Representation learning in multiplex graphs: Where and how to fuse information?

Piotr Bielak[1 [0000−0002−1487−2569]] and Tomasz Kajdanowicz[1 [0000−0002−8417−1012]]

Department of Artificial Intelligence,
Wroclaw University of Science and Technology,
Wrocław, Poland
piotr.bielak@pwr.edu.pl

Abstract. In recent years, unsupervised and self-supervised graph representation learning has gained popularity in the research community. However, most proposed methods are focused on homogeneous networks, whereas real-world graphs often contain multiple node and edge types. Multiplex graphs, a special type of heterogeneous graphs, possess richer information, provide better modeling capabilities and integrate more detailed data from potentially different sources. The diverse edge types in multiplex graphs provide more context and insights into the underlying processes of representation learning. In this paper, we tackle the problem of learning representations for nodes in multiplex networks in an unsupervised or self-supervised manner. To that end, we explore diverse information fusion schemes performed at different levels of the graph processing pipeline. The detailed analysis and experimental evaluation of various scenarios inspired us to propose improvements in how to construct GNN architectures that deal with multiplex graphs.

Keywords: representation learning · graph neural networks · multiplex graphs · unsupervised learning · self-supervised learning.

1 Introduction

Real-world data often exhibits a complex and heterogeneous nature, leading to challenges and opportunities in graph representation learning. From social networks, where multiple relationship types exist among users, to biological networks, where various interaction types occur among molecules, a multiplex view is often more appropriate to capture the entirety of the network structure. Multiplex networks are characterized by the presence of multiple edge types (graph layers), span across a common set of nodes. They provide richer information, better modeling capabilities, and facilitate the integration of more detailed data from diverse sources. A multiplex network is not simply a combination of homogeneous graphs but a fundamentally more complex system encapsulating a range of various interaction types. Despite the rich information inherent to multiplex networks, the research focus in graph representation learning has been

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63778-0_1

overwhelmingly concentrated on methods for homogeneous graphs. This has left a substantial gap in understanding how to leverage the unique features of multiplex networks effectively.

To this end, we examine the problem of learning representations for nodes in multiplex networks from the perspective of unsupervised and self-supervised learning. Unlike supervised learning, these approaches do not require labeled data and, hence, offer a flexible and scalable method for learning node representations. In particular, we investigate various information fusion schemes that can be performed at different stages of the representation learning pipeline to harness the richness of multiplex networks. These schemes are designed to integrate the diverse edge types in multiplex graphs, providing more context and insights into the underlying processes.

This paper details an extensive analysis and experimental evaluation of various information fusion scenarios. We hope our work advances our understanding of multiplex network representation learning and paves the way for developing more robust, efficient, and versatile multiplex network methods.

Our contributions can be summarized as follows:

1. We propose an **information fusion taxonomy** for multiplex networks.
2. We provide an **extensive experimental evaluation** of existing representation learning approaches in multiplex networks. We categorize them according to our proposed taxonomy. We evaluate the node embeddings in three tasks: node classification, node clustering and similarity search.
3. We **identify research gaps** in each fusion strategy category and provide preliminary implementations of those methods. We evaluate their performance and compare them to existing approaches. We **provide a limitations analysis of existing methods** and identify future directions in representation learning for multiplex networks.

2 Related work

Graph representation learning. It has emerged as a vital area of research with applications spanning various domains, such as social networks, bioinformatics, or recommender systems. The essence lies in learning continuous, low-dimensional representations of nodes that capture their structural and attribute information within the graph. Plenty of methods have been developed, each focused on a different aspect of network embeddings, such as structure, attributes, learning paradigm or scalability. Shallow methods, such as DeepWalk [9], LINE [11] use a simple notion of graph coding through random walks or objectives that optimize first and second-order node similarity. More complex graph neural networks, such as GCN[5] or GAT [13], implement the message-passing algorithm over graph edges combined with various message aggregation schemes.

Un- and self-supervised graph representation learning. Inspired by the success of contrastive methods in other domains, the procedures were adapted to graphs. Early approaches were based on the autoencoder architecture (GAE

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63778-0_1

[4]). Another method, DGI [14], employed a GNN to learn node embeddings and maximized the mutual information between node embeddings and the graph embedding (readout function) by discriminating nodes in the original graph from nodes in a corrupted graph. GRACE [20] and GraphCL [18] utilized contrastive learning on graphs. All the previous methods rely on negative sampling which yields a high complexity. Negative-sample-free methods, such as BGRL [12] or GBT [1] use graph augmentations combined with either an asymmetric pipeline architecture or a decorrelation-based loss to prevent representation collapse and learn node embeddings in a self-supervised manner.

Multiplex graph representation learning. Despite the advances in representation learning for homogeneous graphs, research on multiplex graphs is relatively limited. MHGCN [19] utilizes a weighted sum of adjacency matrices of the multiplex graph layers, where the combination weights are learnable. The resulting graph is passed into a standard GCN backbone while the whole model is trained using a link prediction objective with negative sampling. DMGI [8], extends the idea of DGI to multiplex graphs. It applies a GCN backbone with the DGI objective for each graph layer. Node embeddings are fused using a trainable lookup embedding that optimized via a loss function that minimizes the distance layer-wise node embeddings of the original graph and maximizes the distance to embeddings from the corrupted graph. HDGI [10] also builds upon the DGI method. It applies a GCN backbone at each graph layer and fuses the resulting vectors into a single node embedding using a semantic attention mechanism. Such scheme is applied to the original graph and its corrupted version. The two node embedding sets are processed by DGI's original loss function. Contrary to previous approaches, S^2MGRL [7] extends the idea of GBT [1] to multiplex networks. For each graph layer, the model first applies an MLP followed by a GCN block. The loss function applies the Barlow Twins objective between (a) the outputs of the MLPs and GCNs, and (b) all pairs of GCN outputs to ensure correlation between embeddings from all graph layers. This training step (and all previously introduced multiplex graph representation learning approaches) is carried out in an un- or self-supervised manner. However, S^2MGRL trains an attention mechanism on the frozen embedding from all layers to obtain the final node embeddings, utilizing node labels on the output.

3 Information fusion in multiplex graphs

Let us start with basic definitions and the introduction of the multiplex graph representation learning pipeline (see: Figure 1). Next, we will introduce a taxonomy of information fusion methods in multiplex graphs, followed by a detailed discussion of existing methods from each category and our proposed extensions.

Multiplex graph. We define a multiplex graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_K, \mathbf{X})$, where $\mathcal{V}$ denotes a set of nodes, $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d_{\text{in}}}$ is a matrix whose rows contain d_{in}-dimensional node's features. The K layers of the multiplex graph are

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63778-0_1

defined by the K sets of edges: $\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_K$, where each set $\mathcal{E}_i \in \mathcal{V} \times \mathcal{V}$ contains edges between pairs of nodes. Note that the nodes and their features are shared across all graph layers.

Multiplex graph representation learning. When learning embeddings for nodes in multiplex graphs, the input graph is processed by a representation learning model f_θ (parametrized by θ), which computes embedding vectors $\mathbf{Z} \in \mathbb{R}^{|\mathcal{V}| \times d}$ for all nodes in the graph. This representation learning model is often a graph neural network (GNN) based on the message-passing paradigm. In particular, each graph neural network layer is comprised of two phases – message generation and message aggregation (see Equation 1):

$$\begin{aligned} \mathbf{m}_i &= \text{MESSAGE}(\mathbf{h}_i^{(in)}) \\ \mathbf{h}_u^{(out)} &= \text{UPDATE}\left(\mathbf{h}_u^{(in)}, \text{AGG}\left(\{\mathbf{m}_v, v \in \mathcal{N}(u)\}\right)\right) \end{aligned} \tag{1}$$

where MESSAGE$(\cdot)$ is a function that converts node i's input features $h_i^{(in)}$ into a message vector $\mathbf{m}_i$ (for the first GNN layer, we use $H^{(in)} = \mathbf{X}$). Incoming messages from node u's neighbors $\mathcal{N}(u)$ are aggregated using a permutation-invariant function AGG$(\cdot)$. Finally, the u's output representation (embedding) is computed via the UPDATE$(\cdot)$ function, which takes into account both the node's input features $h_u^{(in)}$ and the aggregated neighbor messages.

GNN layers are modified to accommodate the multiple edge types found in multiplex graphs, or separate GNNs are applied on each multiplex graph layer. In the latter case, which is often used in existing multiplex representation learning methods, the output of those GNNs must be fused into a single representation vector for each node.

The resulting node embedding vectors can be applied in various downstream tasks, such as node classification or link prediction. For instance, a node classifier can be trained in a supervised manner on the frozen node representation vectors and corresponding labels.

Multiplex graph information fusion taxonomy. The complex and rich structure of multiplex graphs and their representation learning and processing pipeline enables us to identify several locations for applying information fusion. We propose the following taxonomy of information fusion in multiplex graphs:

- **Graph-level fusion** – methods that directly modify the graph structure and fuse the information from several multiplex graph layers together,
- **GNN-level fusion** – methods that either (a) utilize a dedicated GNN architecture that is specifically designed for multiplex graphs, or (b) models that utilize standard GNNs in combination with trainable fusion mechanisms (such as attention) to produce fused embedding vectors for nodes,
- **Embedding-level fusion** – methods that firstly precompute node embeddings in each multiplex graph layer and then fuse the frozen embeddings into a single embedding per node (post-hoc methods),

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63778-0_1

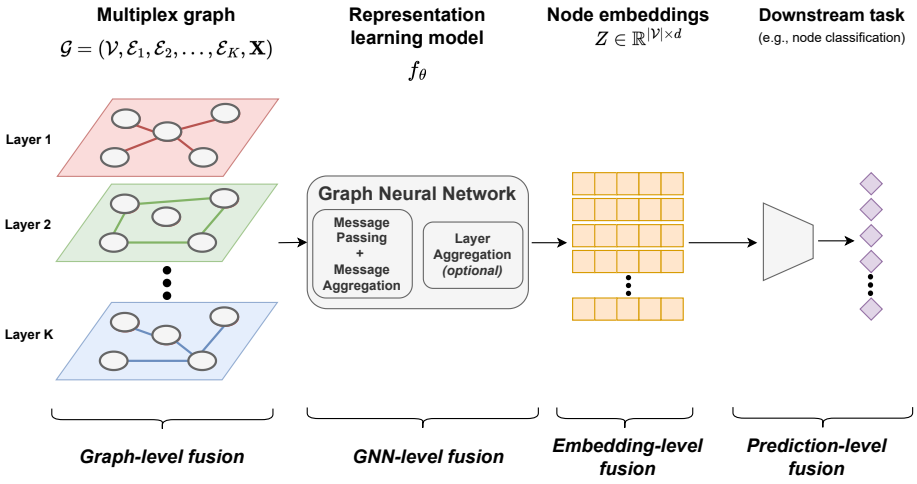

Fig. 1. Multiplex graph processing pipeline and information fusion taxonomy.

– **Prediction-level fusion** – methods that utilize precomputed node embeddings at each multiplex graph layer and then train downstream task models to obtain and fuse their predictions (note that these methods are not part of the unsupervised or self-supervised representation learning pipeline).

Such taxonomy can capture various multiplex graph representation learning methods and categorize them according to where the critical information fusion step is applied. Moreover, one could build another categorization based on the actual fusion mechanism. However, to keep things clearer, we introduce them in detail in Sections 3.3-3.6.

3.1 Experimental setup and evaluation

In the following sections, we will introduce the existing methods for each category in our proposed taxonomy, along with an experimental evaluation and comparison of the methods' performance in selected downstream tasks. We implemented all models using the PyTorch-Geometric [2] library, and we utilize the DVC package [6] to ensure reproducibility. We make our code and data publicly available at `https://github.com/graphml-lab-pwr/multiplex-fusion`. Due to page limits, we leave the details about the hyperparameter search ranges and the final chosen ones in the code repository. Let us now introduce the utilized datasets and downstream tasks with the appropriate evaluation protocols.

Datasets. We selected six real-world datasets frequently utilized in representation learning papers for multiplex graphs [8,10,7,19]. We summarize the statistics of those datasets in Table 1.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63778-0_1

Table 1. Dataset statistics. We summarize the number of nodes ($|\mathcal{V}|$), the names of particular graph layers along with the number of edges in each layer ($_{(\textbf{Layer } i)}$ $|\mathcal{E}_i|$), the node feature dimension (d_{in}) and the number of classes in the node classification task.

	ACM	**Amazon**	**Freebase**	**IMDB**	**Cora**	**CiteSeer**
$\lvert\mathcal{V}\rvert$	3,025	7,621	3,492	3,550	2,708	3,327
$_{(\textbf{Layer } i)}$ $\lvert\mathcal{E}_i\rvert$	(PAP) 29,281	(IBI) 1,104,257	(MAM) 254,702	(MAM) 66,428	(CIT) 10,556	(CIT) 9,104
	(PSP) 2,210,761	(IOI) 14,305	(MDM) 8,404	(MDM) 13,788	(KNN) 54,160	(KNN) 66,540
		(IVI) 266,237	(MWM) 10,706			
d_{in}	1,870	2,000	3,492	2,000	1,433	3,703
#classes	3	4	3	3	7	6

- ACM [15] is an academic paper dataset with nodes assigned to one of 3 classes: database, wireless communication, and data mining. The multiplex layers are built from 2 meta-paths obtained from this graph: Paper-Author-Paper (PAP) and Paper-Subject-Paper (PSP). Node features are TF-IDF encodings of the papers' keywords. Note that the PSP layer has significantly more edges than the PAP layer.
- Amazon [3] is a multiplex network dataset of items bought on Amazon. The graph layers refer to common activities of users, i.e., also-bought (IBI), bought-together (IOI) and also-viewed (IVI). Each item can be categorized into one of 4 classes – Beauty, Automotive, Patio Lawn and Garden, and Baby. Note that there is a disproportion in the size of particular graph layers. Node features are built as bag-of-words encodings of the item descriptions.
- Freebase [16] is a movies graph dataset (from the Freebase KG) with nodes divided into three classes – Action, Comedy and Drama. Graph layers are built upon the following meta-paths: Movie-Actor-Movie (MAM), Movie-Director-Movie (MDM) and Movie-Writer-Movie (MWM). No inherent node features are available, so a one-hot encoding is utilized.
- IMDB [17] is another movie graph dataset extracted from the IMDB knowledge graph. Nodes are movies, and each is assigned one of 3 classes – Action, Comedy and Drama. Graph layers are built upon the following meta-paths: Movie-Actor-Movie (MAM) and Movie-Director-Movie (MDM). The node features are TF-IDF encodings of movie metadata, such as title or language.
- Cora, CiteSeer [5] are two citation graphs where nodes are research papers divided into seven and six different research areas, respectively. Node features are bag-of-words encodings of paper abstracts. These datasets are inherently homogeneous, and the original edges are put into the CIT layer. We build the multiplex graph by extending the original networks by another graph layer — a paper similarity layer. We find each node's k-nearest neighbors (KNN) based on the cosine similarity of node features and select the top 10 similar papers (i.e., $k = 10$).

Downstream tasks. The utilized datasets are equipped with node labels, so a node classification is a natural choice for the downstream evaluation of node representations. However, to fully explore the generalization abilities of the learned

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63778-0_1

embedding vectors, we employ two additional tasks – node clustering and node similarity search [8]. For all tasks, we collect various metrics (provided in the code repository). To ensure clarity, we focus only on one metric per task in the paper's result tables. Before executing the downstream evaluation, we freeze the representation backbone model and extract the node embeddings.

- node classification (**Clf**) – we follow the standard evaluation protocol for unsupervised and self-supervised node representation learning methods [14]. We train a logistic regression classifier on the node embeddings and corresponding label pairs from the training set. We provide the Macro-F1 (**MaF1**) metric on the test set. Additionally, we use the this task for the models' hyperparameter searches. We select the models with the highest Macro-F1 score on the validation set.
- node clustering (**Clu**) – we train a K-Means model on the node embeddings and measure the clustering performance using the Normalized Mutual Information (**NMI**) score on the test set. We train the clustering model ten times with different seeds and average the scores to obtain the final NMI score.
- node similarity search – we compute the cosine similarity scores of the node embeddings between all pairs of nodes and build a ranking for each node according to the similarity score. Next, we calculate the ratio of the nodes that belong to the same class within top-5 ranked nodes (**Sim@5**).

Table notation. In all result tables (Tab. 2-7), we utilize the same notations and present the mean and std (given in parentheses) of the Macro-F1, NMI and Sim@5 scores as percentages in the node classification, node clustering and similarity search tasks, respectively. Please refer to the "***Methods.***" paragraphs in corresponding sections for explanations of model names. Shaded rows denote our proposed methods and extensions.

3.2 Baseline approaches (no fusion)

Methods. We examine two baseline approaches – `Layers` and `Features`. In the first approach, we train the DGI [14] model on each graph layer separately and evaluate the resulting embeddings. We selected DGI due to its popularity in multiplex representation learning method design and its ability to learn node embeddings in an unsupervised way. The `Features` approach does not involve any learning but evaluates the initial node features as the node representations.

Discussion. Results are given in Table 2. In most cases, we observe that different graph layers perform differently in the downstream tasks. For the ACM dataset, the smaller layer PAP performs much better in the node classification and similarity search tasks than the much larger PSP layer. However, the clustering performance is better on the PSP layer. Utilizing the node features also yields a decent performance; however, it is not as good as for the PAP layer. A similar situation happens for the IMDB dataset, where the larger MAM layer performs worse than the MDM layer – this time in all three downstream tasks.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63778-0_1

Table 2. Downstream tasks performance of methods **without information fusion**.

	ACM			**Amazon**			**Freebase**		
	Clf (MaF1)	**Clu** (NMI)	**Sim@5**	**Clf** (MaF1)	**Clu** (NMI)	**Sim@5**	**Clf** (MaF1)	**Clu** (NMI)	**Sim@5**
`Layers`	PAP			IBI			MAM		
	82.33 (6.51)	37.01 (6.06)	85.26 (0.56)	26.50 (0.79)	0.15 (0.04)	31.06 (0.25)	51.79 (1.24)	15.82 (4.44)	55.72 (0.28)
	PSP			IOI			MDM		
	56.17 (5.16)	50.42 (1.57)	66.06 (2.88)	35.63 (1.09)	1.01 (0.53)	48.83 (1.08)	35.97 (1.28)	0.87 (0.08)	41.28 (1.06)
				IVI			MWM		
				27.08 (0.63)	0.14 (0.03)	29.94 (0.33)	25.26 (1.00)	0.27 (0.09)	41.54 (1.12)
`Features`	73.30 (0.00)	14.83 (4.14)	71.75 (0.00)	71.27 (0.00)	4.11 (1.80)	84.09 (0.00)	20.18 (0.00)	0.41 (0.03)	32.04 (0.00)
	IMDB			**Cora**			**CiteSeer**		
	Clf (MaF1)	**Clu** (NMI)	**Sim@5**	**Clf** (MaF1)	**Clu** (NMI)	**Sim@5**	**Clf** (MaF1)	**Clu** (NMI)	**Sim@5**
`Layers`	MAM			CIT			CIT		
	41.99 (2.92)	0.48 (0.28)	45.98 (0.30)	65.32 (3.37)	25.70 (5.26)	75.49 (1.37)	58.50 (3.74)	27.36 (4.74)	63.47 (0.82)
	MDM			KNN			KNN		
	51.50 (2.01)	5.09 (1.36)	51.68 (0.36)	51.87 (4.57)	16.99 (1.01)	57.67 (1.17)	55.93 (1.89)	24.08 (1.49)	61.23 (0.32)
`Features`	56.59 (0.00)	6.75 (2.01)	51.64 (0.00)	55.42 (0.00)	13.47 (0.63)	50.10 (0.00)	58.73 (0.00)	18.64 (2.22)	53.48 (0.00)

The node features yield the best overall performance. Another case worth noting occurs in the Amazon dataset – the node features outperform all layer information with about 40 pp difference in the node classification task, 50 pp for the similarity search task, and about 4-40 times better performance on the clustering task. Note that the IBI and IVI layers achieve only 0.15% and 0.14% NMI, which are inferior scores. In the case of the artificially constructed Cora and CiteSeer datasets, we observe that the added KNN layers achieve similar performance to the node features, which is expected as these datasets are highly homophilic.

3.3 Graph-level fusion

Methods. Information fusion in multiplex graph representation learning can be applied in the graphs themselves. Not much research has been carried out, and the most prominent existing method for graph-level fusion is `MHGCN` [19]. The method consists of a GCN backbone trained using a link prediction objective with negative edge sampling. For a K-layer multiplex graph, the method jointly learns K edge weights: $\beta_1, \beta_2, \ldots, \beta_K$, which are assigned to each edge in the corresponding graph layer. Finally, the multiplex graph is converted into a homogeneous weighted graph, where the edge weights are summed over the graph layers. For instance, if edge (u, v) exists in three graph layers: i, j and k, the final edge weight is given as: $\beta^{(u,v)} = \beta_i + \beta_j + \beta_k$. We compare the `MHGCN` method with an approach where we flatten (`Flattened`) the input graph into a non-weighted homogeneous graph with multi-edges, i.e., $\mathcal{G} = (\mathcal{V}, \mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_K, \mathbf{X}) \rightarrow G_{\text{flattened}} = (\mathcal{V}, \mathcal{E}_\Sigma, \mathbf{X})$, where $\mathcal{E}_\Sigma = \mathcal{E}_1 \cup \mathcal{E}_2 \cup \ldots \cup \mathcal{E}_K$ is the multi-set of edges (an edge between two nodes might occur in several graph layers). We evaluate the performance of several popular node representation learning methods for homogenous graphs – the supervised `GCN` and `GAT` models, the unsupervised DeepWalk (`DW`) and `DGI` models.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63778-0_1

Discussion. Results are given in Table 3. The `MHGCN` method did not perform well on the ACM, Amazon and IMDB datasets compared to baseline approaches (no fusion). This might be related to the overall high number of edges (approx. 2M and 1.3M, respectively) combined with the link prediction objective – mining the negative edges might be troublesome. We leave the evaluation of a modified model training objective as future work. On the contrary, for the Freebase dataset, we observe an approx. 11 pp improvement in the node classification task along with substantial improvement in the similarity search task and comparable results for the clustering task. Similarly, Cora and CiteSeer also benefit from the learnable graph-level fusion approach, which increases performance by up to 15 pp. When it comes to the `Flattened` graph methods, we firstly observe a consistently better performance for the supervised models compared to unsupervised ones, which is related to the direct access to node labels. However, the `DW` and `DGI` methods achieve similar results to `MHGCN` on ACM and Amazon, while for other datasets, the results are 5-10 pp worse.

Table 3. Downstream tasks performance of **graph-level fusion methods**.

		ACM			Amazon			Freebase		
		Clf (MaF1)	**Clu** (NMI)	**Sim@5**	**Clf** (MaF1)	**Clu** (NMI)	**Sim@5**	**Clf** (MaF1)	**Clu** (NMI)	**Sim@5**
Flattened	GCN	70.19 (0.22)	50.12 (0.35)	73.38 (0.23)	27.84 (0.17)	0.81 (0.11)	27.88 (0.20)	60.70 (0.55)	18.71 (1.20)	56.91 (0.43)
	GAT	70.34 (2.89)	44.35 (2.10)	72.17 (1.16)	25.88 (2.06)	0.20 (0.10)	30.08 (1.02)	57.22 (1.24)	16.54 (1.26)	56.80 (0.42)
	DW	64.25 (0.64)	37.53 (1.82)	62.98 (0.79)	24.45 (0.39)	0.04 (0.01)	26.79 (0.39)	49.20 (1.55)	17.01 (0.47)	53.67 (0.79)
	DGI	59.54 (0.78)	36.07 (7.00)	72.85 (0.55)	25.60 (0.92)	0.12 (0.02)	28.38 (0.15)	51.22 (0.57)	17.86 (0.79)	55.72 (0.51)
MHGCN		63.35 (0.88)	38.89 (0.00)	71.05 (3.13)	23.49 (1.69)	0.04 (0.01)	29.44 (0.17)	62.75 (0.20)	14.48 (1.59)	60.84 (0.61)
		IMDB			**Cora**			**CiteSeer**		
		Clf (MaF1)	**Clu** (NMI)	**Sim@5**	**Clf** (MaF1)	**Clu** (NMI)	**Sim@5**	**Clf** (MaF1)	**Clu** (NMI)	**Sim@5**
Flattened	GCN	57.04 (0.50)	10.69 (0.21)	49.01 (1.03)	68.76 (0.45)	41.11 (0.82)	66.65 (0.33)	64.27 (0.22)	39.93 (0.75)	62.28 (0.73)
	GAT	54.81 (0.67)	10.11 (0.49)	46.90 (0.61)	65.29 (1.68)	36.99 (1.94)	59.94 (1.18)	60.53 (1.84)	32.84 (3.10)	60.17 (1.37)
	DW	38.09 (0.78)	0.94 (0.19)	38.11 (0.60)	61.50 (1.25)	39.71 (1.00)	53.86 (1.24)	54.16 (1.35)	40.38 (1.07)	53.74 (0.32)
	DGI	40.51 (4.44)	0.27 (0.08)	48.84 (0.49)	58.09 (3.60)	21.53 (3.48)	62.57 (0.64)	60.43 (2.25)	29.08 (2.38)	63.38 (0.71)
MHGCN		49.77 (0.83)	2.69 (0.78)	50.07 (0.56)	71.22 (1.17)	35.61 (1.70)	67.09 (0.25)	62.55 (0.84)	39.64 (2.13)	63.02 (0.71)

3.4 Prediction-level fusion

Methods. In this scenario, we assume that precomputed node embedding vectors exist for each multiplex graph layer. We utilize the layer-wise node representations of the `DGI` method from the baseline experiments in Section 3.2. Based on those embeddings and the node labels, we train logistic regression classifiers and combine their predictions to obtain a prediction-level information fusion scheme. In particular, we validate two approaches – `soft` and `hard` ensemble voting. In the `hard` voting case, we first obtain predicted node labels from each classifier and find the most occurring label (also known as majority voting). In the `soft` case, we first obtain the class probabilities, average them over all classifiers, and select the label with the highest probability. Note that such a scheme is applicable only in the node classification task.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63778-0_1

Discussion. Results are given in Table 4. We observe that the `soft` approach performs consistently better on all datasets compared to `hard`. One might lose too much information about the actual class distribution and model confidence by discarding the actual probabilities. Compared to baseline methods and graph-level fusion, we do not observe a significant improvement, yet sometimes, the performance is worse than just utilizing a single graph layer. The information fusion occurs too late in the graph processing pipeline, making discovering complex relationships in the multiplex data harder.

Table 4. Downstream tasks performance of **prediction-level fusion methods**.

		ACM **Clf** (MaF1)	**Amazon** **Clf** (MaF1)	**Freebase** **Clf** (MaF1)	**IMDB** **Clf** (MaF1)	**Cora** **Clf** (MaF1)	**CiteSeer** **Clf** (MaF1)
Vote	`soft`	81.95 (6.23)	36.20 (1.01)	51.08 (1.17)	49.73 (3.02)	65.38 (4.12)	60.40 (2.71)
	`hard`	58.32 (6.81)	31.34 (0.89)	45.18 (1.04)	42.13 (3.51)	61.68 (2.77)	57.32 (1.98)

3.5 GNN-level fusion

Methods. This case is where most existing methods apply the fusion mechanism. It seems a natural choice to perform information fusion as a part of the representation learning model. `DMGI` [8] applies a GCN backbone with the DGI objective at each graph layer and learns the fused node representations as a trainable embedding lookup matrix. The additional loss term minimizes the distance to positive layer-wise embeddings (original graph) and maximizes the distance to negative ones (corrupted graph). `HDGI` [10] employs a similar approach but uses a semantic attention mechanism to fuse the layer-wise node embeddings. Instead of applying the DGI loss to each graph layer, the model applies it to the fused embeddings. `S`^{2}`MGRL` [7] builds upon GBT [1] and extends it to multiplex networks. Each graph layer is processed by an MLP, followed by a GCN block. The loss function is twofold: (1) a Barlow Twins (BT) loss term between the outputs of the MLPs and GCNs, and (2) the BT loss computed between all pairs of GCN outputs. Contrary to previous approaches, S^{2}MGRL is not fully un-/self-supervised. It trains an attention mechanism on pairs of frozen layer-wise embeddings and node labels to fuse node embeddings. We propose extensions to the above methods: `F(GBT, *)` and `F(DGI, *)`, where $*$ denotes the fusion method (attention `Att` or concatenation with linear projection `CL`). For `F(GBT, *)` we employ a similar scheme as `S`^{2}`MGRL`, but instead of computing the BT loss for all graph layer pairs, we compute the BT loss between each graph layer output and the fused node embedding (which scales linearly). For the `F(DGI, ·)`, we apply the DGI loss at each graph layer, and additionally on the fused node embeddings (of the original and corrupted graph). Such settings are both inductive and self-supervised.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63778-0_1

Table 5. Downstream tasks performance of **GNN-level fusion methods**.

	ACM			Amazon			Freebase		
	Clf (MaF1)	**Clu** (NMI)	**Sim@5**	**Clf** (MaF1)	**Clu** (NMI)	**Sim@5**	**Clf** (MaF1)	**Clu** (NMI)	**Sim@5**
DMGI	87.76 (0.82)	57.29 (4.47)	87.94 (0.52)	73.15 (1.85)	36.11 (2.60)	79.27 (0.59)	49.79 (1.97)	15.36 (0.49)	54.83 (2.16)
HDGI	85.01 (2.47)	56.33 (9.53)	87.28 (0.64)	37.16 (3.53)	1.21 (0.96)	44.40 (2.25)	47.00 (2.55)	16.58 (0.51)	55.29 (0.53)
S^2MGRL	86.14 (2.14)	43.02 (25.10)	85.79 (1.04)	50.99 (3.78)	3.91 (1.86)	58.27 (2.80)	47.63 (7.63)	7.22 (3.79)	57.41 (1.46)
F(GBT,Att)	74.52 (3.22)	22.90 (12.59)	78.45 (2.04)	48.06 (2.64)	0.76 (0.17)	52.51 (0.85)	49.20 (2.48)	2.01 (1.21)	55.53 (1.57)
F(GBT,CL)	81.37 (2.18)	30.54 (23.06)	83.41 (1.51)	50.54 (4.17)	1.41 (1.66)	52.22 (3.07)	48.26 (2.07)	3.02 (2.53)	55.51 (1.12)
F(DGI,Att)	86.68 (2.16)	67.57 (4.36)	89.05 (0.45)	31.17 (3.54)	0.39 (0.16)	44.86 (2.97)	49.79 (0.98)	16.09 (1.17)	57.50 (0.59)
F(DGI,CL)	84.11 (4.30)	61.13 (6.28)	89.21 (0.29)	34.24 (2.68)	0.28 (0.10)	42.28 (3.38)	49.21 (0.97)	0.80 (0.57)	52.97 (0.95)

	IMDB			Cora			CiteSeer		
	Clf (MaF1)	**Clu** (NMI)	**Sim@5**	**Clf** (MaF1)	**Clu** (NMI)	**Sim@5**	**Clf** (MaF1)	**Clu** (NMI)	**Sim@5**
DMGI	60.81 (0.66)	19.70 (0.49)	60.05 (0.63)	73.55 (0.56)	42.30 (2.70)	72.78 (0.83)	66.13 (1.75)	43.37 (0.59)	66.84 (0.56)
HDGI	52.60 (1.70)	13.66 (1.42)	55.45 (1.36)	61.45 (2.53)	34.18 (3.00)	67.43 (2.80)	60.54 (1.18)	40.01 (2.18)	64.06 (1.54)
S^2MGRL	44.23 (8.44)	1.36 (1.76)	48.84 (2.55)	61.03 (7.64)	19.99 (11.78)	67.86 (1.71)	52.66 (3.98)	20.96 (8.81)	62.30 (0.61)
F(GBT,Att)	53.70 (0.95)	2.00 (2.06)	50.39 (0.42)	68.70 (2.22)	22.01 (3.50)	68.60 (1.01)	59.30 (2.40)	19.95 (6.32)	61.36 (1.20)
F(GBT,CL)	56.30 (1.08)	0.53 (0.39)	52.59 (0.48)	70.57 (1.61)	21.61 (1.48)	69.87 (1.21)	60.39 (2.24)	18.61 (5.33)	62.54 (0.98)
F(DGI,Att)	55.56 (1.53)	11.85 (3.88)	56.59 (0.74)	66.44 (1.01)	35.18 (0.46)	68.18 (0.71)	62.25 (2.04)	36.58 (3.17)	62.58 (0.52)
F(DGI,CL)	52.29 (3.40)	1.03 (1.58)	53.83 (1.02)	62.33 (3.68)	13.51 (4.36)	66.40 (1.15)	60.88 (1.49)	17.55 (1.49)	64.08 (0.74)

Discussion. Results are given in Table 5. We observe that the lookup embedding-based fusion approach of the DMGI method consistently delivers the best results on the node classification task compared to the other two approaches. A similar observation can be made for the clustering and similarity search task except for the Freebase dataset, where the performance is alike. Compared to previous fusion scenarios, the results of DMGI are best overall on all datasets except for Freebase, where the model achieves slightly worse results. One could argue that such lookup embedding is the best fusion mechanism. However, note that it does not allow us to easily obtain embeddings for newly added nodes or changes in the graph structure (it is transductive). HDGI's attention mechanism proves to perform similarly to DMGI on ACM and Freebase, worse on IMDB, Cora and CiteSeer, but on the Amazon dataset, we observe an approx. 35 pp difference in the node classification and similarity search tasks. The utilization of the BT objective in S^2MGRL achieves comparable performance to the aforementioned methods on ACM, Freebase and Cora, but on Amazon and IMDB, the difference is more substantial. For our proposed extensions, F(GBT, $*$) and F(DGI, $*$), we observe that both variants provide competitive performance to existing multiplex approaches. If we compare the results to HDGI, which is the only model that is both inductive and self-supervised, the performance gains are even greater. For instance, 56% vs. 52% MaF1 on IMDB, 70% vs. 61% MaF1 on Cora, or even 50% vs. 37% MaF1 on Amazon.

3.6 Embedding-level fusion

Methods. A basic fusion approach often presented in multiplex representation learning papers is first to precompute node embeddings for each graph layer separately and then fuse them by computing the average vectors (over all graph layers). We denote that approach as Mean and use it in combination with a

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63778-0_1

supervised GCN, GAT and unsupervised DeepWalk (DW) and DGI models (same set of models as in the flattened graph case – see Section 3.3). We extend these embedding-level (post-hoc) fusion methods by examining other non-trainable fusion operators (i.e., Concat, Min, Max, Sum). We also build models with trainable fusion mechanisms – i.e., attention (Att), concatenation with a linear projection (CL) and a lookup embedding (Lk). We utilize an MSE or BT loss computed between the layer-wise embedding vectors and the fused output representation.

Table 6. Downstream tasks performance of **embedding-level fusion methods**.

		ACM			Amazon			Freebase		
		Clf (MaF1)	**Clu** (NMI)	**Sim@5**	**Clf** (MaF1)	**Clu** (NMI)	**Sim@5**	**Clf** (MaF1)	**Clu** (NMI)	**Sim@5**
GCN	Mean	84.66 (0.16)	62.53 (0.53)	85.61 (0.11)	45.31 (11.90)	9.06 (8.14)	46.05 (5.44)	58.32 (2.69)	20.24 (1.05)	58.07 (0.66)
GAT	Mean	85.03 (0.98)	59.55 (4.51)	85.36 (0.16)	35.83 (9.66)	4.18 (6.28)	41.13 (4.05)	50.38 (6.87)	15.81 (3.89)	57.44 (0.65)
DW	Mean	69.21 (0.96)	35.63 (3.20)	64.04 (3.23)	24.33 (0.59)	0.04 (0.02)	26.74 (0.53)	41.61 (2.48)	8.75 (3.98)	43.28 (2.37)
DGI	Mean	80.14 (8.60)	41.12 (4.15)	85.95 (1.04)	31.93 (0.43)	0.36 (0.09)	42.93 (0.58)	49.22 (2.47)	6.57 (8.05)	54.36 (1.42)
DGI	Concat	82.42 (6.51)	42.02 (4.81)	85.90 (1.09)	36.21 (1.02)	0.45 (0.13)	46.42 (0.81)	50.90 (1.22)	7.52 (7.62)	54.82 (1.56)
DGI	Min	77.80 (9.57)	42.40 (7.76)	85.69 (0.98)	28.22 (2.20)	0.72 (0.04)	39.45 (0.33)	50.60 (1.44)	8.51 (7.69)	55.35 (0.80)
DGI	Max	82.21 (6.30)	40.35 (4.88)	85.94 (1.06)	28.49 (1.00)	0.72 (0.09)	37.45 (0.51)	49.89 (2.58)	9.08 (8.84)	54.66 (0.82)
DGI	Sum	82.44 (6.42)	41.12 (4.15)	85.95 (1.04)	31.96 (0.44)	0.36 (0.09)	42.93 (0.58)	50.53 (1.53)	6.57 (8.05)	54.36 (1.42)
DGI	Att,BT	80.37 (9.83)	50.97 (6.63)	86.17 (0.90)	31.60 (0.48)	0.31 (0.06)	42.04 (1.25)	49.67 (2.03)	8.07 (7.38)	54.48 (1.13)
DGI	Att,MSE	80.12 (8.57)	41.48 (4.60)	85.95 (1.02)	31.95 (0.43)	0.39 (0.08)	42.92 (0.58)	49.24 (2.50)	6.57 (8.06)	54.37 (1.38)
DGI	CL,BT	85.39 (1.79)	36.26 (6.87)	86.10 (0.92)	30.25 (1.05)	0.21 (0.15)	34.17 (2.03)	50.24 (2.01)	7.77 (3.56)	54.23 (0.66)
DGI	CL,MSE	80.17 (9.18)	39.82 (3.85)	85.85 (0.73)	32.43 (0.93)	0.31 (0.14)	42.30 (1.00)	49.04 (2.55)	6.36 (8.04)	54.00 (1.08)
DGI	Lk,BT	82.71 (1.03)	43.93 (7.08)	83.65 (1.83)	27.86 (0.65)	0.13 (0.05)	33.75 (1.35)	45.18 (1.12)	8.44 (6.00)	53.12 (0.87)
DGI	Lk,MSE	33.59 (0.64)	0.09 (0.01)	32.67 (0.28)	24.67 (0.51)	0.05 (0.01)	26.75 (0.18)	33.08 (2.63)	0.22 (0.08)	36.98 (0.84)

		IMDB			Cora			CiteSeer		
		Clf (MaF1)	**Clu** (NMI)	**Sim@5**	**Clf** (MaF1)	**Clu** (NMI)	**Sim@5**	**Clf** (MaF1)	**Clu** (NMI)	**Sim@5**
GCN	Mean	61.63 (0.83)	16.39 (0.82)	55.41 (0.47)	76.88 (1.44)	48.40 (4.13)	74.73 (1.50)	63.50 (0.86)	39.51 (1.79)	63.12 (0.77)
GAT	Mean	54.57 (4.41)	6.46 (2.00)	51.14 (1.28)	73.94 (3.18)	46.98 (5.84)	72.52 (2.01)	64.10 (0.72)	39.20 (2.42)	64.52 (1.09)
DW	Mean	34.02 (0.80)	0.11 (0.05)	35.03 (0.54)	54.89 (2.54)	34.13 (1.98)	49.82 (1.61)	44.32 (1.41)	27.10 (2.49)	43.33 (1.30)
DGI	Mean	49.05 (3.22)	0.80 (0.86)	51.64 (0.59)	62.63 (4.33)	22.96 (1.42)	69.18 (2.55)	58.37 (2.34)	27.61 (1.48)	64.01 (0.87)
DGI	Concat	49.72 (2.93)	0.61 (0.50)	52.15 (0.43)	65.51 (4.09)	22.84 (0.72)	70.41 (2.44)	60.62 (2.49)	28.82 (2.29)	64.47 (1.04)
DGI	Min	49.20 (2.74)	0.63 (0.51)	51.48 (0.86)	63.81 (6.87)	23.90 (5.20)	69.86 (1.75)	58.96 (4.59)	30.47 (2.45)	64.36 (0.58)
DGI	Max	47.27 (3.78)	0.47 (0.25)	51.12 (0.75)	60.24 (4.91)	19.37 (1.81)	68.53 (3.38)	57.62 (1.97)	23.70 (1.83)	63.24 (0.94)
DGI	Sum	49.17 (3.09)	0.80 (0.86)	51.64 (0.59)	63.42 (4.18)	22.96 (1.42)	69.18 (2.55)	59.00 (2.24)	27.61 (1.48)	64.01 (0.87)
DGI	Att,BT	50.76 (0.81)	1.46 (1.63)	51.66 (0.73)	66.34 (3.20)	29.10 (3.10)	70.99 (1.11)	60.71 (2.89)	31.49 (4.04)	64.36 (0.93)
DGI	Att,MSE	49.05 (3.31)	0.81 (0.89)	51.62 (0.62)	62.56 (4.29)	22.49 (1.45)	69.15 (2.56)	58.37 (2.34)	27.73 (1.43)	64.01 (0.89)
DGI	CL,BT	52.82 (1.67)	1.51 (0.74)	50.88 (1.33)	68.16 (2.22)	32.67 (3.13)	70.40 (0.78)	61.19 (2.11)	33.92 (2.93)	62.88 (0.47)
DGI	CL,MSE	46.95 (2.50)	0.57 (0.55)	50.62 (1.51)	61.61 (4.11)	22.27 (1.65)	68.11 (2.22)	57.91 (3.24)	27.57 (0.71)	63.52 (0.82)
DGI	Lk,BT	50.50 (1.78)	2.26 (2.28)	49.40 (1.32)	68.43 (2.03)	28.44 (3.30)	66.39 (0.47)	54.76 (1.54)	31.86 (0.37)	60.92 (0.94)
DGI	Lk,MSE	33.11 (0.86)	0.06 (0.01)	34.03 (0.53)	14.12 (1.16)	1.09 (0.08)	18.22 (0.34)	16.85 (1.17)	0.78 (0.12)	18.22 (0.65)

Discussion. Results are given in Table 6. The supervised GCN and GAT models perform better than the unsupervised ones. However, in some cases, the performance gap is small (e.g., GAT and DGI on Freebase in the node classification task). Regarding other non-trainable operators in conjunction with the DGI model, we observe that Concat often delivers the best performance, however, at the cost of higher-dimensional node embedding vectors. This approach would be infeasible when dealing with multiplex networks with many layers (edge types). The other operators (Min, Max, Sum) perform quite similarly to each other on all datasets. However, on the ACM dataset, we observe that using either Max or Sum aggregation leads to a performance increase of about 2 pp compared to Mean along at a

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63778-0_1

reduced std. Trainable operators also allow for an increase in the performance in downstream tasks in most cases – on ACM, the best choice is the `CL, BT` model with approx. 85% MaF1 and 86% Sim@5, while for clustering the best score is achieved by `Att, BT`. On Amazon and Freebase, the results are not significantly different. On IMDB and CiteSeer, we obtain about 2-3pp increase in node classification, and on Cora `CL, BT` achieves about 68% MaF1, while outperforming the results of `DGI, Mean` on clustering and similarity search tasks.

4 Conclusions, research gaps and future work

In this paper, we tackled the problem of node representation learning in multiplex graphs. We proposed a novel taxonomy for categorizing embedding methods based on the location where the information fusion from different graph layers (edge types) occurs. In particular, we identified graph-level, GNN-level, posthoc embedding-level and prediction-level fusion approaches. These are based either on non-trainable basic operators (e.g., averaging) or more complex trainable ones (e.g., attention). We summarize a detailed comparison of methods and fusion approaches in Table 7. Combined with our experimental evaluation on three downstream tasks and six multiplex graph datasets, we find the following limitations of existing approaches and propose guidelines for future research:

Model architecture. Most un-/self-supervised representation learning methods for multiplex graphs are based on the DGI method. Self-supervised learning has proved to achieve SoTA performance on homogeneous graph datasets. We propose to evaluate other base architectures. The attempt to utilize the G-BT architecture on multiplex graphs, i.e., S^2MGRL, revealed promising results. However, the final proposed model does not work in a fully unsupervised fashion and requires node labels to train the fusion mechanism.

Inductive learning. `DMGI` achieves the best results on most datasets and downstream tasks. However, the utilized lookup embedding-based fusion mechanism does not provide inductive capabilities. This is especially important in real-world applications where the graph data is subject to change.

New type of inductive models. Inductivity in terms of multiplex graphs can be viewed twofold: (A) modification of a single graph layer (addition of nodes or edges), where most GNN-based methods are capable of inferring new embeddings in such scenarios, (B) addition of new multiplex graph layers. Virtually all methods requires a whole model retraining when adding a new graph layer. Methods based on graph flattening are capable of layer-inductivity at the cost of rather poor performance in downstream tasks.

Multiplex GNN layer. We did not find any attempts to build dedicated GNN layer architectures designed for multiplex graph data. Proposed models instead utilize existing ones and apply them to each graph layer separately, relying on information fusion to be executed in a later step.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63778-0_1

Model scalability. Moreover, applying GNNs at each layer separately yields a high number of model parameters. We propose to explore the application of GNN sharing. For instance, this could be achieved by designing new positional encodings for multiplex graph layers.

Table 7. Comparison of multiplex graph representation learning and fusion methods.

Name		Unsuperivsed?	Inductive?		Fusion location/level				Fusion type
			Node/edge	Layer	Graph	GNN	Emb.	Pred.	
Flattened	GCN	✗	✓	✓	✓	✗	✗	✗	graph flattening
	GAT	✗	✓	✓	✓	✗	✗	✗	
	DW	✓	✗	✗	✓	✗	✗	✗	
	DGI	✓	✓	✓	✓	✗	✗	✗	
MHGCN		✓	✓	✗	✓	✗	✗	✗	weighted sum of adj. matrices
DMGI		✓	✗	✗	✗	✓	✗	✗	lookup
HDGI		✓	✓	✗	✗	✓	✗	✗	semantic attention
S^2MGRL		✓(fusion: ✗)	✓	✗	✗	✓	✗	✗	supervised attention
F(GBT, ∗)		✓	✓	✗	✗	✓	✗	✗	attention / concat with linear proj.
F(DGI, ∗)		✓	✓	✗	✗	✓	✗	✗	
GCN	Mean	✗	✓	✗	✗	✗	✓	✗	mean
GAT		✗	✓	✗	✗	✗	✓	✗	
DW		✓	✗	✗	✗	✗	✓	✗	
DGI		✓	✓	✗	✗	✗	✓	✗	
DGI	Concat	✓	✓	✗	✗	✗	✓	✗	concat
	Min	✓	✓	✗	✗	✗	✓	✗	min
	Max	✓	✓	✗	✗	✗	✓	✗	max
	Sum	✓	✓	✗	✗	✗	✓	✗	sum
DGI	Att,BT	✓	✓	✗	✗	✗	✓	✗	attention
	Att,MSE	✓	✓	✗	✗	✗	✓	✗	
	CL,BT	✓	✓	✗	✗	✗	✓	✗	concat with linear proj.
	CL,MSE	✓	✓	✗	✗	✗	✓	✗	
	Lk,BT	✓	✗	✗	✗	✗	✓	✗	lookup
	Lk,MSE	✓	✗	✗	✗	✗	✓	✗	
Vote	soft	✗	✓	✗	✗	✗	✗	✓	ensemble voting
	hard	✗	✓	✗	✗	✗	✗	✓	

Acknowledgments

The project was partially supported by the National Science Centre, Poland (grant number 2021/41/N/ST6/03694), the European Union under the Horizon Europe grant OMINO (grant number 101086321) and Polish Ministry of Science, as well as statutory funds from the Department of Artificial Intelligence.

References

1. Bielak, P., Kajdanowicz, T., Chawla, N.V.: Graph barlow twins: A self-supervised representation learning framework for graphs. Knowledge-Based Systems **256**, 109631 (2022). https://doi.org/10.1016/j.knosys.2022.109631
2. Fey, M., Lenssen, J.E.: Fast graph representation learning with PyTorch Geometric. In: ICLR Workshop on Representation Learning on Graphs and Manifolds (2019)

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63778-0_1

3. He, R., McAuley, J.: Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In: WWW '16 (2016). https://doi.org/10.1145/2872427.2883037
4. Kipf, T.N., Welling, M.: Variational graph auto-encoders. NIPS Workshop on Bayesian Deep Learning (2016)
5. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: International Conference on Learning Representations (ICLR) (2017)
6. Kuprieiev, R., Petrov, D., Pachhai, S., Redzyński, P., da Costa-Luis, C., Schepanovski, A., Rowlands, P., Shcheklein, I., Orpinel, J., Santos, F., Taskaya, B., Sharma, A., Zhanibek, Gao, Hodovic, D., Grigorev, A., Earl, nik123, Dash, N., Vyshnya, G., maykulkarni, Hora, M., Vera, Mangal, S., Baranowski, W., Wolff, C., Maslakov, A., Khamutov, A., Benoy, K.: Dvc: Data version control - git for data & models (May 2021). https://doi.org/10.5281/zenodo.4733984
7. Mo, Y., Chen, Y., Peng, L., Shi, X., Zhu, X.: Simple self-supervised multiplex graph representation learning. In: Proceedings of the 30th ACM International Conference on Multimedia. MM '22 (2022). https://doi.org/10.1145/3503161.3547949
8. Park, C., Kim, D., Han, J., Yu, H.: Unsupervised attributed multiplex network embedding. In: AAAI '20 (2020). https://doi.org/10.1609/aaai.v34i04.5985
9. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: KDD '14 (2014). https://doi.org/10.1145/2623330.2623732
10. Ren, Y., Liu, B., Huang, C., Dai, P., Bo, L., Zhang, J.: Heterogeneous deep graph infomax. ArXiv **abs/1911.08538** (2019)
11. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: Line: Large-scale information network embedding. In: WWW. ACM (2015)
12. Thakoor, S., Tallec, C., Azar, M.G., Munos, R., Veličković, P., Valko, M.: Bootstrapped representation learning on graphs. In: ICLR 2021 Workshop on Geometrical and Topological Representation Learning (2021)
13. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph Attention Networks. International Conference on Learning Representations (2018)
14. Veličković, P., Fedus, W., Hamilton, W.L., Liò, P., Bengio, Y., Hjelm, R.D.: Deep Graph Infomax. In: International Conference on Learning Representations (2019)
15. Wang, X., Ji, H., Shi, C., Wang, B., Ye, Y., Cui, P., Yu, P.S.: Heterogeneous graph attention network. In: WWW '19 (2019). https://doi.org/10.1145/3308558.3313562
16. Wang, X., Liu, N., Han, H., Shi, C.: Self-supervised heterogeneous graph neural network with co-contrastive learning. In: KDD '21 (2021). https://doi.org/10.1145/3447548.3467415
17. Wu, C.Y., Beutel, A., Ahmed, A., Smola, A.J.: Explaining reviews and ratings with paco: Poisson additive co-clustering. In: WWW '16 (2016). https://doi.org/10.1145/2872518.2889400
18. You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., Shen, Y.: Graph contrastive learning with augmentations. In: NeurIPS '20 (2020)
19. Yu, P., Fu, C., Yu, Y., Huang, C., Zhao, Z., Dong, J.: Multiplex heterogeneous graph convolutional network. In: KDD '22 (2022). https://doi.org/10.1145/3534678.3539482
20. Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., Wang, L.: Deep Graph Contrastive Representation Learning. In: ICML Workshop on Graph Representation Learning and Beyond (2020)

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63778-0_1