Highlights

MPXGAT: An Attention based Deep Learning Model for Multiplex Graphs Embedding

Marco Bongiovanni, Luca Gallo, Roberto Grasso, Alfredo Pulvirenti

- We introduce MPXGAT, an attention-based model for embedding multiplex networks

- MPXGAT is effective in predicting links across layers of real-world multiplexes

- MPXGAT shows significant improvement over state-of-the-art competitors

MPXGAT: An Attention based Deep Learning Model for Multiplex Graphs Embedding

Marco Bongiovanni[a], Luca Gallo[b], Roberto Grasso[c], Alfredo Pulvirenti[d,*]

[a]*Department of Mathematics and Computer Science, University of Catania, Via Santa Sofia, 64, Catania, 95125, Italy*
[b]*Department of Network and Data Science, Central European University, Quellenstraße 51, Vienna, 1100, Austria*
[c]*Department of Physics and Astronomy, University of Catania, Via Santa Sofia, 64, Catania, 95125, Italy*
[d]*Department of Clinical and Experimental Medicine, University of Catania, Via Santa Sofia, 64, Catania, 95125, Italy*

Abstract

Graph representation learning has rapidly emerged as a pivotal field of study. Despite its growing popularity, the majority of research has been confined to embedding single-layer graphs, which fall short in representing complex systems with multifaceted relationships. To bridge this gap, we introduce MPXGAT, an innovative attention-based deep learning model tailored to multiplex graph embedding. Leveraging the robustness of Graph Attention Networks (GATs), MPXGAT captures the structure of multiplex networks by harnessing both intra-layer and inter-layer connections. This dual exploitation facilitates accurate link prediction within and across the network's multiple layers. Our comprehensive experimental evaluation, conducted on various benchmark datasets, confirms that MPXGAT consistently outper-

---

*Corresponding author

Email addresses: `marco.bongiovanni94@gmail.com` (Marco Bongiovanni), `alfredo.pulvirenti@unict.it` (Alfredo Pulvirenti)

forms state-of-the-art competing algorithms.

1. Introduction and Related Works

In the last decades, graphs have proved to be a fundamental mathematical tool to model various real-world complex systems. From transportation systems to power grids, from the network of our social relationships to that of neurons in our brains, complex networks are all around us. Due to such ubiquity, network and graph theory have imposed themselves in many research fields, from engineering to physics, social science, and biology [1, 2, 3, 4].

A topic that has recently received considerable interest in computer science is that of how to efficiently represent large-scale graphs [5, 6, 7]. Particularly, graph embedding methods, which consist in projecting the elements of a graph, i.e., vertices, edges, and motifs, to a low-dimensional vector space by preserving some of the graph properties, have shown to be very successful in graph representation [8]. These embedding techniques are suitable for multiple applications, as they can be used in downstream learning tasks, including node classification [9], link prediction [10], and community detection [11].

We can broadly categorize graph embedding methods into traditional graph embedding and graph neural networks (GNNs) based embedding methods [8]. The first group consists of algorithms that represent graphs relying on techniques such as random walks [12, 13, 14] and matrix factorization methods. These shallow approaches have several drawbacks that limit their efficiency and effectiveness. First, they do not share any parameters between the nodes in the encoder function, which maps each node to a vector rep-

resentation, making them statistically and computationally inefficient. Second, they ignore the node attributes during the encoding process, leading to a lower quality of the embeddings. Third, they are transductive, meaning they can only generate embeddings for the nodes seen during the training phase [15].

Embedding methods based on GNNs go beyond such limitations using more sophisticated encoders accounting for the graph structure and the node attributes. The key feature of a GNN is that the embedding of a node in the graph is obtained by *aggregating* the embeddings of the node's neighbors [8]. In the last few years, a large variety of methods have been developed [6, 16, 17, 18], with the most notable examples including Graph Convolutional Networks (GCNs) [19], GraphSAGE [20] and Graph Attention networks (GATs) [16].

Although graphs are widely recognized for their versatility in modeling complex systems, traditional graph structures are limited to depicting a singular relationship type among system entities. This simplistic representation is insufficient for systems where entities engage in diverse interactions [3]. For example, social systems often involve individuals connected through various relationships such as friendship, kinship, or professional collaboration, and they may communicate through multiple channels like direct contact, phone, or online platforms [21, 22]. Similarly, in biological systems, proteins may interact genetically, physically, or through spatial proximity [23].

To encapsulate the complexity of such systems, advanced mathematical structures like multidimensional and multiplex graphs are employed [24, 25, 26]. Multidimensional graphs, or edge-labeled multigraphs, feature vertices

connected by edges of different labels, each signifying a distinct interaction type within the system. Multiplex graphs, on the other hand, consist of multiple interconnected layers, with each layer dedicated to a specific relationship type.

The key distinction between multigraphs and multiplex networks lies in the representation of system units. In multigraphs, a unit is depicted as a single node linked to others through various connection types. Multiplex networks [27, 22], however, represent units as a set of nodes distributed across different layers, connected by inter-layer links. Typically, a unit is represented in only a subset of layers, indicating connections through certain relationship types. Furthermore, it is often the case that only a portion of the inter-layer links connecting the multiple representations of the same unit is known [26].

Multi-relational network embedding is a challenging problem recently garnering significant research interest. Various methods to embed networks with multiple types of nodes and links, such as multidimensional networks and multiplex networks, have been proposed. Some of these methods are based on shallow embedding approaches [28, 29, 30, 31], while others use graph convolutional networks (GCNs) [32, 33, 34] or graph attention networks (GATs) [35]. However, none of these methods can solve the problem of predicting links between different layers of a multilayer network. This problem is important when the network structure is incomplete or heterogeneous. For example, some methods assume that all nodes are present in every layer, which is not realistic in many cases. Other methods (such as [36]; [37]; [38]) do not distinguish between intra-layer and inter-layer links, ignoring the diversity and complexity of multilayer networks. Very recently, Multi-

plexSAGE, a generalization of GraphSAGE aimed at embedding multiplex networks by relaxing these hypothesis, have been proposed [39], yet there is still a need for new methods that can address the problem of predicting inter-layer links in a general, flexible and more reliable way.

The inter-layer link prediction problem finds many crucial applications: i) linking user identities across different online social networks (OSNs), an emerging task in social media that has attracted increasing attention in recent years [40]. User identity linkage finds potential impact in different domains, from recommendation systems to cybersecurity [41]; ii) identifying the same genes or proteins across different biological networks, such as gene expression, protein interaction, metabolic pathways, etc. This can help to discover the molecular mechanisms of diseases and to find potential drug targets [42]; iii) matching the same entities across different knowledge graphs [43], which can help to enrich the semantic information and to improve query answering and reasoning capabilities. The range of possible applications of inter-layer link prediction ultimately motivates the development of embedding algorithms for multiplex networks that are able to distinguish between intra-layer and inter-layer links, and to reconstruct both connectivity patterns.

Predicting inter-layer connections in multiplex networks differs from graph matching or alignment. Graph matching [44] identifies isomorphisms between graphs, ensuring identical topology. In contrast, inter-layer prediction uncovers missing links between nodes representing the same unit across layers, with diverse connectivity. Global graph alignment [45] finds optimal correspondences between multiple graphs, considering both structure and node labels. However, inter-layer link prediction focuses on a specific bijection between

5

two graphs, i.e., layers of a multiplex network. Also, while graph alignment is general, inter-layer prediction is specific to multiplex networks.

In this paper, we introduce MPXGAT, an attention based deep learning model for multiplex graphs embedding. Our methodology, based on GATs, consists in embedding the nodes of a multiplex network by leveraging the information about their intra-layer and inter-layer connections, allowing for link prediction tasks both within the same layer and across different layers. We carry out a thorough experimental analysis on three benchmark datasets, showing that MPXGAT out-performs state-of-the-art competing algorithms. We conclude with an in-depth study of the model main features, proving how their use positively impacts the performance of the algorithm itself.

2. Methods

In this section, we provide details of our model, *MPXGAT*, which can embed multiplex networks, namely networks where multiple types of link exist. We first provide some preliminary notions, next, we introduce the mathematical formulation of the model, and finally, we describe the algorithmic implementation.

2.1. Preliminary Notions

We start by defining the basic concepts of simple and multiplex graphs A graph (network) is a mathematical structure consisting of a set of vertices (nodes) and a set of edges (links), where each edge represents a relation between a pair of vertices. Simple graphs are able to capture only one kind of relationship, so they are limited when it comes to characterizing systems where multiple types of interactions coexist. The need of a higher degree of

6

expressiveness motivates the use of multiplex graphs. We can define multiplex graphs as a graphs composed by two different sub-networks, which we refer to as horizontal and vertical. The horizontal network consists of a collection of simple graphs, called (horizontal) layers. In this setup, each unit of a system can be represented as a node in one or more layers, with each layer referring to a certain type of relationship. We refer to the connections within a given layer as intra-layer links. The second sub-network, namely the vertical network, consists of a single-layer graph formed by the set of edges connecting nodes across different layers. We assume that a node i on a layer α can be connected to at most one node j on another layer β, i.e., the two nodes represent the same unit of the system. Also, we assume that if node i on layer α is connected to node j on layer β, and if node j is connected to node k on layer γ, then nodes i and k are also connected. Under these hypothesis, the vertical network consists in a collection of connected components, which can be either cliques or isolated nodes. The edges in the vertical network will be referred to as inter-layer links.

Formally, a multiplex graph is a set $\mathbb{V}$ of N nodes that are connected through L different layers. We assume that N_α nodes are present in each layer $\alpha \in \{0, 1, \ldots, L\}$, such that $N_1 + \cdots + N_L = N$. Each layer α is a graph $\mathcal{G}_\alpha = (\mathbb{V}_\alpha, \mathcal{E}_\alpha)$ where $\mathbb{V}_\alpha \in \mathbb{V}$ is the set of the N_α nodes and $\mathcal{E}_\alpha$ is the set of edges connecting them. The node sets for each layer are disjoint, i.e., $\mathbb{V}_\alpha \cap \mathbb{V}_\beta = \emptyset$ for $\alpha \neq \beta$, and their union makes the set of all nodes $\mathbb{V}$. The set of intra-layer links for the horizontal network can be defined as $\mathcal{E}_{\text{intra}} = \bigcup_{\alpha=1}^{L} \mathcal{E}_\alpha$. Hence, we define the horizontal network as $\mathcal{G}_H = (\mathbb{V}, \mathcal{E}_{\text{intra}})$. The vertical network can be instead defined as $\mathcal{G}_V = (\mathbb{V}, \mathcal{E}_{\text{inter}})$, where $\mathcal{E}_{\text{inter}} =$

$\{(i, j) \in \mathbb{V} \times \mathbb{V} \mid \exists \alpha, \beta \in \{0, 1, \dots, L\} \ s.t. \ (i, j) \in \mathcal{E}_\alpha \times \mathcal{E}_\beta\}$ is the set of inter-layer edges. Figure 1 shows an illustration of a multiplex graph.
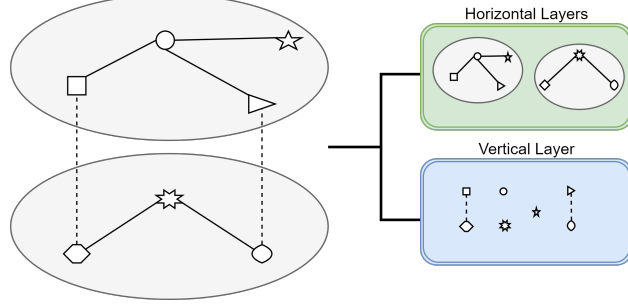

Figure 1: A toy example of a multiplex network with 2 horizontal layers. The solid edges represent the intra-layer connections while the dashed edges are the inter-layer edges.

2.2. The MPXGAT General Framework

The main idea of MPXGAT is to generate two separate embeddings for each node in two different phases. In the first phase, each node is embedded according to the horizontal layers, where it has multiple types of relation with other nodes. In the second phase, nodes are embedded according to the vertical network, where they are linked to their counterparts on different layers. In the following, we will use the superscript $\cdot^H$ when we refer to the horizontal network, while the superscript $\cdot^V$ will refer to the vertical one. The notation used, together with the most relevant variables of our model, are reported in Table 1.

Our algorithm uses two sub-models, one for each embedding phase: MPXGAT-H and MPXGAT-V (an illustration of the model structure is shown in Figure 2). In the most general framework, MPXGAT-H applies a series of GAT
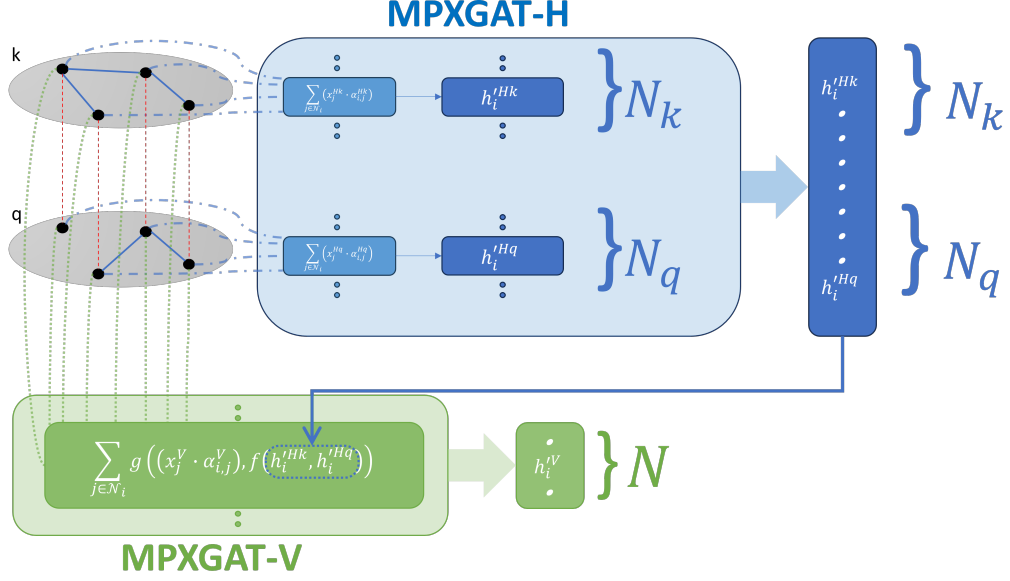
Figure 2: The structure of the MPXGAT model. In this toy example, it is applied on a multiplex network with 2 horizontal layers where the solid blue edges represent the intra-layer connections while the dashed red edges are the inter-layer edges. The data is provided to the *MPXGAT-H* throughout the dot-and-dash blue lines. Once processed these are used to feed the *MPXGAT-V* together with the inter-layer links (the dotted green lines). The output of the model consists of both horizontal and vertical nodes embedding.

convolutional layers independently to each layer of the horizontal network

Mathematically, the embedding is described by the following equations:

$$e_{i,j}^{H_k} = \text{LeakyReLU}\left(\left[\boldsymbol{W}^{H_k} \cdot \boldsymbol{h}_i^{H_k} \cdot (\boldsymbol{v}^{H_k})^T || \boldsymbol{W}^{H_k} \cdot \boldsymbol{h}_j^{H_k} \cdot (\boldsymbol{v}^{H_k})^T\right]\right) \quad (1a)$$

$$\alpha_{i,j}^{H_k} = \text{softmax}(e_{i,j}^{H_k}) = \frac{\exp(e_{i,j}^{H_k})}{\sum_{z \in \mathcal{N}_i^{H_k}} \exp(e_{i,z}^{H_k})} \quad (1b)$$

$$\boldsymbol{h}_i'^{H_k} = \sigma\left(\sum_{j \in \mathcal{N}_i^{H_k}} \alpha_{i,j}^{H_k} \cdot \boldsymbol{W}^{H_k} \cdot \boldsymbol{h}_j^{H_k}\right) \quad (1c)$$

Equations 1 govern the convolutional layer as described in the GAT model [16]. In particular, in Equation 1c, $\boldsymbol{h}_i'^{H_k}$ represents the horizontal embedding of the node i that belongs to the layer k, while Equation 1b describes the self attention mechanism in layer k.

Leveraging the horizontal embeddings, MPXGAT-V generates the vertical embeddings by making use of a custom mechanism inspired by the one applied in the GAT model, which we call GAT-V. The equations to compute the vertical embedding of node i are the following:

$$e_{i,j}^{V^{k,q}} = \text{LeakyReLU}\left(\left[\boldsymbol{W}^V \cdot \boldsymbol{h}_i^V \cdot (\boldsymbol{v}^V)^T || \boldsymbol{W}^V \cdot \boldsymbol{h}_j^V \cdot (\boldsymbol{v}^V)^T\right]\right) \quad (2a)$$

$$\alpha_{i,j}^{V^{k,q}} = \text{softmax}(e_{i,j}^{V^{k,q}}) = \frac{\exp(e_{i,j}^{V^{k,q}})}{\sum_{z \in \mathcal{N}_i^V} \exp(e_{i,z}^V)} \quad (2b)$$

$$\boldsymbol{h}_i'^V = \sigma\left(\sum_{j \in \mathcal{N}_i^V} g\left(\left[\alpha_{i,j}^V \cdot \boldsymbol{W}^V \cdot \boldsymbol{h}_j^V\right], f(\boldsymbol{h}_i^{H_k}, \boldsymbol{h}_j^{H_q})\right)\right) \quad (2c)$$

Equations 2 contain two novel elements. The first is the function f, which transforms the final horizontal embeddings to the same dimensional space of the vertical embeddings. Such a function can be a simple linear transformation of the node horizontal embeddings, or it can be more complex, depending on the specific task considered. The second feature we introduce

10

is the function g, which combines the data obtained from the current vertical embedding with the results of the the function f.

2.3. MPXGAT Implementation for Link Prediction

We now present an implementation of *MPXGAT* that is suitable for link prediction. We customize the attention mechanisms of both *MPXGAT-H* and *MPXGAT-V*, which are described in Equations 1a, 1b and 2a, 2b, respectively. In particular:

- in Equations 1a and 2a, instead of a single weight matrix for both i and j nodes, i.e., $\boldsymbol{W}^H$, two different weight matrices, $\left(\boldsymbol{W}_i^H, \boldsymbol{W}_j^H\right)$, are used. By doing that, we improve the representation capabilities of the model, at the expense of a greater number of parameters. The same choice is done for the weight vector, where instead of a single vector $\boldsymbol{v}^H$, we use two, $\left(\boldsymbol{v}_i^H, \boldsymbol{v}_j^H\right)$. Moreover, to improve the generalization capabilities of the model, the concatenation operator $\|$ is implemented as a sum $+$, and both parts of this operation are augmented with the introduction of a bias vector;

- in Equations 1b and 2b an additional dropout mechanism is applied to the attention coefficients. In this way, we sub-sample the paths present in the graph, granting the model a better generalization capability.

A further addition is the use of *multiple attention heads*, consisting in calculating different embeddings of the same nodes applying multiple times the convolutional layer. This generates multiple node representations that highlight different kinds of patterns about the nodes, as each attention head can be seen as a distinct information channel representing a certain aspect of

11

nodes. The implemented sub-models concatenates the embeddings for each attention head in the hidden convolutional layers of the model, and uses a mean operation for the final one.

Finally, for both sub-models the activation function σ in Equations 1c and 2c we consider the *LeakyReLU*.

The general *MPXGAT* framework rely on the f and g functions for the embedding, which can be specified according to the particular task we want to solve. Here, the two functions f and g introduced in Equation 2c have the following characteristics:

- f performs a linear transformation of the horizontal embedding of the source node, $\boldsymbol{h}_i^H$, to the same dimensional space of its vertical embedding, $\boldsymbol{h}_i^V$, here described with $\boldsymbol{x}_i^H$. This is done through an additional weight matrix $\boldsymbol{Z}^H$. The result is then multiplied by a vector of parameters $\boldsymbol{v}_i^H$, whose role is to enhance the patterns from the horizontal embeddings. Finally, the LeakyReLU function is applied.

- The function g performs a weighted sum of the output of the function f (see Equation 3c) and the data computed with the convolutional layer for the vertical network (see Equation 3d). A scalar parameter β, automatically inferred during the learning process, is introduced to tune the information coming from the horizontal and vertical embeddings, respectively. In particular when $\beta = 0$ the horizontal embedding is not considered, while for $\beta = 0.5$ both the components are considered with the same weight.

Formally, the equations describing the implementation of *MPXGAT*

for link predictions are:

$$\boldsymbol{x}_i^H = \boldsymbol{h}_i^H \cdot \boldsymbol{Z}^H + \boldsymbol{b}_{h_i} \tag{3a}$$

$$\alpha_i^H = \text{LeakyReLU} \left(\ \boldsymbol{x}_i^H \cdot \boldsymbol{v}_i^H \right) \tag{3b}$$

$$\boldsymbol{m}_i^H = f(\boldsymbol{x}_i^H) = \alpha_i^H * \boldsymbol{x}_i^H \tag{3c}$$

$$\boldsymbol{m}_{i,j}^V = \alpha_{i,j}^V \cdot \boldsymbol{x}_j^V \tag{3d}$$

$$g\left(\boldsymbol{m}^V, \boldsymbol{m}^H\right) = \left(\boldsymbol{m}_{i,j}^V * \left(1 - \text{ReLU}\left(\beta\right)\right)\right) + \left(\left(\boldsymbol{m}_i^H\right) * \text{ReLU}\left(\beta\right)\right) \tag{3e}$$

A few aspects of our model are worth remarking. First, we assume neither to have the same number of nodes in each horizontal layer nor to have all possible inter-layer links, making our algorithm a valuable tool for reconstructing inter-layer connectivity. Second, *MPXGAT* is quite flexible, as it supports the use of node features and edge weights. In particular, different features can be used by the two sub-models, since the horizontal and the vertical networks are defined and processed as separated entities. As we will show in the next section, these characteristics guarantee our model good performances even in datasets where the edge density in the horizontal or vertical networks is high, which represents a strong limit of algorithms based on GraphSAGE [39].

3. Results

In this section, we present the results obtained by *MPXGAT* in predicting both intra-layer and inter-layer links. We first present the datasets studied, as well as the experimental setup used for the analysis, and the competing methods considered as benchmarks for our algorithm. Then, we test the

13

performances of *MPXGAT* in the link prediction task. We round up the analysis by assessing the impact of the horizontal and vertical sub-models on the embedding procedure.

3.1. Dataset

To assess the performance of *MPXGAT* in predicting both intra-layer and inter-layer connections, we employ the same datasets that have been originally used to test *MultiplexSAGE* [39]. These include data relative to three types of real-world multiplex networks, namely a collaboration, a biological, and an online social networks. Specifically, we considered the following datasets.

arXiv [46]. The arXiv multiplex network represents collaborations in various research topics published on the pre-print archive. Each layer of the network corresponds to a different research category or theme. The network was obtained by selecting papers published before May 2014 that contain the keyword *networks* in their titles or abstracts.

Drosophila [47]. This multiplex network represents the interactions between proteins and genes in the common fruit fly, i.e., Drosophila melanogaster, with each layer corresponding to interactions of various types. The dataset was collected from the Biological General Repository for Interaction Datasets (BioGRID), with data updated until January 2014.

ff-tt-yt [48]. This multiplex network is derived from Friendfeed (ff), a social media aggregation platform where users can link their accounts from various online social networks (OSNs). The network comprises users who have registered a single Twitter (tw) account and a sole YouTube (yt) account on Friendfeed. Additionally, the Twitter and YouTube accounts are linked

to a single Friendfeed account.

For each empirical dataset, we consider exclusively the largest connected component of the multiplex networks and we treat all networks as undirected and unweighted. Details about the largest connected component for each of these datasets are provided in Table 2.

We remark that, for all datasets considered in our analysis, nodes are not provided with any external features. Hence, we associate to each node, n, a one-hot encoding vector defined by the Kronecker function, $\delta_{i,n}$.

3.2. Experimental setup

To assess the performance of *MPXGAT*, we consider the experimental setup delineated in [39]. We partition the data following a multiple step procedure. First, we randomly select 20% of the network nodes, labeling them as *marked nodes*. We then define test and training sets. Both sets encompass positive and negative examples, the former corresponding to actual links within the network, while the latter consisting in pairs of unconnected nodes. In the test set, positive examples comprise a subset of 20% of the intra-layer links and all inter-layer links among the marked nodes. Conversely, positive examples in the training set include all remaining intra-layer and inter-layer links in the multiplex network. As negative examples within the test set, we include 20% of all possible negative intra-layer links among the marked nodes, as well as all possible inter-layer links between them. The negative examples in the training set correspond to the remaining pairs of unconnected nodes.

To test the performance of our algorithm, instead of conducting a single experiment for each dataset, we repeat the embedding procedure multiple

times. For each repetition, we randomly select a subset of marked nodes, defining the training and test sets accordingly. To obtain the best parameter settings, we apply a grid search method.

3.3. Competing Methods

We conduct a comparative analysis of *MPXGAT* against three competing methodologies, specifically *GraphSAGE*, *GATNE* and *MultiplexSAGE*.

GraphSAGE [20]. *GraphSAGE* is an inductive node embedding algorithm that leverages node features to learn an embedding function capable of generalizing to unseen nodes. It was originally designed for single-layer network embeddings, so in our experiments, we apply it without distinguishing between intra-layer and inter-layer links.

GATNE [35]. *GATNE* is an embedding algorithm for attributed heterogeneous networks, encompassing multigraphs with diverse node and edge types. To adapt *GATNE* to our specific task, we introduced two categories of edges, denoted as intra-layer and inter-layer links. This adjustment allows us to create a multigraph that the algorithm can learn to embed effectively.

MultiplexSAGE [39]. *MultiplexSAGE* represents an extension of the GraphSAGE algorithm, specifically tailored for embedding multiplex networks. Its key feature is the distinction made between inter-layer and intra-layer links that allows for the prediction of both intra-layer and inter-layer connectivity patterns.

Other relevant methods such as [31], [33], [28], [49], [29], [30] and [34] can not be used as competing methods, as they assume that each layer has the same number of nodes and that all inter-layer connections are known. Therefore, as we are interested in predicting both intra-layer and inter-layer links,

these methods are not suitable benchmarks for evaluating the performance of *MPXGAT*.

3.4. Embedding Multiplex Networks

Our first analysis concerns the prediction of both intra-layer and inter-layer links when embedding a multiplex network. For each embedding, we evaluate the Area Under the Receiveing Operating Characteristic (ROC) Curve (AUC), and take an average over the different repetitions as a performance metric. We consider the standard deviation as an indicator of statistical error. Table 3 provides an overview of the performances obtained with *MPXGAT*, *GraphSAGE*, *GATNE*, and *MultiplexSAGE*. We note that *GATNE* outperforms *GraphSAGE* and *MultiplexSAGE* in intra-layer link prediction, having a higher average AUC for all three datasets. However, *MPXGAT* has comparable performances with *GATNE* on the ff-ww-tt and the Drosophila dataset, while the latter performs better on the arXiv dataset. Yet, with regards to inter-layer link prediction, *MPXGAT* clearly performs better than all other algorithms, including *MultiplexSAGE*, which is explicitly designed for that task. Overall, if we consider the general performance of the methods without distinguishing intra- and inter-layer connections, our algorithm emerges as the best solution, as reported in Table 4.

As described in Section 2, *MPXGAT* employs two distinct embeddings, dealing with the horizontal, i.e., intra-layer, and vertical, i.e., inter-layer, embeddings, respectively. In contrast, the other models rely on a single embedding that serves both tasks. As we will further investigate in the next section, it is this architectural difference that allows *MPXGAT* to predict with a certain reliability both intra-layer and inter-layer links.

3.5. Measure the impact of Horizontal Embeddings

We now establish the impact of the horizontal embeddings on the performance of *MPXGAT*. To do so, we conduct two experiments. In the first, we perform the embedding with *MPXGAT*, but instead of using MPXGAT-V, which leverages the horizontal embeddings generated by MPXGAT-H, for embedding the vertical network, we consider a standard GAT [16], thus ignoring the contribution of MPXGAT-H. Table 5 reports the average AUC for the inter-layer link prediction in both configurations. We observe that neglecting the horizontal embeddings leads to worse performances across all datasets, suggesting that including the MPXGAT-V submodel increases the algorithm adaptability and predictive power.

To validate the significance of these findings we performed a Welch's T-test. The p-values are 6.00×10^{-7}, 9.10×10^{-10}, and 1.10×10^{-7} for the Drosophila, arXiv, and ff-tw-yt datasets, respectively, confirming the statistical significance of our finding.

In the second experiment, we keep the original model architecture but instead of providing MPXGAT-V with the horizontal embeddings generated by MPXGAT-H, we replace them with random embeddings, i.e., vectors whose components are random values.

Table 6 shows the results of the comparison with regards to the inter-layer link prediction. We note that for both the ff-tw-yt and the arXiv dataset, using the horizontal embeddings increases the performance of the prediction task, while we observe similar values of the average AUC for the Drosophila dataset. The statistical significance of this result is confirmed by the Welch's T-test, for which we obtain p-values equal to $6.10 \cdot 10^{-6}$, 0.75, $3.60 \cdot 10^{-4}$ for

the arXiv, Drosophila, and ff-ww-tt datasets, respectively.

We conjecture that this outcome is due to the structure of the multiplex network, both in terms of the size of the different layers, i.e., the number of nodes laying on them, and of the connectivity patterns, both within the same layer (e.g., randomness, clustering and community organization, and across different ones, i.e., overlapping).

4. Conclusions

In this paper we have introduced *MPXGAT*, an attention based deep learning model for the embedding of multiplex graphs. Through a comprehensive experimental analysis we showed that MPXGAT out-performs state-of-the-art competing algorithms. Future work will be aimed at better understanding how the community structure withing each layer influences the performances of the algorithm in terms of the reliability of predicted inter-layer relations. An implementation of *MPXGAT* in Python is available at https://github.com/MarcoB46/MPXGAT.

References

[1] M. E. Newman, The structure and function of complex networks, SIAM review 45 (2) (2003) 167–256.

[2] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, D.-U. Hwang, Complex networks: Structure and dynamics, Physics reports 424 (4-5) (2006) 175–308.

[3] V. Latora, V. Nicosia, G. Russo, Complex networks: principles, methods and applications, Cambridge University Press, 2017.

[4] M. Newman, Networks, Oxford university press, 2018.

[5] D. Bacciu, F. Errica, A. Micheli, M. Podda, A gentle introduction to deep learning for graphs, Neural Networks 129 (2020) 203–221.

[6] F. Chen, Y.-C. Wang, B. Wang, C.-C. J. Kuo, Graph representation learning: a survey, APSIPA Transactions on Signal and Information Processing 9 (2020) e15.

[7] M. Grohe, word2vec, node2vec, graph2vec, x2vec: Towards a theory of vector embeddings of structured data, in: Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, 2020, pp. 1–16.

[8] S. Khoshraftar, A. An, A survey on graph representation learning methods, arXiv preprint arXiv:2204.01855 (2022).

[9] Y. Rong, W. Huang, T. Xu, J. Huang, Dropedge: Towards deep graph convolutional networks on node classification, arXiv preprint arXiv:1907.10903 (2019).

[10] L. Lü, T. Zhou, Link prediction in complex networks: A survey, Physica A: statistical mechanics and its applications 390 (6) (2011) 1150–1170.

[11] S. Fortunato, Community detection in graphs, Physics reports 486 (3-5) (2010) 75–174.

[12] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: Online learning of social representations, in: Proceedings of the 20th ACM SIGKDD interna-

tional conference on Knowledge discovery and data mining, 2014, pp. 701–710.

[13] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, Q. Mei, Line: Large-scale information network embedding, in: Proceedings of the 24th international conference on world wide web, 2015, pp. 1067–1077.

[14] A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, 2016, pp. 855–864.

[15] W. L. Hamilton, R. Ying, J. Leskovec, Representation learning on graphs: Methods and applications, arXiv preprint arXiv:1709.05584 (2017).

[16] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, arXiv preprint arXiv:1710.10903 (2017).

[17] M. Liu, H. Gao, S. Ji, Towards deeper graph neural networks, in: Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining, 2020, pp. 338–348.

[18] H. Gao, S. Ji, Graph u-nets, in: international conference on machine learning, PMLR, 2019, pp. 2083–2092.

[19] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, arXiv preprint arXiv:1609.02907 (2016).

[20] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning

on large graphs, Advances in neural information processing systems 30 (2017).

[21] M. Szell, R. Lambiotte, S. Thurner, Multirelational organization of large-scale social networks in an online world, Proceedings of the National Academy of Sciences 107 (31) (2010) 13636–13641.

[22] F. Battiston, V. Nicosia, V. Latora, Structural measures for multiplex networks, Physical Review E 89 (3) (2014) 032804.

[23] M. De Domenico, V. Nicosia, A. Arenas, V. Latora, Structural reducibility of multilayer networks, Nature communications 6 (1) (2015) 6864.

[24] S. Boccaletti, G. Bianconi, R. Criado, C. I. Del Genio, J. Gómez-Gardenes, M. Romance, I. Sendina-Nadal, Z. Wang, M. Zanin, The structure and dynamics of multilayer networks, Physics reports 544 (1) (2014) 1–122.

[25] M. Berlingerio, M. Coscia, F. Giannotti, A. Monreale, D. Pedreschi, Foundations of multidimensional network analysis, in: 2011 international conference on advances in social networks analysis and mining, IEEE, 2011, pp. 485–489.

[26] E. Cozzo, G. F. De Arruda, F. A. Rodrigues, Y. Moreno, Multiplex networks: basic formalism and structural properties, Vol. 10, Springer, 2018.

[27] M. De Domenico, A. Solé-Ribalta, E. Cozzo, M. Kivelä, Y. Moreno, M. A. Porter, S. Gómez, A. Arenas, Mathematical formulation of multilayer networks, Physical Review X 3 (4) (2013) 041022.

[28] W. Liu, P.-Y. Chen, S. Yeung, T. Suzumura, L. Chen, Principled multilayer network embedding, in: 2017 IEEE International Conference on Data Mining Workshops (ICDMW), IEEE, 2017, pp. 134–141.

[29] H. Zhang, L. Qiu, L. Yi, Y. Song, Scalable multiplex network embedding., in: IJCAI, Vol. 18, 2018, pp. 3082–3088.

[30] Y. Shi, F. Han, X. He, X. He, C. Yang, J. Luo, J. Han, mvn2vec: Preservation and collaboration in multi-view network embedding, arXiv preprint arXiv:1801.06597 (2018).

[31] M. Gong, W. Liu, Y. Xie, Z. Tang, M. Xu, Heuristic 3d interactive walks for multilayer network embedding, IEEE Transactions on Knowledge and Data Engineering 34 (7) (2020) 3309–3321.

[32] C. Yang, A. Pal, A. Zhai, N. Pancha, J. Han, C. Rosenberg, J. Leskovec, Multisage: Empowering gcn with contextualized multi-embeddings on web-scale multipartite networks, in: Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining, 2020, pp. 2434–2443.

[33] V. N. Ioannidis, A. G. Marques, G. B. Giannakis, Tensor graph convolutional networks for multi-relational and robust learning, IEEE Transactions on Signal Processing 68 (2020) 6535–6546.

[34] Z. Huang, X. Li, Y. Ye, M. K. Ng, Mr-gcn: Multi-relational graph convolutional networks based on generalized tensor product., in: IJCAI, Vol. 20, 2020, pp. 1258–1264.

[35] Y. Cen, X. Zou, J. Zhang, H. Yang, J. Zhou, J. Tang, Representation learning for attributed multiplex heterogeneous network, in: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, 2019, pp. 1358–1368.

[36] A. Behrouz, M. Seltzer, Anomaly detection in multiplex dynamic networks: from blockchain security to brain disease prediction (2022). arXiv:2211.08378.

[37] A. Behrouz, F. Hashemi, Cs-mlgcn: Multiplex graph convolutional networks for community search in multiplex networks, in: Proceedings of the 31st ACM International Conference on Information & Knowledge Management, CIKM '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 3828–3832. doi:10.1145/3511808.3557572. URL https://doi.org/10.1145/3511808.3557572

[38] M. Qu, J. Tang, J. Shang, X. Ren, M. Zhang, J. Han, An attention-based collaboration framework for multi-view network representation learning (2017). arXiv:1709.06636.

[39] L. Gallo, V. Latora, A. Pulvirenti, Multiplexsage: A multiplex embedding algorithm for inter-layer link prediction, IEEE Transactions on Neural Networks and Learning Systems (2023).

[40] K. Shu, S. Wang, J. Tang, R. Zafarani, H. Liu, User identity linkage across online social networks: A review, Acm Sigkdd Explorations Newsletter 18 (2) (2017) 5–17.

[41] R. Tang, S. Jiang, X. Chen, H. Wang, W. Wang, W. Wang, Interlayer link prediction in multiplex social networks: An iterative degree penalty algorithm, Knowledge-Based Systems 194 (2020) 105598.

[42] P. B. Jain, T. T. Nguyen, J. Mináč, L. E. Muller, R. C. Budzinski, Composed solutions of synchronized patterns in multiplex networks of kuramoto oscillators, Chaos: An Interdisciplinary Journal of Nonlinear Science 33 (10) (2023).

[43] M. Azmy, P. Shi, J. Lin, I. F. Ilyas, Matching entities across different knowledge graphs with graph embeddings, arXiv preprint arXiv:1903.06607 (2019).

[44] R. Mathon, A note on the graph isomorphism counting problem, Information Processing Letters 8 (3) (1979) 131–136.

[45] C.-Y. Ma, C.-S. Liao, A review of protein–protein interaction network alignment: From pathway comparison to global alignment, Computational and Structural Biotechnology Journal 18 (2020) 2647–2656.

[46] M. De Domenico, A. Lancichinetti, A. Arenas, M. Rosvall, Identifying modular flows on multilayer networks reveals highly overlapping organization in interconnected systems, Phys. Rev. X 5 (2015) 011027. `doi:10.1103/PhysRevX.5.011027`.
URL `https://link.aps.org/doi/10.1103/PhysRevX.5.011027`

[47] C. Stark, B.-J. Breitkreutz, T. Reguly, L. Boucher, A. Breitkreutz, M. Tyers, Biogrid: a general repository for interaction datasets, Nu-

cleic Acids Research 34 (Database issue) (2006) D535–D539. doi: `10.1093/nar/gkj109`.

[48] M. Dickison, M. Magnani, L. Rossi, Multilayer Social Networks, Cambridge University Press, 2016.

[49] M. Qu, J. Tang, J. Shang, X. Ren, M. Zhang, J. Han, An attention-based collaboration framework for multi-view network representation learning, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM '17, Association for Computing Machinery, New York, NY, USA, 2017, p. 1767–1776. doi: `10.1145/3132847.3133021`.
URL `https://doi.org/10.1145/3132847.3133021`

Table 1: Description of the variables involved in this work. Note that superscripts $\cdot^H$ and $\cdot^V$ are omitted for the sake of readability

VARIABLE	DESCRIPTION
$N \in \mathbb{N}$	Number of nodes in the multiplex graph
$\mathbb{V}$	Set of nodes in the multiplex graph
$L \in \mathbb{N}$	Total number of horizontal layers of the multiplex graph
$\alpha \in \mathbb{N}$	Index of the horizontal layer in the multiplex graph
N_α	Number of nodes within the horizontal layer of index α
$\mathbb{V}_\alpha$	Set of nodes within the horizontal layer of index α
$\mathcal{E}_\alpha$	Set of edges within the horizontal layer α
$\mathcal{G}_\alpha$	Layer of index α in the horizontal network
$\mathcal{E}_{\text{intra}}$	Set of intra-layer edges for the horizontal network
$\mathcal{E}_{\text{inter}}$	Set of inter-layer edges for the horizontal network
$\mathcal{G}_H$	Horizontal network of the multiplex graph
$\mathcal{G}_V$	Vertical network of the multiplex graph
$F \in \mathbb{N}$	Initial node feature dimensionality
$F' \in \mathbb{N}$	Final node feature dimensionality
$i \in \mathbb{N}$	Source node
$j \in \mathbb{N}$	Destination node
$k \in \mathbb{N}$	Source node horizontal layer's index
$q \in \mathbb{N}$	Destination node horizontal layer's index
$\mathcal{N}_i$	Set of nodes connected to node i
$\boldsymbol{h}_i \in \mathbb{R}^F$	Embedding of node i
$\boldsymbol{v} \in \mathbb{R}^{F'}$	Attention weight vector
$\boldsymbol{W} \in \mathbb{R}^{F' \times F}$	Weight matrix, used to linearly transform the node embeddings
$e_{i,j} \in \mathbb{R}$	Attention coefficient between node i and j
$\alpha_{i,j} \in \mathbb{R}$	Normalized attention coefficient between nodes i and j
$\boldsymbol{h}_i \in \mathbb{R}^{F'}$	Updated embedding of node i after a forward pass
σ	Activation function (in our case LeakyReLU)
f	Function used to transform the horizontal embeddings to the same dimensional space of the vertical embedding
g	Function used to combine the data obtained from the vertical network with the result of the application of the function f

Table 2: Dataset Information including the number of nodes, edges, and average degree for the largest connected component of each dataset (arXiv, Drosophila, and ff-tw-yt)

Dataset	Nodes	Edges	Avg. Degree
arXiv	19,310	20,738	1.07
Drosophila	11,867	5,171	0.44
ff-tw-yt	11,827	6,028	0.51

Table 3: Performance comparison on intra-layer and inter-layer link prediction across the three distinct datasets: ff-tw-yt, Drosophila, and arXiv. The assessment is based on the AUC metric, using the standard deviation as error metric. The best performing tool is highlighted in boldface

Algorithm	ff-tw-yt		Drosophila		arXiv	
	intra	*inter*	*intra*	*inter*	*intra*	*inter*
GraphSAGE	0.47 ($\pm$ 0.02)	0.56 ($\pm$ 0.02)	0.54 ($\pm$ 0.02)	0.63 ($\pm$ 0.02)	0.72 ($\pm$ 0.02)	0.70 ($\pm$ 0.01)
GATNE	**0.83 ($\pm$ 0.01)**	0.47 ($\pm$ 0.01)	**0.78 ($\pm$ 0.01)**	0.55 ($\pm$ 0.01)	**0.91 ($\pm$ 0.01)**	0.63 ($\pm$ 0.01)
MultiplexSAGE	0.48 ($\pm$ 0.02)	0.62 ($\pm$ 0.02)	0.51 ($\pm$ 0.01)	0.77 ($\pm$ 0.02)	0.71 ($\pm$ 0.02)	0.83 ($\pm$ 0.01)
MPXGAT	0.76 ($\pm$ 0.06)	**0.83 ($\pm$ 0.01)**	0.76 ($\pm$ 0.05)	**0.86 ($\pm$ 0.02)**	0.80 ($\pm$ 0.02)	**0.84 ($\pm$ 0.01)**

Table 4: Overall AUC performaces calculated as a weighted sums of the results shown in Table 3, based on the number of edges used to evaluate the models.

Algorithm	ff-tw-yt	Drosophila	arXiv
GraphSAGE	0.49 ($\pm$ 0.02)	0.57 ($\pm$ 0.01)	0.70 ($\pm$ 0.01)
GATNE	0.72 ($\pm$ 0.01)	0.69 ($\pm$ 0.02)	0.72 ($\pm$ 0.01)
MultiplexSAGE	0.52 ($\pm$ 0.02)	0.61 ($\pm$ 0.01)	0.79 ($\pm$ 0.01)
MPXGAT	**0.78 ($\pm$ 0.03)**	**0.80 ($\pm$ 0.03)**	**0.82 ($\pm$ 0.04)**

Table 5: Results of the ongoing experiment aimed at assessing the impact of excluding horizontal embeddings on inter-layer link prediction. The performance is evaluated in terms of AUC across the three datasets. The second model, which omits horizontal embeddings, exhibits lower performance in inter-layer link prediction across all datasets.

Algorithm	ff-tw-yt	Drosophila	arXiv
MPXGAT-V (layer GAT-V*)*	**0.83 ± (0.01)**	**0.86 ± (0.01)**	**0.84 ± (0.01)**
GAT (layer GAT*)*	0.72 ± (0.02)	0.78 ± (0.02)	0.78 ± (0.01)

Table 6: Results of the experiment investigating the impact of replacing meaningful horizontal embeddings with random embeddings. The performance is evaluated in terms of AUC. The model with random embeddings performs worse in two out of the three datasets compared to the model with actual embeddings, indicating a significant decrease in predictive accuracy.

Algorithm	ff-tw-yt	Drosophila	arXiv
MPXGAT-V (actual embedding)	**0.83 ± (0.01)**	0.86 ± (0.01)	**0.84 ± (0.01)**
MPXGAT-V (random embedding)	0.80 ± (0.02)	0.86 ± (0.01)	0.81 ± (0.01)