# Heterogeneous Graph Neural Network with Distance Encoding

## ABSTRACT

Heterogeneous graph neural network (HGNN), which employs neural network to heterogeneous graph, has shown superior performance and attracted considerable research interest. However, HGNN inherits the limitation of representational power from GNN via learning *individual* node embeddings based on their structural neighbors, largely ignoring the potential correlations between nodes and leading to sub-optimal performance. In fact, the complex correlation between nodes (e.g., distance or relation) is crucial for many graph mining tasks, such as link prediction, relation prediction, and subgraph prediction. *How to establish the correlation between node embeddings and improve the representational power of HGNN is still an open problem.* To solve the above problem, we propose heterogeneous distance encoding (HDE) and inject it into HGNN, which fundamentally improves the representational power of HGNN. Specifically, we define heterogeneous shortest path distance to describe the relative distance between nodes, and then jointly encode such distances for multiple nodes of interest to establish their correlation. After that, we inject the encoded correlation into the neighbor aggregating process of HGNN to learn more representative embedding for downstream tasks. More importantly, the proposed HDE relies only on the graph structure and ensures the inductive ability of HGNN. Significant improvements on both transductive and inductive tasks demonstrate the effectiveness of HDE in improving the representational power of HGNN.

## CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

## KEYWORDS

datasets, neural networks, gaze detection, text tagging

**ACM Reference Format:**
. 2018. Heterogeneous Graph Neural Network with Distance Encoding. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/1122445.1122456

## 1 INTRODUCTION

Heterogeneous graphs, which consist of diverse types of nodes/edges and rich semantics, have been widely used for abstracting and modeling complex systems, such as academic graphs [24, 30], transportation systems [7], drug reactions [1], and financial analysis [34]. As shown in Figure 1(a), the academic heterogeneous graph contains three types of nodes: paper, author, and term, as well as different types of relations between them. Over the past decade, traditional heterogeneous graph mining [16], especially path based methods [15, 21], has been well studied. Meta-path [21], a composite relation connecting two nodes, is used for measuring the node pair similarity. Recently, deep learning based technologies gradually extend to mine graph-structured data [11, 12, 23]. Heterogeneous graph neural network (HGNN), as a deep graph representation learning model, has shown its superiority over traditional heterogeneous graph analysis [15, 21] and aroused considerable research interest.

The prevailing HGNNs focus on how to handle the heterogeneity of graph and learn *individual* node embedding via aggregating its structural neighbors. However, such learning paradigm of HGNN inherits the limitation of representational power from GNN [27] and fails to establish the *correlation* among multiple nodes [12, 18], thus leading to sub-optimal performance on many tasks with multiple nodes, such as link prediction and relation prediction. As shown in Figure 1(b), both nodes $p_2$ and $p_3$ individually aggregate information from one author and two terms, so their embeddings will be the same embedding according to their structures (node attributes are ignored here). As a result, given two node pairs $(p_1, p_2)$ and $(p_1, p_3)$, HGNN cannot distinguish which link is more likely to exist and predicts the same existence probability to them (i.e., $\hat{y}_{p_1,p_2} = \hat{y}_{p_1,p_3}$), shown in Figure 1(c). In fact, node $p_1$ is relatively closer to node $p_2$ than node $p_3$, so link $(p_1, p_2)$ is more likely to exist. The limited representational power of HGNN may greatly hurt the performance of downstream tasks and largely restricts its applications.

One important reason causing the limited representational power of HGNN lies in individually aggregating structural neighbors, while ignoring the correlation between nodes (e.g., distance and relation), which is crucial for representational power [12, 14, 28]. Capturing the correlation among multiple nodes is extremely useful for the tasks related to more than one single node, such as link prediction, relation prediction, and subgraph prediction. For example, node pair close to each other tend to create tie (e.g., triadic closure phenomenon [17]), providing valuable information for link prediction. Recently, several works [12, 14, 28, 31] try to integrate diverse correlations into the learning process of homogeneous graph neural network. For example, DEGNN [12] utilizes the shortest path distance (*SPD*) to capture the correlation between nodes, and then inject such correlation into node embedding to improve the representational power of GNNs. However, the above correlation modeling technologies do not consider diverse types of edges, so they cannot be directly applied to heterogeneous graph.

In fact, establishing the correlation between nodes in the heterogeneous graph encounters much more challenges because there exist diverse types of connections (a.k.a, paths) between nodes on heterogeneous graph (e.g., paper-term, paper-author, and their combinations). Traditional measures (e.g., shortest path distance) cannot sufficiently measure the correlation between nodes on heterogeneous graph because they only focus on path length and

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

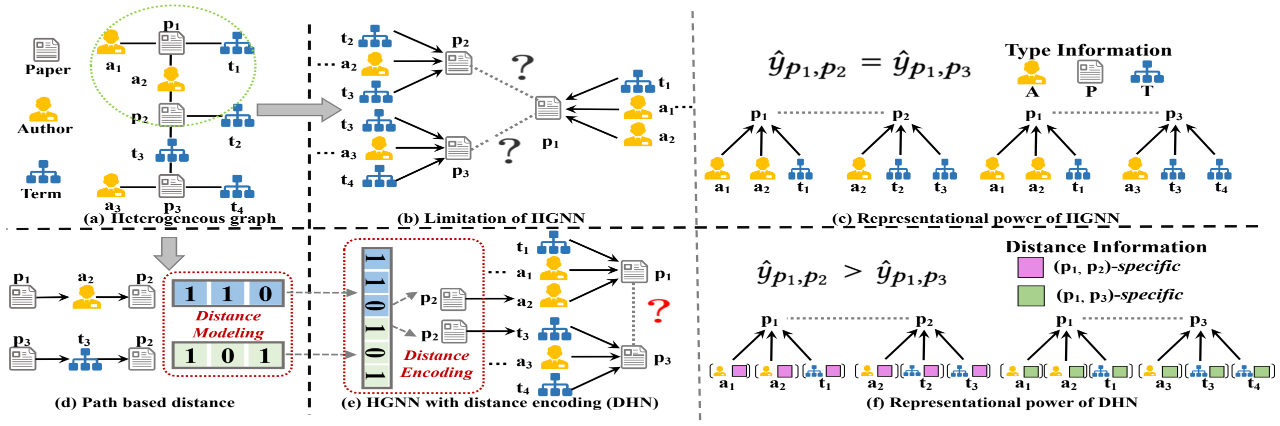


**Figure 1: An introduction to academic heterogeneous graph and the comparison between HGNN and DHN. (a) The academic heterogeneous graph contains three types of nodes and two types of edges. Different paths reveal different distances between nodes. (b) The limitation of HGNN comes from the aggregating process which individually learns node embeddings based on their structural neighbors. (c) The representational power of HGNN is limited and fails to distinguish which link should exist (i.e., $\hat{y}_{p_1,p_2} = \hat{y}_{p_1,p_3}$). (d) Node pair connected via different paths show different distances. (e) The proposed DHN injects the correlation information into the aggregating process via encoding distance between nodes. (f) The representational power of DHN is significantly improved with the help of distance information (i.e., $\hat{y}_{p_1,p_2} > \hat{y}_{p_1,p_3}$).**

largely ignore the influence of path type, leading to inappropriate results. As shown in Figure 1(d), node pair $(p_2, p_1)$ shows different correlation from node pair $(p_2, p_3)$ because they are connected via different types of paths (paper-author-paper v.s. paper-term-paper), while the traditional *SPD* [12] assigns the same relative distance to them (i.e., path length are both 2).

Deliberately considering the above analysis, to improve the representational power of HGNN via correlation modeling, we need to address the following new challenges:

•**Heterogeneity of graph.** The heterogeneity is an intrinsic property of heterogeneous graph, i.e., various types of nodes and edges. For example, different types of nodes/edges have different traits and may play different roles in a specific graph mining task. How to handle such complex heterogeneous information is an urgent problem that needs to be solved.

•**Heterogeneous correlation modeling.** The correlation between nodes is crucial for many graph mining tasks, and ignoring such correlation greatly limits the representational power of HGNN. For example, given a node pair $(u, v)$, if they both show strong correlations to node $i$ (e.g., directly linked or relatively closed), then the probability of link $(u, v)$ existing is quite high. Taking the heterogeneous graph shown in Figure 1(a) as an example, paper $p_1$ tends to establish reference relation with paper $p_2$ because they are both written by author $a_2$ and two short paths $(p_1, a_2)$ and $(a_2, p_2)$ both indicate strong correlations. As can be seen, author $a_2$ serves as the medium to establish the correlation between $p_1$ and $p_2$, providing valuable information to predict the reference relation $(p_1, p_2)$. In contrast, paper $p_1$ may not establish reference relation with paper $p_3$ because they are connected with the medium of $p_2$ via two long paths $(p_1, a_2, p_2)$ and $(p_2, t_3, p_3)$ (i.e., weak correlation). Although the effectiveness of correlation information has been verified in applications such as similar search and co-authorship prediction [21], how to incorporate the correlations between nodes into HGNN and improve its representational power is still an open problem.

•**Heterogeneous distance modeling.** Intuitively, diverse correlations between nodes can be revealed via the multiple types of paths. Considering the heterogeneity of graph, when modeling relative distance between nodes, we need to consider both path length and path type simultaneously, which is beyond the ability of traditional shortest path distance [12]. Only considering path length, while ignoring path types in heterogeneous graph may lead to improper distance measure. For example, node pair $(p_1, p_2)$ connected via $(p_1, a_2, p_2)$ indicates co-authoring (paper-author-paper) relation, while node pair $(p_3, p_2)$ connected via $(p_3, t_3, p_2)$ indicates term co-occurrence relation. Although two different paths have the same length, from the perspective of path type, path $(p_1, a_2, p_2)$ indicates larger similarity than path $(p_3, t_3, p_2)$ because co-authoring relation is more reliable than term co-occurrence relation. Therefore, simply calculating shortest path distance will weaken the semantic information provided by different types of paths. How to properly measure the relative distance between nodes on heterogeneous graph remains to be solved.

In this paper, we design heterogeneous distance encoding technology (HDE) to handle the above three challenges. Then we inject HDE into the neighbor aggregating process of HGNN, which gives a model **D**istance encoding based **H**eterogeneous graph neural **N**etwork (called DHN). Specifically, we first formulate the heterogeneous shortest path distance to measure the relative distance between nodes and design HDE via joining such distances among multiple nodes to encode their correlation. After that, the proposed DHN injects the encoded correlation into the aggregating process to learn more representative embedding for link prediction. By comparing Figure 1(b) and (e), we can see that, correlation modeling pairs the proposed DHN with significantly more representation power as opposed to previous HGNN.

The contributions of our work are summarized as follows:

•We first point out the limitaion of representational power of HGNN and show its limitation in learning structural representation. To the best of knowledge, we are the first one to observe this fundamental issue of HGNN. To enhance its representational power, we propose heterogeneous distance encoding (HDE) to capture the correlation between nodes via encoding their relative distance.

•We design a more powerful HGNN, called DHN, which injects the proposed HDE into the aggregating process to improve its representational power, and then apply it to learn more representative embedding for link prediction as a proof of concept.

•Extensive experiments on both transductive and inductive link prediction demonstrate the superiority of the proposed DHN over the state-of-the-arts with significant improvements (up to 13.3% gains in AUC). More importantly, we discover the characteristics of HDE and demystify how it improves the performance of HGNN.

## 2 RELATED WORK

Graph neural networks generalize deep learning to graph-structured data, which follows the message-passing framework to receive messages from neighbors and apply neural network to update node embedding. Michaël et al. [2] propose a spectral graph convolutional network based on $K$-order Chebyshev polynomials and GCN [11] simplifies it via a localized approximation. GraphSAGE [6] and GAT [23] design diverse aggregators to aggregate neighbor information. Recently, some works [12, 28, 31] leverage the position/location/distance to enhance the neighbor aggregating process. PGNN [28] uses SPDs between each node and pre-selected anchor sets to encode the absolute distance between nodes. But it holds worse inductive/generalization capability. SEAL [31] extracts local subgraph around each target link and maps subgraph pattern to link existence. Li et al. [12] propose distance encoding to improve the representational power of GNNs with both theoretical guarantees and empirical efficiency. However, all the above graph neural networks focus on homogeneous graphs.

Some works [3, 4, 9, 24, 29] extend GNNs to heterogeneous graphs. [24] and [30] both leverage hierarchical aggregation to capture rich semantics. MEIRec [3] and IntentGC [33] apply HGNN to solve the intent recommendation. Hu et al. [8] propose a heterogeneous graph attention network for short text classification. GTN [29] learns a soft selection of edge types and generate meta-paths automatically, solving the problem of meta-path selection. HGT [9] adopts heterogeneous mutual attention to aggregate meta relation triplet, and MAGNN [4] leverages relational rotation encoder to aggregate meta-path instances. In summary, all above HGNNs only focus on the heterogeneity of graph and rich semantic fusion, largely ignoring the correlation between nodes.

## 3 PRELIMINARY

*Definition 3.1.* **Heterogeneous Graph** [19]. A heterogeneous graph, denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, consists of an object set $\mathcal{V}$ and a link set $\mathcal{E}$. A heterogeneous graph is also associated with a node type mapping function $\phi : \mathcal{V} \rightarrow \mathcal{A}$ and a link type mapping function $\psi : \mathcal{E} \rightarrow \mathcal{R}$. $\mathcal{A}$ and $\mathcal{R}$ denote the sets of predefined object types and link types, where $|\mathcal{A}| + |\mathcal{R}| > 2$.

**Example.** Taking the heterogeneous graph shown in Figure 1 as an example, it consists of three types of nodes: paper ($P$), author ($A$), and term ($T$) (a.k.a, $\mathcal{A} = \{P, A, T\}$). Given a node $u$, we can obtain its type index $j = \phi(u)$ and corresponding node type $\mathcal{A}_j$. For example, the type index of node $p_1$ is 0 (i.e., $\phi(p_1) = 0$) and the corresponding node type is $P = \mathcal{A}_0$. Also, we can define the homogeneous graph which only consists of one type of node and edge (i.e., $|\mathcal{A}| + |\mathcal{R}| = 2$).

Then, we define the heterogeneous enclosing subgraph to extract the local structural information around the target node set remaining to predict in the specific task.

*Definition 3.2.* **Heterogeneous Enclosing Subgraph**. Given a target node set $\mathcal{S} \subset \mathcal{V}$ in the heterogeneous graph, the $k$-hop heterogeneous enclosing subgraph for $\mathcal{S}$, denoted as $\mathcal{G}^k_{\mathcal{S}}$, is the subgraph induced from the heterogeneous graph by the union of the $k$-hop neighbors of all nodes in $\mathcal{S}$.

**Example.** Taking node pair $(p_1, a_2)$ as an example ($\mathcal{S} = \{p_1, a_2\}$), the 1-hop heterogeneous enclosing subgraph $\mathcal{G}^1_{\{p_1,a_2\}}$ is shown within the green circle in Figure 1(a), which consists of $p_1$'s 1-hop neighbors $\{a_1, t_1, p_1, a_2\}$, $a_2$'s 1-hop neighbors $\{p_1, p_2, a_2\}$, as well as the induced edges. Note that the heterogeneous enclosing subgraph is not necessary to be a connected graph if nodes $u$ and $v$ are too far from each other while $k$ is relatively small.

Lastly, we briefly review the shortest path distance (*SPD*), which is designed to measure the relative distance between nodes in homogeneous graphs.

*Definition 3.3.* **Shortest Path Distance**. Given a node pair $(u, v)$, a path $\rho$ is defined as a node sequence from node $v$ to $u$, denoted as $\rho_{v \rightsquigarrow u} = (w_0, w_1, \cdots, w_p), w_0 = v, w_p = u$. Among all possible paths $\mathcal{P}_{v \rightsquigarrow u}$, the length of the shortest path (a.k.a, *SPD*), denoted as $spd(u|v) \in \mathbb{R}$, is able to capture the relative distance from node $v$ to node $u$, shown as follows:

$$spd(u|v) = \min \left\{ |\rho_{v \rightsquigarrow u}| \middle| \forall \rho_{v \rightsquigarrow u} \in \mathcal{P}_{v \rightsquigarrow u} \right\}, \tag{1}$$

where $|\rho_{v \rightsquigarrow u}|$ denotes the number of nodes in the path $\rho_{v \rightsquigarrow u}$ except the first node $v$.

**Example**. As shown in Figure 2, the shortest path $\rho_{p_1 \rightsquigarrow p_2}$ from node $p_1$ to node $p_2$ is $(p_1, a_2, p_2)$ and the corresponding $spd(p_2|p_1)$ is 2. Practically, the relative distance of a node pair revealed by *SPD* is highly related to their correlation (e.g., similarity) and has been widely used in homogeneous graph analysis [12, 28]. Intuitively, a node pair connected via the smaller *SPD* show larger similarity and tend to belong to the same community or establish a link.

## 4 HETEROGENEOUS DISTANCE ENCODING

In this section, we first propose heterogeneous shortest path distance to describe the relative distance between nodes. Then, given a target node set remaining to predict for downstream task (e.g., a node pair for link prediction), we further show how to capture the correlations between the target node set and a node via the heterogeneous distance encoding. Note that the target node set can be arbitrary in size for different tasks, such as one single node for node classification, two nodes for link/relation prediction or even more nodes for subgraph prediction and graph classification.

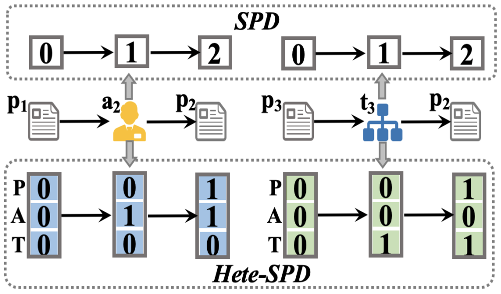


**Figure 2: An illustrative example of** *SPD* **and** *Hete-SPD* **calculating $\rho_{p_1 \rightsquigarrow p_2}$ and $\rho_{p_3 \rightsquigarrow p_2}$.** *SPD* **does not consider the path type and assigns the same distance to them (i.e.,** $spd(p_2|p_1) = spd(p_2|p_3) = 2$**). While,** *Hete-SPD* **calculates the relative distance with regard to node type in different dimensions and is able to distinguish two different paths (i.e.,** $\mathbf{d}(p_2|p_1) = [1, 1, 0]$ **and** $\mathbf{d}(p_2|p_3) = [1, 0, 1]$**).**

A simple way to model relative distance on the heterogeneous graph is to calculate the *SPD* between nodes. However, the simple *SPD* only focuses on path length and largely ignores path types, leading to sub-optimal performance on heterogeneous graph analysis. For example, node pair $(p_1, p_2)$ connected by $(p_1, a_2, p_2)$ indicates quite different correlations from node pair $(p_3, p_2)$ connected by $(p_3, t_3, p_2)$, while *SPD* will assign the same value to them.

To properly measure the relative distance between nodes on the heterogeneous graph, we formally define heterogeneous shortest path distance (*Hete-SPD*) based on the full consideration of both path length and path types, as follows:

*Definition 4.1.* **Heterogeneous Shortest Path Distance**. Given the node pair $(u, v)$, heterogeneous shortest path distance describes the relative distance from node $v$ to node $u$, denoted as $\mathbf{d}(u|v) \in \mathbb{R}^{|\mathcal{A}|}$, considering both path length and path type simultaneously. The $j$-th dimension of *Hete-SPD*, which captures the relative distance with regard to node type $\mathcal{A}_j$, is shown as follows:

$$\mathbf{d}_j(u|v) = \min\left\{ \left|\rho_{v\rightsquigarrow u}\right|_{\phi(w)=j} \middle| \forall \rho_{v\rightsquigarrow u} \in \mathcal{P}_{v\rightsquigarrow u} \right\}, \tag{2}$$

where $|\rho_{v\rightsquigarrow u}|_{\phi(w)=j}$ denotes the number of nodes belonging to type-$j$ in the path $\rho_{v\rightsquigarrow u}$ except the first node $v$.

**Example**. As shown in Figure 2(a), with the full consideration of heterogeneity, the *Hete-SPD* $\mathbf{d}(p_2|p_1)$ is $[1, 1, 0]$, indicating that the heterogeneous shortest path from node $p_1$ to node $p_2$ goes through one author ($a_2$) and one paper ($p_2$). On the other hand, the *Hete-SPD* $\mathbf{d}(p_2|p_3)$ is $[1, 0, 1]$. As can be seen, *Hete-SPD* will assign different distances to node pairs $(p_3, p_2)$ and $(p_1, p_2)$ even they share the same *SPD*s. In summary, *Hete-SPD* is significantly different from *SPD* as follows: (1) *Hete-SPD* fully considers the heterogeneity of graph and individually calculates the node-type-specific relative distance in different dimensions. Thus, *Hete-SPD* is able to comprehensively describe the relative distance between nodes in the heterogeneous graph. (2) *Hete-SPD* is an $|\mathcal{A}|$-dimensional vector, while *SPD* is a scalar. So *Hete-SPD* has a higher capacity of modeling correlation and is more powerful than *SPD*. Besides node type counting, it is possible to utilize edge type counting for *Hete-SPD* calculation, which may even further improve relation prediction in knowledge graphs [22]. We leave it as a future work.

Note that computing *Hete-SPD* is not much more complex than computing *SPD*. We may still start with associating each node with an $|\mathcal{A}|$-dimensional vector and use breadth-first search (BFS), although $|\mathcal{A}|$ many queues, each per object type, should be maintained to keep track of the BFS procedure. In practice, we will set a maximum distance (typically 2-3) to truncate each dimension of *Hete-SPD* (Eq.(5) later). So the complexity of computing truncated *Hete-SPD* for all pairs of nodes is at most $O(|\mathcal{A}| \cdot |E|)$, where $|E|$ is the total number of links. As in a heterogeneous graph, $|\mathcal{A}|$ is typically assumed to be a small constant, so there is not much more computation overhead than a standard forward pass of HGNN..

Based on the proposed heterogeneous shortest path distance, we further define **H**eterogeneous **D**istance **E**ncoding (HDE) to model the correlation between a target node set $\mathcal{S}$ and a node $i$ based on *Hete-SPD*.

*Definition 4.2.* **Heterogeneous Distance Encoding**. Given a target node set $\mathcal{S} \subset \mathcal{V}$ in heterogeneous graph, the heterogeneous distance encoding of node $i$, denoted as $\mathbf{h}_i^{\mathcal{S}}$, is the combination of *Hete-SPD* from all nodes in $\mathcal{S}$ to node $i$, as follows:

$$\mathbf{h}_i^{\mathcal{S}} = F\left( Enc\big(\mathbf{d}(i|v)\big) \middle| v \in \mathcal{S} \right), \tag{3}$$

where $F$ denotes the fusion function (e.g., sum operator), $Enc$ is the encoding function (e.g., onehot encoding).

**Example.** Given node pair $(u, v)$ remaining to predict (i.e., $\mathcal{S} = \{u, v\}$), the heterogeneous distance encoding of node $i$ with regard to node pair $\{u, v\}$ is shown as follows:

$$\mathbf{h}_i^{\{u,v\}} = F\left( Enc\big(\mathbf{d}(i|u)\big), Enc\big(\mathbf{d}(i|v)\big) \right). \tag{4}$$

Besides node pair $(u, v)$, $\mathcal{S}$ can also be a single node $v$ or triangle $(u, v, w)$ or even the entire graph. More specifically, $F$ is the concatenate operator and $Enc$ is the concatenation of element-wise onehot encoding, as follows:

$$Enc\big(\mathbf{d}(i|u)\big) = \Big\Vert_j \, onehot\left( \min\left(\mathbf{d}_j(i|u), d_j^{\max}\right) \right), \tag{5}$$

where $d_j^{\max}$ is the maximum distance to be encoded for node type $\mathcal{A}_j$, which helps prevent overfitting the noisy structure in an overly large neighborhood and decreases the computation complexity.

Deeper insight into the $(u, v)$-specific HDE of node $i$ (a.k.a., $\mathbf{h}_i^{\{u,v\}}$), we can find that node $i$ serves as the bridge to connect nodes $u$ and $v$, measuring the relative distance of node pair $(u, v)$ indirectly. Capturing relative distance is critical for graph mining tasks, especially link prediction. For example, if $\mathbf{d}(i|u)$ and $\mathbf{d}(i|v)$ both indicate the small relative distances, then we can conclude that nodes $u$ and $v$ are relatively close to each other, indicating the higher existence probability of link $(u, v)$. Note that the heterogeneity of node $i$ (i.e., node type) is also crucial for the proposed HDE. For example, two papers connected via a term $(p_1, t_3, p_2)$ or an author $(p_3, a_2, p_2)$ indicate different correlations. On the other hand, the proposed HDE is beneficial for inductive learning and does not depend on node identities, which fundamentally differs from positional node embeddings such as node2vec [5] or one-hot node identifiers. When performing prediction on unseen nodes, heterogeneous distance encoding, which purely depends on the graph structure, is able to initialize meaningful node embedding for unseen nodes. In summary, the proposed HDE ensures the inductive and generalization ability of HGNN.

## 5 HETEROGENEOUS DISTANCE ENCODING FOR LINK PREDICTION

In this section, we take link prediction as an evaluation task to verify the effectiveness of heterogeneous distance encoding. Specifically, we inject heterogeneous distance encoding into the neighbor aggregating process. The basic idea of DHN is to capture the correlations between nodes and integrates such correlation into the aggregating process of HGNN. Specifically, given a node pair $(u, v)$ remaining to predict, we calculate the heterogeneous distance encoding to capture the correlations between them and use it to initialize neighbor embeddings. After that, the proposed DHN aggregates the initial embeddings of heterogeneous neighbors and inject the correlation between nodes into the final node embedding. Lastly, we use the final embedding of the node pair to predict the probability of link $(u, v)$ existing.

### 5.1 Node Embedding Initialization

Node embedding initialization is the cornerstone of the HGNN. Significantly different from previous works [24, 30], which independently initialize node embedding, we initialize node embedding via *HDE* to capture the correlations between nodes. Specifically, given the link $(u, v)$, we utilize both $(u, v)$-specific *HDE* and heterogeneous type encoding of each node to initialize its embedding.

**Initialization via heterogeneous distance encoding.** Given the node pair $(u, v)$, we first calculate the corresponding *HDE* from node pair $(u, v)$ to the remaining nodes. Practically, we extract $k$-hop heterogeneous enclosing subgraph $\mathcal{G}^k_{\{u,v\}}$ of node pair and only calculate the *HDE* for node $i \in \mathcal{G}^k_{\{u,v\}}$. The reasons are twofold: (1) Heterogeneous shortest path calculation for the entire graph is time-consuming. Actually, the most relevant information is concentrated around the node pair. $k$-hops heterogeneous enclosing subgraph reduces the search space and accelerates distance encoding calculation. (2) The extracted subgraph can be used for mini-batch training, improving the scalability of HGNN and making it apply to industrial-sized graphs.

**Initialization via heterogeneous type encoding.** Besides heterogeneous distance encoding, we also utilize heterogeneous node type encoding to initialize node embedding, as follows:

$$\mathbf{c}_i = onehot(\phi(i)), \tag{6}$$

where $\mathbf{c}_i$ is an $|\mathcal{A}|$-dimension vector, indicating the type of node $i$. Taking node $p_1$ shown in Figure 1 as an example, its type index $j = \phi(p_1) = 0$ and the corresponding heterogeneous type encoding is $[1, 0, 0]$. Different from heterogeneous distance encoding, heterogeneous type encoding aims to capture the characteristic of different types of nodes, which is independent of node pair $(u, v)$. Lastly, we concatenate heterogeneous type encoding and heterogeneous distance encoding of node $i$, and project it via MLP as the initial embedding to predict the existence of link $(u, v)$, as follows:

$$\mathbf{e}_i^{\{u,v\}} = \sigma(\mathbf{W}_0 \cdot \mathbf{c}_i || \mathbf{h}_i^{\{u,v\}} + \mathbf{b}_0), \tag{7}$$

where $\mathbf{e}_i^{\{u,v\}}$ denotes the $(u, v)$-specific initial embedding of node $i$, $\sigma$ denotes the activation function, $\mathbf{W}_0$ and $\mathbf{b}_0$ denote the weight matrix and the bias vector, respectively. Eq. 7 actually projects raw distance information and type information as a learnable vector which can be optimized via back-propagation.

### 5.2 Heterogeneous Graph Convolution

After initializing the $(u, v)$-specific node embedding, we further design heterogeneous graph convolution to aggregate diverse types of neighbors in the heterogeneous enclosing subgraph, and update node embeddings to predict the existence of link.

Before predicting the existence of link $(u, v)$, we need to aggregate neighbor embedding to update the embeddings of both nodes $u$ and $v$. Taking node $u$ as an example, we first sample a fixed number of neighbors of node $u$ from the corresponding heterogeneous enclosing subgraph, denoted as $\mathcal{N}_u^{\{u,v\}}$. Then, the proposed DHN can obtain the neighbor based embedding of node $u$ via multi-layers aggregating, as follows:

$$\mathbf{x}_{\mathcal{N}_u,l}^{\{u,v\}} = Agg(\{\mathbf{x}_{i,l-1}^{\{u,v\}} | \forall i \in \mathcal{N}_u^{\{u,v\}}\}), \tag{8}$$

where $\mathbf{x}_{\mathcal{N}_u,l}^{\{u,v\}}$ denotes $l$-layer neighbor based embedding of node $u$, $\mathbf{x}_{i,l}^{\{u,v\}}$ denotes the embedding of node $i$ via $l$-layer aggregating and $\mathbf{x}_{i,0}^{\{u,v\}}$ is the initial node embedding $\mathbf{e}_i^{\{u,v\}}$. The previous works [9, 24] usually improve the performance of HGNN via designing a better aggregating function. Unlike them, we aim to fundamentally improve the representational power by injecting the correlation between nodes into aggregating process, rather than designing a new model architecture. So we keep the model simple and adopt the *MeanPooling* to perform aggregating process, as follows:

$$\mathbf{x}_{\mathcal{N}_u,l}^{\{u,v\}} = MeanPooling(\{\mathbf{x}_{i,l-1}^{\{u,v\}} | \forall i \in \mathcal{N}_u^{\{u,v\}}\}). \tag{9}$$

To emphasize the property of root node $u$ explicitly, we concatenate root node embedding $\mathbf{x}_{u,l-1}^{\{u,v\}}$ and neighbor based embedding $\mathbf{x}_{\mathcal{N}_u,l}^{\{u,v\}}$ to update root node embedding, as follows:

$$\mathbf{x}_{u,l}^{\{u,v\}} = \sigma(\mathbf{W}^l \cdot (\mathbf{x}_{u,l-1}^{\{u,v\}} || \mathbf{x}_{\mathcal{N}_u,l}^{\{u,v\}}) + \mathbf{b}^l), \tag{10}$$

where $\mathbf{W}^l$ and $\mathbf{b}^l$ denote the weight matrix and bias vector, respectively. After $L$-layer aggregating, we obtain the final node embedding $\mathbf{z}_u^{\{u,v\}} = \mathbf{x}_{u,L}^{\{u,v\}}$. Note that $\mathbf{z}_u^{\{u,v\}}$ is $(u, v)$-specific and only used for predicting the existence of link $(u, v)$. The same process can be done to learn the final embedding of node $v$, denoted as $\mathbf{z}_v^{\{u,v\}}$.

### 5.3 Loss Function and Optimization

Given a $(u, v)$-specific embedding of node pair (i.e., $\mathbf{z}_u^{\{u,v\}}$ and $\mathbf{z}_v^{\{u,v\}}$), we feed the concatenation of them into MLP to predict the existence of link $(u, v)$. Here we formulate link prediction as a binary classification problem, and the predict score $\hat{y}_{u,v}$ is shown

**Table 1: Statistics of the datasets.**

| Datasets | A-B | #A | #B | #A-B |
|---|---|---|---|---|
| LastFM | Artist-Tag | 1181 | 539 | 1500 |
| | User-Artist | 1496 | 1755 | 3000 |
| ACM | Term-Paper | 381 | 769 | 1500 |
| | Paper-Author | 1000 | 2527 | 3000 |
| IMDB | Movie-Actor | 3061 | 1374 | 7071 |
| | Movie-Director | 3061 | 861 | 3061 |

as follows:

$$\hat{y}_{u,v} = \sigma\left(\mathbf{W}_1 \cdot (\mathbf{z}_u^{\{u,v\}} || \mathbf{z}_v^{\{u,v\}}) + b_1\right), \tag{11}$$

where $\mathbf{W}_1$ and $b_1$ denote the weight vector and bias scalar, respectively. Finally, we calculate cross-entropy loss and optimize the model, as follows:

$$\mathcal{L} = \sum_{(u,v)\in\mathcal{E}^+\cup\mathcal{E}^-} \left(y_{u,v} \log \hat{y}_{u,v} + \left(1 - y_{u,v}\right) \log\left(1 - \hat{y}_{u,v}\right)\right), \tag{12}$$

where $(u, v) \in \mathcal{E}$ and $(u, v') \in \mathcal{E}^-$ denote the set of observed (positive) node pairs and the set of negative node pairs sampled from all unobserved node pairs, respectively.

Although we only take the link prediction task as an example to explain and verify the representational power of heterogeneous distance encoding, it can be widely used in many graph mining tasks, such as link type/relation prediction [20], hyperlink prediction and its type prediction [32], and subgraph prediction [13]. One only needs to adapt a node pair into a set of nodes. DHN should theoretically outperform traditional HGNNs. Moreover, as long as there is categorical node (or link) features, HDE can provide consistently more power than traditional distance encoding [12].

## 6 EXPERIMENT

### 6.1 Datasets

As shown in Table 1, we conduct experiments on three real-world heterogeneous graphs: **LastFM**[1] is an online music dataset, which contains three types of nodes, i.e., user (U), artist (A), tag (T), and two types of edges, i.e., artist-tag (A-T), user-artist (U-A). **ACM**[2] is a bibliographic dataset in computer science, which contains three types of nodes, i.e., author (A), paper (P), term (T), and two types of edges, i.e., paper-author (P-A), term-paper (T-P). **IMDB**[3] contains knowledge about movies, which contains three types of nodes, i.e., actor (A), movie (M), director (D), and two types of edges, i.e., movie-actor (M-A), movie-director (M-D).

### 6.2 Baselines and Implementation Details

We compare with some state-of-the-art baselines including both homogeneous GNNs (i.e., GCN, GAT, GraphSAGE, GIN, and DEGNN) and heterogeneous graph neural networks (i.e., HAN, MEIRec, and HGT), to verify the effectiveness of the proposed DHN.

•GCN [11]: It is a classical graph convolutional network which is designed for homogeneous graphs.

•GAT [23]: It is an attention-based homogeneous GNN which considers the importance of neighbor and improves the aggregating process.

•GraphSAGE (SAGE for short) [6]: It is a classical GNN which leverages sampler and aggregator to embed homogeneous graph.

•GIN [27]: It is a classical homogeneous GNN which is as powerful as the WL test with a simple aggregating function.

•DEGNN [12]: It is a homogeneous graph neural network which captures distance between the node set and improves the representation power of GNNs.

•HAN [24]: It is a HGNN which employs node-level attention and semantic-level attention simultaneously.

•MEIRec [3]: It is a HGNN which leverages heterogeneous neighbor sampling for large-scale heterogeneous graph.

•HGT [9]: It is a HGNN which aggregates information via meta relation triplet based on heterogeneous mutual attention. We remove the relative temporal encoding in HGT, because our datasets are static heterogeneous graphs.

For previous HGNNs including HAN, HGT, and MEIRec, we use heterogeneous type encoding to initialize node embedding. For homogeneous GNNs except DEGNN, we only use zero as initial node embedding. We implement the proposed DHN using Tensorflow 1.8.0. Here we randomly initialize parameters with the Gaussian distribution and leverage Adam [10] to optimize the model. For the proposed DHN, we set the learning rate to 0.01, the batch size to 32, the hops of neighbor to 2, the size of heterogeneous enclosing subgraph to 4, the $d_j^{\max}$ to 3, the number of sampled neighbor to 5, the dimension of node embedding to 128, the regularization parameter to 5e-3. For GCN, GAT, GraphSAGE, GIN, DEGNN, HAN, MEIRec, and HGT, we split exactly the same training set, validation set, and testing set to ensure fairness, and optimize their parameters using the validation set. For a fair comparison, we set the embedding dimension to 128 for all the models. We will release the code and datasets after the paper being accepted.

### 6.3 Link Prediction

Link prediction aims to infer unknown links given an observed graph structure, which has been widely used to test the generalization ability of graph neural networks. Here we adopt two types of tasks for evaluation: transductive and inductive link prediction.

*6.3.1 Transductive Link Prediction.* Considering diverse types of edges existing in heterogeneous graph, we individually perform transductive link prediction for all types of edges, providing extensive and comprehensive evaluation. Here *transductive* means that when randomly removing edges from the heterogeneous graph, we need to possibly ensure all nodes are seen in the training data. To generate negative samples, we randomly sample an equal number of node pairs from the graph which have no edge connecting them. We split the chosen edges and negative samples into validation and test. In our experiments, we test both 80%-10%-10% (8/1/1) and 60%-20%-20% (6/2/2) splitting for training, validation, and testing. Here we select AUC and accuracy as the evaluation metrics. The experimental results are shown in Table 2.

Based on Table 2, we have the following observations:

[1]https://www.last.fm/
[2]http://dl.acm.org/
[3]https://www.kaggle.com/carolzhangdc/imdb-5000-movie-dataset

**Table 2: Qantitative results on the transductive link prediction with two different splitting (e.g., 8/1/1 and 6/2/2). The larger values, the better performace. Best results are indicated in bold. The Imp.(%) indicates the percentage of improvements gained by the proposed DHN compared to the best baseline.**

| Split | Models | LastFM | | | | ACM | | | | IMDB | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | A-T | | U-A | | P-A | | T-P | | M-A | | M-D | |
| | | Acc | AUC | Acc | AUC | Acc | AUC | Acc | AUC | Acc | AUC | Acc | AUC |
| 8/1/1 | GCN | 0.5012 | 0.5031 | 0.5021 | 0.5041 | 0.5012 | 0.5043 | 0.5031 | 0.5087 | 0.5051 | 0.5098 | 0.5089 | 0.5102 |
| | SAGE | 0.7343 | 0.8324 | 0.6766 | 0.7361 | 0.7741 | 0.7781 | 0.6512 | 0.7812 | 0.5423 | 0.6323 | 0.8482 | 0.8901 |
| | GAT | 0.5054 | 0.5061 | 0.5102 | 0.5121 | 0.5012 | 0.5067 | 0.5121 | 0.5198 | 0.5101 | 0.5144 | 0.5123 | 0.5192 |
| | GIN | 0.8471 | 0.9306 | 0.7281 | 0.7919 | 0.7728 | 0.8372 | 0.7833 | 0.8232 | 0.7362 | 0.8045 | 0.8483 | 0.9374 |
| | DEGNN | 0.8532 | 0.9372 | 0.7333 | 0.8049 | 0.8512 | 0.9179 | 0.7978 | 0.8551 | 0.7857 | 0.8828 | 0.9418 | 0.9626 |
| | HAN | 0.8154 | 0.8732 | 0.7112 | 0.7812 | 0.8423 | 0.8877 | 0.8123 | 0.8931 | 0.7949 | 0.8799 | 0.9477 | 0.9584 |
| | MEIRec | 0.8427 | 0.9324 | 0.7221 | 0.8632 | 0.8408 | 0.9018 | 0.8151 | 0.8986 | 0.7906 | 0.8523 | 0.9575 | 0.9664 |
| | HGT | 0.8612 | 0.9412 | 0.7412 | 0.8195 | 0.8612 | 0.9232 | 0.8268 | 0.9099 | 0.8121 | 0.9121 | 0.9677 | 0.9723 |
| | DHN | **0.9521** | **0.9743** | **0.9623** | **0.9781** | **0.9523** | **0.9832** | **0.8531** | **0.9281** | **0.8333** | **0.9422** | **0.9912** | **0.9931** |
| | Imp. (%) | 10.6 | 3.5 | 29.8 | 13.3 | 10.6 | 6.5 | 3.2 | 2.1 | 2.6 | 3.3 | 2.4 | 2.4 |
| 6/2/2 | GCN | 0.5058 | 0.5098 | 0.5089 | 0.5078 | 0.5102 | 0.5178 | 0.5023 | 0.5056 | 0.5032 | 0.5061 | 0.5121 | 0.5198 |
| | SAGE | 0.6548 | 0.7419 | 0.5916 | 0.6332 | 0.6873 | 0.6993 | 0.5733 | 0.6263 | 0.5412 | 0.6198 | 0.7189 | 0.786 |
| | GAT | 0.5123 | 0.5191 | 0.5123 | 0.5201 | 0.5287 | 0.5302 | 0.5231 | 0.5112 | 0.5187 | 0.5298 | 0.5341 | 0.5421 |
| | GIN | 0.7789 | 0.8368 | 0.6452 | 0.7132 | 0.6857 | 0.7667 | 0.7151 | 0.7981 | 0.6812 | 0.7441 | 0.7696 | 0.8637 |
| | DEGNN | 0.7778 | 0.8493 | 0.6508 | 0.7191 | 0.7928 | 0.8441 | 0.7433 | 0.8196 | 0.7491 | 0.8422 | 0.8571 | 0.9166 |
| | HAN | 0.8011 | 0.8879 | 0.6892 | 0.7279 | 0.8258 | 0.8762 | 0.7366 | 0.8075 | 0.7607 | 0.8328 | 0.9093 | 0.9038 |
| | MEIRec | 0.7812 | 0.9098 | 0.7425 | 0.8147 | 0.8499 | 0.9105 | 0.7688 | 0.8388 | 0.7953 | 0.8517 | 0.9281 | 0.9376 |
| | HGT | 0.8312 | 0.9341 | 0.7609 | 0.8531 | 0.8812 | 0.9321 | 0.7923 | 0.8512 | 0.8381 | 0.8831 | 0.9541 | 0.9512 |
| | DHN | **0.9612** | **0.9732** | **0.9322** | **0.9767** | **0.9512** | **0.9874** | **0.8612** | **0.9151** | **0.9012** | **0.9645** | **0.9965** | **0.9901** |
| | Imp. (%) | 15.6 | 4.2 | 22.5 | 14.5 | 7.9 | 5.9 | 8.7 | 7.5 | 7.5 | 9.2 | 4.4 | 4.1 |

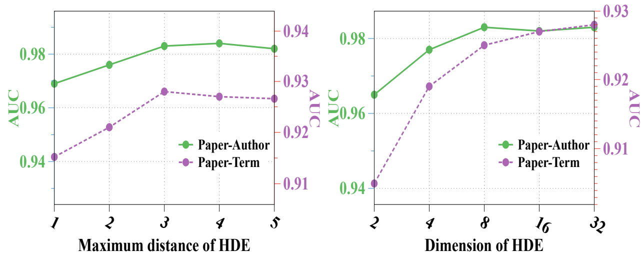
**Table 3: Qantitative results on the inductive link prediction. The larger values, the better performace. Best results are indicated in bold. The Imp.(%) indicates the percentage of improvements gained by the proposed DHN compared to the best baseline.**

| Models | LastFM | | | | ACM | | | | IMDB | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | A-T | | U-A | | P-A | | T-P | | M-A | | M-D | |
| | Acc | AUC | Acc | AUC | Acc | AUC | Acc | AUC | Acc | AUC | Acc | AUC |
| GCN | 0.5021 | 0.5091 | 0.5031 | 0.5089 | 0.5121 | 0.5152 | 0.5012 | 0.5321 | 0.5089 | 0.5102 | 0.5131 | 0.5234 |
| SAGE | 0.6311 | 0.5945 | 0.6681 | 0.7029 | 0.5722 | 0.5212 | 0.5212 | 0.5444 | 0.6132 | 0.6231 | 0.6312 | 0.6421 |
| GAT | 0.5121 | 0.5212 | 0.5091 | 0.5102 | 0.5213 | 0.5298 | 0.5132 | 0.5273 | 0.5102 | 0.5301 | 0.5202 | 0.5352 |
| GIN | 0.8033 | 0.8253 | 0.6683 | 0.7116 | 0.6023 | 0.6231 | 0.6171 | 0.6157 | 0.5512 | 0.6412 | 0.7041 | 0.7501 |
| DEGNN | 0.8197 | 0.8749 | 0.6612 | 0.7221 | 0.7221 | 0.8013 | 0.6328 | 0.7072 | 0.6723 | 0.7122 | 0.7921 | 0.8121 |
| HAN | 0.7951 | 0.8512 | 0.7885 | 0.8399 | 0.8191 | 0.8523 | 0.7891 | 0.8689 | 0.8061 | 0.8615 | 0.9022 | 0.9312 |
| MEIRec | 0.8033 | 0.8762 | 0.8028 | 0.8387 | 0.8333 | 0.9019 | 0.7732 | 0.8532 | 0.8022 | 0.8614 | 0.9373 | 0.9531 |
| HGT | 0.8344 | 0.8922 | 0.8301 | 0.8621 | 0.8512 | 0.9213 | 0.7922 | 0.8753 | 0.8112 | 0.8921 | 0.9521 | 0.9652 |
| DHN | **0.8867** | **0.9488** | **0.9268** | **0.9815** | **0.9122** | **0.9507** | **0.8205** | **0.9485** | **0.8295** | **0.9446** | **0.9819** | **0.9951** |
| Imp. (%) | 6.2 | 6.3 | 11.6 | 13.8 | 7.1 | 3.2 | 3.5 | 8.3 | 2.2 | 5.8 | 3.1 | 3.0 |

•The proposed DHN consistently performs better than all baselines with significant improvements on all datasets. When predicting the U-A relation on the LastFM dataset, the superiority of DHN is up to 29.8% and 13.3% improvements in accuracy and AUC, respectively. It demonstrates the effectiveness of heterogeneous distance encoding in modeling node pair correlation.

•Compared to the traditional GNNs which individually learn node embedding (e.g., GCN, GraphSAGE, GIN, HAN, MEIRec, and HGT), both DEGNN and DHN are able to capture the correlations between nodes via distance modeling and achieve significant better performance. To go deep into DEGNN and DHN, we find that *Hete-SPD* is more powerful than *SPD* in modeling node pair correlation on heterogeneous graph because it fully considers both path length and path type for relative distance modeling.

•With the full consideration of heterogeneity of graph, HGNNs including HAN, MEIRec, HGT, and DHN usually show superiorities over homogeneous GNNs. When predicting heterogeneous links (e.g., paper-author), node type encoding may provide potentially valuable information and improves the performance.



(a) Maximum distance of HDE (b) Dimension of HDE

**Figure 3: Effectiveness of HDE w.r.t maximum distance and dimension.**

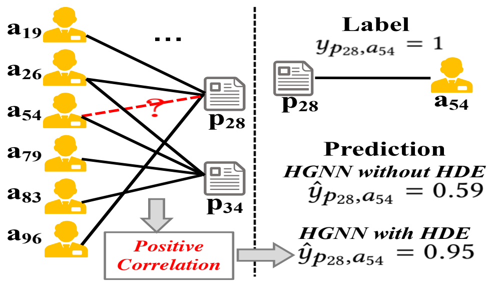


**Figure 4: A case study that shows how the proposed HDE improves the representational power of HGNN via correlation modeling (i.e., positive correlation improves the probability of link $(p_{28}, a_{54})$ existing).**

•Among all homogeneous GNNs, GAT, GCN, and GraphSAGE, which are bounded by the 1-WL test, fail to perform well. On the other hand, GIN and DEGNN are more powerful than traditional GNNs, achieving competitive performance.

*6.3.2 Inductive Link Prediction.* Besides transductive link prediction, we further conduct inductive link prediction [25, 26] to evaluate the generalization ability of the proposed DHN on unobserved links. Here *inductive* means the model should be able to efficiently predict the links associated with nodes that are not observed in the training data. Specifically, we follow two steps to split the data: 1) we use the same setting of the transductive task to first split the links into training/validation/testing sets; 2) we randomly select 20% nodes, remove any links associated with them from the training set, and remove any links not associated with them in the validation and testing sets. The experimental results of inductive link prediction are shown in Table 3.

Based on Table 3, we have the following observations:

•Similar to transductive link prediction, the proposed DHN consistently shows its superiorities over all baselines on all three datasets, verifying its strong inductive ability. In brief, the proposed DHN is able to generate meaningful embeddings for unseen nodes and predicts inductive links effectively.

•Among all models, DEGNN shows better generalization because it captures node pair correlation based solely on the graph structure via *SPD*. However, it fails to outperform the proposed DHN due to the lack of heterogeneity modeling in distance calculation.

•HGNNs including HAN, MEIRec, and HGT outperform homogeneous GNNs with significant gaps because they are able to capture the characteristics of different types of nodes, indicating the necessity of heterogeneity modeling.

## 6.4 Analysis of Heterogeneous Distance Encoding

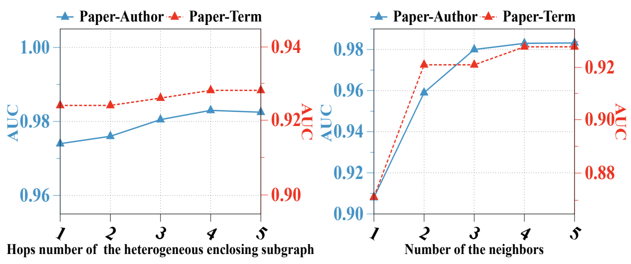
The correlation modeling via the proposed HDE plays a key role in improving the representational power of HGNN. Practically, we truncate the proposed HDE with the maximum distance to prevent overfitting and project it as the learnable vector to improve its capacity. As shown in Figure 3, we further discover the characteristic of the proposed HDE (w.r.t. maximum distance and dimension) and have the following observations: From Figure 3(a), we find that the relative distance in a small range (typically 2-3) provides valuable correlation information, and large distance may be unnecessary. Comparing the results in Figure 3(a) and those in Table 2, we can see that even with $d^{max} = 1$, DHN performs far more better than all the baselines. It makes sense because local structure around nodes is beneficial for link prediction [31] while long-range connections may introduce noise and lead to overfitting. As shown in Figure 3(b), we can see that with the growth of the dimension, the effectiveness of HDE rises first and then starts to keep stable, which means it needs enough dimension to encode the correlation between nodes and larger dimension may introduce additional redundancies.

## 6.5 Case Study

To intuitively show the effectiveness of the proposed HDE in improving the representational power of HGNN, we take paper-author pair $(p_{28}, a_{54})$[4] on ACM dataset as an example to show how paper-author prediction changes with/without HDE. Note that the link $(p_{28}, a_{54})$ actually exists in the dataset (i.e., $y_{p_{28},a_{54}} = 1$). To predict whether the link $(p_{28}, a_{54})$ exists, we first extract corresponding heterogeneous enclosing subgraph (six authors[5], two papers[6], and induced edges) and show them in Figure 4.

As shown in Figure 4, we can find that: When predicting the existence of link $(p_{28}, a_{54})$, HGNN without HDE is not sure whether the link should exist and leads to sub-optimal prediction (i.e., $\hat{y}_{p_{28},a_{54}} = 0.59$). It is because although $(p_{28}, a_{54})$ are close to each other, HGNN without HDE fails to capture such positive correlation (small distance) due to its limited representational power. In contrast, HGNN with HDE is pretty sure that link $(p_{28}, a_{54})$ should exist (i.e., $\hat{y}_{p_{28},a_{54}} = 0.95$) because the proposed HDE injects the correlation into the embedding and enables HGNN to distinguish their relative distance. So we can conclude that HDE significantly improves the representational power of HGNN and leads to better performance.

[4] $p_{28}$: Michele Berlingerio, Fabio Pinelli, Mirco Nanni, Fosca Giannotti. Temporal mining for interactive workflow data analysis, KDD'09. $a_{54}$: Fabio Pinelli.
[5] $a_{54}$, $a_{19}$: Mirco Nanni, $a_{26}$: Fosca Giannotti, $a_{79}$: Roberto Trasarti, $a_{83}$: Anna Monreale, $a_{96}$: Michele Berlingerio.
[6] $p_{28}$, $p_{34}$: Anna Monreale, Fabio Pinelli, Roberto Trasarti, Fosca Giannotti. WhereNext: a location predictor on trajectory pattern mining, KDD'09.



(a) Hop of heterogeneous enclosing subgraph.

(b) Number of neighbors.

**Figure 5: Parameter sensitivity of DHN.**

## 6.6 Parameters Study

We investigate the sensitivity of parameters on the ACM dataset and report the performance of DHN on both paper-author prediction and paper-term prediction with various parameters in Figure 5.

•**Hop of heterogeneous enclosing subgraph.** We show how different hops of heterogeneous enclosing subgraph affect the model performance. The results are shown in Figure 5(b). We can find that the performance of DHN rises slowly and achieves the best performance when the hop is set to 4. Again, comparing Figure 5(a) and Table 2, we can see that even with only one-hop enclosing subgraph, DHN has already significantly outperform all the baselines. After that, the performance of DHN keeps stable. It is because we need enough local structural information to encode the correlation between nodes. And, more hops will increase the time cost and may be unnecessary.

•**Number of the neighbor.** As illustrated in Figure 5(d), we can see that the performance of our model significantly improves as the number of neighbors grows. When sampling 5 neighbors, the proposed DHN achieves the best performance. This confirms that the neighbor information can effectively improve node embedding.

## 7 CONCLUSION

In this paper, we first point out the limitation of representational power of HGNN. After that, we study the problem of correlation modeling in the heterogeneous graph and design provably a more powerful HGNN, called DHN, which injects the correlation into the aggregating process to learn more representative embeddings. Specifically, we formulate heterogeneous shortest path distance to model the relative distance between nodes and then propose a novel heterogeneous distance encoding to encode the relative distance as the correlation between nodes. Then, we inject the proposed HDE into the aggregating process of HGNN, which gives a model DHN. Extensive experimental results on both transductive and inductive tasks verify the superiority of the proposed DHN.

## REFERENCES

[1] Monica Agrawal, Marinka Zitnik, and Jure Leskovec. 2018. Modeling Polypharmacy Side Effects with Graph Convolutional Networks. *ISMB.*

[2] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *NeurIPS.*

[3] Shaohua Fan, Junxiong Zhu, Xiaotian Han, Chuan Shi, Linmei Hu, Biyu Ma, and Yongliang Li. 2019. Metapath-guided Heterogeneous Graph Neural Network for Intent Recommendation. In *KDD.* 2478–2486.

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

[5] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *KDD.* 855–864.

[6] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NeurIPS.* 1024–1034.

[7] Huiting Hong, Yucheng Lin, Xiaoqing Yang, Zang Li, Kung Fu, Zheng Wang, Xiaohu Qie, and Jieping Ye. 2020. HetETA: Heterogeneous information network embedding for estimating time of arrival. In *KDD.* 2444–2454.

[8] Linmei Hu, Tianchi Yang, Chuan Shi, Houye Ji, and Xiaoli Li. 2019. Heterogeneous Graph Attention Networks for Semi-supervised Short Text Classification. In *EMNLP.* 4823–4832.

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

[10] Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR.*

[11] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR.*

[12] Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. 2020. Distance Encoding – Design Provably More Powerful Graph Neural Networks for Structural Representation Learning. In *NeurIPS.*

[13] Changping Meng, S Chandra Mouli, Bruno Ribeiro, and Jennifer Neville. 2018. Subgraph pattern neural networks for high-order graph evolution prediction. In *AAAI.*

[14] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. 2020. Geom-GCN: Geometric Graph Convolutional Networks. In *ICLR.*

[15] Chuan Shi, Xiangnan Kong, Yue Huang, Philip S Yu, and Bin Wu. 2014. HeteSim: A General Framework for Relevance Measure in Heterogeneous Networks. *IEEE Transactions on Knowledge and Data Engineering* (2014).

[16] Chuan Shi, Yitong Li, Jiawei Zhang, Yizhou Sun, and Philip S. Yu. 2017. A Survey of Heterogeneous Information Network Analysis. *IEEE Transactions on Knowledge and Data Engineering* 29 (2017), 17–37.

[17] Han Shi, Haozheng Fan, and James T.Kwok. 2020. Effective Decoding in Graph Auto-Encoder Using Triadic Closure. In *AAAI.* 906–913.

[18] Balasubramaniam Srinivasan and Bruno. Ribeiro. 2020. On the Equivalence between Positional Node Embeddings and Structural Graph Representations. In *ICLR.*

[19] Yizhou Sun and Jiawei Han. 2013. Mining heterogeneous information networks: a structural analysis approach. *Acm Sigkdd Explorations Newsletter* 14, 2 (2013), 20–28.

[20] Yizhou Sun, Jiawei Han, Charu C Aggarwal, and Nitesh V Chawla. 2012. When will it happen? relationship prediction in heterogeneous information networks. In *WSDM.* 663–672.

[21] 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.

[22] Komal K. Teru, Etienne Denis, and William L. Hamilton. 2020. Inductive Relation Prediction by Subgraph Reasoning. In *ICML.* 9448–9457.

[23] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR.*

[24] 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.

[25] Yanbang Wang, Yen-Yu Chang, Liu Yunyu, Jure Leskovec, and Pan Li. 2021. Inductive Representation Learning in Temporal Networks via Causal Anonymous Walks. In *ICLR.*

[26] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. 2020. Inductive representation learning on temporal graphs. In *ICLR.*

[27] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks. In *ICLR.*

[28] Jiaxuan You, Rex Ying, and Jure Leskovec. 2019. Position-aware graph neural networks. In *ICML.* 7134–7143.

[29] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and H. Kim. 2019. Graph Transformer Networks. In *NeurIPS.* 11983–11993.

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

[31] Muhan Zhang and Yixin Chen. 2018. Link Prediction Based on Graph Neural Networks. In *NeurIPS.*

[32] Muhan Zhang, Zhicheng Cui, Shali Jiang, and Yixin Chen. 2018. Beyond link prediction: Predicting hyperlinks in adjacency space. In *AAAI.*

[33] Jun Zhao, Zhou Zhou, Ziyu Guan, Wei Zhao, Wei Ning, Guang Qiu, and Xiaofei He. 2019. IntentGC: a Scalable Graph Convolution Framework Fusing Heterogeneous Information for Recommendation. In *KDD.* 2347–2357.

[34] Qiwei Zhong, Yang Liu, Xiang Ao, Binbin Hu, Jinghua Feng, Jiayu Tang, and Qing He. 2020. Financial Defaulter Detection on Online Credit Payment via Multi-view Attributed Heterogeneous Information Network. In *WWW.*

**Table 4: List of hyperparameters and their value / range.**

| Hyperparameters | Value / Range | Notes |
|---|---|---|
| Batch size | 1, 32, 64, 128 | |
| Optimizer | SGD, Adam | |
| Activation | ReLU, Elu | |
| Learning rate | 0.005, 0.01 | |
| Seed | 0,1,2,3,4 | Random seed for initialization |
| Dropout | 0, 0.2 | Dropout rate in MLP |
| Hidden dim. | 32, 64, 128 | Dimension of hidden layer in MLP |
| Layers | 1, 2, 3 | Number of aggregating layer of GNNs |
| Neighbors | 1, 2, 3, 4, 5 | Number of sampled neighbors for GraphSAGE and DHN |
| Hops | 1, 2, 3, 4, 5 | Hops of enclosing subgraph for DEGNN and DHN |
| $d^{max}$ | 1, 2, 3, 4, 5 | Maximum of distance for DEGNN and DHN |
| Heads | 8 | Number of attention head for GAT and HAN |

## A DETAILS OF THE EXPERIMENTS

### A.1 Baseline Details

We take both homogeneous GNNs and HGNN as baselines. We first introduce the implementation of baselines and then discuss hyper-parameters tuning.

Regarding homogeneous GNN models, GCN is implemented according to Equation (9) of [11] with self-loops added. GraphSAGE is implemented according to Algorithm 1 of Section 3.1 in [6][7]. Mean pooling is used as the neighborhood aggregation function. GAT layer is adopted from the code released in the paper [23][8]. GIN is implemented by adapting the code provided by the original paper [27][9], where we use the sum-pooling aggregation and multi-linear perception to aggregate neighbors. In all three baselines described above, ReLU nonlinearities are applied to the output of each hidden layer, followed by a Dropout layer. PGNN layer is implemented by adapting the code provided by the original paper [28][10]. DEGNN is implemented by adapting the code provided by the original paper [12][11]. As we focus on learning structural representation with inductive capability, all the homogeneous GNNs use zero as input features. We use the heterogeneous type encoding (i.e., node type) as the initial features. Regarding heterogeneous GNN models, HAN layer is adopted from the code released in [24][12]. For MEIRec, we use the code provided by the original paper [3][13] and select average function to aggregate neighbors. And, HGT [9] is adopted from the original implement released in GitHub[14].

### A.2 Hyperparameters Tuning

Table 4 lists the most important hyperparameters' at a glance, which applies to both the baselines and our proposed models. Grid search is used to find the best hyperparameters combination. The models are sufficiently trained till the cross entropy loss converges and we report the best model over different random seeds.

[7]https://github.com/tkipf/gcn
[8]https://github.com/PetarV-/GAT
[9]https://github.com/weihua916/powerful-gnns
[10]https://github.com/JiaxuanYou/P-GNN
[11]https://github.com/snap-stanford/distance-encoding
[12]https://github.com/Jhy1993/HAN
[13]https://github.com/googlebaba/KDD2019-MEIRec
[14]https://github.com/acbull/pyHGT