Graph-Level Heterogeneous Information Network Embeddings for Cardholder Transaction Analysis

Farouk Damoun, Hamida Seba, Jean Hilger, Radu State

Graph-Level Heterogeneous Information Network Embeddings for Cardholder Transaction Analysis

Farouk Damoun[1,2*], Hamida Seba[2], Jean Hilger[1], Radu State[1]

[1*]University of Luxembourg, Luxembourg, Luxembourg.
[2]UCBL, CNRS, INSA Lyon, LIRIS, UMR5205, Villeurbanne, France.

*Corresponding author(s). E-mail(s): farouk.damoun@uni.lu;
Contributing authors: hamida.seba@univ-lyon1.fr; jean.hilger@uni.lu;
radu.state@uni.lu;

Abstract

Graph-related applications, including classification, regression, and clustering, have seen significant advancements with the development of graph neural networks (GNNs). However, a gap remains in effectively using these models for heterogeneous graphs, as current methods primarily focus on homogeneous graphs, often overlooking potentially valuable semantic information. To address this issue, our work introduces a novel approach, G-HIN2VEC (Graph-Level Heterogeneous Information Network to Vector), specifically designed to generate heterogeneous graph representations. This method uniquely leverages a single graph to learn its own embeddings without relying on a graph dataset, by sharing model parameters across the dataset. Inspired by recent developments in unsupervised learning in natural language processing, G-HIN2VEC employs a negative sampling technique to learn graph-level embedding matrices from a variety of metapaths. This approach has been applied to real-world credit card data, facilitating the analysis of cardholder transactions through three downstream applications—graph-level regression and classification tasks, including age and income prediction, and gender classification. G-HIN2VEC outperforms traditional methods, demonstrating improvements in gender classification accuracy by 2. 45% and income prediction R-squared (R2) by 7. 19%. Furthermore, for age prediction, we achieved an increase of 6. 55% in the mean absolute error (MAE) compared to DiffPool, a strong baseline.

Keywords: Unsupervised learning, Learning latent representations, Graph Neural Networks, Heterogeneous Information Network , Financial Transaction Analysis

1 Introduction

Financial institutions manage a variety of transaction types, encompassing a spectrum of financial assets, such as loans, insurance, and investments. In today's increasingly digital era, banks, as ubiquitous entities, offer several financial card products that dominate the retail banking sector. The past decade has seen an impressive increase in cashless payments through plastic and virtual cards, leading to exponential growth in the digital footprint of card transactions [1].

Analyzing customer transactions is pivotal for various banking applications, including but not limited to behavior modeling, product recommendation, and advertising strategies [1]. Recent research in financial data analysis [2, 3] suggests the utility of graph data structures to encapsulate the topological information inherent in tabular data, thereby extending and enriching available customer data. Currently, deep learning algorithms have been refined and enhanced, expanding their applicability to a wider range of problems [4–6].

Despite the advantages, graph data structures introduce additional complexity, with each graph component characterized by unique attributes and local topology. Tasks such as graph classification, clustering, and regression, which require vectorial representations of graphs for the application of machine learning algorithms, become more challenging. Techniques to overcome this challenge include Graph Kernel algorithms [7], acting as hand-crafted feature extractors, and embedding algorithms that provide dense latent graph encoding [8].

However, most of these embedding techniques have been designed for homogeneous networks [8], and therefore their application is limited to graphs with a single type of nodes and edges. However, real-world networks often comprise heterogeneous information networks (HINs), where there are different types of interaction between graph components. Graph Neural Networks (GNNs) offer potential solutions for such graphs. GNNs have proven their effectiveness in various graph-related tasks, but they often fail to preserve the network structure fully, especially for HINs [9, 10]. Graph analysis tasks are often confined by the characteristics of the problem at hand and the applied embedding techniques [11]. In scenarios involving node-level tasks such as node regression and classification, the embeddings are typically designed to portray low-dimensional representations of nodes. For example, the classification of scientific publications using citation networks exemplifies a node-level task. Conversely, for graph-level tasks, such as graph classification and regression, the embeddings provide a low-dimensional representation of the entire graph. Examples of these tasks include prediction of protein functions using chemical compound graphs and detection of malware based on call graphs [12, 13].

Recently, random walk-based embedding methods have displayed significant performance improvements in various homogeneous graph (HG) downstream tasks by

[1] According to the Federal Prof. of Credit Card Operations of Depository Institutions report, 2020 ref.https://www.federalreserve.gov/publications/files/ccprofit2020.pdf

preserving the network structure [8]. However, their utility is considerably limited when applied to a heterogeneous schema. Concurrently, much of the existing research on graph embedding remains heavily focused on learning node representations, leaving a gap in the comprehensive understanding and handling of complex heterogeneous networks.

In this work, our neural network architecture, called G-HIN2Vec, is specifically designed to embed entire heterogeneous graphs. It relies on joint learning architectures [14] and random walk-based methods for node embedding. We evaluate the quality of the generated heterogeneous graph embeddings across various downstream tasks using different metrics for credit card transactions. Based on a preliminary literature review, we carefully selected our baseline methods. The experimental results show that G-HIN2VEC embeddings outperform the baseline methods HIN2VEC [15] and Metapath2Vec [16]. In particular, G-HIN2VEC exhibits a 2.45% improvement in the precision of the gender classification and a 7.19% increase in the R-squared income prediction (R2) over the strongest baselines, with a loss in the Mean Absolute Error (MAE) in age prediction by 6.55% compared to DiffPool [17]. This shows the effectiveness of our approach in enhancing customer understanding in retail banking. Our contributions are as follows.

- We propose an ego-centric graph model for cardholder transactions to build an unlabeled heterogeneous graph dataset inspired by egocentric thinking.
- We introduce G-HIN2Vec, a new unsupervised representation learning method that employs a double-triplet loss function as task-agnostic approach for entire heterogeneous graph representations.
- We experimentally benchmark various GNN models against G-HIN2VEC on real-world credit card datasets for multiple downstream tasks, including graph classification, regression, and clustering tasks.

2 Related Work

This section reviews three primary research lines for our work, focusing on heterogeneous information networks, graph-level embedding, and financial card transaction data analysis.

- **Heterogeneous Information Network (HIN) Embedding:** Heterogeneous Information Network Embedding has emerged as a significant area of research, primarily due to its ability to effectively represent complex real-world data. These networks encompass interactions among various types of objects, offering a more detailed representation of real-world scenarios.
  Recent algorithms in HIN embedding primarily focus on node or edge-level embedding, as evidenced in studies by [15, 16, 18], This focus, however, introduces constraints when addressing graph-level downstream tasks. In a recent comprehensive review [19], 13 HIN embedding methods were benchmarked on four distinct datasets designed for node- and edge-level tasks. These studies have consistently demonstrated that the effectiveness of an HIN embedding algorithm depends on

the specific requirements of the downstream task, underlying the graph schema, and other experimental variables, as highlighted in [9, 10]. A critical limitation in the field is the lack of a unified framework for HIN embeddings, as pointed out by [8]. This absence has been identified as a significant barrier, particularly when addressing graph-level downstream tasks.

- **Graph-level Embedding:** Despite the abundance of research on node, edge, or subgraph embedding algorithms, studies on entire graph embedding algorithms are relatively sparse [12]. This impacts the downstream graph-related tasks for HIN networks such as graph classification, regression, and clustering. Conversely, HG embeddings have been extensively studied and are well-established in various downstream tasks [20]. Both graph types embedding algorithms seek to preserve topological and semantic similarity across a graph dataset. As discussed in [12] and due to lack of HIN graph-level embedding algorithms, researchers often resort to transforming HINs into HG datasets. This transformation is a proxy approach to leverage existing homogeneous graph embedding algorithms, such as Graph2Vec [20] and Deep Graph Kernels (DGK) [21], for generating graph-level embeddings. While this method has led to improvements in performance, it unfortunately discards potentially valuable semantic information. This limitation has pushed research towards developing embedding models tailored for HINs. UGraphEmb [22] addresses the issue as a supervised training model that relies on existing heuristics for graph similarity [23, 24], which is computationally intensive and not scalable. Instead of proxy or supervised approaches, HIN representation can be generated via aggregate functions, such as the average layer, to transform node/edge embeddings into graph embeddings, which often leads to suboptimal results [25], This is due to the absence of unsupervised HIN graph-level embedding algorithms.
- **Financial Card Transactions:** While the field of card transaction research focuses primarily on fraud detection, current graph-based methodologies predominantly utilize simple graphs [2]. Due to the heterogeneity of the data, this approach does not fully capture the intrinsic complexity of financial networks. This limitation highlights the growing need for HIN-based GNNs for financial transactions. For instance, HINs have been used for the detection of malicious Alipay accounts as a node-level task [13], risky transactions in a B2B network [26], and for bankruptcy prediction in credit risk assessment [27]. However, beyond fraud detection, card transaction data is being increasingly leveraged to gain insights into transaction entities. For example, techniques such as text compression have been used to understand the lifestyles of cardholders [1]. In addition, graph-based approaches have facilitated the analysis of complex relationships between cardholders and merchants as a node-level task [3], and the refinement of marketing strategies as a node-level task [28]. These diverse applications underscore the versatility of HINs in unraveling complex interactions within heterogeneous financial networks, as comprehensively reviewed in [29].

3 Problem Definition

Throughout this section, we introduce the notions of HIN and random metapath walks and discuss the problem addressed in this work. Table 1 shows the mathematical symbols and their definitions used in this article.

Symbol	Definition
$\mathbb{G}$	Dataset of all ego-centric cardholder graphs.
G_k	The k^{th} ego-centric cardholder graph in $\mathbb{G}$.
$\mathcal{N}_G$, $\mathcal{R}_G$	Set of nodes and edges in G.
N_G, E_G	Set of node and edge types in G.
τ_N, τ_E	Type mapping functions for nodes and edges.
A_k, D_k, W_k	Adjacency, diagonal, and transition matrices of G_k.
$\mathcal{P}$, p, π_{mp}	Predefined set of metapaths, a metapath, and its instance.
γ	Teleportation parameter used during random walks.
M	Quantity of negative samples generated during training.
ϕ	Learned graph-level embedding vectors for $\mathbb{G}$.
$\mathbf{h}$	Hidden state representation derived from the model.
α_1, α_2	Margins used in the double-triplet loss function.
θ	Set of all parameters in the G-HIN2VEC model.
W_v, W_e, W_G	weight matrices learned for nodes, edges, and graphs.

Table 1: Mathematical Symbols and Definitions

Definition 1. ***Heterogeneous Information Network,*** *Heterogeneous Information Network, denoted HIN, is defined as a network* $G = (\mathcal{N}_G, \mathcal{R}_G, \tau_N, \tau_E, W)$ *where* G *is a weighted directed graph, where* $\mathcal{N}_G$ *and* $\mathcal{R}_G$ *denote the set of nodes and edges,* $W \in \mathbb{R}^{|\mathcal{N}_G| x |\mathcal{N}_G|}$ *is the weight matrix, and* τ_N *and* τ_E *are type mapping functions for nodes and edges, respectively. In HINs, node types are represented by* $\tau_N : \mathcal{N}_G \rightarrow N_G$*, where* N_G *is a set of node types, and the relations between nodes are indicated by* $\tau_E : \mathcal{R}_G \rightarrow E_G$*, where* E_G *is a set of edge types.*

For HINs, metapaths are used to generate random sequences guided by a predefined schema, defined as follows:

Definition 2. ***Random Metapath Walk.*** *Given the schema of* G *as defined in Def. 1, a metapath* $p \in \mathcal{P}$ *is defined as the sequence of triplets* (n_i, e_i, n_{i+1}) *where* $n_i \in N_G$ *are the source and target node types, and* $e \in E_G$ *is the edge type,* p *is denoted as follows:*

$$p : \{(n_0, e_0, n_1) \rightarrow (n_1, e_1, n_2) ... \rightarrow (n_{l-1}, e_{l-1}, n_l)\} \tag{1}$$

An instance $\pi_{mp} \sim p(G)$ *is a randomized process that begins at* $v_i; \tau_N(v_i) = n_i$*, and at each iteration moves with respect to a specific sampling strategy* $Pr(.)$*, to the next* v_j*. After* l *iterations, the random metapath walk instance is denoted as the sequence* $\pi_{mp} = \{< v_i, e_i, v_{i+1} > \sim Pr(v_i | v_{i-1}; p \in \mathcal{P})\}_{i=0}^{l}$ *following the naive sampling strategy, defined as a uniform random walk constrained by neighboring node types according to the metapath schema, similar to the approaches in [30, 31]. This ensures that each step of the walk follows the heterogeneous relationships defined by the network schema.*

$$Pr(v_i | v_{i-1}; p \in \mathcal{P}) = \begin{cases} \frac{1}{|\mathcal{N}_{n_i}(v_{i-1})|} & e_i \in E \text{ and } \tau_E(e_i) = r_i \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

where $\mathcal{N}_{n_i}(v)$ denotes the neighbors of v of type n_i, and v_i denotes the node i^{th} in the metapath sequence and $e_i = (v_{i-1}, v_i)$. It is important to mention that n_i and r_i denote the node and relation type of the i^{th} element in the metapath schema in p, respectively. A metapath instance π_{mp} is completed after l iterations, ensuring that the start and end node types match ($\tau_N(n_0) = \tau_N(n_l)$), as required by the schema. This symmetry signifies a structured path through the network, following the sampling strategy of Equation 2. If the path length l allows, the sequence is considered valid and reflects the relationships defined by the metapath p; otherwise, it's discarded as an incomplete. The recursive aspect of the walk, $Pr(v_{l+1}|v_l; p \in \mathcal{P}) = Pr(v_0|v_l; p \in \mathcal{P})$, allows the process to be continuous, maintaining the integrity of the network structure. This methodology is common in HG [30, 31] and adapted for heterogeneous graphs in [16].

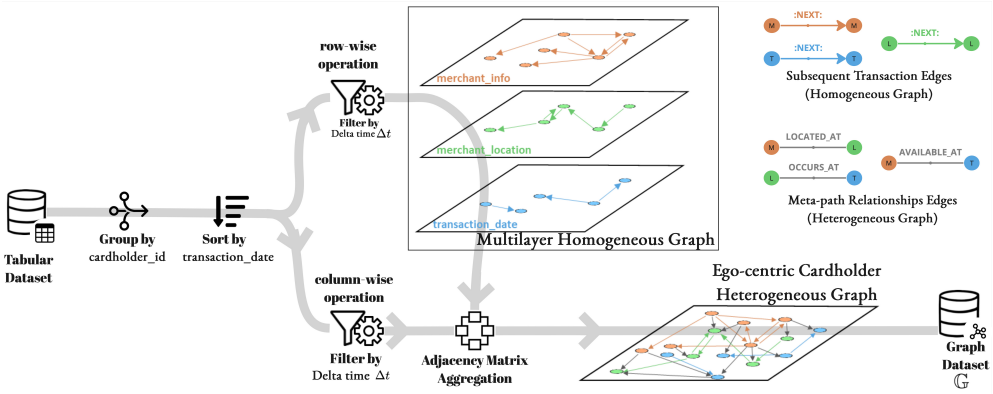


Fig. 1: Illustration of transforming credit card transaction data into an ego-centric cardholder heterogeneous graph dataset, illustrating the process of layering multiple graphs and metapaths to encapsulate the diverse financial interaction patterns within the network.

Problem Analysis and Definition for Cardholder Transactions. Instead of using a tripartite large transaction network to model the problem as a node-level representation [3], we propose to build an ego-centric cardholder graph dataset $\mathbb{G}$ to analyze and uncover collective hidden behavior patterns as a graph-level representation problem.

Definition 3. ***Ego-centric cardholder graph.*** *Given a set of graphs $\mathbb{G} = \{G_i\}_{i=1}^{n}$, G_k is the k^{th} heterogeneous ego-centric cardholder graph, including merchant names and code categories (MCC), locations, and discretized time units [32], as nodes as nodes $\mathcal{N}_{G_k} \subset \mathcal{N}_G$,*

$$\mathcal{N}_G = \bigcup_{c}^{cols} \mathcal{N}_{G_c} \tag{3}$$

The term 'cols' refers specifically to the following attributes within the transaction data:`merchant_info`, `merchant_location`, and `transaction_date`. A detailed description of these fields is provided in Section 5.1. The cardholder's transaction is the edge $\mathcal{R}_{G_k} = \{(v_i, v_j) \in \mathcal{N}_{G_k} \times \mathcal{N}_{G_k}, w^k_{i,j} \in W_k : (v_i, v_j, w^k_{i,j})\}$ and the cardholder's transition matrix $W_k \in [0,1]^{V_k \times V_k}$ is defined from the adjacency matrix G_k denoted A_k, where $[A_k]_{ij}$ represents the total consecutive pairs of transactions within a specific time window Δt, set to 24 hours, if the node types are the same in the cardholder's historical transaction records,,

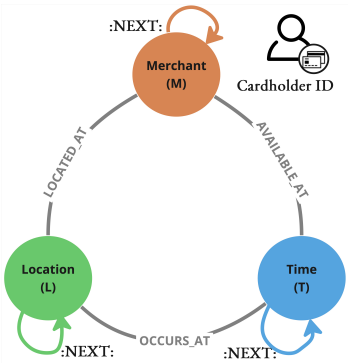


Fig. 2: Ego-centric cardholder heterogeneous graph schema

$$[A_k]_{ij} = \underbrace{\sum_{\tau_N(v_i)=\tau_N(v_j)} [1]_{\Delta t}}_{\text{row-wise operation}} + \underbrace{\sum_{\tau_N(v_i)\neq\tau_N(v_j)} 1}_{\text{column-wise operation}} \tag{4}$$

The transition matrix W_k is refined with a 2-hop step with a column stochastic matrix $W_k = (A_k D_k^{-1})^2$, where D_k is the diagonal matrix with $D_{ij} = \sum_{j=1}^{|\mathcal{N}_{G_k}|} A_{k,(i,j)}$. We amplify the signal in historical data by refining the transition matrix as a weight matrix to enrich the topological and semantic features in G_k. To avoid any confusion, we refer to self-loops in the graph schema as 'next transaction' edges, denoted by :NEXT:, as shown in Figure 2.

Problem Definition. After defining the ego-centric cardholder graph dataset $\mathbb{G}$, we aim to learn the vector representations $\phi \in \mathbb{R}^{|\mathbb{G}| \times d}$ for every graph $G_i \in \mathbb{G}$, where d is the embedding vector size, $d > 0$.

Problem 1. (Learn from graph substructures) Given a graph $G_k \in \mathbb{G}$, and a set of metapath instances $\pi_{\text{mp}} \sim p$, learn an embedding function $f_G : G_k \rightarrow \mathbf{h}_{G_k} \in \mathbb{R}^d$ that preserves the substructure properties of G_k in the latent graph embeddings space ϕ.

Problem 2. (Avoid aggregation functions) The aggregation function operates on graph substructures to represent G_k using average, sum, or max pooling layers; this technique ignores the graph topology. Given a graph $G_k \in \mathbb{G}$, learn ϕ that preserves the proximity between similar graphs in $\mathbb{G}$ and avoid explicit proxy aggregation functions that operate on latent substructure graph embedding.

4 The proposed approach

Our framework is designed based on an intuitive analogy of graph-level embeddings to define a custom loss function. In this section, we first introduce our motivation and present G-HIN2VEC as an unsupervised learning framework for heterogeneous graphs.

4.1 Motivation

The intuition behind our framework is based on a simple analogy with graph kernels for graph-level embeddings. Given two graphs G_i and G_j, for each substructure of an atomic graph a distribution-based measure g is quantified as the similarity between two graphs,

$$D(G_i, G_j) = \sum_{v=1}^{U} g(\mathbf{h}_{i,v}, \mathbf{h}_{j,v}) \tag{5}$$

where $U = \mathcal{N}(G_i) \cap \mathcal{N}(G_j)$ is the set of common substructures in both graphs. In this case nodes where $\mathcal{N}(G_i)$ represent nodes of G_i and $\mathbf{h}_{i,v}$ could indicates the color of the node v in G_i for WL-OA [33], or a random graphlet starting from v for graphlet kernel [23] or a subgraph where the ego node is v for SP-kernel [24], and D is considered as a graph representation kernel. In GNNs, g is applied to atomic graph substructures and D is considered as a graph representation,

$$D(G_i, \varnothing) = \sum_{v=1}^{U} g(\mathbf{h}(i, v), \varnothing) = \sum_{v=1}^{N_{G_i}} w_v \mathbf{h}(i, v) \tag{6}$$

Here, g could be a static pooling operation such as average operation (where $w_v = \frac{1}{|N_{G_i}|}$) or a learned aggregation layer [17]. Regardless of the local properties encoder function Q could be added and applied to $\mathbb{G}$ to represent G_i by an n-dimensional vector representing the distances from all graphs.

$$Q(G_i) = \bigcup_{G_j}^{\mathbb{G}} \{D(G_i, G_j)\} \tag{7}$$

In Equation 5 and 6, g is seen as a distance function that quantifies the dispersion within and between graphs for different local D and global Q representations. Our goal is to learn the graph representation by quantifying this dispersion without the need for explicit aggregation of the graph substructures. Simultaneously, our goal is to avoid using the true distance matrix (Equation 7) for global properties.

4.2 G-HIN2VEC Embedding Learning

The key novelty of G-HIN2VEC is to determine if a metapath instance π_{mp} from a given semantic context $p \in \mathcal{P}$ is a subset of the heterogeneous graph G_i, "Does $\pi_{\text{mp}} \subseteq G_i$?", thereby learning graph proximity through intra-graph substructures.

Given $\mathbb{G}$ and $p^+ \in \mathcal{P}$, the objective is to maximize the probability,

$$\arg\max_{\theta} \prod_{G_i \in G} \prod_{\pi_{\text{mp}} \in p^+(G_i)} Pr(\pi^{(.)}_{\text{mp}_i} | G_i; \theta) \tag{8}$$

where $\pi_{\text{mp}}^{(.)} = \{< v_{i,k}, e_{i,k}, v_{i,k+1} >\}_{k=0}^{l}$ is a sequence of triplets with respect to p^+ , which can be defined as a semantic substructures of G_i, and $Pr(\pi_{\text{mp}}^{(.)}|G_i;\theta)$ represents the conditional probability of having $\pi_{\text{mp}}^{(.)}$ given G_i.

To learn a graph representation $\phi \in \mathbb{R}^{|\mathbb{G}|\times d}$, we made a simple extension to the heterogeneous skip-gram model presented in [16], to incorporate the metapath schema to learn from graph substructure by maximizing logarithmic probability,

$$\arg\max_{\theta_1,\theta_2} \sum_{G_i \in G} \sum_{\pi_{\text{mp}} \in p^+(G_i)} \sum_{t=1}^{l} log(Pr(\pi_{\text{mp}}^{(t+)}|G_i;\theta_1)\, Pr(\pi_{\text{mp}}^{(\neq t+)}|G_i;\theta_2)) \tag{9}$$

Here, it is assumed that the positive metapath context instance $\pi_{\text{mp}}^{(\neq t+)} = \{< v_{i,k}, e_{i,k}, v_{i,k+1} >\}_{k=0,k\neq t}^{l}$ and the target instance $\pi_{\text{mp}}^{(t+)} = \{< v_{i,t}, e_{i,t}, v_{i,t+1} >\}$ are independent for a given G_i, where $\pi_{\text{mp}}^{(t+)}$ represents the t^{th} triplet in the metapath sequence and $\pi_{\text{mp}}^{(\neq t+)}$ represents the entire sequence excluding the t^{th} triplet.

In general, a heterogeneous graph requires more than one metapath to capture rich semantics; for this, G-HIN2VEC expands Equation 9 to cover the entire set of predefined metapaths $\mathcal{P}$,

$$\begin{aligned}\arg\max_{\theta_1,\theta_2} \sum_{G_i \in \mathbb{G}} \sum_{p^+ \in \mathcal{P}} \sum_{\pi_{\text{mp}} \in p^+(G_i)} \sum_{t=1}^{l} & logPr(\pi_{\text{mp}}^{(t+)}|G_i;\theta_1) \\ & + logPr(\pi_{\text{mp}}^{(\neq t+)}|G_i;\theta_2)\end{aligned} \tag{10}$$

where $Pr(\pi_{\text{mp}}^{(.)}|G_i)$ is defined as

$$\frac{exp(\mathbf{h}^{+}_{\pi_{\text{mp}}^{(.)}} \cdot \mathbf{h}_{G_i})}{\sum_{G_j \in \mathbb{G}} \sum_{\pi_{\text{mp}_j} \in p^+(G_i)} exp(\mathbf{h}^{+}_{\pi_{\text{mp}_j}^{(.)}} \cdot \mathbf{h}_{G_j})} \tag{11}$$

In order to train our G-HIN2VEC model more efficiently and avoid the exhaustive computation of considering every substructure within the entire graph dataset for each metapath in $\mathcal{P}$ and target element in $\pi_{\text{mp}}^{(.)}$, we adopt a negative sampling technique. Specifically, instead of using negative samples from different graphs in the dataset, we generate negative samples directly within the graph of interest using Algorithm 1. This allows us to create more contextually relevant and computationally efficient negative instances. For every positive instance, we generate 5 negative instances, setting the M negative sample instances to 5 for each positive instance, thus maintaining a 1:5 ratio. This ratio is a tunable hyperparameter, and while our current infrastructure supports a 1:5 ratio, this can be adjusted according to the scale and capacity of the training setup. The sampling distribution for each metapath in $\mathcal{P}$ is carefully specified, drawing inspiration from [14]. In the same direction, Equation 10 can be expressed as

a distance-based objective function following Equation 5 and 6 to customize the loss function introduced in [15][20]. Therefore, we have the following objective:

$$\mathscr{L} = \sum_{\substack{\pi^+_{\text{mp}} \sim p^+(G_i) \\ \pi^-_{\text{mp}} \sim p^-(G_i) \\ (p^+, p^-) \sim \mathcal{P} \\ p^+ \neq p^-}} \underbrace{\Big[g_{nn}(\mathbf{h}^+_{\pi^{t+}_{\text{mp}}}, \mathbf{h}^+_{G_i})^2 - g_{nn}(\mathbf{h}^-_{\pi^{t-}_{\text{mp}}}, \mathbf{h}^-_{G_i})^2 + \alpha_1 \Big]_+}_{\substack{Node \\ Triplet}\text{ : Triplet node loss}} + \underbrace{\Big[f_{nn}(\mathbf{h}^+_{\pi^{\neq t+}_{\text{mp}}}, \mathbf{h}^+_{G_i})^2 - f_{nn}(\mathbf{h}^-_{\pi^{\neq t-}_{\text{mp}}}, \mathbf{h}^-_{G_i})^2 + \alpha_2 \Big]_+}_{\substack{Graph \\ Triplet}\text{ : Triplet graph loss}} \tag{12}$$

where $[\cdot]_+ = max(\cdot, 0)$, and $p^- \underset{unif.}{\sim} \mathcal{P}$ is the negative metapath and π^-_{mp} is the negative metapath instance defined as a random sequence of triplets with respect to p^-. Note that π^-_{mp} does not necessarily exist in $\mathbb{G}$ following Algorithm 1.

Hidden substructure representations are the output of a shared weight network for each triplet element, nodes and edge, $\mathbf{h}_{v_k} = \sigma(W_v \vec{x}_{v_k})$ and $\mathbf{h}_{e_k} = \sigma(W_e \vec{x}_{e_k})$, respectively, with $\vec{x}_v \in \mathbb{R}^{|\mathcal{N}_G|}$ and $\vec{x}_e \in \mathbb{R}^{|E|}$ are the one-hot indicator vectors, and $W_v \in \mathbb{R}^{|\mathcal{N}_G| \times d}$ and $W_r \in \mathbb{R}^{|E| \times d}$ are weight matrices. The same applies for the hidden representation of the graph $\mathbf{h}_{G_i} = \sigma(W_G \vec{x}_{G_i})$, with $\vec{x}_{G_i} \in \mathbb{R}^{\mathbb{G}}$ and $W_G \in \mathbb{R}^{\mathbb{G} \times d}$ are the embedding and weight matrices, $\sigma(\cdot)$ is the sigmoid function.Then a metapath-type-aware concatenation mechanism is used to stabilize the learning process with respect to the heterogeneity of metapaths as a triplet concatenation operation $\mathbf{h}_{\pi^{(\cdot)}_{\text{mp}}} = \text{Concat}(\pi^{(\cdot)}_{\text{mp}})$, defined as

$$\mathbf{h}_{\pi^{(\cdot)}_{\text{mp}}} = W_p \Big(\Big\|_{k=1}^{l} \big(\frac{1-\beta}{2} \mathbf{h}_{v_k} \big) \odot (\beta \mathbf{h}_{e_k}) \odot \big(\frac{1-\beta}{2} \mathbf{h}_{v_{k+1}} \big) \Big) \tag{13}$$

where $\mathbf{h}_{\pi^{(\cdot)}_{\text{mp}}} \in \mathbb{R}^c$ is the hidden representation of $\pi^{(\cdot)}_{\text{mp}}$ and $W_p \in \mathbb{R}^{d \times c}$ is a linear transformation of a nonlinear function to project the hidden representations of the substructure to a specific output dimension, and $\beta \in [0, 1]$ is a signal amplifier for triplet relations, set to 0.4. Same for graph hidden representation where $W_p \mathbf{h}_{G_i} \in \mathbb{R}^c$. Equation 13 is a metapath-aware operation used for similarity measurement, without an activation function.

In summary, given the projected feature vector for G_i and $\pi^{(\cdot)}_{\text{mp}}$ positive and negative contexts, we learn two distinct functions to measure the similarity between the hidden representations of the graph and nodes, $< \mathbf{h}^+_{\pi^{t+}_{\text{mp}}}, \mathbf{h}^+_{G_i}, >$ and $< \mathbf{h}^-_{\pi^{t-}_{\text{mp}}}, \mathbf{h}^-_{G_i}, >$, respectively, with $g_{nn}(\cdot)$, and between the hidden representations of the graph and the metapath instance, $< \mathbf{h}^+_{\pi^{\neq t+}_{\text{mp}}}, \mathbf{h}^+_{G_i}, >$ and $< \mathbf{h}^-_{\pi^{\neq t-}_{\text{mp}}}, \mathbf{h}^-_{G_i}, >$, respectively, with $f_{nn}(\cdot)$. Following [34], both $g_{nn}(\cdot)$ and $f_{nn}(\cdot)$ are fully connected layers with a one-dimensional output, instead of using static functions such as Euclidean distance or cosine similarity.

As $g_{nn}(\cdot)$ and $f_{nn}(\cdot)$ represents a learned metric where the largest the value more the hidden representations are dissimilar, therefore, the dissimilarity and probability defined in Equation 10 must be correlated positively with both functions. Similarly to [34], we define both functions with a softmax activation layer to normalize the output in $[0, 1]^2$ and keep only one dimension as a dissimilarity value formulated as follows:

$$\begin{cases} g_{nn}(\mathbf{h}^{+}_{\pi^{(\cdot)}_{\text{mp}}}, \mathbf{h}^{+}_{G_i}) = \text{softmax}(W_g^T \cdot (\mathbf{h}^{+}_{\pi^{(\cdot)}_{\text{mp}}}, \mathbf{h}^{+}_{G_i}) + B_g)_0 \\ \\ f_{nn}(\mathbf{h}^{+}_{\pi^{(\cdot)}_{\text{mp}}}, \mathbf{h}^{+}_{G_i}) = \text{softmax}(W_f^T \cdot (\mathbf{h}^{+}_{\pi^{(\cdot)}_{\text{mp}}}, \mathbf{h}^{+}_{G_i}) + B_f)_0 \end{cases} \tag{14}$$

where $W_g \in \mathbb{R}^{2c\times 2}$, $W_f \in \mathbb{R}^{2c\times 2}$, $B_g \in \mathbb{R}^{2c\times 1}$, and $B_f \in \mathbb{R}^{2c\times 1}$, are the weight matrices for the learned metric function g_{nn} and f_{nn} and their bias terms, respectively.

Algorithm 1: MetapathSeqNoising Function.

Input : π^{+}_{mp}, p^{-}, l, λ
Output : π^{-}_{mp} negative metapath instance.

$l' = \lceil \lambda l \rceil$; // Number of noise edges ;
$(v_t^{-}, v_{t+1}^{-}) \leftarrow \text{Shuffle}(\{v \in V | \tau_N(v) \in p^{-}\}, l')$;
$e_t^{-} \leftarrow \text{Shuffle}(\{e \in E | \tau_E(v_t^{-}, v_{t+1}^{-}) \in p^{-}\}, l')$;
// Noise injection in π^{+}_{mp} as a concatenation process;
$l_1 = \lceil \frac{l-l'}{2} \rceil$ and $l_2 = l - l_1$;;
$\pi^{-}_{\text{mp}} \leftarrow \left\{\pi^{+}_{i,k}\right\}_{k=1}^{l_1} \oplus \left\{< v_t^{-}, e_t^{-}, v_{t+1}^{-} >\right\}_{t=1}^{l'} \oplus \left\{\pi^{+}_{i,k}\right\}_{k=l_2+1}^{l}$;

Return : π^{-}_{mp}

In our loss definition in Equation 12, we employ two margin threshold terms, α_1 and $\alpha_2 \in [0, 1]$, to modulate the similarity and dissimilarity within and between graph representations. Specifically, α_1, set to 1, regulates intra-graph similarity, ensuring that positive samples of nodes and substructures within the same graph are embedded closely together in the embedding space. On the other hand, α_2, set to 0.6, governs inter-graph dissimilarity, ensuring that negative samples of nodes and substructures are sufficiently separated in the embedding space.In the unsupervised learning context, where graph labels are unavailable, we refine the weights of G-HIN2VEC by minimizing a double-triplet loss function. This process is part of our training data preparation, which aims to generate embeddings for the graph dataset, nodes, and relations. These embeddings are denoted as $\phi \in \mathbb{R}^{|\mathbb{G}|\times d}$ for the graph-level, $X_v \in \mathbb{R}^{|\mathcal{N}_G|\times d}$ for nodes, and $X_r \in \mathbb{R}^{|\mathcal{R}_G|\times d}$ for relations. Our focus in this work is on the representation at the graph-level ϕ.

Algorithm 2: G-HIN2VEC training algorithm.

Input : The heterogeneous graph dataset $\mathbb{G}$,
metapaths $\mathcal{P} = \{p_1, ...p_k\}; k > 0$,
global learned weight W_j, batch size B
walk length l, negative samples M, noise level λ

Output : The global weight W_{i+1} matrix.

$G = \text{Shuffle}(\mathbb{G}, B)$; // Shuffle and Random B samples.

for $i = 1, ..., B$ **do**

$p^+ = \text{Shuffle}(p, 1)$;

$p^- = \text{Shuffle}(\{p \in \mathcal{P} | p \neq p^+\}, M)$;

$\pi^+_{\text{mp}} := \text{MetapathSeq}(G_i, p^+, l)$;

for $m = 1, ..., M$ **do**

// M random negative samples, set to 5.

$\pi^-_{\text{mp}} = \text{MetapathSeqNoising}(\pi^+_{\text{mp}}, p^-, l, \lambda)$;

$\mathbf{h}_{G_i} = \sigma(W_G \vec{x}_{G_i})$;

for $k = 1, ..., l$ **do**

// For each Triplet in π^+_{mp} and π^-_{mp};

$< v^+_{i,k}, e^+_{i,k}, v^+_{i,k+1} > \leftarrow \pi^+_{i,k}$;

$< v^-_{i,k}, e^-_{i,k}, v^-_{i,k+1} > \leftarrow \pi^-_{i,k}$;

// Relation hidden representation transformations;

$\mathbf{h}_{e^+_{i,k}} = \sigma(W_r \vec{e}^+_{i,k})$;

$\mathbf{h}_{e^-_{i,k}} = \sigma(W_r \vec{e}^-_{i,k})$;

// Node hidden representation transformations;

$\mathbf{h}_{v^+_{i,k}}, \mathbf{h}_{v^+_{i,k+1}} = \sigma(W_v \vec{v}^+_{i,k}), \sigma(W_v \vec{v}^+_{i,k+1})$;

$\mathbf{h}_{v^-_{i,k}}, \mathbf{h}_{v^-_{i,k+1}} = \sigma(W_v \vec{v}^-_{i,k}), \sigma(W_v \vec{v}^-_{i,k+1})$;

end

$t = \text{Shuffle}(\text{interval}[1, l], 1)$// Sample target context ;

// Representations alignment and concatenation;

$\mathbf{h}^+_{G_i} = W_{p^+} \mathbf{h}_{G_i}$; $\mathbf{h}^-_{G_i} = W_{p^-} \mathbf{h}_{G_i}$;

$\mathbf{h}^+_{\pi^{t+}_{\text{mp}}} = \text{Concat}(\mathbf{h}_{\pi^{t+}_{\text{mp}}})$; $\mathbf{h}^-_{\pi^{t-}_{\text{mp}}} = \text{Concat}(\mathbf{h}_{\pi^{t-}_{\text{mp}}})$;

$\mathbf{h}^+_{\pi^{\neq t+}_{\text{mp}}} = \text{Concat}(\mathbf{h}_{\pi^{\neq t+}_{\text{mp}}})$; $\mathbf{h}^-_{\pi^{\neq t-}_{\text{mp}}} = \text{Concat}(\mathbf{h}_{\pi^{\neq t-}_{\text{mp}}})$;

// Output layer as learned dissimilarity metric;

$d^+_{node}, d^-_{node} = g_{nn}(\mathbf{h}^+_{\pi^{t+}_{\text{mp}}}, \mathbf{h}^+_{G_i})^2, g_{nn}(\mathbf{h}^-_{\pi^{t-}_{\text{mp}}}, \mathbf{h}^-_{G_i})^2$;

$d^+_{mp}, d^-_{mp} = f_{nn}(\mathbf{h}^+_{\pi^{\neq t+}_{\text{mp}}}, \mathbf{h}^+_{G_i})^2, f_{nn}(\mathbf{h}^-_{\pi^{\neq t-}_{\text{mp}}}, \mathbf{h}^-_{G_i})^2$;

// Cost function Equation 12;

$Cost_i += [d^+_{node} - d^-_{node} + \alpha_1]_+ + [d^+_{mp} - d^-_{mp} + \alpha_2]_+$;

end

end

$W_{i+1} := \text{miniBatchSGD}(W_i, Cost_i)$ // Weight updates;

4.3 Training Data Preparation

As introduced previously, G-HIN2VEC uses the negative sampling technique to generate training data. Therefore, data preparation is a crucial phase in our approach. As detailed in Algorithm 2 and illustrated in Figure 3, we train our model in mini-batches, sampled from $\mathbb{G}$, and then generate the positive context π^{+}_{mp} from G_i with a predefined metapath $p^{+} \underset{unif.}{\sim} \mathcal{P}$ following our custom sampling strategy.

$$Pr(v^i|v^{i-1};p^+) = \begin{cases} \beta \frac{1}{|\mathcal{N}_{n_i}(v^{i-1})|} & e_i \in \mathcal{R}_G \text{ and } \tau_E(e_i) = r_i \\ (1-\beta)\frac{1}{|\mathcal{N}_{\neq n_i}(v^{i-1})|} & e_i \in \mathcal{R}_G \text{ and } \tau_E(e_i) \neq r_i \\ 0 & e_i \notin \mathcal{R}_G; \text{ where } e_i = (v^i, v^{i-1}) \end{cases} \tag{15}$$

where β is the teleportation term to escape nodes with high centrality and$\mathcal{N}_{\neq n_i}(v)$ denotes the neighbor v^i of different types of nodes $n_i \in p^+$, denoted by $MetapathSeq$ in Algorithm 1.

As in Figure 3, our approach introduces noise to the original positive metapath instances π^{+}_{mp}, creating negative contexts π^{-}_{mp} within the same graph rather than from different graphs, This is achieved by perturbing π^{+}_{mp} with alternative metapaths p^- uniformly sampled from $\mathcal{P}$. The noise level $\lambda \in [0, 1]$ is set to 0.3; a process we denote as $MetapathSeqNoising$. After generating both positive and negative instances, we apply a series of linear transformations to derive the final triplet hidden representations to estimate the dissimilarity metrics.

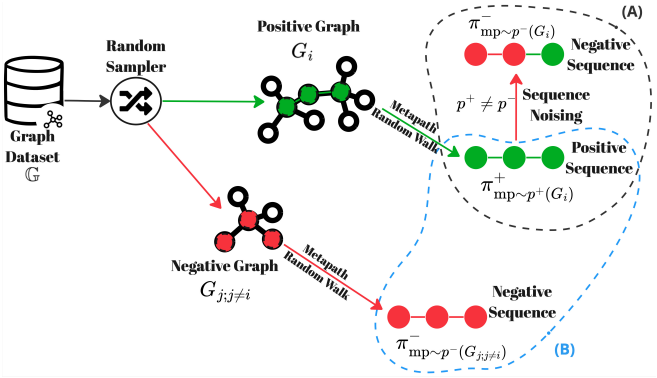


Fig. 3: G-HIN2VEC training data generation process, highlighting the **novel approach (A)** that deviates from traditional methods (B). It showcases the creation of positive ● and negative ● samples in an unsupervised learning context.

Depending on the quantified metrics in our unsupervised framework, we optimize the global model weights by minimizing the double-triplet loss via back-propagation and gradient descent to learn meaningful graph-level embeddings for the heterogeneous graph dataset.

5 Experiments

In this section, we evaluate the proposed model in a real-world credit card transaction dataset from a European bank. Our empirical analysis focuses on qualitative and quantitative analyses. It is important to mention that the dataset used in the current research work is fully aligned with the General Data Protection Regulation (GDPR) in the European Union to protect personal data as defined in Article 4.

5.1 Datasets and Tasks

We conducted our experiments on a dataset comprising records from 100K cardholders and transactions from 2019 to 2021. The dataset's schema includes the transaction table: `cardholder_id` (as unique `INT`), `merchant_info` (as `VARCHAR`), representing merchant code category and merchant name such as '5661-Shoe Stores', `merchant_location` (as `VARCHAR`), detailing the location like 'Paris-France', and `transaction_date` (as `DATE`), indicating when the transaction occurred, for instance, '2019-06-21'. This raw data was transformed into an ego-centric cardholder graph dataset, as elaborated in Section 3.3. Furthermore, we used the cardholder table: `cardholder_id` (as unique `INT`), `Gender` (as `VARCHAR`), `Age` (as `FLOAT`) and `Income` (as `FLOAT`). All types are according to SQL data types.
The cardholder socio-demographic attributes used for supervised machine learning experiments include:

- **Gender**, $Y^i_{Gender} \in \{0,1\}, \forall i \in \{1,..,n\}$) as a binary attribute indicating the i^{th} cardholder's gender, where 0 represents one gender and 1 represents another.
- **Income**, $Y^i_{Income} \in [0,1], \forall i \in \{1,..,n\}$)as a normalized attribute representing the i^{th} cardholder's income.
- **Age**, $Y^i_{Age} \in [0,1] \forall i \in \{1,..,n\}$ as a normalized attribute representing the i^{th} cardholder's age.

For quantitative analysis, we performed a binary classification task on Y_{Gender} and a regression task on Y_{Age} and Y_{Income} to benchmark different hidden representation models on predictive tasks for the ego-centric graph of cardholders. On the other hand, we also performed a qualitative analysis based on cardholder representations as a cluster analysis task to illustrate and interpret hidden communities.

5.2 Baselines and Experimental Settings

In this section, we outline the baseline methods used for comparison with our G-HIN2VEC approach, focusing on homogeneous and heterogeneous graph embeddings, as well as other graph-level representation techniques. These baselines are detailed in Table 2 and grouped into three primary categories. Our G-HIN2VEC method is implemented using the StellarGraph framework, a Python library for machine learning on graphs. All graph embeddings for the cardholder were learned using G-HIN2VEC and the baseline models. For G-HIN2VEC specifically, the experimental settings include a dropout rate of 0.5, and the use of the SGD optimizer. We adjust the margin thresholds α_1 and α_2 in our loss function, in Equation 12, to 1 and 0.5, respectively, with a learning rate of 0.005. Additionally, weight decay is accompanied by an L2 penalty set at 0.001 over 50 epochs.

After the pretraining phase on the graph dataset of cardholders, we benchmarked the performance of the generated embeddings on a set of downstream tasks. Linear and logistic regression were implemented on the graph embedding vector for regression and classification tasks, respectively. The dataset was divided into training, validation, and

Category	Category Description	Method	Method Description
Vectorization-based	Uses node and graph features to represent graphs, focusing on node frequency, graph statistics,and structural properties such as centrality measures, graph density, and spectral properties.	N-grams ($N = 1, 2$)	Captures local graph features through individual and pairs of nodes frequency, emphasizing simpler patterns.
		Bag-of-Features (BoF)	Incorporates comprehensive graph features, including statistics on node degree, clustering coefficients, and triangles and squares subgraph counts.
Kernel-based	Employing graph kernels, such as outlined in Equation 5 to map graphs into a vector space. This transformation facilitates the computation of a distance matrix for comparing graphs, as illustrated in Equation 7.	Graphlet kernel (GK) [23]	Represents graphs based on the frequency of small subgraphs (graphlets) throughout the graph dataset, ideal for capturing local topology.
		Shortest path kernel (SPK) [24]	Encodes each graph by the shortest path lengths between nodes within the graph dataset, reflecting global connectivity.
		Weisfeiler-Lehman framework (WLG) [33]	Refines graph representation through iterative relabeling based on neighborhood structures to capture structural changes over multiple scales, reflecting both local and global structural differences throughout the graph dataset.
		Deep GK, Deep SP, Deep WL [21]	Apply deep learning techniques with traditional graph kernels to generate graph-level embeddings. All variants capture different complex structural graph features in the graph dataset.
GNN-based	Uses deep learning architectures to generate node-level and graph-level embeddings, for node-level architectures an aggregation function is used to produce graph-level embeddings from node–level from node-level and edge-level embeddings, as in Equation 6.	Metapath2vec [16]	Uses guided random walks by metapaths in heterogeneous graphs to generate node embeddings. For graph-level features, the graph node embeddings are then aggregated using mean to produce graph-level embeddings.
		HIN2VEC [15]	Generate node and relation embeddings in heterogeneous networks by considering multiple types of nodes and relationships. For graph-level features, graph nodes and relationships are aggregated by average to form graph-level embeddings.
		Graph2Vec [20]	Treat the entire graph as an individual document in a corpus, employing a document embedding approach to learn graph embeddings. This method captures global graph properties and structural features.
		DiffPool [17]	Implement a differentiable graph pooling module that hierarchically aggregates nodes embeddings into clusters, forming a new, coarser graph at each layer, until final cluster as the graph-level representations.

Table 2: Detailed overview of graph-based methods categorized by their approach and core features, including graph-level and heterogeneity, with a focus on graph representations.

test sets based on the cardholder ID to ensure that all transactions related to a single cardholder fall into the same group. The split ratio was 2:1:1 for training, validation, and test sets, respectively. The performance results reported on Table 4 are the average of 10-fold cross-validation test sets, with the statistical significance of each metric validated by calculating the p-value. Performance for graph-level representation in the classification task was evaluated using averaged accuracy, AUC, and F1 metrics, and for regression tasks using R-squared (R2) and mean absolute error (MAE) metrics.

5.3 Empirical Validation

We performed empirical validation on the graph dataset following the experimental settings described earlier. For non-graph-level representation models such as HIN2Vec, we added a simple averaging layer on top of each node-level representation to generate graph representations. In heterogeneous guided random walk-based GNNs, the definition of the metapath is required.

For this purpose, experts provided a set of metapaths to answer specific business questions (BQ), as shown in Table 3. Furthermore, metapath-free baselines ignore the predefined metapaths for both node- and graph-level representation methods,

and this will not be considered in both experimental analyses due to the original implementations.

5.4 Quantitative Analysis

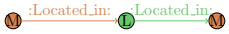
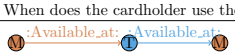
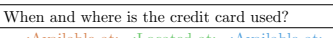
Category	Method	Gender		Income		Age	
		Acc.	AUC	R2	MAE	R2	MAE
Vectorization based	N-gram (1 & 2-grams)	0.6133	0.601	0.1067	0.1531	0.1045	12.7074
	Bag of Features (BoF)	0.7295	0.722	0.1751*	0.1409*	0.2994	10.9955
Kernel based	Graphlet kernel [23] (GK)	0.5825	0.5647	0.0449	0.1618	0.0846	12.5174
	Shortest path kernel [24] (SPK)	0.5477	0.5002	0.0698	0.1597	0.0873	12.9465
	Weisfeiler-Lehman [33] (WL)	0.7001	0.6993	0.2030	0.1410	0.4301	9.8510
Deep Kernel based [21]	Deep GK	0.5399	0.5185	0.0472	0.1469	0.0842	11.4422
	Deep SP	0.5566	0.5137	0.0798	0.1629	0.0922	13.1960*
	Deep WL	0.7032*	0.7292**	0.2255	0.1381*	0.4717	9.5978**
GNN based	Metapath2vec$^{\ddagger\parallel}$ [16]	0.6102	0.5586	0.1105**	0.1489	0.1349	12.3272
	Graph2Vec†§ [20]	0.7220	0.7331	0.2394*	0.1354	0.5384	9.4021
	HIN2VEC$^{\ddagger\parallel}$ [15]	0.7809	0.7906	0.2318	0.1399*	0.6385*	6.6019
	DiffPool†§ [17]	0.8047*	0.8194	0.3088**	0.1212**	**0.7108***	**5.9277***
	G-HIN2VEC‡§	**0.8244***	**0.8311***	**0.3310***	**0.1137****	0.6843**	6.3157*

Table 4: Performance of various graph-based methods in predicting cardholder attributes. Superscripts denote model compatibility: ‡ for heterogeneous graphs, † for homogeneous graphs, $^{\parallel}$ for node-level, and § for graph-level downstream tasks. Statistical significance is denoted as: $^{**}p < 0.01$, $^{*}p < 0.05$.

5.4.1 Regression tasks.

In regression tasks targeting age and income prediction, our evaluation reveals that vectorization and graph kernel-based methods struggle to effectively model the complex relationships inherent in heterogeneous graph data, resulting in inferior performance. In contrast, GNN-based models show enhanced outcomes due to their advanced architectures, which are better suited to capture the complex interactions within these graphs. Notably, our G-HIN2VEC model demonstrates significant improvements over the strong baseline, DiffPool, with a 7.18% increase in R-squared and a 6.19% decrease in mean absolute error for income prediction, despite its non-Gaussian distribution. For age prediction, which follows a Gaussian distribution, G-HIN2VEC matches the performance of DiffPool. These results affirm the robustness of G-HIN2VEC in generating effective graph-level embeddings for regression tasks, highlighting its potential for broader application in complex graph-based analysis.

Table 3: Set of business questions as metapaths by financial experts, where ● Merchant, ● Location, and ● Time stand for node types.

BQ 1:	What spending habit does the cardholder have?
Metapath 1	
BQ 2:	Where does the cardholder use the credit card?
Metapath 2	
BQ 3:	When does the cardholder use the credit card?
Metapath 3	
BQ 4:	When and where is the credit card used?
Metapath 4	

5.4.2 Classification task.

The gender classification performance of various methods is comprehensively detailed in Table 4. Among the benchmarked methods, vectorization methods are observed to outperform graph kernel-based methods, showcasing their better suitability for classification tasks involving categorical data. In contrast, among the baselines, DiffPool stands out as the most effective model. It highlights the benefits of utilizing sophisticated aggregation techniques rather than averaging at the top of the node-level, thus enhancing the quality of graph-level embeddings. In the specific case of our G-HIN2VEC model, significant improvements in gender classification are evident, with increases of 2.44% in accuracy, 1.85% in F1 score, and 1.43% in AUC compared to DiffPool. These statistically significant results demonstrate G-HIN2VEC's superior capability in leveraging complex graph-level embeddings for classification tasks. Additionally, the analysis of the graph structures within our dataset, particularly their size distribution and negative assortativity coefficients as shown in Figure 4, supports these outcomes. This structural characteristic suggests that G-HIN2VEC not only effectively handles disassortative graphs but also benefits from adjustments such as the incorporation of a teleportation term, β, in our sampling strategy (see Equation 15), which enhances model robustness and confirms the quantitative findings of improved performance. Furthermore, a primary limitation of GNN models, as highlighted in [35], involves their sensitivity to assortativity during node-level tasks, which can extend to graph-level tasks, potentially impacting performance; our model's design addresses and mitigates this limitation effectively.

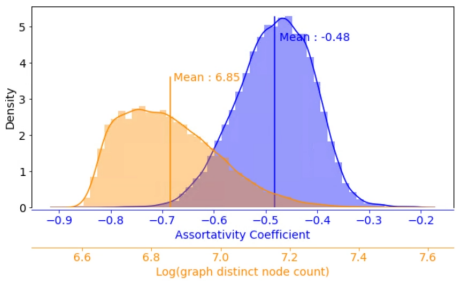


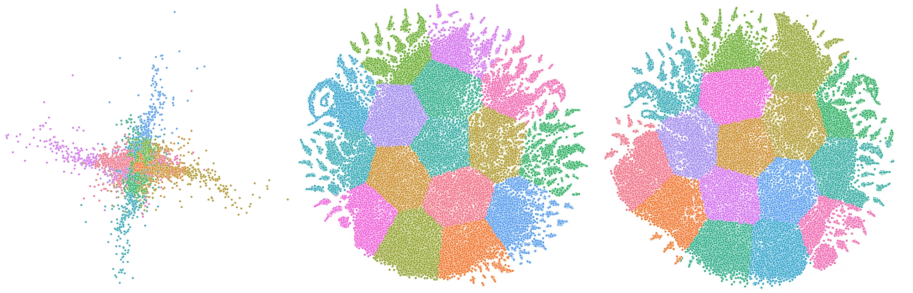
Fig. 4: The probability distribution of (a) the log graph sizes and (b) the assortativity coefficient of graphs in the dataset.

5.5 Qualitative analysis

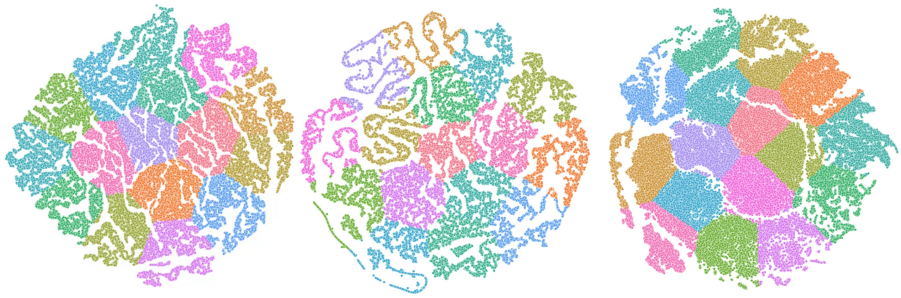
In this section, we investigated the cluster assignment of cardholder behavior through the graph-level embeddings. Figure 5 presents a t-SNE visualization of cardholder embeddings, where each color signifies a distinct cluster, corresponding to a unique behavioral pattern in credit card usage. Our assumption is that cardholders who are semantically similar in using credit cards tend to be embedded in close proximity, as shown in [20] and [16] for graph- and node-level representations, respectively.

The clustering of embeddings into 16 distinct groups, as determined by the elbow method using k-means, reveals a granular view of cardholder behaviors, with each cluster capturing a particular lifestyle preference that aligns with previous findings in credit card usage research [1]. In post-clustering analysis, we engaged domain experts to identify cluster identities, extracting and examining metapath instances that were

sampled during training. This analysis aimed to quantify the occurrence of directed sequences within each cluster, thus providing a narrative for the lifestyle categories annotated in Figure 5. Thus, two cardholders are semantically similar if their credit card usage behavior is similar with respect to a set of predefined metapaths in Table 3.



(a) Weisfeiler-Lehman [33] (b) Graph2Vec [20] (c) Metapath2Vec [16]



(d) Hin2Vec [15] (e) DiffPool [17] (f) G-HIN2VEC (Ours)

Fig. 5: Cardholder embeddings t-SNE visualization of 16 identified clusters using different embedding algorithms. Each color represents a distinct cluster; ● Grocery Shoppers, ● Commuters ● Young Workers ● High-Tech ● Elderly ● Restaurant-goers ● Drivers ● Musicophiles ● Bookworm/Bibliophiles ● Business ● FastFood-goers ● Householders ● Cash-Only ● Gamblers ● Peripatetic/Travelers ● Epicureans.

The t-SNE visualizations highlight how well different graph-level embedding methods group similar cardholder behaviors. The Weisfeiler-Lehman method results in overlapping clusters, suggesting that it may not be as effective in identifying the finer details of the graph-level features. Graph2Vec and Metapath2Vec improve upon this with more defined clusters, reflecting their stronger ability to map out both the individual elements and graph-level structures. HIN2VEC goes a step further, creating even more distinct clusters that show its capability to capture the variety within the graph dataset, and the DiffPool method stands out with its highly distinct and compact clusters, which means that it is particularly good at understanding complex patterns. Our G-HIN2VEC method generates embeddings that are more distinct than

Metapath2Vec but not as tightly clustered as HIN2VEC and DiffPool, suggesting that it provides a detailed, yet comprehensive overview of cardholder behavior. This aligns with the solid results we see from our quantitative analysis, where our method shows a strong ability to accurately group cardholder behaviors.

6 Conclusion And Future Work

In this paper, we introduce G-HIN2VEC, a novel approach to generate heterogeneous graph-level embeddings. Our method utilizes a double-triplet loss and an unsupervised learning framework that incorporates negative sampling, eliminating the need for graph-to-graph matching. This allows for more efficient learning without relying on negative samples from the entire graph dataset. We used a real-world financial dataset, modeling it as ego-centric graphs for cardholders to benchmark various graph-level representation models. Our results show competitive performance of G-HIN2VEC with state-of-the-art models, with notable improvements in gender classification accuracy of 2.45% and income prediction R-squared (R2) of 7.19%. Furthermore, for age prediction, we achieved a 6.55% increase in mean absolute error (MAE) compared to the strong baseline, DiffPool.

Looking ahead, we plan to enhance G-HIN2VEC by integrating dynamic metapath selection mechanisms to address node and edge type imbalances in Heterogeneous Information Networks (HINs). We also aim to extend the application of G-HIN2VEC to a wider range of graph-level downstream tasks. Exploring innovative negative sampling techniques, such as sequence noising and nonsampling approaches, will be another focus to further improve graph-level representation learning within an unsupervised framework. A current limitation of our methodology is its reliance on a transductive learning model, which may not perform well with unseen data. To overcome this, our future work will focus on an inductive model that can generalize to new, unseen graph instances, enhancing the applicability and utility of our approach in the field of graph representation learning.

Acknowledgments

We would like to acknowledge the invaluable knowledge exchange facilitated by the Data Analysis Lab team and our industry partners' experts. Partial funding for this work was provided by the Luxembourg National Research Fund (FNR) under grant number 15829274, supplemented by ANR-20-CE39-0008.

Data Availability

Due to privacy considerations and adherence to the General Data Protection Regulation in the European Union (GDPR) as defined in Article 4, the data supporting this study are not publicly available.

Declarations

Conflict of Interest

The authors declare that they have no conflict of interest.

Ethical Approval

The authors approve that the research presented in this article is conducted following the principles of ethical and professional conduct.

Consent to Participate

Not applicable.

Consent for Publication

Not applicable.

References

[1] Di Clemente, R., Luengo-Oroz, M., Travizano, M., Xu, S., Vaitla, B., González, M.C.: Sequences of purchases in credit card data reveal lifestyles in urban populations. Nature communications **9**(1), 3330 (2018)

[2] Ren, Y., Zhu, H., Zhang, J., Dai, P., Bo, L.: Ensemfdet: An ensemble approach to fraud detection based on bipartite graph. In: 2021 IEEE 37th International Conference on Data Engineering (ICDE), pp. 2039–2044 (2021). IEEE

[3] Khazane, A., Rider, J., Serpe, M., Gogoglou, A., Hines, K., Bruss, C.B., Serpe, R.: Deeptrax: Embedding graphs of financial transactions. In: 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), pp. 126–133 (2019). IEEE

[4] Wang, R., Zhuang, Z., Tao, H., Paszke, W., Stojanovic, V.: Q-learning based fault estimation and fault tolerant iterative learning control for mimo systems. ISA transactions **142**, 123–135 (2023)

[5] Song, X., Wu, N., Song, S., Zhang, Y., Stojanovic, V.: Bipartite synchronization for cooperative-competitive neural networks with reaction–diffusion terms via dual event-triggered mechanism. Neurocomputing **550**, 126498 (2023)

[6] Song, X., Wu, N., Song, S., Stojanovic, V.: Switching-like event-triggered state estimation for reaction–diffusion neural networks against dos attacks. Neural Processing Letters **55**(7), 8997–9018 (2023)

[7] Vishwanathan, S.V.N., Schraudolph, N.N., Kondor, R., Borgwardt, K.M.: Graph kernels. The Journal of Machine Learning Research **11**, 1201–1242 (2010)

[8] Goyal, P., Ferrara, E.: Graph embedding techniques, applications, and performance: A survey. Knowledge-Based Systems **151**, 78–94 (2018)

[9] Wang, X., Bo, D., Shi, C., Fan, S., Ye, Y., Philip, S.Y.: A survey on heterogeneous graph embedding: methods, techniques, applications and sources. IEEE Transactions on Big Data **9**(2), 415–436 (2022)

[10] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Philip, S.Y.: A comprehensive survey on graph neural networks. IEEE transactions on neural networks and learning systems **32**(1), 4–24 (2020)

[11] Cai, H., Zheng, V.W., Chang, K.C.-C.: A comprehensive survey of graph embedding: Problems, techniques, and applications. IEEE transactions on knowledge and data engineering **30**(9), 1616–1637 (2018)

[12] Ma, G., Ahmed, N.K., Willke, T.L., Yu, P.S.: Deep graph similarity learning: A survey. Data Mining and Knowledge Discovery **35**, 688–725 (2021)

[13] Liu, Z., Chen, C., Yang, X., Zhou, J., Li, X., Song, L.: Heterogeneous graph neural networks for malicious account detection. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, pp. 2077–2085 (2018)

[14] Sun, L., He, L., Huang, Z., Cao, B., Xia, C., Wei, X., Philip, S.Y.: Joint embedding of meta-path and meta-graph for heterogeneous information networks. In: 2018 IEEE International Conference on Big Knowledge (ICBK), pp. 131–138 (2018). IEEE

[15] Fu, T.-y., Lee, W.-C., Lei, Z.: Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, pp. 1797–1806 (2017)

[16] Dong, Y., Chawla, N.V., Swami, A.: metapath2vec: Scalable representation learning for heterogeneous networks. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 135–144 (2017)

[17] Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., Leskovec, J.: Hierarchical graph representation learning with differentiable pooling. Advances in neural information processing systems **31** (2018)

[18] Wang, Y., Duan, Z., Liao, B., Wu, F., Zhuang, Y.: Heterogeneous attributed network embedding with graph convolutional networks. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 10061–10062 (2019)

[19] Yang, C., Xiao, Y., Zhang, Y., Sun, Y., Han, J.: Heterogeneous network representation learning: A unified framework with survey and benchmark. IEEE Transactions on Knowledge and Data Engineering **34**(10), 4854–4873 (2020)

[20] Narayanan, A., Chandramohan, M., Venkatesan, R., Chen, L., Liu, Y., Jaiswal, S.: graph2vec: Learning distributed representations of graphs. ArXiv **abs/1707.05005** (2017)

[21] Yanardag, P., Vishwanathan, S.: Deep graph kernels. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1365–1374 (2015)

[22] Bai, Y., Ding, H., Qiao, Y., Marinovic, A., Gu, K., Chen, T., Sun, Y., Wang, W.: Unsupervised inductive graph-level representation learning via graph-graph proximity, 1988–1994 (2019)

[23] Borgwardt, K.M., Kriegel, H.-P.: Shortest-path kernels on graphs. In: Fifth IEEE International Conference on Data Mining (ICDM'05), p. 8 (2005). IEEE

[24] Shervashidze, N., Vishwanathan, S., Petri, T., Mehlhorn, K., Borgwardt, K.: Efficient graphlet kernels for large graph comparison. In: Artificial Intelligence and Statistics, pp. 488–495 (2009). PMLR

[25] Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? In: International Conference on Learning Representations (2018)

[26] Rao, S.X., Zhang, S., Han, Z., Zhang, Z., Min, W., Chen, Z., Shan, Y., Zhao, Y., Zhang, C.: xfraud: explainable fraud transaction detection. arXiv preprint arXiv:2011.12193 (2020)

[27] Zheng, Y., Lee, V.C., Wu, Z., Pan, S.: Heterogeneous graph attention network for small and medium-sized enterprises bankruptcy prediction. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 140–151 (2021). Springer

[28] Liu, Z., Wang, D., Yu, Q., Zhang, Z., Shen, Y., Ma, J., Zhong, W., Gu, J., Zhou, J., Yang, S., *et al.*: Graph representation learning for merchant incentive optimization in mobile payment marketing. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, pp. 2577–2584 (2019)

[29] Wang, J., Zhang, S., Xiao, Y., Song, R.: A review on graph neural network methods in financial applications. Journal of Data Science **20**(2), 111–134 (2022)

[30] Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 855–864 (2016)

[31] Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 701–710 (2014)

[32] Dyreson, C.E., Evans, W.S., Lin, H., Snodgrass, R.T.: Efficiently supporting temporal granularities. IEEE Transactions on Knowledge and Data Engineering **12**(4), 568–587 (2000)

[33] Shervashidze, N., Schweitzer, P., Van Leeuwen, E.J., Mehlhorn, K., Borgwardt, K.M.: Weisfeiler-lehman graph kernels. Journal of Machine Learning Research **12**(9) (2011)

[34] Wang, F., Zuo, W., Lin, L., Zhang, D., Zhang, L.: Joint learning of single-image and cross-image representations for person re-identification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1288–1296 (2016)

[35] Suresh, S., Budde, V., Neville, J., Li, P., Ma, J.: Breaking the limit of graph neural networks by improving the assortativity of graphs with local mixing patterns. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, pp. 1541–1551 (2021)