	2,802	146	7	87	3/5
MFB-50	1,415	11	8,409	1,528	13	2,426	472	10	486	3/5
MFB-75	2,225	15	5,271	1,363	16	4,008	675	11	803	3/4
MFB-100	2,013	19	11,658	2,406	5	4,514	808	5	904	3/5

Table 6: Hyperparameters used to create fully inductive knowledge hypergraph datasets.

HP	JF-25	JF-50	JF-75	JF-100	WP-25	WP-50	WP-75	WP-100
n_{train}	1000	1000	1200	1200	900	800	1000	1000
n_{test}	900	800	1200	1200	800	1000	1000	1000
p_{rel}	0.4	0.5	0.4	0.5	0.4	0.3	0.5	0.5
p_{tri}	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0
HP	**WD-25**	**WD-50**	**WD-75**	**WD-100**	**MFB-25**	**MFB-50**	**MFB-75**	**MFB-100**
n_{train}	700	1000	10000	10000	100	100	80	120
n_{test}	1200	1000	8000	8000	95	85	80	100
p_{rel}	0.25	0.5	0.5	0.5	0.5	0.6	0.5	0.5
p_{tri}	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0

Given a knowledge hypergraph $G = (V, E, R)$ with $n = |V|$ nodes, $m = |R|$ relations, and maximum arity k, we start by representing the knowledge hypergraph via sparse tensors: the edge index $\boldsymbol{E} \in \mathbb{N}^{k\times|E|}$ and corresponding edge types $\boldsymbol{r} \in \mathbb{N}^{|E|}$. Each column of $\boldsymbol{E}$ lists the k participating nodes for a hyperedge, with each edge associated with its relation type.

To encode positional interactions between relations, we perform sparse matrix multiplication in the following steps:

1. For each position $a \in \{1, \cdots, k\}$, we construct sparse matrices $\boldsymbol{E}_a \in \mathbb{R}^{n\times m}$ where each nonzero entry indicates the presence of an entity at position a for a given relation type.
2. For each pair of positions $(a, b) \in \{1, \cdots, k\} \times \{1, \cdots, k\}$, we compute a sparse matrix multiplication:
$$\boldsymbol{A}_{a2b} = \text{spmm}(\boldsymbol{E}_a^\top, \boldsymbol{E}_b) \in \mathbb{R}^{m\times m}.$$
Here, $(\boldsymbol{A}_{a2b})_{i,j}$ is nonzero if there exists an entity that simultaneously plays position a in a hyperedge of relation i and position b in a hyperedge of relation j.

This operation systematically captures all intersections between hyperedges that share at least one common node, generalized across different positions.

C Computational Resources

All the pretraining experiments is carried out on a single NVIDIA H100 80GB, and the rest of the experiments are carried out using a NVIDIA A10 24GB. Pretraining of HYPER over a single H100 with parameter specified in Appendix F takes 4 days, while fine-tuning and end-to-end training typically require less than 3 hours.

HYPER is implemented primarily using PyTorch and PyTorch Geometric (Fey & Lenssen, 2019), with its core hypergraph message passing implemented via a custom-built Triton kernel[3]. This optimization approximately halves the training time and reduces memory consumption by a factor of five on average. Instead of explicitly materializing all hyperedge messages, as is done in PyTorch Geometric, we directly write neighboring features to the corresponding memory locations during aggregation. While the naive materialization approach incurs $O(k|E|)$ memory complexity, where k denotes the maximum arity and $|E|$ the number of hyperedges, our Triton-based approach achieves $O(|V|)$ memory complexity, depending only on the number of nodes, which enables efficient and scalable training of HYPER models.

D Additional Experiments on Knowledge Graphs

In addition to the knowledge hypergraph inductive settings, we also evaluate our models on inductive knowledge graph link prediction tasks where both nodes and relations can be unseen during training **(Q6)**. This setting presents the most challenging scenario as it requires models to generalize to entirely new knowledge domains with both unseen entities and relation types. We also include inductive node-only knowledge graph link prediction to further strengthen our point.

Datasets. For inductive on both nodes and relations task, we includes 13 datasets in INGRAM (Lee et al., 2023): FB-25, FB-50, FB-75, FB-100, WK-25, WK-50, WK-75, WK-100, NL-0, NL-25, NL-50, NL-75, NL-100; and 10 datasets in MTDEA (Zhou et al., 2023a): MT1 tax, MT1 health, MT2 org, MT2 sci, MT3 art, MT3 infra, MT4 sci, MT4 health, Metafram, FBNELL. We also include inductive link prediction on nodes only experiments, containing 12 datasets from GraIL (Teru et al., 2020): WN-v1, WN-v2, WN-v3, WN-v4, FB-v1, FB-v2, FB-v3, FB-v4, NL-v1, NL-v2, NL-v3, NL-v4; 4 datasets from INDIGO (Liu et al., 2021): HM 1k, HM 3k, HM 5k, HM Indigo; and 2 datasets from Nodepiece (Galkin et al., 2022): ILPC Small, ILPC Large.

Baseline. We included the zero-shot version of all the models and also include an existing knowledge graph foundation model as baseline, ULTRA (Galkin et al., 2024), shown in Table 8, Table 9. Notably, following standard convention, for every triplet $r(u, v)$ in a knowledge graph, we also include its inverse triplet $r^{-1}(v, u)$, where r^{-1} denotes a newly introduced relation symbol representing the inverse of r for ULTRA. However, HYPER does not need this procedure as the entity encoder employs a variant of HCNet (Huang et al., 2025b), which uses bi-directional message-passing and automatically considers the message from the inverse direction.

Results and Discussion. We observe that HYPER achieves comparable performance to ULTRA in zero-shot inductive link prediction on knowledge graphs. Across both node-only and node-and-relation inductive benchmarks, HYPER performs on par with ULTRA, and often outperforms it on datasets with higher relational diversity or structure. These results demonstrate that the architectural inductive bias of HYPER, originally designed for knowledge hypergraphs, also transfers well to standard knowledge graphs, without compromising generalization ability.

E Complexity and Scalability Analysis of HYPER

To answer **Q7**, we first examine the theoretical computational complexity of HYPER in Appendix E.1, then present its empirical scalability results when applying on FB15k-237 Appendix E.2.

[3]https://github.com/triton-lang/triton

E.1 Theoretical Computational Complexity

In this section, we analyze the computational complexity of HYPER. Let $G = (V, E, R)$ denote the input knowledge hypergraph, where $n = |V|$, $m = |E|$, and $|R|$ are the number of entities, hyperedges, and relation types, respectively. Let k be the maximum arity of R, d the hidden dimension, and T the number of message-passing layers in the relation encoder, and denote L as the number of message-passing layers in the entity encoder.

Relation Graph Construction The complexity of generating the relation graph in HYPER arises from computing pairwise positional interactions between relation types across hyperedges of arbitrary arity. Unlike knowledge graphs, where each relation involves exactly two fixed positions (head and tail), knowledge hypergraphs induce up to k^2 positional interaction types for a maximum arity k. For each position $a \in \{1, \dots, k\}$, we construct sparse matrices $\boldsymbol{E}_a \in \mathbb{R}^{n \times m}$ that index entities by their position and relation type. Then, for every pair (a, b), we perform a sparse matrix multiplication: $\mathrm{spmm}(\boldsymbol{E}_a^\top, \boldsymbol{E}_b)$. Each such multiplication has a worst-case complexity of $\mathcal{O}(\mathsf{nnz}(\boldsymbol{E}_a^\top) \cdot \mathsf{nnz}(\boldsymbol{E}_b))$, where $\mathsf{nnz}(\cdot)$ denotes the number of nonzero entries. Since there are k^2 position pairs, the total time complexity of constructing the relation graph becomes

$$\mathcal{O}(k^2 \cdot \max_{\{a,b\}} \{\mathsf{nnz}(\boldsymbol{E}_a^\top) \cdot \mathsf{nnz}(\boldsymbol{E}_b)\}).$$

In practice, this is significantly accelerated by sparse tensor and batching across position pairs. Without sparse matrix multiplication, the naive construction would require iterating over all hyperedge pairs, resulting in $\mathcal{O}(k^2|E|^2)$ complexity, which is infeasible for large-scale datasets.

Additionally, for the positional interaction encoders, we associate a positional encoding vector $\mathsf{Enc}_{\text{PI}}((a, b)) \in \mathbb{R}^d$. This construction requires $\mathcal{O}(k^2 d)$ time and space to compute and store.

Relation Encoder The relation encoder in HYPER performs T layers of message passing over the relation graph $G_{\text{rel}} = (V_{\text{rel}}, E_{\text{rel}}, R_{\text{rel}})$, as constructed before.

There are at most k^2 position pairs per pair of relation types, where k is the maximum arity, so the total number of edges is bounded by

$$|E_{\text{rel}}| = \mathcal{O}(|R|^2 k^2).$$

In each message passing layer, each relation node aggregates messages from up to $|R|^2 k^2$ neighbors, with each edge contributing a message via the corresponding positional interaction embedding $\boldsymbol{x}_{a,b} = \mathsf{Enc}_{\text{PI}}((a, b)) \in \mathbb{R}^d$. Each node then applies an update with cost $\mathcal{O}(d^2)$. Thus, the total complexity of the relation encoder over T layers is

$$\mathcal{O}\left(T(|R|^2 k^2 d + |R| d^2)\right).$$

Entity Encoder After obtaining relation embeddings from the relation encoder, HYPER applies L layers of conditional message passing over the original knowledge hypergraph $G = (V, E, R)$ using HCNet (Huang et al., 2025b). In each layer, every entity $v \in V$ aggregates messages from its incident hyperedges $e \in E(v)$, where each hyperedge contributes a query-conditioned message, taking $\mathcal{O}(L(k|E|d)$, that incorporates its relation embedding $h^{(T)}_{\rho(e)|q} \in \mathbb{R}^d$, followed by a relation-specific MLP, which takes $\mathcal{O}(L|R|d^2)$. Each entity then updates its representation through a neural update function with cost $\mathcal{O}(d^2)$.

The total complexity of the entity encoder over L layers is thus

$$\mathcal{O}(L(k|E|d + |V|d^2 + |R|d^2)).$$

E.2 Scalability Analysis

To empirically assess the scalability of HYPER, we compare HYPER with ULTRA, a prominent knowledge graph foundation model, and HCNet, a state-of-the-art node-inductive method on link prediction with knowledge hypergraph. All experiments are conducted on the *transductive* knowledge graph dataset FB15k-237 using a batch size of 64 to ensure a fair comparison among all three

methods. We summarize the model parameter size, training/inference times, and GPU memory usage.

Compared with HCNets, HYPER's training and inference times are approximately doubled since HYPER employs *two* HCNet encoders, one for relations and one for entities. We argue that this overhead represents a reasonable trade-off for the substantial performance improvements and stronger inductive generalization demonstrated by HYPER compared with HCNets.

Compared with ULTRA, the main bottleneck of scalability is the complex modeling of knowledge graphs as knowledge hypergraphs. These differences essentially reduce to the difference between HCNet and NBFNets. For a detailed discussion, we refer the reader to Huang et al. (2025b).

F FURTHER EXPERIMENTAL DETAILS

In this section, we provide detailed experimental configurations and dataset statistics. In particular, Table 12 summarizes the training corpora used for each model variant across knowledge graph and knowledge hypergraph settings. Tables 13 and 14 present arity distributions and structural statistics for the node-inductive datasets, while Table 15 reports the corresponding statistics for pretraining datasets. For inductive link prediction involving unseen entities and relations, we provide comprehensive dataset breakdowns in Tables 16 and 17.

We also include the complete performance tables together with standard deviation for the node-inductive and node-relation inductive settings shown in Tables 18 and 19, respectively. Table 20 lists all hyperparameter choices used for pretraining, fine-tuning, and end-to-end training of HYPER. Finally, Table 21 specifies the dataset-specific training schedules for each experimental regime.

F.1 HYPERPARAMETER DETAILS FOR BASELINES

For G-MPNNs, we adopt the best-performing hyperparameters from the original codebase. Specifically, we set the input dimension $d = 64$, hidden dimension $h = 150$, and dropout rate to 0.5. We use a training batch size $b = 128$, evaluation batch size $B = 4$, and negative sampling ratio $nr = 10$. The learning rate is set to 0.0005, and models are trained for up to 5000 epochs with validation evaluated every 5 epochs. Aggregation is performed using the max.

For HCNet, we use a 6-layer encoder with an input dimension of 64 and a hidden dimension of 64 for all layers. We adopt `sum` as the aggregation function and enable shortcut connections to facilitate training. Optimization is performed using AdamW with a learning rate of 5×10^{-4}. Training is conducted with a batch size of 8, using the same number of epochs and batches per epoch as HYPER, with validation performed every 100 steps. The model is trained with 256 adversarial negatives sampled per positive example, and strict negative sampling is enforced to prevent overlap with true triples.

F.2 ADDITIONAL BASELINES FOR KGFMS

To provide a more comprehensive evaluation of Knowledge Graph Foundation Models (KGFMs), we expanded our analysis beyond ULTRA, which served as the representative KGFM in the main paper. We additionally evaluated KG-ICL (Cui et al., 2024), a model that has demonstrated strong performance on inductive knowledge graph completion tasks. For KG-ICL, we performed zero-shot experiments across all 16 proposed datasets after reification with 3 variants of KG-ICL using 4 Layer, 5 Layer, and 6 Layer encoders, respectively. We highlight that KG-ICL learns over different pretraining mix than ULTRA (FB-v1, NL-v1, and Codex Small). The results, summarized in Table 11, indicate that KG-ICL's performance is substantially lower than that of HYPER, further supporting the conclusion that existing KGFMs exhibit limited generalization on hypergraph structures.

F.3 ARCHITECTURE CHOICES OF HYPER

Both the relation and entity encoders in HYPER follow the design based on HCNets (Huang et al., 2025b), with a minor variant on the relation-specific message functions.

Positional Interactions. In practice, we implement Enc_{PI} as a two-layer multilayer perceptron (MLP) over concatenated sinusoidal encodings of the input positions. Let $\boldsymbol{p}_a, \boldsymbol{p}_b \in \mathbb{R}^d$ denote the sinusoidal positional encodings of positions a and b, respectively. Then, the embedding corresponding to the interaction (a, b) is computed as $\mathbf{x}_{a,b} = \text{MLP}([\boldsymbol{p}_a \,\|\, \boldsymbol{p}_b])$, where MLP denotes a shared two-layer feedforward network with ReLU activations. This produces a dense embedding that captures the interaction between the two positions. Empirically, we find that this instantiation of Enc_{PI} enables strong generalization across knowledge hypergraphs with varying arities and relational structures.

Relation Encoder. The relation encoder applies an HCNet over the constructed relation graph $(V_{\text{rel}}, E_{\text{rel}}, R_{\text{rel}})$. Here, each node $r \in V_{\text{rel}}$ represents a relation type in G, and an edge captures the induced interactions among relations. For each relation $r \in V_{\text{rel}}$, HCNet iteratively updates its representation $\boldsymbol{h}^{(t)}_{r|\boldsymbol{q}}$ as:

$$\boldsymbol{h}^{(0)}_{r|\boldsymbol{q}} = \textsc{init}_{\text{rel}}(r, \boldsymbol{q}),$$
$$\boldsymbol{h}^{(t+1)}_{r|\boldsymbol{q}} = \textsc{up}_{\text{rel}}\Big(\boldsymbol{h}^{(t)}_{r|\boldsymbol{q}}, \textsc{agg}_{\text{rel}}\big(\{\!\!\{\textsc{msg}_{\rho(e)}\big(\{(\boldsymbol{h}^{(t)}_{r'|\boldsymbol{q}}, j) \mid (r', j) \in \mathcal{N}_{\text{rel}\,i}(e)\}, \boldsymbol{q}\big) \mid (e, i) \in E_{\text{rel}}(r)\}\!\!\}\big)\Big),$$

where $E_{\text{rel}}(r)$ is the set of edge-position pairs incident to r, and $\mathcal{N}_{\text{rel}\,i}(e)$ is the positional neighborhood of hyperedge e at position i. After T layers, we obtain the final relation encoding $\boldsymbol{h}^{(T)}_{r|\boldsymbol{q}}$. Here, $\textsc{init}_{\text{rel}}$, $\textsc{up}_{\text{rel}}$, $\textsc{agg}_{\text{rel}}$, and $\textsc{msg}_{\rho(e)}$ are differentiable initialization, update, aggregation, and fundamental relation-specific message functions, respectively. The initialization function $\textsc{init}_{\text{rel}}$ is designed to satisfy *generalized target node distinguishability* as formalized in Huang et al. (2025b).

Empirically, we initialize the query node $q \in V_{\text{rel}}$ with an all-one vector and all other relation nodes with zero vectors.

In the experiments, we adopt the fundamental relation-specific message function $\textsc{msg}_{r_{\text{fund}}}$ using the fundamental relation embedding $\mathbf{r}_{a,b}$. Specifically, given a set of neighbor features $\{(\boldsymbol{h}^{(\ell)}_{w|\boldsymbol{q}}, j) \mid (w, j) \in \mathcal{N}_{\text{rel}\,i}(e)\}$ for hyperedge e and center position i, the message is computed as:

$$\textsc{msg}_{r_{a,b}}\Big(\{(\boldsymbol{h}^{(t)}_{w|\boldsymbol{q}}, j) \mid (w, j) \in \mathcal{N}_{\text{rel}\,i}(e)\}\Big) = \left(\bigodot_{j \neq i}\Big(\alpha^{(t)}\boldsymbol{h}^{(t)}_{e(j)|\boldsymbol{q}} + (1 - \alpha^{(t)})\boldsymbol{p}_j\Big)\right) \odot \mathbf{x}_{a,b},$$

where $\odot$ is the elemental-wise multiplication, $\alpha^{(\ell)}$ is a learnable scalar, $\boldsymbol{p}_j$ is the sinusoidal positional encoding at position j, and $\mathbf{x}_{a,b}$ is the fundamental relation embedding computed as described earlier.

Entity Encoder. In the context of the entity encoder, we apply a separate HCNet over the original knowledge hypergraph $G = (V, E, R)$. Each node $v \in V$ aggregates information from its incident hyperedges, incorporating the relation embeddings $\boldsymbol{h}^{(T)}_{r|\boldsymbol{q}}$ obtained from the relation encoder.

Given a query $\boldsymbol{q} = (q, \tilde{\boldsymbol{u}}, t)$, where $\tilde{\boldsymbol{u}} = (u_1, \ldots, u_k)$ denotes the entities in the hyperedge and t is the target position, each node $v \in V$ receives an initial representation defined as:

$$\boldsymbol{h}^{(0)}_{v|\boldsymbol{q}} = \sum_{i \neq t} \mathbb{1}_{v=u_i} \cdot (\boldsymbol{p}_i + \boldsymbol{z}_q),$$

where $\boldsymbol{p}_i \in \mathbb{R}^d$ is the positional encoding at position i, and $\boldsymbol{z}_q \in \mathbb{R}^d$ is a learned embedding for the query relation q.

HYPER then iteratively updates the node representations $\boldsymbol{h}^{(\ell)}_{v|\boldsymbol{q}}$ as:

$$\boldsymbol{h}^{(0)}_{v|\boldsymbol{q}} = \textsc{init}(v, \boldsymbol{q}),$$
$$\boldsymbol{h}^{(\ell+1)}_{v|\boldsymbol{q}} = \textsc{up}\Big(\boldsymbol{h}^{(\ell)}_{v|\boldsymbol{q}}, \textsc{agg}\big(\{\!\!\{\textsc{msg}_{\rho(e)}\big(\{(\boldsymbol{h}^{(\ell)}_{w|\boldsymbol{q}}, j) \,|\, (w, j) \in \mathcal{N}_i(e)\}, \boldsymbol{h}^{(T)}_{\rho(e)|\boldsymbol{q}}, \boldsymbol{q}\big) \,|\, (e, i) \in E(v)\}\!\!\}\big)\Big).$$

where $\textsc{init}$, $\textsc{up}$, $\textsc{agg}$, and $\textsc{msg}_r$ are differentiable initialization, update, aggregation, and relation-specific message functions, respectively, with $\textsc{init}$ satisfying *generalized targets node distinguishability* (Huang et al., 2025b). After L layers of message passing, we obtain the final entity encoding

$\boldsymbol{h}^{(L)}_{v|\boldsymbol{q}}$. A final unary decoder $\mathsf{Dec} : \mathbb{R}^{d(L)} \to [0,1]$ predicts the score for completing the missing position t in the query $\boldsymbol{q}$.

Empirically, we select the relation-specific message function $\text{MSG}_{\rho(e)}$ to be

$$\text{MSG}_r\left(\{(\boldsymbol{h}^{(\ell)}_{w|\boldsymbol{q}}, j) \mid (w,j) \in \mathcal{N}_i(e)\}\right) = \left(\bigodot_{j\neq i}\left(\alpha^{(\ell)}\boldsymbol{h}^{(\ell)}_{e(j)|\boldsymbol{q}} + (1-\alpha^{(\ell)})\boldsymbol{p}_j\right)\right) \odot \text{MLP}^{(\ell)}(\boldsymbol{h}^{(T)}_{\rho(e)|\boldsymbol{q}}),$$

where additionally $\text{MLP}^{(\ell)}$ is a 2-layer MLP with ReLU to transform the relation representation most suitable for each specific layer during message passing.

Update. We use summation as the aggregation operator for both relation and entity nodes. Each node updates its representation via a two-layer MLP applied to the concatenation of its current state and the aggregated message:

$$\boldsymbol{h}^{(\ell+1)}_{v|\boldsymbol{q}} = \text{MLP}^{(\ell)}\left([\boldsymbol{h}^{(\ell)}_{v|\boldsymbol{q}} \,\|\, \text{AGGREGATE}^{(\ell)}_{v|\boldsymbol{q}}]\right),$$

where $\text{AGGREGATE}^{(\ell)}_{v|\boldsymbol{q}}$ denotes the sum of incoming messages to node v under query $\boldsymbol{q}$ at layer ℓ, and $\|$ represents vector concatenation.

Other. We also apply layer normalization and shortcut connections after aggregation and before the ReLU activation in both encoders.

F.4 Training Objective

Following prior work (Huang et al., 2025b), we train HYPER under the *partial completeness assumption* (Galárraga et al., 2013), where each k-ary fact $q(u_1, \ldots, u_k)$ is used to generate training samples by randomly masking one position $1 \le t \le k$. Given a query $\boldsymbol{q} = (q, \tilde{\boldsymbol{u}}, t)$, we model the conditional probability of entity $v \in V$ filling the missing position as $p(v|\boldsymbol{q}) = \sigma(\mathsf{Dec}(\boldsymbol{h}^{(L)}_{v|\boldsymbol{q}}))$, where Dec is a two-layer MLP and σ denotes the sigmoid activation. We optimize the following self-adversarial negative sampling loss (Sun et al., 2019):

$$\mathcal{L}(v|\boldsymbol{q}) = -\log p(v|\boldsymbol{q}) - \sum_{i=1}^{n} w_{i,\alpha} \log(1 - p(v'_i|\boldsymbol{q})),$$

where v'_i are corrupted negative samples, n is the number of negatives per query, α being the adversarial temperature, and $w_{i,\alpha}$ are the importance weights defined by

$$w_{i,\alpha} = \text{Softmax}\left(\frac{\log(1 - p(v'_i|\boldsymbol{q}))}{\alpha}\right).$$

To mitigate overfitting, we exclude edges that directly connect query node pairs during training. The best model checkpoint is selected based on validation performance. Following the implementation of ULTRA (Galkin et al., 2024), for pertaining over multiple knowledge graph and knowledge hypergraphs, for each batch, we sample from one of the pretrained (hyper)graphs with probability proportional to the number of edges it contains.

Table 7: Arity distribution across node-relation inductive datasets.

Dataset	Arity	Training Graph	Inference Graph	Training %	Inference %
JF-25	2	1585	266	47.02%	25.19%
	3	1441	670	42.75%	63.45%
	4	326	120	9.67%	11.36%
	≥5	19	0	0.56%	0.00%
JF-50	2	1942	321	55.11%	24.85%
	3	1297	692	36.80%	53.56%
	4	285	279	8.09%	21.59%
	≥5	0	0	0.00%	0.00%
JF-75	2	2641	824	61.60%	48.56%
	3	848	846	19.78%	49.85%
	4	779	27	18.17%	1.59%
	≥5	19	0	0.44%	0.00%
JF-100	2	1349	1637	55.08%	75.82%
	3	570	283	23.27%	13.11%
	4	511	159	20.87%	7.36%
	≥5	19	80	0.78%	3.71%
WD-25	2	4,331	2,799	79.00%	90.12%
	3	612	162	11.16%	5.22%
	4	463	144	8.45%	4.64%
	≥5	76	1	1.39%	0.03%
WP-50	2	7709	4355	80.84%	99.20%
	3	1106	28	11.60%	0.64%
	4	667	3	6.99%	0.07%
	≥5	54	4	0.57%	0.09%
WP-75	2	6471	6121	69.80%	98.39%
	3	1725	82	18.61%	1.32%
	4	1026	15	11.07%	0.24%
	≥5	49	3	0.53%	0.05%
WP-100	2	6471	7413	69.80%	98.63%
	3	1725	91	18.61%	1.21%
	4	1026	6	11.07%	0.08%
	≥5	49	6	0.53%	0.08%
WD-25	2	3,680	1,941	87.60%	93.81%
	3	211	114	5.02%	5.51%
	4	279	12	6.64%	0.58%
	≥5	31	2	0.74%	0.10%
WD-50	2	3,238	2,127	78.08%	90.40%
	3	417	86	10.06%	3.65%
	4	438	136	10.56%	5.78%
	≥5	54	4	1.30%	0.17%
WD-75	2	4,900	5,669	77.72%	97.22%
	3	769	139	12.20%	2.38%
	4	548	22	8.69%	0.38%
	≥5	88	1	1.40%	0.02%
WD-100	2	5,858	3,631	80.57%	91.90%
	3	906	186	12.46%	4.71%
	4	397	134	5.46%	3.39%
	≥5	110	0	1.51%	0.00%
MFB-25	2	137	1555	1.67%	55.50%
	3	8045	831	98.33%	29.66%
	4	0	0	0.00%	0.00%
	≥5	0	416	0.00%	14.85%
MFB-50	2	149	1400	1.77%	57.71%
	3	8260	756	98.23%	31.16%
	4	0	0	0.00%	0.00%
	≥5	0	270	0.00%	11.13%
MFB-75	2	2774	368	52.63%	9.18%
	3	2497	3639	47.37%	90.79%
	4	0	1	0.00%	0.02%
	≥5	0	0	0.00%	0.00%
MFB-100	2	726	3234	6.23%	71.64%
	3	10932	370	93.77%	8.20%
	4	0	0	0.00%	0.00%
	≥5	0	910	0.00%	20.16%

Table 8: Zero-shot experiment results on node and relation inductive knowledge graph datasets

Method	FB-25		FB-50		FB-75		FB-100		WK-25		WK-50		WK-75		WK-100	
	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10
ULTRA(3KG)	**0.388**	**0.640**	**0.338**	**0.543**	**0.403**	**0.604**	**0.449**	**0.642**	**0.316**	**0.532**	**0.166**	**0.324**	**0.365**	**0.537**	0.164	**0.286**
HYPER(3KG)	0.372	0.614	0.313	0.513	0.373	0.568	0.412	0.598	0.276	0.410	0.145	0.281	0.334	0.460	**0.171**	0.271
HYPER(4HG)	0.277	0.538	0.225	0.427	0.287	0.503	0.336	0.567	0.215	0.422	0.117	0.245	0.280	0.491	0.125	0.247
HYPER(3KG + 2HG)	0.382	0.635	0.326	0.535	0.389	0.598	0.434	0.632	0.281	0.428	0.158	0.280	0.365	0.522	0.160	0.280

Method	NL-25		NL-50		NL-75		NL-100		MT1-tax		MT1-health		MT2-org		MT2-sci	
	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10
ULTRA(3G)	**0.395**	**0.569**	**0.407**	**0.570**	**0.368**	**0.547**	0.471	0.651	0.224	0.305	0.298	0.374	**0.095**	**0.159**	**0.258**	0.354
HYPER(3KG)	0.321	0.550	0.350	0.520	0.320	0.483	0.415	0.627	**0.234**	0.306	**0.361**	**0.431**	0.088	0.142	0.256	0.339
HYPER(4HG)	0.214	0.431	0.226	0.480	0.252	0.455	0.333	0.618	0.200	0.274	0.266	0.358	0.063	0.116	0.195	0.320
HYPER(3KG + 2HG)	0.360	0.558	0.376	0.547	0.342	0.540	**0.473**	**0.685**	0.204	**0.396**	0.222	0.399	0.087	0.149	**0.258**	**0.428**

Method	MT3-art		MT3-infra		MT4-sci		MT4-health		Metafam		FBNELL		NL-0		Average	
	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10
ULTRA(3KG)	0.259	0.402	**0.619**	**0.755**	**0.274**	**0.449**	**0.624**	**0.737**	0.238	0.644	**0.485**	**0.652**	**0.342**	0.523	**0.345**	0.513
HYPER(3KG)	0.257	0.402	0.562	0.695	0.259	0.415	0.547	0.723	0.395	0.804	0.447	0.617	0.312	0.501	0.318	0.492
HYPER(4HG)	0.152	0.265	0.363	0.451	0.232	0.412	0.380	0.556	0.191	0.606	0.320	0.537	0.171	0.393	0.228	0.419
HYPER(3KG + 2HG)	**0.270**	**0.425**	0.573	0.716	0.270	0.441	0.560	0.724	**0.457**	**0.875**	0.450	0.639	0.334	**0.526**	0.336	**0.520**

Table 9: Zero-shot experiment results on node inductive knowledge graph datasets. The best result for each dataset is in **bold**.

Method	WN-v1		WN-v2		WN-v3		WN-v4		FB-v1		FB-v2	
	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10
ULTRA(3KG)	0.648	0.768	0.663	0.765	0.376	0.476	0.611	0.705	**0.498**	**0.656**	**0.512**	**0.700**
HYPER(3KG)	**0.703**	**0.799**	0.681	**0.788**	**0.400**	**0.522**	**0.644**	**0.721**	0.450	0.622	0.474	0.668
HYPER(4HG)	0.530	0.720	0.533	0.691	0.287	0.392	0.514	0.652	0.263	0.476	0.308	0.527
HYPER(3KG + 2HG)	0.702	0.782	**0.686**	0.785	0.385	0.503	0.640	0.710	0.454	0.648	0.480	0.695

Method	FB-v3		FB-v4		NL-v1		NL-v2		NL-v3		NL-v4	
	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10
ULTRA(3KG)	**0.491**	**0.654**	**0.486**	**0.677**	**0.785**	**0.913**	**0.526**	0.707	**0.515**	0.702	0.479	0.712
HYPER(3KG)	0.460	0.627	0.460	0.653	0.619	0.868	0.514	0.719	0.510	0.692	0.468	0.697
HYPER(4HG)	0.276	0.482	0.280	0.504	0.516	0.863	0.345	0.639	0.340	0.610	0.269	0.582
HYPER(3KG + 2HG)	0.466	0.648	0.460	0.663	0.570	0.719	0.521	**0.741**	0.509	**0.705**	**0.501**	**0.728**

Method	ILPC Small		ILPC Large		HM 1k		HM 3k		HM 5k		HM Indigo	
	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10
ULTRA(3KG)	**0.302**	0.443	0.290	**0.424**	**0.059**	0.092	**0.037**	0.077	**0.034**	0.071	**0.440**	**0.648**
HYPER(3KG)	0.291	0.438	**0.293**	0.412	0.046	0.092	0.036	0.073	0.033	0.069	0.437	0.644
HYPER(4HG)	0.169	0.347	0.183	0.327	0.027	0.075	0.024	0.064	0.024	0.058	0.298	0.484
HYPER(3KG + 2HG)	0.296	**0.448**	0.289	0.417	0.043	**0.106**	**0.037**	**0.092**	**0.034**	**0.086**	0.401	0.614

Table 10: Scalability comparison on FB15k-237 with batch size = 64.

Model	# Parameters	Training Time (s/batch)	Inference Time (s/batch)	GPU Memory (GB)
ULTRA	168,705	1.19	0.066	12.87
HCNet	159,297	2.64	0.156	18.03
HYPER	225,409	4.51	0.272	25.30

Table 11: Average zero-shot inference MRR over 16 newly proposed dataset comparison across KG-ICL†, ULTRA†, and HYPER variants.

Model	Average MRR
KG-ICL† (4 Layer)	0.139
KG-ICL† (5 Layer)	0.048
KG-ICL† (6 Layer)	0.143
ULTRA† (3KG)	0.142
ULTRA† (4KG)	0.150
ULTRA† (50KG)	0.154
HYPER (3KG)	0.161
HYPER (4HG)	0.182
HYPER (3KG+2HG)	**0.236**

Table 12: Training datasets for model variants

Model	Knowledge Hypergraph				Knowledge Graph				
	JF17K	FB-AUTO	Wikipeople	MFB15K	FB15k-237	WN18RR	CodEx Medium	NELL995	Others(46G)
ULTRA(3G)					✓	✓	✓		
ULTRA(4G)					✓	✓	✓	✓	
ULTRA(50G)					✓	✓	✓	✓	✓
HYPER(3KG)					✓	✓	✓		
HYPER(4HG)	✓	✓	✓	✓					
HYPER(3KG + 2HG)	✓		✓		✓	✓	✓		
HYPER(end2end) HCNet	Trained directly on target dataset's training graph								

Table 13: Arity distribution across node inductive datasets.

Dataset	Arity	Training Graph	Inference Graph	Training %	Inference %
JF-IND	2	264	9	4.28%	2.93%
	3	4586	216	74.36%	70.36%
	4	1317	82	21.36%	26.71%
	≥5	0	0	0.00%	0.00%
WP-IND	2	0	0	0.00%	0.00%
	3	3375	476	81.54%	86.39%
	4	764	75	18.46%	13.61%
	≥5	0	0	0.00%	0.00%
MFB-IND	2	0	0	0.00%	0.00%
	3	336733	7527	100.00%	100.00%
	4	0	0	0.00%	0.00%
	≥5	0	0	0.00%	0.00%

Table 14: Dataset statistics of inductive link prediction task with knowledge hypergraph.

Statistic	JF-IND	WP-IND	MFB-IND
# seen vertices	4,685	4,463	3,283
# train hyperedges	6,167	4,139	336,733
# unseen vertices	100	100	500
# relations	31	32	12
# max arity	4	4	3

Table 15: Dataset statistics of pretrained knowledge hypergraphs and knowledge graphs with respective arity.

Dataset	FB-AUTO	WikiPeople	JF17K	MFB15K	FB15k237	WN18RR	CoDEx-M
$\|V\|$	3,410	47,765	29,177	10,314	14541	40943	17050
$\|R\|$	8	707	327	71	237	11	51
# train	6,778	305,725	61,104	415,375	272115	86835	185584
# valid	2,255	38,223	15,275	39,348	17535	3034	10310
# test	2,180	38,281	24,915	38,797	20466	3134	10311
# max arity	5	9	6	5	2	2	2
# arity$=2$	3,786	337,914	56,322	82,247	310,116	93,003	206,205
# arity$=3$	0	25,820	34,550	400,027	0	0	0
# arity$=4$	215	15,188	9,509	26	0	0	0
# arity≥ 5	7,212	3,307	2,267	11,220	0	0	0

Table 16: Dataset statistics for inductive on both node and relation link prediction datasets. Triples are the number of edges given at training, validation, or test graphs, respectively, whereas Valid and Test denote triples to be predicted in the validation and test graphs.

Dataset	**Training Graph**			**Validation Graph**				**Test Graph**			
	Entities	**Rels**	**Triples**	**Entities**	**Rels**	**Triples**	**Valid**	**Entities**	**Rels**	**Triples**	**Test**
FB-25	5190	163	91571	4097	216	17147	5716	4097	216	17147	5716
FB-50	5190	153	85375	4445	205	11636	3879	4445	205	11636	3879
FB-75	4659	134	62809	2792	186	9316	3106	2792	186	9316	3106
FB-100	4659	134	62809	2624	77	6987	2329	2624	77	6987	2329
WK-25	12659	47	41873	3228	74	3391	1130	3228	74	3391	1131
WK-50	12022	72	82481	9328	93	9672	3224	9328	93	9672	3225
WK-75	6853	52	28741	2722	65	3430	1143	2722	65	3430	1144
WK-100	9784	67	49875	12136	37	13487	4496	12136	37	13487	4496
NL-0	1814	134	7796	2026	112	2287	763	2026	112	2287	763
NL-25	4396	106	17578	2146	120	2230	743	2146	120	2230	744
NL-50	4396	106	17578	2335	119	2576	859	2335	119	2576	859
NL-75	2607	96	11058	1578	116	1818	606	1578	116	1818	607
NL-100	1258	55	7832	1709	53	2378	793	1709	53	2378	793
Metafam	1316	28	13821	1316	28	13821	590	656	28	7257	184
FBNELL	4636	100	10275	4636	100	10275	1055	4752	183	10685	597
Wiki MT1 tax	10000	10	17178	10000	10	17178	1908	10000	9	16526	1834
Wiki MT1 health	10000	7	14371	10000	7	14371	1596	10000	7	14110	1566
Wiki MT2 org	10000	10	23233	10000	10	23233	2581	10000	11	21976	2441
Wiki MT2 sci	10000	16	16471	10000	16	16471	1830	10000	16	14852	1650
Wiki MT3 art	10000	45	27262	10000	45	27262	3026	10000	45	28023	3113
Wiki MT3 infra	10000	24	21990	10000	24	21990	2443	10000	27	21646	2405
Wiki MT4 sci	10000	42	12576	10000	42	12576	1397	10000	42	12516	1388
Wiki MT4 health	10000	21	15539	10000	21	15539	1725	10000	20	15337	1703

Table 17: Dataset statistics for inductive-e link prediction datasets. Triples are the number of edges given at training, validation, or test graphs, respectively, whereas Valid and Test denote triples to be predicted in the validation and test graphs.

Dataset	Rels	Training Graph		Validation Graph			Test Graph		
		Entities	Triples	Entities	Triples	Valid	Entities	Triples	Test
FB-v1	180	1594	4245	1594	4245	489	1093	1993	411
FB-v2	200	2608	9739	2608	9739	1166	1660	4145	947
FB-v3	215	3668	17986	3668	17986	2194	2501	7406	1731
FB-v4	219	4707	27203	4707	27203	3352	3051	11714	2840
WN-v1	9	2746	5410	2746	5410	630	922	1618	373
WN-v2	10	6954	15262	6954	15262	1838	2757	4011	852
WN-v3	11	12078	25901	12078	25901	3097	5084	6327	1143
WN-v4	9	3861	7940	3861	7940	934	7084	12334	2823
NL-v1	14	3103	4687	3103	4687	414	225	833	201
NL-v2	88	2564	8219	2564	8219	922	2086	4586	935
NL-v3	142	4647	16393	4647	16393	1851	3566	8048	1620
NL-v4	76	2092	7546	2092	7546	876	2795	7073	1447
ILPC Small	48	10230	78616	6653	20960	2908	6653	20960	2902
ILPC Large	65	46626	202446	29246	77044	10179	29246	77044	10184
HM 1k	11	36237	93364	36311	93364	1771	9899	18638	476
HM 3k	11	32118	71097	32250	71097	1201	19218	38285	1349
HM 5k	11	28601	57601	28744	57601	900	23792	48425	2124
HM Indigo	229	12721	121601	12797	121601	14121	14775	250195	14904

Table 18: Experiment result on node and relation inductive knowledge hypergraph datasets.

Method	JF-25				JF-50				JF-75				JF-100			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
G-MPNN	0.006	0.004	0.004	0.007	0.003	0.000	0.000	0.003	0.001	0.000	0.000	0.000	0.002	0.000	0.000	0.000
HCNet	0.011	0.004	0.007	0.011	0.009	0.000	0.000	0.024	0.069	0.038	0.072	0.125	0.028	0.000	0.018	0.054
HYPER(end2end)	0.202 ±.003	0.117 ±.002	0.226 ±.006	0.346 ±.005	**0.468** ±.004	**0.358** ±.007	0.540 ±.002	0.653 ±.003	0.207 ±.005	**0.125** ±.004	0.226 ±.006	0.357 ±.003	**0.198** ±.008	**0.107** ±.002	0.161 ±.004	0.411 ±.003
ULTRA†(3KG)(0-shot)	0.119	0.000	0.166	0.399	0.304	0.143	0.427	0.629	0.109	0.038	0.116	0.241	0.091	0.036	0.054	0.232
ULTRA†(4KG)(0-shot)	0.099	0.016	0.078	0.343	0.325	0.188	0.382	0.648	0.102	0.038	0.085	0.251	0.132	0.089	0.107	0.214
ULTRA†(50KG)(0-shot)	0.147	0.071	0.145	0.368	0.407	0.285	0.513	0.605	0.126	0.082	0.154	0.207	0.111	0.089	0.125	0.161
HYPER(3KG)(0-shot)	0.148	0.071	0.152	0.318	0.297	0.212	0.336	0.460	0.112	0.041	0.132	0.254	0.130	0.018	0.107	0.375
HYPER(4HG)(0-shot)	0.187	0.095	0.219	0.360	0.377	0.239	0.476	0.608	0.188	0.110	0.204	**0.370**	0.181	0.089	0.161	**0.464**
HYPER(3KG + 2HG)(0-shot)	0.216	0.122	**0.233**	**0.413**	0.455	0.325	**0.556**	0.664	**0.213**	0.122	0.231	0.367	0.173	0.071	**0.179**	0.446
HYPER(3KG + 2HG)(finetuned)	**0.217** ±.001	**0.131** ±.002	0.226 ±.004	0.389 ±.006	0.456 ±.003	0.331 ±.005	0.554 ±.002	**0.672** ±.001	0.209 ±.004	0.119 ±.006	**0.238** ±.003	0.361 ±.007	0.176 ±.005	0.089 ±.003	0.161 ±.008	0.393 ±.002

Method	WP-25				WP-50				WP-75				WP-100			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
G-MPNN	0.005	0.003	0.005	0.006	0.002	0.001	0.000	0.002	0.001	0.000	0.001	0.000	0.000	0.000	0.000	0.001
HCNet	0.104	0.048	0.114	0.230	0.050	0.025	0.059	0.087	0.019	0.010	0.020	0.032	0.003	0.000	0.000	0.003
HYPER(end2end)	0.159 ±.003	0.071 ±.004	**0.172** ±.005	0.358 ±.006	0.143 ±.002	0.082 ±.003	0.157 ±.004	0.260 ±.007	0.139 ±.003	0.072 ±.001	0.129 ±.005	0.294 ±.006	0.202 ±.002	0.106 ±.003	0.190 ±.004	0.418 ±.007
ULTRA†(3KG)(0-shot)	0.040	0.010	0.044	0.108	0.070	0.034	0.082	0.150	0.067	0.026	0.068	0.160	0.071	0.039	0.077	0.148
ULTRA†(4KG)(0-shot)	0.047	0.004	0.054	0.145	0.091	0.044	0.107	0.176	0.089	0.038	0.086	0.229	0.086	0.051	0.093	0.161
ULTRA†(50KG)(0-shot)	0.045	0.011	0.044	0.151	0.071	0.040	0.080	0.133	0.045	0.021	0.050	0.101	0.065	0.035	0.074	0.151
HYPER(3KG)(0-shot)	0.143	0.057	0.138	0.349	0.147	0.073	0.159	**0.327**	0.186	0.097	0.186	**0.391**	0.221	0.106	**0.241**	**0.498**
HYPER(4HG)(0-shot)	0.075	0.033	0.094	0.186	0.068	0.056	0.080	0.081	0.086	0.066	0.107	0.114	0.168	0.080	0.190	0.360
HYPER(3KG + 2HG)(0-shot)	0.132	0.058	0.151	0.296	0.152	0.086	0.178	0.295	0.192	0.107	**0.201**	0.384	**0.222**	**0.132**	0.209	0.453
HYPER(3KG + 2HG)(finetuned)	**0.169** ±.003	**0.078** ±.002	0.164 ±.004	**0.399** ±.005	**0.171** ±.001	**0.103** ±.006	**0.201** ±.002	0.306 ±.007	**0.194** ±.003	**0.112** ±.004	0.199 ±.002	0.375 ±.005	0.210 ±.006	0.116 ±.003	0.206 ±.004	0.424 ±.002

Method	WD-25				WD-50				WD-75				WD-100			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
G-MPNN	0.001	0.000	0.000	0.000	0.001	0.000	0.000	0.000	0.001	0.000	0.000	0.001	0.001	0.000	0.000	0.001
HCNet	0.086	0.050	0.096	0.136	0.043	0.027	0.044	0.060	0.015	0.006	0.023	0.030	0.007	0.004	0.005	0.007
HYPER(end2end)	0.215 ±.002	0.132 ±.006	0.225 ±.005	0.394 ±.004	0.205 ±.007	0.153 ±.003	0.219 ±.006	0.317 ±.002	**0.172** ±.003	**0.105** ±.004	**0.194** ±.003	**0.298** ±.007	0.205 ±.002	0.139 ±.008	0.226 ±.004	0.342 ±.005
ULTRA†(3KG)(0-shot)	0.171	0.096	0.175	0.351	0.201	0.137	0.208	0.339	0.149	0.088	0.157	0.266	0.176	0.118	0.186	0.298
ULTRA†(4KG)(0-shot)	0.094	0.073	0.096	0.142	0.141	0.104	0.148	0.224	0.054	0.043	0.061	0.076	0.075	0.063	0.081	0.102
ULTRA†(50KG)(0-shot)	0.062	0.063	0.063	0.063	0.124	0.115	0.131	0.148	0.104	0.073	0.120	0.170	0.150	0.114	0.167	0.233
HYPER(3KG)(0-shot)	0.167	0.010	0.172	0.331	0.158	0.104	0.175	0.295	0.123	0.005	0.142	0.255	0.146	0.077	0.168	0.281
HYPER(4HG)(0-shot)	0.087	0.076	0.093	0.103	0.158	0.142	0.164	0.197	0.057	0.051	0.062	0.064	0.165	0.139	0.177	0.233
HYPER(3KG + 2HG)(0-shot)	0.223	**0.156**	0.225	**0.404**	0.200	0.148	0.208	0.317	0.154	0.093	0.168	0.275	0.182	0.133	0.177	0.286
HYPER(3KG + 2HG)(finetuned)	**0.225** ±.003	0.146 ±.005	**0.245** ±.004	0.397 ±.006	**0.234** ±.002	**0.186** ±.007	**0.230** ±.003	**0.355** ±.001	0.166 ±.005	0.101 ±.003	0.189 ±.006	0.294 ±.004	**0.210** ±.002	**0.140** ±.008	**0.235** ±.003	**0.351** ±.005

Method	MFB-25				MFB-50				MFB-75				MFB-100			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
G-MPNN	0.002	0.000	0.000	0.004	0.004	0.001	0.003	0.007	0.007	0.003	0.004	0.010	0.003	0.000	0.002	0.005
HCNet	0.033	0.008	0.008	0.108	0.026	0.013	0.022	0.041	0.016	0.007	0.009	0.021	0.082	0.028	0.085	0.227
HYPER(end2end)	0.332 ±.005	0.221 ±.004	0.388 ±.003	0.533 ±.002	0.200 ±.006	0.105 ±.004	0.251 ±.007	0.374 ±.005	0.135 ±.003	0.070 ±.006	0.143 ±.002	0.255 ±.008	0.222 ±.001	0.169 ±.005	0.228 ±.004	0.317 ±.003
ULTRA†(3KG)(0-shot)	0.209	0.071	0.304	0.504	0.153	0.055	0.189	0.377	0.062	0.011	0.025	0.182	0.222	0.104	0.284	0.433
ULTRA†(4KG)(0-shot)	0.343	0.242	0.388	0.542	0.215	0.122	0.249	**0.398**	0.111	0.060	0.106	0.230	0.274	0.187	0.314	0.435
ULTRA†(50KG)(0-shot)	0.310	0.217	0.371	0.479	0.218	0.134	0.261	0.369	0.100	0.056	0.086	0.219	0.262	0.185	0.309	0.416
HYPER(3KG)(0-shot)	0.248	0.167	0.283	0.396	0.191	0.123	0.216	0.296	0.039	0.016	0.029	0.073	0.276	0.198	0.311	0.416
HYPER(4HG)(0-shot)	0.349	0.258	0.400	0.546	0.244	**0.169**	0.286	0.382	0.139	0.082	0.140	0.244	0.278	0.195	0.316	0.441
HYPER(3KG + 2HG)(0-shot)	**0.363**	**0.263**	**0.417**	**0.550**	**0.250**	0.167	**0.287**	0.393	0.140	0.077	0.140	0.260	**0.299**	**0.214**	**0.339**	0.449
HYPER(3KG + 2HG)(finetuned)	0.347 ±.004	0.229 ±.006	0.408 ±.003	0.533 ±.002	0.243 ±.005	0.163 ±.006	0.286 ±.007	0.391 ±.004	**0.158** ±.002	**0.088** ±.005	**0.161** ±.003	**0.302** ±.006	0.275 ±.002	0.197 ±.008	0.290 ±.003	**0.452** ±.004

Table 19: Experiment result on node-inductive knowledge hypergraph datasets.

Method	JF-IND			WP-IND			MFB-IND		
	MRR	H@1	H@3	MRR	H@1	H@3	MRR	H@1	H@3
HGNN	0.102	0.086	0.128	0.072	0.045	0.112	0.121	0.076	0.114
HyperGCN	0.099	0.088	0.133	0.075	0.049	0.111	0.118	0.074	0.117
G-MPNN	0.219	0.155	0.236	0.177	0.108	0.191	0.124	0.071	0.123
RD-MPNN	0.402	0.308	0.453	0.304	0.238	0.328	0.122	0.082	0.125
HCNet	0.435	0.357	0.495	0.414	0.352	0.451	0.368	0.223	0.417
HYPER(end2end)	0.422 ±.004	0.320 ±.006	0.483 ±.007	0.435 ±.005	0.367 ±.004	0.471 ±.006	0.427 ±.003	0.290 ±.005	0.499 ±.004
ULTRA†(3KG)(0-shot)	0.173	0.043	0.220	0.101	0.000	0.041	0.054	0.003	0.026
ULTRA†(4KG)(0-shot)	0.286	0.171	0.322	0.183	0.029	0.250	0.163	0.069	0.165
ULTRA†(50KG)(0-shot)	0.346	0.255	0.381	0.286	0.218	0.295	0.149	0.056	0.135
HYPER(3KG)(0-shot)	0.263	0.177	0.281	0.259	0.176	0.307	0.184	0.123	0.196
HYPER(4HG)(0-shot)	0.403	0.277	0.501	0.375	0.297	0.410	**0.497**	**0.351**	**0.582**
HYPER(3KG + 2HG)(0-shot)	0.459	0.365	0.515	0.415	0.338	0.454	0.404	0.267	0.480
HYPER(3KG + 2HG)(finetuned)	**0.463** ±.002	**0.373** ±.003	**0.517** ±.008	**0.446** ±.008	**0.379** ±.009	**0.482** ±.007	0.455 ±.003	0.318 ±.007	0.530 ±.005

Table 20: HYPER hyper-parameters for pretraining, fine-tuning, and end-to-end training.

	Hyperparameter	HYPER
Positional Interaction Encoder	# Layers	2
	Hidden dimension	64
	Dropout	0
	Activation	ReLU
Relation Encoder	# Layers T	6
	Hidden dimension	64
	Dropout	0
	Activation	ReLU
	Norm	LayerNorm
Entity Encoder	# Layers L	6
	Hidden dimension	64
	Dec	2-layer MLP
	Dropout	0
	Activation	ReLU
	Norm	LayerNorm
Pre-training	Optimizer	AdamW
	Learning rate	0.0005
	Training steps	30,000
	Adversarial temperature	1
	# Negatives	512
	Batch size	32
Fine-tuning	Optimizer	AdamW
	Learning rate	0.0005
	Adversarial temperature	1
	# Negatives	256
	Batch size	8
End-to-End	Optimizer	AdamW
	Learning rate	0.0005
	Adversarial temperature	1
	# Negatives	256
	Batch size	8

Table 21: Hyperparameters for fine-tuning and training end-to-end for HYPER.

Datasets	Finetune		End-to-End	
	Epoch	Batch per Epoch	Epoch	Batch per Epoch
JF 25-100	3	full	10	full
WP 25-100	3	full	10	full
MFB 25-100	3	full	10	full
WD 25-100	3	full	10	full
JF-IND	1	full	20	full
WP-IND	1	full	20	full
MFB-IND	1	2000	4	10000
FB 25-100	3	full	10	full
WK 25-100	3	full	10	full
NL 0-100	3	full	10	full
MT1-MT4	3	full	10	full
Metafam, FBNELL	3	full	10	full
FB v1-v4	1	full	10	full
WN v1-v4	1	full	10	full
NL v1-v4	3	full	10	full
ILPC Small	3	full	10	full
ILPC Large	1	1000	10	1000
HM 1k-5k, Indigo	1	100	10	1000