MULTIPLEX GRAPH NEURAL NETWORKS FOR HETEROGENEOUS GRAPHS

by

Joshua Melton

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Computing & Information Systems

Charlotte

2025

Approved by:

_____
Dr. Siddharth Krishnan

_____
Dr. Razvan Bunescu

_____
Dr. Gabriel Terejanu

_____
Dr. Ahmed Helmy

_____
Dr. Shannon Reid

ABSTRACT

JOSHUA MELTON. Multiplex graph neural networks for heterogeneous graphs.
(Under the direction of DR. SIDDHARTH KRISHNAN)

Graph neural networks (GNNs) have become effective learning techniques for many downstream network mining tasks including node and graph classification, link prediction, and network reconstruction. However, most GNN methods have been developed for homogeneous networks with only a single type of node and edge. Heterogeneous graphs are vital because they naturally model complex, real-world systems by representing diverse types of entities and the multiple kinds of relationships connecting them, which is essential for tasks like social network analysis, biological network mapping, and cybersecurity applications.

To address these gaps, this work introduces a multiplex graph neural network for heterogeneous graphs. To model heterogeneity, we represent graphs as multiplex networks consisting of a set of layer graphs and a coupling graph that links node instantiations across multiple layers. We parameterize layer-specific representations of nodes and design a novel coupling attention mechanism that models the importance of multi-relational contexts for different types of nodes and edges in heterogeneous graphs. We further develop two complementary coupling structures: node invariant coupling suitable for node- and graph-level tasks, and node equivariant coupling suitable for link-level tasks.

We conduct extensive experiments on real-world datasets for link prediction in both transductive and inductive contexts and for graph classification that demonstrate the superior performance of muxGNN over state-of-the-art heterogeneous GNNs. In addition, we show that muxGNN's coupling attention discovers interpretable connections between different relations in heterogeneous networks.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor, Dr. Siddharth Krishnan for introducing me to the wonderful world of graph machine learning and for his guidance, mentorship, and patience throughout my Ph.D studies. I am also grateful to the other member of my dissertation committee, Dr. Gabriel Terejanu, Dr. Razvan Bunescu, Dr. Ahmed Helmy, and Dr. Shannon Reid for their feedback and support throughout this process.

TABLE OF CONTENTS

LIST OF FIGURES vi

CHAPTER 1: INTRODUCTION 1

1.1. Problem Statement 3

CHAPTER 2: LITERATURE REVIEW 5

2.1. Background 5

2.2. Heterogeneous Graph Embedding Methods 6

2.2.1. Type-based Embedding 8

2.2.2. Relation-based Embedding 12

2.2.3. Metapath-based Embedding 18

2.3. Temporal Graph Neural Networks 21

2.3.1. Discrete-Time Temporal GNNs 22

2.3.2. Continuous-Time Temporal GNNs 22

CHAPTER 3: METHODOLOGY 24

3.1. Preliminaries 24

3.2. muxGNN Framework 26

3.2.1. Multiplex Spectral Convolution 26

3.2.2. Layer Graph Representation 30

3.2.3. Coupling Graph Attention 31

3.2.4. Model Optimization 34

3.3. Bipartite Node Labeling 36

CHAPTER 4: SOCIAL NETWORK APPLICATIONS 39

4.1. Hate Speech Detection 39

4.1.1. Weak Supervised Training 41

4.2. Link Prediction 43

4.2.1. Experimental Setup 43

4.2.2. Results & Discussion 45

4.3. Stance Labeling 46

4.3.1. Data Collection 47

4.3.2. Heuristic Labeling 48

4.3.3. GNN Stance Labeling 51

4.3.4. Results & Discussion 52

CHAPTER 5: BIOLOGICAL NETWORK APPLICATIONS 59

5.1. Link Prediction 59

5.1.1. Experimental Setup 59

5.1.2. Results & Discussion 61

5.2. Coupling Graph Attention Analysis 62

5.2.1. Ablation Study 63

5.2.2. Tissue PPI Layer Coupling 64

CHAPTER 6: DATA PROVENANCE NETWORK APPLICATIONS 66

6.1. Graph Classification 66

6.1.1. Experimental Setup 67

6.1.2. Results & Discussion 68

6.2. Advanced Persistent Threats 69

CHAPTER 7: CONCLUSION 71

7.1. Summary of Contributions 71

7.2. Limitations 72

7.3. Future Research Directions 73

7.3.1. Temporal Multiplex Graph Neural Networks 73

7.3.2. Oversmoothing Problem 74

7.3.3. Theoretical Expressive Power 75

7.3.4. Multi-modality & LLM Integration 76

REFERENCES 77

LIST OF FIGURES

FIGURE 2.1: Schematic of a heterogeneous academic citation network. 6

FIGURE 2.2: Illustration of GATNE models. GATNE-T only uses network structure information while GATNE-I considers both structure information and node attributes. The output layer of heterogeneous skip-gram specifies one set of multinomial distributions for each node type in the neighborhood of the input node v. In this example, $\mathcal{V} = \mathcal{V}1 \cup \mathcal{V}2 \cup \mathcal{V}3$ and $K1, K2, K3$ specify the size of v's neighborhood on each node type respectively [30]. 9

FIGURE 2.3: (a) Overall architecture of HetGNN: it first samples fixed-sized heterogeneous neighbors for each node, next encodes each node content embedding via NN-1, then aggregates content embeddings of the sampled heterogeneous neighbors through NN-2 and NN-3, and finally optimizes the model via a graph context loss. (b) NN-1: node heterogeneous content encoder; (c) NN-2: type-based neighbor aggregator; (d) NN-3: heterogeneous type combination [17]. 10

FIGURE 2.4: Diagram for computing the update of a single graph node/entity (red) in the R-GCN model. Activations (d-dimensional vectors) from neighboring nodes (dark blue) are gathered and then transformed for each relation type individually (for both in- and outgoing edges). The resulting representation (green) is accumulated in a (normalized) sum and passed through an activation function (such as the ReLU). This per-node update can be computed in parallel with shared parameters across the whole graph [15]. 14

FIGURE 2.5: Architecture of Heterogeneous Graph Transformer. Given a sampled heterogeneous subgraph with t as the target node, s_1 & s_2 as source nodes, HGT takes its edges $e_1 = (s_1, t)$ & $e_2 = (s_2, t)$ and their corresponding relations $\langle \tau(s_1), \phi(e_1), \tau(t) \rangle$ & $\langle \tau(s_2), \phi(e_2), \tau(t) \rangle$ as input to learn a contextualized representation $H^{(L)}$ for each node, which can be used for downstream tasks [19]. 15

FIGURE 2.6: The overall framework of HAN. (a) All types of nodes are projected into a unified feature space and the weight of meta-path based node pair can be learned via node-level attention. (b) Joint learning the weight of each meta-path and fuse the semantic-specific node embedding via semantic-level attention. (c) Calculate the loss and end-to-end optimization for the proposed HAN [18]. 19

FIGURE 3.1: Schematic of a multilayer representation of a heterogeneous graph. (a) A multiplex network consisting of relation-specific layer graphs with separate node instantiations coupled across relations. (b) Node invariant coupling graph structure in which a node's relation-specific representations are directly coupled to the supra-node to produce a singular representation of a node in a heterogeneous graph. (c) Node equivariant coupling graph structure in which node-relation instantiations are coupled with one another to produce a set of representations characterizing a node in a heterogeneous graph. 25

FIGURE 3.2: Overall framework of the muxGNN model. (a) The relation-specific representation of a node generated by local neighborhood aggregation across each relation in which a node participates. (b) The node-relation pair representations of a supra-node are fused through coupling weights learned via node invariant coupling attention (c) Node equivariant attention learns sets of coupling weights for a node from the perspective of each participating relation as the dominant layer with the associated relative perturbations of the other relations. 29

FIGURE 4.1: The model topology depicted illustrates the CNN/GRU feature component and the tunable classifier. These models are combined into an ensemble model for predicting the presence of hate speech in online posts. 40

FIGURE 4.2: Weak supervised bounds algorithm that calculates a lower and upper bound for each class. 42

FIGURE 4.3: Timeline of the number of Twitter conversations about climate change started each day. 49

FIGURE 4.4: Timeline of the number of Twitter conversations about gun control started each day. 50

FIGURE 5.1: Ablation study and dimension analysis of muxGNN's coupling graph attention. 63

FIGURE 5.2: Distributions of coupling attention values for each protein in the (a) Brain, (b) CNS, (c) Leukocyte, and (d) NS layers in the Tissue-PPI dataset. 64

CHAPTER 1: INTRODUCTION

Graphs are a powerful data structure for modeling relational information between a set of entities (nodes) and their interactions (edges). Many real-world domains generate relational data that can be viewed as graphs, including social networks, bibliographic and citation networks, traffic networks, and protein interaction networks. In the past several years, research on using machine learning to analyze graphs has received extensive attention due to the ubiquity of graphs across application areas and the expressive power of graphs to model non-Euclidean data. Neural architectures such as recurrent neural networks (RNNs), convolutional neural networks (CNNs), and transformers operate on regularly connected data, such as 1D linear sequences, 2D grids, or complete graphs, but such architectures are unsuited for handling the arbitrary connectedness that occurs within graphs. To address these shortcomings, the field of geometric deep learning [1] seeks to extend deep neural models to non-Euclidean data, and over the past several years, graph neural networks (GNNs) have emerged as highly effective deep learning tools for analysis on graphs for tasks such as node classification [2], link prediction [3, 4], community detection [5], and graph classification.

Most GNNs operate on homogeneous networks [6–10], where there are no distinctions between different types of nodes or different types of connections in the graph. Heterogeneous graphs, which are capable of composing different types of entities and relations are becoming increasingly common in many real-world applications due to their ability to encode richer semantic and structural information in real-world data, and with an ever-increasing amount of publicly available graph data, work on extending GNNs to heterogeneous graphs has become a burgeoning research area.

Learning low-dimensional representations of graphs and graph substructures is a key problem for many graph analysis tasks. Traditionally, matrix (e.g. adjacency matrix) factorization methods [11,12] have been utilized to generate latent dimension features for a graph, but the limitations of matrix factorization methods for learning on heterogeneous graphs are particularly relevant due to the computational costs associated with decomposing a large-scale matrix representation of a heterogeneous graph. Furthermore, many such methods suffer from statistical performance drawbacks when applied to heterogeneous networks [13, 14]. For homogeneous graphs, such methods have largely been superseded by graph neural networks that learn an inductive transformation function that maps an input feature space to a lower-dimension space while preserving the structural and semantic information contained in the graph. Although there have been abundant studies for embedding homogeneous networks consisting of only a single type of node and edge, such techniques cannot be directly applied to heterogeneous graphs due to the heterogeneity of the data. In particular, i) the structure of heterogeneous networks is typically dependent on various relations [15] or metapaths [16], which suggests that the structure of heterogeneous graphs can vary considerably depending on which relations are considered or how metapaths are constructed; ii) different node and edge types can have different attributes which are generally drawn from distinct distributions, which must be overcome when designing methods for heterogeneous graph representation learning [17]. This is especially relevant for the design of GNNs for heterogeneous networks as such architectures must aggregate, transform, and combine feature information from multiple distributions [18,19]; and iii) heterogeneous graphs are often highly application dependent, so developing general representation learning methods for such networks is challenging. For example, metapath-based methods often require significant domain expertise to design appropriate metapaths that capture the relevant semantic and structural information contained in heterogeneous graphs. In addition, the statistical properties of

heterogeneous networks can vary greatly between different application areas such as cybersecurity [20, 21], knowledge graphs [15, 22], or recommendation systems [4, 23].

To address the aforementioned issues, numerous heterogeneous embedding methods have been proposed, many of which have proven successful in real world applications such as malware detection [21, 24–26], recommender systems [4, 23], and healthcare systems [27, 28]. A key property in these proposed methods is how the heterogeneous network is decomposed into separate semantic and structural contexts and how these contexts are then transformed and combined during the embedding process. Despite this similarity of approaches in how many heterogeneous graph framework handle the challenge of the multifaceted nature of structures in heterogeneous graphs, the literature to date lacks a common theoretical basis for decomposing heterogeneous graphs in multiple contexts and lacks a shared language for understanding and comparing methodologies.

1.1 Problem Statement

Graph embedding methods face many difficulties due to the semantic and structural heterogeneity of heterogeneous graphs. Furthermore, many existing heterogeneous GNN models in the literature lack a formal theoretical basis and instead rely on an intuitive extension of homogeneous GNN methods to heterogeneous networks and empirical results. As a result, much of the literature on heterogeneous graph neural networks lacks a common theoretical language for understanding and comparing proposed methods.

To remedy this problem, we develop a multiplex representation of heterogeneous networks as a unifying idea that i) allows for derivation of many heterogeneous graph embedding frameworks from first principles of graph convolutions as an approximation of signal convolution over the graph Laplacian; and 2) encapsulates many of the most common and successfully heterogeneous GNN frameworks from the literature. Based on the definition of a multiplex convolution, we present muxGNN as

a heterogeneous GNN framework that can produce rich semantic embeddings from heterogeneous networks suitable for node-level, link-level, or graph-level downstream tasks. We demonstrate the efficacy and interpretability of our method in three domain areas: social networks, biological networks, and data provenance networks.

CHAPTER 2: LITERATURE REVIEW

2.1 Background

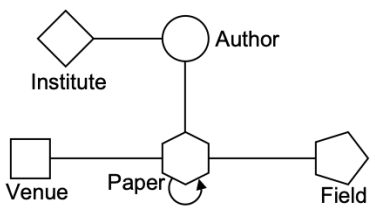
In this section, we formally introduce definitions of heterogeneous and attributed graphs as well as several important concepts in heterogeneous graph representation learning: relations, metapaths, and relation-specific subgraphs. We further define a formal statement of heterogeneous graph embedding. An example schema of heterogeneous academic citation network is illustrated in Figure 2.1

Definition 1. Heterogeneous Graph: A heterogeneous graph is defined as a graph $G = (\mathcal{V}, \mathcal{E}, T_v, T_e)$ where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges. Each node $v \in \mathcal{V}$ is associated with a type mapping function $\phi(v) : \mathcal{V} \rightarrow T_v$ and each edge $e \in \mathcal{E}$ is associated with a mapping $\psi(e) : \mathcal{E} \rightarrow T_e$, where T_v and T_e are the sets of node types and edge types, respectively.

Definition 2. Attributed Graph: A graph is an attributed graph if the nodes $\mathcal{V}$ or edges $\mathcal{E}$ are associated with node or edge attributes. These attributes are given by node attribute function $\alpha : \mathcal{V} \rightarrow \mathcal{A}$ with $\mathcal{A} = \{\boldsymbol{a_i} \,|\, v_i \in \mathcal{V}\}$, where $\boldsymbol{a_i}$ is the feature vector associated with node v_i, and edge attribute function $\omega : \mathcal{E} \rightarrow \mathcal{W}$, with $\mathcal{W} = \{\boldsymbol{w_i} \,|\, e_i \in \mathcal{E}\}$, where $\boldsymbol{w_i}$ is the feature vector associated with edge e_i.

Definition 3. Relation: For an edge $e = (u, v)$ in a heterogeneous graph, we define the relation triplet r as $\langle \phi(u), \psi(e), \phi(v) \rangle$. Each edge is associated with a mapping $\rho(e) : \mathcal{E} \rightarrow R$, where R is the set of all relations present in the graph. We can assume that there may exist multiple relations between different types of nodes in many real-world heterogeneous graphs.

For example, in a user-item heterogeneous graph. The statement "a user buys an



(a) The schema of heterogeneous academic networks

Author	*is_(first/last/other)_author_of*	Paper
Author	*is_affiliated_with*	Institute
Paper	*is_published_(conf/journal)_at*	Venue
Paper	*has_(L_1−L_5)_field_of*	Field
Paper	*has_citation_to*	Paper

(b) The meta relations of heterogeneous academic networks

Figure 2.1: Schematic of a heterogeneous academic citation network.

item" corresponds to the relation triplet $r = \langle \text{user},\ \text{buys},\ \text{item} \rangle$, where $\phi(u) =$ "user", $\psi(e) =$ "buys", and $\phi(v) =$ "item".

Definition 4. Metapath: A metapath P of length l is defined as a path $v_1 \xrightarrow{r_1} v_2 \xrightarrow{r_2} \ldots \xrightarrow{r_1} v_{l+1}$, which describes a composite relation $r_1 \circ r_2 \circ \ldots \circ r_1$ between two nodes v_1 and v_{l+1}, where $\circ$ denote the composition operator on relations.

Definition 5. Relation Subgraph: Given a heterogeneous graph, $G = (\mathcal{V}, \mathcal{E}, T_v, T_e)$, a relation subgraph G_r is a subgraph of G that contains all nodes and edges with relation r. That is, each relation subgraph G_r's node set is the subset of nodes such that $\mathcal{V}_r = \{v \subset \mathcal{V} \,|\, \phi(v) \in r\}$, and its edge set is the subset of $\mathcal{E}$ such that $\mathcal{E}_r = \{e \in \mathcal{E} \,|\, \rho(e) \in r\}$. The adjacency matrix $\mathbf{A}_r \in \mathbb{R}^{|\mathcal{V}_{\phi(u)}| \times |\mathcal{V}_{\phi(v)}|}$, where $\mathbf{A}_r[u, v] = 1$ if $\langle \phi(u), \psi(e), \phi(v) \rangle = r$, otherwise $\mathbf{A}_r[u, v] = 0$.

Definition 6. Heterogeneous Graph Embedding: [13] A heterogeneous graph embedding method aims to learn a transformation function $f : \mathcal{V} \to \mathbb{R}^d$ that maps nodes in a heterogeneous graph to a low-dimensional Euclidean space with $d \ll |\mathcal{V}|$ while preserving the structural and semantic information in the graph.

2.2 Heterogeneous Graph Embedding Methods

The variety of node and link types in heterogeneous graphs creates different graph structures and semantic information augmented by node and edge attributes. In order to learn representation methods that can produce embeddings that capture

the heterogeneous structures and rich attributes, we need to consider heterogeneous graphs from several aspects during the embedding process. In the literature, this has typically been accomplished by decomposing a heterogeneous graph into a number of separate views, to which a specific embedding model is applied [15–19, 29–31]. These multiple embeddings are then aggregated by some means to produce a rich embedding of heterogeneous graphs and graph structures that captures the multi-faceted nature of nodes in such graphs. Based on the aforementioned challenges, we categorize the existing methods into three categorized based on how each method decomposes the heterogeneous graph for embedding: (1) *Type-based decomposition.* The methods belonging to this category break down a heterogeneous graph separately based on the node and edge types present in the network. To each node and edge type, a separate transformation function is applied during the embedding process. These methods primarily focus on capturing the attribute distributions of node types and capturing the structural information defined by the edge types in the network. These methods can be especially useful for heterogeneous graphs that are only node or edge heterogeneous, such as a social interaction networks that describes different types of interactions between users (2) *Relation-based decomposition.* The methods incorporate more information beyond the separate node and edge types by decomposing heterogeneous graphs along their relations. By jointly considering relation triples as the found, these methods are capable of capturing more nuanced information about the semantics of how different node and edge types interact along specific relations. These methods are generally applicable to heterogeneous graphs with multiple node and edges types and are particularly relevant for certain application areas such as knowledge graph analysis. (3) *Metapath-based decomposition.* Models in this category utilize manually curated metapaths to define composite relations in heterogeneous graphs. These methods do not directly decompose the heterogeneous graph itself as do the prior two categories. Instead, these methods derive new views of the

heterogeneous graph based on the composite meta-relations. As such, these methods often accompanied by type-based models that can generate representations from the metapath derived graphs.

2.2.1 Type-based Embedding

Unlike homogeneous graphs, heterogeneous networks consist of different types of nodes and edges that each contain different semantics. To distinguish the various types in heterogeneous graphs, these methods utilize separate transformations to project nodes to different metric spaces rather than the unified latent space that is used for homogeneous graph embedding methods. A representative example of type-based representation learning is MNE [29]. Each node in the network has a base node embedding $\mathbf{b}_i$ that is specific to the node's type and is shared across all edge types in which the node participates. This common embedding serves as a bridge to transfer information across different edge types. To capture the edge heterogeneity of the network, each node's base embedding is augmented by a series of low-dimensional embeddings for each edge type. MNE produces a set of embeddings for a given node that each correspond to a particular edge type. For a node i, its embedding with respect to edge type e is given by:

$$\mathbf{h}_i^e = \mathbf{b}_i + w^e \mathbf{X}^{e\top} \mathbf{u}_i^e \tag{2.1}$$

where $\mathbf{b}_i$ is the base node embedding and $\mathbf{u}_i^e$ is the edge type specific embedding. $\mathbf{X}^e$ is a trainable transformation matrix that aligns the base node and edge embeddings to the same latent space, and w^e denotes the overall importance of edge type e. MNE directly optimizes the base node embeddings, edge embeddings, and transformation matrices by using a skip-gram technique to represent information of multi-type random walks on the network.

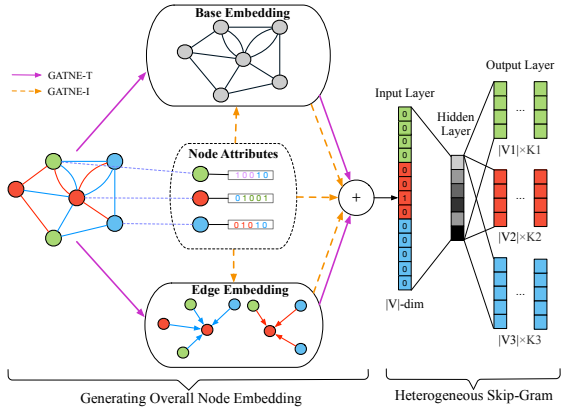


Figure 2.2: Illustration of GATNE models. GATNE-T only uses network structure information while GATNE-I considers both structure information and node attributes. The output layer of heterogeneous skip-gram specifies one set of multinomial distributions for each node type in the neighborhood of the input node v. In this example, $\mathcal{V} = \mathcal{V}1 \cup \mathcal{V}2 \cup \mathcal{V}3$ and $K1, K2, K3$ specify the size of v's neighborhood on each node type respectively [30].

Extending the work of MNE to incorporate GNNs and attention mechanisms, GATNE [30] computes edge-type specific embeddings for nodes in an inductive manner that also incorporates node attributes. Instead of directly learning a base node embedding, GATNE projects a node's initial features to a latent space via a node type specific transformation matrix. To compute the edge type specific embeddings, multiple graphSAGE GNNs are used to aggregate embeddings from a node's neighbors along each edge type into a set of low-dimensional edge embeddings that capture the local structure of a node long each edge axis. A semantic attention mechanisms is then employed to dynamically compute the specific importances of each edge type to the particular target edge type, which is then combined with the base node embedding similar to MNE to compute a final representation for node i along edge type e as:

$$\mathbf{h}_i^e = \mathbf{b}_i + w^e \mathbf{X}^{e\top} \mathbf{U}_i \mathbf{a}_i^e \tag{2.2}$$

where $\mathbf{b}_i$ is the base node embedding, $\mathbf{U}_i$ is the set of edge type embeddings, and $\mathbf{a}_i^e$

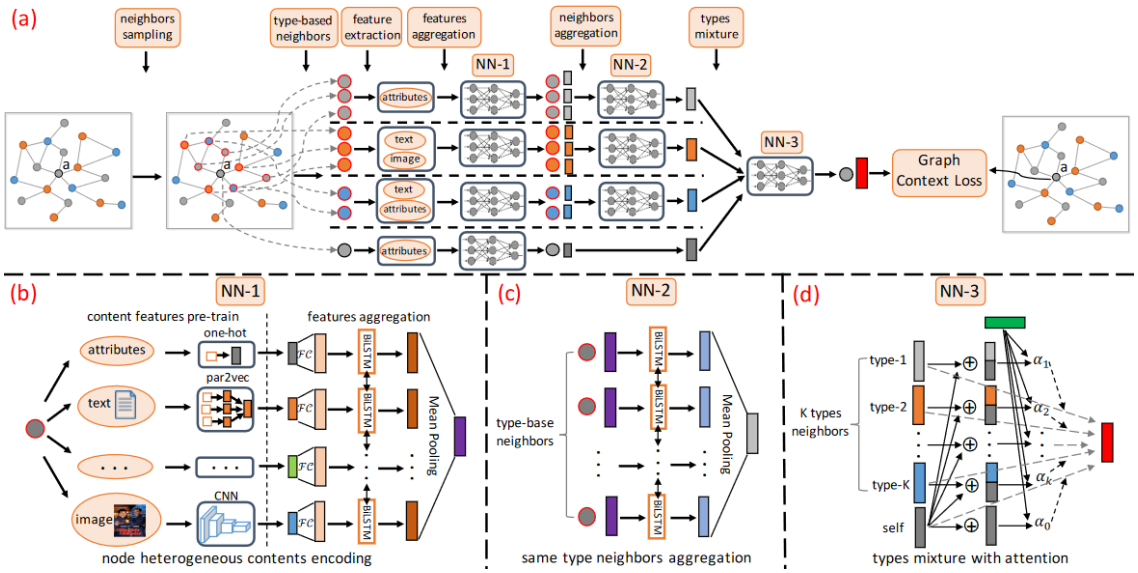


Figure 2.3: (a) Overall architecture of HetGNN: it first samples fix-sized heterogeneous neighbors for each node, next encodes each node content embedding via NN-1, then aggregates content embeddings of the sampled heterogeneous neighbors through NN-2 and NN-3, and finally optimizes the model via a graph context loss. (b) NN-1: node heterogeneous content encoder; (c) NN-2: type-based neighbor aggregator; (d) NN-3: heterogeneous type combination [17].

is the dynamic attention weights for aggregating the edge type embeddings. $\mathbf{X}^e$ is the transformation matrix that aligns the based and edge embeddings, akin to MNE.

HetGNN [17] is another representative example of type-based embedding of heterogeneous graphs that is particularly focused on graphs with complex and multi-modal node attributes. Unlike MNE and GATNE, which employ a schema of base node type embedding augmented by small edge type embeddings. HetGNN consists of three stages: feature aggregation, neighbor aggregation, and type aggregation. The feature aggregation step is designed to learn fused embeddings from different attributes:

$$f_1(v) = \frac{\sum_{i \in C_v} \overrightarrow{\text{LSTM}\{\mathcal{FC}(\mathbf{h}_i)\}} \bigoplus \overleftarrow{\text{LSTM}\{\mathcal{FC}(\mathbf{h}_i)\}}}{|C_v|} \tag{2.3}$$

where C_v is the type or distribution of node v's attributes, and $\mathbf{h}_i$ is the i-th attributes of node v. A bidirectional Long Short-Term Memory (bi-LSTM) is used to combine the embeddings learned by the attribute encoder $\mathcal{FC}$. The neighbor aggregation step

aggregates information about nodes of the same type using a bi-LSTM to encode structural information:

$$f_2^t(v) = \frac{\sum_{u \in \mathcal{N}_t(v)} \overrightarrow{\text{LSTM}\{f_1(u)} \bigoplus \overleftarrow{\text{LSTM}\{f_1(u)}}{|\mathcal{N}_t(v)|} \tag{2.4}$$

where $\mathcal{N}_t(v)$ is the first-order neighbors of node v with node type t. The final type aggregation step utilizes an attention mechanism to implicitly compute the importances of each type's neighbors to node v as:

$$\mathbf{h}_v = \alpha_{v,v} f_1(v) + \sum_{t \in T_v} \alpha_{v,t} f_2^t(v) \tag{2.5}$$

where $\mathbf{h}_v$ is the final embedding of node v. T_v is the set of node types in the network, and α denotes the attention weight of each node type computed for node v. Like MNE and GATNE, a heterogeneous skip-gram loss is used to train the embedding models.

Other structural embedding methods like PMNE and MVE propose methods for projecting multiple views of a heterogeneous network to a continuous vector space. PMNE uses a node2vec embedding of a homogeneous version of the graph to train a merging function for the heterogeneous embeddings. MVE, similar to GATNE and HetGNN, uses an attention mechanism as its merging function. Neither of these methods make use of node attributes, which are generally only considered in GNN-based methods like GATNE and HetGNN. For link prediction tasks, methods such as MNE and GATNE are suitable as they generate embeddings of nodes with respect to particular edge types in the graph. Methods that produce a single consolidated embedding for a node like HetGNN are better suited for node-level tasks.

In summary, type-based methods decompose heterogeneous graphs separately along their node types and edge types. Each type is projected to a distinct latent space and merging/alignment functions are employed to combine the separate feature rep-

resentations into a unified embedding of a node.

2.2.2 Relation-based Embedding

In a similar vein to the type-based methods, relation-based methods approach the problem of heterogeneity in heterogeneous graphs by treating each relation in the graph in different metric spaces, as opposed to a unified metric space. One representative work is PME [32] that uses a relation-specific transformation matrix to transform nodes into different metric spaces. By doing so, nodes that are connected to one another across different relations can be close to each other in different metric spaces, allowing PME to capture heterogeneous semantic node contexts. The distance function is defined as follows

$$S_r(v_i, v_j) = w_{ij}||\mathbf{M}_r\mathbf{h}_i - \mathbf{M}_r\mathbf{h}_j||_2 \tag{2.6}$$

where $\mathbf{h}_i$ and $\mathbf{h}_j \in \mathbb{R}^{d\times 1}$ are the node embeddings of nodes i and j, respectively. $\mathbf{M}_r \in \mathbb{R}^{d\times d}$ is the projection matrix for relation r, and w_{ij} represents the edge weight for the link between node i and j on relation r. Eq. 2.6 may be viewed as a metric learning function:

$$||\mathbf{M}_r(\mathbf{h}_i - \mathbf{h}_j)||_2 = \sqrt{(\mathbf{h}_i - \mathbf{h}_j)^\top \mathbf{M}_r^\top \mathbf{M}_r(\mathbf{h}_i - \mathbf{h}_j)} \tag{2.7}$$

where $\mathbf{M}_r^\top\mathbf{M}_r \in \mathbb{R}^{d\times d}$ is the metric matrix of Mahalanobis distance [33]. PME thus incorporates the various relations across nodes when minimizing the distance between them, and utilizes a relation triplet margin loss that makes node pairs connected by relation r closer to each other than other nodes pairs without relation r.

Other methods have defined similar relation-specific matrices to capture network heterogeneity, but rather than minimize the distance between node pairs along relations, EOE [34] and HeGAN [35] use the relation transformation matrix $\mathbf{M}_r$ to maximize the similarity between two nodes, formulated as:

$$S_r(v_i, v_j) = \frac{1}{1 + \exp(-\mathbf{h}_i^\top \mathbf{M}_r \mathbf{h}_j)} \tag{2.8}$$

EOE was particularly designed for coupled heterogeneous graphs consisting of two distinct but related sub-graphs. Links are divided into intra-graph and inter-graph links. Intra-graph links connect nodes of the same type and inter-graph links connect nodes with different types. Eq. 2.8 is used as a similarity function to maximize the closeness of two nodes along inter-graph links. HeGAN employs a generative adversarial network (GAN) to compute embeddings for heterogeneous graphs and uses Eq. 2.8 as the discriminator to evaluate node embeddings produced by the generator.

In addition to distance minimization and similarity maximization methods discussed prior, methods such as AspEM [36] and HEER [37] seek to maximize the probability of existing connections of particular relations via a relational similarity probability function.

A representative example of GCN-based relation methods for heterogeneous graphs is R-GCN [15]. R-GCN decomposes a heterogeneous graph into relational subgraphs and applies a GCN layer with separate transformation matrices to each relational layer:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\sum_{r \in R} \sum_{j \in \mathcal{N}_r(i)} \frac{1}{c_{i,r}} \mathbf{W}_r^l \mathbf{h}_j^{(l)} + \mathbf{W}_0^{(l)} \mathbf{h}_i^{(l)}\right) \tag{2.9}$$

where $\mathcal{N}_r(i)$ is the set of node neighbors of i along relation r, and $c_{i,r}$ is a problem-specific normalization constant. Eq. 2.9 accumulates transformed feature vectors of neighboring nodes through a normalized sum, similar to regular GCNs. Distinctly for heterogeneous graphs R-GCN introduces relation-specific transformations to account for the heterogeneous link structure of such networks. Like most GNNs, R-GCN can stack multiple layers to incorporate information from higher order neighborhoods of nodes and also includes a special self-loop message $\mathbf{W}_0 \mathbf{h}_i$ that preserves information

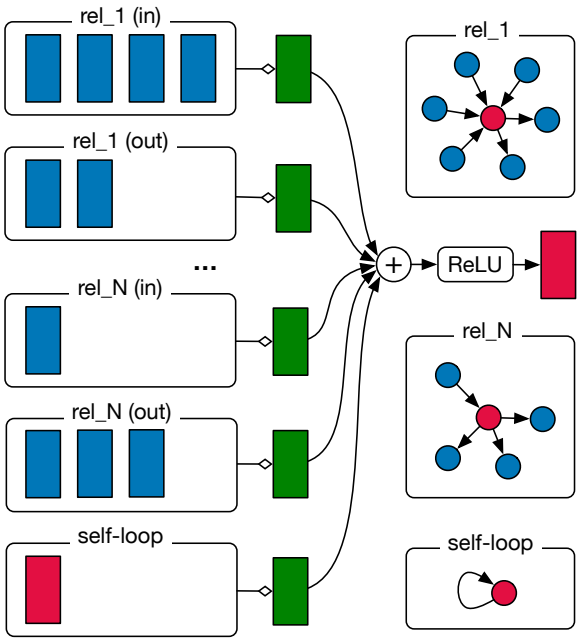


Figure 2.4: Diagram for computing the update of a single graph node/entity (red) in the R-GCN model. Activations (d-dimensional vectors) from neighboring nodes (dark blue) are gathered and then transformed for each relation type individually (for both in- and outgoing edges). The resulting representation (green) is accumulated in a (normalized) sum and passed through an activation function (such as the ReLU). This per-node update can be computed in parallel with shared parameters across the whole graph [15].

about a node's own features across each relation. R-GCN also addresses one of the major drawbacks to relation-based methods, in that the number of transformation scales with the number of relations—which for certain domains, like knowledge graphs, can grow into the hundreds or thousands of relations. To offset this scaling issue, R-GCN proposes two regularization methods which help to prevent overfitting on rare relations and reduce the growth in the number of parameters. In *basis*-decomposition, each $\mathbf{W}_r$ is defined as follows:

$$\mathbf{W}_r^{(l)} = \sum_{b=1}^{B} a_{rb}^{(l)} \mathbf{V}_b^{(l)} \tag{2.10}$$

i.e., each relation transformation matrix is the linear combination of basis transformations $\mathbf{V}_b$ with coefficients a_{rb}. In this way, only the coefficients depend on the number

of relations and the basis transformations are defined by a fixed number of bases. In *block-diagonal*-decomposition. The relation transformation matrices are defined as a sum over a set of low-dimensional matrices. Basis decomposition therefore forms a sort of weight sharing across relations and block decomposition encodes a coupling of sets of variables into groups that are more tightly coupled within the graph than without. Importantly, these regularization techniques are not specific to R-GCN and can be adapted to many of the relation-based GNN methods present in the literature.

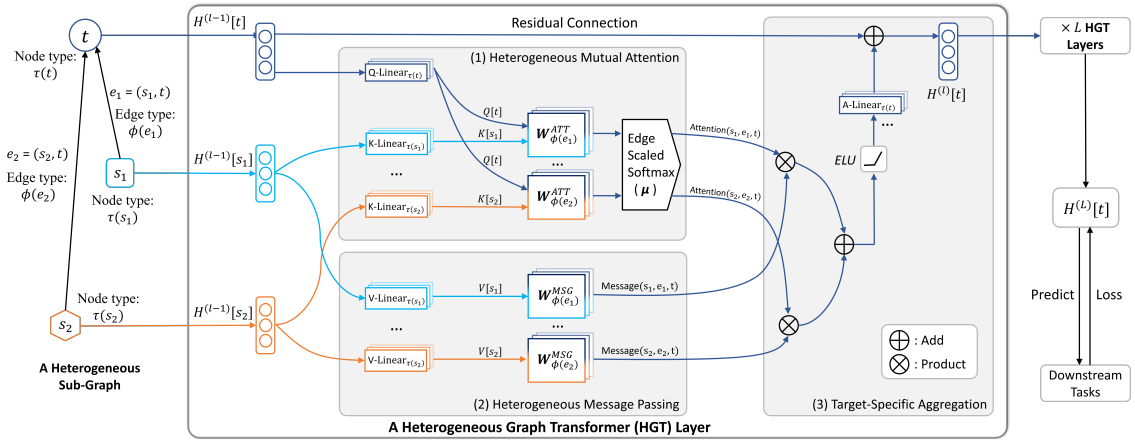


Figure 2.5: Architecture of Heterogeneous Graph Transformer. Given a sampled heterogeneous subgraph with t as the target node, s_1 & s_2 as source nodes, HGT takes its edges $e_1 = (s_1, t)$ & $e_2 = (s_2, t)$ and their corresponding relations $\langle \tau(s_1), \phi(e_1), \tau(t) \rangle$ & $\langle \tau(s_2), \phi(e_2), \tau(t) \rangle$ as input to learn a contextualized representation $H^{(L)}$ for each node, which can be used for downstream tasks [19].

Another important class of relational GNN methods for embedding heterogeneous graphs employs a transformer-style architecture instead of a GCN model. A representative example of this class of models is Heterogeneous Graph Transformer (HGT) [19]. Its embedding process can be broken up into three components: heterogeneous mutual attention, heterogeneous message passing, and target-specific aggregation. In the first step, heterogeneous mutual attention is calculated between source and target node pairs grounded by their relation triplets. Each target node v is mapped to a Query vector, and the source node u is mapped to a key vector. The dot product between the query and key vectors is calculated as the attention score.

Different from standard transformers where a single set of projections is used for all words, for heterogeneous graphs each relation should have a distinct set of projection weights. To avoid the parameter scaling issues described above in the context of R-GCN, HGT further decomposes the individual attention projection matrices based on source node type, edge type, and target node type. Specifically, the attention score for an edge $e = (u, v)$ is given by:

$$\alpha(u, e, v) = \mathbf{W}^{k}_{\phi(u)}\mathbf{h}_u \mathbf{W}^{\text{ATT}}_{\psi(e)} \mathbf{W}^{q}_{\phi(v)}\mathbf{h}_v \cdot \frac{\mu_{\langle \phi(u), \psi(e), \phi(v) \rangle}}{\sqrt{d}} \tag{2.11}$$

where $\mathbf{W}^{k}_{\phi(u)}\mathbf{h}_u$ is the key projection of the source node and $\mathbf{W}^{q}_{\phi(v)}\mathbf{h}_v$ is the query projection of the target node. To account for the relational nature of heterogeneous graphs, wherein two nodes may interact across multiple edge types. An edge type specific weight $\mathbf{W}^{\text{ATT}}_{\psi(e)}$ is included to capture different semantic relations even between the same pair of nodes. Furthermore, since not all relations contribute equally to target nodes, a prior tensor μ is included to denote the general significance of each relation in the network. For a given target node, the attention vectors from all of its neighbors $\mathcal{N}(v)$ are gathered and normalized via softmax. Parallel to the attention computation, relation specific message passing is performed, similar to R-GCN where node features from each neighbor are transformed by a node type specific transformation matrix followed by an edge type specific projection matrix:

$$\mathbf{m}(u, e, v) = \mathbf{W}^{m}_{\phi(u)}\mathbf{h}_u \mathbf{W}^{\text{MSG}}_{\phi(e)} \tag{2.12}$$

Target-specific aggregation is then performed by combining the attention scores and messages for each neighbor node adjacent to the target and a final projection of target node's vector back to the target specific distribution, which is accomplished via a node type specific linear projection and residual connection:

$$\tilde{\mathbf{h}}_v = \sigma\left(\bigoplus_{u \in \mathcal{N}(v)} \alpha(u, e, v) \cdot \mathbf{m}(u, e, v)\right) \tag{2.13}$$

$$\mathbf{h}_v = \mathbf{W}_{\phi(v)}\tilde{\mathbf{h}}_v + \tilde{\mathbf{h}}_v \tag{2.14}$$

Like R-GCN and other GNN methods, HGT layers can be stacked in succession to incorporate information from higher order neighborhoods of nodes. Additionally, the attention and message passing functions can be made into multi-headed operations with separate smaller transformation matrices that are aggregated together, enabling HGT to capture more rich semantic and structural information. Notably, the combination of relational attention and stacking of multiple HGT layers allows the model to implicitly discover relevant metapaths for nodes without the need to explicitly define metapaths. Heterogeneous Graph Structural Attention Neural Network (HetSANN) [31] is another relational GNN method that uses type-aware attention layers to project a node and its neighbors to a shared latent space. Unlike HGT, which uses transformer query-key-value attention, HetSANN uses a type-aware attention layer to project a node and its neighbors to a common embedding space and then aggregates the adjacent node features using a GAT-style neural network layer.

In summary, relation-based methods develop heterogeneous graph representation learning techniques by decomposing heterogeneous graphs along their relation triplets. Using relations as the fundamental basis for graph analysis as opposed to individual separate node and edge types allows these methods to capture richer semantic and structural information about nodes. For efficiency purposes, some relational methods like HGT further decompose relations into node and edge type parameters, whereas others like R-GCN use regularization techniques to avoid the parameter scaling issues that can accompany relational models. These models serve as a middle ground between the type-based methods that may miss out on certain semantic nuances of in-

teractions between nodes and metapath-based methods which can potentially extract higher order semantic information but require significant work to manually define relevant and informative metapath schemata.

2.2.3 Metapath-based Embedding

Metapath-based approaches to representation learning on heterogeneous graphs aim to preserve the heterogeneity of structural and semantic information by constructing metapath-based neighbors when encoding nodes. These methods define composite relations derived from a sequence of individual relations in the heterogeneous graph that form the basis of how heterogeneous graphs are decomposed into separate contexts for embedding purposes.

Metapath2vec [16] and related node embedding techniques [38,39] rely on metapath-guided random walks to generate heterogeneous node contexts for embedding node in heterogeneous graphs. These methods use a shallow transductive model and do not incorporate node attribute information during the embedding process. GNN-based metapath methods for heterogeneous graph representation learning overcome these issues by leveraging heterogeneous message passing functions to learn and inductive transformation function that incorporates node attributes as well as structural information about their metapath defined neighbor contexts. One representative example of these type of methods is Heterogeneous Attention Network (HAN) [18].

HAN utilizes a hierarchical attention mechanism to capture both within-metapath structural and semantic information as well as cross-metapath heterogeneous contexts of nodes in a two-stage embedding process. In the first stage, HAN uses a node-level self attention mechanism to learn the importances of neighboring nodes within a particular metapath as:

$$\alpha_{ij}^{m} = \frac{\exp(\sigma(\mathbf{a}_m^\top \cdot [\mathbf{h}_i' || \mathbf{h}_j']))}{\sum_{k \in \mathcal{N}_m(i)} \exp(\sigma(\mathbf{a}_m^\top \cdot [\mathbf{h}_i' || \mathbf{h}_j']))} \tag{2.15}$$

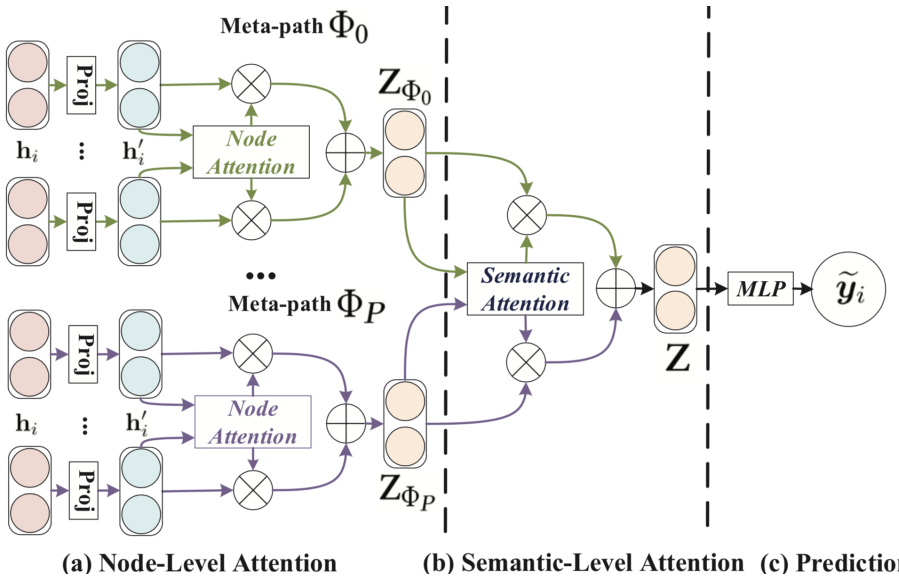


Figure 2.6: The overall framework of HAN. (a) All types of nodes are projected into a unified feature space and the weight of meta-path based node pair can be learned via node-level attention. (b) Joint learning the weight of each meta-path and fuse the semantic-specific node embedding via semantic-level attention. (c) Calculate the loss and end-to-end optimization for the proposed HAN [18].

where $\mathcal{N}_m(i)$ is the set of neighbors of node i along metapath m. α_{ij}^m is the attention score for neighbor node j to node i on metapath m. This attention weight is utilized in a GAT-style [9] message passing layer that aggregates the information about node i's neighbors on metapath m weighted by the attention score α_{ij}^m:

$$\tilde{\mathbf{h}}_\mathbf{i}^\mathbf{m} = \sigma\left(\sum_{j \in \mathcal{N}_m(i)} \alpha_{ij}^m \cdot \mathbf{h}_j\right) \tag{2.16}$$

where $\tilde{\mathbf{h}}_i^m$ is the learned embedding on node i for metapath m.

To capture the different semantic information encoded by different metapaths, HAN next uses a semantic level attention mechanism to compute the relative importances of each metapath for a particular node. Given a set of p metapaths $\{m_0, m_1, \ldots, m_p\}$, after the first metapath-specific node embedding step, a node i has p node embeddings $\{\tilde{\mathbf{h}}_i^{m_0}, \tilde{\mathbf{h}}_i^{m_1}, \ldots, \tilde{\mathbf{h}}_i^{m_p}\}$. To aggregate these different embeddings for a node, HAN employs a semantic attention mechanism:

$$w_i^m = \frac{1}{|\mathcal{V}|} \sum_{i \in V} \mathbf{q}^\top \cdot \tanh(\mathbf{W} \cdot \tilde{\mathbf{h}}_i^m) \tag{2.17}$$

where $\mathbf{q}$ is the semantic metapath-level attention vector and $\mathbf{W}$ is an attention transformation matrix that projects the intermediate node embedding to the semantic attention space. Both the semantic attention vector and transformation matrix parameters are shared across all metapaths during the embedding process of HAN. The raw semantic attention scores w_i^m are normalized across the set of metapaths to produce an attention weight β_i^m. The metapath-specific representations of a node are then aggregated via a weighted sum with the semantic attention weights to produce a final embedding of node i that incorporates structural information from each metapath's neighbors and heterogeneous semantic information encoded across the set of metapaths:

$$\mathbf{h}_i = \sum_{k=1}^{p} \beta_i^m \cdot \tilde{\mathbf{h}}_i^m \tag{2.18}$$

Inspired by HAN, which proposed a two-stage hierarchical process for embedding nodes in heterogeneous graphs, a series of metapath-based methods were proposed [40–43]. HAHE [40] also employs a two-stage hierarchical structure for embedding heterogeneous graphs, but in HAHE, cosine similarity is used to in place of attention mechanisms to compute the relative node-level and semantic-level importances. Another method that employs a GAT-style attentional GNN similar to HAN is Metapath Aggregation Graph Neural Network (MAGNN) [41]. MAGNN likewise employs a two-stage hierarchical method for encoding metapath-specific structural information and heterogeneous semantic contexts. Distinct from HAN though, MAGNN incorporate all of the intermediate nodes along a particular metapath neighborhood of a target node when conducting the node-level embedding step.

In summary, metapath-based methods for embedding heterogeneous graphs define

a set of composite relations in the network that form the basic unit of the learning process. Unlike, type-based and relation-based methods, metapath-based methods for heterogeneous graph representation learning to not directly decompose heterogeneous graphs its components or component subgraphs. Instead, they compose a set of derivative graphs based on the composite relation paths defined by metapath schemata. These metapaths are capable of capturing higher-order structural and semantic information from heterogeneous networks by reaching neighboring nodes and graph structures that would otherwise not be included during the embedding process if a model was limited to the existing links present in the network. The major limitation of such methods is the design of the metapath schemata, as these models are highly dependent on the choice of metapaths and constructing effective metapaths often requires extensive domain knowledge and as such are highly application-specific.

2.3 Temporal Graph Neural Networks

Before the advent of Temporal Graph Neural Networks, research into dynamic network analysis relied on two main approaches: sequential modeling and traditional dynamic network models. Recurrent Neural Networks (RNNs) and their variants (LSTMs, GRUs) represent the gold standard for sequential modeling and have been adapted to dynamic graphs by processing a sequence of static graph snapshots. However, standard RNNs struggle to effectively fuse the topological information from the GNN layer with the temporal dependencies, frequently suffering from information loss and high computational costs when dealing with rapidly changing structures. Conversely, traditional dynamic network analysis methods often involve matrix factorization or statistical models (e.g., Exponential Random Graph Models, Hidden Markov Models) applied to graph sequences. While robust for capturing high-level temporal patterns, these techniques generally lack the capacity for deep, non-linear feature extraction and do not scale well to large, dense, or sparse networks common in modern applications. Temporal GNNs address the challenge of simultaneously cap-

turing both structural dependencies and temporal dynamics. These techniques can be broadly categorized based on how they handle the flow of time as either discrete steps or continuous time.

2.3.1 Discrete-Time Temporal GNNs

Discrete-Time Temporal GNNs operate by modeling a dynamic graph as a sequence of discrete, static snapshots $G^{(1)}, G^{(2)}, \ldots, G^{(T)}$. The general approach involves a two-step process: first, a static GNN (such as GCN or GAT) performs spatial aggregation within each snapshot $G^{(t)}$ to compute local embeddings, and second, a sequential model, often an RNN or a temporal convolutional network, is used to propagate the aggregated graph information from time t to $t+1$. Foundational approaches in this area, such as the Graph Convolutional Recurrent Network (GCRN) [44] and the Diffusion Convolutional Recurrent Neural Network (DCRNN) [45], explicitly couple GCNs with recurrent layers to capture spatio-temporal dependencies, proving particularly effective in tasks like traffic forecasting. Other well-known models in this vein include the Spatio-Temporal Graph Convolutional Network (STGCN) [46], which employs a temporal convolutional network for sequential modeling and has been shown to be effective for large-scale time series data. More advanced methods, including DyGNN [47] and EvolveGCN [48], introduce mechanisms to dynamically update the GNN parameters using a recurrent unit, which allows the model to adapt its graph convolution and aggregation strategy based on the ongoing temporal evolution. Recent efforts have also integrated full self-attention across the sequence of snapshots, as seen in models like DySAT [49], to capture more complex, long-range temporal dependencies.

2.3.2 Continuous-Time Temporal GNNs

Continuous-Time Temporal GNNs are specifically designed for asynchronous, event-based data where interactions occur at arbitrary, continuous time stamps t_i. This

modeling paradigm is essential for accurately capturing real-world dynamic processes, such as those found in social networks or transaction systems. Key architectures in this category primarily rely on temporal memory mechanisms to process the unordered sequence of events. Specifically, models focusing on temporal message passing, such as TGN [50] and JODIE [51], maintain a dedicated temporal memory for each node. This memory is dynamically updated upon every interaction and serves to summarize past activities, which is then used to compute time-aware embeddings during prediction or message aggregation. Beyond explicit memory, other continuous-time Temporal GNNs utilize sophisticated self-attention mechanisms to weigh the importance of past events. These attention layers, similar to those in transformer architectures, are modified to incorporate time encoding, allowing models like TGAT [52] to aggregate neighbors' information based on their relative time difference to the target node, providing a flexible and powerful way to model temporal relevance. Further refinements in this space have focused on enhancing temporal awareness, such as CTDNE (Continuous-Time Dynamic Network Embedding) [53], which utilizes self-attention to learn representations that better track node evolution in real-time.

CHAPTER 3: METHODOLOGY

In this chapter, we first introduce the concept of a multiplex, which is a multilayer representation of a complex network. A multiplex interpretation of heterogeneous graph representation learning provides a potential comprehensive construct that can incorporate both type-based, relation-based, and metapath-based methods under a single unifying language. We then introduce our multiplex-based GNN framework for heterogeneous graph representation learning, published in [54]. muxGNN leverages a multiplex representation of heterogeneous or temporal graphs to produce rich semantic embeddings for a broad variety of downstream graph learning tasks.

3.1 Preliminaries

Heterogeneous graphs and multiplex networks are important abstractions for modeling relational data in many real-world complex systems [55, 56]. Formally, they are defined as:

Definition 1. Heterogeneous Graph: A heterogeneous graph is defined as a graph $G = (\mathcal{V}, \mathcal{E}, T_v, T_e)$ where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges. Each node $v \in \mathcal{V}$ is associated with a type mapping function $\phi(v) : \mathcal{V} \to T_v$ and each edge $e \in \mathcal{E}$ is associated with a mapping $\psi(e) : \mathcal{E} \to T_e$, where T_v and T_e are the sets of node types and edge types, respectively.

Definition 2. Relation: For an edge $e = (u, v)$ in a heterogeneous graph, we define the relation r as $\langle \phi(u), \psi(e), \phi(v) \rangle$. Each edge is associated with a mapping $\text{rel}(e) : \mathcal{E} \to R$, where R is the set of all relations present in the graph. We can assume that there may exist multiple relations between different types of nodes in many real-world heterogeneous graphs.

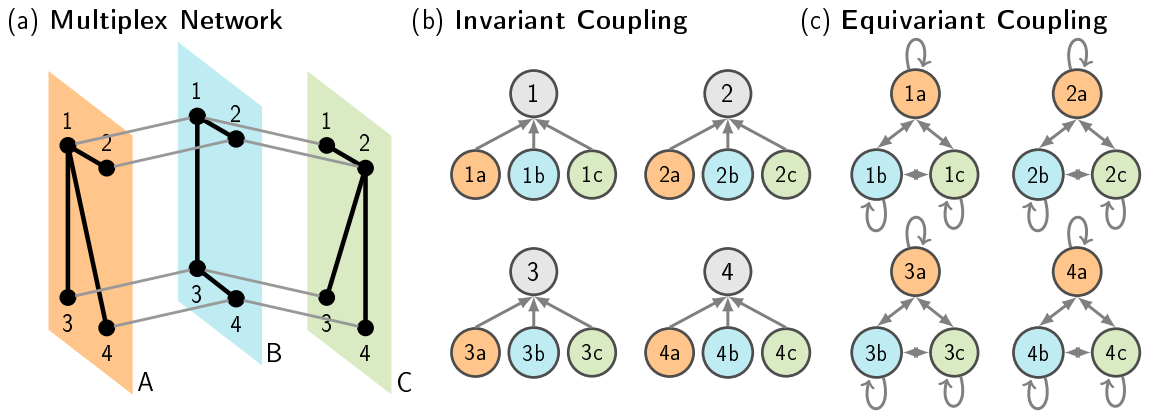


Figure 3.1: Schematic of a multilayer representation of a heterogeneous graph. (a) A multiplex network consisting of relation-specific layer graphs with separate node instantiations coupled across relations. (b) Node invariant coupling graph structure in which a node's relation-specific representations are directly coupled to the supra-node to produce a singular representation of a node in a heterogeneous graph. (c) Node equivariant coupling graph structure in which node-relation instantiations are coupled with one another to produce a set of representations characterizing a node in a heterogeneous graph.

Definition 3. Multiplex Network: A multiplex network is a multi-layer representation of a graph defined as a quadruple $\boldsymbol{M} = (\mathcal{V}, \mathcal{R}, B, \mathbb{M})$. $\mathcal{V}$ corresponds to the node set $\mathcal{V}$ in the heterogeneous graph and represents the set of *supra-nodes* that are separately instantiated in each layer in the multiplex system. The set of relations is represented by $\mathcal{R}$, which we call the relation layer set, as the relations in the heterogeneous graph will form the basis for each layer graph in the multiplex network. The separate instantiations of each supra-node $v \in \mathcal{V}$ across each relation $r \in \mathcal{R}$ is given by a binary relation on $\mathcal{V}$ and $\mathcal{R}$, such that $B = (\mathcal{V}, \mathcal{R}, \beta)$, $\beta \subseteq \mathcal{V} \times \mathcal{R}$. In other words, the statement $(v, r) \in \beta$ reads *node v participates in relation r.*

The set of layer-graphs $\mathbb{M} = \{G^r\}_{r \in R}$ defines the relation specific graphs on node-layer pairs. Each layer graph G^r's node set is subset of the node-layer pairs such that $\beta^r = \{(v, i) \subset \beta \,|\, i = r\}$, and its edge set is the subset of $\mathcal{E}$ such that $\mathcal{E}^r = \{e \in \mathcal{E} \,|\, \text{rel}(e) = r\}$, i.e. the relation triple of edge e corresponds to r.

Lastly, we may define the interlayer coupling graph G^C on β in which there is an

edge between two node-layer pairs (v, i) and (u, j) only if $v = u$. In other words, the graph G^C defines the coupling between the same node $v \in \mathcal{V}$ across different layers. These four components together define a multiplex system encoding both intra- and inter-relation interactions in a heterogeneous graph.

3.2 muxGNN Framework

In this section, we describe the muxGNN framework [54]. The idea in muxGNN is to decompose the multiplex representation of a heterogeneous graph into two components defined by the layer graphs and the coupling graph. This allows muxGNN to define a convolution over the entire multiplex by separately parameterizing weight matrices for the specific instantiations of nodes in each layer graph and parameterizing a dynamic coupling mechanism over the coupling graph.

3.2.1 Multiplex Spectral Convolution

Let $\boldsymbol{M} = (\mathcal{V}, \mathcal{R}, B, \mathbb{M})$ be a multiplex network with nodes $\mathcal{V} = \{1, \ldots, N\}$ and $\mathcal{R} = \{1, \ldots, R\}$ layers. Interactions between node-layer pairs are encoded by the set of layer graphs $\mathbb{M} = \{G^r\}$, and the coupling between separate instantiations of the same node across pairs of layers is encoded by the coupling graph G^C.

We can represent the system of layer graphs as the set of adjacency matrices $\{\boldsymbol{A}^r\}$ such that $A^r_{ij} = 1$ only if $(i, j) \in G^r$ and $A^r_{ij} = 0$ otherwise. If we restrict coupling to only between a node in one layer and that same node in other layers, the multiplex network is "diagonally" coupled, and we can define an $N \times N$ coupling adjacency matrix $\boldsymbol{C}$ such that $C_{ij} = 1$ only if $((i, n), (j, m)) \in G^C$, i.e. i and j correspond to the same node in different layers [55, 57]. We can then construct the *supra-adjacency* block diagonal matrix of the multiplex graph as:

$$\mathcal{A} = \bigoplus_{r \in R} \boldsymbol{A}^r + \boldsymbol{C} = \begin{pmatrix} \boldsymbol{A}^1 & \boldsymbol{C} & \dots & \boldsymbol{C} \\ \boldsymbol{C} & \boldsymbol{A}^2 & \dots & \boldsymbol{C} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{C} & \boldsymbol{C} & \dots & \boldsymbol{A}^R \end{pmatrix} \tag{3.1}$$

We may also consider the multiplex *supra-Laplacian* derived from the Laplacians of the layer graphs as $\boldsymbol{L}^r = \boldsymbol{D}^r - \boldsymbol{A}^r$, where $\boldsymbol{D}^r = \text{diag}(d_1^r, \dots, d_N^r)$ is the diagonal degree matrix for layer graph G^r, and $\boldsymbol{L}^C = \boldsymbol{D}^C - \boldsymbol{C}$, where $\boldsymbol{D}^C = \text{diag}(d_1^c, \dots, d_N^c)$ is the multiplexity degree matrix containing the number of layer in which a node participates along the diagonal. Furthermore, by considering quotient graphs over the multiplex network, we may decompose the supra-Laplacian of the whole multiplex into two components defined by the supra-Laplacian of the layer graphs and the supra-Laplacian of the coupling graph [57, 58].

$$\boldsymbol{\mathcal{L}} = \boldsymbol{\mathcal{L}}^R + \boldsymbol{\mathcal{L}}^C \tag{3.2}$$

where $\boldsymbol{\mathcal{L}}^R$ is the supra-Laplacian of the independent layer graphs and $\boldsymbol{\mathcal{L}}^C$ is the supra-Laplacian of the interlayer coupling graph. The layer supra-Laplacian may be expressed as:

$$\boldsymbol{\mathcal{L}}^R = \bigoplus_{r \in R} \boldsymbol{L}^r = \begin{pmatrix} \boldsymbol{L}^1 & 0 & \dots & 0 \\ 0 & \boldsymbol{L}^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \boldsymbol{L}^R \end{pmatrix} \tag{3.3}$$

where $\boldsymbol{L}^r$ represents the Laplacian matrix of graph G^r. As a block diagonal matrix with independent entries, the spectrum of the layer supra-Laplacian $\boldsymbol{\mathcal{L}}^R$ is given by the union of the spectra of the individual layer graph Laplacians:

$$\begin{aligned} \Lambda^R &= \bigcup_r \Lambda^r \\ s.t. \quad \lambda^R_{max} &= \max_r \lambda_r \end{aligned} \tag{3.4}$$

The interlayer coupling supra-Laplacian may be expressed as the Kronecker product of the coupling graph Laplacian, $\boldsymbol{L}^C$ and the $N \times N$ identity matrix I:

$$\boldsymbol{\mathcal{L}}^C = \boldsymbol{L}^C \otimes I \tag{3.5}$$

The spectrum of the Kronecker product of two matrices is formed by the product of the eigenvalues of the individual matrices. Since the identity matrix only has eigenvalues equal to one, the spectrum of the coupling supra-Laplacian is spectrum of $\boldsymbol{L}^C$ with a multiplicity N.

If we consider the eigendecomposition of the multiplex supra-Laplacian, $\boldsymbol{\mathcal{L}} = \boldsymbol{U}\Lambda\boldsymbol{U}^T$ where $\boldsymbol{U}$ is the eigenvector matrix and Λ is the diagonal eigenvalue matrix of $\boldsymbol{\mathcal{L}}$, it has been shown that the eigenvalues of the layer supra-Laplacian and coupling supra-Laplacian are also eigenvalues of the parent multiplex supra-Laplacian [57]. In particular, if we let $\boldsymbol{u}$ be any of the eigenvectors of $\boldsymbol{\mathcal{L}}^C$ with corresponding eigenvalue λ, then $\boldsymbol{u}$ is also an eigenvector of $\boldsymbol{\mathcal{L}}^R$ with zero eigenvalue, and thus,

$$\boldsymbol{\mathcal{L}}\boldsymbol{u} = \left(\boldsymbol{\mathcal{L}}^R + \boldsymbol{\mathcal{L}}^C\right)\boldsymbol{u} = \lambda\boldsymbol{u} \tag{3.6}$$

We can alternatively characterize the spectrum of the multiplex supra-Laplacian by considering the coupling supra-Laplacian as a perturbation matrix acting on the layer supra-Laplacian, such that:

$$\bar{\lambda}^R = \lambda^R + \Delta\lambda \tag{3.7}$$

where λ^R is an eigenvalue of the layer graph supra-Laplacian, and $\Delta\lambda$ is the perturbation due to the interlayer coupling [55].

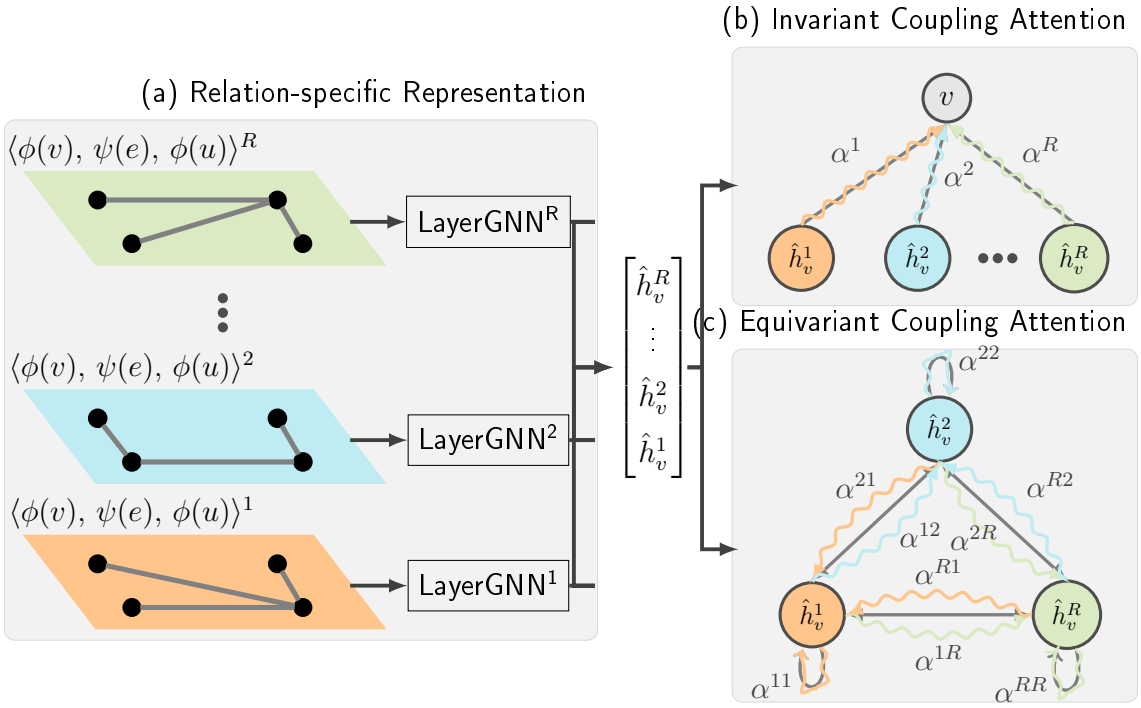


Figure 3.2: Overall framework of the muxGNN model. (a) The relation-specific representation of a node generated by local neighborhood aggregation across each relation in which a node participates. (b) The node-relation pair representations of a supra-node are fused through coupling weights learned via node invariant coupling attention (c) Node equivariant attention learns sets of coupling weights for a node from the perspective of each participating relation as the dominant layer with the associated relative perturbations of the other relations.

We may therefore exploit the shared eigenbasis and the relationship between the eigenvalues of the multiplex with the spectra of the layer graphs and coupling graph to define a multiplex graph convolution as:

$$(g_\theta, h_\theta) \star x_i = \boldsymbol{U} \left(g_\theta(\lambda^R) + h_\theta(\lambda^C) \right) \boldsymbol{U}^T x_i \tag{3.8}$$

where g_θ and h_θ are functions on the eigenvalues of the independent layer supra-Laplacian $\boldsymbol{\mathcal{L}}^R$ and the coupling supra-Laplacian $\boldsymbol{\mathcal{L}}^C$, respectively. Therefore, a convolution over a multiplex representation of a heterogeneous network can be parameterized using graph neural networks applied hierarchically over the relation graphs and the coupling graph.

3.2.2 Layer Graph Representation

We have shown that a convolution over a multiplex system may be decomposed as two operators over the relation-specific layer graphs and over the coupling graph. Furthermore, Equation 3.4 shows that the spectrum of the layer graph supra-Laplacian is itself composed of the spectra of the independent layer graphs, so we may separately parameterize convolution over each independent layer graph. Directly learning the eigenvalues of the Laplacian is generally inappropriate due to the fact that they are expensive to compute, non-localized, and for the most part do not generalize well. Instead, a common solution has been to "spatialize" the spectral operation by making the eigenvalues related to the eigenvalues of the Laplacian through a localized spatial function with coefficients defined by $c_{vu} = (f(\boldsymbol{L}))_{vu}$ [1,6]. We are therefore able to approximate the convolution, $g_\theta(\Lambda^R)$, over the layer graph supra-Laplacian by applying a graph neural network over each relation-specific layer graph in the multiplex.

We consider three implementations from two prominent families of graph neural networks—convolutional and attentional—as the layer-wise operator in muxGNN. The neighbor coefficients may be fixed weights as in convolutional GNNs or may be computed implicitly, as in attentional GNNs. Specifically, we evaluate GCN [6] and GIN [10] as examples of convolutional GNNs and GAT [9] as an example attentional GNN.

For each layer graph $G^r \in \mathbb{M}$ in the multiplex, we compute the layer-specific representations of a node-layer pair $(v, r) \in G^r$ using a layer-wise GCN operator as:

$$\hat{\boldsymbol{h}}_{v,r}^{(k)} = \sigma\left(\mathbf{W}^r \sum_{u \in \tilde{\mathcal{N}}(v,r)} \frac{1}{\sqrt{d_v^r d_u^r}} \boldsymbol{h}_{u,r}^{(k-1)}\right) \tag{3.9}$$

where $\tilde{\mathcal{N}}(v, r)$ is the neighborhood of node v in layer r with added self-loop, $\mathbf{W}^r$ is a relation-specific trainable weight matrix, σ is a non-linear activation function, and d_v^r and d_u^r are the degree of nodes v and u in layer r. This function corresponds to a

first-order approximation of a spectral convolution over the symmetrically normalized relation layer graph Laplacian.

We also consider the layer-wise GIN operator is defined as:

$$\hat{\boldsymbol{h}}_{v,r}^{(k)} = \text{MLP}^r \left((1 + \epsilon^r) \cdot \boldsymbol{h}_{v,r}^{(k-1)} + \sum_{u \in \mathcal{N}(v,r)} \boldsymbol{h}_{u,r}^{(k-1)} \right) \tag{3.10}$$

Compared with the GCN operator, GIN notably replaces the single layer transformation with a multi-layer perceptron (MLP) and with a simple unnormalized summation function applied directly over the adjacency structure of the relation layer graph. The sum aggregator's ability to represent the entirety of a multiset is particularly important for multiplex networks.

Thirdly, we also consider a GAT layer operator, defined as:

$$\hat{\boldsymbol{h}}_{v,r}^{(k)} = \sigma \left(\mathbf{W}^r \sum_{u \in \tilde{\mathcal{N}}(v,r)} a_{vu}^r \boldsymbol{h}_{u,r}^{(k-1)} \right) \tag{3.11}$$

where aggregation coefficient a_{vu}^r is computed dynamically in a node-dependent manner via edge softmax attention [9]. In this case, the GAT function is a first-order approximation of a convolution applied over the weighted adjacency matrix, where the edge weights are computed via the attention mechanism.

3.2.3 Coupling Graph Attention

The second component of muxGNN, and the essential component for encoding the interlayer coupling dynamics that distinguish a multiplex system from a set of independent graphs, is the coupling graph attention mechanism. From the construction of the multiplex supra-Laplacian defined in Section 3.2.1, we may further define a weighted supra-Laplacian with weight parameter α as:

$$\bar{\boldsymbol{\mathcal{L}}} = \bigoplus_{r \in R} \boldsymbol{L}^r + \alpha(\boldsymbol{L}^C \otimes I) \tag{3.12}$$

We can then approximate a convolution over the weighted coupling supra-Laplacian by parameterizing an attentional graph neural network over the coupling graph. By limiting coupling to within the instantiations of the same supra-node, the edge structure of the coupling graph is formed separately by all the node-layer pairs representing the same supra-node. Each supra-node $v \in \mathcal{V}$ is represented in the coupling graph by the set of feature vectors of its separate node-layer instantiations computed by the layer graph GNNs:

$$\hat{\boldsymbol{h}}_v^{(k)} = \{\hat{\boldsymbol{h}}_{v,r}^{(k)} \,|\, r \in R\} \tag{3.13}$$

We can define two edge arrangements that arise naturally from the construction of multiplex system from the heterogeneous graph, illustrated in Figure 3.1. In the *node invariant* case, the node-layer pairs are all directly connected to a synthetic supra-node. Such a coupling structure is appropriate for node-level and graph-level tasks. To compute the node invariant coupling of a node based on its node-relation instantiations, we define an attentional GNN with coefficients computed via an attention mechanism that can capture various types of semantic information to learn the importance of each relation for a give node [18, 30, 59]. The representation of a supra-node v is given as the weighted sum of the learned layer representations for each of its associated node-relation instantiations:

$$\boldsymbol{h}_v^{(k)} = \sum_{r \in R} \alpha_v^r \cdot \hat{\boldsymbol{h}}_{v,r}^{(k)} \tag{3.14}$$

The coupling attention measures how important a give node-relation instantiation is for the overall supra-node. First, we project the relation-specific embedding of a node through a non-linear transformation, and then we compute the importance as the similarity of the transformed representation with a relation-level attention vector $\mathbf{q}_r$ as:

$$a_v^r = \mathbf{q}_r^{\mathrm{T}} \cdot \tanh(\mathbf{W}^r \cdot \hat{\boldsymbol{h}}_{v,r}) \tag{3.15}$$

The coupling weight of relation r for node v can be obtained by normalizing the importances of each relation in which a supra-node participates using the softmax function:

$$\alpha_v^r = \frac{\exp(a_v^r)}{\sum_{r \in R} \exp(a_v^r)} \tag{3.16}$$

In the node invariant coupling case, the attention mechanism applied over the composite supra-node learns the "natural" dominant layer and relative perturbations due to participation in other relations for each node in the multiplex. This supra-node level embedding computed by muxGNN with node invariant coupling is well-suited for node-level and graph-level tasks on heterogeneous networks.

We can further extend the coupling attention to operate in a *node equivariant* manner that is more suitable for link-level tasks on heterogeneous graphs. By employing multiple attention heads, each associated with a particular relation, we can enforce that the coupling attention learns multiple coupling relationships for a particular supra-node from the perspective of each relation as the "dominant" layer. Rather than a single semantic attention vector, each coupling attention head is associated with a set of relation attention vectors. The attentional GNN is applied to each node-relation instantiation in the coupling clique associated with a supra-node:

$$\left(\boldsymbol{h}_v^1, \ldots, \boldsymbol{h}_v^R\right)^{(k)} = \Big\Vert_{r \in R} \operatorname{attn}^r \left(\hat{\boldsymbol{h}}_v^{(k)}\right) \tag{3.17}$$

In this way, muxGNN with node equivariant coupling produces a set of embeddings associated with a particular supra-node representing multiple views of the node from the perspective of each relation in which the node participates.

The hierarchical application of relation-layer neighborhood aggregation with cross

relation coupling attention allows muxGNN to learn node representations that incorporate soft meta-paths across multiple relation paths by stacking muxGNN layers over the k-hop neighborhoods of node v. The final representation, or set of representations in the equivariant case, for a node v after K muxGNN layers is:

$$\boldsymbol{z}_v = \boldsymbol{h}_v^{(K)} \tag{3.18}$$

For graph-level tasks, we apply a permutation invariant readout function the set of node embeddings for a given graph. In our experiments we use a *sum* readout as:

$$\boldsymbol{z}_{\boldsymbol{M}} = \sum_{v \in \mathcal{V}} \boldsymbol{z}_v \tag{3.19}$$

3.2.4 Model Optimization

In this section, we describe the semi-supervised training process for the muxGNN framework. We use random walks over the multiplex graph to generate node sequences and optimize the model to learn node representations that maximize the similarity of co-occurring node pairs [16,30]. Specifically, we conduct random walks along each layer graph G^r, such that the transition probability at each step is:

$$p(u \,|\, v, G^r) = \begin{cases} \frac{1}{|\mathcal{N}(v,r)|} & (v,u) \in G^r \\ 0 & (v,u) \notin G^r \end{cases} \tag{3.20}$$

where $\mathcal{N}(v,r)$ denotes the neighborhood of node v in layer r. The random walker conducts a set of walks for each node along every layer in which the node participates, capturing the heterogeneous contexts of a node in the multiplex network. A random walk of length l in G^r defines a path $W^r = (v_1, v_2, ..., v_l)$. Given a context sliding window size c, we define the layer context of v in G^r as:

$$C = \{v_j \,|\, v_i \in W^r,\, |i-j| \leq c,\, j \neq i\} \tag{3.21}$$

The objective for a node v with a context C along a path is to minimize the negative log-likelihood:

$$-\log P_\theta\left(\{u \,|\, u \in C\} \,|\, v\right) = \sum_{u \in W} \log P_\theta(u \,|\, v) \tag{3.22}$$

where θ represents the model parameters. We utilize the softmax function as the probability of a node u given v:

$$P_\theta(u \,|\, v) = \frac{\exp(\boldsymbol{c}_u^T \cdot \boldsymbol{z}_v^r)}{\sum_{w \in G^r} \exp(\boldsymbol{c}_w^T \cdot \boldsymbol{z}_v^r)} \tag{3.23}$$

where $u, v \in G^r$, $\boldsymbol{c}_u$ is the context embedding of node u, and $\boldsymbol{z}_v^r$ is the embedding for node v on layer r. To approximate the objective function, we use negative sampling for each context triplet (v, u, r) as:

$$E = -\log \sigma(\boldsymbol{c}_u^T \cdot \boldsymbol{z}_v^r) - \sum_{n=1}^{N} \mathbb{E}_{w \sim P_{(G^r)}}[\log \sigma(-\boldsymbol{c}_w^T \cdot \boldsymbol{z}_v^r)] \tag{3.24}$$

where σ is the sigmoid function. N corresponds to the number of negative samples drawn for each positive sample, and w is a node drawn randomly from noise distribution $P_{(G^r)}$ defined on the corresponding node set of layer graph G^r. $P_{(G^r)}$ may be either a uniform distribution or a log-uniform distribution ordered by node degree. As many real-world networks follow a power law degree distribution, a log-uniform distribution weights the probability of a node being sampled proportional to the node's degree.

3.3 Bipartite Node Labeling

In this section, we describe the bipartite node labeling algorithm developed in [60] to heuristically label a graph with a minimal set of initial labelled nodes. A bipartite

graph $\mathcal{G} = (V, E)$ consists of two disjoint node sets with an edge $(u, v) \in E$ indicates that a node u from one node set is connected to a node v from the other node set. The weight $w(u, v)$ represents the weight of the edge between node u and node v. Using this graph, we develop a reciprocal label propagation-based method for automatically assigning labels to the nodes in the bipartite network.

Based on a small (3-5) set of seed nodes associated with each label, we first propagate the label associated with each seed node across the bipartite graph to the set of nodes connected with the seed nodes. The importance of a node's label to a particular destination node is weighted by the corresponding edge weight, as described in Algorithm 2. Nodes from the opposite node set are then assigned a label based on the weight for each label group. Node labels are then propagated back to the opposite node set across the edges in the graph, as described in Algorithm 3. Each node is then scored based on the difference between the normalized weight from nodes of each label group. Node scores are normalized via min-max normalization, and the mean and standard deviation of the scores is computed. Nodes that score one or more standard deviations above or below the mean are assigned the corresponding label. We then repeat alternating label propagation until the model converges, i.e. the set of labeled nodes is the same from the prior iteration to the next.

While the bipartite label propagation method can quickly label a number of nodes with high precision and is well suited for determining nodes that are highly associated with one particular stance group as compared with the other, nodes that are connected across both label groups albeit in different contexts will remain unlabeled. Employing the set of labeled nodes from this heuristic as a supervised training set for training a classification GNN over the entire graph allows for effective labeling of all nodes in the graph.

Algorithm 1 Bipartite stance label propagation algorithm

Require: Bipartite graph G, seed node set V^0, s stance label

$V^0 \leftarrow \{(v, s) \in$ Set of labeled seed nodes$\}$
$U^0 \leftarrow \emptyset$
while $U^{(k-1)} \neq U^k$ **do**
    $U^k \leftarrow propagateVToU\left(G,\, V^{(k-1)}\right)$
    $V^k \leftarrow propagateUToV\left(G,\, U^k\right)$
end while
return U^K, V^K

Algorithm 2 Propagate node set V labels to node set U

Require: Bipartite graph G, labeled node set V, s stance label, $\mathcal{N}$ neighbor set

u_counts $= \{\}$
for $V, s \in V$ **do**
    for $u, w \in \mathcal{N}(v)$ **do**
        u_counts$[u][s]$ += w
    end for
end for
for $u \in$ u_counts **do**
    $U \leftarrow (u,\, \max_s$ (u_counts[u]))
end for
return U

Algorithm 3 Propagate node set U labels to node set V

Require: Bipartite graph G, labeled node set U, s stance label, $\mathcal{N}$ neighbor set

v_counts $= \{\}$
for $u, s \in U$ **do**
    for $V \in \mathcal{N}(u)$ **do**
        v_counts$[v][s]$ += 1
    end for
end for
for $V \in$ v_counts **do**
    $v_{score} = \frac{\text{v\_counts}[v][s_2]}{|U_{s2}|} - \frac{\text{v\_counts}[v][s_1]}{|U_{s1}|}$
end for
$\mu = \text{mean}(v_{score} \,|\, v \in \text{v\_counts})$
$\sigma = \text{stdev}(v_{score} \,|\, v \in \text{v\_counts})$
for $V \in$ v_counts **do**
    if $v_{score} \geq \mu + \sigma$ **then**
        $V \leftarrow (v, s_1)$
    else if $v_{score} \leq \mu - \sigma$ **then**
        $V \leftarrow (v, s_2)$
    end if
end for
return V

CHAPTER 4: SOCIAL NETWORK APPLICATIONS

Social networks can be effectively represented as heterogeneous graphs, where nodes and edges are of multiple types, capturing the diversity of entities and their interactions. Unlike homogeneous graphs that treat all nodes and edges uniformly, heterogeneous graphs model complex relationships by distinguishing between different node types (e.g., users, posts, tags) and edge types (e.g., follows, likes, comments). This richer structure allows for a more accurate and nuanced representation of real-world social networks, enabling tasks such as community detection, recommendation, and influence analysis to leverage the semantics of diverse interactions. By incorporating this heterogeneity, graph-based algorithms can better exploit the contextual and relational information inherent in social media platforms. In the following sections, we describe our work applying text-based and graph-based techniques to the problems of hate speech detection, link prediction, and stance labeling in the domain of social network analysis.

4.1 Hate Speech Detection

We develop DeL-haTE [61], a novel deep learning tunable ensemble framework designed for hate speech detection on social media, addressing key challenges like the lack of clearly labeled data, evolving vocabulary, and the absence of baseline models for fringe outlets like Gab. The framework's core contributions are threefold: first, it utilizes an ensemble of deep learning models combining the strengths of state-of-the-art approaches, specifically leveraging CNN and GRU layers to extract higher-order and sequential features from word embeddings. Second, it incorporates a tuning factor that uses transfer learning to classify hate speech on unlabeled datasets like Gab by

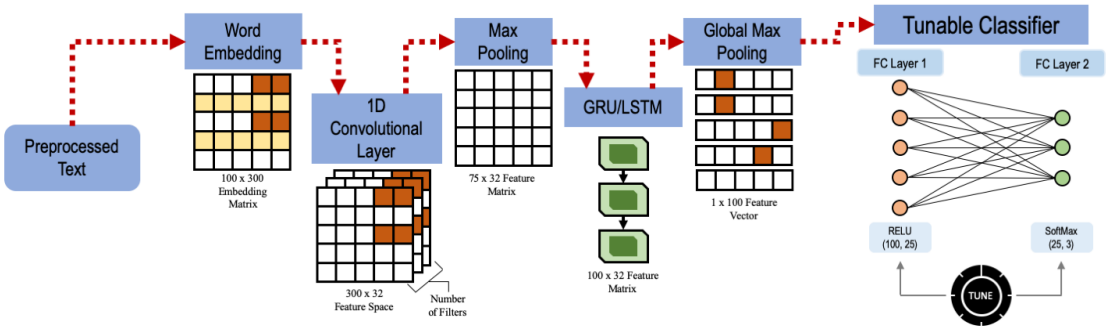


Figure 4.1: The model topology depicted illustrates the CNN/GRU feature component and the tunable classifier. These models are combined into an ensemble model for predicting the presence of hate speech in online posts.

Table 4.1: Distribution of Hateful (H), Offensive (O), and Neither (N) data samples in multiple datasets. Hateful posts are only 5%-11% of the total corpus for machine learning training

Dataset	**Class (in %)**			**# Samples**
	H	O	N	
HON	5.74	77.41	16.85	22,305
OLID	8.24	25.09	66.67	11,916
Combined	6.61	59.19	34.20	34,221
Gab - Test	11.19	22.05	66.76	1,465
Gab - Train	–	–	–	90,899

tuning the classifier component of the model on a small, manually labeled sample of the target data. Finally, it develops a weak supervised learning methodology that allows the framework to be trained entirely on abundant unlabeled data, relying on lexicons to set per-class bounds for a weakly supervised cross-entropy loss function.

Two of the major challenges in identifying hate speech online are the severe class imbalance and the ever-shifting lexicon of posters online. This framework addresses the challenge of evolving, unlabeled data through two primary mechanisms: transfer learning with classifier tuning and weak supervised training. For transfer learning, the model leverages existing knowledge by training classifier ensembles on labeled source datasets (HON [62] and OLID [63]) and then tunes these pre-trained models using a small, balanced sample of manually labeled posts from the target dataset (Gab).

This tuning procedure adapts the model to the target data's specific characteristics by freezing the feature extraction layers (CNN-RNN) and only training the final classifier layers (FC layers) on the new data. Alternatively, the weak supervision methodology allows the model to train and tune entirely on large volumes of unlabeled data. This is achieved by using heuristics based on hate, offensive, and positive word lexicons to programmatically generate per-class bounds and a corresponding weak supervised loss function, effectively extending the model's applicability to new, unannotated data like Gab posts.

4.1.1 Weak Supervised Training

Utilizing weak supervision during model training is a possible alternative to reliance on manually annotated hate speech datasets. The major benefit of weak supervision is the greatly expanded pool of training observations by leveraging abundant unlabeled social media data. For our classifier models, the framework utilizes a weakly supervised form of cross-entropy as the loss function f. This loss function is the multi-class extension of the weak supervised loss function utilized by [64] for binary classification of cyberbullying. Our weak supervised loss relies on lexicons of words indicative and counter-indicative of hate speech and offensive language. For a post containing n unique words, let n_h, n_o, and n_p denote the number of hate words, offensive words, and positive words, respectively. For each post, we calculate the bounds of each class using the algorithm described in Figure 4.2.

If this bound is violated, the loss function penalizes the model using the weak supervised loss function:

$$f(y_m) = \sum_{c \in C} (-log(min\{1, 1 + y_{m,c} - lb_{m,c}\}) \\ -log\{min(1, 1 + ub_{m,c} - y_{m,c}\})) * w_c$$

With the lack of *a priori* class labels, this form of weak supervised training is

```
Algorithm – 1: Bounds
Require: Post "p"
num_unique ← number of unique words in p
num_hate ← number of hate words in p
num_offensive ← number of offensive words in p
num_positive ← number of positive words in p

if (hate word in p):
    hate_LB ← (num_hate + num_offensive) / num_unique
    hate_UB ← 1
    offensive_LB ← 0.5 * num_offensive / num_unique
    offensive_UB ←1 - (num_hate + num_positive) / num_unique
    neither_LB ← 0
else if (offensive word in p):
     hate_LB ← 0
     hate_UB ← 1 - (num_offensive + num_positive) / num_unique
     offensive_LB ← (num_offensive + 0.5 * num_positive) / num_unique
     offensive_UB ← 1 - 0.5 * num_positive / num_unique
     neither_LB ← 0.5 * num_positive / num_unique
else:
    neither_LB ← num_positive / num_unique
neither_UB ← 1 - (num_hate + num_offensive) / num_unique

return  ((hate_LB,hate_UB), (offensive_LB,offensive_UB), (neither_LB,neither_UB))
```

Figure 4.2: Weak supervised bounds algorithm that calculates a lower and upper bound for each class.

susceptible to the class imbalance problem due to the small proportion of hate speech in the overall content on social media. To mitigate this issue, a per-class weight w_c is applied to the weak supervised loss contribution for each class. One benefit of our weak supervised heuristic is the tunable per-class weight applied during the loss calculation which can be adjusted for varying class balances in datasets.

While the techniques of transfer learning help to address the challenge of developing hate speech detection models for social networks where rapidly evolving language and extreme class imbalance are present, the reliance on textual content alone limits the capability of the model to explicit hate speech terminology, where euphemistic speech or in-group lexicons remain difficult to detect in online content. To address these extant challenges, it is critical to incorporate the graph structure and user dynamics in social networks to augment the text of online posts when developing models for analyzing content on social networks. This fact motivates the work on muxGNN to develop a graph neural network framework that could adequately represent the complexities of heterogeneous user interactions in social networks.

Table 4.2: Summary of social network datasets used in our experiments.

Name	# Node	# Edge	# Layer	Type
Twitter	28,473	91,726	3	Social Network
YouTube	2,000	1,310,544	5	Social Network

4.2 Link Prediction

We conduct link prediction evaluation using muxGNN with equivariant coupling on two different social network datasets: Twitter[1] and YouTube[2]. For the link prediction results, we employ area under the curve (AUC) of the receiver operating characteristic (ROC) and precision-recall (PR) curves and F1 score as evaluation metrics to measure the performance of different models. The code and datasets used in this work are available at `https://github.com/muxGNN/muxGNN`. The details of the social network datasets used in our experiments are listed in Table 4.2.

We compare our method with five state-of-the-art heterogeneous GNN frameworks, including MNE [29], R-GCN [15], and R-graphSAGE as examples of convolutional graph neural network models, and GATNE [30] and HAN [18] as examples of attentional GNN models. In addition, we select two random walk-based graph embedding methods, node2vec [65] and DeepWalk [66], for comparison. To adapt these methods to heterogeneous networks, we conduct the random walks independently over each layer graph. To explore the performance of different GNN models as the basis for the layer graph representation, we present results for muxGNN with GCN [6], GAT [9], and GIN [10] as the message passing function in the layer graph component.

4.2.1 Experimental Setup

To evaluate the performance of our method on link prediction, we conduct one set of experiments in a transductive setting—where all nodes in the test graph are also present in the training graph—and one set of experiments in an inductive context—

[1] https://snap.stanford.edu/data/higgs-twitter.html
[2] http://socialcomputing.asu.edu/datasets/YouTube

Table 4.3: Mean performance (ROC-AUC %, F1 % and PR-AUC %) on link prediction in a transductive context. OOT: Out of Time (24hrs).

	Twitter			YouTube		
	ROC-AUC	F1	PR-AUC	ROC-AUC	F1	PR-AUC
node2vec	72.58	71.94	82.23	71.21	65.36	70.32
DeepWalk	76.88	72.42	83.10	71.11	65.52	70.04
MNE	OOT	OOT	OOT	82.30	75.03	75.03
R-graphSAGE	74.31	70.77	77.77	81.02	73.91	77.58
R-GCN	92.76	85.85	92.88	80.21	73.36	75.52
GATNE-I	92.94	86.20	93.20	84.47	76.83	76.83
HAN	92.76	85.70	92.47	80.43	73.43	76.84
MuxGNN$_{GCN}$	88.95	81.61	88.97	82.93	75.56	79.82
MuxGNN$_{GAT}$	90.23	83.37	90.40	83.11	75.50	80.01
MuxGNN$_{GIN}$	**93.74**	**86.85**	**94.18**	**85.14**	**77.36**	**82.40**

where some nodes in the test graph are not present in the training graph. We utilize the train/val/test splits for the Amazon and YouTube datasets provided by [30] with 85% training links and 5% and 10% positive validation and positive testing links, respectively. The Twitter dataset is similarly constructed. For the Tissue Protein-Protein Interaction dataset, we use five-fold cross validation with 80% training links and 5% and 15% positive validation and test links, respectively. For all validation and test sets, an equivalent number of non-existent links are randomly selected from the graph as negative samples to augment the positive links.

The embedding size of all models is set to 200 dimensions with all GNN models using two graph convolution layers corresponding to the 2-hop neighborhood for all nodes. For all random walks, we use 20 walks of length 10 steps with a sliding window context of size 5. The number of edge attention heads in HAN and the GAT variant of muxGNN is set to two, and the heterogeneous attention of GATNE, HAN, and the coupling graph attention size for all muxGNN variants is set to 16 dimensions. All models are trained for a maximum of 50 epochs with early stopping based on the validation ROC-AUC.

4.2.2 Results & Discussion

The link prediction results in a transductive setting are presented in Table 4.3. It can be seen that the random walk based methods cannot achieve satisfactory performance on all datasets. Their performance notably degrades as the number of different node and edge types increases in the heterogeneous graph. In general, the attentional GNN models GATNE, HAN, and HGT outperform the convolutional graph neural networks MNE, R-graphSAGE, and R-GCN. Frameworks like GATNE, HAN, and our muxGNN utilize a two-stage encoding process whereby GNN modules learn latent representations of graph structures based on the network structure that are then pipelined to an attention mechanism to aggregate and combine. By contrast, HGT integrates a transformer-style query-key-value attention mechanism into their message propagation resulting in a framework where the message propagation and attention are intertwined. Our experiments indicate that this transformer-style module does not provide any significant benefits over the GNN then attention two-stage approach. Compared with transformer-style attention, our semantic attention mechanism requires fewer parameters—requiring two transformation matrices compared with three–and can successfully operate with low dimensional attention representations as illustrated in Figure 5.1. By leveraging a multiplex representation of heterogeneous graphs, muxGNN can flexibly substitute in different GNN modules with minimal alteration to the overall framework, allowing the framework to easily adapt to particular data sources and problems where a certain flavor of GNN is best suited. Additionally, such flexibility enables muxGNN to readily adapt to new GNN architectures as they are developed, allowing novel frameworks to be extended to heterogeneous graphs.

The graph convolution function proposed by [10] uses the sum operation as the permutation invariant aggregation function because the sum captures the full multi-set of node neighbors. In a multiplex network, it is especially important to retain the

full multiset of neighbors in each layer graph. In most heterogeneous graph datasets, there is only a single supra-node feature representation that is repeated across each relation layer. Additionally, the construction of a multilayer heterogeneous graph representations increases the likelihood of repeated or highly similar local neighborhood topologies across the relation layer graphs. All layer-specific representations together encompass a supra-node's structural role in the heterogeneous network. Because a particular supra-node is likely to appear multiple times across the set of layer-specific representations, resulting in a high multiplicity for a given supra-node across neighborhood sets in each layer. Compared with a *sum* aggregator as the permutation invariant function in the layer graph neural network function, a *mean* aggregator only captures the distribution of elements and the *max* ignores multiplicities altogether and reduces a multiset to a simple set [10]. As such, the layer graph representations generated by GIN retain the full multiset of a given supra-node's representations in each layer graph. This allows the coupling graph attention layer to operate over the full multiset of layer-specific representations for a given supra-node across the entire multiplex graph. The experimental results show that our method can learn better features to represent a target link for prediction in heterogeneous networks from multiple domains. In addition, our method is more stable than other baseline methods even as the heterogeneity of the networks increases.

4.3 Stance Labeling

Stance labeling, or stance detection, is the task of determining a user's attitude toward a specific target or topic on social networks based on their online content. This is a challenging problem because stance is often expressed subtly, implicitly, or through nuanced language, making methods that rely solely on textual analysis insufficient. A promising and increasingly necessary technique to overcome these limitations is socially infused text mining, which strategically incorporates two main data sources for a more holistic understanding: the textual content of a user's posts (what they

say) and the social context derived from the network. Specifically, it leverages network structure (like connection patterns and follower relationships) and community dynamics (the prevailing opinions within a user's social circles) to infer stance. By integrating a user's linguistic expression with the influence and relationships present in their user interaction graph, socially infused text mining leverages the principle of homophily to provide a more robust and accurate stance label.

4.3.1 Data Collection

We collect tweets related to two prominent and contentious political issues—climate change and gun control. The keywords listed in Table 4.4 are used to scrape the initial set of relevant tweets related to each topic. To fully capture conversation cascades and user interactions, we then recursively collect all referenced tweets from each tweet in the initially matched set. This ensures that all available tweets from relevant conversations are included in our analysis, even if particular tweets do not explicitly use any of the keywords. The climate change dataset consists of tweets published between June 1st, 2021 and May 31st, 2022. In all, the climate change dataset consists of a total of 46M tweets authored by 4.8M unique users and contains 726,378 conversation threads of at least three tweets. The gun control dataset consists of tweets published between January 1st, 2022 and December 31st, 2022. In all, the gun control dataset consists of a total of 14.4M tweets from 2.66M unique users containing 335,000 conversation threads of at least three tweets.

These issues are good models for understanding the differences in behavior on social media for topics that are highly event driven, like gun control, and those that are less focused on particular events, like climate change. Figure 4.3 illustrates that conversation threads related to climate change are started at a fairly consistent rate over time with some spikes in conversation activity connected with specific events.

Table 4.4: Set of keywords used to collect tweets related to Climate Change and Gun Control from Twitter.

Climate Change		Gun Control	
climate change	global warming	gun control	gun rights
climate hoax	global warming hoax	second amendment	2nd amendment
global cooling	#ActOnClimate	#guncontrol	#guncontrolnow
#ClimateChange	#climatechangehoax	#gunreform	#gunviolence
#globalwarminghoax	#globalcooling	#endgunviolence	#2a
#globalwarmingisahoax	#climatehoax	#nra	#gunrights
		#secondamendement	#shallnotbeinfringed
		#righttobeararms	

But compared with the conversation activity on Twitter surrounding gun control, illustrated in Figure 4.4, the frequency of discussions surrounding gun control spikes sharply in response to mass shooting events. The summer of 2022 saw several deadly and highly publicized mass shootings in Buffalo, Uvalde, and Chicago that generated large amounts of discourse surrounding gun control and gun violence on social media, illustrated by the sharp spikes from the end of May through late July. Shortly after mass shooting events though, the level of discourse abates, returning to near baseline levels between early August until November, when a shooting event at UVA caused a small spike in conversation activity. Event driven issues such as gun control exhibit different posting behavior and user interactions that we hypothesize lead to differences in polarization and require different techniques for intervention and mitigation of polarization on social media. Developing approaches for stance labeling that are both topic independent and robust to different types of discourse online is an important step for facilitating further analysis of polarization online.

4.3.2 Heuristic Labeling

As described in Section 3.3, we apply our bipartite labeling heuristic to the climate change and gun control tweet datasets. From the set of collected tweets for a given topic, we first construct a bipartite graph connecting users with the hashtags that each user posted. The user-hashtag bipartite graph $\mathcal{G} = (V, E)$ consists of two disjoint node sets containing the users and the hashtags posted in tweets by the user set. An

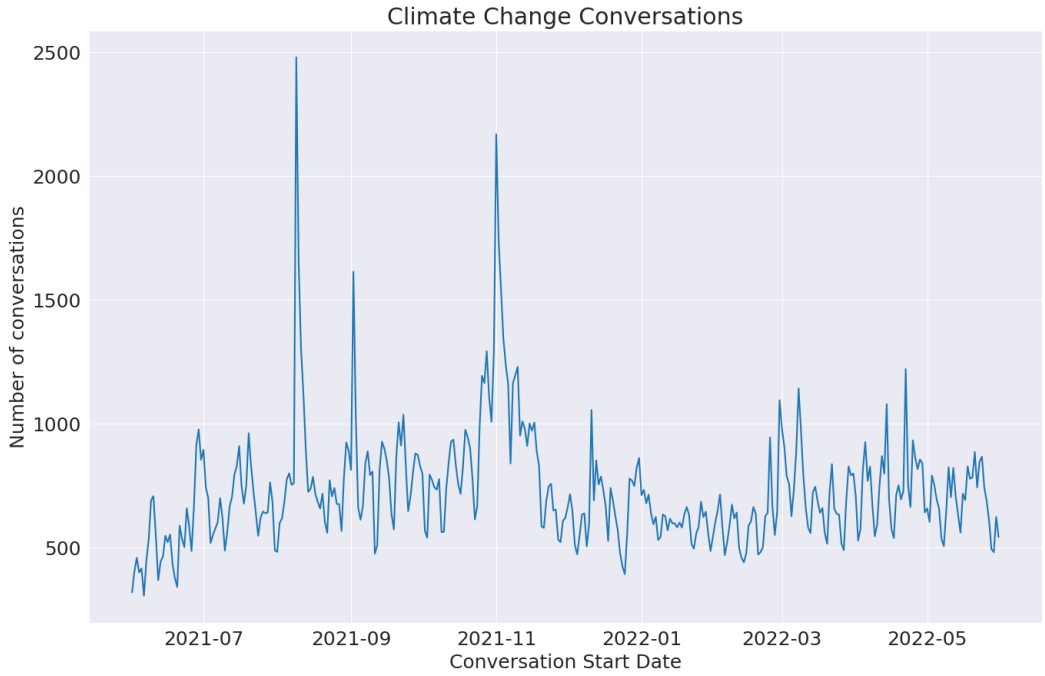


Figure 4.3: Timeline of the number of Twitter conversations about climate change started each day.

edge $(u, h) \in E$ indicates that a user u posted a tweet containing hashtag h. The weight $w(u, h)$ represents the number of times the author posted a particular hashtag across all of their authored tweets.

Based on the set of seed hashtags listed in Table 4.5 associated with each stance, we first propagate the stance associated with each seed hashtag across the user-hashtag

Table 4.5: Sets of seed hashtags used for heuristic stance labeling of users in the climate change and gun control datasets.

Climate Change		Gun Control	
Believe	Disbelieve	Pro	Anti
actonclimate	climatechangehoax	guncontrolnow	shallnotbeinfringed
climatecrisis	globalwarminghoax	endgunviolence	righttobeararms
climateaction	globalcooling	gunreform	gunrights
climateemergency	globalwarmingisahoax		
climateactionnow	climatehoax		

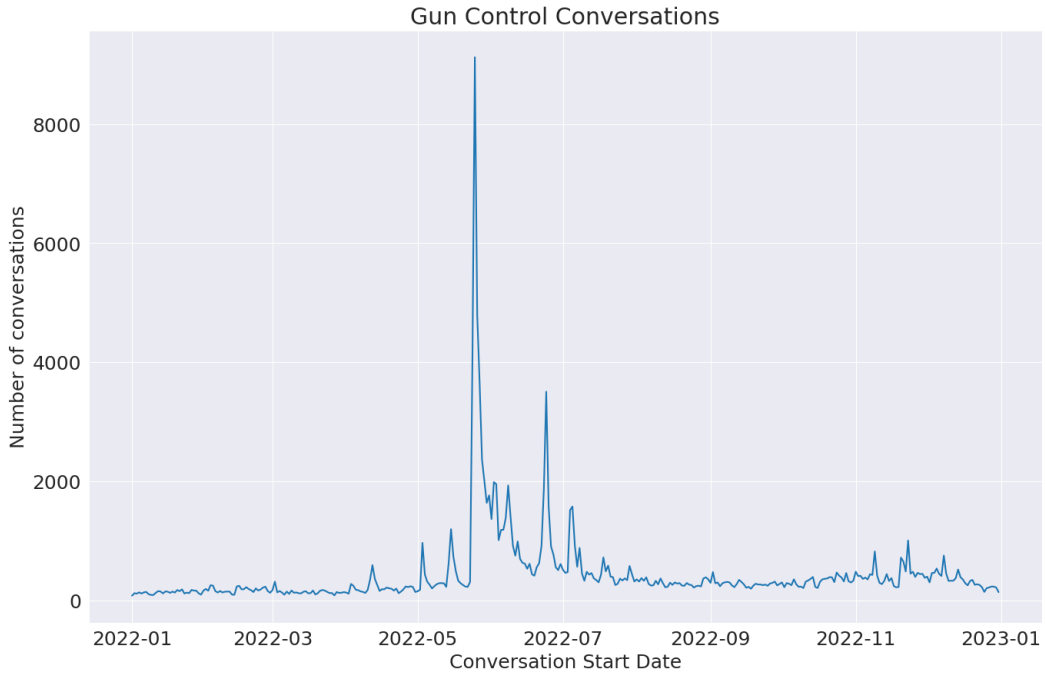


Figure 4.4: Timeline of the number of Twitter conversations about gun control started each day.

bipartite graph to the set of users who posted a given labeled hashtag. The importance of a hashtag's stance to a particular destination user is weighted by the number of times the user posted each hashtag. Users are then assigned a label based on the number of hashtag usages for each stance group. User labels are then propagated back to the hashtags used by each user. Each hashtag is then scored based on the difference between the normalized count of usages from users of each stance group. Hashtag labels are scored according to the method listed in Algorithm 3. Hashtags that score one or more standard deviations above or below the mean are assigned the corresponding stance label. We then repeat alternating label propagation until the model converges, i.e. the set of labeled users is the same from the prior iteration to the next.

While the user-hashtag bipartite label propagation method can quickly label a number of users with high precision. It suffers from the limitation that a large proportion

of users do not use hashtags, and while it is well suited for determining hashtags that are highly associated with one particular stance group as compared with the other, hashtags that are used across both stances albeit in different contexts will remain unlabeled.

4.3.3 GNN Stance Labeling

The user-hashtag stance labeling method can efficiently determine the stance group of users and discover important hashtags associated with each stance group, but it is limited by the fact that not all users post hashtags and that hashtags consist of only a small part of the overall textual content contained in tweets. Furthermore, the method does not take into account any social interactions between users. The hashtag-based heuristic has high precision but relatively low recall due to the limited breadth of users covered by the stance labeling method. To develop a scalable model capable of labeling users in large-scale datasets, we employ the user-hashtag bipartite method as a means to generate a soft-labeled set of users from which we train a graph neural network classifier over the attributed user-user interaction graph.

We define a user interaction as a retweet, mention, reply, or quote from the author of the tweet towards another user. We then construct a weighted, signed, and attributed user-user interaction network $G = (V, E, X)$, where V is the set of users and a directed edge $(u, v)_i \in E$ connects the author user to the target user of their interaction as defined above. The edge weight $w(u, v)_i$ corresponds to the sentiment of the tweet, scored by a BERTweet-based sentiment analysis model [67]. To characterize the overall interactions between users, we consolidate individual tweets between users by combining each directed edge $(u, v)_i$ into a single edge (u, v) with the composite edge weight $w(u, v)$ computed as the mean sentiment score across all tweets between two users. Each user $u \in V$ is also associated with a feature vector $x_u \in X$ corresponding to the element-wise mean of the BERTweet embeddings of all tweets authored by the user. We train a graph neural network model as a stance label classifier using semi-

supervised learning with the set of seed users labeled by our hashtag-based heuristic.

For the GNN, we compare two implementations from the prominent families of graph neural networks—convolutional and attentional. For convolutional GNNs, the neighbor coefficients are fixed weights; whereas, in attentional GNNs these weights are computed implicitly by an attention mechanism. Specifically, we evaluate graphSAGE [8] as an example of a convolutional GNN and graph attention network (GAT) [9] as an example attentional GNN.

GraphSAGE defines a message passing function as:

$$\hat{\boldsymbol{h}}_v^{(k)} = \sigma\left(\mathbf{W}^{\mathbf{k}} \sum_{u \in \tilde{\mathcal{N}}(v)} \frac{1}{|\tilde{\mathcal{N}}|} \boldsymbol{h}_u^{(k-1)}\right) \tag{4.1}$$

where $\tilde{\mathcal{N}}(v)$ is the local neighborhood of node v with added self-loop, $\mathbf{W}$ is a trainable weight matrix, and σ is a non-linear activation function. We use the mean aggregator to aggregate the neighboring node features.

We also consider a GAT layer operator, defined as:

$$\hat{\boldsymbol{h}}_v^{(k)} = \sigma\left(\mathbf{W}^{\mathbf{k}} \sum_{u \in \tilde{\mathcal{N}}(v)} a_{vu} \boldsymbol{h}_u^{(k-1)}\right) \tag{4.2}$$

where aggregation coefficient a_{vu} is computed dynamically in a node-dependent manner via edge softmax attention. In this case, the GAT function is a first-order approximation of a convolution applied over the weighted adjacency matrix, where the edge weights are computed via the attention mechanism.

4.3.4 Results & Discussion

We compare our stance labeling method against six transformer language models that leverage the textual content of tweets to represent users. With each language model, we compute a user embedding by computing the element-wise mean of the embedding of all the users' tweets from BERT, RoBERTa, and BERTweet models.

Table 4.6: Models utilized for user stance labeling and indication whether the models incorporate textual based information and/or network structural information.

Model Type	Model	Text Content	Network Structure
Random	Weighted random	✗	✗
Transformer	BERT-base-uncased	✓	✗
	BERT-large-uncased	✓	✗
	RoBERTa-base	✓	✗
	RoBERTa-large	✓	✗
	BERTweet	✓	✗
	GPT-4	✓	✗
GNN	GraphSAGE	✓	✓
	GAT	✓	✓

Using the set of labeled users from our heuristic method, we train an MLP classifier to predict user stance label.

We evaluate classification performance of the stance labeling methods using a set of manually annotated users and a set of users annotated via GPT-4 using zero-shot classification. Two social science experts independently annotated a set of 250 users for the climate change dataset and 350 users for the gun control dataset by analyzing the top 20 most popular tweets from each user. We provide the same set of tweets to GPT-4 and prompt the LLM to provide the sentiment of content with respect to gun control or climate change. For the gun control dataset, the function call prompt is: "The stance of the content. Is the message pro gun control, anti gun control, or neutral?" with three classification options of "pro", "anti", or "neutral". For the climate change dataset, "The stance of the content. Does the message capture the user's belief in climate change, disbelief, or is it neutral?" with classification options of "belief", "disbelief", or "neutral". We then aggregate the stance label for each individual tweet to determine a user's overall stance. Users who have a majority of tweets belonging to one stance are labeled accordingly. Users whose tweets are all labeled as neutral or have an equivalent number of tweets labeled as both stances are classified as undetermined.

Table 4.7: Results for stance labeling classification on the `Gun Control` dataset ($N = 350$). The best scores for each model type are shown in bold and the best overall scores are underlined.

	GPT4 Labels			Manual Labels		
Model	Prec.	Recall	F1	Prec.	Recall	F1
Weighted random	47.36	49.99	39.45	40.76	49.92	38.11
BERT-base-uncased	**84.25**	85.38	**84.66**	83.08	82.75	82.88
BERT-large-uncased	80.47	82.16	79.96	81.44	81.71	81.13
RoBERTa-base	83.33	83.94	83.60	84.79	83.99	84.25
RoBERTa-large	81.39	82.69	79.52	84.41	84.18	83.14
BERTweet	83.59	**85.43**	83.97	85.45	**85.76**	85.13
GPT-4	-	-	-	**89.81**	85.17	**87.05**
GraphSAGE	**87.58**	88.75	**88.02**	91.09	90.46	90.70
GAT	87.17	**88.99**	87.56	**91.28**	**91.46**	**91.36**

Using the set of manual and GPT-4 annotated users for the two datasets, we compare the classification performance of both text-based methods and our GNN-based method informed by socially infused text mining. We additionally compare the performance of GPT-4 for zero-shot stance labeling of users against the manual annotation by domain experts.

We report the macro averaged precision, recall, and F1 score for each model, and we use the F1 score as the primary metric due to the class imbalance in the datasets. Metrics reported are averaged across five trials for all models.

The classification results for the stance labeling models on the gun control dataset are shown in Table 4.7. On the gun control dataset, BERT and RoBERTa transformer methods achieve a macro averaged F1 score of between 82%-84% when evaluated on the set of manually annotated users. BERTweet, having been finetuned on tweets, surpasses the performance of the other transformer models achieving an F1 score of 85.13%. GPT-4 zero-shot classification outperforms the trained BERTweet classifier by two percent on F1 score and a four percent increase in precision while maintaining an equivalent recall performance.

Table 4.8: Results for stance labeling classification on the `Climate Change` dataset ($N = 250$). The best scores for each model type are shown in bold and the best overall scores are underlined.

	GPT4 Labels			Manual Labels		
Model	Prec.	Recall	F1	Prec.	Recall	F1
Weighted random	47.36	49.99	39.45	45.26	47.55	42.87
BERT-base-uncased	76.74	79.82	77.36	78.60	82.54	79.36
BERT-large-uncased	75.63	79.14	73.86	76.11	80.28	74.17
RoBERTa-base	76.30	79.92	76.21	76.63	80.92	76.50
RoBERTa-large	75.72	79.36	75.16	76.91	81.33	76.29
BERTweet	**77.26**	**80.78**	**77.64**	78.35	82.66	78.78
GPT-4	-	-	-	**85.36**	**86.08**	**85.71**
GraphSAGE	**84.62**	86.06	**85.25**	**83.80**	85.91	**84.67**
GAT	83.73	**86.43**	84.70	83.47	**87.02**	84.62

For the gun control dataset, the GNN-based models that incorporate network structure information with the content-based representations of users produced by BERTweet, outperform both the trained text-based classifiers and zero-shot classification with GPT-4. Using graphSAGE and GAT to enrich users' content-based representations results in an improvement in stance labeling classification compared with the unenriched BERTweet embeddings. With GAT, incorporating users' local neighborhood information with their text-based embeddings improves classification performance by four percent in F1 score. Compared with zero-shot classification using GPT-4, the BERTweet + GAT model performs 3%-4% better in F1 score on classifying user stance labels for the manually labeled user set. Compared with graphSAGE, which takes the mean representation of a node's neighborhood effectively weighting each incident edge on the node equally, GAT utilizes edge softmax attention to dynamically compute edge weights when aggregating node neighborhood information. The dynamic computation of edge importance for users results in a marginal improvement in classification performance.

The classification results for the stance labeling models on the climate change

dataset are shown in Table 4.8. Compared with the gun control dataset, where pro gun control users outnumber anti gun control users by roughly a ratio of 2:1 in the heuristic labeled user set, the imbalance in the climate change dataset is significant with climate change believers outnumbering disbelievers by a ratio of nearly 10:1. We also observe that climate change related conversations are less topic-focused than gun control discussions on Twitter. Due in part to the severe class imbalance in the dataset in addition to the less focused of climate change conversation activity on Twitter, stance determination of users in the climate change dataset is more difficult than in the gun control dataset.

Of the BERT and RoBERTa based classifier models, BERT-base-uncased performs the best with an F1 score of 79.36%, narrowly outperforming BERTweet with an F1 score of 78.78%. Enriching the BERTweet user embeddings with structural information from the user interaction graph improves the classification performance by six percent to 84.67% F1 score. Despite the performance uplift from the user social network, for the climate change dataset zero-shot classification with GPT-4 outperforms all other models with an F1 score of 85.71%, outperforming other transformer methods by six percent and GNN-based approaches by one percent.

In all, enriching text-based representations of users with structural information from the user-interaction graph results in a significant improvement in classification performance compared with the text-based embeddings alone. Zero-shot classification of user stances using GPT-4 consistently outperforms BERT and RoBERTa-based classifier models, but incorporating social network information surpasses GPT-4 classification performance in the gun control dataset and significantly narrows the gap in the climate change dataset.

Furthermore, this study highlights a significant methodological gap in online polarization research, stemming from serious divides in how social science and computer science researchers approach user stance and integrate each other's work. Social sci-

ence provides crucial tools like narrative analysis to grasp the cultural and historical contingencies of online arguments, which inform the creation of collective identity. This qualitative expertise is essential for understanding the human-level subtleties of polarization and radicalization, particularly in the aftermath of events involving radicalized individuals. Qualitative research offers tools like narrative analysis to analyze the cultural and historical contingency of the terms, beliefs, and issues narrators address in their writings [68]. The small narratives created from tweets can provide insights into the range of ongoing societal arguments that create collective identity online [69]. The wealth of this data adds critical information to understanding how people discuss their thoughts and opinions about polarizing topics like gun rights or climate change, and the ability to combine these two approaches can greatly benefit the understanding of online polarization. Conversely, big data and computational methods offer scalability and volume not achievable with traditional methods, which is necessary given the rapid pace of online posting. The core issue is that neither approach is sufficient alone: the integration of subject-matter expertise with AI is critical for applying big data to human behavior research and informing policy.

The difficulty in automating stance detection lies primarily in the necessity to dichotomize beliefs that exist on a spectrum, where the majority of users fall in the nuanced space between clear-cut extremes. Automated approaches frequently mislabel users due to their difficulty in discerning written cues like tone, connotation, coded words, or clarifying pictures—for example, a deceptive tweet text clarified by a contradictory image. Furthermore, users often frame their arguments in neutral language or even the language of their opposition, requiring the full narrative context of a user's history to correctly parse their stance. However, relying solely on human qualitative coding presents its own limitations; it is highly time-consuming and delayed, forcing researchers to limit their scope to a small subset of data. This trade-off between efficiency and accuracy means that human coders will inevitably introduce

miscoded information on ambiguous cases. Ultimately, the paper concludes that both AI and human coding are subject to issues, error, and disagreement, emphasizing that a mixed-methods approach combining the nuance of qualitative research with the scope of computational methods is the necessary force multiplier to accurately and effectively understand the complexity of online polarization.

CHAPTER 5: BIOLOGICAL NETWORK APPLICATIONS

Biological networks, particularly protein-protein interaction (PPI) networks, can be represented as heterogeneous graphs to capture the complexity and context-specific nature of molecular interactions. In this framework, proteins are modeled as nodes, while interactions between them form the edges; however, the heterogeneity arises from incorporating additional dimensions such as tissue type, protein function, or cellular localization. For example, a protein may interact with different partners in the brain than in the liver, and these context-dependent interactions can be encoded as tissue-specific edge types or layered subgraphs. By structuring PPI networks in this way, heterogeneous graphs allow researchers to analyze how protein interactions vary across tissues, uncover functional specialization, and better understand the molecular basis of tissue-specific diseases. This representation supports more precise modeling and inference in systems biology and translational research.

5.1 Link Prediction

5.1.1 Experimental Setup

In the biological domain, we conduct link prediction evaluation using our muxGNN with equivariant coupling on a Tissue Protein-Protein Interaction (PPI)[1]. The details of the dataset are shown in Table 5.1.

We compare our proposed method with five state-of-the-art heterogeneous GNN

[1] http://snap.stanford.edu/ohmnet/

Table 5.1: Summary of the biological network dataset used in our experiments.

Name	# Node	# Edge	# Layer	Type
Tissue PPI	4,360	527,850	10	Biology

frameworks, including MNE [29], R-GCN [15], and R-graphSAGE as examples of convolutional graph neural network models, and GATNE [30], HAN [18] and HGT [19] as examples of attentional GNN models. In addition, we select two random walk-based graph embedding methods, node2vec [65] and DeepWalk [66], for comparison. To adapt these methods to heterogeneous networks, we conduct the random walks independently over each layer graph. To explore the performance of different GNN models as the basis for the layer graph representation, we present results for muxGNN with GCN [6], GAT [9], and GIN [10] as the message passing function in the layer graph component.

To evaluate the performance of our proposed method on link prediction, we conduct one set of experiments in a transductive setting—where all nodes in the test graph are also present in the training graph—and one set of experiments in an inductive context—where some nodes in the test graph are not present in the training graph. For the Tissue Protein-Protein Interaction dataset, we use five-fold cross validation with 80% training links and 5% and 15% positive validation and test links, respectively. For all validation and test sets, an equivalent number of non-existent links are randomly selected from the graph as negative samples to augment the positive links.

The embedding size of all models is set to 200 dimensions with all GNN models using two graph convolution layers corresponding to the 2-hop neighborhood for all nodes. For all random walks, we use 20 walks of length 10 steps with a sliding window context of size 5. The number of edge attention heads in HAN and the GAT variant of muxGNN is set to two, and the heterogeneous attention of GATNE, HAN, and the coupling graph attention size for all muxGNN variants is set to 16 dimensions. For HGT the number of transformer attention heads is set to four. All models are trained for a maximum of 50 epochs with early stopping based on the validation ROC-AUC.

For the inductive experiment, we randomly mask 15% of nodes from the original Tissue PPI network. From the reduced graph, we mask an additional 20% positive

links as a validation set, along with an equivalent number of negative links. The training graph therefore has 85% of nodes and 80% training links of the training node set. In the test graph, 50% of links incident on the masked nodes are re-added with the remaining 50% of links as positive test samples, augmented by an equivalent number of non-existent links incident on the masked nodes as negative samples.

5.1.2 Results & Discussion

The link prediction results in a transductive setting for the biological network, presented in the left columns of Table 5.2, follow a similar pattern to the results observed for social networks, shown in Table 4.3. It can be seen that the random walk based methods continue the trend observed with the social network datasets where their performance notably degrades as the number of different node and edge types increases in the heterogeneous graph. In general, the attentional GNN models GATNE, HAN, and HGT outperform the convolutional graph neural networks MNE, R-graphSAGE, and R-GCN. The notable exception in the performance of GATNE on the Tissue PPI dataset. The node attributes in this dataset are a binary vector indicating biological functions in which the protein participates. These feature vectors are highly sparse compared with the dense features in the other datasets, and we hypothesize that this fact may be responsible for the relative low performance of GATNE on this dataset when compared with the other datasets and baseline models.

The link prediction results in an inductive setting are presented in righthand columns of Table 5.2. For these experiments, we include all baseline models that learn an inductive transformation function and are thus capable of link prediction on nodes previously unseen in training. As in the transductive setting, the GIN variant of muxGNN outperforms all baselines. The GAT variant also performs well, equaling the performance of the strongest baseline competitor in HAN. We observe that muxGNN$_{GAT}$ experiences the smallest drop in performance between the inductive and transductive settings. GATNE experiences an approximately 8% reduction in

Table 5.2: Mean performance (ROC-AUC %, F1 % and PR-AUC %) on link prediction in transductive and inductive contexts.

	Tissue PPI					
	Transductive			Inductive		
	ROC-AUC	F1	PR-AUC	ROC-AUC	F1	PR-AUC
node2vec	51.30	64.04	66.40	-	-	-
DeepWalk	58.48	67.16	69.45	-	-	-
MNE	OOT	OOT	OOT	-	-	-
R-graphSAGE	71.15	65.46	70.71	66.10	61.59	63.15
R-GCN	84.19	75.98	83.08	78.08	70.98	76.81
GATNE-I	79.83	71.78	77.81	71.36	67.64	70.63
HAN	93.05	85.98	93.02	87.53	79.59	88.12
MuxGNN$_{GCN}$	84.58	78.17	78.56	82.47	75.02	82.98
MuxGNN$_{GAT}$	88.40	83.15	79.99	87.48	79.62	88.06
MuxGNN$_{GIN}$	**94.38**	**87.69**	**93.22**	**90.44**	**82.74**	**90.97**

performance in the inductive setting compared with the transductive setting, which we may attribute to the highly sparse node features in this dataset. The other models in our experiments see an approximately 3-5% reduction in performance across all three metrics in the inductive setting compared with the transductive setting. By contrast muxGNN$_{GAT}$ achieves a roughly equivalent performance regardless of whether all nodes in the test set were present during training. Despite this, muxGNN$_{GIN}$ still achieves the best performance on the inductive link prediction task, outperforming all baselines and other muxGNN variants.

5.2 Coupling Graph Attention Analysis

The core component of muxGNN is the coupling graph attention layer. To further analyze the effects of inter-layer coupling, we conduct an ablation study by removing coupling attention from muxGNN. We also investigate the layer coupling attention weight distributions for protein-tissue pairs in the Tissue PPI dataset and show that our coupling attention recovers expected biological connections between tissue layers.

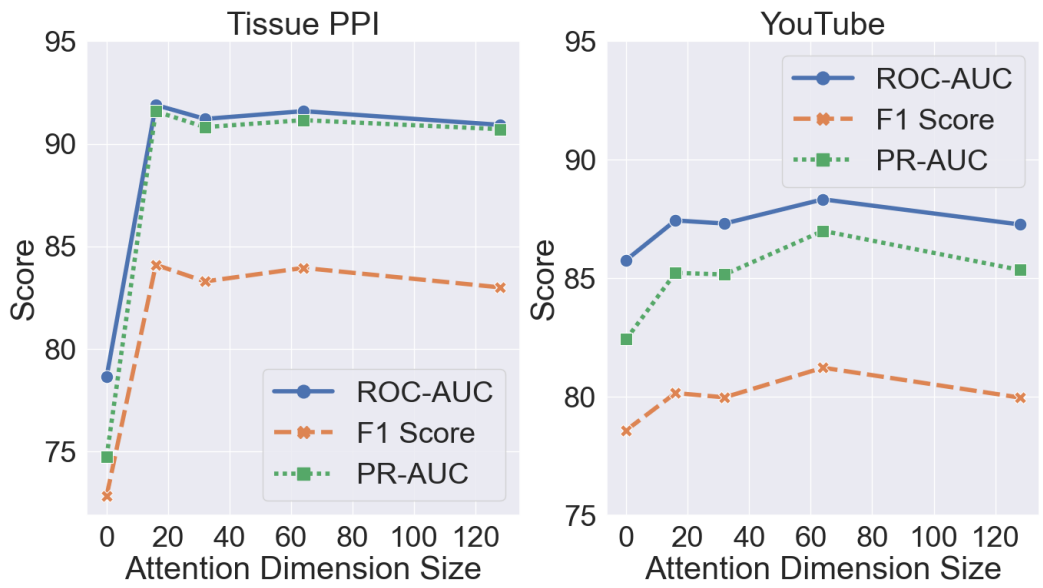


Figure 5.1: Ablation study and dimension analysis of muxGNN's coupling graph attention.

5.2.1 Ablation Study

In the ablation study, the coupling graph attention layer is removed from muxGNN. Each layer graph convolution is performed independently, and representations of a node in a given layer encode only it's local neighborhood information within that layer. Figure 5.1 illustrates the performance of muxGNN$_{GIN}$ on the Tissue PPI and YouTube datasets. As can be seen, removing the coupling graph attention layer from muxGNN reduces overall performance on both datasets; though, the extent of performance decrease varies between datasets. The layers in the YouTube dataset correspond to different user actions on the social network-retweet, quote, or reply. Our link prediction experiments indicate that user interactions of a particular type are largely independent of a user's behavior of other types. That is to say, whether a user is likely to retweet another user is only marginally influenced by that user's reply and quoting behavior. What our experiments indicate is that there is only weak inter-layer coupling in the YouTube dataset, and there is only a marginal perturbation of a node's representation in one layer by the other node-layer pairs. By contrast in

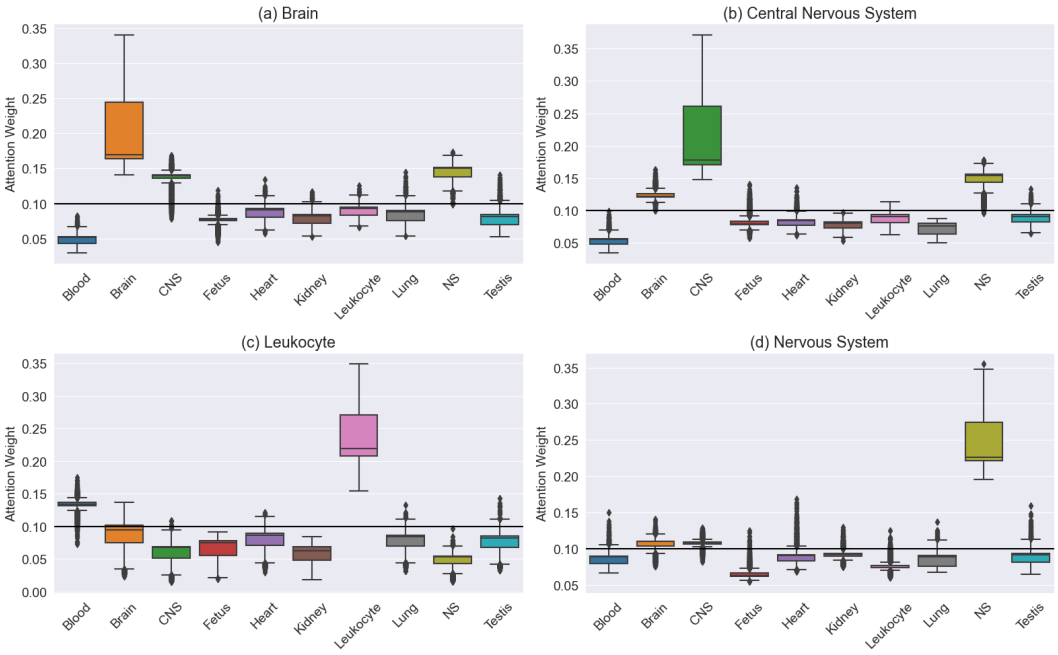


Figure 5.2: Distributions of coupling attention values for each protein in the (a) Brain, (b) CNS, (c) Leukocyte, and (d) NS layers in the Tissue-PPI dataset.

the Tissue PPI dataset, inclusion of the coupling graph attention layer is essential to accurately capture a protein's role across multiple tissue types. Here we observe strong inter-layer coupling that significantly impacts the likelihood of two proteins interacting in any particular tissue. We further explore the coupling attention in the Tissue PPI dataset in the following section.

5.2.2 Tissue PPI Layer Coupling

The coupling graph attention is critical for capturing the multi-faceted context of nodes in heterogeneous networks. In addition to improving the performance of muxGNN on downstream graph learning tasks, we hypothesize that our coupling attention discovers interpretable connections in the network, which provides a level of explainability to muxGNN that is not present in other heterogeneous graph embedding frameworks. We examine the learned attention weights for proteins in each tissue layer in the Tissue PPI dataset. For our experiments, we extract the ten largest layers in network corresponding to the blood, brain, central nervous system (CNS), fetus,

heart, kidney, leukocyte, lung, nervous system (NS), and testis. Figure 5.2 illustrates the distribution of attention values for every protein node in the network with respect to each tissue type in the dataset. It can be seen that the muxGNN's coupling attention identifies the importance of biologically-related tissues in predicting protein interactions. Figure 5.2a shows the attention distribution for the brain layer where we note that both the CNS and nervous system layers are attended to by the model when learning latent representation of proteins in the brain. Similarly, Figure 5.2b shows the attention distributions for the CNS layer, where both the brain and NS layers are important. By contrast, Figure 5.2d shows that the more general nervous system layer does not attend as highly over the brain and CNS layers compared with other tissue layers. These findings indicate that the many proteins roles in the brain and CNS are tightly coupled, which is reflected in the perturbation of layer-specific representations of such protein nodes. Lastly, Figure 5.2c shows that the attention values for the leukocyte layer, where we see that white blood cell proteins attend to other proteins expressed in the blood. In all, our analysis of the attention weight distributions learned by muxGNN illustrate that the coupling graph attention produces interpretable and biologically consistent importance weights across the tissue types present in the protein-protein interaction network.

CHAPTER 6: DATA PROVENANCE NETWORK APPLICATIONS

Data provenance graphs, which track the origin and transformation of data within a system, can be effectively modeled as heterogeneous graphs to capture the variety of entities and actions involved in complex computing environments. In these graphs, nodes represent different types of system elements—such as processes, files, network sockets, or users—while edges denote diverse relationships like read, write, execute, or communication events. This heterogeneity enables the graph to reflect the multifaceted and interdependent nature of system behavior. Such detailed structure is particularly valuable for anomaly detection and Advanced Persistent Threat (APT) detection, where subtle, context-specific deviations from normal activity patterns can signal malicious behavior. By analyzing the semantic relationships and sequences of interactions across different node and edge types, heterogeneous provenance graphs provide a rich foundation for identifying suspicious activity paths, lateral movement, and data exfiltration attempts that may evade simpler monitoring approaches.

6.1 Graph Classification

Graph classification on data provenance graphs is a critical problem focused on determining if a graph representing data lineage and operations is benign (normal) or malicious (an attack). These graphs capture how data, processes, and resources interact over time, making them invaluable for cybersecurity forensics. The core challenge lies in learning robust features from the complex, often large, and evolving graph structure that can accurately distinguish subtle patterns of an attack, such as data exfiltration or unauthorized access, from legitimate system behavior.

Table 6.1: Summary of data provenance datasets used in our experiments. For the graph classification datasets, the average number of nodes, edges, and relation layers is reported.

Name	# Graph	# Node	# Edge	# Layer	Type
StreamSpot	600	8,411	149,618	19	Data Provenance
Unicorn-SC1	150	261,133	969,141	48	Data Provenance

6.1.1 Experimental Setup

To evaluate muxGNN on the task of heterogeneous graph classification, we conduct a set of experiments to classify data provenance graphs as generated from benign execution patterns or from attack executions. We utilize two advanced persistent threat (APT) datasets and accompanying baseline models from StreamSpot [70, 71] and Unicorn [72]. The StreamSpot model uses Locality Sensitive Hashing (LSH) to compute vector representations of graphs that preserve similarity. The StreamSpot dataset consists of data provenance graphs generated from normal web browsing activity, video gaming, and software downloading as a benign class. The positive class consists of graphs generated from a drive-by-download attack on the host machine via a malicious URL. The Unicorn model uses the Weisfeiler-Lehman (WL) subtree kernel to embed graphs while preserving similarity, and its associated dataset provides data provenance graphs generated from CamFlow [73], a whole-system data provenance monitoring tool. Benign system executions compose the negative class, and the positive class is composed of a set of graphs generated from Trojan horse and remote code execution attacks via a malicious URL. Both models rely on graph summarization techniques to generate representations of graphs. To compute a graph level embedding, a *sum* readout function is applied to the set supra-node embeddings of a graph. We set the graph embedding dimension to 64 and use a single hop neighborhood for neighbor aggregation. An MLP classifier is jointly trained with the muxGNN embedding model using cross entropy loss. For both datasets, we use five-fold cross

Table 6.2: Mean performance (Accuracy, Precision, Recall, and F1) of different methods on data provenance graph classification. Only two digits of precision reported by [72]. Recall and F-score not reported by [70].

	StreamSpot				Unicorn-SC1			
	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
StreamSpot	0.66	0.74	-	-	-	-	-	-
Unicorn (k = 1)	0.51	1.0	0.60	0.68	-	-	-	-
Unicorn (k = 3)	0.98	0.93	**0.96**	0.94	0.90	0.85	**0.96**	**0.90**
MuxGNN$_{GCN}$ (k=1)	0.9916	1.0	0.9470	0.9727	0.9402	0.9167	0.7229	0.8078
MuxGNN$_{GAT}$ (k=1)	0.9916	1.0	0.9470	0.9727	0.9649	1.0	0.7667	0.8677
MuxGNN$_{GIN}$ (k=1)	**0.9919**	**1.0**	0.9516	**0.9752**	**0.9674**	**1.0**	0.7967	0.8866

validation, such that our model is trained on 80% of the available graphs with 20% of graphs held out as an unseen test set. We report the mean classification accuracy, precision, recall, and F1 score across all test folds.

6.1.2 Results & Discussion

The graph classification results are presented in Table 6.2. MuxGNN achieves near perfect performance on the StreamSpot dataset, whose isolated scenarios capture simplified workloads and are not representative of typical workloads in modern systems. That said, such graphs provide a standard for comparing different threat detection models and are a similar design to many of today's microarchitecture services. Compared with StreamSpot's baseline model, both the graph kernel based method and muxGNN's graph neural network approach realize a significant increase in performance. When compared with Unicorn's graph kernel approach, considering both the single-hop and three-hop neighborhoods of nodes, we see that muxGNN outperforms both variants when considering only a one-hop neighborhood of nodes.

Methods based on the Weisfeiler-Lehman isomorphism test and subtree kernel are well-known for their discriminative power and have many connections to graph neural network methods. Graph neural networks can be seen as an extension of Weisfeiler-Lehman graph isopmorhism test with non-linear neural networks employed in place of the hashing function [8, 10]. These improvements provide GNNs with greater

expressive capability when transforming graph signals over the local topology of the network.

The Unicorn dataset consists of only 150 total graphs, which is a very limited size for training neural network models. The small, unbalanced dataset along with the heterogeneity and large individual size of the provenance graphs presents a challenging dataset for GNNs optimized via gradient descent both in terms of convergence and avoiding overfitting. Despite these challenges, the one-hop muxGNN model is competitive with the corresponding three-hop Unicorn model. MuxGNN achieves a significant improvement in overall accuracy and in precision, while Unicorn maintains strong performance in recall, with both models overall F1-score approximately equivalent. In addition, the increased computation and memory complexity of Unicorn requires summarization of the provenance graphs prior to embedding generation. In such a way, muxGNN better captures the vertical heterogeneity of a node across the multiple activity types through its coupling attention. By leveraging the advantages of GNNs, such as node batching and weight sharing, muxGNN is able to operate over the entire provenance graphs when computing graph embeddings.

6.2 Advanced Persistent Threats

An Advanced Persistent Threat (APT) is a sophisticated, long-term cyberattack campaign in which a highly skilled, often state-sponsored, actor gains unauthorized access to a network. The primary goal is to remain undetected for an extended period, sometimes months or years, while quietly moving through the network to steal sensitive data, espionage, or prepare for sabotage. We develop two works in collaboration with cybersecurity researchers that utilize muxGNN as a core analysis tool for data provenance graphs in a cohesive effort to tackle both the modeling of threats and the crucial issue of data availability.

PROV-GEM [74] focuses on the analytic methodology, presenting a novel GNN-based solution for graph classification, which uses a multiplex GNN with a relational

self-attention mechanism to explicitly handle the complexity and heterogeneity of provenance graphs. By generating "multi-embeddings" that capture relation-specific context and then aggregating them via semantic attention, the GNN is optimized to discern subtle malicious patterns embedded within the graph's structure, allowing it to classify the entire system state as benign or compromised.

Flurry [75], recognizing that effective GNN training requires realistic and varied data, addresses the data generation problem. It provides the FLURRY framework to dynamically simulate system activity and generate provenance graphs that contain a configurable mix of simultaneous benign and diverse malicious behaviors. This provides the necessary data realism and volume that is lacking in public datasets. Crucially, Flurry complements PROV-GEM's methodology by offering a pipeline to transform raw provenance into attributed heterogeneous graphs containing rich node metadata and multi-faceted node interactions suitable for analysis by multiplex graph neural networks like muxGNN.

CHAPTER 7: CONCLUSION

7.1 Summary of Contributions

This dissertation presents three principal contributions, all rooted in extending graph neural network capabilities to handle complex, heterogeneous data structures. The foundational element is the derivation of a multiplex spectral convolution operator from the first principles of GNNs. This theoretical contribution is crucial, as it provides the mathematical basis for effectively propagating and aggregating information across multiple, distinct types of relationships (layers) that coexist in a single graph, moving beyond the limitations of standard single-layer convolutions.

Building on this operator, the work introduces a new, versatile heterogeneous GNN framework called muxGNN. The framework's core innovation is a novel coupling attention mechanism. Unlike simpler aggregation schemes, this mechanism allows muxGNN to dynamically and adaptively weight the contribution of each layer and the interactions between them. This focus on "coupling" allows the model to selectively integrate heterogeneous information that is most relevant for a given task, making muxGNN highly adaptable and applicable to fundamental graph learning problems such as node classification, link prediction, and graph classification.

Finally, the work validates muxGNN's versatility and performance across three diverse and challenging application domains. In social networks, the model proves effective in important network mining tasks like hate speech detection, link prediction, and stance labeling. For biological networks, it not only achieves strong performance in link prediction, in both transductive and inductive scenarios, but also provides a key contribution to model transparency by establishing the interpretability of its coupling attention mechanism. This means the model can shed light on which specific

biological relationships are most critical for its predictions. Lastly, in the domain of data provenance graphs, muxGNN successfully tackles graph classification problems, including the critical detection of advanced persistent threats and various malicious system behaviors, demonstrating its utility in cybersecurity and system anomaly detection.

7.2 Limitations

While this work offers important contributions, it is not without limitations. The muxGNN framework, while capable of expressing complex relationships between entities in heterogeneous graphs, scales in parameter count with the number of relations in the graph, limiting the suitability of the model to extremely large graphs with high relation counts—such as large knowledge graphs. Techniques such as bases decomposition and block diagonal decomposition [15] have been proposed as means for relational GNNs to address this parameter scaling issue. In addition to parameter scaling, muxGNN—and GNNs in general—are limited by domain specificity. Additionally, graph neural networks, including muxGNN, are most effective when trained on a specific domain set. They are capable of strong performance on in-domain data but generally do not extend well to other domains. With the impressive performance of foundational language models and their applicability to a wide variety of domains. The area of graph machine learning lags behind in developing large scale, diverse training datasets and architectures capable of exploiting ubiquitous but highly variable network data that exists in the world. Thirdly, as muxGNN relies on the neighbor aggregation paradigm of GNNs, it suffers from the oversmoothing problem inherent in this architectural style. It can extract important local relational information from the network and by stacking layers expand to subsume a larger subgraph of the network, but muxGNN is limited in information it can extract from the global context in networks.

7.3 Future Research Directions

The current landscape of multiplex GNNs is constrained by fundamental limitations inherited from general graph neural networks, particularly concerning the trade-off between expressive power, model depth, and handling complex relational schemas. Extending multiplex GNNs to more complex graph structures, including hierarchical dependencies and dynamic temporal graphs, and addressing the limits of expressivity of GNNs form two of the major areas for future improvement and research in the domain of multiplex graph neural networks.

7.3.1 Temporal Multiplex Graph Neural Networks

7.3.1.1 Discrete-time Temporal muxGNN

A discrete-time extension of the muxGNN framework is designed to model evolving systems by treating layers in the multiplex structure as discrete time slices. Each layer, L_t, represents the graph structure at time t. The GNN processes the node features and local topology within each layer L_t, generating a time-specific node embedding $h_v^{(t)}$, for node v. To integrate the temporal dimension, a masked attention mechanism is introduced, which acts as the inter-layer coupling. Specifically, when computing the updated embedding for node v at time $t+1$, $h_v^{(t+1)}$, this mechanism allows the node to attend only to its own previous embeddings from all past time slices $\{h_v^{(1)}, h_v^{(2)}, \ldots, h_v^{(t)}\}$. This causal masking ensures that the information flow respects the direction of time, effectively coupling past events for nodes across the entire sequence of time slices and enabling the GNN to learn time-dependent node representations while maintaining the expressive power of spatial message passing within each layer.

7.3.1.2 Continuous-time Temporal muxGNN

A continuous-time extension of the muxGNN framework retains the relational layer design of the multiplex graph used in muxGNN. This multiplex structure dynami-

cally grows as edges are added to the graph in a continuous-time fashion. To process intra-relational, inter-relational, and temporal dimensions of such as system, the relation-specific encoding and coupling attention mechanism of muxGNN are utilized to capture the heterogeneous structural information in the network. A masked temporal attention mechanism captures the temporal information for nodes by selectively retaining and integrating information from past events that are most relevant to a specific node, providing a persistent event-driven memory to inform a node's representation.

7.3.2 Oversmoothing Problem

The reliance of standard GNNs on iterative neighborhood aggregation poses a significant barrier to scalability and representation power in multiplex systems. The over-smoothing issue fundamentally restricts GNN models from stacking more aggregation layers. This phenomenon, observed across various GNN variants including GTN, HGTN, and MHGCN, dictates that increasing convolutional depth beyond a shallow limit results in indistinguishable node vectors, thus severely constraining the representation quality of high-order heterogeneous relations. The constraint imposed by the message-passing paradigm necessitates innovative architectural strategies that decouple global information capture from layer depth. A promising direction involves methods like Behavior Pattern Modeling (BBP) [76], which define and aggregate basic behavioral patterns across layers to capture local information. By aggregating high-level patterns or structural motifs rather than raw neighbor features, multiplex GNNs can access high-order information and global context without relying on the physical stacking of aggregation layers, thereby mitigating the depth barrier while retaining expressive power.

The capacity of GNNs to handle long-range dependencies is intrinsically tied to their depth. To circumvent the over-smoothing barrier, future multiplex GNNs must be augmented with explicit structural awareness, thereby reducing the reliance on

message passing for propagating positional context. Positional Encodings are crucial for providing position awareness and increasing the expressive capacity of GNNs, especially in architectures that borrow principles from transformers. In particular, developing positional encodings that are specifically tailored to multiplex topology, integrating both global and layer-specific adjacency information into the structural encoding process. By pre-encoding global structural context in a computationally efficient manner, multiplex GNNs can utilize shallower aggregation layers primarily for semantic fusion, significantly mitigating the constraints imposed by traditional message passing depth

7.3.3 Theoretical Expressive Power

Furthermore, while this work takes strides in solidifying the theoretical foundation of multiplex GNNs for heterogeneous graphs, the theoretical understanding of these models has lagged behind empirical architectural development, particularly concerning their representation power and asymptotic limits. GNN expressiveness is traditionally benchmarked against the Weisfeiler-Lehman (WL) hierarchy. However, this measure is fundamentally coarse, qualitative, and often fails to capture crucial practical requirements, such as a model's ability to encode substructures necessary for complex tasks. Transitioning to quantitative expressivity measures, such as homomorphism expressivity [77], which rigorously quantifies a GNN's ability to count graph substructures under homomorphism. This provides a complete framework for direct expressivity comparisons between complex MGNN architectures. Concurrently, for modeling the properties of extremely large, dense multiplex networks, the theoretical foundation must include the limit theory of dense multiplex networks, analogous to graphons for simple graphs [78]. This emerging theory, termed "Multiplexons" [79], is essential for deriving reliable structural features like limiting degree distributions and clustering coefficients.

7.3.4 Multi-modality & LLM Integration

Finally, real-world multiplex networks are almost universally feature-rich, encompassing data beyond simple graph structure (e.g., text, images, temporal signals). Future multiplex GNNs must leverage advanced multi-modal integration techniques and the capabilities of large foundation models. Important future work remains for integrating graph machine learning with the rapid advances in large language models (LLMs) for multi-modal learning. The next stage of multiplex GNNs involves leveraging LLMs not just for generating high-dimensional node attribute vectors from rich associated text (e.g., user reviews or scientific abstracts), but also for more proactive tasks. LLMs can be used to improve the graph topological structure by inferring complex, unobserved relations or assisting in the selection and refinement of meta-paths based on external knowledge, thereby addressing the persistent meta-path bias problem. Leveraging LLMs in this fashion transforms multiplex GNNs into active knowledge synthesis systems capable of handling diverse modalities of data in a dynamic environment, providing better real-world functionality to deliver robust models for complex interdependent systems.

REFERENCES

[1] M. M. Bronstein, J. Bruna, T. Cohen, and P. Velickovic, "Geometric deep learning: Grids, groups, graphs, geodesics, and gauges," *CoRR*, vol. abs/2104.13478, 2021.

[2] S. Bhagat, G. Cormode, and S. Muthukrishnan, "Node classification in social networks," in *Social Network Data Analytics*, pp. 115–148, Springer, 2011.

[3] B. Taskar, M.-F. Wong, P. Abbeel, and D. Koller, "Link prediction in relational data," in *NeurIPS*, p. 659–666, 2003.

[4] C. Shi, B. Hu, W. X. Zhao, and S. Y. Philip, "Heterogeneous information network embedding for recommendation," *TKDE*, vol. 31, no. 2, pp. 357–370, 2018.

[5] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3-5, pp. 75–174, 2010.

[6] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," in *ICLR*, 2017.

[7] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *NeurIPS*, vol. 29, pp. 3844–3852, 2016.

[8] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NeurIPS*, pp. 1024–1034, 2017.

[9] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," in *ICLR*, 2018.

[10] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," in *ICLR*, 2019.

[11] M. E. Newman, "Modularity and community structure in networks," *Proceedings of the national academy of sciences*, vol. 103, no. 23, pp. 8577–8582, 2006.

[12] B. Weisfeiler and A. A. Lehman, "A reduction of a graph to a canonical form and an algebra arising during this reduction," *Nauchno-Technicheskaya Informatsia*, vol. 2, no. 9, pp. 12–16, 1968.

[13] C. Shi, Y. Li, J. Zhang, Y. Sun, and P. S. Yu, "A survey of heterogeneous information network analysis," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 1, pp. 17–37, 2017.

[14] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 5, pp. 833–852, 2018.

[15] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *The Semantic Web*, (Cham), pp. 593–607, Springer International Publishing, 2018.

[16] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *SIGKDD*, pp. 135–144, 2017.

[17] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla, "Heterogeneous graph neural network," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, (New York, NY, USA), p. 793–803, Association for Computing Machinery, 2019.

[18] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, "Heterogeneous graph attention network," in *WWW*, pp. 2022–2032, 2019.

[19] Z. Hu, Y. Dong, K. Wang, and Y. Sun, "Heterogeneous graph transformer," in *WWW*, pp. 2704–2710, ACM / IW3C2, 2020.

[20] B. Hu, Z. Zhang, C. Shi, J. Zhou, X. Li, and Y. Qi, "Cash-out user detection based on attributed heterogeneous information network with a hierarchical attention mechanism," in *AAAI*, pp. 946–953, 2019.

[21] S. Hou, Y. Ye, Y. Song, and M. Abdulhayoglu, "Hindroid: An intelligent android malware detection system based on structured heterogeneous information network," in *KDD*, pp. 1507–1515, 2017.

[22] A. Rossi, D. Barbosa, D. Firmani, A. Matinata, and P. Merialdo, "Knowledge graph embedding for link prediction: A comparative analysis," *ACM TKDD*, vol. 15, no. 2, pp. 1–49, 2021.

[23] B. Hu, C. Shi, W. X. Zhao, and P. S. Yu, "Leveraging meta-path based context for top-n recommendation with a neural co-attention model," in *KDD*, pp. 1531–1540, ACM, 2018.

[24] Y. Fan, S. Hou, Y. Zhang, Y. Ye, and M. Abdulhayoglu, "Gotcha-sly malware! scorpion a metagraph2vec based malware detection system," in *KDD*, pp. 253–262, 2018.

[25] Y. Ye, S. Hou, L. Chen, J. Lei, W. Wan, J. Wang, Q. Xiong, and F. Shao, "Out-of-sample node representation learning for heterogeneous graph in real-time android malware detection," in *IJCAI*, pp. 4150–4156, 2019.

[26] S. Hou, Y. Fan, Y. Zhang, Y. Ye, J. Lei, W. Wan, J. Wang, Q. Xiong, and F. Shao, "αcyber: Enhancing robustness of android malware detection system against adversarial attacks on heterogeneous graph based model," in *CIKM*, pp. 609–618, 2019.

[27] Y. Cao, H. Peng, and P. S. Yu, "Multi-information source HIN for medical concept embedding," in *PAKDD (2)*, vol. 12085 of *Lecture Notes in Computer Science*, pp. 396–408, Springer, 2020.

[28] A. Hosseini, T. Chen, W. Wu, Y. Sun, and M. Sarrafzadeh, "Heteromed: Heterogeneous information network for medical diagnosis," in *CIKM*, pp. 763–772, 2018.

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

[30] Y. Cen, X. Zou, J. Zhang, H. Yang, J. Zhou, and J. Tang, "Representation learning for attributed multiplex heterogeneous network," in *SIGKDD*, pp. 1358–1368, 2019.

[31] H. Hong, H. Guo, Y. Lin, X. Yang, Z. Li, and J. Ye, "An attention-based graph neural network for heterogeneous structural learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, pp. 4132–4139, 2020.

[32] H. Chen, H. Yin, W. Wang, H. Wang, Q. V. H. Nguyen, and X. Li, "Pme: projected metric embedding on heterogeneous networks for link prediction," in *KDD*, pp. 1177–1186, ACM, 2018.

[33] J. L. Suárez, S. García, and F. Herrera, "A tutorial on distance metric learning: Mathematical foundations, algorithms and software," *arXiv preprint arXiv:1812.05944*, 2018.

[34] L. Xu, X. Wei, J. Cao, and P. S. Yu, "Embedding of embedding (eoe): Joint embedding for coupled heterogeneous networks," in *WSDM*, pp. 741–749, ACM, 2017.

[35] B. Hu, Y. Fang, and C. Shi, "Adversarial learning on heterogeneous information networks," in *KDD*, pp. 120–129, ACM, 2019.

[36] Y. Shi, H. Gui, Q. Zhu, L. Kaplan, and J. Han, "Aspem: Embedding learning by aspects in heterogeneous information networks," in *SDM*, pp. 144–152, SIAM, 2018.

[37] Y. Shi, Q. Zhu, F. Guo, C. Zhang, and J. Han, "Easing embedding learning by comprehensive transcription of heterogeneous information networks," in *KDD*, pp. 2190–2199, ACM, 2018.

[38] T.-y. Fu, W.-C. Lee, and Z. Lei, "Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning," in *CIKM*, pp. 1797–1806, ACM, 2017.

[39] S. Lee, C. Park, and H. Yu, "Bhin2vec: Balancing the type of relation in heterogeneous information network," in *CIKM*, pp. 619–628, 2019.

[40] S. Zhou, J. Bu, X. Wang, J. Chen, and C. Wang, "Hahe: Hierarchical attentive heterogeneous information network embedding," *arXiv preprint arXiv:1902.01475*, 2019.

[41] X. Fu, J. Zhang, Z. Meng, and I. King, "Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding," in *Proceedings of The Web Conference 2020*, WWW '20, (New York, NY, USA), p. 2331–2341, Association for Computing Machinery, 2020.

[42] Y. Chang, C. Chen, W. Hu, Z. Zheng, X. Zhou, and S. Chen, "Megnn: Meta-path extracted graph neural network for heterogeneous graph representation learning," *Knowledge-Based Systems*, vol. 235, p. 107611, 2022.

[43] X. Wang, M. Zhu, D. Bo, P. Cui, C. Shi, and J. Pei, "Am-gcn: Adaptive multi-channel graph convolutional networks," in *Proceedings of the 26th ACM SIGKDD International conference on knowledge discovery & data mining*, pp. 1243–1253, 2020.

[44] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson, "Structured sequence modeling with graph convolutional recurrent networks," in *Neural Information Processing* (L. Cheng, A. C. S. Leung, and S. Ozawa, eds.), (Cham), pp. 362–373, Springer International Publishing, 2018.

[45] Y. Huang, Y. Weng, S. Yu, and X. Chen, "Diffusion convolutional recurrent neural network with rank influence learning for traffic forecasting," pp. 678–685, 08 2019.

[46] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pp. 3634–3640, International Joint Conferences on Artificial Intelligence Organization, 7 2018.

[47] C. Chen, K. Li, X. Zou, and Y. Li, "Dygnn: Algorithm and architecture support of dynamic pruning for graph neural networks," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pp. 1201–1206, 2021.

[48] A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, T. Kaler, T. B. Schardl, and C. E. Leiserson, "EvolveGCN: Evolving graph convolutional networks for dynamic graphs," in *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

[49] A. Sankar, Y. Wu, L. Gou, W. Zhang, and H. Yang, "Dysat: Deep neural representation learning on dynamic graphs via self-attention networks," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, WSDM '20, (New York, NY, USA), p. 519–527, Association for Computing Machinery, 2020.

[50] E. Rossi, B. Chamberlain, F. Frasca, D. Eynard, F. Monti, and M. Bronstein, "Temporal graph networks for deep learning on dynamic graphs," in *ICML 2020 Workshop on Graph Representation Learning*, 2020.

[51] S. Kumar, X. Zhang, and J. Leskovec, "Predicting dynamic embedding trajectory in temporal interaction networks," in *Proceedings of the 25th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2019.

[52] da Xu, chuanwei ruan, evren korpeoglu, sushant kumar, and kannan achan, "Inductive representation learning on temporal graphs," in *International Conference on Learning Representations (ICLR)*, 2020.

[53] G. H. Nguyen, J. B. Lee, R. A. Rossi, N. K. Ahmed, E. Koh, and S. Kim, "Continuous-time dynamic network embeddings," in *Companion Proceedings of the The Web Conference 2018*, WWW '18, (Republic and Canton of Geneva, CHE), p. 969–976, International World Wide Web Conferences Steering Committee, 2018.

[54] J. Melton and S. Krishnan, "muxGNN: Multiplex graph neural network for heterogeneous graphs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 9, pp. 11067–11078, 2023.

[55] E. Cozzo, G. F. d. Arruda, F. A. Rodrigues, and Y. Moreno, "Multilayer networks: metrics and spectral properties," in *Interconnected networks*, pp. 17–35, Springer, 2016.

[56] M. Kivelä, A. Arenas, M. Barthelemy, J. P. Gleeson, Y. Moreno, and M. A. Porter, "Multilayer networks," *Journal of Complex Networks*, vol. 2, no. 3, pp. 203–271, 2014.

[57] A. Solé-Ribalta, M. De Domenico, N. E. Kouvaris, A. Díaz-Guilera, S. Gómez, and A. Arenas, "Spectral properties of the laplacian of multiplex networks," *Phys. Rev. E*, vol. 88, p. 032807, Sep 2013.

[58] R. Sanchez-Garcia, E. Cozzo, and Y. Moreno, "Dimensionality reduction and spectral properties of multiplex networks," *Physical review. E, Statistical, nonlinear, and soft matter physics*, vol. 89, 11 2013.

[59] Z. Lin, M. Feng, C. D. Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, "A structured self-attentive sentence embedding," *ArXiv*, vol. abs/1703.03130, 2017.

[60] J. Melton, S. Reid, G. Terejanu, and S. Krishnan, "Two-stage stance labeling: User-hashtag heuristics with graph neural networks," in *Social Networks Analysis and Mining* (L. M. Aiello, T. Chakraborty, and S. Gaito, eds.), (Cham), pp. 245–261, Springer Nature Switzerland, 2025.

[61] J. Melton, A. Bagavathi, and S. Krishnan, "Del-hate: A deep learning tunable ensemble for hate speech detection," in *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 1015–1022, 2020.

[62] T. Davidson, D. Warmsley, M. Macy, and I. Weber, "Automated hate speech detection and the problem of offensive language," in *Proceedings of the 11th International AAAI Conference on Web and Social Media*, ICWSM '17, pp. 512–515, 2017.

[63] M. Zampieri, S. Malmasi, P. Nakov, S. Rosenthal, N. Farra, and R. Kumar, "Predicting the Type and Target of Offensive Posts in Social Media," in *Proceedings of NAACL*, 2019.

[64] E. Raisi and B. Huang, "Weakly supervised cyberbullying detection using co-trained ensembles of embedding models," in *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pp. 479–486, 2018.

[65] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *SIGKDD*, pp. 855–864, 2016.

[66] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *SIGKDD*, p. 701–710, 2014.

[67] D. Q. Nguyen, T. Vu, and A. Tuan Nguyen, "BERTweet: A pre-trained language model for English tweets," in *EMNLP* (Q. Liu and D. Schlangen, eds.), pp. 9–14, ACL, Oct. 2020.

[68] G. C. Rosenwald and R. L. Ochberg, *Storied lives: The cultural politics of self-understanding*. Yale University Press, 1992.

[69] C. Coupland and A. D. Brown, "Constructing organizational identities on the web: A case study of royal dutch/shell," *Journal of management studies*, vol. 41, no. 8, pp. 1325–1347, 2004.

[70] E. Manzoor, S. Momeni, V. Venkatakrishnan, and L. Akoglu, "Fast memory-efficient anomaly detection in streaming heterogeneous graphs," *SIGKDD*, 2016.

[71] X. Han, "Streamspot dataset," 2018.

[72] X. Han, T. F. J. Pasquier, A. Bates, J. Mickens, and M. I. Seltzer, "Unicorn: Runtime provenance-based detector for advanced persistent threats," in *NDSS*, The Internet Society, 2020.

[73] T. Pasquier, X. Han, M. Goldstein, T. Moyer, D. Eyers, M. Seltzer, and J. Bacon, "Practical whole-system provenance capture," in *Symposium on Cloud Computing (SoCC'17)*, ACM, 2017.

[74] M. Kapoor, J. Melton, M. Ridenhour, S. Krishnan, and T. Moyer, "Prov-gem: Automated provenance analysis framework using graph embeddings," in *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 1720–1727, 2021.

[75] M. Kapoor, J. Melton, M. Ridenhour, T. Moyer, and S. Krishnan, "Flurry: A fast framework for provenance graph generation for representation learning," in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, CIKM '22, (New York, NY, USA), p. 4887–4891, Association for Computing Machinery, 2022.

[76] C. Fu, G. Zheng, C. Huang, Y. Yu, and J. Dong, "Multiplex heterogeneous graph neural network with behavior pattern modeling," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, (New York, NY, USA), p. 482–494, Association for Computing Machinery, 2023.

[77] B. Zhang, J. Gai, Y. Du, Q. Ye, D. He, and L. Wang, "Beyond weisfeiler-lehman: A quantitative framework for GNN expressiveness," in *The Twelfth International Conference on Learning Representations*, 2024.

[78] H. Pham, T.-A. Ta, T. Jacobs, R. Burkholz, and L. Tran-Thanh, "The graphon limit hypothesis: Understanding neural network pruning via infinite width analysis," 2025.

[79] A. Ganguly and B. B. Bhattacharya, "Multiplexons: Limits of multiplex networks," 2025.