# Latent Network Summarization: Bridging Network Embedding and Summarization

Di Jin
University of Michigan
dijin@umich.edu

Ryan A. Rossi
Adobe Research
rrossi@adobe.com

Eunyee Koh
Adobe Research
eunyee@adobe.com

Sungchul Kim
Adobe Research
sukim@adobe.com

Anup Rao
Adobe Research
anuprao@adobe.com

Danai Koutra
University of Michigan
dkoutra@umich.edu

## ABSTRACT

Motivated by the computational and storage challenges that dense embeddings pose, we introduce the problem of *latent network summarization* that aims to learn a compact, latent representation of the graph structure with dimensionality that is *independent* of the input graph size (*i.e.*, #nodes and #edges), while retaining the ability to derive node representations on the fly. We propose Multi-LENS, an inductive multi-level latent network summarization approach that leverages a set of *relational operators* and *relational functions* (compositions of operators) to capture the structure of egonets and higher-order subgraphs, respectively. The structure is stored in low-rank, size-independent structural feature matrices, which along with the relational functions comprise our latent network summary. Multi-LENS is general and naturally supports both homogeneous *and* heterogeneous graphs with or without directionality, weights, attributes or labels. Extensive experiments on real graphs show $3.5 - 34.3\%$ improvement in AUC for link prediction, while requiring $80 - 2152\times$ *less* output storage space than baseline embedding methods on *large* datasets. As application areas, we show the effectiveness of Multi-LENS in detecting anomalies and events in the Enron email communication graph and Twitter co-mention graph.

**ACM Reference Format:**
Di Jin, Ryan A. Rossi, Eunyee Koh, Sungchul Kim, Anup Rao, and Danai Koutra. 2019. Latent Network Summarization: Bridging Network Embedding and Summarization. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3292500.3330992

## 1 INTRODUCTION

Recent advances in representation learning for graphs have led to a variety of proximity-based and structural embeddings that achieve superior performance in specific downstream tasks, such as link prediction, node classification, and alignment [10, 13, 28]. At the same time though, the learned, $K$-dimensional node embeddings are dense (with real values), and pose computational and storage challenges especially for massive graphs. By following the conventional setting of $K = 128$ for the dimensionality, a graph of 1 billion nodes requires roughly 1TB for its embeddings. Moreover, this dense representation often requires significantly more space to store than the original, sparse adjacency matrix of a graph. For example, for the datasets that we consider in our empirical analysis, the learned embeddings from existing representation learning techniques require $3 - 48\times$ more space than the original edge files.

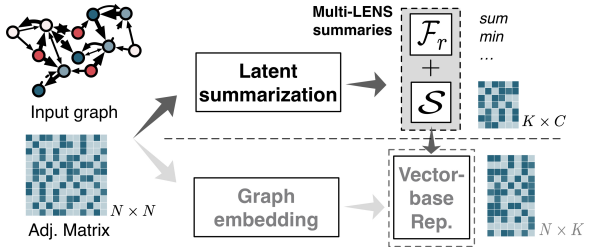



**Figure 1: Our proposed approach to Latent Network Summarization called Multi-LENS produces a summary consisting of relational functions $\mathcal{F}_r$ and node-independent matrices $\mathcal{S}$ of size $K \times C$. Thus, while embedding methods output $N$ node embeddings of dimensionality $K$, latent summarization methods produce an output that is independent of $N$ and thus is graph-size independent. Despite not storing the embeddings, Multi-LENS can derive them on the fly.**

To address these shortcomings, we introduce the problem of *latent network summarization*. Informally, the goal is to find a low-dimensional representation in a latent space such that it is *independent* of the graph size, *i.e.*, the number of nodes and edges. Among other tasks, the representation should support *on-the-fly* computation of *specific* node embeddings. Latent network summarization and network embedding are *complementary* learning tasks with fundamentally different goals and outputs, as shown in Fig. 1. In particular, the goal of network embedding is to derive $N$ node embedding vectors of $K$ dimensions each that capture node proximity or equivalency. Thus, the output is a $N \times K$ matrix that is *dependent* on the size of the graph (number of nodes) [10, 24]. This is in contrast to the goal of latent network summarization, which is to learn a *size-independent representation* of the graph. *Latent* network summarization also differs from traditional summarization approaches that typically derive supergraphs (*e.g.*, mapping nodes to supernodes) [19], which target different applications and are unable to derive node embeddings.


*KDD '19, August 4–8, 2019, Anchorage, AK, USA*

ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330992

To efficiently solve the latent network summarization problem, we propose MULTI-LENS (Multi-level Latent Network Summarization), an inductive framework that is based on graph function compositions. In a nutshell, the method begins with a set of arbitrary graph features (*e.g.*, degree) and iteratively uses generally-defined relational operators over neighborhoods to derive deeper function compositions that capture graph features at multiple *levels* (or distances). Low-rank approximation is then used to derive the best-fit subspace vectors of network features across levels. Thus, the latent summary given by MULTI-LENS comprises graph functions and latent vectors, both of which are independent of the graph size. Our main contributions are summarized as follows:

- **Novel Problem Formulation.** We introduce and formulate the problem of *latent network summarization*, which is complementary yet fundamentally different from network embedding.
- **Computational Framework.** We propose MULTI-LENS, which expresses a class of methods for latent network summarization. MULTI-LENS naturally supports inductive learning, on-the-fly embedding computation for all or a *subset* of nodes.
- **Time- and Space-efficiency.** MULTI-LENS is *scalable* with time complexity linear on the number of edges, and *space-efficient* with size independent of the graph size. Besides, it is *parallelizable* as the node computations are independent of each other.
- **Empirical analysis on real datasets.** We apply MULTI-LENS to event detection and link prediction over real-world heterogeneous graphs and show that it is 3.5%-34.3% more accurate than state-of-the-art embedding methods while requiring 80-2152× less output storage space for datasets with *millions* of edges.

Next we formally introduce the latent network summarization problem and then describe our proposed framework.

## 2 LATENT NETWORK SUMMARIZATION

Intuitively, the problem of *latent network summarization* aims to learn a compressed representation that captures the main structural information of the network and depends only on the complexity of the network instead of its size. More formally:

DEFINITION 1 (LATENT NETWORK SUMMARIZATION). *Given an arbitrary graph $G = (V, E)$ with $|V| = N$ nodes and $|E| = M$ edges, the goal of latent network summarization is to map the graph $G$ to a low-dimensional $K \times C$ representation $\mathcal{J}$ that summarizes the structure of $G$, where $K, C \ll N, M$ are independent of the graph size. The output latent representations should be usable in data mining tasks, and sufficient to derive all or a subset of node embeddings on the fly for learning tasks (e.g., link prediction, classification).*

Compared to the network embedding problem, latent network summarization differs in that it aims to derive a *size-independent* representation of the graph. This can be achieved in the form of supergraphs [19] (in the original graph space) or aggregated clusters trivially, but the compressed latent network summary in Definition 1 also needs to be able to derive the node embeddings, which is not the goal of traditional graph summarization methods.

In general, based on our definition, a latent network summarization approach should satisfy the following key properties: **(P1)** generality to handle arbitrary network with multiple node types, relationship types, edge weights, directionality, unipartite or k-partite structure, *etc.* **(P2)** high compression rate, **(P3)** natural support of inductive learning, and **(P4)** ability to on-the-fly derive node embeddings used in follow-up tasks.

**Table 1: Summary of symbols and notations**

| Symbol | Definition |
|---|---|
| $G = (V, E)$ | heterogeneous network with $\|V\| = N$ nodes and $\|E\| = M$ edges |
| $\mathbf{A}$ | adjacency matrix of $G$ with row $i$ $\mathbf{A}_{i,:}$ and column $i$ $\mathbf{A}_{:,i}$ |
| $\mathcal{T}_V$, $\mathcal{T}_E$ | sets of object types and edge types, respectively |
| $\mathcal{N}_i$, $\mathcal{N}_i^t$ | non-typed / types (1-hop) neighborhood or egonet of node $i$ |
| $\ell$, $L$ | index for level & total number of levels (*i.e.*, max order of a rel. fns) |
| $\mathcal{B}$ | $=\{\mathbf{b}_i\}$ set of initial feature vectors in length $N$ |
| $\mathcal{F}_r$ | $=\{\mathcal{F}_r^{(1)}, \ldots, \mathcal{F}_r^{(L)}\}$, ordered set of relational functions across levels |
| $\mathcal{F}_b$ | $=\{f_{b_i}\}$, set of base graph functions (special relational functions) |
| $\Phi$ | $=\{\phi_i\}$, set of relational operators |
| $\mathbf{F}^{(0)}$ | $N \times \|\mathcal{B}\|$ base feature matrix derived by the base graph functions $\mathcal{F}_b$ |
| $\mathbf{F}^{(\ell)}$ | $N \times \left(\|\mathcal{B}\| \cdot \|\Phi\|^{\ell}\right)$ generated feature matrix for level $\ell$ |
| $K^{(\ell)}$, $K$ | dimensionality of embeddings at level-$\ell$ and the final dimensionality |
| $\mathbf{H}^{(\ell)}$ | $N \times \|\mathcal{F}_r^{(\ell)}\|$ histogram-based representation of feature matrix $\mathbf{F}^{(\ell)}$ |
| $\mathbf{S}^{(\ell)}$ | low-rank latent graph summary at level $\ell$ |

## 3 MULTI-LENS FRAMEWORK

To efficiently address the problem of latent network summarization introduced in Section 2, we propose MULTI-LENS, which expresses a class of latent network summarization methods that satisfies all desired properties **(P1-P4)**. The summary $\mathcal{J}$ given by MULTI-LENS contains (i) necessary operators for aggregating node-wise structural features automatically and (ii) subspace vectors on which to derive the embeddings. We give the overview in Figure 2 and list the main symbols and notations used in this work in Table 1.

At a high level, MULTI-LENS leverages generally-defined relational operators to capture structural information from node neighborhoods in arbitrary types of networks. It recursively applies these operators over node neighborhoods to produce both linear and non-linear functions that characterize each node at different distances (§ 3.2). To efficiently derive the contextual space vectors, MULTI-LENS first generates histogram-based heterogeneous contexts for nodes (§ 3.3), and then obtains the summary via low-dimensional approximation (§ 3.4). We include the empirical justification of our design choices in the Appendix. Before discussing each step and its rationale, we first present some preliminaries that serve as building blocks for MULTI-LENS.

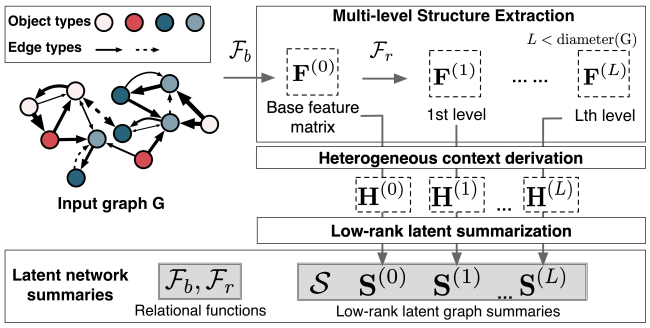



**Figure 2: Overview of MULTI-LENS. Dashed boxes: intermediate results that do not need to store; shaded boxes: outputs that need storing. The size of the latent network summaries, $\mathcal{J} = \{\mathcal{F}, \mathcal{S}\}$, is independent of $N, M$.**

## 3.1 Preliminaries

Recall that our proposed problem definition (§ 2) applies to any arbitrary graph **(P1)**. As a general class, we refer to heterogeneous (information) networks or typed networks.

Definition 2 (Heterogeneous network). *A heterogeneous network is defined as $G = (V, E, \theta, \xi)$ with node-set $V$, edge-set $E$, a function $\theta : V \rightarrow \mathcal{T}_V$ mapping nodes to their types, and a function $\xi : E \rightarrow \mathcal{T}_E$ mapping edges to their types.*

We assume that the network is directed and weighted with unweighted and undirected graphs as special cases. For simplicity, we will refer to a graph as $G(V, E)$. Within heterogeneous networks, the typed neighborhood or egonet[1] $\mathcal{N}_t$ of a node is defined as follows:

Definition 3 (Typed neighborhood $\mathcal{N}^t$). *Given an arbitrary node $i$ in graph $G = (V, E)$, the typed $t$* neighborhood $\mathcal{N}_i^t$ *is the set of nodes with type $t$ that are reachable by following directed edges $e \in E$ originating from $i$ with 1-hop distance and $i$ itself.*

The *neighborhood* of node $i$, $\mathcal{N}_i$, is a superset of the typed neighborhood $\mathcal{N}_i^t$, and includes nodes in the neighborhood of $i$ regardless of their types. Higher-order neighborhoods are defined similarly, but more computationally expensive to explore. For example, the $k$-hop neighborhood denotes the set of nodes reachable following directed edges $e \in E$ originating from node $i$ within $k$-hop distance.

The goal of latent network summarization is to find a size-independent representation that captures the *structure* of the network and its underlying nodes in the latent space. Capturing the structure depends on the semantics of the network (e.g., weighted, directed), and thus different ways are needed for different input networks types. To generalize to *arbitrary* networks, we leverage relational operators and functions [28].

Definition 4 (Relational operator). *A relational operator $\phi(\mathbf{x}, \mathcal{R}) \in \Phi$ is defined as a basic function (*e.g.*, sum) that operates on a feature vector $\mathbf{x}$ associated with a set of related elements $\mathcal{R}$ and returns a single value.*

For example, let $\mathbf{x}$ be an $N \times 1$ vector and $\mathcal{R}$ the neighborhood $\mathcal{N}_i$ of node $i$. For $\phi$ being the *sum*, $\phi(\mathbf{x}, \mathcal{R})$ would return the count of neighbors reachable from node $i$ (unweighted out degree).

Definition 5 (Relational function). *A relational function $f \in \mathcal{F}$ is defined as a composition of relational operators $f = \big(\phi_1 \circ \cdots \circ \phi_{h-1} \circ \phi_h\big)(\mathbf{x}, \mathcal{R})$ applied to feature values in $\mathbf{x}$ associated with the set of related nodes $\mathcal{R}$. We say that $f$ is order-$h$ iff the feature vector $\mathbf{x}$ is applied to $h$ relational operators.*

Together, relational operators and relational functions comprise the building blocks of our proposed method, Multi-LENS. Iterative computations over the graph or a subgraph (*e.g.*, node neighborhood) generalize for inductive/across-network transfer learning tasks. Moreover, relational functions are general and can be used to derive commonly-used graph statistics. As an example, the out-degree of a specific node is derived by applying order-1 relational functions on the adjacency matrix over its the egonet, *i.e.*, out-deg(i) = $\sum(\mathbf{A}_{i:}, \mathcal{N})$ regardless of object types.

[1] In this work we use neighborhood and egonet interchangeably.

## 3.2 Multi-level Structure Extraction

We now start describing our proposed method, Multi-LENS. The first step is to extract multi-level strcuture around the nodes. To this end, as we show in Figure 2, Multi-LENS first generates a set of simple node-level features to form the base feature matrix $\mathbf{F}^{(0)}$ via the so-called base graph functions $\mathcal{F}_b$. It then composes new functions by iteratively applying a set of relational operators $\Phi$ over the neighborhood to generate new features. Operations in both $\mathcal{F}_b$ and $\Phi$ are generally defined to satisfy **(P1)**.

*3.2.1 Base Graph Functions.* As a special relational function, each base graph function $f_b \in \mathcal{F}_b$ consists of relational operators that perform on an initial $N \times 1$ feature vector $\mathbf{b} \in \mathcal{B}$. The vector $\mathbf{b}$ could be given as the row/column of the adjacency matrix corresponding to node $i$, or some other derived vector related to the node (*e.g.*, its distance or influence to every node in the graph). Following [28], the simplest case is $f_b = \sum$, which captures simple base features such as in/out/total degrees. We denote applying the same base function to the egonets of all the nodes in graph $G$ as follows:

$$f_b\langle \mathbf{b}, \mathcal{N}\rangle = [f_b(\mathbf{b}, \mathcal{N}_1), f_b(\mathbf{b}, \mathcal{N}_2), \ldots, f_b(\mathbf{b}, \mathcal{N}_N)]^T, \mathbf{b} \in \mathcal{B} \quad (1)$$

which forms an $N \times 1$ vector. For example, $f_b = \sum\langle \mathbf{A}_{i:}, \mathcal{N}\rangle$ enumerates the out-degree of all nodes in $G$. By applying $f_b$ on each initial feature $\mathbf{b}$, *e.g.*, $\mathbf{1}^{N\times 1}$ or row/column of adjacency matrix $\mathbf{A}$, we obtain the $N \times B$ base matrix $\mathbf{F}^{(0)}$:

$$\mathbf{F}^{(0)} = [f_b\langle \mathbf{b}_1, \mathcal{N}\rangle, f_b\langle \mathbf{b}_2, \mathcal{N}\rangle, \ldots, f_b\langle \mathbf{b}_B, \mathcal{N}\rangle], \mathbf{b}_{1\cdots B} \in \mathcal{B} \quad (2)$$

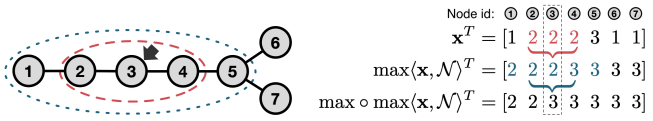
which aggregates all structural features of the nodes within $\mathcal{N}$. The specific choice of initial vectors $\mathbf{b}$ is not very important as the composed relational functions (§ 3.2.2) extensively incorporate both linear and nonlinear structural information automatically. We empirically justify Multi-LENS on the link prediction task over different choices of $\mathcal{B}$ to show its insensitivity in Appendix § B.3.

*3.2.2 Relational Function Compositions.* To derive complex & non-linear node features automatically, Multi-LENS iteratively applies operators $\phi \in \Phi$ (e.g., *mean*, *variance*, *sum*, *max*, *min*, *l2-distance*) to lower-order functions, resulting in function compositions. $\ell$ such compositions of functions over a node's egonet $\mathcal{N}_i$ captures *higher-order structural features* associated with the $\ell-$hop neighborhoods. For example, assuming $\mathbf{x}$ is the vector consisting of node-wise degrees, the *max* operator captures the maximum degree in the neighborhood $\mathcal{N}$ of a node. The application of the *max* operator to all the nodes forms a new feature vector $\max\langle \mathbf{x}, \mathcal{N}\rangle$ where each entry records the maximum degree in the corresponding neighborhood. Fig. 3 shows that the maximum degree of node $\{2, 3, 4\}$ is aggregated for node 3 in $\max\langle \mathbf{x}, \mathcal{N}\rangle$ By iteratively applying *max* to $\max\langle \mathbf{x}, \mathcal{N}\rangle$ in the same way, the maximum value from broader neighborhood $\mathcal{N}$ is aggregated, which is equivalent to finding the maximum degree in the 2-hop neighborhood. Fig. 3b depicts this process for node 3.

Formally, at level $\ell \in \{1, \ldots, L\}$, a new function is composed as:

$$f^{(\ell)} = \phi \circ f^{(\ell-1)}, \forall \phi \in \Phi \quad (3)$$

where $L <$ `diam`$(G)$ or the diameter of $G$, and $f^{(0)} = f_b$(§ 3.2.1). We formally define some operators $\phi \in \Phi$ in Appendix B.1. Applying $f^{(\ell)}$ to $\mathbf{F}^{(0)}$ generates order-$\ell$ structural features of the graph as $\mathbf{F}^{(\ell)}$. In practice, Multi-LENS recursively generates $\mathbf{F}^{(\ell)}$ from $\mathbf{F}^{(\ell-1)}$




**(a) 1- and 2-hop neighborhood of node 3**

**(b) $\max \circ \max\langle \mathbf{x}, \mathcal{N} \rangle$ captures the max degree in 2-hop neighborhood**

**Figure 3: The composition of relational functions incorporates node degrees (column vector x) in expanded subgraphs.**

by applying a total of $|\Phi|$ operators. The particular order in which relational operators are applied records how a function is generated. MULTI-LENS then collects the composed relational functions per level into $\mathcal{F}_r$ as a part of the latent summary.

In terms of space, Equation (3) indicates the dimension of $\mathcal{F}_r$ grows exponentially with $|\Phi|$, *i.e.*, $|\mathcal{F}_r^{(\ell)}| = |\mathcal{B}||\Phi|^{(\ell)}$, which is also the number of columns in $\mathbf{F}^{(\ell)}$. However, the max level $L$ is bounded with the diameter of $G$, that is $L \leq \texttt{diam}(G) - 1$ because functions with orders higher than that will capture the same repeated structural information. Therefore, the size of $\mathcal{F}_r$ is also bounded with $L$. Although the number of relational functions grows exponentially, real-world graphs are extremely dense with small diameters $\texttt{diam}(G) \propto \log\log N$ [7]. In our experiments in § 4, $|\mathcal{F}_r| \approx 1000$ for $|\mathcal{B}| = 3$ base functions, $|\Phi| = 7$ operators, and $L = 2$ levels.

## 3.3 Heterogeneous Context

So far we have discussed how to obtain the base structural feature matrix $\mathbf{F}^{(0)}$ and the multi-level structural feature representations $\mathbf{F}^{(\ell)}$ by recursively employing the relational functions. As we show empirically in supplementary material B.2, directly deriving the structural embeddings based on these representations leads to low performance due to skewness in the extracted structural features. Here we discuss an intermediate transformation of the generated matrices that helps capture rich contextual patterns in the neighborhoods of each node, and eventually leads to a powerful summary.

*3.3.1 Handling skewness.* For simplicity, we first discuss the case of a homogeneous network $G$ with a single node and edge type, and undirected edges. To handle the skewness in the higher-order structural features (§ 3.2) and more effectively capture the structural identity of each node within its context (i.e., non-typed neighborhood), we opt for an intuitive approach: for each node $i$ and each base/higher-order feature $j$, we create a histogram $\mathbf{h}_{ij}$ with $c$ bins for the nodes in its neighborhood $\mathcal{N}_i$. Variants of this approach are used to capture node context in existing representation learning methods, such as struc2vec [25] and xNetMF [13]. In our setting, the structural identity of node $i$ is given as the concatenation of all its feature-specific histograms.

$$\mathbf{h}_i = \left[\, \mathbf{h}_{i1} \;\; \mathbf{h}_{i2} \;\; \cdots \;\; \mathbf{h}_{iZ} \,\right], \tag{4}$$

where $Z = |\mathcal{B}| + \sum_{\ell=1}^{L} |\mathcal{B}| \cdot |\Phi|^{\ell}$ is the total number of histograms, or the number of base and higher-order features. Each histogram is in logarithmic scale to better describe the power-law-like distribution in graph features and has a total of $c$ bins. By stacking all the nodes' structural identities vertically, we obtain a rich histogram-based context matrix $\mathbf{H} = \left[\, \mathbf{h}_1; \mathbf{h}_2; \cdots; \mathbf{h}_N \,\right]$ as shown in Fig. 4.

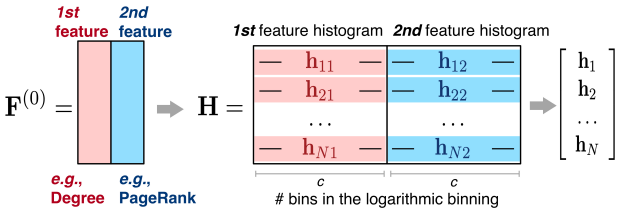



**Figure 4: Example of creating histogram-based matrix representation $\mathbf{H}^{(0)}$ with $Z = 2$ features in the base feature matrix $\mathbf{F}^{(0)}$. A single object / edge type and no edge directionality is assumed here for simplicity.**

*3.3.2 Handling object/edge types and directionality.* The histogram-based representation that we described above can be readily extended to handle any arbitrary network $G$ with multiple object types, edge types and directed edges **(P1)**. The idea is to capture the structural identity of each node $i$ within its *different* contexts:

- $\mathcal{N}_i^t$ or $\mathcal{N}_i^\tau$: the neighborhood that contains only nodes of type $t \in \mathcal{T}_V$ or edges of type $\tau \in \mathcal{T}_E$, and
- $\mathcal{N}_i^+$ or $\mathcal{N}_i^-$: the neighborhood with only outgoing or incoming edges for node $i$.

For example, to handle different object types, we create a context matrix $\mathbf{H}_o^t$ by restricting the histograms on neighbors of type $t$, $\mathcal{N}_i^t$. These per-type matrices can be stacked into a tensor $\underline{\mathcal{H}}$, with each slice corresponding to a node-level histogram, $\mathbf{H}_o^t$ of object type, $t$. Alternatively, the tensor can be matricized by frontal slices. By further restricting the neighborhoods to contain specific edge types and/or directionality in a similar manner, we can obtain the histogram-based representations $\mathbf{H}_e^t$ and $\mathbf{H}_d^t$, respectively.

By imposing all of the restrictions at once, we can also obtain context matrix $\mathbf{H}$ that accounts for all types of heterogeneity. We discuss this step with more details that may be necessary for reproducibility in § C of the supplementary material.

## 3.4 Latent Summarization

The previous two subsections can be seen as a general framework for automatically extracting, linear and non-linear, higher-order structural features that constitute the nodes' contexts at multiple levels $\ell$. Unlike embedding methods that generate graph-size dependent node representations, we seek to derive a compressed latent representation of $G$ **(P2)** that supports on-the-fly generation of node embeddings and (inductive) downstream tasks **(P3,P4)**. Although graph summarization methods [19] are relevant as they represent an input graph with a summary or supergraph, it is infeasible to generate latent node representations due to the incurred information loss. Thus, such methods, which have different end goals, do not satisfy **(P4)**.

*3.4.1 Multi-level Summarization.* MULTI-LENS explores node similarity based on the assumption that similar nodes should have similar structural context over neighborhoods of different hops. Given the histogram-based context matrix $\mathbf{H}^{(\ell)}$ that captures the heterogeneity of feature values associated with the $\ell-$order egonets in $G$ (§ 3.2.2), MULTI-LENS obtains the level-$\ell$ summarized representation $\mathbf{S}^{(\ell)}$ via factorization $\mathbf{H}^{(\ell)} = \mathbf{Y}^{(\ell)}\mathbf{S}^{(\ell)}$, where $\mathbf{Y}^{(\ell)}$ is the dense node embedding matrix that we do *not* store. Then, the latent summary $\mathcal{J}$ consists of the set of relational functions $\mathcal{F}_r$ (§ 3.2), and

the multi-level summarized representations $\mathcal{S} = \{\mathbf{S}^{(1)}, \ldots, \mathbf{S}^{(\ell)}\}$. Though any technique can be used (e.g., NMF), we give the factors based on SVD for illustration:

$$\texttt{level-}\ell\ \texttt{node embeddings (not stored):}\ \mathbf{Y}^{(\ell)} = \mathbf{U}^{(\ell)}\sqrt{\Sigma^{(\ell)}} \quad (5)$$

$$\texttt{level-}\ell\ \texttt{summarized representation:}\quad \mathbf{S}^{(\ell)} = \sqrt{\Sigma^{(\ell)}}\mathbf{V}^{(\ell)T} \quad (6)$$

where $\Sigma^{(\ell)}$ are the singular values of $\mathbf{H}^{(\ell)}$, and $\mathbf{U}^{(\ell)T}$, $\mathbf{V}^{(\ell)T}$ are its left and right singular vectors, respectively.

Intuitively, $\mathbf{S}^{(\ell)}$ contains the best-fit $K^{(\ell)}$-dimensional subspace vectors for node context $\mathbf{H}^{(\ell)}$ in the neighborhood at order-$\ell$. The summary representations across different orders form the hierarchical summarization of $G$ that contains both local and global structural information, and the derived embedding matrix $\mathbf{Y}^{(\ell)}$ also preserves node similarity at multiple levels. There is no need to store any of the intermediate matrices $\mathbf{F}^{(\ell)}$ and $\mathbf{H}^{(\ell)}$, nor the node embeddings $\mathbf{Y}^{(\ell)}$. The former two matrices can be derived on the fly given the composed relational functions $\mathcal{F}_r$. Then, the latter can be efficiently estimated using the obtained sparse $\mathbf{H}^{(\ell)}$ matrix and the stored summarized matrix $\mathbf{S}^{(\ell)}$ through SVD (§ 3.6 gives more details). Moreover, since the elements of the summary $\mathcal{S}$, i.e., the relational functions $\mathcal{F}_r$ and the factorized matrices, are independent of the nodes or edges of the input graph, both require trivial storage and achieve compression efficiency **(P2)**. We provide the pseudo-code of Multi-LENS in Algorithm 1.

We note that the relational functions $\mathcal{F}_r$ are a key enabling factor of our summarization approach. Without them, other embedding methods cannot benefit from our proposed summarized representations $\mathcal{S}$, nor reconstruct the node context and embeddings.

*3.4.2 Inductive Summaries* **(P3)**. The higher-order features derived from the set of relational functions $\mathcal{F}_r$ are structural, and thus generalize across graphs [1, 13, 14] and are independent of node IDs. As such, the factorized matrices in $\mathcal{S}$ learned on $G$ can be transferred to another graph $G'$ to learn the node embeddings $\mathbf{Y}^{(\ell)}$ of a new, previously unseen graph $G'$ as:

$$\mathbf{Y}'^{(\ell)} = \mathbf{H}'^{(\ell)}(\mathbf{S}^{(\ell)})^{\dagger} \quad (7)$$

where $\mathbf{S}^{(\ell)} \in \mathcal{S}$ is learned on $G$, $\dagger$ denotes the pseudo-inverse, and $\mathbf{H}'^{(\ell)}$ is obtained via applying $\mathcal{F}_r$ to $G'$. The pseudo-inverse, $(\mathbf{S}^{(\ell)})^{\dagger}$ can be computed efficiently through SVD as long as the rank of $\mathbf{S}^{(\ell)}$ is limited (*e.g.*, empirically setting $K^{(\ell)} \leq 128$) [4].

Equation (7) requires the same dimensionality $K^{(\ell)} = K'^{(\ell)}$ and the same number of bins of histogram context matrices $c = c'$ at each level $\ell$. The embeddings learned inductively reflect the node-wise structural difference between graphs, $G$ and $G'$, which can be used in applications of graph mining and time-evolving analysis. We present an application of temporal event detection in § 4.4.

*3.4.3 On-the-fly embedding derivation* **(P4)**. Given the summarized matrix $\mathbf{S}^{(\ell)}$ at level $\ell$, the embeddings of specific nodes that are previously seen or unseen can be derived efficiently. Multi-LENS first applies $\mathcal{F}_r$ to derive their heterogeneous context $\mathbf{H}^{(l)}_{\text{sub}}$ based on the graph structure, and then obtains the embeddings via Eq. (7). We concatenate $\mathbf{Y}^{(\ell)}$ given as output at each level to form the final node embeddings [31]. Given that the dimension of embeddings is $K^{(\ell)}$ at level $\ell$, the final embedding dimension is $K = \sum_{\ell=1}^{L} K^{(\ell)}$.

**Algorithm 1** Multi-LENS

**Input:** (un)directed heterogeneous graph $G$, a set of relational operators $\Phi$; layer-wise embedding dimensionality $K^{(\ell)}$, for $K = \sum_{\ell=1}^{L} K^{(\ell)}$ dimensions in total; number of bins $c$ for the histogram representation
**Output:** Summary $\mathcal{J} = \{\mathcal{F}, \mathcal{S}\}$

$\mathcal{F}_r \leftarrow f_b$ ▷ Base graph functions: Eq. (1)
Initialize $\mathbf{F}^{(0)}$ ▷ Base feature matrix Eq. (2)
**for** $\ell = 1, \ldots, L$ **do** ▷ multi-level summarization
  **for** $i = 1, \ldots, |\Phi|$ **do** ▷ relational operators
    **parallel for** $j = 1, \ldots, BR^{\ell-1}$ **do** ▷ Columns in $\mathbf{F}^{(l)}$
      $f = \phi_i \circ f_j^{(\ell-1)}$ ▷ Compose func. in $\mathcal{F}_r^{(\ell-1)}$
      $\mathbf{F}^{(\ell)} = \mathbf{F}^{(\ell)} \cup \phi_i \langle \mathbf{F}_{:,j}^{(\ell-1)}, \mathcal{N} \rangle$ ▷ Feature concatenation
  Derive heterogeneous context $\mathbf{H}^{(\ell)}$ ▷ § 3.3 and Eq. (10)
  $\mathbf{S}^{(\ell)} = \sqrt{\Sigma^{(\ell)}}\mathbf{V}^{(\ell)T}$ ▷ SVD: $\mathbf{H}^{(\ell)} = \mathbf{U}^{(\ell)}\Sigma^{(\ell)}\mathbf{V}^{(\ell)T}$
  $\mathcal{F}_r \leftarrow \mathcal{F}_r \cup f$, $\mathcal{S} \leftarrow \mathcal{S} \cup \mathbf{S}^{(\ell)}$

## 3.5 Generalization

Here we discuss the generalizations of our proposed approach to labeled and attributed graphs. It is straightforward to see that homogeneous, bipartite, signed, and labeled graphs are all special cases of heterogeneous graphs with $|\mathcal{T}_V| = |\mathcal{T}_E| = 1$ types, $|\mathcal{T}_V| = 2$ and $|\mathcal{T}_E| = 1$ types, $|\mathcal{T}_V| = 1$ and $|\mathcal{T}_E| = 2$ types, and $|\mathcal{T}_V| =$ #(node labels) and $|\mathcal{T}_E| =$ #(edge labels) types, respectively. Therefore, our approach naturally generalizes to all of these graphs. Other special cases include k-partite and attributed graphs.

Multi-LENS also supports attributed graphs that have multiple attributes per node or edge (instead of a single label): Given an initial set of attributes organized in an attribute matrix $\mathbf{F}_a$, we can concatenate $\mathbf{F}_a$ with the base attribute matrix and apply our approach as before. Alternatively, we can transform the graph into a labeled one by applying a labeling function $\chi : \mathbf{x} \rightarrow y$ that maps every node's attribute vector $\mathbf{x}$ to a label $y$ [1]. Besides, our proposed method is easy to parallelize as the relational functions are applied to the subgraphs of each node independently, and the feature values are computed independently.

## 3.6 Complexity Analysis

*3.6.1 Computational Complexity.* Multi-LENS is linear to the number of nodes $N$ and edges $M$. Per level, it derives the histogram-based context matrix $\mathbf{H}^{(\ell)}$ and performs a rank-$K^{(\ell)}$ approximation.

Lemma 3.1. *The computational complexity of Multi-LENS is*

$$O((c|\mathcal{F}_r||\mathcal{T}_V||\mathcal{T}_E| + K^2)N + M).$$

We give the proof in Appendix A.1. As indicated in § 3.2, the number of features in $\mathbf{H}^{(\ell)}$ across $L$ layers is equivalent to the number of composed relational functions $|\mathcal{F}_r|$. Since $|\mathcal{F}_r|$ is bounded with $L$ and $L < \texttt{diam}(G)$, the term $(c|\mathcal{F}_r||\mathcal{T}_V||\mathcal{T}_E| + K^2)$ forms a constant related to graph heterogeneity and structure.

*3.6.2 Space Complexity.* The runtime and output compression space complexity of Multi-LENS is given in Lemma 3.2. In the runtime at level $\ell$, Multi-LENS leverages $\mathbf{F}^{(\ell-1)}$ to derive $\mathbf{F}^{(\ell)}$ and $\mathbf{H}^{(\ell)}$, which comprise two terms in the runtime space complexity. We detail the proof in Appendix A.2

Lemma 3.2. *The Multi-LENS space complexity during runtime is* $O((c|\mathcal{F}_r||\mathcal{T}_V||\mathcal{T}_E| + |\mathcal{F}_r|)N)$. *The space needed for the* output *of Multi-LENS is* $O(cK|\mathcal{F}_r||\mathcal{T}_V||\mathcal{T}_E| + |\mathcal{F}_r|)$.

The output of Multi-LENS that needs to be stored (i.e., set of relational functions $\mathcal{F}_r$ and summary matrices in $\mathcal{S}$) is independent of $N, M$. Compared with output embeddings with complexity $O(NK)$ given by existing methods, Multi-LENS satisfies the crucial property we desire **(P2)** from latent summarization (Def. 1).

# 4 EXPERIMENTS

In our evaluation we aim to answer four research questions:

- **Q1** How much space do the Multi-LENS summaries save (P2)?
- **Q2** How does Multi-LENS perform in machine learning tasks, such as link prediction in heterogeneous graphs (P1)?
- **Q3** How well does it perform in inductive tasks (P3)?
- **Q4** Does Multi-LENS scale well with the network size?

We have discussed on-the-fly embedding derivation (P4) in § 3.4.3.

## 4.1 Experimental Setup

*4.1.1 Data.* In accordance with **(P1)**, we use a variety of real-world heterogeneous network data from Network Repository [26]. We present their statistics in Table 2.

- **Facebook** [11] is a homogeneous network that represents friendship relation between users.
- **Yahoo! Messenger Logs** [28] is a heterogeneous network of Yahoo! messenger communication patterns, where edges indicate message exchanges. The users are associated with the locations from which they have sent messages.
- **DBpedia**[2] is an unweighted, heterogeneous subgraph from DBpedia project consisting of 4 types of entities and 3 types of relations: user-occupation, user-work ID, work ID-genre.
- **Digg**[2] is a heterogeneous network that records the voting behaviors of users to stories they like. Node types include users and stories. Each edge represents one vote or a friendship.
- **Bibsonomy**[2] is a k-partite network that represents the behaviors of users assigning tags to publications.

*4.1.2 Baselines.* We compare Multi-LENS with baselines commonly used in graph summarization, matrix factorization and representation learning over networks, namely, they are: **(1)** Node aggregation or NA for short [2, 33], **(2)** Spectral embedding or SE [32], **(3)** LINE [31], **(4)** DeepWalk or DW [23], **(5)** Node2vec or n2vec [11], **(6)** struc2vec or s2vec [25], **(7)** DNGR [6], **(8)** GraRep or GR [5], **(9)** Metapath2vec or m2vec [8], and **(10)** AspEm [30], **(11)** Graph2Gauss or G2G [3]. To run baselines that do not explicitly support heterogeneous graphs, we align nodes of the input graph according to their object types and re-order the IDs to form the homogeneous representation. In node aggregation, CoSum [33] ran out of memory due to the computation of pairwise node similarity. We use Louvain [2] as an alternative that scales to large graphs and forms the basis of many node aggregation methods.

*4.1.3 Configuration.* We evaluate Multi-LENS with $L = 1$ and $L = 2$ to capture subgraph structural features in 1-hop and 2-hop neighborhoods, respectively, against the optimal performance achieved by the baselines. We derive in-/out- and total degrees to construct the $N \times 3$ base feature matrix $\mathbf{F}^{(0)}$. Totally, we generate $\approx 1000$ composed functions, each of which corresponds to a column vector in $\mathbf{F}$. For fairness, we do not employ parallelization and terminate processes exceeding 1 day. The output dimensions of all node representations are set to be $K = 128$. We also provide an ablation study in terms of the choice of initial vectors, different sets of relational operators in supplementary material B.2-B.4. For reproducibility, we detail the configuration of all baselines and Multi-LENS in Appendix B.1. The source code is available at https://github.com/GemsLab/MultiLENS.

[2] http://networkrepository.com/

**Table 2: Statistics for the heterogeneous networks that we use in our experiments.**

| Data | #Nodes | #Edges | #Node Types | Graph Type |
|---|---|---|---|---|
| facebook | 4 039 | 88 234 | 1 | unweighted |
| yahoo-msg | 100 058 | 1 057 050 | 2 | weighted |
| dbpedia | 495 936 | 921 710 | 4 | unweighted |
| digg | 283 183 | 4 742 055 | 2 | unweighted |
| bibsonomy | 977 914 | 3 754 828 | 3 | weighted |

## 4.2 Compression rate of Multi-LENS

The most important question for our latent summarization method (Q1) is about how well it compresses large scale heterogeneous data **(P2)**. To show Multi-LENS's benefits over existing embedding methods, we measure the storage space for the generated embeddings by the baselines that ran successfully. In Table 3 we report the space required by the Multi-LENS summaries in MB, and the space that the outputs of our baselines require *relative* to the corresponding Multi-LENS summary. We observe that the latent summaries generated by Multi-LENS take up very little space, well under 1MB each. The embeddings of the representation learning baselines take up $80 - 2152\times$ more space than the Multi-LENS summaries on the larger datasets. On Facebook, which is a small dataset with 4K nodes, the summarization benefit is limited; the baseline methods need about $3 - 12\times$ more space. In addition, the node-aggregation approach takes up to $12\times$ storage space compared to our latent summaries, since it generates an $N \times 1$ vector that depends on graph size to map each node to a supernode. This further demonstrates the advantage of our graph-size independent latent summarization.

**Table 3: Output storage space required for embedding methods relative to the Multi-LENS summaries (given in MB). Multi-LENS requires $3-2152\times$ less output storage space than embedding methods.**

| Data | SE | LINE | n2vec | DW | m2vec | AspEm | G2G | **ML** (MB) |
|---|---|---|---|---|---|---|---|---|
| facebook | 8.13x | 8.48x | 12.79x | 12.84x | 3.82x | 8.50x | 9.17x | **0.58** |
| yahoo | 187.1x | 180.0x | 242.2x | 231.0x | 79.8x | 197.4x | 195.8x | **0.62** |
| dbpedia | 710.0x | 714.2x | 996.4x | 996.2x | - | 749.2x | 743.6x | **0.81** |
| digg | 608.2x | 612.8x | 848.9x | 830.3x | 259.9x | 641.7x | 635.2x | **0.54** |
| bibson. | 1512.1x | 1523.0x | 2152.5x | 2152.5x | - | 1595.8x | - | **0.75** |

**Table 4: Link prediction: node embeddings derived by MULTI-LENS (ML) outperforms all baselines measured by every evaluation metric. Specifically, MULTI-LENS outperforms embedding baselines by 3.46% ∼ 34.34% in AUC and 3.71% ∼ 31.33% in F1 on average. It outperforms even more over the aggregation-based methods. The asterisk * denotes statistically significant improvement over the best baseline at $p < 0.01$ in a two-sided t-test. OOT = Out Of Time (12 hours), OOM = Out Of Memory (16GB).**

| Data | Metric | NA | SE | LINE | DW | n2vec | GR | s2vec | DNGR | m2vec | AspEm | G2G | **ML**($L = 1$) | **ML**($L = 2$) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| facebook | **AUC** | 0.6213 | 0.6717 | 0.7948 | 0.7396 | 0.7428 | 0.8157 | 0.8155 | 0.7894 | 0.7495 | 0.5886 | 0.7968 | 0.8703 | **0.8709*** |
| | **ACC** | 0.5545 | 0.5995 | 0.7210 | 0.6460 | 0.6544 | 0.7368 | 0.7388 | 0.7062 | 0.7051 | 0.5628 | 0.7274 | **0.7920*** | 0.7904 |
| | **F1 macro** | 0.5544 | 0.5716 | 0.7210 | 0.6296 | 0.6478 | 0.7367 | 0.7387 | 0.7060 | 0.7041 | 0.5628 | 0.7273 | **0.7920*** | 0.7905 |
| yahoo-msg | **AUC** | 0.7189 | 0.5375 | 0.6745 | 0.7715 | 0.7830 | 0.7535 | | | 0.6708 | 0.5587 | 0.6988 | 0.8443 | **0.8446*** |
| | **ACC** | 0.2811 | 0.5224 | 0.6269 | 0.6927 | 0.7036 | 0.6825 | OOT | OOM | 0.6164 | 0.5379 | 0.6564 | **0.7587*** | **0.7587*** |
| | **F1 macro** | 0.2343 | 0.5221 | 0.6265 | 0.6897 | 0.7016 | 0.6821 | | | 0.6145 | 0.5377 | 0.6562 | **0.7577*** | **0.7577*** |
| dbpedia | **AUC** | 0.6002 | 0.5211 | 0.9632 | 0.8739 | 0.8774 | | | | | 0.6364 | 0.7384 | **0.9820*** | 0.9809 |
| | **ACC** | 0.3998 | 0.5399 | 0.9111 | 0.8436 | 0.8436 | OOM | OOT | OOM | OOT | 0.5869 | 0.6625 | **0.9186** | 0.9151 |
| | **F1 macro** | 0.2968 | 0.4539 | 0.9110 | 0.8402 | 0.8402 | | | | | 0.5860 | 0.6613 | **0.9186** | 0.9150 |
| digg | **AUC** | 0.7199 | 0.6625 | 0.9405 | 0.9664 | 0.9681 | | | | 0.9552 | 0.5644 | 0.8978 | **0.9894*** | 0.9893 |
| | **ACC** | 0.2801 | 0.6512 | 0.8709 | 0.9023 | 0.9049 | OOM | OOT | OOM | 0.8891 | 0.5459 | 0.8492 | **0.9596*** | 0.9590 |
| | **F1 macro** | 0.2660 | 0.6223 | 0.8709 | 0.9019 | 0.9046 | | | | 0.8890 | 0.5459 | 0.8492 | **0.9595*** | 0.9590 |
| bibsonomy | **AUC** | 0.7836 | 0.6694 | 0.9750 | 0.6172 | 0.6173 | | | | | 0.6127 | | **0.9909*** | 0.9909 |
| | **ACC** | 0.2164 | 0.6532 | 0.9350 | 0.5814 | 0.5816 | OOM | OOT | OOM | OOT | 0.5790 | OOM | **0.9485*** | 0.9466 |
| | **F1 macro** | 0.2070 | 0.6064 | 0.9349 | 0.5781 | 0.5782 | | | | | 0.5772 | | **0.9485*** | 0.9466 |

## 4.3 Link Prediction in Heterogeneous Graphs

For Q2, we investigate the performance of MULTI-LENS in link prediction task over heterogeneous graphs (**P1**). We use logistic regression with regularization strength = 1.0 and stopping criteria= $10^{-4}$. An edge $e_{ij}$ is represented by the concatenating the embeddings of its source and destination: $emb(e_{ij}) = [emb(i), emb(j)]$ as used in [28]. For each dataset $G(V, E)$, we create the subgraph $G'(V, E')$ by keeping all the nodes but randomly removing ∼ 40% edges. We run all methods on $G'$ to get node embeddings and randomly select $10\%|E|$ edges as the training data. Out of the removed edges, 25% ($10\%|E|$) are used as missing links for testing. We also randomly create the same amount of "fake edges" for both training and testing. Table 4 illustrates the prediction performance measured with AUC, ACC, and F1 macro scores.

We observe that MULTI-LENS outperforms the baselines measured by every evaluation metric. MULTI-LENS outperforms embedding baselines by 3.46% ∼ 34.34% in AUC and 3.71% ∼ 31.33% in F1 score. For runnable baselines designed for node embeddings in homogeneous graphs (baseline **3 - 8**), the experimental result is expected as MULTI-LENS incorporates heterogeneous contexts within 2-neighborhood in the node representation. It is worth noting that MULTI-LENS outperforms Metapath2vec and AspEm, both of which are designed for heterogeneous graphs. One reason behind is the inappropriate meta-schema specified, as Metapath2vec and AspEm require predefined meta-path / aspect(s) in the embedding. On the contrary, MULTI-LENS does not require extra input and captures graph heterogeneity automatically. We also observe the time and runtime space efficiency of MULTI-LENS when comparing with neural-network based methods (DNGR, G2G), GraRep and struc2vec on large graphs. Although the use of relational operators is similar to information propagation in neural-networks, MULTI-LENS requires less computational resource with promising results. Moreover, the MULTI-LENS summaries for both $L = 1$ and $L = 2$ levels achieve promising results, but generally we observe that there is a slight drop in accuracy for higher levels. This indicates that node context at higher levels may incorporate noisy, less-relevant higher-order structural features (§ 3.2.2).

## 4.4 Inductive Anomaly Detection

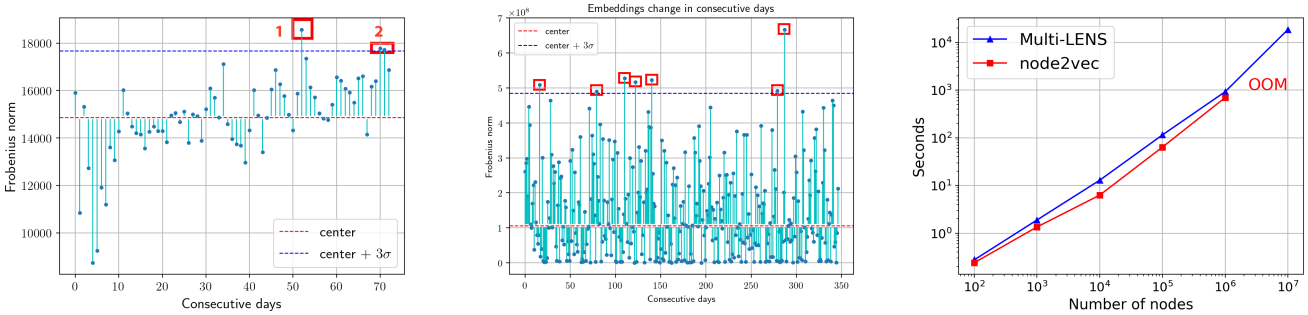
To answer Q3 about inductive learning, we first perform anomalous subgraph detection on both synthetic and real-world graphs. We also showcase the application of MULTI-LENS summaries on real-world event detection, in an inductive setting (**P3**).

*4.4.1 Anomalous Subgraph Detection.* Following the literature [21], we first generate two "background" graphs, $G_1$ and $G_2$. We then induce an anomalous subgraph into $G_2$ by randomly selecting $n$ nodes and adding edges to form an anomalous ER subgraph with $p$ and $n$ shown in Table 5. We leverage the summary learned from $G_1$ to learn node embeddings in $G_2$, we identify the top-$n$ nodes with the highest change in euclidean distance as anomalies, and report the precision in Table 5. In the synthetic setting, we generate two Erdős-Rényi (ER) graphs, $G_1^{\text{syn}}$ and $G_2^{\text{syn}}$, with $10^4$ nodes and average degree 10 ($p^{\text{back}} = 10^{-3}$). In the real-graph setting, we construct $G_1^{\text{real}}$ and $G_2^{\text{real}}$ using two consecutive daily graphs in the bibsonomy dataset.

In the synthetic scenario, we observe that MULTI-LENS gives promising results by successfully detecting nodes with the most deviating embedding values, except when the size of injection is small. In the case of very sparse ER injections ($p = 0.1$), the anomalies are not detectable over the natural structural deviation between $G_1^{\text{syn}}$ and $G_2^{\text{syn}}$. However, denser injections ($p \geq 0.3$) affect more significantly the background graph structure, which in turn leads to notable change in the MULTI-LENS embeddings for the affected

**Table 5: Anomalous Erdős-Rényi (ER) subgraphs (with $n$ nodes and probability $p$) detection precision on both synthetic and real-world graphs.**

| | REAL GRAPH | | | | | SYNTHETIC GRAPH | | |
|---|---|---|---|---|---|---|---|---|
| **p \ n** | **100** | **200** | **300** | **400** | **500** | **50** | **75** | **100** |
| **0.1** | 0.200 | 0.780 | 0.950 | 0.973 | 0.980 | 0.06 | 0.3333 | 0.81 |
| **0.3** | 0.870 | 0.960 | 0.990 | 0.995 | 0.996 | 1 | 1 | 1 |
| **0.5** | 0.920 | 0.990 | 0.993 | 1 | 1 | 1 | 1 | 1 |
| **0.7** | 0.940 | 0.990 | 1 | 1 | 1 | 1 | 1 | 1 |
| **0.9** | 0.980 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |




**(a) `Twitter`: Consecutive embeddings change in Twitter during 05/12/2014–07/31/2014.**

**(b) `Enron`: Consecutive embeddings change in weekdays during 01/01/2001–5/01/2002.**

**(c) Runtime (in sec) for Multi-LENS vs. node2vec.**

**Figure 5: (a)-(b) Major event detection in real world datasets; (c) Runtime reported on ER graphs with $d_{\text{avg}} = 10$. Multi-LENS scales similarly to node2vec with less memory requirement while node2vec runs out of memory on the graph with $10^7$ nodes.**

subset of nodes. For real-world graphs, we also observe that Multi-LENS successfully detects anomalous patterns when the injection is relatively dense, even when the background graphs have complex structural patterns. This demonstrates that Multi-LENS can effectively detect global changes in graph structures.

*4.4.2 Graph-based Event Detection.* We further apply Multi-LENS to real-world graphs to detect events that appear unusual or anomalous with respect to the global temporal behavior of the complex network. The datasets we used are the `Twitter`[3] and `Enron`[4] graphs. `Twitter` has totally 308 499 nodes and 2 601 834 edges lasting from 05/12/2014 to 07/31/2014, and `Enron` has totally 80848 nodes and 2 233 042 edges lasting from 01/01/2001 to 05/01/2002. Similar to the synthetic scenario, we split the temporal graph into consecutive daily subgraphs and adopt the summary learned from $G_{t-1}$ to get node embeddings of $G_t$. Intuitively, large distances between node embeddings of consecutive daily graphs indicate abrupt changes of graph structures, which may signal events.

Fig. 5a shows the change of Frobenius norm between keyword / hashtag embeddings in consecutive instances of the daily `Twitter` co-mentioning activity. The two marked days are $3\sigma$ (stdev) units away from the median value [17], which correspond to serious events: (1) the Gaza-Israel conflict and (2) Ebola Virus Outbreak. Compared with other events in the same time period, the detected ones are the most impactful in terms of the number of people affected, and the attraction they drew as they are related to terrorism or domestic security. Similarly for `Enron`, we detect several events based on the change of employee embeddings in the Enron corpus from the daily message-exchange behavior. We highlight these events, which correspond to notable ones in the company's history, in Fig. 5b and provide detailed information in Appendix B.5.

## 4.5 Scalability of Multi-LENS

Finally, Q4 concerns the scalability of our approach. To that end, we generate Erdős-Rényi graphs with average degree $d_{\text{avg}} = 10$, while varying the number of nodes from $10^2$ to $10^7$. For reference, we compare it against one of the fastest and most scalable baselines, node2vec. As shown in Fig. 5c, node2vec runs out of memory on the graph with $10^7$ nodes, whereas Multi-LENS scales almost as well as node2vec and to bigger graphs, while also using less space.

# 5 RELATED WORK

We qualitatively compare Multi-LENS to summarization and embedding methods in Table 6.

*Node embeddings.* Node embedding or representation learning has been an active area which aims to preserve a notion of similarity over the graph in node representations [10, 28]. For instance, [6, 8, 11, 23, 31] define node similarity in terms of proximity (based on the adjacency or positive pointwise mutual information matrix) using random walks (RW) or deep neural networks (DNN). More relevant to Multi-LENS are approaches capturing similar node behavioral patterns (roles) or structural similarity [1, 13, 27, 28]. For instance, struc2vec and xNetMF [13, 25] define similarity based on node degrees, while DeepGL [28] learns deep inductive relational functions applied to graph invariants such as degree and triangle counts. [18, 24] investigate theoretical connection between matrix factorization and the skip-gram architecture. To handle heterogeneous graphs, metapath2vec [8] captures semantic and structural information by performing RW on predefined metapaths. There are also works based on specific characteristics in heterogeneous graphs. For example, AspEm represents underlying semantic facets as multiple "aspects" and selects a subset to embed based on datasetwide statistics. Unlike above methods that generate dense embeddings of fixed dimensionality, Multi-LENS derives compact and multi-level latent summaries that can be used to generate node embeddings without specifying extra input. The use of relational operators is also related to recent neural-network based methods. For example, the *mean* operator is related to the mean aggregator of GraphSAGE [12] and the propagation rule of GCN [16]. But unlike these and other neural-network based methods that propagate information and learn embeddings based on features from the local neighborhood, Multi-LENS learns latent subspace vectors of node contexts as the summary.

*Summarization.* We give an overview of graph summarization methods, and refer the interested reader to a comprehensive survey [19]. Most summarization works fall into 3 categories: (1) aggregation-based which group nodes [22] or edges [20]) into super-nodes/edges

[3] http://odds.cs.stonybrook.edu/twittersecurity-dataset/
[4] http://odds.cs.stonybrook.edu/enroninc-dataset/

**Table 6: Qualitative comparison of Multi-LENS to existing summarization and embedding methods. Does the method: handle heterogeneous graphs; yield an output that is size-independent, but node-specific, and representations that are independent of node proximity; support inductive learning and scale well (i.e., it is subquatratic on the network size)?**

| | Input | Representations / Output | | | Method | |
|---|---|---|---|---|---|---|
| | **Hetero-geneity** | **Size indep.** | **Node specific** | **Proxim. indep.** | **Scalable** | **Induc.** |
| Aggregation [2] | ✓ | ✗ | ✗ | ✗ | ✓ | ✗ |
| Cosum [33] | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ |
| AspEm [30] | ✓ | ✗ | ✓ | ✗ | ✓ | ✗ |
| metapath2vec [8] | ✓ | ✗ | ✓ | ✗ | ✓ | ✗ |
| n2vec [11], LINE [31] | ✗ | ✗ | ✓ | ✗ | ✓ | ✗ |
| struc2vec [25] | ✗ | ✗ | ✓ | ✓ | ✗ | ✗ |
| DNGR [6] | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ |
| GraphSAGE [12] | ✓ | ✗ | ✓ | ✓ | ✓ | ✓ |
| Multi-LENS | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

based on application-oriented criteria or existing clustering algorithms; (2) abstraction-based which remove less informative nodes or edges; and (3) compression-based [29] which aim to minimize the number of bits required to store the input graph. Summarization methods have a variety of goals, including query efficiency, pattern understanding, storage reduction, interactive visualization, and domain-specific feature selection. The most relevant work is CoSum [33], which tackles entity resolution by aggregating nodes into supernodes based on their labels and structural similarity. Unlike these methods, Multi-LENS applies to *any type* of graphs and generates summaries independent of nodes/edges in a *latent* graph space. Moreover, it is general and not tailored to specific ML tasks.

## 6 CONCLUSION

This work introduced the problem of *latent network summarization* and described a general computational framework, Multi-LENS to learn such space-efficient latent node summaries of the graph that are completely independent of the size of the network. The output (size) of latent network summarization depends only on the complexity and heterogeneity of the network, and captures its key structural behavior. Compared to embedding methods, the latent summaries generated by our proposed method require 80-2152× **less** output storage space for graphs with millions of edges, while achieving significant improvement in AUC and F1 score for the link prediction task. Overall, the experiments demonstrate the effectiveness of Multi-LENS for link prediction, anomaly and event detection, as well as its scalability and space efficiency.

## ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under Grant No. IIS 1845491, Army Young Investigator Award No. W911NF1810397, an Adobe Digital Experience research faculty award, and an Amazon faculty award. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or other funding parties. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

## REFERENCES

[1] Nesreen K. Ahmed, Ryan A. Rossi, Rong Zhou, John Boaz Lee, Xiangnan Kong, Theodore L. Willke, and Hoda Eldardiry. 2018. Learning Role-based Graph Embeddings. In *IJCAI StarAI*.
[2] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *JSTAT* 2008, 10 (2008), P10008.
[3] Aleksandar Bojchevski and Stephan Günnemann. 2018. Deep Gaussian Embedding of Graphs: Unsupervised Inductive Learning via Ranking. (2018).
[4] Matthew Brand. 2006. Fast low-rank modifications of the thin singular value decomposition. *Linear algebra and its applications* 415, 1 (2006), 20–30.
[5] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. Grarep: Learning graph representations with global structural information. In *CIKM*. 891–900.
[6] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2016. Deep Neural Networks for Learning Graph Representations.. In *AAAI*. 1145–1152.
[7] Reuven Cohen and Shlomo Havlin. 2003. Scale-free networks are ultrasmall. *Physical review letters* 90, 5 (2003), 058701.
[8] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *KDD*. 135–144.
[9] Alan Frieze, Ravi Kannan, and Santosh Vempala. 2004. Fast Monte-Carlo algorithms for finding low-rank approximations. *JACM* 51, 6 (2004), 1025–1041.
[10] Palash Goyal and Emilio Ferrara. 2018. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems* 151 (2018), 78–94.
[11] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *KDD*. 855–864.
[12] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NIPS*. 1024–1034.
[13] Mark Heimann, Haoming Shen, Tara Safavi, and Danai Koutra. 2018. REGAL: Representation Learning-based Graph Alignment. In *CIKM*.
[14] Keith Henderson, Brian Gallagher, Tina Eliassi-Rad, Hanghang Tong, Sugato Basu, Leman Akoglu, Danai Koutra, Christos Faloutsos, and Lei Li. 2012. RolX: Structural Role Extraction & Mining in Large Graphs. In *SIGKDD*. 1231–1239.
[15] Di Jin and Danai Koutra. 2017. Exploratory analysis of graph data by leveraging domain knowledge. In *ICDM*. 187–196.
[16] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.
[17] Danai Koutra, Joshua T Vogelstein, and Christos Faloutsos. 2013. Deltacon: A principled massive-graph similarity function. In *SDM*. SIAM, 162–170.
[18] Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *NIPS*. 2177–2185.
[19] Yike Liu, Tara Safavi, Abhilash Dighe, and Danai Koutra. 2018. Graph Summarization Methods and Applications: A Survey. *CSUR* 51, 3 (2018), 62.
[20] Antonio Maccioni and Daniel J Abadi. 2016. Scalable pattern matching over compressed graphs via dedensification. In *SIGKDD*. ACM, 1755–1764.
[21] Benjamin A Miller, Michelle S Beard, Patrick J Wolfe, and Nadya T Bliss. 2015. A spectral framework for anomalous subgraph detection. *IEEE TSP* 63, 16 (2015), 4191–4206.
[22] Saket Navlakha, Rajeev Rastogi, and Nisheeth Shrivastava. 2008. Graph summarization with bounded error. In *SIGMOD*. 419–432.
[23] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *KDD*. 701–710.
[24] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. 2018. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *WSDM*. 459–467.
[25] Leonardo F.R. Ribeiro, Pedro H.P. Saverese, and Daniel R. Figueiredo. 2017. Struc2Vec: Learning Node Representations from Structural Identity. In *SIGKDD*.
[26] Ryan A. Rossi and Nesreen K. Ahmed. 2015. The Network Data Repository with Interactive Graph Analytics and Visualization. In *AAAI*. http://networkrepository.com
[27] Ryan A. Rossi and Nesreen K. Ahmed. 2015. Role Discovery in Networks. *TKDE* 27, 4 (April 2015), 1112–1131.
[28] Ryan A. Rossi, Rong Zhou, and Nesreen K. Ahmed. 2018. Deep Inductive Network Representation Learning. In *WWW BigNet*.
[29] Neil Shah, Danai Koutra, Tianmin Zou, Brian Gallagher, and Christos Faloutsos. 2015. Timecrunch: Interpretable dynamic graph summarization. In *KDD*. 1055–1064.
[30] Yu Shi, Huan Gui, Qi Zhu, Lance Kaplan, and Jiawei Han. 2018. AspEm: Embedding Learning by Aspects in Heterogeneous Information Networks. In *SDM*. SIAM, 144–152.
[31] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale Information Network Embedding. In *WWW*. 1067–1077.
[32] Lei Tang and Huan Liu. 2011. Leveraging social media networks for classification. *Data Mining and Knowledge Discovery* 23, 3 (2011), 447–478.
[33] Linhong Zhu, Majid Ghasemi-Gol, Pedro Szekely, Aram Galstyan, and Craig A Knoblock. 2016. Unsupervised entity resolution on multi-type graphs. In *ISWC*. 649–667.

# Supplementary Material on Reproducibility

## A COMPLEXITY ANALYSIS PROOF

### A.1 Computational complexity

Proof. The computational complexity of Multi-LENS includes deriving (a) distribution-based matrix representation $\mathbf{H}$ and (b) its low-rank approximation.

The computational unit of Multi-LENS is the relational operation performed over the egonet of a specific node. Searching the neighbors for all node $i \in V$ has complexity $O(N+M)$ through BFS. The complexity of step (a) is linear to $|\mathcal{F}_r|$, as indicated in § 3.3, this number is $|\mathcal{B}||\Phi|^\ell \cdot 2|\mathcal{T}_V||\mathcal{T}_E|c$.

Based on the sparsity of $\mathbf{H}$, Multi-LENS performs SVD efficiently through fast Monte-Carlo Algorithm by extracting the most significant $K$ singular values [9] with computational complexity $O(K^2N)$. Therefore step (b) can be accomplished in $O((K^{(\ell)})^2N)$ by extracting the most significant $K^{(\ell)}$ singular values at level $\ell$. Furthermore, deriving all $K$ singular values has $O(K^2N)$ complexity as $\sum_{\ell=1}^{L}(K^{(\ell)})^2 \leq (\sum_{\ell=1}^{L} K^{(\ell)})^2 = K^2$. The overall computational complexity is thus $O(N|\mathcal{F}_r||\mathcal{T}_E||\mathcal{T}_V|c + K^2N + M)$. Note that both $|\Phi|$ and $L$ are small constants in our proposed method (*e.g.*, $|\Phi| = 7$ and $L \leq 2$). Multi-LENS scales linearly with the number of nodes and edges $(N + M)$ in $G$. □

### A.2 Space Complexity

Proof. In the runtime at level $\ell$, Multi-LENS stores $\mathbf{F}^{(\ell-1)}$ to dervie $\mathbf{F}^{(\ell)}$ and $\mathbf{H}^{(\ell)}$, which take $O(N|\mathcal{F}_r|)$ and $O(c|\mathcal{F}_r||\mathcal{T}_V||\mathcal{T}_E|N)$ space, respectively. SVD can be performed with $p \ll N$ sampled rows. For the output, storing the set of ordered compositions of relational functions in the summary requires space complexity $O(|\mathcal{F}_r|)$. For the set of matrices $\mathcal{S}$, we store $\mathbf{S}^{(\ell)}$ across all $L$ levels. As shown in the time complexity analysis, the number of binned features (columns) in $\mathbf{H}$ over all levels is $2|\mathcal{F}_r||\mathcal{T}_V||\mathcal{T}_E|c$, which includes incorporating $|\mathcal{T}_V|$ object types with both in-/out- directionality and all edge types. The size of the output summarization matrices is thus $O(K|\mathcal{F}_r||\mathcal{T}_V||\mathcal{T}_E|c)$, which is related to the graph heterogeneity and structure and independent of the network size $N, M$. □

## B EXPERIMENTAL DETAILS

### B.1 Configuration

We run all experiments on Mac OS platform with 2.5GHz Intel Core i7 and 16GB memory. We configure the baselines as follows: we use 2nd-LINE to incorporate 2-order proximity in the graph; we run node2vec with grid searching over $p, q \in \{0.25, 0.50, 1, 2, 4\}$ as mentioned in [11] and report the best. For GraRep, we set $k = 2$ to incorporate 2-step relational information. For DNGR, we follow the paper to set the random surfing probability $\alpha = 0.98$ and use a 3-layer neural network model where the hidden layer has 1024 nodes. For Metapath2vec, we retain the same settings (number of walks = 1000, walk length = 100) to generate walk paths and adopt a similar the meta-path "Type 1-Type 2-Type 1" as the "A-P-A" schema as suggested in the paper. For Multi-LENS, although arbitrary relational functions can be used, we use order-1 $f_b = \sum$ as the base graph function for simplicity in our experiments. To begin with, we derive in-/out- and total degrees to construct the $N \times 3$ base feature matrix $\mathbf{F}^{(0)}$ denoted as $\left[f_b\langle \mathbf{b}_1, \mathcal{N}\rangle, f_b\langle \mathbf{b}_2, \mathcal{N}\rangle, f_b\langle \mathbf{b}_3, \mathcal{N}\rangle\right]$ where $\mathbf{b}_1 = \mathbf{A}_{i:}$, $\mathbf{b}_2 = \mathbf{A}_{:i}$, and $\mathbf{b}_3 = (\mathbf{A} + \mathbf{A}^T)_{i:}$, for $i \in V$. We set $L = 2$ to construct order-2 relational functions to equivalently incorporate 2-order proximity as LINE does, but we do not limit other methods to incorporate higher order proximity. All other settings are kept default. In table 7, we list all relational operators used in the experiment.

**Table 7: Relational operators used in the experiment**

| $\phi$ | Definition | $\phi$ | Definition |
|---|---|---|---|
| *max/min* | $\max/\min_{i\in\mathcal{S}} \mathbf{x}_i$ | *variance* | $\frac{1}{\lvert\mathcal{S}\rvert}\sum_{i\in\mathcal{S}} \mathbf{x}_i^2 - (\frac{1}{\lvert\mathcal{S}\rvert}\sum_{i\in\mathcal{S}} \mathbf{x}_i)^2$ |
| *sum* | $\sum_{i\in\mathcal{S}} \mathbf{x}_i$ | *l1-distance* | $\sum_{j\in\mathcal{S}} \lvert x_i - x_j\rvert$ |
| *mean* | $\frac{1}{\lvert\mathcal{S}\rvert}\sum_{i\in\mathcal{S}} \mathbf{x}_i$ | *l2-distance* | $\sum_{j\in\mathcal{S}} (x_i - x_j)^2$ |

### B.2 Heterogeneous Context

We justify the effectiveness to derive heterogeneous context of feature matrix $\mathbf{F}$. We perform the link prediction task on yahoo-msg and dbpedia datasets w./w.o. using $\mathbf{H}$ following the setup indicated in §B.1. The result is as follows. We observe the significant improvement in performance when deriving the histogram heterogeneous context, which empirically supports our claim in §3.3.

**Table 8: Link prediction performance w./w.o. H**

| Data | Metric | w.o. $\mathbf{H}$ | w. $\mathbf{H}$ |
|---|---|---|---|
| yahoo-msg | **AUC** | 0.7919 | **0.8443** |
| | **ACC** | 0.7122 | **0.7587** |
| | **F1 macro** | 0.7111 | **0.7577** |
| dbpedia | **AUC** | 0.9369 | **0.9820** |
| | **ACC** | 0.9023 | **0.9197** |
| | **F1 macro** | 0.9020 | **0.9197** |

### B.3 Choice of initial vectors

In this subsection we justify the impact of initial vectors $\mathcal{B}$ to the performance on two datasets, yahoo-msg and dbpedia. We follow the configuration in §B.1 and run Multi-LENS with different initial vectors indicated in Table 9. We observe that there is no significant difference when using different $\mathcal{B}$. This is as expected as the three vectors are linearly correlated and shows the power of relational operators to derive complex higher-order features.

**Table 9: Link prediction performance with different $\mathcal{B}$**

| Data | Metric | $\mathcal{B} = \{\mathbf{b}_1\}$ | $\mathcal{B} = \{\mathbf{b}_1, \mathbf{b}_2\}$ | $\mathcal{B} = \{\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3\}$ |
|---|---|---|---|---|
| yahoo-msg | **AUC** | 0.8439 | **0.8443** | **0.8443** |
| | **ACC** | 0.7558 | 0.7559 | **0.7587** |
| | **F1 macro** | 0.7545 | 0.7547 | **0.7577** |
| dbpedia | **AUC** | **0.9821** | **0.9821** | 0.9820 |
| | **ACC** | 0.9164 | 0.9168 | **0.9197** |
| | **F1 macro** | 0.9164 | 0.9168 | **0.9197** |

### B.4 Choice of relational operators

In this subsection we justify the impact of relational operators $\mathbf{\Phi}$ to the performance on yahhoo-msg dataset. To explore the effect of each operator, we set $\mathcal{F}_b = \mathbf{\Phi}$, and set $\mathcal{B} = \{\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3\}$ with $L = 1$. To illustrate the effectiveness of the operators only, we do not derive histogram-based node contexts. We observe that *sum* and *mean* have potentially higher impacts to the performance than *max* and *min*. Using the combination of *l1*- and *12*-distance produces the worst performance.

**Table 10: Link prediction performance with different $\Phi$**

| $\Phi$ | AUC | ACC | F1 macro |
|---|---|---|---|
| {*max*} | 0.7317 | 0.6582 | 0.6526 |
| {*max, min*} | 0.7727 | 0.6888 | 0.6873 |
| {*sum, mean, var*} | 0.8274 | 0.7584 | 0.7584 |
| {*l1, l2*} | 0.6906 | 0.6311 | 0.6307 |
| {*max, min, sum*} | 0.8283 | 0.7445 | 0.7436 |
| {*max, min, sum, mean*} | **0.8336** | **0.7611** | **0.7609** |

## B.5 Detailed Event detection

Events detected in Fig. 5b: (1) The quarterly conference call where Jeffrey Skilling, Enron's CEO, reports "outstanding" status of the company; (2) The infamous quarterly conference call; (3) FERC institutes price caps across the western United States; (4) The California energy crisis ends; (5) Skilling announces desire to resign to Kenneth Lay, founder of Enron; (6) Baxter, former Enron vice chairman, commits suicide, and (7) Enron executives Andrew Fastow and Michael Kopper invoke the Fifth Amendment before Congress.

# C HETEROGENEITY CONTEXT IN DETAIL

## C.1 Histogram representation

Specific feature values in $\mathbf{F}^{(\ell)}$ derived by $\ell$-order composed relational functions could be prodigious due to the power-law nature of real-world graphs (*e.g.*, total degree), which leads to underrepresentation of other features in the summary. Among various techniques to handle skewness, we describe the $N \times 1$ feature vector by the distribution of its unique values, on which we apply logarithmic binning [15]. The justification of our decision in Appendix B.2 shows that binning is necessary to improve performance and incorporate multiple operators in Multi-LENS.

For feature vector $\mathbf{x}$, a set of nodes in $\mathcal{N}$ and $c$ bins, logarithmic binning returns a vector of length $c$:

$$\Psi(\mathbf{x}, \mathcal{N}, c) = [C(0), C(1), \ldots, C(\log_a(c))] \tag{8}$$

where $C(v)$ counts the occurrence of value $v$: $C(v) = \sum_{i \in \mathcal{N}} \delta(v, \mathbf{x}_i)$. In $C(v)$, $\delta$ is the Kronecker delta (a.k.a indicator) function, $a$ is the logarithm base, and $c = \max\{\max(\mathbf{x}), c\}$. We set $c$ to the value exceeding the maximum feature value ($\max(\mathbf{x})$) regardless of object types to make sure that the output bin counts remain the same across all features. We can explicitly fill in 0s in Eq. (8) in the case of $c > \max(\mathbf{x})$. Similar to Eq. (1), we use $\Psi\langle \mathbf{x}, \mathcal{S}', c\rangle$ to denote the process of applying $\Psi$ function over all nodes in $V$ (rows of $\mathbf{F}$) to get the $N \times c$ log-based histogram matrix. Further, we denote the process of applying $\Psi$ on all feature vectors (columns of $\mathbf{F}$) as $\mathbf{H} = \Psi\langle \mathbf{F}, \mathcal{N}, c\rangle$. We use $\mathbf{H}$ to denote the resultant histogram-based feature matrix. In the next subsection, we will explain how to apply $\Psi$ on different related subsets $\mathcal{R} \subseteq \mathcal{N}$ to incorporate heterogeneity in the summary.

## C.2 Heterogeneous contexts

*C.2.1 Object types.* In heterogeneous graphs, the interaction patterns between a node and its typed neighbors reveal important behavioral information. Intuitively, similar entities have similar interaction patterns with every single type of neighbor. For example, in the author-paper-venue networks, authors submitting papers to the same track at the same conference have higher similarity than authors submitting to different tracks at the same conference. To describe how a specific node $i$ interacts with objects of type $t$, Multi-LENS collects typed $t$ neighbors by setting $\mathcal{R} = \mathcal{N}_i^t$ and computes the "localized" histogram of a specific feature vector $\mathbf{x}$ through $\Psi(\mathbf{x}, \mathcal{N}_i^t, c)$. Repeating this process for nodes $i \in V$ forms an $N \times c$ distribution matrix $\Psi\langle \mathbf{x}, \mathcal{N}^t, c\rangle$.

Multi-LENS enumerates all types of neighbors within $\mathcal{N}$ to incorporate complete interaction patterns for each node in the graph. This process can be seen as introducing one more dimension, the object types, to $\mathbf{H}$ to form a tensor, as shown in Fig. 6. We flatten the tensor through horizontal concatenation and denote it as $\mathbf{H}$:

$$\mathbf{H}_o = \left[\Psi\langle \mathbf{F}, \mathcal{N}^{T_1}, c\rangle, \Psi\langle \mathbf{F}, \mathcal{N}^{T_2}, c\rangle, \ldots, \Psi\langle \mathbf{F}, \mathcal{N}^{t}, c\rangle\right] \tag{9}$$

where $t = \{1, \ldots, |\mathcal{T}_V|\}$

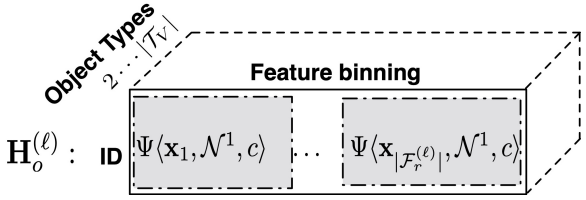



**Figure 6: At level $\ell$, enumerating object types in $\mathcal{N}$ introduces one more dimension to the feature matrix and leads to a tensor. Note that $\Psi\langle \mathbf{x}, \mathcal{N}, c\rangle$ is an $N \times c$ matrix (shaded area). Layer $t$ of the tensor is also denoted as $\Psi\langle \mathbf{F}, \mathcal{N}^t, c\rangle$ for brevity.**

*C.2.2 Edge directionality.* So far we assume the input graph is undirected by focusing on nodes in $\mathcal{N}$ and search for neighbors in the 1-hop neighborhood regardless of edge directions. Multi-LENS handles the directed input graphs by differentiating nodes from the out-neighborhood and in-neighborhood. The process is almost identical to the undirected case, but instead of using $\mathcal{N}$ in Equation (9), we consider its two disjoint subsets $\mathcal{N}^+$ and $\mathcal{N}^-$ with incoming and outgoing edges, respectively. The resultant histogram-based feature matrices are denoted as $\mathbf{H}_o^+$ and $\mathbf{H}_o^-$, respectively. Again, we horizontally concatenate them to get the feature matrix incorporating edge directionality $\mathbf{H}_d$ as $\mathbf{H}_d = \left[\mathbf{H}_o^+, \mathbf{H}_o^-\right]$.

*C.2.3 Edge types.* Edge types in heterogeneous graphs play an important role in determining graph semantics and structure. The same connection between a pair of nodes with different edge types could convey entirely different meanings (e.g., an edge could indicate "retweet" or "reply" in a Twitter-communication network). This is especially important when the input is a multi-layer graph model. To handle multiple edge types, Multi-LENS constructs subgraphs $g(V, E_\tau)$ restricted to a specific edge type $\tau \in \mathcal{T}_E$. For each subgraph, Multi-LENS repeats the process to obtain the corresponding feature matrix $\mathbf{H}_d$ per edge type that incorporates both node types and edge directionality. We denote the feature matrix $\mathbf{H}_d$ with respect to edge type $\tau$ as $\mathbf{H}_d^\tau$. Thus, by concatenating them horizontally we obtain the final histogram-based context representation denoted as:

$$\mathbf{H}_e = \left[\mathbf{H}_d^1, \mathbf{H}_d^2, \ldots, \mathbf{H}_d^\tau\right] \tag{10}$$

where $\tau = \{1, \ldots, |\mathcal{T}_E|\}$. Therefore, $\mathbf{H}_e$ captures all three aspects of heterogeneity of graph $G$ with size $N \times 2|\mathcal{T}_V||\mathcal{T}_E|c \cdot |\mathcal{F}_r|$. We use $\mathbf{H}$ to denote this representation for brevity.