Zero-shot Transfer Learning on Heterogeneous Graphs via Knowledge Transfer Networks

Minji Yoon*[1], John Palowitch†, Dustin Zelle†, Ziniu Hu‡[2], Ruslan Salakhutdinov*, Bryan Perozzi†
*Carnegie Mellon University, ‡University of California Los Angeles, †Google Research
{minjiy,rsalakhu}@andrew.cmu.edu,bull@cs.ucla.edu,{palowitch,dzelle,hubris}@google.com

ABSTRACT

Industrial ecosystems are commonly represented as heterogeneous graphs (HG) composed of multiple node/edge types. Heterogeneous graph neural networks (HGNNs) are applied to HGs to learn deep context-informed node representations. However, many HG datasets from industrial applications suffer from label imbalance between node types. As there is no direct way to learn using labels rooted at different node types, HGNNs have been applied to only a few node types with abundant labels. In this work, we propose a zero-shot transfer learning module for HGNNs called a Knowledge Transfer Network (KTN) that transfers knowledge from *label-abundant* node types to *zero-labeled* node types through rich relational information given in the HG.

KEYWORDS

Heterogeneous Graph Neural Networks; Transfer Learning

1 INTRODUCTION

To learn powerful features representing the complex multimodal structure of heterogeneous graphs (HG), various heterogeneous graph neural networks (HGNN) have been proposed [5, 12, 15, 17]. A common issue in industrial applications of HGNNs is the label imbalance among different node types. For instance, publicly available *content* nodes – such as those representing video, text, and image content – are abundantly labeled, whereas labels for other types – such as *user* or *account* nodes – may be much more expensive to collect (e.g., due to privacy restrictions). This means that in most standard settings, HGNN models can only learn to make good inferences for a few label-abundant node types, and can usually not make any inferences for the remaining node types, given the absence of any labels for them.

If there is a pair of *label-abundant* and *zero-labeled* node types that share an inference task, could we transfer knowledge between them? One body of work has focused on transferring knowledge between nodes of the *same* type from two *different* HGs (i.e., graph-to-graph transfer learning) [6, 16]. However, these approaches are not applicable in many real-world scenarios. First, any large-scale external HG that could be used in a graph-to-graph transfer learning setting would almost surely be proprietary. Second, even if practitioners could obtain access to an external industrial HG, it is unlikely the distribution of that (source) graph would match their target graph well enough to apply transfer learning. Finally, node types suffering label scarcity are likely to suffer the same issue on other HGs (e.g., user nodes).

In this paper, we introduce a zero-shot transfer learning approach for a *single* HG (assumed to be fully owned by the practitioners), transferring knowledge from labelled to unlabeled node types. This setting is distinct from any graph-to-graph transfer learning scenarios since the source and target domains exist in the same HG dataset and are assumed to have different node types. Our model utilizes the shared context between source and target node types; for instance, in an e-commerce network, the latent (unknown) labels of user nodes can be strongly correlated with spending/reviewing patterns that are encoded in the cross-edges between user nodes and product/review nodes. We propose a novel zero-shot transfer learning problem for this HG learning setting as follows:

Problem Definition 1. **Zero-shot transfer learning for cross-type inference on a HG:** *Given a HG with node types* $\{\mathbf{s}, \mathbf{t}, \cdots\}$ *with abundant labels for source type* $\mathbf{s}$ *but no labels for target type* $\mathbf{t}$*, can we train HGNNs to infer the labels of target-type nodes?*

To solve this problem, we first analyze that HGNNs learn entirely different *feature extractors* for nodes and edges of different types and present some empirical consequences. Then we theoretically show how feature extractors across node types relate to each other and how their output distributions could be represented in terms of each other. We model this theoretical relationship between two feature extractors as a Knowledge Transfer Network (KTN), which can be optimized to transform target embeddings to fit the source domain distribution. Our main contributions are:

- **Novel and practical problem definition:** To the best of our knowledge, KTN is the first cross-type transfer learning method designed for heterogeneous graphs.
- **Generality:** KTN is a principled approach analytically induced from the architecture of HGNNs, thus applicable to any HGNN models, showing up to 960% performance improvement for zero-labeled node inference across 6 different HGNN models.
- **Effectiveness:** We show that KTN outperforms state-of-the-art transfer learning methods, being up to 73.3% higher in MRR on 8 different transfer learning tasks on HGs.

2 RELATED WORK

Here we cover two classes of learning approaches that are related to KTN, but are ultimately distinct and fall short of a fully-rigorous approach to zero-shot cross-type inference in HGs.

Zero-shot domain adaptation (DA) transfers knowledge from a source domain with abundant labels to a target domain that follows

[1,2] Work done while interning at Google.

DLG'22, August 15, 2022, Washington, DC, USA

a similar distribution to the source domain but lacks labels. MMD-based DA methods [8, 10, 14] minimize the maximum mean discrepancy (MMD) [4] between the mean embeddings of two distributions in reproducing kernel Hilbert space. Adversarial DA methods [2, 9] learn domain-invariant features by a min-max game between the domain classifier and the feature extractor. Optimal transport-based DA methods [13] estimate the empirical Wasserstein distance [11] between two domains and minimize the distance in an adversarial manner. Due to the assumption that source and target domains have the same modality, the standard DA setting assumes identical feature extractors across source and target domains.

Label propagation (LP) [19] passes each node's labels to their neighbors according to normalized edge weights. LP relies on only a graph's edges and is therefore easily applied to a heterogeneous graph – labels are simply propagated across edges, regardless of type. In this paper, we also evaluate a similarly-simple baseline, embedding propagation (EP). Similar to LP, EP recursively propagates source embeddings (computed using source labels) until they reach the target nodes. EP exploits both node attribute information and the node adjacencies but only uses the source node embeddings.

3 PRELIMINARIES

In this section, we review heterogeneous graphs and heterogeneous graph neural networks (HGNNs).

3.1 Heterogeneous graph

Heterogeneous graphs (HGs) are an important abstraction for modeling the relational data of multi-modal systems. Formally, a heterogeneous graph is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{T}, \mathcal{R})$ where the node set $\mathcal{V}$; the edge set $\mathcal{E}$; the set of node types $\mathcal{T}$ with associated map $\tau : \mathcal{V} \mapsto \mathcal{T}$; the set of relation types $\mathcal{R}$ with associated map $\phi : \mathcal{E} \mapsto \mathcal{R}$. We additionally assume the existence of a node attribute vector $x_i \in \mathcal{X}_{\tau(i)}$ for each $i \in \mathcal{V}$, where $\mathcal{X}_t$ is an attribute space specific to nodes of type t . This allows $\mathcal{G}$ to represent nodes with different attribute modalities such as images, text, or booleans.

3.2 Heterogeneous Graph Neural Networks

Various HGNN models have been proposed [5, 12, 15, 17]. Fully-specified HGNN models have specialized parameters for each node type [5], edge type [12], and meta-path type [1] to most effectively utilize the complex relationships encoded in the HG data structure. In this paper, we use a flavor of HGNN known as a Heterogeneous Message-Passing Neural Network (HMPNN) as our base model on which to demonstrate KTN (though KTN can be implemented in almost any HGNN, as we show in experiments in Section 6). The HMPNN merely extends the standard MPNN [3] by specializing all transformation and message matrices in each node/edge type. With its generality, HMPNN is itself a base model for RGCN [12] and HGT [5], and is also widely used as a default HGNN model in popular GNN libraries (e.g., pyG, TF-GNN, DGL).

In a HMPNN, for any node j, the embedding of node j at the *l-th* layer is obtained with the following generic formulation:

$$h_j^{(l)} = \textbf{Transform}^{(l)}\left(\textbf{Aggregate}^{(l)}(\mathcal{E}(j))\right) \tag{1}$$

where $\mathcal{E}(j) = \{(i, j) \in \mathcal{E} : i, j \in \mathcal{V}\}$ denotes all the edges which connect (directionally) to j. The above operations typically involve type-specific parameters to exploit the inherent multiplicity of modalities in heterogeneous graphs. First, we define a linear **Message** function:

$$\textbf{Message}^{(l)}(i, j) = M^{(l)}_{\phi((i,j))} \cdot \left(h_i^{(l-1)} \parallel h_j^{(l-1)}\right) \tag{2}$$

where $M_r^{(l)}$ are the specific message passing parameters for each $r \in \mathcal{R}$ and each of L GNN layers. Then defining $\mathcal{E}_r(j)$ as the set of edges of type r pointing to node j, the **Aggregate** function mean-pools messages by edge type, and concatenates:

$$\textbf{Aggregate}^{(l)}(\mathcal{E}(j)) = \Big\|_{r\in\mathcal{R}} \frac{1}{|\mathcal{E}_r(j)|} \sum_{e\in\mathcal{E}_r(j)} \textbf{Message}^{(l)}(e) \tag{3}$$

Finally, the **Transform** function maps the message into a type-specific latent space:

$$\textbf{Transform}^{(l)}(j) = \alpha(W^{(l)}_{\tau(j)} \cdot \textbf{Aggregate}^{(l)}(\mathcal{E}(j))) \tag{4}$$

By stacking L layers, HMPNN outputs highly contextualized final node representations, and the final node representations can be fed into another model to perform downstream heterogeneous network tasks, such as node classification or link prediction.

4 CROSS-TYPE FEATURE EXTRACTOR TRANSFORMATIONS IN HGNNS

We define $f_t : \mathcal{G} \mapsto \mathbb{R}^d$ to be the "feature extractor" of an HGNN, which is composed of parameters participating in mapping input node attributes of type t into a shared feature space $\mathbb{R}^d$. In this section, we derive a strict transformation between feature extractors in HMPNNs and use that expression to inspire a trainable transfer learning module called KTN in the following section. For simplicity, throughout this section we ignore the activation $\alpha(\cdot)$ within the **Transform** function in Equation (4).

4.1 Feature extractors in HMPNNs

We first reason intuitively about the differences between f_s and f_t when $s \neq t$, using a toy heterogeneous graph shown in Figure 1(a). Consider nodes v_1 and v_2, noticing that $\tau(1) \neq \tau(2)$. Using HMPNN's equations (2)-(4) from Section 3.2, for any $l \in \{0, \ldots, L-1\}$ we have

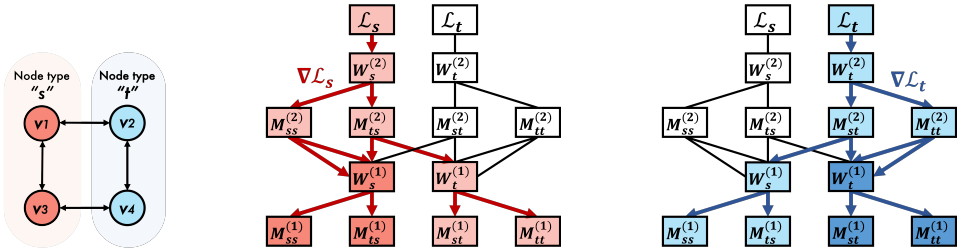
$$h_1^{(l)} = W_s^{(l)}\left[M_{ss}^{(l)}\left(h_3^{(l-1)} \parallel h_1^{(l-1)}\right) \parallel M_{ts}^{(l)}\left(h_2^{(l-1)} \parallel h_1^{(l-1)}\right)\right] \tag{5}$$

$$h_2^{(l)} = W_t^{(l)}\left[M_{st}^{(l)}\left(h_1^{(l-1)} \parallel h_2^{(l-1)}\right) \parallel M_{tt}^{(l)}\left(h_4^{(l-1)} \parallel h_2^{(l-1)}\right)\right] \tag{6}$$

where $h_j^{(0)} = x_j$. From these equations, we see that $h_1^{(l)}$ and $h_2^{(l)}$, which are features of different node types, are extracted using *disjoint* sets of model parameters at l-th layer. In a 2-layer HMPNN, this creates unique gradient backpropagation paths between the two node types, as illustrated in Figures 1(b)-1(c). In other words, even though the same HMPNN is applied to node types s and t, the feature extractors f_s and f_t have different computational paths. Therefore they project node features into different latent spaces, and have different update equations during training.

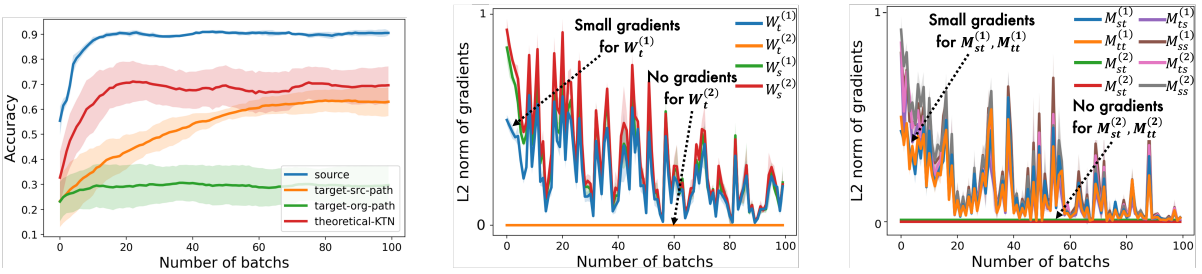
4.2 Empirical gap between f_s and f_t

Here we study the experimental consequences of the above observation via simulation. We first construct a synthetic graph extending the 2-type graph in Figure 1(a) to have multiple nodes per-type



(a) Toy graph (b) Gradient path for feature extractor f_s (c) Gradient path for feature extractor f_t

Figure 1: Illustration of a toy heterogeneous graph and the gradient paths for feature extractors f_s and f_t. Colored arrows in figures (b) and (c) show that the same HGNN nonetheless produces different gradient paths for each feature extractor. The color density of each box in (b) and (c) is proportional to the degree of participation of the corresponding parameter in each feature extractor.



(a) Test accuracy across various feature extractors (b) L2 norms of gradients of $W_{\tau(\cdot)}$ (c) L2 norms of gradients of $M_{\phi(\cdot)}$

Figure 2: An HGNN trained on a source domain underfits a target domain even on a "nice" heterogeneous graph. (a) Performance on the simulated heterogeneous graph for 4 kinds of feature extractors (*source*: source extractor f_s on source domain, *target-src-path*: source extractor f_s on target domain, *target-org-path*: target extractor f_t on target domain, and *theoretical-KTN*: target extractor f_t on target domain using KTN). (b-c) L2 norms of gradients of parameters $W_{\tau(\cdot)}$ and $M_{\phi(\cdot)}$ in the HGNN.

and multiple classes. To maximize the effects of having different feature extractors, we sample source and target nodes from the same feature distributions, and each class is well-separated in both the graph and feature space.

On such a well-aligned heterogeneous graph, there may seem to be no need for transfer learning from f_t to f_s. However, when we train the HMPNN model solely on s-type nodes, as shown in Figure 2(a) we find the test accuracy for s-type nodes to be high (90%, blue line), and the test accuracy for t-type nodes to be quite low (25%, green line). Now, if, instead, we make the t-type nodes use the source feature extractor f_s, much more transfer learning is possible (~65%, orange line). This shows that the different feature extractors present in the HMPNN model result in a significant performance drop, and simply matching input data distributions can not solve the problem.

To analyze this phenomenon at the level of backpropagation, in Figures 2(b)-2(c) we show the magnitude of gradients passed to parameters of each node types. We find that the final-layer parameters for type-t nodes ($W_t^{(2)}, M_{st}^{(2)}$ and $M_{tt}^{(2)}$) have zero gradients, and the first-layer parameters for t-type nodes ($W_t^{(1)}, M_{st}^{(1)}$ and $M_{tt}^{(1)}$) have much smaller gradients than their s-type counterparts ($W_s^{(1)}, M_{ts}^{(1)}$ and $M_{ss}^{(1)}$). This is because they contribute to f_s less than f_t.

This case study shows that even when an HGNN is trained on a relatively simple, balanced, and class-separated heterogeneous graph, a model trained only on the source domain node type cannot transfer to the target domain node type.

4.3 Relationship between feature extractors

Here, we derive a strict transformation between f_s and f_t, which will motivate the core transfer learning component of our proposed KTN model.

Theorem 4.1. *Given a heterogeneous graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{T}, \mathcal{R}\}$. For any layer $l > 0$, define the set of $(l-1)$-th layer HMPNN parameters as*

$$Q^{(l-1)} = \{M_r^{(l-1)} : r \in \mathcal{R}\} \cup \{W_t^{(l-1)} : t \in \mathcal{T}\}. \tag{7}$$

Let A be the total $n \times n$ adjacency matrix. Then for any $s, t \in \mathcal{T}$ there exist matrices $A_{ts}^* = a_{ts}(A)$ and $Q_{ts}^* = q_{ts}(Q^{(l-1)})$ such that

$$H_s^{(l)} = A_{ts}^* H_t^{(l)} Q_{ts}^* \tag{8}$$

where $a_{ts}(\cdot)$ and $q_{ts}(\cdot)$ are matrix functions that depend only on s, t.

Sketch of Proof. f_s and f_t are built inside one HMPNN model and interchange intermediate feature embeddings with each other. Both $H_s^{(l)}$ and $H_t^{(l)}$ are computed using the previous layer's intermediate embeddings $H_s^{(l-1)}, H_t^{(l-1)}$, and any other connected node type embeddings $H_x^{(l-1)}$ at the l-th HMPNN layer. Therefore $H_s^{(l)}$ and $H_t^{(l)}$ can be mathematically presented by each other using the $(l-1)$-th layer embeddings as connecting points. Using this intuition and HMPNN's equations, we can easily prove Theorem 4.1.

Notice that in Equation 8, Q^*_{ts} acts as a mapping matrix that maps $H^{(L)}_t$ into the source domain, then A^*_{ts} aggregates the mapped embeddings into s-type nodes. To examine the implications, we run the same experiment as described in Section 4.2, while this time mapping the target features $H^{(L)}_t$ into the source domain by multiplying with Q^*_{ts} in Equation 8 before passing over to a task classifier. We see via the red line in Figure 2(a) that, with this mapping, the accuracy in the target domain becomes much closer to the accuracy in the source domain (~70%). Thus, we use this theoretical transformation as a foundation for our trainable HGNN transfer learning module.

5 KTN: TRAINABLE CROSS-TYPE TRANSFER LEARNING MODULE FOR HGNNS

We begin by noting Equation 8 in Theorem 4.1 has a similar form to a single-layer graph convolutional network [7] with a deterministic transformation matrix (Q^*_{ts}) and a combination of adjacency matrices directing from target node type t to source node type s (A^*_{ts}). Instead of hand-computing the mapping function Q^*_{ts} for arbitrary HGs and HGNNs, we *learn* the mapping function by modelling Equation 8 as a trainable graph convolutional network, named the Knowledge Transfer Network, $\mathbf{t}_{\text{KTN}}(\cdot)$. KTN replaces Q^*_{ts} and A^*_{ts} in Equation 8 as follows:

$$\mathbf{t}_{\text{KTN}}(H^{(L)}_t) = A_{ts} H^{(L)}_t T_{ts} \tag{9}$$

$$\mathcal{L}_{\text{KTN}} = \left\| H^{(L)}_s - \mathbf{t}_{\text{KTN}}(H^{(L)}_t) \right\|_2 \tag{10}$$

where A_{ts} is an adjacency matrix from node type t to s, and T_{ts} is a trainable transformation matrix. By minimizing $\mathcal{L}_{\text{KTN}}$, T_{ts} is optimized to a mapping function of the target domain into the source domain.

5.1 Algorithm

We minimize a classification loss $\mathcal{L}_{\text{CL}}$ and a transfer loss $\mathcal{L}_{\text{KTN}}$ jointly with regard to a HGNN model $\mathbf{f}$, a classifier $\mathbf{g}$, and a knowledge transfer network $\mathbf{t}_{\text{KTN}}$ as follows:

$$\min_{\mathbf{f},\, \mathbf{g},\, \mathbf{t}_{\text{KTN}}} \mathcal{L}_{\text{CL}}(\mathbf{g}(\mathbf{f}(X_s)), Y_s) + \lambda \left\| \mathbf{f}(X_s) - \mathbf{t}_{\text{KTN}}(\mathbf{f}(X_t)) \right\|_2$$

where λ is a hyperparameter regulating the effect of $\mathcal{L}_{\text{KTN}}$. During a training step on the source domain, after computing the node embeddings $H^{(L)}_s$ and $H^{(L)}_t$, we map $H^{(L)}_t$ to the source domain using $\mathbf{t}_{\text{KTN}}$ and compute $\mathcal{L}_{\text{KTN}}$. Then, we update the models using gradients of $\mathcal{L}_{\text{CL}}$ (computed using only source labels) and $\mathcal{L}_{\text{KTN}}$. During the test phase on the target domain, after we get node embeddings $H^{(L)}_t$ from the trained HGNN model, we map $H^{(L)}_t$ into the source domain using the trained transformation matrix T_{ts}. Finally we pass the transformed target embeddings ($H^{(L)}_t T_{ts}$) into the classifier which was trained on the source domain.

6 EXPERIMENTS

6.1 Datasets

Open Academic Graph (OAG) [18] is composed of five types of nodes: papers, authors, institutions, venues, fields, and their corresponding relationships. Paper and author nodes have text-based attributes, while institution, venue, and field nodes have text- and graph structure-based attributes. Paper, author, and venue nodes are labeled with research fields in two hierarchical levels, L1 and L2.

6.2 Baselines

We compare KTN with 6 *Zero-shot DA* methods (DAN [8], JAN [10], DANN [2], CDAN [9], CDAN-E [9], and WDGRL [13]), and two traditional graph mining methods (LP and EP [19]).

6.3 Zero-shot transfer learning

We run 8 different zero-shot transfer learning tasks on the OAG graph. We run each experiment 3 times and report the average value. Each heterogeneous graph has node classification tasks for both source and target node types. Only source node types have labels, while target node types have none during training. The performance is evaluated by NDCG and MRR — widely adopted ranking metrics [5]. We use HMPNN as a base HGNN model.

In Table 1, our proposed method KTN consistently outperforms all baselines on all tasks by up to 73.3% higher in MRR. When we compare with the base accuracy using the model pretrained on the source domain without any transfer learning (3rd column, *Base*), the results are even more impressive. We see our method KTN provides relative gains of up to 340% higher MRR without using any labels from the target domain. These results show the clear effectiveness of KTN on zero-shot transfer learning tasks on a heterogeneous graph.

6.4 Generality of KTN

Here, we use 6 different HGNN models, R-GCN [12], HAN [15], HGT [5], MAGNN [1], MPNN [3], and HMPNN. MPNN maps all node types to the shared embedding space using projection matrices at the beginning and then applies MPNN layers designed for homogeneous graphs. In Table 2, KTN improves accuracy on the target nodes across all HGNN models by up to 960%. This shows the strong generality of KTN.

7 CONCLUSION

In this work, we proposed the first cross-type zero-shot transfer learning method for heterogeneous graphs. Our method, Knowledge Transfer Networks (KTN) for Heterogeneous Graph Neural Networks, transfers knowledge from *label-abundant* node types to *label-scarce* node types. We illustrate KTN handily improves HGNN performances up to 960% for zero-labeled node types across 6 different HGNN models and outperforms many challenging baselines up to 73% higher in MRR.

REFERENCES

[1] Xinyu Fu, Jiani Zhang, Ziqiao Meng, and Irwin King. 2020. Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding. In *Proceedings of The Web Conference 2020*. 2331–2341.

[2] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *The journal of machine learning research* 17, 1 (2016), 2096–2030.

[3] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *International conference on machine learning*. PMLR, 1263–1272.

Table 1: Open Academic Graph on Computer Science field. The gain column shows the relative gain of our method over using no transfer learning (*Base* column). o.o.m denotes *out-of-memory* errors.

Task	Metric	Base	DAN	JAN	DANN	CDAN	CDAN-E	WDGRL	LP	EP	KTN (gain)
P-A (L1)	NDCG	0.399	0.452	0.405	0.292	0.262	0.261	0.260	0.178	0.425	**0.623 (56%)**
	MRR	0.297	0.361	0.314	0.179	0.129	0.111	0.138	0.041	0.363	**0.629 (112%)**
A-P (L1)	NDCG	0.401	0.566	0.598	0.294	0.364	0.246	0.195	0.153	0.557	**0.733 (83%)**
	MRR	0.318	0.508	0.544	0.229	0.270	0.090	0.047	0.022	0.507	**0.711 (123%)**
A-V (L1)	NDCG	0.459	0.457	0.470	0.382	0.346	0.359	0.403	0.207	0.461	**0.671 (46%)**
	MRR	0.364	0.413	0.458	0.341	0.205	0.253	0.327	0.011	0.389	**0.698 (92%)**
V-A (L1)	NDCG	0.283	0.443	0.435	0.242	0.372	0.418	0.272	0.153	0.154	**0.584 (107%)**
	MRR	0.133	0.365	0.345	0.094	0.241	0.444	0.144	0.006	0.006	**0.586 (340%)**
P-A (L2)	NDCG	0.229	0.230	o.o.m	0.239	o.o.m	o.o.m	0.168	o.o.m	0.215	**0.282 (23%)**
	MRR	0.121	0.118	o.o.m	0.140	o.o.m	o.o.m	0.020	o.o.m	0.143	**0.2248 (86%)**
A-P (L2)	NDCG	0.197	0.162	o.o.m	0.204	0.158	0.161	0.132	o.o.m	0.208	**0.287 (46%)**
	MRR	0.095	0.052	o.o.m	0.106	0.032	0.045	0.017	o.o.m	0.132	**0.242 (155%)**
A-V (L2)	NDCG	0.347	0.329	0.295	0.325	0.288	0.273	0.289	o.o.m	0.297	**0.402 (16%)**
	MRR	0.310	0.296	0.198	0.223	0.128	0.097	0.110	o.o.m	0.227	**0.399 (29%)**
V-A (L2)	NDCG	0.235	0.249	0.251	0.214	0.197	0.205	0.217	o.o.m	0.119	**0.252 (7%)**
	MRR	0.129	0.157	0.161	0.090	0.044	0.068	0.085	o.o.m	0.000	**0.166 (28%)**

Table 2: KTN on different HGNN models. *Source* column shows accuracy for source node types. *Base* and *KTN* columns show accuracy for target node types without/with using KTN, respectively. *Gain* column shows the relative gain of our method over using no transfer learning.

		P-A (L1)				A-P (L1)			
HGNN type	Metric	Source	Base	KTN	Gain	Source	Base	KTN	Gain
R-GCN	NDCG	0.765	0.337	0.577	**71.12%**	0.648	0.388	0.647	**66.82%**
	MRR	0.757	0.236	0.587	**148.73%**	0.623	0.270	0.611	**126.18%**
HAN	NDCG	0.476	0.179	0.520	**190.56%**	0.515	0.182	0.512	**181.33%**
	MRR	0.416	0.047	0.497	**960.55%**	0.509	0.055	0.527	**850.90%**
HGT	NDCG	0.757	0.294	0.574	**95.07%**	0.670	0.283	0.581	**104.83%**
	MRR	0.749	0.178	0.563	**216.17%**	0.670	0.149	0.565	**279.52%**
MAGNN	NDCG	0.657	0.463	0.574	**24.01%**	0.676	0.557	0.622	**11.68%**
	MRR	0.631	0.378	0.556	**47.33%**	0.680	0.509	0.592	**16.14%**
MPNN	NDCG	0.602	0.443	0.590	**33.11%**	0.646	0.307	0.621	**101.92%**
	MRR	0.572	0.319	0.575	**80.10%**	0.660	0.145	0.595	**311.42%**
HMPNN	NDCG	0.789	0.399	0.623	**56.14%**	0.671	0.401	0.733	**82.88%**
	MRR	0.777	0.297	0.629	**111.86%**	0.661	0.318	0.711	**123.30%**

[4] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. 2012. A kernel two-sample test. *The Journal of Machine Learning Research* 13, 1 (2012), 723–773.
[5] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous graph transformer. In *Proceedings of The Web Conference 2020*. 2704–2710.
[6] Tiancheng Huang, Ke Xu, and Donglin Wang. 2020. DA-HGT: Domain Adaptive Heterogeneous Graph Transformer. *arXiv preprint arXiv:2012.05688* (2020).
[7] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
[8] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. 2015. Learning transferable features with deep adaptation networks. In *International conference on machine learning*. PMLR, 97–105.
[9] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. 2017. Conditional adversarial domain adaptation. *arXiv preprint arXiv:1705.10667* (2017).
[10] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. 2017. Deep transfer learning with joint adaptation networks. In *International conference on machine learning*. PMLR, 2208–2217.
[11] Ievgen Redko, Amaury Habrard, and Marc Sebban. 2017. Theoretical analysis of domain adaptation with optimal transport. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 737–753.
[12] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European semantic web conference*. Springer, 593–607.
[13] Jian Shen, Yanru Qu, Weinan Zhang, and Yong Yu. 2018. Wasserstein distance guided representation learning for domain adaptation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.
[14] Baochen Sun, Jiashi Feng, and Kate Saenko. 2016. Return of frustratingly easy domain adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 30.
[15] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous graph attention network. In *The World Wide Web Conference*. 2022–2032.
[16] Shuwen Yang, Guojie Song, Yilun Jin, and Lun Du. 2021. Domain adaptive classification on heterogeneous information networks. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*. 1410–1416.
[17] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V Chawla. 2019. Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 793–803.
[18] Fanjin Zhang, Xiao Liu, Jie Tang, Yuxiao Dong, Peiran Yao, Jie Zhang, Xiaotao Gu, Yan Wang, Bin Shao, Rui Li, et al. 2019. Oag: Toward linking large-scale heterogeneous entity graphs. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2585–2595.
[19] Xiaojin Zhu. 2005. *Semi-supervised learning with graphs*. Carnegie Mellon University.