## Student's Declaration

I hereby declare that the work presented in the report entitled **"Network Embedding For Multilayer Graph"** submitted by me for the partial fulfillment of the requirements for the degree of *Bachelor of Technology* in *Computer Science & Engineering* at Indraprastha Institute of Information Technology, Delhi, is an authentic record of my work carried out under guidance of **Dr. Tanmoy Chakraborty**. Due acknowledgements have been given in the report to all material used. This work has not been submitted anywhere else for the reward of any other degree.

**Aman Roy** **Place & Date: 11 December 2019**
**...(student's name)...**

## Certificate

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

**.....** **Place & Date: .....**
**..(advisors' name)...**

## Acknowledgments

I would like to thank Dr. Tanmoy Chakraborty, Hridoy Dutta(Ph.D Scholar at IIIT-D) and Vinayak Kumar for their constant support and guidance to enable me to better understand concepts and different ideas. I would also like to thank my advisor, Dr. Tanmoy Chakraborty for giving me an opportunity to pursue this project.

## Work Distribution

Since I am working alone on this project, there was no distribution of work.

# Contents

# Chapter 1

# Introduction

Representation learning of networks has gained considerable attention in recent times [6, 7, 12, 16, 17, 19, 20]. The goal of this body of research is to learn a low dimensional, dense representation for each node in a network while preserving structural information about the neighborhood of the node. Seminal works, such as DeepWalk [16] and Node2Vec [7], were inspired by the prowess of unsupervised feature learning in Natural Language Processing domain. They applied the Skip-gram algorithm of Word2Vec [15] to learn network embeddings such that nodes which have same neighborhood context will end up getting similar representations. These embeddings which are trained in a purely unsupervised manner have proven to be effective in a variety of downstream social network applications, *e.g.,* link prediction, node clustering, multi-label classification of nodes, etc.

Majority of the research in network embedding has dealt with networks where nodes share a single form of relationship. However, most real-world networks are *multigraphs* as multifaceted relationships are quite common between their nodes. This is known as the *multiplexity* property [22] in social networks. For instance, a pair of users on a social network like Facebook can be related through friendship, messaging, etc. In a scientific network, researchers can share a link by virtue of being co-authors on a paper or by citing each other's works. A high quality representation of a node in a multigraph should not only capture information about its neighboring nodes but also encode the relationships that exists with its neighbors. Hence, network embedding methods built for homogeneous networks must be extended so that they are able to characterise the rich context present in the multiple types of edges. Early work in multigraph embedding [12, 25] represents a multigraph as a *multilayer network* for ease of analysis. The multilayered representation is created by stacking multiple edge homogeneous networks together. Every node in the multigraph, which has $k$ types of edges incident on it, gets split into $k$ nodes in the multilayered network such that each layer has edges of a single type only. Also, connections are added amongst the $k$ counterparts that are created from the same node on the multigraph. Now, one can train embeddings for each layer separately and aggregate different layer-specific embeddings to arrive at a unified embedding for a node in the multigraph [23, 25]. However, such an approach fails to capture interactions between layers. To address this, Liu *et al.* [12] proposed to learn embeddings from node sequences generated by random walks where switching between one layer to another is possible. The layer switching probability used in the random walk needs to be empirically tuned. In practice, such hyperparameter tuning is laborious. Moreover, it is not reasonable to assume that each layer is equally important; hence uniformly choosing a layer when switching can be ineffective.

In this work, we propose Multigraph2Vec++[1], a novel method for learning node embeddings in multigraphs. Multigraph2Vec++ is a graph attention network based embedding generation method. It employs a novel way to capture the multilayer context and supervise the node embeddings on the basis

[1]Code and partial datasets are available at [9].

of high priority and low priority nodes. The method uses a novel attention based strategy to aggregate multi-relational context of a node. It also uses a novel loss method to finally do the supervision. Multigraph2Vec++ preserves multi-relational interaction among nodes via generating a (edge) heterogeneous context.

Experiments on three real-world datasets show that Multigraph2Vec++ outperforms seven baselines – it beats the best baseline by 6.30% higher AUC score (averaged over all datasets and all layers) on link prediction experiment and improves MNE very substantially in node classification experiment.

In short, our contributions are threefold: (i) capturing the multirealtional context of a node using a novel attention based method for multilayer graph, (ii) supervising the node embeddings using a novel loss function, (iii) a comprehensive analysis to show the superiority of Multigraph2Vec++.

# Chapter 2

# Related Work and Problem Statement

In this section, we present a brief literature survey of the embedding methods developed for different types of networks.

**Homogeneous Network Embedding:** Representation learning for homogeneous networks (simple network with only one type of node and one type of edge) has been studied extensively. DeepWalk [16] follows Skip-gram model [14] a two phase algorithm for learning node embedding. Node2Vec [7] employs a second order random walk governed by two parameters that control the breadth first search and depth first search nature of the random walker. LINE [20] learns the node embedding such that a certain measure of proximity among the nodes in the network is preserved in the embedding space. Matrix factorization approaches such as Spectral clustering [19] perform eigendecomposition of a certain representation of the graph like normalized Laplacian matrix and then chooses top few eigenvectors as features for representing nodes.
Recently, inspired from the success of convolutional neural networks in computer vision, a number of network embedding methods have surfaced with redefined convolution operation for graphs. These are termed as Graph Convolutional Networks (GCN). Bruna et al. [2] developed a variant of GCN based on spectral graph theory. Hamilton et al. [8] performed convolution in graph domain by aggregating information of neighboring nodes. In [8], computation can be performed on batch of nodes instead of the whole graph which improves the efficiency of the network. In addition to GCN, a number of other Graph Neural Network approaches have also been proposed. Abu-El-Haija et al. [1] parameterizes the context distribution function and use softmax attention to learn the importance of $k^{th}$ hop neighbors. Graph generative networks, on the other hand, aim to generate structures from data. You et al. [24] generated node and edges alternatively in adversarial setting. Li et al. [10] modified the graph Neural network framework to use Gated Recurrent Units and back propagation through time.

**Heterogeneous Network Embedding:** A heterogeneous information network (HIN) consists of multiple types of edges and nodes with only one edge connecting any two nodes. Metapath2vec [6] samples a heterogeneous context for a node using random walks guided by the predefined metapaths (a path consisting of a specific sequence of relationships/edge-types). Chang *et al.* [3] used the content and the linkage structure to generate important cues for creating a unified feature representation of the underlying network. HERec [17] generates meaningful node sequences via random walk. PME [4] is based on metric learning and captures both first-order and second-order proximities in a unified way.

**Multidimensional Network Embedding:** Despite its profound relevance in real-world scenario, limited attempts have been made to address multilayer/multidimensional embedding, compared to the vast amount of literature on simple graphs. MNE [25] generates a $d$-dimensional layer-specific embedding for each node by combining its $d$-dimensional base embedding (which remains common across layers), a transformation matrix (which is learned for each layer) and an $s$-dimensional auxiliary layer-specific

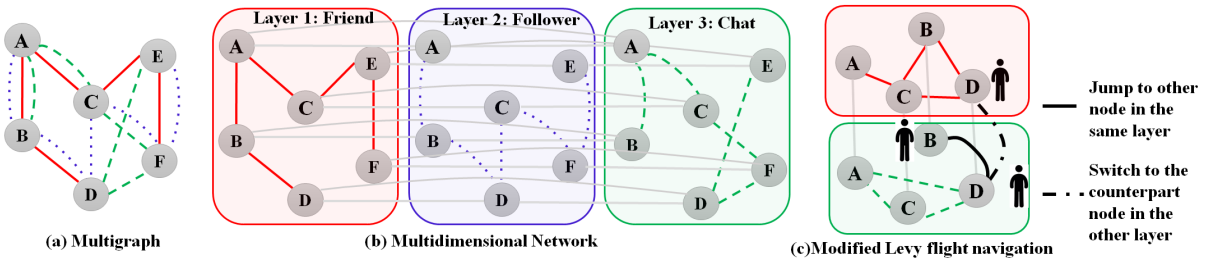


Figure 2.1: (a) A toy example of a multigraph representing a social network (e.g, Facebook) where nodes are users, and they are connected via at most three types of edges (red: friend, blue: follower, green: chat). It is not necessary that all pairs of nodes are always connected by three different edges. For example, nodes $C$ and $E$ are connected by only friendship link. But they do not follow each other and do not chat. (b) Each edge type mentioned in Fig (a) forms a layer/dimension in the corresponding multidimensional network. Node set remains same across layers, and each layer is homogeneous in terms of the edge type. (c) Illustration of the modified Levy flight strategy on a multidimensional network. The figure shows the different elementary steps that the random walker can adopt.

embedding ($s << d$). Ma *et al.* [13] considered all the immediate neighbors (in each layer) of a node as its context and then employed a specific Skip-gram with a softmax taking into consideration node categories and layer information to obtain the embeddings for each node. PMNE [12] extends Node2Vec by introducing another parameter that allows the random walker to traverse across layers while sampling the context for each node. It does not learn this parameter, rather tunes it manually. ASNE [11] is a graph embedding algorithm‘which learns an embedding of nodes and fuses the node representations with node attributes. It learns the joint feature-proximal representations using a probabilistic factorization model. **Comparison with Multigraph2Vec++:** None of the homogeneous and heterogeneous network embedding methods can directly be applied to multigraphs. We adopted Node2vec, LINE, WatchYourStep [**?** ], to our setting as baselines (Section 5). Multigraph2Vec++ samples (edge-)heterogeneous contexts for multigraphs. Multigraph2Vec++ is different from PMNE, MNE and ASNE as it uses a novel attention based strategy to aggregate multirelational context of a node. It also allows to capture those nodes in context which might be disconnected from the current node. It uses a novel loss function which is used to supervise the node embeddings and update the attention parameters.

## 2.1 Problem Statement

We address the problem of learning a low dimensional representation for the nodes in a multigraph (see Definition 1), which can be further used for downstream applications.

**Definition 1 (Multigraph).** *It is defined as a directed graph (directed, for the sake of genralization) $G = (V, E, L)$, where $V$ is the set of vertices, $L$ is the set of edge types and $E$ is the set of triplets $(v_i, v_j, l)$ representing an edge of type $l$ directed from node $v_i$ to $v_j$, where $v_i, v_j \in V$ and $l \in L$. There can be multiple edges of varying types between any two vertices.*

Figure 2.1(a) shows a toy example of a multigraph. Nodes are assumed to be of same type. For the sake of better representation, we unfold a multigraph to a *multidimensional network*. Multidimensional networks, a special type of multilayer network[1], are networks with multiple kinds of edge types [5]. There are as many layers/dimension as that of edge types, and the edges between nodes within each layer are simply all the edges with a given type (see Figure 2.1).

**Definition 2 (Multidimensional Network).** *It can be defined as a multilayer network $G = (V, E, L)$ having $|L|$ layers or dimensions. $V$ denotes a set of $N$ unique nodes. A node $v_i \in V$ in layer $l \in L$*

[1]Multilayer network is a stacked representation of multiple single layers. Multidimensional network is a special type of multilayer network which is edge-homogeneous i.e., each layer represents a particular type of relationship among nodes.

*is denoted by* $v_i^l$ , $1 \leq i \leq N;\ 1 \leq l \leq |L|$. *Each edge* $E_{i,j}^l \in E$ *is a tuple* $(v_i, v_j, l)$ *representing an edge of type* $l$ *emanating from node* $v_i^l$ *to node* $v_j^l$, *where* $v_i, v_j \in V$. *Essentially,* $G$ *consists of a total of* $|V| \times |L|$ *number of nodes. For simplicity, we assume that all nodes in* $V$ *are present in all the layers. If a node is absent in a particular layer* $l$ *(i.e. no edge of type* $l$ *connects to that node), we add the node as an isolated node in layer* $l$.

**Definition 3** (**Problem Statement).** *Given a multigraph* $G = (V, E, L)$, *our aim is to learn a low dimensional representation (embedding) for each node* $v_i \in V$, *i.e.,* $X_i \in R^D$, *where* $D << |V|$.

# Chapter 3

# Multigraph2vec

We propose Multigraph2Vec++ which is essentially a Skip-gram model with a novel neighborhood (context) sampling strategy for multigraphs.

## 3.1 Graph Attention Network

The architecture of Graph Attention Network(GAT) was proposed by [21] which was used to learn node embeddings which was used in performing node classification as underlying task both in inductive and transductive settings. In order to obtain embeddings for a target node $v_i$, it uses the cross entropy loss to update the weights and attention parameters.
The input to GAT is a set of node features $\overrightarrow{h} = \{\overrightarrow{h_1}, \overrightarrow{h_2}, ...., \overrightarrow{h_N}\}$ where each $\overrightarrow{h_1} \in R^F$ where N is the number of nodes in graph and F is the number of features one has for each node. The GAT layer produces a new set of node features(of different embedding dimension $F^{'}$), $\overrightarrow{h^{'}} = \{\overrightarrow{h_1^{'}}, \overrightarrow{h_2^{'}}, ...., \overrightarrow{h_N^{'}}\}$ as its output.
In order to have sufficient expressive power a linear transformation is applied on input features to transform it into higher level features. To incorporate linear transformation, a shared weight vector $\mathbf{W} \in R^{F*F^{'}}$ is applied on input features to every node followed by a shared attentional mechanism $a : R^{F*F^{'}} \rightarrow R$ which computes attention coefficient as

$$e_{i,j} = a(\mathbf{W}h_i, \mathbf{W}h_j)$$

$e_{i,j}$ tells the importance of node features of node j for node i. Due to its general structure a node can attend every other node despite not having direct connections between nodes. The computed attention coefficient is normalized using softmax function over its neighborhood $N_i$ to get attention score $a_{i,j}$.

$$a_{i,j} = softmax(e_{i,j}) = \frac{exp(e_{i,j})}{\sum_{k \in N_i} exp(e_{i,k})}$$

To capture multiple type of relations and stabilize the training of self attention, multi-headed attention is used where we concatenate the embedding output by all the attention heads.

$$X_i = \Big\Vert_{k=1}^{K} \sigma \left( \sum_{j \in N_i} a_{i,j}^k W^k \overrightarrow{h_j} \right)$$

## 3.2 Graph Attention Network for Multilayer Graph

We propose a novel node embedding method for multilayer graph which extends the attention architecture for multilayer graph and employs an unsupervised loss which can be used to generate node embeddings for link prediction and node classification task.
Embedding for a node $v_i$ in a multilayer graph is generated by aggregating embeddings of the node $v_i$ in individual layers i.e. $\{X_i^1, X_i^2, ...., X_i^L\}$. To incorporate the multirelation in the graph we are using the same set of linear transformation parameter $\mathbf{W}$ and attention head parameters $\overrightarrow{a}$ to learn the embedding for different layers thus same set of parameters learns the different type of relations in the graph essentially capturing the different type of complex relations that can be present between two nodes in a multilayer graph.
Hence the updated equation of the node embedding for multilayer graph with multiheaded attention head becomes

$$X_i = \Big\Vert_{l=1}^{L} \Big\Vert_{k=1}^{K} \sigma \left( \sum_{j \in N_i^l} a_{i,j}^k W^k \overrightarrow{h_j} \right)$$

The shared linear transformation and multiheaded attention is same for every layer, however the difference comes in the neighborhood $N_i^l$ for each layer which ensures that same embedding captures different context around a node in different layer. Next subsection discusses the generation of neighborhood for a particular node in different layer.

### 3.2.1 Constructing High Priority Set or Neighborhood for a Node

The neighborhood of a node $v_i$ is dependent on the nodes which are connected with $v_i$ in layer l and also those nodes whose node features are similar to that of the node features of $v_i$ i.e.

$$N_i^l = \{v_j : \phi(h_i^l, h_j^l) > \epsilon \;||\; E_{i,j}^l \in E^l\}$$

where $\phi$ is a similarity function and $\epsilon$ is a similarity threshold. We chose cosine similarity to compute similarity between node features and after hyper tuning kept a similarity threshold of 0.9.
The neighborhood $N_i^l$ of a node $v_i^l$ consist of all those nodes whose similarity score is above a certain threshold $\epsilon$ or the node is directly connected to $v_i$ in layer l. In this way we are not only restricting our neighborhood to immediate neighbors but are also considering the possibility of edge forming between nodes which have similar node features. The nodes coming in $N_i^l$ are also high priority nodes for $v_i$ in layer l and will be used for creating $H^l$ high priority node information matrix which will be used in final loss function. Section 4.3 discusses what do we mean by high priority nodes how $H^l$ is exactly formed.

### 3.2.2 Constructing Low Priority Set for a Node

Similar to construction of neighborhood for a node, we construct low priority set $R_i^l$ for a a particular node $v_i$ in layer l. It is based upon dissimilarity between node features and also contain sampled nodes from degree distribution of those nodes in layer l which does not come in $N_l^i$.

$$R_l^i = \{v_j : \phi(h_i^l, h_j^l) < \mu \;||\; sample(\{d_i : v_i \notin N_i^l\}, n)\}$$

$sample(\{d_j : v_j \notin N_i^l\}, n)\}$ means sample n nodes from degree distribution of all those nodes in layer l which are not a part of neighborhood set of $v_i$. We set $\mu$ to be equal to 0.2 after doing hyperparameter tuning and kept $n$ to be equal to degree of $d_i$. If that number of nodes cannot be sampled then we returned all nodes which are not there in $N_l^i$. Nodes coming in $R_i^l$ are also low priority nodes for $v_i$in layer l and will be used for creating $L^l$ low priority node information matrix which will also be used in final loss function.

## 3.3 Loss function for Multigraph2vec++

GAT originally proposed a supervised loss function, however in our architecture we need an unsupervised loss function which can supervise our embedding for the underlying link prediction and node classification task. We used Graph Likelihood like loss similar to [1] to supervise the embeddings that we get from the final layer of our graph attention layer $h_i^{'}$. Our loss function is defined as

$$\prod_{l=1}^{L}\left(\prod_{u,v\in V}\sigma\left(f(X^l)_{v,u}\right)^{H^l_{v,u}}\left(1-\sigma\left(f(X^l)_{v,u}\right)\right)^{L^l_{v,u}}\right)$$

- $f(X)$: It is a function which takes $X$ as input and produces the score for each node pair. It is defined as $XX^T$ and $f(X^l)_{v,u}$ is the score for node pair (v,u) in layer l.
- $H$: It is a symmetric adjacency matrix ($H \in R^{N*N}$) which is constructed using neighborhood of a node at a particular layer i.e. using $N_i^l$. $H^l_{v,u}$ is positive and non zero if $u \in N_v^l$ i.e. u is a neighbor of v in layer l. It is constructed in such a way that it maximises the likelihood of those node pairs (v,u) which are high priority nodes (i.e. comes in neighborhood of each other) for each other.
- $L$: Similar to an adjacency matrix which stores information of high priority nodes, there is an adjacency matrix which stores the low priority nodes information i.e. $L \in R^{N*N}$. It is constructed using lower priority set of a node at a particular layer i.e. using $R_i^l$. $L^l_{v,u}$ is positive and non zero if $u \in R_v^l$ i.e. u is a low priority node for v in layer l. It is constructed to ensure that it minimises the sigmoid score of those node pairs (v,u) which are low priority nodes for each other thereby maximising the likelihood.

Maximising the loss function maximises the likelihood for high priority nodes and minimises the likelihood for low priority nodes. Also maximising the individual layer likelihood maximises the overall likelihood for both layer. It comes naturally to minimise the negative log likelihood of above loss function. After incorporating regularization term and negative log likelihood loss our final loss function becomes,

$$\min_{\mathbf{X},\mathbf{W},\mathbf{a}}\sum_{l=1}^{L}||H^l\circ\log(\sigma\left(f(X^l)\right))+L^l\circ\log(1-\sigma\left(f(X^l)\right)\Big)||_1+\beta||\mathbf{W}||_1+\gamma(\sum_{k=1}^{K}||\mathbf{a^k}||_1)$$

# Chapter 4

# Dataset Description

Three multigraph datasets are used in our experiments. Table 4.1 summarizes the datasets.

**EU Projects (EU):** It[1] consists of organizations that participated in projects funded by the European Union under the FP7 and the H2020 framework programs. Two types of edges exist – FP7 collaboration and H2020 collaboration. Some metadata node attributes are provided for each node – the number of H2020 and FP7 projects they are involved, coordinated, and participated in, and their country of origin. Metadata based node pair feature vector is constructed by taking a difference of the two corresponding node attribute vectors, with the last element of the vector being 0 if the country of origin is not same, 1 otherwise.

**Hike:** We took a sample dataset of Hike Messenger, a popular social app [23]. It is a multigraph having two types of edges, namely the Contact Book and Friends. Users form the nodes of the network. Contact book type edge connects two nodes if they are in phone contact book of each other, while friend type edge connects two users if they are friends of one another on the platform. We were also provided with eight numeric node attributes – number of messages sent/received, number of stories posted/liked/viewed etc. in a certain period of time. Metadata-based node pair feature vector is constructed by taking the difference of the two corresponding node attribute vectors.

**Lazega Dataset:** This social network consists of three kinds of directed relationships/edges (co-work, friendship, and advice) between partners and associates of a corporate partnership [18]. Node-level metadata information is not present in this dataset.

Table 4.1: Summary of the datasets.

| Dataset | # of nodes | Layer (# edges) | Metadata |
|---|---|---|---|
| EU | 1319 | FP7 (114845 )<br>H2020 (75013) | Yes |
| Hike | 1230 | Contact Book (2605)<br>Friend (4666) | Yes |
| Lazega | 71 | Co-work (892)<br>Friendship (575)<br>Advice (1104) | No |

[1] http://data.europa.eu/euodp/en/data/dataset/cordisfp7projects

# Chapter 5

# Evaluation and Conclusion

We show the efficiency of Multigraph2Vec++ on link prediction task [25][7].

## 5.1 Experimental Setup

We set the similarity threshold $\epsilon$ to 90%. We set the dissimilarity threshold $\mu$ to 20%. We used a single layer linear network along with $k$ multiheaded attention parameter for generating embeddings in latent space. Network structure based attributes of a node at any layer are – its degree, clustering coefficient, betweenness centrality, closeness centrality computed for the corresponding layer's network. Network Structure based attributes along with metadata based attributes(wherever available) were used as node attributes. To be fair to the baseline methods, we set their parameters as reported as the best parameter setting in the corresponding papers.

We evaluate the ability of Multigraph2Vec++ to predict the presence of a specific type of link between any two nodes given all other links between them. We pose it as a binary classification problem. Let us assume that we wish to evaluate the performance for layer $l$. We then proceed by splitting $tr\%$ and $ts\%$ (set to be 75% and 25%, respectively in our experiments) of the edges in layer $l$ into training set ($tr_{positive}$) and test set ($ts_{positive}$), thereby obtaining positive class samples for training and testing. Similarly, we split the 'no edge/absent edges' in layer $l$ into training set ($tr_{negative}$) and test set ($ts_{negative}$), thereby obtaining negative class samples for training and testing. We then learn the node embeddings on the training set $tr = tr_{postive} \cup tr_{negative}$.

Once the node embeddings are learned, $d$ dimensional edge representation for each edge in training and test sets is obtained by averaging the corresponding node embeddings. Due to real-world networks being sparse, the training set is highly imbalanced. It contains an overwhelming proportion of negative samples (no-edge) compared to that of positive samples. We then train a logistic regression model on the training set (formed after removing class imbalance) and test the performance of the model on the test disset. We repeat the expemu 50 t20es. We used a single layer linear network along with k multiheaded attention weights for generating embeddings. and report the average AUC score (area under the ROC curve).

## 5.2 Baseline Methods

We compare Multigraph2Vec++ with single-layer methods, namely Node2Vec [7], Watch Your Step [**?** ]), multi-layer methods, namely PMNE [12] and MNE [25] and attributed network embedding method

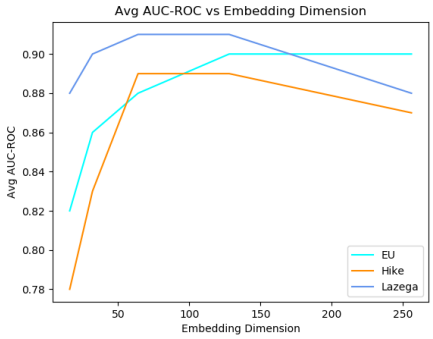


Figure 5.1: Variation in the performance of Multigraph2Vec++ w.r.t. the increasing dimension of embedding $d$ on link prediction. AUC scores averaged across layers are plotted. Number of epochs $n$ = 160 and Number of attention heads $K$ = 8.

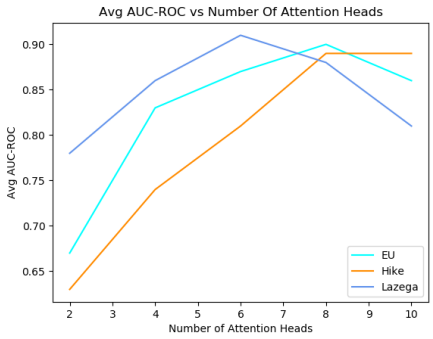


Figure 5.2: Variation in the performance of Multigraph2Vec++ w.r.t. number of attention heads $d$ on link prediction. AUC scores averaged across layers are plotted. Embedding dimension $d$ = 128 and Number of epochs $n$ = 160.

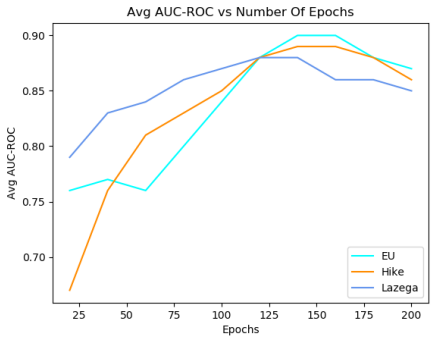


Figure 5.3: Variation in the performance of Multigraph2Vec++ w.r.t. the increasing dimension of embedding $d$ on link prediction. AUC scores averaged across layers are plotted. Embedding dimension $d$ = 128 and Number of epochs $n$ = 160

Table 5.1: Layer-wise performance (AUC) of different competing methods in the task of link prediction on three datasets. The accuracy is reported after averaging the performance across 50 iterations. The results are reported with network-property based attributes, and the dimension of the embedding vector is set to 128.

| Method | EU | | Hike | | Lazega | | |
|---|---|---|---|---|---|---|---|
| | Layer1 | Layer2 | Layer1 | Layer2 | Layer1 | Layer2 | Layer3 |
| Watch Your Step | 0.78 | 0.81 | 0.79 | 0.76 | 0.80 | 0.81 | 0.83 |
| Node2Vec | 0.74 | 0.77 | 0.73 | 0.75 | 0.79 | 0.77 | 0.81 |
| ASNE | 0.82 | 0.83 | 0.81 | 0.84 | 0.84 | 0.85 | 0.86 |
| MNE | 0.84 | 0.86 | 0.84 | 0.82 | 0.86 | 0.85 | 0.87 |
| PMNE | 0.83 | 0.81 | 0.74 | 0.76 | 0.82 | 0.81 | 0.83 |
| Multigraph2Vec++ | **0.88** | **0.92** | **0.90** | **0.88** | **0.90** | **0.92** | **0.93** |

[11].

Single-layer methods are not straightforwardly applicable for multidimensional networks and are also unable to capture multi-relation/interlayer interactions in the learned embeddings. For these methods, in order to obtain embeddings while evaluating for layer $l$, we only consider the training edges of layer $l$. Multi-layer and attributed methods (PMNE [12], MNE [25]), ASNE [11] are straightforward to apply for our setting, apart from the fact that these methods do not handle isolated nodes. To adapt these baselines for such case, we terminate the random walk if it starts from or reaches an isolated node.

## 5.3 Parameter Sensitivity

We examine how different choices of embedding dimensions, number of epochs for which the network is trained and number of attention heads affect the performance of Multigraph2Vec++ for link prediction. Figure 5.1 shows that with the increase of embedding dimension $d$, Multigraph2Vec++ performs consistently better till the embedding dimension reaches 128 however after that there is either auc-roc score remains same or it decreases a little bit. Figure 5.2 shows Multigraph2Vec++ performs best when number of attention heads is kept 8 for EU and Lazega dataset, however for Lazega which is a smaller dataset, 6 attention heads performed the best. Figure 5.3 shows that the Multigraph2Vec++performs best when it is trained between 150 to 180 epochs for the above datasets, however after that accuracy starts to decrease. Table 5.1) shows the performance of Multigraph2Vec++ based hyperparameters obtained for different dataset from above analysis.

## 5.4 Comparative Analysis

Table 5.1 shows the performance of different embedding methods on three datasets (layer-wise). As expected, multilayer embedding methods – Multigraph2Vec++, PMNE and MNE outperform single-layer embedding methods – Node2Vec and Watch Your Step(however Watch Your Step and PMNE are comparable). This can be attributed by the fact that multilayer methods can capture the useful multi-relational interactions among nodes, while single-layer networks cannot. MNE turns out to be the best baseline. However, Multigraph2Vec++ outperforms MNE by 5.88%, 7.22% and 5.81% higher AUC (relative) on EU, Hike and Lazega datasets respectively (averaged over the layers).

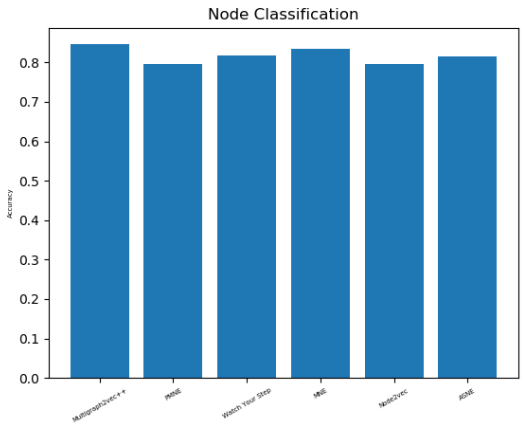


Figure 5.4: Accuracy obtained with different competing methods in multi-class node classification task on the Publication dataset. The accuracy is reported after averaging the performance across 50 iterations. The results are reported with embedding dimension = 128, number of epochs = 200 and number of attention head = 8.

## 5.5 Node Classification

In this setting, each node is assigned a label from a label set $\zeta$. Entire multi-graph is used for unsupervised feature learning i.e., for learning the embedding of each node. Once the embeddings are learned, $tr\%$ (set to be 75% in our experiment) nodes represented by their corresponding embeddings are used for training a multi-class classifier (here we used Logistic Regression classifier with one v/s rest approach). The remaining $ts\%$ (set to be 25% in our experiments) nodes are used as test set for evaluating the classifier performance. Multi-class node classification experiments are performed on the CORA dataset only.

## 5.6 Comparative Analysis

Figure 5.4 shows the performance of different embedding methods on the publication dataset. Multigraph2Vec++ with an accuracy of 84.6% is comparable with other multi-layer embedding methods MNE (accuracy of 83.4%) and PMNE (accuracy-of 79.5%); single-layer methods, Node2Vec and Watch Your Step also achieved an accuracy of 81.7% and 78.7% respectively and attribute based method ASNE achieved an accuracy of 81.5%.

## 5.7 Conclusion

In this paper, we proposed Multigraph2Vec++, a novel multigraph embedding generation method, which allows to learn supervised node embeddings using graph attention networks while preserving multi-relational interactions among nodes, in a principled fashion. We compared Multigraph2Vec++ with five state-of-the-art network embedding methods on three real-world datasets for the task of link prediction and observed significant improvement over these baselines. We also compared our method on Node Classification experiment and it was comparable with other baseline methods(even having considerable improvement over some methods).

## 5.8 Future Work

In terms of future work we will like to see what is the affect if we replace Graph Attention Network Architecture with Graph Convolutional Architecture with the same loss function.

# Bibliography


[1] Sami Abu-El-Haija, Bryan Perozzi, Rami Al-Rfou, and Alex Alemi. Watch your step: Learning graph embeddings through attention. *CoRR*, abs/1710.09599, 2017.

[2] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *CoRR*, abs/1312.6203, 2013.

[3] Shiyu Chang, Wei Han, Jiliang Tang, Guo-Jun Qi, Charu C. Aggarwal, and Thomas S. Huang. Heterogeneous network embedding via deep architectures. In *ACM SIGKDD*, pages 119–128, 2015.

[4] Hongxu Chen, Hongzhi Yin, Weiqing Wang, Hao Wang, Quoc Viet Hung Nguyen, and Xue Li. Pme: projected metric embedding on heterogeneous networks for link prediction. In *ACM SIGKDD*, pages 1177–1186. ACM, 2018.

[5] Manlio De Domenico, Albert Solé-Ribalta, Emanuele Cozzo, Mikko Kivelä, Yamir Moreno, Mason A Porter, Sergio Gómez, and Alex Arenas. Mathematical formulation of multilayer networks. *Physical Review X*, 3(4):041022, 2013.

[6] Yuxiao Dong, Nitesh V. Chawla, and Ananthram Swami. Metapath2vec: Scalable representation learning for heterogeneous networks. In *ACM SIGKDD*, pages 135–144, 2017.

[7] Aditya Grover and Jure Leskovec. Node2vec: Scalable feature learning for networks. In *ACM SIGKDD*, pages 855–864, New York, NY, USA, 2016.

[8] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. *CoRR*, abs/1706.02216, 2017.

[9] Vinayak Kumar, Aman Roy, Debdoot Mukherjee, and Tanmoy Chakraborty. Multigraph2Vec repository: code & partial data. https://tinyurl.com/y5goe7vx, 2019.

[10] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

[11] Lizi Liao, Xiangnan He, Hanwang Zhang, and Tat-Seng Chua. Attributed social network embedding. *CoRR*, abs/1705.04969, 2017.

[12] Weiyi Liu, Pin-Yu Chen, Sailung Yeung, Toyotaro Suzumura, and Lingli Chen. Principled multilayer network embedding. *ICDMW*, pages 134–141, 2017.

[13] Yao Ma, Zhaochun Ren, Ziheng Jiang, Jiliang Tang, and Dawei Yin. Multi-dimensional network embedding with hierarchical structure. In *WSDM*, pages 387–395, 2018.

[14] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[15] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

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

[17] Chuan Shi, Binbin Hu, Wayne Xin Zhao, and S Yu Philip. Heterogeneous information network embedding for recommendation. *IEEE TKDE*, 31(2):357–370, 2019.

[18] Tom AB Snijders, Philippa E Pattison, Garry L Robins, and Mark S Handcock. New specifications for exponential random graph models. *Sociological methodology*, 36(1):99–153, 2006.

[19] Lei Tang and Huan Liu. Leveraging social media networks for classification. *Data Mining and Knowledge Discovery*, 23:447–478, 2010.

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

[21] Petar Veličković, Guillem Cucurull, , Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. *CoRR*, abs/1710.10903, 2018.

[22] Lois M. Verbrugge. Multiplexity in Adult Friendships. *Social Forces*, 57(4):1286–1309, 1979.

[23] Janu Verma, Srishti Gupta, Debdoot Mukherjee, and Tanmoy Chakraborty. Heterogeneous edge embeddings for friend recommendation. *arXiv preprint arXiv:1902.03124*, 2019.

[24] Jiaxuan You, Rex Ying, Xiang Ren, William L. Hamilton, and Jure Leskovec. Graphrnn: A deep generative model for graphs. *CoRR*, abs/1802.08773, 2018.

[25] Hongming Zhang, Liwei Qiu, Lingling Yi, and Yangqiu Song. Scalable multiplex network embedding. In *IJCAI*, pages 3082–3088, 7 2018.