HGK-GNN: Heterogeneous Graph Kernel based Graph Neural Networks

Qingqing Long*
Key Laboratory of Machine Perception (Ministry of Education), Peking University
qingqinglong@pku.edu.cn

Lingjun Xu*
Key Laboratory of Machine Perception (Ministry of Education), Peking University
xlj_rk@pku.edu.cn

Zheng Fang
Key Laboratory of Machine Perception (Ministry of Education), Peking University
fang_z@pku.edu.cn

Guojie Song†
Key Laboratory of Machine Perception (Ministry of Education), Peking University
gjsong@pku.edu.cn

ABSTRACT

While Graph Neural Networks (GNNs) have achieved remarkable results in a variety of applications, recent studies exposed important shortcomings in their ability to capture heterogeneous structures and attributes of an underlying graph. Furthermore, though many Heterogeneous GNN (HGNN) variants have been proposed and have achieved state-of-the-art results, there are limited theoretical understandings of their properties. To this end, we introduce graph kernel to HGNNs and develop a Heterogeneous Graph Kernel-based Graph Neural Networks (HGK-GNN). Specifically, we incorporate the Mahalanobis distance (MD) to build a Heterogeneous Graph Kernel (HGK), and incorporating it into deep neural architectures, thus leveraging a heterogeneous GNN with a heterogeneous aggregation scheme. Also, we mathematically bridge HGK-GNN to metapath-based HGNNs, which are the most popular and effective variants of HGNNs. We theoretically analyze HGK-GNN with the indispensable *Encoder* and *Aggregator* component in metapath-based HGNNs, through which we provide a theoretical perspective to understand the most popular HGNNs. To the best of our knowledge, we are the first to introduce HGK into the field of HGNNs, and mark a first step in the direction of theoretically understanding and analyzing HGNNs. Correspondingly, both graph and node classification experiments are leveraged to evaluate HGK-GNN, where HGK-GNN outperforms a wide range of baselines on six real-world datasets, endorsing the analysis.

CCS CONCEPTS

• **Networks** → *Network structure*; • **Information systems** → *Collaborative and social computing systems and tools*; • **Theory of computation** → *Kernel methods.*

*These authors contributed equally to the work.
†Corresponding Author.

KDD '21, August 14–18, 2021, Virtual Event, Singapore.

ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467429

KEYWORDS

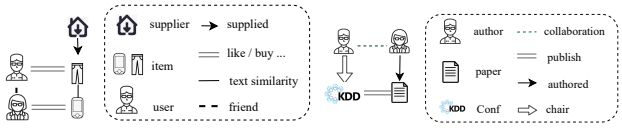
Heterogeneous Graph Convolutional Network, Graph Kernels, Heterogeneous Networks

ACM Reference Format:
Qingqing Long, Lingjun Xu, Zheng Fang, and Guojie Song. 2021. HGK-GNN: Heterogeneous Graph Kernel based Graph Neural Networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3447548.3467429

1 INTRODUCTION

Graph Neural Networks (GNNs) have obtained tremendous success in mining information networks [14, 17, 25, 28], along with sorts of significant theoretical analysis [26, 40]. One of the milestones is that the most widely studied GNNs can be generalized to the mechanism of message passing scheme, which is at most as discriminative as the 1-Weisfeiler-Lehman Test [40]. These theoretical achievements in turn facilitate the design of GNNs.

Since substantial amounts of GNNs merely assume strong homogeneous input, i.e. one node type and one edge type, they fail to generalize to networks with heterogeneity. Heterogeneity is a more general and common principle of many real-world networks, whereby linked nodes often belong to different types, so do the links. The two networks in Fig. 1 are typical examples, both of which are made up of three types of nodes and four kinds of links corresponding to specific entities and relations in practice. Also, in a heterogeneous graph, each node and each link may be associated with some features, and attributes of different types may have different dimensions and semantic meanings.



(a) E-commerce Network (b) Academic Network

Figure 1: HGK-GNN examples: Heterogeneous graph with various node types / features and edge type / features.

Many Heterogeneous GNNs (HGNNs) variants with different heterogeneous neighborhood encoding and aggregation schemes have

been proposed [12, 23, 31, 38] and have empirically achieved state-of-the-art performance in many tasks such as node classification and link prediction. However, these works bear several drawbacks. First, most of them overly rely on hand-craft pre-defined heterogeneous patterns or domain knowledge, such as popular metapaths in HGNNs [9, 38]. In addition, the design of almost all HGNNs is based on empirical intuition, heuristics, and experimental trial-and-error [9, 12, 38]. There is no theoretical analysis or framework to guide and understand the properties, limitations, and design of HGNNs.

We resort to *Graph Kernels* (GKs), a parallel line of work on graph mining. GKs are classical methods to measure similarities between pairwise structures in graphs such as nodes and subgraphs [16, 33, 44]. Since GKs inherently involve comparisons between substructure patterns, several previous works use GKs as instrumental tools in designing and analyzing homogeneous GNNs [5, 21]. Nevertheless, it remains to be a challenge to deal with the heterogeneous structures and attributes in heterogeneous graphs using GKs.

In this work, we propose Heterogeneous Graph Kernel (HGK) based Graph Neural Network (HGK-GNN). Along with our proposed HGK, this paper mark a first step in the direction of theoretical understanding HGNNs. Specifically, we first incorporate Mahalanobis distance (MD) to build a Heterogeneous Graph Kernel (HGK), thus capturing the heterogeneity. To leverage edge information, we adopt the tensor product of an edge and its endpoint node. Based on HGK, we then derive the deep neural architecture following the route introduced in [21]. We also mathematically bridge HGK-GNN to the encoding and aggregating process in metapath-based HGNNs, through which we provide a theoretical perspective to understand existing HGNNs. Correspondingly, we carry out experiments on both graph and node classification on six real-world datasets. HGK-GNN outperforms various baselines, which coincides with the theoretical analysis and endorses its versatility.

We summarize our contributions as follows:

- We propose HGK-GNN, a heterogeneous graph kernel (HGK) based GNN model. We propose an HGK to model heterogeneous patterns and attributes. To the best of our knowledge, we are the first to introduce GKs in the field of HGNNs.
- We derive the HGK to corresponding GNN architectures for computing its kernel mapping, which makes it free from hand-craft pre-defined heterogeneous patterns, such as popular meta-paths in HGNNs, and meanwhile with outstanding performance.
- We theoretically show that HGK-GNN can be viewed as a mixture of *Encoder* and *Aggregator* in metapath-based HGNNs, which are indispensable components in such models. Thus we mark the first step in theoretical understanding HGNNs, along with the aid of HGK.
- Graph and node classification tasks are carried out, where HGK-GNN outperforms various strong baselines, coinciding with the theoretical analysis.

2 RELATED WORK

2.1 Theoretical Achievements on Homogeneous Graph Neural Networks

GNNs, most of which focus on homogeneous graphs, have flourished in recent years [14, 17, 37, 39]. These successful GNN models largely belong to the category of message-passing GNNs that aggregate information from local neighborhoods. In this way, not only can they leverage graph structure and attributes, but is also more efficient in computation.

[26, 40] show that the ability of message-passing GNNs to model structures in graphs is upper bounded by the 1-Weisfeiler-Lehman Isomorphism Test (1-WL Test). Following these analyses, GIN is designed in [40] to approach the limited bound. Henceforth, there are sorts of works to theoretically analyze the properties, limitations, and drawbacks of GNNs [2, 7].

2.2 Heterogeneous Graph Neural Networks

Heterogeneous graphs have long been appealing because of their capability to model interconnected objects of multiple types [41]. In recent years, inspired by the success of message-passing methods in homogeneous networks, there arose several works that bring GNNs in mining heterogeneous graphs.

Some attempts simply use multiple Neural Networks to aggregate different node-type / edge-type aware neighborhood respectively and combine these embeddings to get the final representations [4, 31, 43]. Another common and competitive paradigm defines metapaths and uses them to capture the complicated heterogeneous network structures. For example, HAN [38] adopts two-level attention to learn the importance of neighbors connected by the same kind of metapath and the importance of different metapaths. MAGNN [12] goes a further step by encoding metapath instances before the hierarchical attention, and thus retaining the information of intermediate nodes along the metapath. There are also efforts to avoid a fixed set of metapaths like [15, 42], which utilize multiple transformer layers to learn high-order "soft" metapaths.

These methods, however, to a large extent, depend on pre-defined hand-craft heterogeneous patterns, domain knowledge, and heuristics. Unlike homogeneous GNNs, there lack theoretical analyses on the representational capacity of heterogeneous GNNs.

2.3 Graph Kernels (GKs)

Kernel methods have been studied and applied extensively in machine learning [32]. They evaluate the similarity between pairs of data samples using kernel functions. During this process, the data samples are implicitly projected onto higher dimensional spaces (namely the *Reproducing Kernel Hilbert Space*, RKHS) and therefore they are endowed with richer features to help with classification.

Graph Kernels (GKs), i.e. kernel methods on graphs, calculate the pairwise similarity between nodes or graphs by decomposing them into basic structural units, such as random walks [16], subtrees [33], and shortest paths [3]. These kernels forego the possibility to accommodate to graphs with node / edge types and continuous attributes, which is common in heterogeneous graphs. [18, 27] make extensions from different perspectives to overcome this limitation.

The idea of decomposition, along with the connections between kernels and learning theory, facilitate analysis of the ability of GKs to express graph structures [5, 13, 20, 24]. Due to their appealing theoretical properties, there have also been noticeable efforts in fusing GKs with GNNs. [5, 10, 21] derive GNN architectures for

various graph kernels. However, precious little works of GKs are carried out on heterogeneous GNNs and graphs.

2.4 Mahalanobis Distance (MD)

Mahalanobis distance, abbreviated MD, is a similarity metric for multi-variate data[1]. It considers the correlations between variate from different domains by multiplying the inverse covariance matrix. Mahalanobis distance is introduced to the field of distance metric learning owing to its unique form, replacing the inverse covariance matrix with learnable ones.

Mahalanobis distance has shown its effectiveness in different research areas, such as computer vision [8, 34] and recommendation systems [36]. The abundant heterogeneous information in our scenario inspires us to take Mahalanobis distance into account. Chen et al. [6] use a Mahalanobis distance like edge-type specific loss to learn shallow node embeddings designed for link prediction. However, to the best of our knowledge, there is no existing work that uses Mahalanobis distance in heterogeneous GNNs.

Some prior works like [1, 11, 35] attempt to combine Mahalanobis distance with kernel methods. Nevertheless, these Mahalanobis distance-based kernels are mainly used in SVM, while in our model we fuse the learnable distance metric with graph kernels.

3 PRELIMINARIES

3.1 Heterogeneous Graphs

Definition 1. Heterogeneous Graph. A heterogeneous graph $G = (V, E, t_V, t_E, f_V, f_E)$ is a graph in which nodes and edges are respectively associated with the type mapping functions $t_V(v) : V \to \mathcal{A}$ and $t_E(e) : E \to \mathcal{R}$. $\mathcal{A}$, $\mathcal{R}$ are sets of node and edge types in G, where $|\mathcal{A}| + |\mathcal{R}| > 2$. $f_V(v) : V \to \mathbb{R}^{|f_V(v)|}$ and $f_E(e) : E \to \mathbb{R}^{|f_E(e)|}$ are the feature mapping function of nodes and edges, respectively. $|f_V(v)|$ and $|f_E(e)|$ are the feature length of node v and edge e. We donate e_{u_1,u_2} as an edge in a graph, with node u_1 and u_2 as two endpoints.

Definition 2. Heterogeneous Graph Representation Learning. Given a heterogeneous graph $G = (V, E, t_V, t_E, f_V, f_E)$, the task is to design a model $F(\Theta)$ with parameters Θ to learn d-dimensional embeddings $\Phi(G) \in \mathbb{R}^{|V|\times d} (d \ll |V|)$ that are able to encode heterogeneous semantic information from both nodes and edges. The node embeddings can be utilized in various graph mining tasks, such as link prediction and node classification.

3.2 Graph Kernels

Given two graphs $G_1 = (V_1, E_1, f_{V_1})$ and $G_2 = (V_2, E_2, f_{V_2})$, a graph kernel function $K(G_1, G_2)$ returns a similarity measure between G_1 and G_2 through the following formula:

$$K(G_1, G_2) = \sum_{u_1 \in V_1} \sum_{u_2 \in V_2} \kappa_{base}\left(l_{G_1}(u_1), l_{G_2}(u_2)\right) \tag{1}$$

where $l_G(u)$ denotes a set of local structures centered at node u in graph G, and κ_{base} is a base kernel computing the similarity between two sets of substructures. For simplicity we may omit $l_G(u)$ and rewrite Eqn. 1 as:

$$K(G_1, G_2) = \sum_{u_1 \in V_1} \sum_{u_2 \in V_2} \kappa_{base}(u_1, u_2) \tag{2}$$

as long as the substructure set is clearly stated. We use uppercase letter $K(G_1, G_2)$ to denote graph kernels, $\kappa(u_1, u_2)$ to denote node kernels, and lowercase $k(x, y)$ to denote general kernel functions.

The kernel mapping of a kernel ψ maps a data point into the corresponding RKHS $\mathcal{H}$. Formally, given a kernel $k_*(\cdot,\cdot)$, then the following equation holds for its kernel mapping ψ_*,

$$\forall x_1, x_2, k_*(x_1, x_2) = \langle \psi_*(x_1), \psi_*(x_2) \rangle_{\mathcal{H}_*}, \tag{3}$$

where $\mathcal{H}_*$ is the RKHS of $k_*(\cdot,\cdot)$, which is commonly of infinite dimensionality.

We briefly introduce Random Walk Graph Kernel below.

Walk Kernel. A l-walk kernel $K^{(l)}_{walk}$ compares all length l walks starting from each node in two graphs G_1, G_2,

$$\begin{aligned} \kappa^{(l)}_{walk}(u_1, u_2) &= \sum_{w_1 \in \mathcal{W}^l(G_1,u_1)} \sum_{w_2 \in \mathcal{W}^l(G_2,u_2)} \prod_{i=0}^{l-1} \langle f(\boldsymbol{w}_1^{(i)}), f(\boldsymbol{w}_2^{(i)}) \rangle, \\ K^{(l)}_{walk}(G_1, G_2) &= \sum_{u_1 \in V_1} \sum_{u_2 \in V_2} \kappa^{(l)}_{walk}(u_1, u_2), \end{aligned} \tag{4}$$

where $\boldsymbol{w}_1^{(i)}$ denotes the i-th node in walk $\boldsymbol{w}$.

4 MODEL: HGK-GNN

In this section, we introduce our Heterogeneous Graph Kernel (HGK) based Graph Neural Network (HGK-GNN). Fig. 2 gives an overview. HGK-GNN is derived from our proposed HGK, which introduces Mahalanobis Distance to capture the heterogeneous semantic info from various types / features of nodes and edges.

4.1 Heterogeneous Graph Kernels (HGK)

4.1.1 Incorporating Heterogeneous Edges. We begin with defining the *tensor product* as a conjoint heterogeneous node and edge pairs. The tensor product of two vector a and b is calculated as $a \otimes b = ab^T \in \mathbb{R}^{|a|\times|b|}$. To incorporate graph kernels to our heterogeneous edge features, we first define the kernel of two neighbors, $(u_1, e_{\cdot,u_1})$ and $(u_2, e_{\cdot,u_2})$. For the central node u connected to v by a edge $e_{u,v}$, the corresponding neighbor feature is defined as $f((v, e_{u,v})) = f(v) \otimes f(e_{u,v})$. The calculated tensor contains all bilinear combinations of the two edge features, and serves as a conjoined feature. Then the neighbor kernel is defined as the inner product of the neighbor tensors, i.e.

$$\begin{aligned} \kappa((u_1, e_{\cdot,u_1}), (u_2, e_{\cdot,u_2})) &= \langle f((u_1, e_{\cdot,u_1}), f(u_2, e_{\cdot,u_2})) \rangle \\ &= \langle f(u_1), f(u_2) \rangle \cdot \langle f(e_{\cdot,u_1}), f(e_{\cdot,u_2}) \rangle. \end{aligned} \tag{5}$$

4.1.2 Incorporating Heterogeneous Nodes. Different types of nodes and edges have different heterogeneous feature spaces[12, 38]. Inspired by the significant results and applications of Mahalanobis distance (MD) in the field of distance metric learning [1, 6, 36], we introduce the Mahalanobis Distance into our graph kernels on heterogeneous networks, to capture their separated distributions, along with the measuring the distances among different types. Mahalanobis distance considers the correlation among variables from different domains, and compensates for the deviation drawbacks

[1] https://en.wikipedia.org/wiki/Mahalanobis_distance

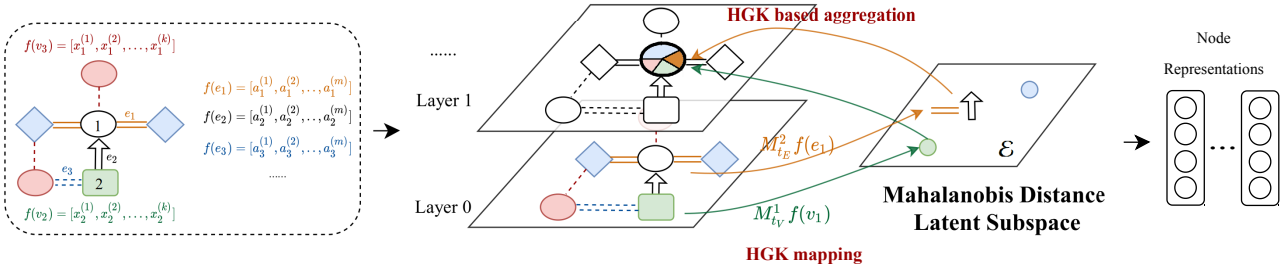


Figure 2: An overview of our proposed Heterogeneous graph kernel based Graph Neural Networks (HGK-GNN). Heterogeneous nodes and edges are mapped into our proposed Mahalanobis Distance Latent Subspace through our proposed Heterogeneous graph kernel (HGK). Then such mappings are aggregated based on the neural architecture derived from HGK to get node representations.

of Euclidean Distance in measuring multiple variables[2]. We derive a Heterogeneous Graph Kernel (HGK) based on Mahalanobis Distance. Specifically, we design the a type-specific transformation matrix $\boldsymbol{M}_1 \in \mathbb{R}^{|f_V(u_1)|\times|f_V(u_2)|}$ and $\boldsymbol{M}_2 \in \mathbb{R}^{|f_E(e_1)|\times|f_E(e_2)|}$to measure the distance of heterogeneous features from heterogeneous nodes and edges.

$$\kappa_M((u_1, e_{\cdot,u_1}), (u_2, e_{\cdot,u_2})) = \langle f(u_1), f(u_2)\rangle_{M_1} \cdot \langle f(e_{\cdot,u_1}), f(e_{\cdot,u_2})\rangle_{M_2},$$

and we define $\langle f(u_1), f(u_2)\rangle_M$ as Mahalanobis graph kernel, which can be written as

$$\langle f(u_1), f(u_2)\rangle_{M_1} = f(u_1)^T \boldsymbol{M}_1 f(u_2),$$

$$\langle f(e_{\cdot,u_1}), f(e_{\cdot,u_2})\rangle_{M_2} = f(e_{\cdot,u_1})^T \boldsymbol{M}_2 f(e_{\cdot,u_2}).$$

Based on the heterogeneous neighbor kernel, a kernel of two l-hop neighborhoods with central node u_1 and u_2 can be defined as $K_M^{(l)}(u_1, u_2) =$

$$\begin{cases} \langle f(u_1), f(u_2)\rangle_{M_1}, & l = 0 \\ \langle f(u_1), f(u_2)\rangle_{M_1} \cdot & \\ \quad \sum_{v_1 \in N(u_1)} \sum_{v_2 \in N(u_2)} K_M^{(l-1)}(v_1, v_2) \cdot & \\ \quad \langle f(e_{\cdot,v_1}), f(e_{\cdot,v_2})\rangle_{M_2} & l > 0 \end{cases} \tag{6}$$

By regarding the lower-hop kernel $\kappa_M^{(l-1)}(u_1, u_2)$, as the inner product of the $(l-1)$-th hidden representations of u_1 and u_2. Furthermore, we derive the l-hop *Heterogeneous Graph Kernel* as

$$K_M^l(G_1, G_2) = \sum_{\boldsymbol{w}_1 \in \mathcal{W}^l(G_1)} \sum_{\boldsymbol{w}_2 \in \mathcal{W}^l(G_2)} \left(\prod_{i=0}^{l-1} \langle f(\boldsymbol{w}_1^{(i)}), f(\boldsymbol{w}_2^{(i)})\rangle_M \times \prod_{i=0}^{l-2} \langle f(e_{\boldsymbol{w}_1^{(i)}, \boldsymbol{w}_1^{(i+1)}}), f(e_{\boldsymbol{w}_2^{(i)}, \boldsymbol{w}_2^{(i+1)}})\rangle_M \right), \tag{7}$$

where $\mathcal{W}^l(G)$ denotes the set of all walk sequences of length l in graph G, and $\boldsymbol{w}_1^{(i)}$ denotes the i-th node in sequence $\boldsymbol{w}_1$[3].

[2]https://en.wikipedia.org/wiki/Mahalanobis_distance

[3]A Weisfeiler-Lehman kernel can also be defined by adopting the *graph relabeling* process, which is detailedly introduced in [21]. We skip this part due to the limited space.

4.2 From HGK to Neural Architectures

Following the route introduced in [21] with the above Random Walk kernel, corresponding neural network architecture of our proposed Heterogeneous Graph Kernel can be derived as

$$\begin{aligned} h^{(0)}(v) =& \boldsymbol{W}_{t_V(v)}^{(0)} f(v) \\ h^{(l)}(v) =& \boldsymbol{W}_{t_V(v)}^{(l)} f(v) \odot \sum_{u \in N(v)} (\boldsymbol{U}_{t_V(v)}^{(l)} h^{(l-1)}(u) \\ & \odot \boldsymbol{U}_{t_E(e_{u,v})}^{(l)} f(e_{u,v})) \qquad 1 < l \le L \end{aligned} \tag{8}$$

where $\odot$ is the element-wise product and $h^{(l)}(v)$ is the cell state vector of node v. The parameter matrices $\boldsymbol{W}_{t_V(v)}^{(l)}$, $\boldsymbol{U}_{t_V(v)}^{(l)}$ and $\boldsymbol{U}_{t_E(e_{u,v})}^{(l)}$ are learnable parameters associated with types of nodes and edges.

Then we use the mean embeddings of nodes to get its graph-level embedding as

$$\Phi(G_i) = \sum_{u \in G_i} \frac{1}{|G_i|} h^{(L)}(v). \tag{9}$$

where $|G_i|$ is the number of nodes in the i-th graph. This architecture further enjoys a similar property to the architecture described in [21]. And we have the following theorem:

Theorem 1. Consider a deep graph kernel NN with L layers. Let the final output state $\Phi(G) = \sum_{u \in G} \frac{1}{|G|} h^{(L)}(v)$.

For $l = 1, \cdots, L$:

(1) $h^{(l)}(v)[i]$ as a function of input v and graph G belongs to the RKHS of kernel $K_M^l(\cdot,\cdot)$;

(2) $\Phi(G)[i]$ belongs to the RKHS of kernel $K_M^L(\cdot,\cdot)$

Proof. Following the proof pipeline in [21], the theorem can be easily proved based on mathematical induction. Due to space limitations, we provide the detailed proof process in the appendix. □

We use semi-supervised multiclass classification as the objective function. We then evaluate the cross-entropy error over all labeled examples

$$\mathcal{L} = -\sum_{l \in \mathcal{Y}_L} \sum_{f=1}^{F} Y_{lf} \ln Z_{lf}, \tag{10}$$

where $\mathcal{Y}_L$ is the set of node indices that have labels in the node classification task, or set of graph indices that have labels in the graph classification task. Z_{lf} denotes the prediction of labels output by a linear layer with an activation function, inputing $h^{(l)}(v)$ in node classification task, and $\Phi(G_i)$ in graph classification task.

5 THEORETICAL ANALYSIS

In this section, we mathematically bridge HGK-GNN to metapath-based HGNNs, which are the most popular and effective variants of HGNNs. Specifically, we theoretically analyze HGK-GNN with the indispensable *Encoder* and *Aggregator* component in metapath-based HGNNs, through which we provide a theoretical perspective to understand the most popular HGNNs.

We begin with the formal definition of relative concepts.

Definition 3. Metapath. A metapath P of length l is a path in the form of $A_0 \xrightarrow{R_1} A_1 \xrightarrow{R_2} \cdots \xrightarrow{R_l} A_l$ (abbreviated as $A_0A_1\cdots A_l$), where $A_i \in \mathcal{A}$ for $i = 0, 1, \ldots, l$ and $R_i \in \mathcal{R}$ for $i = 1, 2, \ldots, l$. A metapath can describe a composite relation $R = R_1 \circ R_2 \circ \cdots \circ R_l$ between node type A_0 and A_l.

We denote $\mathbf{P}$ as the set of all considered metapaths in a heterogeneous graph. For instance, in Fig. 1 (b), we consider $\mathbf{P} = \{Author - Paper - Author(APA), Author - Paper - Conference - Paper - Author(APCPA)\}$ and P can be either (APA) or $(APCPA)$.

Definition 4. Metapath Instance. Given a heterogeneous graph G and a metapath P, a metapath instance of P is a path $p = u_0 \xrightarrow{e_{u_0,u_1}} u_1 \xrightarrow{e_{u_1,u_2}} \cdots \xrightarrow{e_{u_{l-1},u_l}} u_l$ (abbreviated as $u_0u_1\cdots u_l$) following the schema of P. For simplicity, we denote a node $v \in p$ if $v \in \{v_0, v_1, \ldots, v_l | v_i \in P\}$.

And we denote set of all metapath instances of P as $\mathcal{P}$. For example, if P refers to the metapath ACA, then $\mathcal{P} = \{p = u_0u_1u_2 | t_V(u_0) = A, t_V(u_1) = C, t_V(u_2) = A\}$.

Definition 5. Metapath-Based Heterogeneous Graph Neural Networks (MP-HGNN). MP-HGNNs are a class of heterogeneous graph neural networks with metapath-based aggregators. HAN[38] and MAGNN[12] are two typical examples of MP-HGNN. MP-HGNN can be summarized as follows:

$$\begin{aligned} h^{(0)}(v) &= \Gamma^1_{t_V(v)} f_V(v) \\ h_p^{(l)}(v) &= ENCODE(p(v), \{h^{(l-1)}(u) | u \in p(v)\}) \\ h_P^{(l)}(v) &= AGGREGATE_1(\{h_p^{(l)}(v) | p \in \mathcal{P}\}) \\ h^{(l)}(v) &= AGGREGATE_2(\{h_P^{(l)}(v) | P \in \boldsymbol{P}\}), \end{aligned} \tag{11}$$

where $\Gamma_{t_V(v)}$ are learnable matrices related to node types and $p(v)$ is a metapath instance starting from v. $ENCODE$, $AGGREGATE_1$, $AGGREGATE_2$ are functions which vary in different MP-HGNN models. Take HAN [38] as an example, $ENCODE$ outputs the feature of the node connected to v via the metapath instance $p(v)$ and two aggregate functions are node- and semantic-level attention. And $p(v)$ is fixed in $ENCODE$ among layers.

Theorem 2. Let $\mathbf{P}_L$ denote the set of all possible metapaths of length L. Given an MP-HGNN where $AGGREGATE_1$ and $AGGREGATE_2$ are both SUM aggregators, and $ENCODE$ is defined as:

$$\begin{aligned} \boldsymbol{o}_L &= h(u_L) \\ \boldsymbol{o}_{l-1} &= h(u_{l-1}) \odot (\boldsymbol{o}_l \odot \Gamma^2_{t_E(e_{u_{l-1},u_l})} f_E(e_{u_{l-1},u_l})), l = 1, 2, \ldots, L \\ ENCODE(p(v), \{h(u) | u \in p(v)\}) &= \frac{\boldsymbol{o}_0}{L}, \end{aligned}$$

where $p(v) = vu_1 \cdots u_L$, then this MP-HGNN is a special case in the framework of L-layer HGK-GNN.

Proof. Denote set of walks with length L starting from v as $\Phi_L(v)$.

$$\begin{aligned} &HGK - GNN(v) \\ =&\boldsymbol{W}^{(L)}_{t_V(v)} f_V(v) \odot \sum_{u_1 \in N(v)} \boldsymbol{U}^{(L)}_{t_V(v)} ((\cdots \odot \sum_{u_{L-1} \in N(u_{L-2})} \boldsymbol{U}^{(1)}_{t_V(u_{L-2})} \\ &(\boldsymbol{W}^{(1)}_{t_V(u_{L-1})} f_V(u_{L-1}) \odot \sum_{u_L \in N(u_{L-1})} (\boldsymbol{U}^{(1)}_{t_V(u_{L-1})} \boldsymbol{W}^{(0)}_{t_V(u_L)} f_V(u_L) \odot \\ &\boldsymbol{U}^{(1)}_{t_E(e_{u_{L-1},u_L})} f_E(e_{u_{L-1},u_L}))) \odot \boldsymbol{U}^{(2)}_{t_E(e_{u_{L-2},u_{L-1}})} f_E(e_{u_{L-2},u_{L-1}}))) \odot \\ &\cdots)) \odot \boldsymbol{U}^{(L)}_{t_E(e_{v,u_1})} f_E(e_{v,u_1})) \end{aligned} \tag{12}$$

$$\begin{aligned} &MP - HGNN(v) \\ =&AGGREGATE_2(\{AGGREGATE_1 \\ &(\{ENCODE(p(v), \{h(u) | u \in p(v)\}) | p \in \mathcal{P}\}) | P \in \boldsymbol{P}_L\}) \\ =&\sum_{P \in \boldsymbol{P}_L} \sum_{p \in \mathcal{P}} (\frac{1}{L} h(v) \odot ((h(u_1) \odot ((\cdots \odot ((h(u_{L-1}) \odot (h(u_L) \odot \\ &\Gamma^2_{t_E(e_{u_{L-1},u_L})} f_E(e_{u_{L-1},u_L}))) \odot \Gamma^2_{t_E(e_{u_{L-1},u_{L-1}})} f_E(e_{u_{L-2},u_{L-1}}))) \odot \\ &\cdots)) \odot \Gamma^2_{t_E(e_{v,u_1})} f_E(e_{v,u_1}))) \\ =&\sum_{p \in \Phi_L(v)} (\frac{1}{L} \Gamma^1_{t_V(v)} f_V(v) \odot \Gamma^2_{t_E(e_{v,u_1})} f_E(e_{v,u_1}) \odot \Gamma^1_{t_V(u_1)} f_V(u_1) \odot \\ &\cdots \odot \Gamma^2_{t_E(e_{u_{L-1},u_L})} f_E(e_{u_{L-1},u_L}) \odot \Gamma^1_{t_V(u_L)} f_V(u_L)) \\ =&\frac{1}{L} \Gamma^1_{t_V(v)} f_V(v) \odot \sum_{u_1 \in N(v)} ((\cdots \odot \sum_{u_{L-1} \in N(u_{L-2})} (\Gamma^1_{t_V(u_{L-1})} f_V(u_{L-1}) \odot \\ &\sum_{u_L \in N(u_{L-1})} (\Gamma^1_{t_V(u_L)} f_V(u_L) \odot \Gamma^2_{t_E(e_{u_{L-1},u_L})} f_E(e_{u_{L-1},u_L}))) \odot \\ &\Gamma^2_{t_E(e_{u_{L-2},u_{L-1}})} f_E(e_{u_{L-2},u_{L-1}}))) \odot \cdots)) \odot \Gamma^2_{t_E(e_{v,u_1})} f_E(e_{v,u_1})) \end{aligned} \tag{13}$$

Let $\boldsymbol{I}$ denotes the identity matrix. We show that $MP{-}HGNN(v) = HGK - GNN(v)$, when the parameters of HGK-GNN take special values as follows:

$$\begin{aligned} \boldsymbol{W}^{(L)}_{t_V(v)} &= \frac{1}{L} \Gamma^1_{t_V(v)}, \\ \boldsymbol{W}^{(i)}_{t_V(v)} &= \Gamma^1_{t_V(v)}, \qquad i = 0, 1, \ldots, L-1 \\ \boldsymbol{U}^{(i)}_{t_V(v)} &= \boldsymbol{I}, \quad \boldsymbol{U}^{(i)}_{t_E(e_{u,v})} = \Gamma^2_{t_E(e_{u,v})}, i = 1, 2, \ldots, L \end{aligned}$$

□

Remarks. The MP-HGNN defined above is a variant of MAGNN with RotateE encoder [12], which is one of the best-performing models. We would like to point out that this analysis can be generalized to any MP-HGNNs.

Corollary 1. Consider the MP-HGNN defined in Theorem 2. There exist parameters $\Theta = \{\boldsymbol{W}, \boldsymbol{U}\}$ so that for any given (G, v_1, v_2), $\delta(MP-HGNN(v_1), MP-HGNN(v_2)) \geq$
$\delta(HGK-GNN(v_1;\Theta), HGK-GNN(v_2;\Theta))$. Here the $\delta(\cdot)$ is the Dirac function such that $\delta(a_1, a_2) = 1$ if $a_1 = a_2$ and 0 otherwise. Thus we conclude that HGK-GNN is at least as powerful as MP-HGNN.

6 EXPERIMENTS

In this section, we introduce our empirical evaluations on HGK-GNN. We first introduce experimental settings, before presenting experimental results. Specifically, our evaluations consist of:

- Contrastive quantitative evaluations, including graph classification and node classification on real-world graph datasets.
- Self-evaluations, including analysis on running efficiency and model parameters.

6.1 Experimental Setup

We first introduce the datasets, comparison methods as well as settings for experimental evaluation.

6.1.1 Datasets. We validate HGK-GNN on graph and node classification tasks. All datasets introduced below are heterogeneous graphs with node / edge types, node / edge features.

Graph Classification. In the graph classification task, all datasets consist of multiple heterogeneous separate subgraphs. Each subgraph is assigned a label. Detailed statistical description of the datasets are shown in table 1. Specifically, we generate the datasets as follows.

- **Cuneiform** is composed of graphs representing 30 Hittite cuneiform signs collected by [19]. Nodes are classified according to the role they play in a wedge. Node feature is constituted by the node's glyph type and 3-dimension coordinates. The feature of a link contains the relative position between two wedges and indicates whether an arrangement edge or not.
- **nr-AR** contains molecular graphs sampled from Tox21 library[4]. Each graph is labeled according to the compound's activity in an AR nuclear receptor signaling assay. The type of a node represents the kind of the atom. Node feature is made up of the atom's coordinates and some other physical attributes. Each link is a chemical bond between two atoms with bond type and stereochemistry as its feature.
- **sr-ARE** is also a subset of Tox21 library. Graph labels are determined by the molecule's activity in an ARE stress pathway assay. Details about nodes and edges are the same as nr-AR.

Node Classification. In the node classification task, one type of nodes in each network is to be classified. Statistics of the datasets are summarized in table 2.

- **Alibaba-user** is collected from trading records on the e-commerce platform Alibaba[5]. We use characteristics such as age, star, and occupation to represent a user and label it using gender. The attribute of an item contains information about its property, price, etc. Each shop is represented by its review and scoring level. The link between a user and an item indicates that the user bought the item, with some information about the advertisement of the item shown to the user before the deal. There exists a link between two items if they belong to the same brand. The feature of a link between an item and a shop contains the item's sales, collected, and presentation level. The feature of items and the link between items are respectively reduced to 128 and 32 dimensions using PCA.
- **Alibaba-item** is also a portion of trading records from Alibaba. The way of constructing the dataset is almost the same as Alibaba-user, except that we label the items with their categories on the platform.
- **DBLP** is a subset of the computer science bibliography website[6]. Each author is represented by a bag-of-words of their paper keywords and is labeled according to their research area. The feature of a term is a vector pre-trained by GloVe [29]. We use the concatenation of the one-hot encoding of the conference itself and its category as its feature. There exists a link between two nodes only if they are connected by at least one paper. The feature of a link is the average bag-of-words representation of the corresponding papers' title, which is reduced to 128 dimensions using PCA.

6.1.2 Baselines. We compare against different kinds of network embedding models.

For **Graph Classification**, the baselines are as follow.

- **Graph Kernels**, including the Random walk kernel [16], the Weisfeiler-Lehman subtree kernel [33] and Subgraph Matching Kernel [18]. The former two models only utilize node types, while subgraph matching kernel can simultaneously take account of the types and features of nodes and edges choosing appropriate base kernels.
- **Homogeneous GNNs**, including three most popular GNNs, which are GCN [17], GAT [37] and GIN [40].
- **Heterogeneous GNNs**. Inspired by [22], we use RGIN, which is a variant of R-GCN [31] fusing GIN.

For **Node Classification**, we use the following categories of existing methods as baselines.

- **Shallow Graph Embedding Models**, including DeepWalk [30] and Metapath2Vec [9], which are representatives in homogeneous and heterogeneous graphs.
- **Homogeneous GNNs**, including GCN [17], GAT [37] and GIN [40].
- **Heterogeneous GNNs**, including two most popular heterogeneous GNNs, which are HAN [38] and MAGNN [12]. In addition, we compare RGIN to HGK-GNN.

Note that two popular HGNNs, HAN and MAGNN, are not able to be conducted on the graph classification task, due to the limitation of pre-defined meta-paths and designed model structures. They require that all meta-paths must exist in each subgraph, and the input subgraphs must have the same nodes, which are not suitable for the heterogeneous graph classification task.

[4] https://tripod.nih.gov/tox21/challenge/data.jsp
[5] https://tianchi.aliyun.com/competition/entrance/231647/information
[6] https://dblp.uni-trier.de/

Dataset	# Graph	Avg. # node	# node type	# Node feature	Avg. # edge	# edge type	# Edge feature	# Class
Cuneiform	267	21.27	4	6	44.80	7	3	30
nr-AR	1,520	20.40	27	4	21.21	41	2	2
sr-ARE	4,392	16.93	43	4	17.26	79	2	2

Table 1: Dataset statistics for graph classification.

Dataset	Field	Nodes			Edges			# Class
		Node type	Number	Feature size	Edge type	Number	Feature size	
Alibaba-user	E-commerce	User (U)	6,492	6	IS	2,388	3	U:2
		Item (I)	1,975	128	II	137,540	32	
		Shop (S)	1,300	6	IU	6,611	2	
Alibaba-item	E-commerce	User (U)	6,637	8	IS	2420	3	I:12
		Item (I)	1,995	128	II	140,590	32	
		Shop (S)	1,312	6	IU	6,758	2	
DBLP	Academic	Author (A)	4,057	334	AA	3,528	128	A:4
		Term (T)	7,723	50	AT	90,685	128	
		Conference (C)	20	24	AC	9,205	128	

Table 2: Dataset statistics for node classification. We use the same train and test split indices for all methods.

6.1.3 Hyperparameter Settings. We take 64-dimensional embeddings for all methods. The parameter settings of other baselines follow the recommended settings in relevant codes. For Subgraph Matching Kernel, we use gaussian kernel for node / edge features and Dirac kernel for node types. For GCN, GAT and GIN, we take 2-layer supervised networks with a hidden layer sized 100. For DeepWalk, the walk length is set to 40. Due to limited GPU memory (16GB) on a single Nvidia P100, we select the most important pre-defined metapaths for HAN and MAGNN. Types of matapaths are set to UIU, IUI and ACA in Alibaba-user, Alibaba-item and DBLP. For HGK-GNN, we take the number of layer in the HGK l as 2, use Adam as the optimizer, and the learning rate is set to 0.001.

6.2 Quantitative Evaluations

In this section we present results both on graph and node classification tasks.

6.2.1 Graph Classification. Graph classification has always been a popular method for comparing the capability of kernel functions and their variants [5]. To verify the effectiveness of HGK-GNN, we conduct graph classification tasks on three heterogeneous datasets. Results are listed in Table 3. We make the following findings.

- Compared with graph kernels, HGK-GNN gains more than 50% improvement on Cuneiform and about 2% on the other two datasets. Thus we conclude that our proposed heterogeneous graph kernel leveraging node / edge types and features is able to capture richer semantics in heterogeneous graphs.
- HGK-GNN, as well as RGIN, perform better than homogeneous GNNs. Such improvements endorse the effectiveness of our heterogeneous GNN structure.
- Compared with RGIN, our model is also able to achieve better performance on all datasets. Since RGIN is a model with edge type and node features, the amelioration shows the superiority of HGK-GNN which considers edge features and note types.

6.2.2 Node Classification. We then carry out node classification experiments. We take macro and micro F1 scores for evaluation. We make the following conclusions based on the results in Table 4.

- HGK-GNN outperforms all comparison baselines across different training proportions and datasets, which demonstrates the strength of HGK-GNN in terms of node classification.
- All GNN-based methods perform consistently better than DeepWalk and Metapath2Vec. That is to say, such network structures considering features can facilitate node embedding.
- Compared with homogeneous and other heterogeneous GNNs, HGK-GNN achieves the best performance on all datasets. The results illustrate the importance to incorporate edge information in the message-passing schema.

6.3 Model Analysis

We carry out efficiency and parameter analysis to provide better understandings of HGK-GNN.

6.3.1 Efficiency. We compare the efficiency of HGK-GNN both on graph and node classification tasks with different competitors, including graph kernel methods, shallow Heterogeneous Network Embedding model, Homogeneous GNNs, and Heterogeneous GNNs. We train all models and report the time needed for convergence on a single machine equipped with a GPU with 12GB RAM.

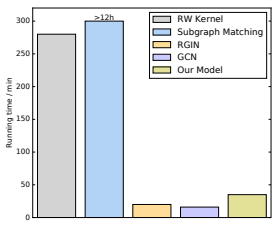
Results are presented in Fig.3. It can be shown that HGK-GNN is comparable with popular graph kernels, e.g. RW Kernel and Subgraph Matching Kernel. For the node classification task, it takes less than three times the time of MAGNN. The results demonstrate that our model is highly efficient and thus scalable.

Model	Cuneiform				nr-AR				sr-ARE			
	Macro-f1		Micro-f1		Macro-f1		Micro-f1		Macro-f1		Micro-f1	
	20%	80%	20%	80%	20%	80%	20%	80%	20%	80%	20%	80%
Random Walk Kernel	1.91	2.03	4.98	6.57	74.31	74.95	81.50	83.50	42.77	42.94	74.12	74.66
Weisfeiler-Lehman Kernel	2.42	2.79	5.16	7.12	76.40	80.97	84.82	88.08	42.84	43.02	74.95	75.51
Subgraph Matching Kernel	2.58	14.51	3.29	15.09	>12h	>12h	>12h	>12h	>12h	>12h	>12h	>12h
GCN	12.49	23.61	16.43	27.30	72.06	72.48	82.15	82.84	43.33	45.86	74.78	76.08
GIN	12.40	23.57	16.64	27.72	71.86	72.14	81.97	82.53	43.41	44.78	74.82	75.80
GAT	13.67	24.02	16.88	28.10	72.53	72.68	83.02	83.11	44.59	45.97	75.92	76.36
RGIN	14.10	25.25	17.84	30.19	**79.22**	81.27	86.80	87.99	50.34	50.96	75.06	75.97
HGK-GNN	**15.23**	**27.93**	**18.04**	**32.91**	77.88	**82.96**	**87.45**	**88.97**	**52.20**	**52.37**	**76.42**	**76.54**

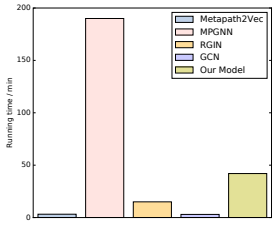
Table 3: Macro-f1 and Micro-f1 scores on graph classification task. We use same train and test split indices for all methods.

Model	Alibaba-user				Alibaba-item				DBLP			
	Macro-f1		Micro-f1		Macro-f1		Micro-f1		Macro-f1		Micro-f1	
	20%	80%	20%	80%	20%	80%	20%	80%	20%	80%	20%	80%
DeepWalk	36.02	38.94	73.45	74.83	17.12	19.42	32.09	34.34	68.27	69.34	70.02	70.28
Metapath2Vec	39.13	40.05	74.28	74.45	17.68	13.99	28.44	30.81	85.72	86.63	87.77	88.13
GCN	41.78	41.12	75.34	75.44	30.93	34.20	64.74	66.67	70.05	75.10	74.32	76.40
GIN	41.8	41.14	75.33	75.39	30.96	32.33	64.79	67.12	71.36	74.57	73.70	74.04
GAT	42.86	42.67	75.89	75.93	35.59	38.94	68.23	69.75	74.85	78.93	76.37	77.21
RGIN	42.81	42.11	75.26	75.53	37.94	40.58	70.75	72.88	87.24	87.65	88.50	89.10
HAN	43.49	43.54	76.45	76.89	69.42	70.12	87.38	90.40	89.19	90.92	90.69	91.13
MAGNN	43.51	43.54	76.80	76.97	70.21	72.30	88.17	91.29	90.15	91.14	90.85	92.04
HGK-GNN	**43.61**	**43.73**	**77.17**	**77.62**	**71.23**	**73.54**	**89.36**	**91.54**	**90.43**	**91.44**	**91.26**	**92.27**

Table 4: Macro-f1 and Micro-f1 scores on node classification task. We use same train and test split indices for all methods.

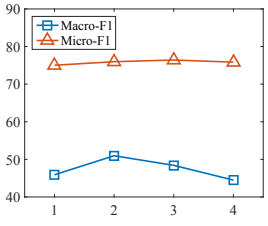


(a) Graph Classification on sr-ARE

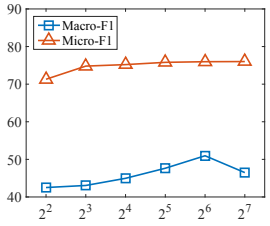


(b) Node Classification on DBLP

Figure 3: The running time on different datasets.



(a) layers of the HGK l



(b) hidden size $|h^{(l)}(u)|$

Figure 4: Parameter analysis of HGK-GNN.

6.3.2 Parameter Analysis. We analyze two parameters in our model, layers of the HGK l, and the embedding size $|h^{(l)}(u)|$. We conduct node classification experiments on sr-ARE dataset, whose results are shown in Fig. 4. From Fig. 4 (a), we can draw the conclusion that $l = 2$ would be the most appropriate, which means one or two order neighborhood information is most associated with the central node. From Fig. 4 (b), we find that larger embedding size improves the performance of HGK-GNN, but over-length hidden size does not necessarily lead to better performance. For HGK-GNN, a size of 64 is appropriate.

7 CONCLUSION

In this paper, we develop HGK-GNN, which is the first to incorporate Heterogeneous Graph Kernel (HGK) into HGNNs. Specifically, we proposed a Heterogeneous Graph Kernel, and derived it upon feature mappings to neural architectures. Correspondingly, extensive experiments are leveraged to evaluate HGK-GNN, where HGK-GNN outperforms a wide range of baselines on real-world datasets, endorsing the analysis.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (Grant No. 61876006 and No. 61572041).

REFERENCES

[1] Shigeo Abe. 2005. Training of support vector machines with Mahalanobis kernels. In *International Conference on Artificial Neural Networks*. Springer, 571–576.
[2] Vikraman Arvind, Frank Fuhlbrück, Johannes Köbler, and Oleg Verbitsky. 2020. On Weisfeiler-Leman invariance: subgraph counts and related graph properties. *J. Comput. System Sci.* (2020).
[3] Karsten M Borgwardt and Hans-Peter Kriegel. 2005. Shortest-path kernels on graphs. In *Fifth IEEE international conference on data mining (ICDM)*. IEEE, 8–pp.
[4] Yukuo Cen, Xu Zou, Jianwei Zhang, Hongxia Yang, Jingren Zhou, and Jie Tang. 2019. Representation learning for attributed multiplex heterogeneous network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1358–1368.
[5] Dexiong Chen, Laurent Jacob, and Julien Mairal. 2020. Convolutional Kernel Networks for Graph-Structured Data. *arXiv preprint arXiv:2003.05189* (2020).
[6] Hongxu Chen, Hongzhi Yin, Weiqing Wang, Hao Wang, Quoc Viet Hung Nguyen, and Xue Li. 2018. PME: projected metric embedding on heterogeneous networks for link prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1177–1186.
[7] Zhengdao Chen, Lei Chen, Soledad Villar, and Joan Bruna. 2020. Can graph neural networks count substructures? *arXiv preprint arXiv:2002.04025* (2020).
[8] Taylor Denouden, Rick Salay, Krzysztof Czarnecki, Vahdat Abdelzad, Buu Phan, and Sachin Vernekar. 2018. Improving reconstruction autoencoder out-of-distribution detection with mahalanobis distance. *arXiv preprint arXiv:1812.02765* (2018).
[9] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 135–144.
[10] Simon S Du, Kangcheng Hou, Russ R Salakhutdinov, Barnabas Poczos, Ruosong Wang, and Keyulu Xu. 2019. Graph neural tangent kernel: Fusing graph neural networks with graph kernels. In *Advances in Neural Information Processing Systems*. 5723–5733.
[11] Mathieu Fauvel, Jocelyn Chanussot, Jon Atli Benediktsson, and Alberto Villa. 2013. Parsimonious Mahalanobis kernel for the classification of high dimensional data. *Pattern Recognition* 46, 3 (2013), 845–854.
[12] Xinyu Fu, Jiani Zhang, Ziqiao Meng, and Irwin King. 2020. MAGNN: Metapath Aggregated Graph Neural Network for Heterogeneous Graph Embedding. In *Proceedings of The Web Conference 2020*. 2331–2341.
[13] Thomas Gärtner, Peter Flach, and Stefan Wrobel. 2003. On graph kernels: Hardness results and efficient alternatives. In *Learning theory and kernel machines*. Springer, 129–143.
[14] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in neural information processing systems*. 1024–1034.
[15] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous graph transformer. In *Proceedings of The Web Conference 2020*. 2704–2710.
[16] U Kang, Hanghang Tong, and Jimeng Sun. 2012. Fast random walk graph kernel. In *Proceedings of the SIAM international conference on data mining*. SIAM, 828–838.
[17] Thomas Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference of Learning Representations*.
[18] Nils Kriege and Petra Mutzel. 2012. Subgraph matching kernels for attributed graphs. *arXiv preprint arXiv:1206.6483* (2012).
[19] Nils M. Kriege, Matthias Fey, Denis Fisseler, Petra Mutzel, and Frank Weichert. [n.d.]. Recognizing Cuneiform Signs Using Graph Based Methods. In *Proceedings of The International Workshop on Cost-Sensitive Learning*. 31–44.
[20] Nils M Kriege, Christopher Morris, Anja Rey, and Christian Sohler. 2018. A Property Testing Framework for the Theoretical Expressivity of Graph Kernels.. In *IJCAI*. 2348–2354.
[21] Tao Lei, Wengong Jin, Regina Barzilay, and Tommi Jaakkola. 2017. Deriving Neural Architectures from Sequence and Graph Kernels. In *International Conference on Machine Learning*. 2024–2033.
[22] Xin Liu, Haojie Pan, Mutian He, Yangqiu Song, Xin Jiang, and Lifeng Shang. 2020. Neural subgraph isomorphism counting. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1959–1969.
[23] Qingqing Long, Yilun Jin, Guojie Song, Yi Li, and Wei Lin. 2020. Graph Structural-topic Neural Network. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1065–1073.
[24] Qingqing Long, Yilun Jin, Yi Wu, and Guojie Song. 2021. Theoretically Improving Graph Neural Networks via Anonymous Walk Graph Kernels. *arXiv preprint arXiv:2104.02995* (2021).
[25] Qingqing Long, Yiming Wang, Lun Du, Guojie Song, Yilun Jin, and Wei Lin. 2019. Hierarchical Community Structure Preserving Network Embedding: A Subspace Approach. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 409–418.
[26] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. 2019. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 4602–4609.
[27] Marion Neumann, Roman Garnett, Christian Bauckhage, and Kristian Kersting. 2016. Propagation kernels: efficient graph kernels from propagated information. *Machine Learning* 102, 2 (2016), 209–245.
[28] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. 2016. Learning convolutional neural networks for graphs. In *International conference on machine learning*. 2014–2023.
[29] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.
[30] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 701–710.
[31] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*. Springer, 593–607.
[32] John Shawe-Taylor, Nello Cristianini, et al. 2004. *Kernel methods for pattern analysis*. Cambridge university press.
[33] Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. 2011. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research* 12, 9 (2011).
[34] Feroz Ahmed Siddiky, Mohammed Shamsul Alam, Tanveer Ahsan, and Mohammed Saifur Rahim. 2007. An efficient approach to rotation invariant face detection using PCA, generalized regression neural network and Mahalanobis distance by reducing search space. In *2007 10th international conference on computer and information technology*. IEEE, 1–6.
[35] Genyun Sun, Xueqian Rong, Aizhu Zhang, Hui Huang, Jun Rong, and Xuming Zhang. 2019. Multi-scale mahalanobis kernel-based support vector machine for classification of high-resolution remote sensing images. *Cognitive Computation* (2019), 1–8.
[36] Thanh Tran, Renee Sweeney, and Kyumin Lee. 2019. Adversarial Mahalanobis Distance-Based Attentive Song Recommender for Automatic Playlist Continuation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 245–254.
[37] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint:1710.10903* (2017).
[38] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous graph attention network. In *The World Wide Web Conference*. 2022–2032.
[39] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying Graph Convolutional Networks. In *International Conference on Machine Learning*. 6861–6871.
[40] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How Powerful are Graph Neural Networks?. In *International Conference on Learning Representations*.
[41] Carl Yang, Yuxin Xiao, Yu Zhang, Yizhou Sun, and Jiawei Han. 2020. Heterogeneous Network Representation Learning: Survey, Benchmark, Evaluation, and Beyond. *arXiv preprint arXiv:2004.00216* (2020).
[42] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. 2019. Graph transformer networks. In *Advances in Neural Information Processing Systems*. 11983–11993.
[43] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V Chawla. 2019. Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 793–803.
[44] Zhen Zhang, Mianzhi Wang, Yijian Xiang, Yan Huang, and Arye Nehorai. 2018. Retgk: Graph kernels based on return probabilities of random walks. In *Advances in Neural Information Processing Systems*. 3964–3974.

8 APPENDIX

8.1 Proof of Theorem 1

Proof. We first prove the conclusion without considering edges, then the network architecture is constructed as the following,

$$\begin{aligned} h^{(0)}(v) =& \boldsymbol{W}^{(0)}_{t_V(v)} f(v) \\ h^{(l)}(v) =& \boldsymbol{W}^{(l)}_{t_V(v)} f(v) \odot \sum_{u \in N(v)} \boldsymbol{U}^{(l)}_{t_V(v)} h^{(l-1)}(u) \qquad 1 < l \leq L \end{aligned} \tag{14}$$

Suppose

$$h^l(v)[i] = \langle \phi^l_G(v), \psi^l_i \rangle_M,$$

We prove this by induction on l.

When $l = 1$, let $\phi^l_G(v) = f(v), \psi^l_i = \boldsymbol{W}^{(0)}_{t_V(v)i}$, so that the equation is satisfied.

When $l > 1$, we have

$$\begin{aligned} h^{l+1}(v)[i] =& \boldsymbol{W}^{(l+1)}_{t_V(v)} f(v) \odot \sum_{u \in N(v)} \boldsymbol{U}^{(l+1)}_{t_V(v)} h^{(l-1)}(u) \\ =& \langle \boldsymbol{W}^{(l+1)}_{t_V(v)i}, f(v) \rangle \odot \sum_{u \in N(v)} \sum_k U^{(l+1)}_{t_V(v)ik} h^{(l-1)}(u)[k] \\ =& \langle \boldsymbol{W}^{(l+1)}_{t_V(v)i}, f(v) \rangle \odot \sum_{u \in N(v)} \sum_k U^{(l+1)}_{t_V(v)ik} \langle \phi^l_G(u), \psi^l_k \rangle_M \\ =& \langle \boldsymbol{W}^{(l+1)}_{t_V(v)i}, f(v) \rangle \odot \sum_{u \in N(v)} \sum_k U^{(l+1)}_{t_V(v)ik} (\phi^l_G(u))^T M \psi^l_k \\ =& \langle \boldsymbol{W}^{(l+1)}_{t_V(v)i}, f(v) \rangle \odot \sum_{t \in t_V(v)} \sum_k \sum_{\{u | t_V(u) = t \wedge u \in N(v)\}} U^{(l+1)}_{tik} \cdot \\ & (\phi^l_G(u))^T M \psi^l_k \\ =& \langle \boldsymbol{W}^{(l+1)}_{t_V(v)i}, f(v) \rangle \odot \sum_{t \in t_V(v)} \sum_k U^{(l+1)}_{tik} \cdot \\ & \sum_{\{u | t_V(u) = t \wedge u \in N(v)\}} (\phi^l_G(u))^T M \psi^l_k \\ =& \langle \boldsymbol{W}^{(l+1)}_{t_V(v)i}, f(v) \rangle \odot \sum_{t \in t_V(v)} \langle U^{(l+1)}_{ti\cdot} \psi^l, \\ & \sum_{\{u | t_V(u) = t \wedge u \in N(v)\}} (\phi^l_G(u))^T M \rangle \\ =& \langle \boldsymbol{W}^{(l+1)}_{t_V(v)i}, f(v) \rangle \odot \langle \text{concat}_t \left(U^{(l+1)}_{ti\cdot} \psi^l \right), \\ & \text{concat}_t \left(\sum_{\{u | t_V(u) = t \wedge u \in N(v)\}} (\phi^l_G(u))^T M \right) \rangle \\ =& \langle \boldsymbol{W}^{(l+1)}_{t_V(v)i} \otimes \text{concat}_t \left(U^{(l+1)}_{ti\cdot} \psi^l \right), f(v) \otimes \\ & \text{concat}_t \left(\sum_{\{u | t_V(u) = t \wedge u \in N(v)\}} (\phi^l_G(u))^T M \right) \rangle. \end{aligned} \tag{15}$$

Let

$$\phi^{l+1}_G(v) = f(v) \otimes \text{concat}_t \left(\sum_{\{u | t_V(u) = t \wedge u \in N(v)\}} (\phi^l_G(u))^T M \right),$$

$$\psi^{l+1}_i = \boldsymbol{W}^{(l+1)}_{t_V(v)i} \otimes \text{concat}_t \left(U^{(l+1)}_{ti\cdot} \psi^l \right),$$

we have

$$h^{l+1}(v)[i] = \langle \phi^{l+1}_G(v), \psi^{l+1}_i \rangle_M.$$

Thus Theorem 1 (1) is proved.

Through constructing a directed chain $L_{n,k} = (V, E)$ from model parameters, with nodes $V = \{l_n, l_{n-1}, \cdots, l_0\}$ and $E = \{(v_{i+1}, v_i)\}$. The underlying mapping of l_j is ψ^L_i, thus we have

$$\begin{aligned} h_G[i] =& \sum_v h^L(v)[i] \\ =& \sum_{v \in G} \langle \phi^{l+1}_G(v), \psi^{l+1}_i \rangle_M \\ =& \sum_{v \in G} \sum_{v' \in Ln,k} K^{(L)}_M(v, v') \\ =& K_M(G, L_{n,k}) \end{aligned}$$

Now Theorem 1 (2) is also proved. □