Contrastive Pre-Training of GNNs on Heterogeneous Graphs

Xunqiang Jiang
Beijing University of Posts and Telecommunications
China
skd621@bupt.edu.cn

Yuanfu Lu
WeChat Search Application Department, Tencent Inc.
China
luyfroot@gmail.com

Yuan Fang
Singapore Management University
Singapore
yfang@smu.edu.sg

Chuan Shi*
Beijing University of Posts and Telecommunications
China
shichuan@bupt.edu.cn

ABSTRACT

While graph neural networks (GNNs) emerge as the state-of-the-art representation learning methods on graphs, they often require a large amount of labeled data to achieve satisfactory performance, which is often expensive or unavailable. To relieve the label scarcity issue, some pre-training strategies have been devised for GNNs, to learn transferable knowledge from the universal structural properties of the graph. However, existing pre-training strategies are only designed for homogeneous graphs, in which each node and edge belongs to the same type. In contrast, a heterogeneous graph embodies rich semantics, as multiple types of nodes interact with each other via different kinds of edges, which are neglected by existing strategies. In this paper, we propose a novel Contrastive Pre-Training strategy of GNNs on Heterogeneous Graphs (CPT-HG), to capture both the semantic and structural properties in a self-supervised manner. Specifically, we design semantic-aware pre-training tasks at both the relation- and subgraph-levels, and further enhance their representativeness by employing contrastive learning. We conduct extensive experiments on three real-world heterogeneous graphs, and promising results demonstrate the superior ability of our CPT-HG to transfer knowledge to various downstream tasks via pre-training.

CCS CONCEPTS

• **Information systems** → **Data mining**.

KEYWORDS

Pre-training, heterogeneous graph, self-supervised learning

ACM Reference Format:
Xunqiang Jiang, Yuanfu Lu, Yuan Fang, and Chuan Shi*. 2021. Contrastive Pre-Training of GNNs on Heterogeneous Graphs. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21), November 1–5, 2021, Virtual Event, QLD, Australia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3459637.3482332

*corresponding author.

CIKM '21, November 1–5, 2021, Virtual Event, QLD, Australia

ACM ISBN 978-1-4503-8446-9/21/11…$15.00
https://doi.org/10.1145/3459637.3482332

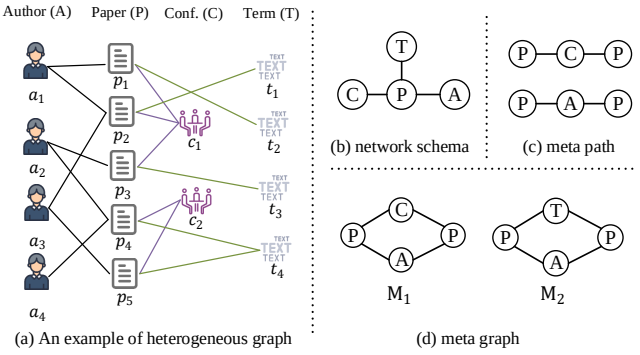


Figure 1: A toy example of heterogeneous graph for bibliographic data.

1 INTRODUCTION

In recent years, graphs have become a powerful abstraction for representing a wide variety of real-world datasets [41, 48, 50]. As an emerging tool for performing machine learning on graph-structured data, graph neural networks (GNNs) learn powerful graph representations by recursively aggregating content (*i.e.,* features or embeddings) from neighboring nodes, thus preserving both content and structure information. They have been demonstrated to improve the performance in various graph applications, such as node and graph classification [10, 18], recommendation systems [7, 45] and graph generation [21]. Generally, a GNN model is trained with (semi-)supervised information in an end-to-end manner, requiring a large volume and variety of labeled data for different downstream tasks. However, in most real-world scenarios, abundant labeled data are usually expensive and even infeasible to obtain, while there exist a large amount of unlabeled data that is easily accessible.

To make full use of the unlabeled graph-structured data, some recent effort takes inspiration from pre-training techniques in natural language processing [5, 20] and computer vision [3, 11], and propose to pre-train a GNN model with self-supervised information in a graph [13, 14, 28]. Broadly, we classify them into two categories according to their training corpus: one is pre-training on multiple graphs and then fine-tuning on a new unseen graph for downstream tasks [13, 28], the other is pre-training on a part

of a large-scale graph while transferring learned knowledge to downstream tasks on the remaining part [14]. For instance, Hu *et al.* [13] present some GNN pre-training strategies for the setting of multiple graphs, which utilizes both node-level and graph-level self-supervised information in a series of graphs (*e.g.,* molecular structures in the biochemical domain). On the other hand, GPT-GNN [14] introduces a self-supervised attributed graph generation task to pre-train a GNN, which learns transferable knowledge some parts of a large graph to facilitate the downstream tasks on other parts of the graph with only a few labels. Although these GNN pre-training methods achieve promising performance, all of them are only designed for homogeneous graphs, in which each node or edge belongs to the same type. In contrast, the so-called heterogeneous graphs [31, 33, 34], where multiple types of nodes interact with each other via different kinds of edges, are neglected by existing strategies.

Objects in a complex interaction system can often be organized into heterogeneous graphs [31, 34], which embody rich semantics and distinct structures resulting from multiple types of nodes and edges. As shown in Figure 1(a), a toy heterogeneous graph is constructed for bibliographic data, which consists of nodes of Author, Paper, Conference and Term types, as well as edges of Author-Paper, Paper-Conference and Paper-Term types. Different types of nodes or edges often exhibit varying network properties such as degree and clustering coefficient [23]. For instance, Conference nodes generally have higher degrees than Author nodes. Moreover, the heterogeneity gives rise to more complex semantic contexts that involve multi-party relationships among a number of nodes, such as $\{p_1, a_1, c_1, p_2\}$, which describes the semantic context of "two papers on similar topics from the same author". Beyond the toy example in bibliographic data, heterogeneous graphs are also ubiquitous in a wide range of domains, such as in e-commerce with users, products, brands and shops interacting in various ways, and in biology where diseases, proteins and drugs relating to each other. Given their prevalence and expressiveness, designing effective GNN pre-training strategies for heterogeneous graphs becomes critical.

Challenges and present work. In this paper, we take the first attempt to pre-train GNN models on a large heterogeneous graph, utilizing the intrinsic semantic and structural information as self-supervision. However, designing pre-training strategies of GNNs for heterogeneous graph is a non-trivial problem, presenting us with two key challenges.

(1) *How to distinguish and tailor to different types of nodes and edges during pre-training?* Existing pre-training strategies for GNNs [13, 14, 28] are only designed for homogeneous graphs, which treat all nodes or edges uniformly. As different node and edge types are the defining characteristics of a heterogeneous graph, it is crucial to encode them into a universal basis intrinsic to the graph, without dependence on any task-related label.

(2) *How to further preserve high-order semantic contexts during pre-training?* As motivated in Figure 1(a), a heterogeneous graph often embodies complex multi-party relationships involving a number of nodes and edges. Existing pre-training approaches usually utilize simple connectivity structures between node pairs as self-supervised information, which cannot explain high-order semantics on a heterogeneous graph. Hence, it is vital to design more advanced structures for pre-training in order to encode complex semantic contexts flexibly.

To tackle the above challenges, we develop a novel **C**onstrastive **P**re-**T**raining strategy of GNNs for **H**eterogeneous **G**raphs, named **CPT-HG**, which preserves heterogeneity in terms of not only node or edge differences individually, but also high-order semantic contexts among multiple nodes and edges collectively. More specifically, for the first challenge, we design a relation-level pre-training task to differentiate the relation type between two nodes of different types (*e.g.,* Author-Paper and Paper-Conference relations), to encode a universal basis for downstream tasks. To enhance the representativeness of the relation-level samples, inspired by contrastive learning [42], we propose to discriminate negative relation-level samples from positive relation-level samples in a contrastive manner. Specifically, we construct the negative relation-level samples from two aspects: (1) negative samples from inconsistent relations where two nodes share an "incorrect" relation distinct from the positive relation, and (2) negative samples from unrelated nodes where two nodes are not linked at all in the graph. To address the second challenge, we propose a subgraph-level pre-training task on a heterogeneous graph. The heterogeneity gives rise to high-order structures such as meta paths [23, 34] and meta graphs [8, 16] that are capable of capturing various semantic contexts, as shown in Figure 1(c) and (d). Considering that meta graph is able to capture richer and subtler semantics than meta path [49], in our subgraph-level strategy, we employ meta graph, rather than meta path, to generate subgraph instances including the positive samples and the negative samples, such that the pre-training can encode high-order semantic contexts that are also relevant to different downstream tasks.

To summarize, we make the following major contributions in this work.

- We address the under-explored setting of GNN pre-training on a heterogeneous graph in a self-supervised manner.
- We propose a novel contrastive pre-training strategy for GNNs on heterogeneous graphs, named CPT-HG, which leverages both the relation- and subgraph-level pre-training tasks to contrastively preserve the rich semantics as a form of transferable knowledge for the downstream tasks.
- On three real-world heterogeneous graphs, we conduct extensive experiments and analysis to demonstrate that our CPT-HG significantly outperforms various state-of-the-art approaches.

2 RELATED WORK

2.1 Graph Neural Network

In recent years, GNNs have received significant attention due to the ability to model graph-structured data, which can naturally capture both graph structures and feature information associated with the graph [41, 48, 50]. Originally proposed, as a framework of utilizing neural networks to learn node representations on graphs [24, 29], GNN is extended to convolution neural networks using spectral methods [2, 4, 43] and message passing architectures to aggregate neighbors' features [1, 10, 18, 26]. For example, Kipf *et al.* [18] have proposed the graph convolutional networks via allocating first-order approximation of spectral graph convolutions. Moreover,

graph attention networks have been proposed to learn the importance between nodes and its neighbors and fuse the neighbors to perform node classification [38]. Besides, some studies have attempted to deploy the GNNs on a heterogeneous graph [15, 30, 40, 47]. Most of these GNN models can be used as the base leaner for our proposed pre-training framework.

2.2 Contrastive Learning

Contrastive learning has recently become a prominent technique in unsupervised learning, which can achieve state-of-the-art results. The key idea of contrastive learning is to contrast semantically similar (positive) and dissimilar (negative) pairs of data points. There are some methods based on mutual information that measure the loss in the latent space by contrastive samples from different distribution [32, 39]. Deep graph infomax [39] extends the deep infomax [12] to graphs and obtains better performance in node classification by learning node representations with the contrastive node and graph encoding. Recently, in computer vision, various frameworks [3, 11, 36] for contrastive learning have been proposed, which design the self-supervised task to learn the representation by distinguishing the similar images from the dissimilar ones. Contrastive learning is also utilized in our model to learn the heterogeneous information, and the loss function, called InfoNCE [37], is employed in this paper for contrastive loss calculation.

2.3 Pre-training on Graphs

To enable more effective learning on graphs, researchers have explored how to pre-train GNNs for node-level representations on unlabeled graph data. Inspired by pre-training techniques in natural language processing [5] and computer vision [3, 11], recent studies [10, 14, 22, 28, 39, 46] have been proposed to pre-train GNNs with self-supervised information. Navarin *et al.* [25] have utilized the graph kernel for pre-training, while another work [13] have propose different strategies to pre-train graph neural networks at both node and graph levels, although labeled data are required at the graph level. More recently, Hu *et al.* [14] have designed the graph generation factorization to guide the base GNN model to reconstruct both the attributes and structure of the input graph. Qiu *et al.* [28] have proposed a contrastive pre-trained model to capture the universal and transferable structural patterns from multiple input graphs and You *et al.* [46] have proposed various graph data augmentations to generate positive/negative samples for conducting contrastive learning. Recently, Lu *et al.* [22] have proposed a pre-training framework which leverages meta-learning to reduce the gap between pre-training and fine-tuning. These methods usually focus on homogeneous graphs, which can not be directly applied for heterogeneous graph.

3 PRELIMINARY

In this section, we give some formal definitions of heterogeneous graphs and related concepts, and formalize the problem of pre-training on heterogeneous graphs.

Definition 3.1. Heterogeneous Graph. A heterogeneous graph [31], denoted as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{R}, \phi, \varphi\}$, is a form of graph, where $\mathcal{V}$ and $\mathcal{E}$ denote the sets of nodes and edges, respectively. It is also associated with a node type mapping function $\phi : \mathcal{V} \rightarrow \mathcal{A}$ and an edge type mapping function $\varphi : \mathcal{E} \rightarrow \mathcal{R}$, where $\mathcal{A}$ and $\mathcal{R}$ denote the sets of node and edge types such that $|\mathcal{A}| + |\mathcal{R}| > 2$.

The network schema $T_{\mathcal{G}} = (\mathcal{A}, \mathcal{R})$ of a heterogeneous graph specifies type constraints on node objects and relationships between the node objects. These constraints make a heterogeneous graph semi-structured, guiding the exploration of the semantics of the network [34].

An example of heterogeneous graph is illustrated in Figure 1(a) on bibliographic data with the network schema in Figure 1(b). Observe that it consists of multiple node types (*i.e.,* author, paper, conference and term) and relation types (*e.g.,* paper-author, paper-conference).

Definition 3.2. Meta Path. A meta path $\mathcal{P}$ [34] is a path defined on the network schema $T_{\mathcal{G}} = (\mathcal{A}, \mathcal{R})$ of the heterogeneous graph $\mathcal{G}$, and is denoted in the form of $A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} \cdots \xrightarrow{R_l} A_{l+1}$, which defines a composite relation $R = R_1 \circ R_2 \circ \cdots \circ R_l$ between type A_1 and A_{l+1}, where $\circ$ denotes the composition operator on relations.

Meta path is a basic analysis tool for heterogeneous graphs, which can extract sub-structure from heterogeneous graphs and embody rich semantics. For example, as illustrated in Figure 1(c), the meta path "Paper-Conference-Paper" (PCP) indicates the semantics of two papers published in the same conference.

Definition 3.3. Meta Graph. A meta graph [8] can be represented as $M = (\mathcal{V}_M, \mathcal{E}_M, \mathcal{A}_M, \mathcal{R}_M)$, where $\mathcal{V}_M \subseteq \mathcal{V}$, $\mathcal{E}_M \subseteq \mathcal{E}$ constrained by $\mathcal{A}_M \subseteq \mathcal{A}$ and $\mathcal{R}_M \subseteq \mathcal{R}$, respectively. On a heterogeneous graph $\mathcal{G}$, let $\mathcal{M} = \{M_1, M_2, \cdots, M_{|\mathcal{M}|}\}$ be a set of meta graphs.

In contrast to meta path, a meta graph can be viewed as a combination of multiple meta paths and embodies richer and subtler semantics. On the other hand, a meta path can also be considered as a special case of a meta graph. For example, in Figure 1(d), the meta graph M_1 (P-A/C-P) indicates the semantics of two papers on similar topics from the same author, which can not indicated by a single meta path.

In the paper, similar to the setting of pre-training strategy on homogeneous graph [14], we pre-train the model on a part of large-scale heterogeneous graph, while fine-tune the downstream task on the remaining part.

Definition 3.4. Pre-training on Heterogeneous Graph. For a heterogeneous graph $\mathcal{G}$, we split the whole graph into two parts for pre-training and fine-tuning. We denote the pre-training graph and fine-tuning graph as $\mathcal{G}^{pre}$ and $\mathcal{G}^{fine}$, respectively. The strategies on heterogeneous graph largely follow a two-step paradigm: 1) Pre-training the GNN f_θ based on the graph $\mathcal{G}^{pre}$. The learned parameter θ is expected to capture the semantic information and structural properties as transferable knowledge. 2) Fine-tuning the pre-trained GNN model with the initial parameter θ for downstream tasks on the graph $\mathcal{G}^{fine}$.

4 THE PROPOSED CPT-HG MODEL

In this section, we develop a novel contrastive pre-training strategy for GNNs on a heterogeneous graph, which preserves the heterogeneous semantics at both the relation- and subgraph-level. Figure 2 presents the overall framework of our proposed CPT-HG. We design semantic-aware pre-training tasks at both the relation- and

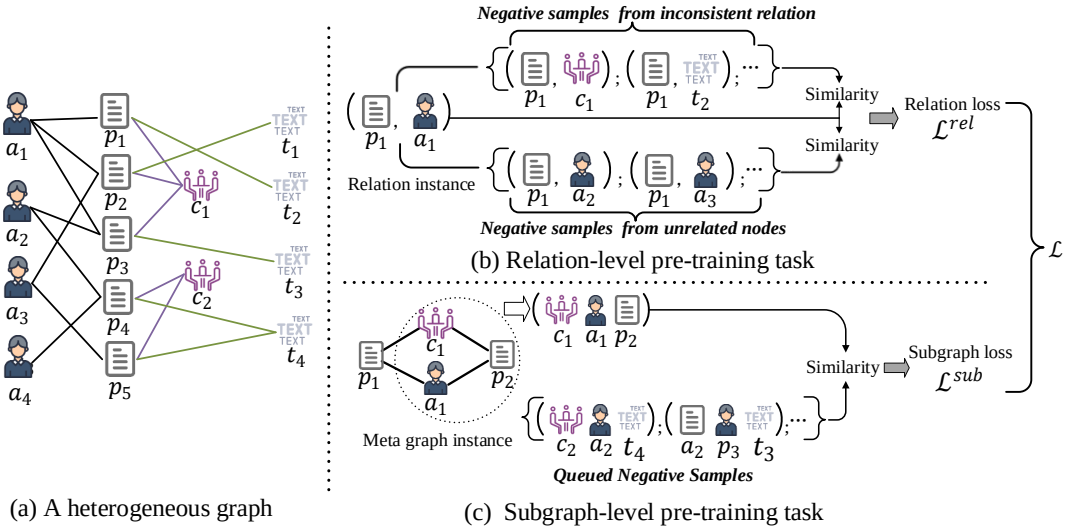


Figure 2: The overall framework of CPT-HG.

subgraph-levels, and further enhance their representativeness by employing contrastive learning. On the one hand, in Figure 2(b) we design the relation-level pre-training task, in which two kinds of negative relation-level samples are discriminated from the positive relation-level sample, to encode the relational semantics which constitute the basis of heterogeneity on a heterogeneous graph. On the other hand, in Figure 2(c) we propose subgraph-level pre-training tasks based on meta graphs to encode high-order semantic contexts. Finally, we jointly optimize the relation- and subgraph-level pre-training tasks to encode richer and subtler semantics into the transferable knowledge for downstream tasks.

4.1 Relation-level Pre-training Task

In existing methods, positive and negative nodes/edges are only differentiated by network structures, such as through a perturbation of the original graph structures and attributes [39]. However, a heterogeneous graph brings rich semantics in multiple types of relations, and thus it is crucial for the learned transferable knowledge to encode such semantics for the downstream tasks.

Given a positive triple $\langle u, R, v\rangle \in \mathcal{P}^{rel}$ on a heterogeneous graph $\mathcal{G}$, nodes $u \in \mathcal{V}$ and $v \in \mathcal{V}$ are connected via relation instances of type $R \in \mathcal{R}$, such as $\langle p_1, P\text{-}A, a_1\rangle$ in Figure 1(a). Here $\mathcal{P}^{rel}$ is the set of positive triples for the pre-training task. We propose to construct negative samples in two ways, including negative samples from inconsistent relations and negative samples from unrelated nodes.

Negative Samples from Inconsistent Relations. As motivated, on a heterogeneous graph, it is imperative to distinguish the positive and negative samples under a specific relation, which retains the relational semantics between two nodes. Thus, CPT-HG is designed to have the ability of evaluating the "authenticity" *w.r.t.* a relation R between a given pair of nodes u and v .

Given the positive triple $\langle u, R, v\rangle \in \mathcal{P}^{rel}$, there exists a node w connecting with u under a relation type R^- that is inconsistent with R. Thus, the triple $\langle u, R^-, w\rangle$ represents a different semantic context from $\langle u, R, v\rangle$. For example, in Figure 2(a), the triple $\langle p_1, P\text{-}A, a_1\rangle$ under the relation P-A indicates that the paper p_1 is written by the author a_1, while the relation P-C connecting papers and conferences, *e.g.,* $\langle p_1, P\text{-}C, c_1\rangle$, represents that the paper p_1 is published in the conference c_1, which can be treated as a negative sample of $\langle p_1, P\text{-}A, a_1\rangle$. Thus, we construct the negative triples $\langle u, R^-, w\rangle$ with the inconsistent relation R^- drawn from the inconsistent relation set $\mathcal{R}^- = \mathcal{R} \setminus \{R\}$. Formally, for the positive triple $\langle u, R, v\rangle \in \mathcal{P}^{rel}$, we define negative samples from inconsistent relations, denoted as $\mathcal{N}^{rel}$, as follows:

$$\mathcal{N}^{rel}_{\langle u,R,v\rangle} = \{\langle u, \varphi(u,w), w\rangle | (u,w) \in \mathcal{E}, \varphi(u,w) \in \mathcal{R}^-\}, \quad (1)$$

where nodes u and w are connected via relation instances of type R^- and we treat edge types as directed edge (*e.g.,* P-A and A-P are two different relation types). Due to the large scale of $\mathcal{N}^{rel}$, here we uniformly sample a subset of $\mathcal{N}^{rel}$ for each inconsistent relations to pre-train our model.

Negative Samples from Unrelated Nodes. To discriminate the positive samples from unconnected structures, we further generate negative samples from unrelated nodes. Existing negative sampling methods simply select edges that is not in the graph [6]. However, such negative samples are believed to be easily distinguishable from the positive samples. Thus, we construct the negative samples from the k-hop neighbors of node u. To be specific, given the positive triple $\langle u, R, v\rangle \in \mathcal{P}^{rel}$, we define the negative samples from unrelated nodes as:

$$\mathcal{N}^{node}_{\langle u,R,v\rangle} = \{\langle u, *, v^-\rangle | v^- \in \mathcal{V}\}. \quad (2)$$

where v^- is the the k-hop neighbors of node u and $*$ means any composite relation connecting u and v^-. In our implementation, we set $k = 5$ for negative samples from unrelated nodes.

We generate the negative samples from the above two strategies to conduct contrastive learning. Thus, to capture the relation information, we minimize the loss for a positive triple $\langle u, R, v\rangle \in \mathcal{P}^{rel}$ with two types of negative samples. For the negative samples with

an inconsistent relation, i.e., $\mathcal{N}^{rel}$, we calculate the following contrastive loss:

$$\mathcal{L}^{rel1} = \sum_{\langle u,R,v\rangle \in \mathcal{P}^{rel}} -\log \frac{\exp\left(\mathbf{h}_u^\top \mathbf{W}_R \mathbf{h}_v\right)}{\sum_{i\in\{v\}\cup\{w|\langle u,R^-,w\rangle\in\mathcal{N}^{rel}_{\langle u,R,v\rangle}\}} \exp\left(\mathbf{h}_u^\top \mathbf{W}_R \mathbf{h}_i\right)}, \tag{3}$$

where $\mathbf{W}_R \in \mathbb{R}^{d\times d}$ is a learnable relation weight matrix for relation R. Here $\mathbf{h}_u$ indicates the representation vector of node u, which is generated by any existing GNN architecture.

For the negative samples from unrelated nodes, i.e., $\mathcal{N}^{node}$, we have the following loss function:

$$\mathcal{L}^{rel2} = \sum_{\langle u,R,v\rangle \in \mathcal{P}^{rel}} -\log \frac{\exp\left(\mathbf{h}_u^\top \mathbf{h}_\mathbf{v}\right)}{\sum_{i\in\{v\}\cup\{v^-|\langle u,*,v^-\rangle\in\mathcal{N}^{node}_{\langle u,R,v\rangle}\}} \exp\left(\mathbf{h}_u^\top \mathbf{h}_\mathbf{i}\right)}. \tag{4}$$

To integrate the information of the two types of negative samples, we calculate the relation-level pre-training loss as:

$$\mathcal{L}^{rel} = \mathcal{L}^{rel1} + \mathcal{L}^{rel2}. \tag{5}$$

So far, we utilize relation discrimination for discovering and persevering the rich semantics in a heterogeneous graph at relation level.

4.2 Subgraph-level Pre-training Task

In order to preserve the high-order semantic contexts on a heterogeneous graph, a natural idea is to employ meta paths to explore high-order relations, thereby incorporating richer semantics [31, 34]. However, there are two weaknesses of the widely used meta paths for pre-training GNNs on heterogeneous graphs: (1) Meta paths have been shown to have limited ability to characterize rich semantics and extract high-order structures, compared to meta graphs [16]. (2) Starting from a source node, the number of nodes a meta path can reach is often too large, while the number of nodes a meta graph from the same source node can cover is fewer due to a more complex and restrictive structure, which makes meta graph more efficient than meta path.

Therefore, we generate meta graph instances to construct the positive samples, which capture the elaborate subgraph structure and the subtle semantic information on a heterogeneous graph. Besides, we take inspiration from computer vision [3, 11] to generate queued negative samples and distinguish the positive and negative samples with contrastive learning [37].

Structural Positive Samples. Given a meta graph $M \in \mathcal{M}$ and a source node u, we define and construct a meta graph instance $m \in \mathcal{I}(M)$ as a set of nodes surrounding u such that they match the meta graph M, where $\mathcal{I}(M)$ denotes the set of all instances of M. For example, in Figure 2(c), for the meta graph $M = P\text{-}(C/A)\text{-}P$, we have a meta graph instance as $m = \{p_1, c_1, a_1, p_2\}$. Based on this definition, we generate the meta graph-based structural positive samples *w.r.t.* the source node u as

$$\mathcal{P}^{sub}_u = \bigcup_{m\in\mathcal{I}(M),\, M\in\mathcal{M}} m \setminus \{u\}, \tag{6}$$

where $\mathcal{M}$ is the set of pre-defined meta graphs. Here, we choose the set of meta graphs with the domain knowledge [16]. Intuitively, the structural positive samples capture not only denser local structural connectivity, but also subtler semantic contexts.

Queued Negative Samples. To construct negative samples, inspired by the dynamic generation of negative samples [11], we design queued negative samples. To be specific, based on positive samples from previous batches in the training process, we generate a negative sample by adding the positive sample from the most recent batch and remove the oldest one. Assuming that the most recent positive sample of the previous training is $\mathcal{P}^{sub}_i(t-1)$ *w.r.t.* node i, then we have the current negative samples as:

$$\mathcal{N}^{sub} = [\mathcal{P}^{sub}_i(t-1), \mathcal{P}^{sub}_j(t-2), \cdots]. \tag{7}$$

Note that we randomly initialize the queue of negative samples and then update the queue during model training. For example, in the Figure 2(c), the positive sample $\{c_1, a_1, p_2\}$ will replace the oldest negative sample in the next batch. It is worth noting that, if the negative samples are generated by a random selection, they are often easily distinguishable from the positive samples on the ground that nodes in negative samples are very likely to be unrelated. Besides, different nodes may have the same structural positive samples. For example, both node p_1 and p_3 have the same structural positive samples, *e.g.*, $\{c_1, a_1, p_2\}$ in Figure 2(a), which lead to suboptimal contrastive learning. To prevent this situation, for the subgraph-level pre-training task for node p_1, we remove such negative samples from the queue.

To incorporate high-order semantic contexts, we then model the likelihood that the source node u is associated with the positive samples while unrelated with the negative samples as follows:

$$\mathcal{L}^{sub} = -\sum_{u\in\mathcal{V}} \sum_{P^+\in\mathcal{P}^{sub}_u} \log \frac{\exp\left(\mathbf{h}_u^\top f(P^+)\right)}{\sum_{P\in\{P^+\}\cup\mathcal{N}^{sub}} \exp\left(\mathbf{h}_u^\top f(P)\right)}, \tag{8}$$

where $f(\cdot)$ is a pooling function (e.g., mean pooling in our work) for obtaining the representation of nodes. Intuitively, by optimizing $\mathcal{L}^{sub}$, CPT-HG is capable of capturing the semantic information through high-order local structures on a heterogeneous graph.

4.3 Training and Optimization

Altogether, to capture the semantic information and preserve the high-order structures at relation- and subgraph-levels, we minimize the following loss for a heterogeneous graph $\mathcal{G}$:

$$\mathcal{L} = \mathcal{L}^{sub} + \lambda \mathcal{L}^{rel}, \tag{9}$$

where λ is a balancing coefficient. We adopt the Adam [17] optimizer to train the proposed CPT-HG and repeat the relation- and subgraph-levels pre-training tasks for more iterations until the model converges.

Algorithm. The model training for CPT-HG is outlined in Algorithm 1. Its basic idea is to generate the negative samples at both levels for conducting contrastive learning. First, we initialize an empty negative queue to store the previously-sampled structural positive samples as more meaningful negative samples. Given node u in the heterogeneous graph, we generate the negative relation-level samples from inconsistent relations and unrelated nodes in line 4. If we sample too many negative relation-level samples that meet the condition in Eq. (1) and Eq. (2), we remove some negative samples randomly if the number of negative samples exceeds the maximum setting. After the negative sampling at the relation level, we calculate the relation-level loss in line 5. Then, following the

Algorithm 1 Algorithm framework of CPT-HG

Require: Heterogeneous graph $\mathcal{G}$, a GNN model f_θ

1: Initialize model parameters θ with Xavier initialization
2: Randomly initialize queued negative samples $\mathcal{N}^{sub}$
3: **for** each node u in $\mathcal{G}$ **do**
4: Generate the negative samples with Eq. (1) and Eq. (2)
5: Calculate $\mathcal{L}^{rel}$ by Eq. (5)
6: Generate the structural positive samples $\mathcal{P}_u^{sub}$ with Eq. (6)
7: **for** each structural positive sample $P^+ \in \mathcal{P}_u^{sub}$ **do**
8: Calculate $\mathcal{L}^{sub}$ by Eq. (8)
9: **end for**
10: Update $\mathcal{N}^{sub}$ with the structural positive sample P^+
11: **end for**
12: Calculate $\mathcal{L}$ by Eq. (9)
13: Update parameters θ
14: **return** the GNN model f_{θ^*}

predefined meta graphs, we generate the structural positive samples to preserve high-order semantic contexts. Then, we calculate the subgraph-level loss for each structural positive samples. After that, we progressively update the negative queue by adding the latest structural positive samples and remove the oldest ones in line 10. Moreover, the structural samples extracted from different nodes can bring in the high-order semantic contexts for contrastive learning. Then, we calculate the loss in Eq. (9) to capture the semantic information and preserve the high-order structures. Afterwards, we can obtain pre-trained model parameters for downstream tasks.

Discussion. The proposed pre-training framework can be universally applied to different GNN models since the pre-training tasks are not related to the implementation of GNNs. In contrast to the previous pre-training methods on a homogeneous graph, our approach can deal with various types of nodes and relations and fuse rich semantic contexts on a heterogeneous graph. In particular, our method can be applied to pre-train both heterogeneous [15, 30, 40] and homogeneous [10, 18, 38] GNNs alike. Our semantics-aware pre-training tasks directly enable homogeneous GNNs to capture transferable semantics on the one hand, and further enhance heterogeneous GNNs in their semantic expressiveness and generalization ability to downstream tasks on the other hand.

We next conduct a complexity analysis of our pre-training procedure. The time complexity consists of two parts: (1) the time complexity of learning node representations with GNNs, which depends on the architectures of the base GNN. Here, we denote it as X; (2) the time complexity of generating pre-training tasks, which is linear *w.r.t.* the number of the negative samples. We denote the number of negative samples at both relation- and subgraph- levels as k_1 and k_2, respectively. The average size of structural positive samples and positive triples is denoted as $|\overline{\mathcal{P}^{sub}}|$ and $|\overline{\mathcal{P}^{rel}}|$, respectively. Thus, the time complexity of our CPT-HG is $O\left(X + N\left(k_1|\overline{\mathcal{P}^{rel}}| + k_2|\overline{\mathcal{P}^{sub}}|\right)d^2\right)$, where d and N is the embedding dimension and the number of nodes, respectively.

5 EXPERIMENTS

In this section, we first conduct extensive experiments on three real-world datasets to evaluate model performance, and then investigate the underlying mechanism of CPT-HG with two ablated models and different GNN architectures. Lastly, we explore the impact of the model settings on task performance.

5.1 Experimental Settings

Dataset. We conduct experiments on three datasets, namely DBLP [9], Yelp [44] and Aminer [35]. The detailed statistics of three datasets are summarized in Table 1.

- DBLP is extracted from the computer science bibliography website[1]. According to the domains of published papers, the authors in DBLP are labeled with four research areas, including *Database*, *Data Mining*, *Artificial Intelligence*, and *Information Retrieval*.
- Yelp is a widely used benchmark dataset, which contains a network of businesses, users, locations and reviews from Yelp Inc[2].
- Aminer is a bibliographic graphs[3]. The papers in Aminer are labeled with 17 research fields, e.g., *Artificial Intelligence*, which are used for node classification.

Pre-Training and Fine-Tuning Setting. We pre-train a GNN model and adopt the pre-trained model weights to initialize models for downstream tasks. Then we fine-tune the GNN models according to the specific downstream tasks on an unseen fine-tuning graph and evaluate the model performance. Specifically, for each dataset, we randomly split the whole graph into two graphs for pre-training and fine-tuning. In DBLP dataset, we randomly split DBLP into pre-training and fine-tuning graphs which respectively contains 50% authors and the other associated nodes. In Yelp dataset, with the ratio of 3:1 of Business node, we randomly split Yelp into pre-training and fine-tuning graphs. In Aminer dataset, we construct the pre-training graph from 11 random fields, and the left is the fine-tuning graph.

Baselines. We compare our proposed CPT-HG with the state-of-art baselines, including a no pre-train method, three graph neural network based methods with unsupervised objectives (*i.e.,* GAE, EdgePred, DGI), and a pre-training method (*i.e.,* GPT-GNN).

- No pre-train method adopts the GNN model to learn node representations and then conducts downstream tasks on the fine-tuning graph.
- GAE [19] focuses on a traditional link prediction task. It randomly masks out a fixed proportion of the edges and train the model to reconstruct these masked edges.
- EdgePred [10] predicts the connectivity of node pairs and forces connected nodes to have similar node embeddings.
- DGI [39] maximizes local mutual information across the graph's patch representations.
- GPT-GNN [14] is a state-of-art model for pre-training GNNs, which reconstructs the attributes and structure of the input graph to learn the transferable knowledge from the input graph.

It is worth noting that our CPT-HG can be implemented for different GNN models. Here, we mainly study HGT [15], the most

[1] https://dblp.uni-trier.de
[2] https://www.yelp.com/dataset
[3] https://www.aminer.cn/citation

Dataset	#Node type	#Nodes	#Edge type	#Edges
DBLP	Author (A)	4,057	P-A	19,645
	Paper (P)	16,670	P-V	16,670
	Term (T)	13,420	P-T	133,039
	Venue (V)	40		
Yelp	Business (B)	7,474	B-T	132,928
	Location (L)	65	B-L	7,474
	Star (S)	9	B-S	7,474
	Term (T)	36,412	T-T	360,676
Aminer	Paper (P)	614,209	P-A	2,311,822
	Author (A)	737,621	P-C	764,246
	Conference (C)	842	P-P	4,665,400
	Terms (T)	80,589	P-T	7,722,124

Table 1: Statistics of the three datasets.

expressive and state-of-the-art GNN architecture for heterogeneous graphs. We also experiment with other popular architectures in Section 5.3, including GCN [18], GAT [38], GraphSAGE [10] and RGCN [30].

Parameter Settings. We implement our CPT-HG with PyTorch [27] and adopt Adaptive Moment Estimation (Adam) [17] optimizer to train the proposed CPT-HG. In the pre-training procedure, we set the dimension of node representation to 64, the number of the base GNN layers to 2, and the head of attention to 1 for all methods. The learning rate is arrange from [0.01, 0.008, 0.005, 0.001]. For the balancing coefficient λ in Eq. (9), we set it as 0.5 in the pre-training procedure. We use early stopping based the performance on validation set with a patience of 10 epochs for model training. For the other parameters of the baselines, we optimize them empirically under the guidance of literature. For our CPT-HG, the maximum number of negative samples from inconsistent relations and unrelated nodes are set to 100 and 200, respectively. The maximum number of queued negative samples at subgraph-level is set to 100. In the pre-training procedure at both levels, we will remove some negative samples randomly if the number of negative samples exceeds the maximum setting. During the experiment, we observe that the number of negative samples, which are generated according to the aforementioned rules, falls within the range of maximum setting in most cases and thus the removal is usually needless to be performed.

Evaluation Protocol. Following the previous work [14], we first pre-train the model (including the baselines and our CPT-HG) by utilizing the self-supervised information in the pre-training graph. Then we fine-tune the pre-trained model with labeled information in downstream tasks (e.g., link prediction). The downstream experiments are run with 10 random seeds, and we report the average experiment results and standard deviation on the test set.

5.2 Performance Comparison

In this section, we empirically compare CPT-HG to baselines in two downstream tasks, including link prediction and node classification.

Link Prediction. After pre-training our CPT-HG and the baselines, we apply the pre-trained model on the fine-tuning graph to predict edges. Specifically, we consider the prediction of Paper-Term in DBLP dataset, Business-Location in Yelp dataset, as well as Paper-Author and Paper-Conference in Aminer dataset. During the fine-tuning process, we randomly divide the edges to be predicted (e.g., Paper-Author in Aminer) with the ratio of 8:1:1 to construct the training, validation and test sets. Following [30] , we randomly sample the same number of unconnected node pairs as the training set to serve as negative samples for model optimization. At last, we minimize the cross-entropy loss to train the GNN model in fine-tuning process, and evaluate the prediction performance with MRR metric [14, 30].

Table 2 demonstrates the link prediction performance on three datasets. Overall, our CPT-HG consistently yields the best performance among all methods on three datasets, which brings an MRR improvement by 2.16%–6.81% compared to the best baseline. The significant improvement attributes to the structural and semantics information modeling for heterogeneous graphs. Compared to the no pre-train baseline, our CPT-HG significantly improves the link prediction performance by 5.83%, 2.64% and 10.85% on three datasets, respectively. The improvements suggest that the contrastive pre-training on heterogeneous graphs is capable of learning transferable and informative knowledge for the downstream tasks. Among different baselines, conventional graph neural network based methods (e.g., GAE) achieve the unsatisfactory performance due to the insufficient use of pre-training graphs. The GPT-GNN performs better by generative pre-training on subgraphs, so as to learn transferable knowledge. However, they still underperform our proposed CPT-HG. We believe the reason is that the relation- and subgraph-level pre-training tasks allow our CPT-HG to make full use of structures and semantics on a heterogeneous graph.

Node Classification. To evaluate model performance in node classification task, the node representation learned by the pre-trained model is fed into a linear classifier to predict the node label. Following [30], we randomly split the labeled nodes with the ratio of 1:2:7 for training, validation and test set. Since there is no label information in Yelp dataset, here we conduct experiments on DBLP and Aminer datasets, and adopt the accuracy as evaluation metric.

As presented in Table 3, we can find that CPT-HG performs consistently much better than all baselines on two datasets. As observed in link prediction tasks, no pre-train method is least competitive due to the lack of rich information in pre-training subgraphs. Generally, GNN based methods which combine the structure and feature information, e.g., DGI, usually perform better than no pre-train method. This indicates that the pre-training process with self-supervision provides useful and discriminated information for node classification. On DBLP dataset, we also observe that the baseline GAE generally achieve better performance than other methods. The reason might be that GAE mainly on modeling relations while relations in DBLP brings much more effective information for node representation than structural information.

Dataset	Link Type	No pre-train	GAE	EgePred	DGI	GPT-GNN	CPT-HG	Improv.
DBLP	Paper-Term	12.34 ± 1.43	12.51 ± 0.71	12.61 ± 0.44	12.47 ± 0.68	12.71 ± 0.35	**13.06** ± 0.42	2.75%
Yelp	Business-Location	45.83 ± 0.42	45.92 ± 0.52	46.10 ± 0.31	45.57 ± 0.64	46.04 ± 0.75	**47.04** ± 0.71	2.03%
Aminer	Paper-Conference	39.23 ± 1.75	40.31 ± 0.78	39.86 ± 1.17	40.74 ± 1.35	41.37 ± 0.76	**42.17** ± 1.23	1.93%
	Paper-Author	5.63 ± 0.73	5.71 ± 0.41	5.62 ± 0.87	5.84 ± 0.52	6.02 ± 0.45	**6.43** ± 0.54	6.81%

Table 2: Experiment results (MRR(%) ± std) in link prediction task on the three datasets. The best method is bolded, and the second best is underlined.

Dataset	Labeled Node Type	No pre-train	GAE	EgePred	DGI	GPT-GNN	CPT-HG	Improve.
DBLP	Author	87.45 ± 0.43	90.56 ± 0.73	89.24 ± 0.57	88.26 ± 0.66	89.57 ± 0.45	**91.45** ± 0.54	0.98%
Aminer	Paper	92.17 ± 0.56	92.72 ± 0.32	93.41 ± 0.46	92.37 ± 0.25	93.75 ± 0.67	**96.32** ± 0.43	2.74%

Table 3: Experiment results (Accuracy(%) ± std) in the node classification task on DBLP and Aminer datasets. The best method is bolded, and the second best is underlined.

Downstream Task	Link Prediction				Node Classification	
Dataset	DBLP	Yelp	Aminer		DBLP	Aminer
Link/Labeled Node Type	Paper-Term	Business-Location	Paper-Conference	Paper-Author	Paper	Author
No pre-train	12.34 ± 1.43	45.83 ± 0.42	39.23 ± 1.75	5.63 ± 0.73	87.45 ± 0.43	92.17 ± 0.56
CPT-HG$_{sub}$	12.65 ± 0.42	**47.15** ± 0.44	41.54 ± 0.33	6.04 ± 0.51	89.57 ± 0.61	94.14 ± 0.54
CPT-HG$_{rel}$	12.79 ± 0.56	46.74 ± 0.65	41.75 ± 0.65	6.24 ± 0.15	**92.45** ± 0.54	95.16 ± 0.32
CPT-HG	**13.06** ± 0.42	47.04 ± 0.71	**42.17** ± 1.23	**6.43** ± 0.54	91.45 ± 0.54	**96.32** ± 0.43

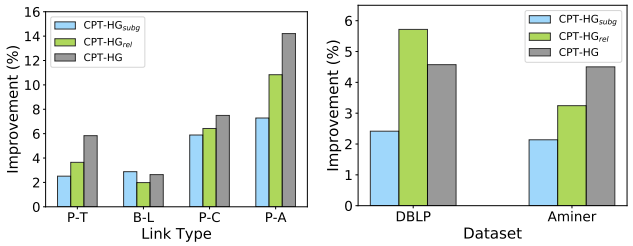
Table 4: Analysis of different ablated models in various downstream tasks.

5.3 Model Analysis

In this section, we investigate the underlying mechanism of CPT-HG from two perspectives: the ablation study of relation and subgraph-level pre-training tasks, as well as the generality analysis of CPT-HG for different GNN architectures.

Ablation Study. We attempt to understand how relation-level and subgraph-level pre-training tasks facilitate the contrastive pre-training on heterogeneous graphs. Towards this end, we conduct an ablation study and consider two ablated variants of CPT-HG namely **CPT-HG$_{sub}$** and **CPT-HG$_{rel}$**. CPT-HG$_{sub}$ only includes the subgraph-level pre-training task for contrastive pre-training that models structure properties on a heterogeneous graph, while CPT-HG$_{rel}$ only employs relation-level pre-training task to capture semantic information. In Table 4 and Figure 3, we report the performance of two ablated models and the improvement over no pre-train baseline in link prediction and node classification tasks.

Overall, all the comparison methods are better than no pre-train baseline as they all achieve significant improvements in three datasets. In particular, our complete CPT-HG achieves the greatest improvement in most cases, indicating the necessary for jointly capturing semantic relations and subgraph structures for pre-training heterogeneous graphs. Compared to CPT-HG$_{sub}$, the improvement brought by CPT-HG$_{rel}$ is more significant. Despite CPT-HG$_{sub}$ encodes structures of graphs, the semantic relations encoded in



(a) Link Prediction. (b) Node Classification.

Figure 3: Improvement of different ablated models over no pre-train baseline.

CPT-HG$_{rel}$ seem more important for node representations in heterogeneous graph [23]. We also observe that CPT-HG$_{rel}$ method achieves the smallest improvement in Yelp dataset, which is reasonable since relations in Yelp is less informative with small number of different types of nodes, as shown in the statistic of Yelp dataset in Table 1. On the other hand, CPT-HG$_{sub}$ significantly improves the link prediction performance by focusing on modeling subgraph structures in a graph.

GNN Architecture Analysis. Since our proposed CPT-HG is not limited to the architectures of GNNs, we further apply CPT-HG to five GNN architectures, including two heterogeneous GNNs

Base Model	No pre-train	CPT-HG	Improvement
HGT	5.63 ± 0.73	6.42 ± 0.54	14.0%
RGCN	4.15 ± 0.43	4.49 ± 0.50	7.55%
GCN	4.79 ± 0.81	5.14 ± 0.62	7.31%
GAT	4.83 ± 0.77	4.32 ± 0.89	-10.6%
GraphSAGE	5.47 ± 0.52	6.02 ± 0.62	10.24%

Table 5: Analysis of different GNN architectures in link prediction on Aminer dataset.

Model	Accuracy	Improvement
No pre-train	92.17 ± 0.56	-
CPT-HG$_{PAP}$	91.52 ± 0.56	-0.7%
CPT-HG$_{PCP}$	90.45 ± 0.41	-1.9%
CPT-HG$_{PATP}$	93.12 ± 0.52	+1.0%
CPT-HG$_{PACP}$	94.71 ± 0.46	+2.6%
CPT-HG	96.32 ± 0.56	+4.5%

Table 6: Analysis of different meta graphs in node classification on Aminer dataset.

(*i.e.,* HGT, RGCN) and there homogeneous GNNs (*i.e.,* GCN, GAT, GraphSAGE), for studying the universality of CPT-HG. Since similar trends are observed in the three datasets, here we only report the link prediction performance on the largest Aminer dataset.

As presented in Table 5, our proposed CPT-HG is capable of enhancing the downstream task performance for most GNN architectures. Besides, the pre-trained HGT achieves the best performance gain among all the GNN models. We think the reason is that CPT-HG can enhance the GNN model performance with the transferable semantic and structural properties on heterogeneous graph. We also observe that the pre-trained GAT obtains an unsatisfactory performance and the reason may be that it's intractable to learn a proper attention heads between pre-training and fine-tuning graphs, and thus it performs worse than no pre-train model.

5.4 Experimental Setting Analysis

Lastly, we investigate the impact of experimental settings on the model performance, including the impact of meta graphs and the size of the pre-training subgraph. Similar to previous analysis, we showcase the results on Aminer dataset since the observations are similar in other datasets.

Impact of Meta Graphs. Following some previous works [16, 49], we empirically validate the performance of CPT-HG with different meta graphs, including PAP, PCP, PATP and PACP. For example, we only utilize the meta graph PAP to conduct the subgraph-level pre-training tasks, denoted as CPT-HG$_{PAP}$ and we utilize all meta graphs for our original CPT-HG. In Table 6, we report the results of different meta graphs compared to the original model. We observe that pre-training our CPT-HG with some meta graphs (*e.g.,* PAP and PCP) hardly learns semantics and structures on a heterogeneous graph and even reduces the model performance. As the structure of the meta graph becomes more complex (e.g., PACP), our CPT-HG achieves more performance gains by encoding more structural information in a graph. In all, the CPT-HG with meta graphs (*i.e.,* PATP and PACP) achieves the best and significant performance improvement due to preserving more rich and subtle semantics, which confirms the benefit of meta graph employed in CPT-HG .

Percentage	MRR	Improvement
No pre-train	5.63 ± 0.56	-
10%	5.54 ± 0.16	-1.5%
50%	5.87 ± 0.41	+4.2%
100%	6.43 ± 0.56	+14.0%

Table 7: Compare the pre-training performance Gain with different percentage of pre-training datasets. Evaluate the Paper-Author link prediction on Aminer.

Impact of Pre-training Subgraph Size. We also explore how the size of pre-training subgraph would affect the model performance, and we utilize {10%, 50%, 100%} percentages of pre-training subgraphs to pre-train the base GNN for downstream tasks. The fine-tuning setting is the same as that in Section 5.2 for fair comparison. In Table 7, we observe that CPT-HG consistently improves the link prediction performance with more pre-training dataset. When the pre-training subgraph is much small (e.g., 10%), our CPT-HG hardly learns useful semantic and structural information for the downstream task, leading to poor performance. This phenomenon demonstrates that the superiority of GNN pre-training requires a large-scale pre-training subgraph.

6 CONCLUSION AND FUTURE WORK

In this work, we took the first attempt to pre-train GNN models on a heterogeneous graph, and presented a novel contrastive pre-training strategy for GNNs on heterogeneous graphs, named CPT-HG. To capture the semantic and structural information on the graph in a transferable form, we leverage both the relation- and subgraph-level pre-training tasks through contrastive learning, which utilize self-supervised information intrinsic to heterogeneous graphs. At the relation level, we designed a pre-training task to differentiate the relation type between two nodes, which encodes the fundamental characteristics of a heterogeneous graph. At the subgraph-level, we proposed a pre-training task to differentiate the subgraph instances of different meta graphs, which encodes the high-order semantic contexts. Extensive experiments on three real-world heterogeneous graphs demonstrated promising results and the superior ability of our CPT-HG to transfer knowledge to various downstream tasks via pre-training.

7 ACKNOWLEDGMENTS

This work is supported in part by the National Natural Science Foundation of China (No. U20B2045, 61772082, 61702296, 62002029) and the Agency for Science, Technology and Research (A*STAR) under its AME Programmatic Funds (Grant No. A20H6b0151). All opinions, findings, conclusions and recommendations are those of the authors and do not reflect the views of the funding agencies.

REFERENCES

[1] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. 2019. MixHop: Higher-Order Graph Convolutional Architectures via Sparsified Neighborhood Mixing. In *ICML*. 21–29.

[2] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2014. Spectral Networks and Locally Connected Networks on Graphs. In *ICLR*.

[3] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. 2020. A Simple Framework for Contrastive Learning of Visual Representations. In *ICML*.

[4] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *NIPS*. 3837–3845.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*. 4171–4186.

[6] Yuxiao Dong, Nitesh V. Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable Representation Learning for Heterogeneous Networks. In *SIGKDD*. 135–144.

[7] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph Neural Networks for Social Recommendation. In *arXiv preprint arXiv:1902.07243*.

[8] Yuan Fang, Wenqing Lin, Vincent W Zheng, Min Wu, Kevin Chen-Chuan Chang, and Xiao-Li Li. 2016. Semantic proximity search on graphs with metagraph-based learning. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*. IEEE, 277–288.

[9] Xinyu Fu, Jiani Zhang, Ziqiao Meng, and Irwin King. 2020. MAGNN: Metapath Aggregated Graph Neural Network for Heterogeneous Graph Embedding. In *WWW*. 2331–2341.

[10] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*. 1024–1034.

[11] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. 2020. Momentum Contrast for Unsupervised Visual Representation Learning. In *CVPR*. 9726–9735.

[12] R. Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Philip Bachman, Adam Trischler, and Yoshua Bengio. 2019. Learning deep representations by mutual information estimation and maximization. In *ICLR*.

[13] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay S. Pande, and Jure Leskovec. 2020. Strategies for Pre-training Graph Neural Networks. In *ICLR*.

[14] Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. 2020. GPT-GNN: Generative Pre-Training of Graph Neural Networks. In *SIGKDD*. 1857–1867.

[15] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous Graph Transformer. In *WWW*. 2704–2710.

[16] Zhipeng Huang, Yudian Zheng, Reynold Cheng, Yizhou Sun, Nikos Mamoulis, and Xiang Li. 2016. Meta Structure: Computing Relevance in Large Heterogeneous Information Networks. In *SIGKDD*. 1595–1604.

[17] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

[18] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. In *ICLR*.

[19] Thomas N. Kipf and Max Welling. 2016. Variational Graph Auto-Encoders. In *CoRR*, Vol. abs/1611.07308.

[20] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *ICLR*.

[21] Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter W Battaglia. 2018. Learning Deep Generative Models of Graphs. In *ICLR*.

[22] Yuanfu Lu, Xunqiang Jiang, Yuan Fang, and Chuan Shi. 2021. Learning to Pre-train Graph Neural Networks. (2021).

[23] Yuanfu Lu, Chuan Shi, Linmei Hu, and Zhiyuan Liu. 2019. Relation Structure-Aware Heterogeneous Information Network Embedding. In *AAAI*. 4456–4463.

[24] Gori Marco, Monfardini Gabriele, and Scarselli Franco. 2005. A new model for learning in graph domains. In *Proceedings of IJCNN*. 729–734.

[25] Nicolò Navarin, Dinh Van Tran, and Alessandro Sperduti. 2018. Pre-training Graph Neural Networks with Kernels. In *CoRR*, Vol. abs/1811.06930.

[26] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. 2016. Learning Convolutional Neural Networks for Graphs.. In *ICML*. 2014–2023.

[27] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NIPS*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.).

[28] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. 2020. GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training. In *SIGKDD*. 1150–1160.

[29] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The Graph Neural Network Model. In *IEEE TNN*. 61–80.

[30] Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling Relational Data with Graph Convolutional Networks. In *ESWC*. 593–607.

[31] Chuan Shi, Yitong Li, Jiawei Zhang, Yizhou Sun, and Philip S. Yu. 2017. A Survey of Heterogeneous Information Network Analysis. In *TKDE*. 17–37.

[32] Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. 2020. InfoGraph: Unsupervised and Semi-supervised Graph-Level Representation Learning via Mutual Information Maximization. In *ICLR*.

[33] Yizhou Sun and Jiawei Han. 2012. Mining heterogeneous information networks: a structural analysis approach. In *SIGKDD Explor.* 20–28.

[34] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S. Yu, and Tianyi Wu. 2011. PathSim: Meta Path-Based Top-K Similarity Search in Heterogeneous Information Networks. In *VLDB*. 992–1003.

[35] Jie Tang. 2016. AMiner: Mining Deep Knowledge from Big Scholar Data. In *WWW*. 373.

[36] Yonglong Tian, Dilip Krishnan, and Phillip Isola. 2019. Contrastive Multiview Coding. In *CoRR*, Vol. abs/1906.05849.

[37] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation Learning with Contrastive Predictive Coding. In *CoRR*, Vol. abs/1807.03748.

[38] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. In *arXiv preprint arXiv:1710.10903*.

[39] Petar Velickovic, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. 2019. Deep Graph Infomax. In *ICLR*.

[40] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S. Yu. 2019. Heterogeneous Graph Attention Network. In *WWW*. 2022–2032.

[41] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. 2020. A Comprehensive Survey on Graph Neural Networks. In *TNNLS*. 1–21.

[42] Zhirong Wu, Yuanjun Xiong, Stella X. Yu, and Dahua Lin. 2018. Unsupervised Feature Learning via Non-Parametric Instance Discrimination. In *CVPR*. 3733–3742.

[43] Bingbing Xu, Huawei Shen, Qi Cao, Yunqi Qiu, and Xueqi Cheng. 2019. Graph Wavelet Neural Network. In *ICLR*.

[44] Carl Yang, Yuxin Xiao, Yu Zhang, Yizhou Sun, and Jiawei Han. 2020. Heterogeneous Network Representation Learning: Survey, Benchmark, Evaluation, and Beyond. In *CoRR*, Vol. abs/2004.00216.

[45] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *SIGKDD*. 974–983.

[46] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. *NIPS* 33 (2020).

[47] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V. Chawla. 2019. Heterogeneous Graph Neural Network. In *SIGKDD*. 793–803.

[48] Ziwei Zhang, Peng Cui, and Wenwu Zhu. 2020. Deep Learning on Graphs: A Survey. In *TKDE*.

[49] Huan Zhao, Quanming Yao, Jianda Li, Yangqiu Song, and Dik Lun Lee. 2017. Meta-Graph Based Recommendation Fusion over Heterogeneous Information Networks. In *SIGKDD*. 635–644.

[50] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. 2018. Graph Neural Networks: A Review of Methods and Applications. In *CoRR*, Vol. abs/1812.08434.