# Variational Graph Auto-Encoders for Heterogeneous Information Network

**Abhishek Dalvi**[1] **Ayan Acharya**[2] **Jing Gao**[3] **Vasant Honavar**[1]
[1]The Pennsylvania State University, University Park, PA.
[2]LinkedIn Inc., Sunnyvale, CA.
[3]Purdue University, West Lafayette, IN.
adalvi@psu.edu ayacharya@linkedin.com jinggao@purdue.edu vuh14@psu.edu

## Abstract

Heterogeneous Information Network (HIN), where nodes and their attributes denote real-world entities and links encode relationships between entities, are ubiquitous in many applications. The presence of multiple types of nodes and links pose significant challenges to the state-of-the-art methods for learning node embeddings from heterogeneous graphs. To address these challenges, we consider three variants of graph variational autoencoder models for heterogeneous networks that avoid the computationally expensive sampling of meta-paths. The proposed methods also maintain uncertainty estimates of node embeddings that help improve generalization performance. We report the results of experiments on link prediction using three different real-world heterogeneous network benchmark data sets that show that the proposed methods significantly outperform state-of-the-art baselines.

## 1 Introduction

Graph-structured data, which naturally encodes the relations between individuals, are ubiquitous in many real world applications, including e-commerce [1–4], social networks [5–7] and molecular structures [8–10], among others. Such data present a range of machine learning problems, including node classification [11] and link prediction [12]. Graph embedding techniques [13] aim to construct low-dimensional information-preserving representations of graph-structured data. The past several decades have seen an explosion of interest in algorithms for mining graph-structured data [14–16]. Of particular interest are Graph Neural Networks (GNNs) [17, 18], a type of deep neural networks [19] that offer state-of-the-art performance on representation learning from graph-structured data. Such techniques have found applications across many domains including recommender systems [20], bioinformatics [21], drug discovery [22], social network analysis [23] and material science [24], among others.

The earliest GNN methods [25–29] focused on node and link classification problems in homogeneous networks, i.e., networks with only one type of nodes and links. However, many real-world applications present heterogeneous graphs, i.e., graphs with multiple types of nodes and links. For example, atoms in a chemical structure can be of different types, and atoms of different types may be linked by different types of chemical bonds (heterogeneous edges). Similarly, a bibliographic graph consisting of multiple types of nodes, namely, authors, papers, topics, and venues contain not only the edges between nodes of the same type, e.g., papers (paper citation) but also edges that link nodes of different types e.g., authors and papers (authorship) or papers and venues (publication). Such applications call for effective GNN methods that work with heterogeneous graphs.

Existing approaches for learning embeddings of heterogeneous graphs rely on sampling meta-paths [30], i.e., sequences of node types encoding relations between node types; for learning node embeddings—different meta-paths express different semantic meanings. Most existing work, e.g.,

NeurIPS 2022 New Frontiers in Graph Learning Workshop (NeurIPS GLFrontiers 2022).

[30], assumes that domain experts specify the meta-paths. While recent work, e.g., Wan et al. [31], offers approaches to discovering meta-paths, their flexibility comes at a high computational cost. Extracting meta-paths from heterogeneous graphs often involves dealing with with the attendant loss of information. Meta-paths fail to account for possible differences in the importance of heterogeneous links based on the types of nodes involved and their influence on nodes for which embedding is being generated. Although some meta-path-free methods have been developed recently, e.g., Hussein et al. [32], they generally limit that the edges to be between nodes of the same type.

Furthermore, heterogeneous graphs are often constructed from real-world data, e.g., collections of articles in the case of citation networks. Both the data from data and the algorithms used to extract heterogeneous graphs can introduce noise, and methods used to learn node embeddings from such graphs should be sufficiently robust to cope with noise.

Against this background, this paper explores three node embedding techniques for heterogeneous graphs. The key contributions of this paper are as follows:

1. We precisely define the problem of learning node embeddings from heterogeneous graphs and highlight some of the deficiencies of the existing methods.
2. We introduce three novel heterogeneous graph embedding methods that directly utilize the information encoded by a heterogeneous graph. These methods utilize a node and link type-aware transformations to aggregate information from local neighborhoods around nodes, taking into account, the heterogeneous nodes and links in the graphs. We maintain uncertainty associated with the learned embeddings of nodes to improve the noise-tolerance of the methods.
3. We evaluate the performance of the proposed models on the link-prediction task on three heterogeneous graphs derived from three real-world data sets (IMDB, DBLP, and AMiner). We present the results of experiments that demonstrate the superiority of the proposed methods over the state-of-the-art baselines. Additionally, we visually inspect the learned low-dimensional representations and analyze the effects of the models' settings on the predicted neighborhood of the graph.

The rest of the paper is organized as follows. Section 2 summarizes related work. Section 3 provides a precise description of the problem of learning node embeddings from heterogeneous graphs. Section 4 elaborates on our assumptions and describes the learning algorithms . Section 5 compares the the proposed methods with that of state-of-the-art baselines using several real-world data sets. Section 6 concludes the paper with a summary and an outline of some directions for further research.

## 2 Related Work

### 2.1 Heterogeneous Graph Neural Networks

Most existing work on heterogeneous graph neural networks (H-GNNs) tend to utilize the meta-paths [30] to adapt current homogeneous graph embedding methods, such as Kipf and Welling [25], Perozzi et al. [33], Grover and Leskovec [34], Wang et al. [35], to work with heterogeneous graphs. Metapath2vec [36] designs meta-paths to guide the random walks in a heterogeneous graph and then follows the skip-gram model to learn the latent representations of the vertices. Inspired by Metapath2vec [36], Shi et al. [37] propose HERec to fuse different representations learned in the view of different meta-path schemes. Both Metapath2vec and HERec are optimized for link prediction.

To optimize the embeddings for a specific task, Chen and Sun [38] unify task-guided and linkage-guided objectives to learn the heterogeneous graph embeddings. Wang et al. [39] introduce two-level hierarchical attention to GNN, in which node-level attention captures the relations between neighboring nodes encoded by the meta paths, and semantic-level attention aggregates multiple meta-paths for each node in the graph.

Unlike previous meta path-based models that perform selective aggregation of information from node neighborhoods, HetSANN [40] directly utilizes a HIN for more informative representations without using meta-paths to improve both efficiency and performance of the resulting methods relative to state-of-the-art baselines.

### 2.2 Variational Methods

Inspired by Kingma and Welling [41], VGAE [26] takes a Bayesian approach to link prediction problem by extending GCNs using a Variational Auto-Encoder [41]. Unlike previous works [33, 34, 42] on unsupervised learning with structured graph data, VGAE treats the latent representations of the nodes as random variables. The posterior distribution associated with such random variables is approximated using factorized variational distribution, whose mean and variance are modelled by multi-layered GNNs. BAM [43] is a stochastic version of the basic the attention mechanism wherein the non-negative elements of the attention matrix are used to represent their uncertainty.

## 3 Problem Statement

We define a heterogeneous graph $\mathcal{G}^{\mathcal{H}} = (\mathcal{V}, \mathcal{E}, \mathcal{R})$, where $\mathcal{V}$ represents the set of nodes, $\mathcal{E}$ represents the set of edges and $\mathcal{R}$ denotes the types of relations that exist in such a graph. We consider a graph with $N = |\mathcal{V}|$ number of nodes.

Each node $v \in \mathcal{V}$ is assumed to have a type $\phi(v) = t \in \mathcal{T}$ where $\phi(\cdot) : \mathcal{V} \rightarrow \mathcal{T}$ is the mapping from $\mathcal{V}$ to $\mathcal{T}$. An edge $e \in \mathcal{E}$ in such a graph is defined as a triplet $e = (i, j, r)$, where $r \in \mathcal{R}$ is the type of relationship between nodes $i \in \mathcal{V}$ and $j \in \mathcal{V}$ and is dictated by the types of the nodes linked by the edge, i.e., $< \phi(i), \phi(j) >$. Each node in the graph is associated with set of node features. For nodes of type $t \in \mathcal{T}$, the set of features is compactly represented by the feature matrix $\mathbf{X}_t \in \mathbb{R}^{N_t \times F_t}$, where $N_t$ is the number of nodes of type $t$ and $F_t$ is the number of the features for nodes of type $t$.

Table 1: Table of Main Symbols

| Symbols | Definition |
|---|---|
| $\mathcal{G}^{\mathcal{H}}$ | Heterogeneous Graph |
| $\mathcal{V}$ | Node/Vertices Set |
| $\mathcal{E}$ | Edge Set |
| $\mathcal{R}$ | Relation Set |
| $\mathcal{T}$ | Node Type Set |
| $\mathbf{A}$ | Adjacency Matrix |
| $\hat{\mathbf{A}}$ | Normalized Adjacency Matrix |
| $\mathbf{D}$ | Degree Matrix |
| $\phi()$ | Mapping function from $\mathcal{V}$ to $\mathcal{T}$ |
| $\mathbf{X}_t$ | Feature Matrix of nodes of type $t$ |
| $\mathcal{X}$ | List of all types of Feature Matrices |
| $l$ | Layer number |
| $\mathbf{H}^{(\ell)}_{\phi(i)}$ | Hidden representation matrix of nodes of type $\phi(i)$ at layer $l$ |
| $\mathbf{z}_i$ | Embedding of Node $i$ |
| $\mathbf{h}^{(\ell)}_i$ | Hidden representation of node $i$ at layer $l$ |
| $\mathbf{W}^{(\ell)}_{\phi(i),\phi(j)}$ | Weight matrix for transforming Nodes of type $\phi(j)$ to $\phi(i)$ at layer $l$ |
| $\mathbf{a}^{(\ell)}_r$ | Attention weights for relation $r$ |
| $\theta^{(\ell)}_{ij}$ | Attention Coefficient between node $i$ and $j$ |
| $f()$ | Activation function |
| $\zeta()$ | Matrix Stacking/Concatenating Operator |

Further, $\mathbf{A}$ denotes the adjacency matrix representation of $\mathcal{G}^{\mathcal{H}}$ with undirected edges, such that $A_{ij} = A_{ji}$ and $A_{ii} = 1$ (self loop). The degree matrix $\mathbf{D}$ is given by:

$$D_{ij} = \begin{cases} \sum_{j=1}^{N} A_{ij} & \text{if } i = j \\ 0 & \text{if otherwise} \end{cases}$$

A normalized adjacency matrix is given by: $\hat{\mathbf{A}} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$.

Our objective is to impute the unobserved edges in the graph. This is achieved by learning the latent representations $\{\mathbf{z}_i\}$ of all the nodes, utilizing the structural information and the attributes of the nodes. The likelihood of an edge between node $i$ and $j$ is predicted using a transformation of the latent representations $\mathbf{z}_i$ and $\mathbf{z}_j$, as explained in the next section. The key notations used in the paper are presented in Table 1.

## 4 Node Embedding Methods for Heterogeneous Graphs

This section introduces three new representation learning algorithms for heterogeneous graphs. We observe that nodes of different types may be described by different sets of features with disparate semantic interpretations. Furthermore, the roles and the importance of nodes may vary according to their type. Hence, we introduce an attention mechanism that can weigh the neighbor's contributions differently. Furthermore, it is helpful to model the uncertainty associated with the latent representation. The models introduced below is designed to achieve all these objectives.

Table 2: Summary of Heterogeneous Variational Graph Auto-encoder Models

| Model | GCN | Het-GCN | HetSANN | Prior on $z$ | Prior on Attn. Coef. |
|---|---|---|---|---|---|
| GVAE | ✓ | × | × | ✓ | N/A |
| H-GVAE | × | ✓ | × | ✓ | N/A |
| HetSANN-GVAE | × | × | ✓ | ✓ | × |
| HetSANN-BAM-GVAE | × | × | ✓ | × | ✓ |

### 4.1 Heterogeneous Graph Variational Auto Encoder

The Heterogeneous Graph Variational Autoencoder (H-GVAE) model incorporates uncertainty in the latent representation of the nodes and uses a node type specific transformation to accommodate the differences in features associated with different types of nodes. More precisely, we seek to compute the posterior distribution of the latent representation of the nodes $p(\mathbf{Z} = \{\mathbf{z}_i\}|\mathcal{X}, \mathbf{A})$, where $\mathbf{z}_i$ is the latent vector corresponding to node $i$. The prior distribution of the latent variables is assumed to be standard normal, i.e., $p(\mathbf{Z}) = \prod_{i=1}^{N} p(\mathbf{z}_i) = \prod_{i=1}^{N} \mathcal{N}(\mathbf{z}_i|0, \mathbf{I})$. Because computing the true posterior is difficult and intractable, we approximate it using a variational distribution of the following form:

$$q(\mathbf{Z}|\mathcal{X}, \mathbf{A}) = \prod_{i=1}^{N} q(\mathbf{z}_i|\mathcal{X}, \mathbf{A})) = \prod_{i=1}^{N} \mathcal{N}(\mathbf{z}_i|\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i). \tag{1}$$

The variational parameters $\boldsymbol{\mu} = \{\boldsymbol{\mu}_i\}$ and $\boldsymbol{\sigma} = \{\boldsymbol{\sigma}_i\}$ are computed by stacking the latent representation matrices for each node type as given below:

$$\boldsymbol{\mu} = \zeta(\text{HGCN}_{\boldsymbol{\mu}}(\mathcal{X}, \mathbf{A})) \text{ and } \log \boldsymbol{\sigma} = \zeta(\text{HGCN}_{\boldsymbol{\sigma}}(\mathcal{X}, \mathbf{A})). \tag{2}$$

Here, $\zeta(\cdot)$ is the stacking operator and HGCN (Heterogeneous Graph Convolution Network) is a special kind of GNN layer that we elaborate next. Using the reparametrization trick [41], one can sample from the variational posterior $\mathbf{z}_i \sim q(\mathbf{z}_i|\mathcal{X}, \mathbf{A})$ as

$$\mathbf{z}_i = \boldsymbol{\mu}_i + \boldsymbol{\sigma}_i \odot \epsilon;\ \epsilon \sim \mathcal{N}(0, 1).$$

While designing the $l^{\text{th}}$ layer of the Heterogeneous Graph Convolution Network (HGCN), we assume that the representation of node $i$ is given by $\mathbf{h}_i^{(\ell)}$, which is updated as follows:

$$\mathbf{h}_i^{(\ell)} = f\big(\sum_{j \in \mathcal{N}_i} \hat{A}_{ij} \mathbf{W}_{\phi(i),\phi(j)}^{(\ell)} \mathbf{h}_j^{(\ell-1)}\big). \tag{3}$$

Here $\mathbf{h}_i^{(0)}$ is set to $\mathbf{X}_i$, the observed attributes of the $i^{\text{th}}$ node. We assume the existence of a weight matrix $\mathbf{W}_{\phi(i),\phi(j)}^{(\ell)}$ for transforming the latent representation of the source node $j$ to match the semantic interpretation of the target node $i$. The learned representations of all nodes of type $\phi(i)$ are compactly preserved in the matrix $\mathbf{H}_{\phi(i)}^{(\ell)}$. Further $\mathcal{H}^{(\ell)}$ contains the collection of all the matrices $\{\mathbf{H}_t^{(\ell)}\}_{t \in \mathcal{T}}$. A layer that employs Eq. (3) is conveniently addressed as Heterogeneous Graph Convolution Network (HGCN).

The transformation $\mathbf{W}_{\phi(i),\phi(j)}^{(\ell)}$ makes HGCN fundamentally different from the standard update equation of other GNN algorithms [29]. Since the different set of nodes may live in disparate semantic space, it is necessary to map the latent representation to a shared semantic space before aggregating information from the neighbors. For example, in an author-paper-venue network, the author's attributes may not have any semantic correlation with those of the venues or the papers. Hence, while aggregating information from the papers a given author has written, it is necessary to transform the latent representation of the papers in the process of learning the latent representation of the author. Note that such transformation is superfluous for a pair of nodes of a similar type and hence is not relevant for learning embedding in a homogeneous graph.

The decoder in HGVAE is an inner product model followed by a Sigmoid activation function. It is derived from the loss function of Mikolov et al. [44] and unsupervised loss of Hamilton et al. [45], which encourages that nodes with high topological similarity must have similar representation:

$$p(\mathbf{A}|\mathbf{Z}) = \prod_{i=1}^{N}\prod_{j=1}^{N} p(A_{ij}|\mathbf{z}_i, \mathbf{z}_j);$$
$$p(A_{ij} = 1|\mathbf{z}_i, \mathbf{z}_j) = \sigma(\mathbf{z}_i^\top \mathbf{z}_j). \tag{4}$$

The weight parameters $\mathbf{W}^{(\ell)}_{\phi(i),\phi(j)}$ are learned by optimizing the Evidence Lower Bound (ELBO) as given below:

$$\mathcal{L} = \mathbb{E}_{q(\mathbf{Z}|\mathcal{X},\mathbf{A})} \log p(\mathbf{A}|\mathbf{Z}) - \mathrm{D}_{\mathrm{KL}}\left[q(\mathbf{Z}|\mathcal{X},\mathbf{A})\|p(\mathbf{Z})\right]. \tag{5}$$

### 4.2 HetSANN-GVAE

The H-GVAE model successfully resolves the problem of utilizing information from HIN for learning node embeddings. Further, motivated by Hong et al. [40] (HetSANN), we introduce a type-aware attention mechanism to implicitly weigh the heterogeneous neighbours of a node in the propagation rule. The HetSANN Graph Variational Autoencoder (HetSANN-GVAE) model is similar to H-GVAE in that it maintains uncertainty in the latent representation, the only difference being that HetSANN layers are used to calculate the parameters of the variational distribution instead of HGCN layers. Thus, the preceding Eq. (2) now assumes the following form:

$$\boldsymbol{\mu} = \zeta(\text{HetSANN}_{\boldsymbol{\mu}}(\mathcal{X}, \mathbf{A})) \text{ and } \log \boldsymbol{\sigma} = \zeta(\text{HetSANN}_{\boldsymbol{\sigma}}(\mathcal{X}, \mathbf{A})).$$

Corresponding to each relation $r$, we compute the attention coefficients $\eta^{(\ell)}_{ij}$, which are parameterized by attention weights $\mathbf{a}^{(\ell)}_r$, and is very similar to the attention mechanism used in NLP applications. The key and query vectors in an attention head are the node features and their neighbourhood features in a relation.

$$\eta^{(\ell)}_{ij} = \text{LeakyReLU}(\mathbf{a}^{(\ell)}_r(\tau(\mathbf{h}^{(\ell-1)}_i, \mathbf{h}^{(\ell-1)}_j)) \tag{6}$$

where

$$\tau(\mathbf{h}^{(\ell-1)}_i, \mathbf{h}^{(\ell-1)}_j) = (\mathbf{W}^{(\ell)}_{\phi(i),\phi(i)}\mathbf{h}^{(\ell-1)}_i || \mathbf{W}^{(\ell)}_{\phi(i),\phi(j)}\mathbf{h}^{(\ell-1)}_j). \tag{7}$$

Here $\eta^{(\ell)}_{ij}$ indicates the importance of node $j$ to node $i$ corresponding to the relation $r$; calculated using the bi-linear scoring function given in Eq. (7). For simplicity, we have assumed in Eq. (6) that there is only one type of edge/relation between nodes of type $\phi(j)$ and $\phi(i)$. However, it is straightforward to generalize our framework to accommodate multiple types of edges between nodes of type $\phi(j)$ and $\phi(i)$. Eq. (6) yields attention coefficients between each pair of nodes in relation $r$, regardless of whether an edge exists between them. We aim to maintain attention coefficients only between nodes and their neighbors in the graph; hence we use the adjacency matrix as a mask to impose structural information about the graph into the attention coefficients. Therefore, $\{\eta^{(\ell)}_{i,j}\}_j$ is computed for all nodes $j \in \mathcal{N}_i$, where $\mathcal{N}_i$ are the neighbour nodes of $i$ and $\eta^{(\ell)}_{ij} = 0$ if $j \notin \mathcal{N}_i$. $\{\eta^{(\ell)}_{ij}\}_j$ values are normalized using softmax function across all neighbors of $i$ yielding the following definition of the attention coefficients:

$$\theta^{(\ell)}_{ij} = \text{softmax}_j(\eta^{(\ell)}_{ij}) = \frac{\exp \eta^{(\ell)}_{ij}}{\sum_{k\in\mathcal{N}_i} \exp \eta^{(\ell)}_{ik}}.$$

The implicit weighting between nodes obtained using type-aware attention is used to aggregate the information from each node's neighbors in the graph in the node's embedding. Unlike Kipf and Welling [25], Wu et al. [27] and the aforementioned H-GVAE, where information from neighbours is aggregated using a pre-computed normalized adjacency matrix, HetSANN-GVAE uses the trainable attention coefficients between nodes aiding the model's performance. Hence, the latent representation are aggregated using $\theta^{(\ell)}_{ij}$ instead of $\hat{\mathbf{A}}$ and (3) changes to:

$$\mathbf{h}^{(\ell)}_i = f(\sum_{j\in\mathcal{N}_i} \theta^{(\ell)}_{ij}\mathbf{W}^{(\ell)}_{\phi(i),\phi(j)}\mathbf{h}^{(\ell-1)}_j).$$

Here, $\mathbf{h}^{(\ell)}_i$ represents the output from only one attention head. The model can accommodate multiple attention heads resulting in multiple $\mathbf{h}^{(\ell)}_i$. These representations can be concatenated or added as suggested by Vaswani et al. [46] for improving the stability of the attention mechanism. The inference model, generator model and the ELBO remain the same as Eq. (1), Eq. (4) and Eq. (5).

### 4.3 HetSANN-BAM-GVAE

Adding uncertainty to the attention weights allows the model to accommodate the node type specific differences in the semantics of nodes and links in a heterogeneous graph. Hence, we further improve the HetSANN aggregation by integrating it with probabilistic attention modules Fan et al. [43].

In HetSANN-BAM-GVAE, we consider a Bayesian model with a prior distribution $p(\mathbf{S})$ and likelihood $p(\mathbf{A}|\mathbf{Z})$; where $\mathbf{S}$ is the set of attention coefficients in the model. Hence, $\theta_{ij}^{(\ell)}$ coefficients of the HetSANN layer are replaced by their stochastic counterparts, i.e., $\text{HetSANN}_{\text{BAM}}$ layer with $S_{ij}^{(\ell)}$ coefficients. The mechanism for calculating $S_{ij}^{(\ell)}$ remains the same as in Eq. (6), but a prior belief is incorporated in the weights, which helps in regularization.

In an attention mechanism, importance weights are calculated between the key-query vector pairs. These weights are non-negative and follow simplex constraints. Hence, the prior distribution over such weights needs to enforce such constraints on the samples. A natural choice for such a prior is the Dirichlet distribution. Unfortunately, Dirichlet distribution is non-reparameterizable and hence cannot be used in a model that uses gradient descent-based optimization. Fan et al. [43] suggest a reparameterizable, non-negative distribution, such as Weibull or Log-Normal, to generate the samples and normalize subsequently to satisfy the simplex constraint.

We use the Weibull distribution, with parameters $k$ and $\lambda$, as the variational distribution. The Gamma distribution, with parameters $\alpha$ and $\beta$, is used as a prior. The parameter $k$ in the Weibull variational posterior is the global hyper-parameter for the attention coefficients. Since we do not use a contextual prior [43], $\alpha$ and $\beta$ are kept fixed hyper parameters for $p(\mathbf{S})$. Using an amortized variational inference, $\lambda$ is computed using:

$$\lambda_{ij}^{(\ell)} = \exp S_{ij}^{(\ell)} / \exp \Gamma(1 + \frac{1}{k}),$$

where $\Gamma$ is the Gamma function. Note that we use a Weibull distribution because it is reparametrizable and produces non-negative samples.

$$\hat{S}_{ij}^{(\ell)} = \lambda_{ij}^{(\ell)} (-\log(1-\epsilon))^{1/k};\ \epsilon \sim \text{Uniform}(0,1)$$

The sampled values $\{\hat{S}_{ij}^{(\ell)}\}_j$ are normalized using the softmax activation to satisfy the simplex constraint of the attention coefficients i.e. $\sum_j \hat{S}_{ij}^{(\ell)} = 1$. These samples are then used in the propagation rule for the $\text{HetSANN}_{\text{BAM}}$ layer:

$$\mathbf{h}_i^{(\ell)} = f(\sum_{j \in \mathcal{N}_i} \hat{S}_{ij}^{(\ell)} \mathbf{W}_{\phi(i),\phi(j)}^{(\ell)} \mathbf{h}_j^{(\ell-1)}). \tag{8}$$

Eq. (8) is the operation which is computed in the $\text{HetSANN}_{\text{BAM}}$ layer. After stacking multiple layers of representations, we obtain the final representation $\mathbf{Z}$:

$$\mathbf{Z} = \zeta(\text{HetSANN}_{\text{BAM}}(\mathcal{X}, \mathbf{A})).$$

Since the Weibull distribution closely resembles the Gamma distribution and the KL divergence of the Weibull distribution from a Gamma distribution has an analytical form [47], the Gamma distribution is used as a prior:

$$\begin{aligned} \text{D}_{\text{KL}}\left(\text{Weibull}(k,\lambda) \| \text{Gamma}(\alpha,\beta)\right) = \gamma\alpha/k - \alpha \log \lambda \\ + \log k + \beta\lambda\Gamma(1+1/k) - \gamma - 1 - \alpha \log \beta + \log \Gamma(\alpha), \end{aligned} \tag{9}$$

where $\gamma$ is the Euler's constant. Putting it all together, the ELBO for the model is:

$$\mathcal{L} = \mathbb{E}_{q(\mathbf{S})} \log p(\mathbf{A}|\mathbf{Z}) - D_{KL}\left[q(\mathbf{S}) \| p(\mathbf{S})\right].$$

Here, $D_{KL}\left[q(\mathbf{S}) \| p(\mathbf{S})\right]$ has a non-analytical form because of the dependencies between different stochastic layers. Hence, the joint distributions $p(\mathbf{S})$ and $q(\mathbf{S})$ are decomposed into a product of conditionals which have an analytic solution. These conditionals are derived using the dependencies between the stochastic layers and has the form $q(\mathbf{S}^{(\ell)}|\mathbf{S}^{(1:\ell-1)})$ and $p(\mathbf{S}^{(\ell)}|\mathbf{S}^{(1:\ell-1)})$. This alternative

tractable KL-Divergence is represented using $\Lambda$ and results in a semi-analytical form of the ELBO as given below:

$$\mathcal{L} = \mathbb{E}_{q(\mathbf{S})} \log p(\mathbf{A}|\mathbf{Z}) - \sum_{\ell=1}^{L} \mathbb{E}_{q(\mathbf{S}^{(1:\ell-1)})} \mathrm{D}_{\mathrm{KL}} \left[ q(\mathbf{S}^{(\ell)}|\mathbf{S}^{(1:\ell-1)}) \middle\| p(\mathbf{S}^{(\ell)}|\mathbf{S}^{(1:\ell-1)}) \right].$$

The generator model remains the same as that specified by Eq. (4). The ELBO is optimized with respect to all of the parameters of the model.

# 5 Experiments

In this section, we empirically compare the performance of each of the models presented in Section 4 with state-of-the-art baselines on the link prediction task using three heterogeneous graph benchmark data sets.

## 5.1 Benchmark Data

We perform experiments with three different data sets:IMDB, DBLP, and AMiner. These data sets also come with labels of the nodes. We ignore such labels since we aim to impute the edges and learn the embeddings based on the adjacency graph (It would be interesting to incorporate information supplied by the node labels to construct a supervised loss function for link prediction). To reduce memory footprint and computational overhead, we further prune the DBLP and the AMiner data sets by randomly removing a fraction of the nodes. The details of the IMDB datasets and the pruned DBLP and AMiner datasets are as follows:

1. **IMDB** is an online database about movies that contains three types of nodes; $4278$ Movies (M), $2081$ Directors (D), and $5257$ Actors (A). The features of the nodes in this data set are represented as bag-of-words, containing the key-words related to a specific type of node. The IMDB data set contains only two types of edges; $4278$ Movie-Director edges and $12828$ Movie-Author edges. For meta-path based schemes we have used *MAM*, *MDM*, *MDMAM*, *MAMDM*, *AMA* and *DMD*.
2. The **DBLP** dataset used for the experiments is a subset of the original dataset, a heterogeneous academic graph data set extracted by Gao et al. [48], Ji et al. [49]. The original dataset contains $4057$ Authors (A), $14328$ Papers (P), $7723$ Terms (T), and $20$ Venues (V) with $19645$ Author-Paper edges, $85810$ Paper-Term edges, and $14328$ Paper-Venue edges. The reduced data set used in this paper contains 2303 Authors (A), $5328$ Papers (P), $4007$ Terms (T), and $9$ Venues (V) with $7127$ Author-Paper edges, $31740$ Paper-Term edges, and $5328$ Paper-Venue edges. The features of the authors, papers, and terms all have bag-of-words representation, whose dimensions are reduced further using PCA [50]. For meta-path based schemes we have used *APA*, *TPT*, *VPV*, *PAP*, *PTP*, *PVP*, *APTPA*, *PTPAPVP*, *TPAPVPT* and *APVPTPA*.
3. **AMiner** is a dataset extracted by Hong et al. [40], where the authors remove the venue nodes from the AMiner academic graph, leaving only $8052$ Authors (A) and $20201$ Papers (P). This data set contains $31224$ publishing relations (Author-Paper), $44551$ citation relations (Paper-Paper), and $32029$ collaboration relations (Author-Author). Features of the papers are a bag of words representing the keywords. Each author in this graph has five features – the number of citations, H-Index, P-index with equal A-index, and P-index with unequal A-index. We further prune this graph, resulting in $3588$ Authors (A) and $8129$ Papers (P) with $7519$ publishing relations (Author-Paper), $7100$ citation relations (Paper-Paper), and $11852$ collaboration relations (Author-Author). Note that not all papers are connected to a legitimate author, making link prediction in such a noisy graph even more challenging. Furthermore, out of the 3 data sets, this one is the most densely connected with only 2 types of nodes. In such a graph, random walks based method usually perform really well; thus providing a challenge to non-random walks methods. For meta-path based schemes we have used *PP*, *PAP*, *PAPP*, *PPAP*, *PAPAP*, *AA*, *APA*, *APAA*, *AAPA* and *APAPA*.

## 5.2 Baseline Methods

We considered the following baseline methods:

1. **Node2vec** [34] is an algorithmic framework for representational learning on homogeneous graphs, which is a generalized version of DeepWalk [33]. We apply it to our datasets by ignoring the heterogeneity of graph structure and dropping all node features. After obtaining the representations for the nodes, edge features are generated using binary operators. These edge features are then used in a logistic regression classifier to predict binary values indicating whether an edge is present. Average, Hadamard, L-1, and L-2 operators are used as the binary operators, and the best results out of the four operators are presented in Table 3

2. **HAN** [39] is a heterogeneous GNN model that computes node representations from multiple meta-path-based homogeneous graphs and uses the attention mechanism to combine them. To generate edges from node representations, we use the inner product decoder from Eq. (4). Meta-path adjacency matrices are obtained by multiplying sub-type adjacency matrices. E.g., in the IMDB dataset, the MAM meta-path matrix is obtained by multiplying the adjacency matrix MvA and AvM (Transpose of MvA).

3. **GVAE** [26] We use a homogeneous GVAE as a baseline model. That is, we treat the heterogeneous graph as if it were a homogeneous graph. Note that this model assumes that the features of all the nodes lie in the same semantic space and are of the same dimensionality. However, as discussed in the previous sections, the nodes in HIN may have attributes of different sizes based on the nodes' type. To resolve such discordance, we equalize the dimensionality of features across node types by padding them with zeros as needed.

### 5.3 Experimental Setup

The edges of the datasets are randomly split into training, test, and validation with 85%, 10%, and 5% split percentages . Node2vec models parameters are the same for all the datasets. For Node2vec we set the walklength to 50 and number of walks to 10. The window size is set to 10 and the embedding dimension size is 128 for both Node2vec.

For GNN based models, we observe overfitting in the IMDB dataset if we use more than one layer. For both DBLP and AMiner, however, we use two layers. For attention based models, We assign three different attention heads for each semantic relation for the models that use attention. For DBLP and AMiner, we use two layers with five attention heads in the first layer and three attention heads in the second layer. Each attention head's values are concatenated in each set if the layer is not the last; otherwise, the values get summed up.

For GNN based models, we set the output dimension size to $32$ for all the experiments with the IMDB data set. We set the hidden layer's size to $32$ and the output dimension to $16$ for the experiments with DBLP and AMiner. For the HetSANN-BAM-GVAE model, the parameters of the Gamma prior distribution are $\alpha = 1e - 17$ and $\beta = 1e - 15$. For the Weibull variational posterior, $k$ is set to $10$. The learning rates for GVAE and H-GVAE are $0.01$ and for HetSANN-GVAE and HetSANN-BAM-GVAE are $0.005$. All the models are trained using the Adam optimizer [51].

We use the same parameters that produce the best results on the validation set to predict links on the test set. We report the mean performance on the test set obtained across 15 runs with random initialization on fixed data set splits.

### 5.4 Link Prediction Results

The results of our experiments are shown in table 3.

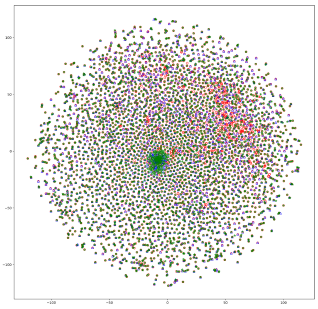
Table 3 shows that that on IMDB and DBLP data, random walk based methods which do not use node features perform substantially worse than the graph neural network based methods. Among the graph based methods, GVAE, which treats a heterogeneous graph as a homogeneous graph, and therefore fails to make use of semantic and structural information provided by the heterogeneous graph, produces unsatisfactory results compared to the rest of the graph based models. Even though H-GVAE does not useattention whereas HAN does, the performance of H-GVAE is comparable to that of HAN on IMDB and DBLP data, and substantially superior to that of HAN on AMiner data. HetSANN-GVAE substantially outperforms H-GVAE. This suggests the effectiveness of utilizing the structural and semantic information in heterogeneous graphs in HetSANN-GVAE. HetSANN-BAM-GVAE outperforms all other methods on all data sets.

Table 3: AUC and AP results Results for Link Prediction

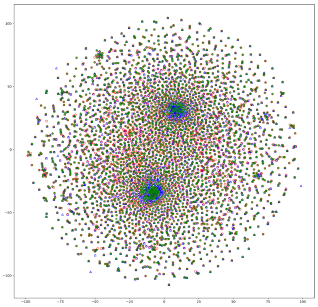
| Dataset | Metrics | Node2vec | GVAE | HAN | H-GVAE | HetSAAN GVAE | HetSAAN BAM GVAE |
|---|---|---|---|---|---|---|---|
| IMDB | AUC | 0.6310 | 0.8907 | 0.9025 | 0.9012 | 0.9441 | **0.9503** |
| | AP | 0.6354 | 0.9166 | 0.9191 | 0.9163 | 0.9522 | **0.9549** |
| DBLP | AUC | 0.8152 | 0.8296 | 0.8676 | 0.8536 | 0.8917 | **0.8985** |
| | AP | 0.7910 | 0.8566 | 0.8814 | 0.8912 | 0.9182 | **0.9214** |
| AMiner | AUC | 0.8565 | 0.8150 | 0.7432 | 0.8411 | 0.8476 | **0.8588** |
| | AP | 0.8531 | 0.8430 | 0.7901 | 0.8634 | 0.8771 | **0.8821** |

We further note that AMiner data presents challenges for most of the models due to the fact that it is partially homogeneous and partially heterogeneous graph. HAN underperforms all the other models on AMiner data. It could be the case that HAN needs better and perhaps longer meta-path, but specifying such a meta-path is itself a notoriously difficult task. Interestingly, Node2vec performs reasonably well compared to the best performing model, i.e., the HetSANN-BAM-GVAE.

The main observation deduced from the T-SNE projections is about the discriminative power of the models—the GVAE model clusters the majority of the nodes whose edges have been inaccurately predicted towards the center. The introduction of transformations for the heterogeneous features and variational attention modules improves the discriminative power as the distance between the clusters increases, an indicator of the ability to distinguish between the different neighborhood structures.



(a) GVAE



(b) HetSANN-BAM-GVAE

Figure 1: T-SNE projections of the embeddings for the IMDB dataset. RGB colors indicate Movies, Directors and Actors respectively.

## 6 Conclusion

We considered three variants of graph variational autoencoders for learning node embeddings from heterogeneous graphs. Two of the three proposed variants outperform several state-of-the-art baselines on three real-world benchmark data sets on the link prediction task. Their superior performance can be attributed to their ability to incorporate the structural and semantic information supplied by heterogeneous graphs, to model the uncertainty of node embeddings, and to to exploit node type specific attention variables. Further improvements are possible, e.g., by using better decoders [52, 53], or utilizing a better prior [54–58]. These possibilities offer some exciting directions for future work.


### Acknowledgements

This work was supported in part by a grant to Suhang Wang and Vasant Honavar from the George Mason University's Criminal Investigations and Network Analysis Center funded by the Department of Homeland Security, and in part by a National Science Foundation Grant (IIS 2041759) to Vasant Honavar. This work has benefited from discussions with Suhang Wang.


## References


[1] Zan Huang, Wingyan Chung, and Hsinchun Chen. A graph model for e-commerce recommender systems. *Journal of the American Society for information science and technology*, 55(3):259–274, 2004.

[2] Zan Huang, Daniel D Zeng, and Hsinchun Chen. Analyzing consumer-product graphs: Empirical findings and applications in recommender systems. *Management science*, 53(7):1146–1164, 2007.

[3] Chuan Shi, Chong Zhou, Xiangnan Kong, Philip S Yu, Gang Liu, and Bai Wang. Heterecom: a semantic-based recommendation system in heterogeneous networks. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1552–1555, 2012.

[4] Yukuo Cen, Xu Zou, Jianwei Zhang, Hongxia Yang, Jingren Zhou, and Jie Tang. Representation learning for attributed multiplex heterogeneous network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1358–1368. ACM, 2019.

[5] Seth A Myers, Aneesh Sharma, Pankaj Gupta, and Jimmy Lin. Information network or social network? the structure of the twitter follow graph. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 493–498, 2014.

[6] Yutao Zhang, Jie Tang, Zhilin Yang, Jian Pei, and Philip S Yu. Cosnet: Connecting heterogeneous social networks with local and global consistency. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1485–1494, 2015.

[7] Qingbo Hu, Sihong Xie, Jiawei Zhang, Qiang Zhu, Songtao Guo, and Philip S Yu. Heterosales: Utilizing heterogeneous social networks to identify the next enterprise customer. In *Proceedings of the 25th International Conference on World Wide Web*, pages 41–50, 2016.

[8] Nenad Trinajstic. *Chemical graph theory*. Routledge, 2018.

[9] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 30(8):595–608, 2016.

[10] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *ICML*, 2017.

[11] Suhang Wang, Jiliang Tang, Charu Aggarwal, and Huan Liu. Linked document embedding for classification. In *CIKM*, pages 115–124. ACM, 2016.

[12] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of SIGKDD*, pages 701–710. ACM, 2014.

[13] William L Hamilton. Graph representation learning. *Synthesis Lectures on Artifical Intelligence and Machine Learning*, 14(3):1–159, 2020.

[14] Deepayan Chakrabarti and Christos Faloutsos. Graph mining: Laws, generators, and algorithms. *ACM computing surveys (CSUR)*, 38(1):2–es, 2006.

[15] U Kang, Charalampos E Tsourakakis, and Christos Faloutsos. Pegasus: A peta-scale graph mining system implementation and observations. In *2009 Ninth IEEE International Conference on Data Mining*, pages 229–238. IEEE, 2009.

[16] Jure Leskovec and Rok Sosič. Snap: A general-purpose network analysis and graph-mining library. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(1):1–20, 2016.

[17] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[18] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.

[19] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[20] Yue Deng. Recommender systems based on graph embedding techniques: A review. *IEEE Access*, 2022.

[21] Hai-Cheng Yi, Zhu-Hong You, De-Shuang Huang, and Chee Keong Kwoh. Graph representation learning in bioinformatics: trends, methods and applications. *Briefings in Bioinformatics*, 23(1): bbab340, 2022.

[22] Fenxiao Chen, Yun-Cheng Wang, Bin Wang, and C-C Jay Kuo. Graph representation learning: a survey. *APSIPA Transactions on Signal and Information Processing*, 9, 2020.

[23] Qiaoyu Tan, Ninghao Liu, and Xia Hu. Deep representation learning for social network analysis. *Frontiers in big Data*, 2:2, 2019.

[24] Weiyi Gong and Qimin Yan. Graph-based deep learning frameworks for molecules and solid-state materials. *Computational Materials Science*, 195:110332, 2021.

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

[26] Thomas Kipf and Max Welling. Variational graph auto-encoders, 2016.

[27] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *Proc. of ICML*, pages 6861–6871, 2019.

[28] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *Proc. of ICLR*, 2018.

[29] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *Proc. of ICLR*, 2019.

[30] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S Yu, and Tianyi Wu. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *Proceedings of the VLDB Endowment*, 4(11):992–1003, 2011.

[31] Guojia Wan, Bo Du, Shirui Pan, and Gholameza Haffari. Reinforcement learning based meta-path discovery in large-scale heterogeneous information networks. In *Proc. of the AAAI Conference on Artificial Intelligence*, pages 6094–6101, 2020.

[32] Rana Hussein, Dingqi Yang, and Philippe Cudré-Mauroux. Are meta-paths necessary? revisiting heterogeneous graph embeddings. In *Proc. of CIKM*, page 437–446, 2018. ISBN 9781450360142.

[33] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. *Proc. of KDD*, 2014.

[34] Aditya Grover and Jure Leskovec. Node2vec: Scalable feature learning for networks. In *Proc. of KDD*, page 855–864, 2016.

[35] Daxin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proc. of KDD*, page 1225–1234, 2016.

[36] Yuxiao Dong, Nitesh V. Chawla, and Ananthram Swami. Metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 135–144, 2017.

[37] Chuan Shi, Binbin Hu, Wayne Xin Zhao, and S Yu Philip. Heterogeneous information network embedding for recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 31 (2):357–370, 2018.

[38] Ting Chen and Yizhou Sun. Task-guided and path-augmented heterogeneous network embedding for author identification. In *Proc. of WSDM*, page 295–304, 2017.

[39] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. Heterogeneous graph attention network. In *Proc. of WWW*, page 2022–2032, 2019.

[40] Huiting Hong, Hantao Guo, Yucheng Lin, Xiaoqing Yang, Zang Li, and Jieping Ye. An attention-based graph neural network for heterogeneous structural learning. In *Proc. of AAAI*, 2020.

[41] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2014.

[42] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proc. of WWW*, page 1067–1077, 2015.

[43] Xinjie Fan, Shujian Zhang, Bo Chen, and Mingyuan Zhou. Bayesian attention modules. In *Proc. of NIPS*, volume 33, pages 16362–16376, 2020.

[44] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Proc. of ICLR*, 2013.

[45] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017.

[46] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proc. of NIPS*, volume 30, 2017.

[47] Hao Zhang, Bo Chen, Dandan Guo, and Mingyuan Zhou. WHAI : Weibull hybrid autoencoding inference for deep topic modeling. In *Proc. of ICLR*, 2018.

[48] Jing Gao, Feng Liang, Wei Fan, Yizhou Sun, and Jiawei Han. Graph-based consensus maximization among multiple supervised and unsupervised models. In *Proc. of NIPS*, 2009.

[49] Ming Ji, Yizhou Sun, Marina Danilevsky, Jiawei Han, and Jing Gao. Graph regularized transductive classification on heterogeneous information networks. In *Proc. of ECML-PKDD*, page 570–586, 2010.

[50] Ian T. Jolliffe. Principal component analysis. In *New York: Springer*, 1986.

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

[52] Mingyuan Zhou. Infinite edge partition models for overlapping community detection and link prediction. In *Proc. of AISTATS*, pages 1135–1143. PMLR, 2015.

[53] Aditya Grover, Aaron Zweig, and Stefano Ermon. Graphite: Iterative generative modeling of graphs. In *Proc. of ICLR*, pages 2434–2444, 2019.

[54] Jakub Tomczak and Max Welling. VAE with a VampPrior. In *Proc. of AISTATS*, pages 1214–1223, 2018.

[55] Amr Ahmed, Nino Shervashidze, Shravan Narayanamurthy, Vanja Josifovski, and Alexander J. Smola. Distributed large-scale natural graph factorization. In *Proc. of WWW*, pages 37–48, 2013.

[56] Arman Hasanzadeh, Ehsan Hajiramezanali, Krishna Narayanan, Nick Duffield, Mingyuan Zhou, and Xiaoning Qian. Semi-implicit graph variational auto-encoders. volume 32, 2019.

[57] Mingzhang Yin and Mingyuan Zhou. Semi-implicit variational inference. In *Proc. of ICML*, 2018.

[58] Ali Lotfi-Rezaabad, Rahi Kalantari, Sriram Vishwanath, Mingyuan Zhou, and Jonathan I. Tamir. Hyperbolic graph embedding with enhanced semi-implicit variational inference. In *Proc. of AISTATS*, 2021.