Article:
Xu, H., Sang, S., Bai, P. et al. (3 more authors) (2023) GripNet: Graph information propagation on supergraph for heterogeneous graphs. Pattern Recognition, 133. 108973. ISSN 0031-3203

https://doi.org/10.1016/j.patcog.2022.108973

Pattern Recognition 133 (2023) 108973

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

GripNet: Graph information propagation on supergraph for heterogeneous graphs

Hao Xu[a], Shengqi Sang[b,c], Peizhen Bai[d], Ruike Li[d], Laurence Yang[a], Haiping Lu[d,*]

[a] Department of Chemical Engineering, Queen's University, Kingston, Ontario, Canada
[b] Department of Physics and Astronomy, University of Waterloo, Ontario, Canada
[c] Perimeter Institute for Theoretical Physics, Waterloo, Ontario, Canada
[d] Department of Computer Science, The University of Sheffield, United Kingdom

ARTICLE INFO

Article history:
Received 18 May 2021
Revised 26 July 2022
Accepted 10 August 2022
Available online 12 August 2022

Keywords:
Graph representation learning
Heterogeneous graph
Data integration
Multi-relational link prediction
Node classification

ABSTRACT

Heterogeneous graph representation learning aims to learn low-dimensional vector representations of different types of entities and relations to empower downstream tasks. Existing popular methods either capture semantic relationships but indirectly leverage node/edge attributes in a complex way, or leverage node/edge attributes directly without taking semantic relationships into account. When involving multiple convolution operations, they also have poor scalability. To overcome these limitations, this paper proposes a flexible and efficient **Gr**aph **i**nformation **p**ropagation **Net**work (GripNet) framework. Specifically, we introduce a new *supergraph* data structure consisting of supervertices and superedges. A supervertex is a semantically-coherent subgraph. A superedge defines an information propagation path between two supervertices. GripNet learns new representations for the supervertex of interest by propagating information along the defined path using multiple layers. We construct multiple large-scale graphs and evaluate GripNet against competing methods to show its superiority in link prediction, node classification, and data integration. The code and data are available at https://github.com/nyxflower/GripNet.

1. Introduction

A heterogeneous graph/network contains multiple types of nodes and relations to represent a wide range of real-world data such as information [1–3], social [4,5], biomedical [6] and chemical [7] networks. Modeling its node/relation properties is important in various machine learning (ML) tasks, e.g., node classification [2,8], clustering/community detection [9], knowledge graph completion [1,10], link prediction [6], and recommendation [4,11]. Graph representation learning (GRL), a.k.a. graph embedding, is a popular solution that embeds entities and/or relations into a low dimensional vector space with the topological information and structure of graph preserved and uses the learned representations for downstream tasks [12]. Heterogeneous GRL (HGRL) algorithms can be categorized into three approaches, based on meta path (MPath), message passing (MPass), and relational learning (RL).

Meta-paths are paths connected by heterogeneous edges, with flexible length and edge types [13], e.g., Author-Paper-Conference-Paper-Author (APCPA) in citation networks. MPath-based approaches transform the given heterogeneous graph into other data structures, e.g., multiple homogeneous graphs [2,14,15] or sequences of entities [16,17], according to the meta-paths to simplify downstream tasks. However, the number of meta-paths and their specific choices are determined manually, which can significantly affect the accuracy and memory cost of downstream tasks, particularly when node heterogeneity is high. Therefore, it is challenging to determine the optimal meta-path set that balances performance and complexity. Besides, to leverage edge attributes, e.g., labels, a (much) larger graph needs to be created by converting attributes into additional nodes, making the problem more complex and challenging [18].

In contrast, MPass- (e.g., RGCN [10]) and RL-based approaches (e.g., DistMult [1]) can leverage node/edge attributes naturally. In MPass-based HGRL [6,10,14], node attributes are passed as messages along edges, with the edge information determining how messages are aggregated via graph convolution operations [19]. RL-based HGRL views a heterogeneous graph as a set of triples (i.e., labeled edges) composed of two nodes and their relation (i.e., edge label) and learns a prediction function for such triples and associated attributes. However, both approaches embed all types of nodes/edges into the same vector space, without modeling

* Corresponding author.
E-mail address: h.lu@sheffield.ac.uk (H. Lu).

https://doi.org/10.1016/j.patcog.2022.108973

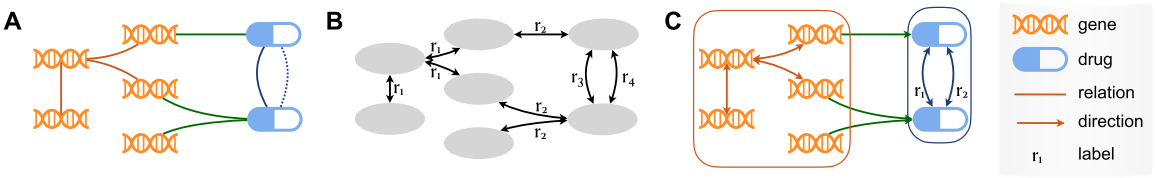

Fig. 1. A motivating example for GripNet: **A** is a biomedical graph containing gene and drug nodes as well as their relations. We are interested in discovering new drug-drug relations within this graph. Previous methods consider genes and drugs holistically and learn their embeddings in the same embedding space, as shown in **B**. In contrast, GripNet aims to learn embeddings of genes and drugs sequentially and in separate embedding spaces, as shown in **C**, leading to more efficient models and better prediction performance.

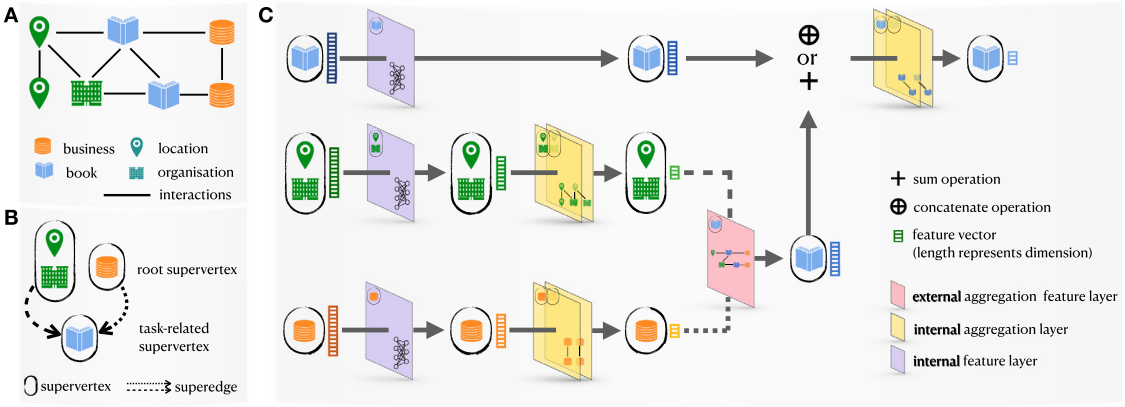

Fig. 2. GripNet illustration (best viewed in color). **A**: A heterogeneous graph H with four *types* of nodes (seven nodes in total), which belong to three semantic *categories* (one color for each category), i.e., two types (location and organization, both in green) are of the same category due to their frequent links. **B**: We segregate the graph in **A** into three *supervertices* connected by two *directed superedges*, forming a directed acyclic *supergraph* $G^S(H)$. The directions of superedges are determined by the *target* of the downstream task, e.g., here we consider the task of book classification, so information flows towards the book supervertex. Each supervertex is a subgraph containing nodes of the same category (color) and edges between them. Each superedge is a *bipartite* subgraph with nodes from two categories (colors) forming two node sets, connected by edges between them. **C**: GripNet architecture for learning new book representation on the supergraph in **B**. Information is propagated from the location&organization (green) and business (orange) supervertices to the book supervertex (blue) via up to three layers. The internal feature layer (purple) reduces the dimension of the original node features. The internal aggregation layer (yellow) aggregates neighborhood information within the supervertex. The external aggregation feature layer (magenta) aggregates information from all parent supervertices. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

the semantics of these entities/relations. Some RL-based methods [20,21] design hand-crafted rule-based features to take the semantic information into account, such as designing a binary *relational feature* based on the logical rule of (drug1,hasTarget,protein1) AND (drug2,hasTarget,protein1). However, these methods not only face similar challenges as determining meta-paths, but also have limited applicability to large-scale, complex HGRL problems due to the need for strong prior knowledge. Besides, because of the expensive graph convolution operations [19,22], scalability is also a major challenge for MPass-based methods.

The above challenges are mainly due to the rich node attributes, relations, and semantic information in heterogeneous graphs. Our main idea is to *segregate the whole graph into semantically-coherent parts, learn embedding within each part, and pass messages between parts following a task-specific propagation path* (Section 3.1).

Figure 1 is a motivating example in a *polypharmacy side effect prediction* task. This task aims to predict undiscovered drug interactions on a given heterogeneous biomedical graph that contains drug and gene nodes and their relations (Fig. 1A). Existing approaches [1,6,21] assume the embeddings of all nodes are in the same embedding space and model all relations in the graph holistically (Fig. 1B). This leads to both high information redundancy and high memory and time costs because the number of gene nodes is typically much larger than that of drug nodes and a side effect edge only links drug nodes. As shown in Fig. 1C, a better way to accomplish the task of side effect prediction is to learn gene and drug embeddings in their respective segregated subgraphs and then propagate gene embeddings to drug embeddings as supplemental information.

To this end, we propose a new HGRL framework named as **Gr**aph **i**nformation **p**ropagation **Net**work (GripNet) to leverage the strengths of three existing approaches:

- Firstly, we introduce a novel *supergraph* data structure that segregates a heterogeneous graph into several semantically-coherent subgraphs, named as *supervertices*, interconnected by heterogeneous bipartite subgraphs, named as *superedges*. By semantically-coherent, we mean that each segregated subgraph contains nodes of the same broad *category* e.g., gene and protein nodes (Section 3.2). We specify the directions of superedges to define the information propagation path between supervertices, in a task-specific manner according to semantic relations and the task of interest. Thus, a supergraph is a directed acyclic graph. Figures 2A and B show an example.
- Secondly, the above supergraph structure enables more efficient and effective embedding. We learn new node representations in supervertices sequentially according to the topological ordering of these supervertices defined by the propagation path (Section 3.3). This allows us to embed nodes in different supervertices into different embedding spaces, e.g., according to their importance/relevance to the task of interest. This not only offers great modeling flexibility, but also improves scalability by enabling efficient implementation, e.g., by choosing low-dimensional embedding space for less important supervertices. Figure 2C shows an example.

GripNet also provides an efficient solution for data integration. We can construct a supergraph from multiple datasets by modeling each dataset as a supervertex and define their relationships by directed superedges to allow flexible data integration. We evaluate

GripNet on link prediction and node classification by constructing seven large-scale complex heterogeneous graph datasets and study its data integration performance. For link prediction, we propose a *categorized negative sampling* strategy to take edge heterogeneity into account. This strategy is generic and can improve the convergence and prediction accuracy of the MPass- and RL-based methods. In addition, we studied the effects of supergraph construction on both the model performance and the memory and time costs, and found only the number of supervertices on the supergraphs has a significant impact on them. Therefore, even if the supergraphs need to be manually determined, GripNet is still simpler than hand-crafted-rules-based methods.

2. Related works

2.1. MPass framework

MPass is a forward-pass phase of Message Passing Neural Networks (MPNN) [7]. It consists of a message function $M_t(\cdot)$ and a vertex update function $U_t(\cdot)$. The MPass phase runs for T time steps, and the hidden states $\mathbf{h}_i^t$ at the tth step for the node i in the graph G are updated based on its messages $\mathbf{m}_i^{t+1}$:

$$\mathbf{m}_i^{t+1} = \sum_{j \in \mathcal{N}_i} M_t(\mathbf{h}_i^t, \mathbf{h}_j^t, \mathbf{e}_{i,j}), \tag{1}$$

$$\mathbf{h}_i^{t+1} = U_t(h_i^t, \mathbf{m}_i^{t+1}), \tag{2}$$

where $\mathcal{N}_i$ denotes the neighbors of node i, and $\mathbf{e}_{i,j}$ denotes the feature of the edge between nodes i and j.

2.2. Encoder-decoder framework for GRL

The Encoder-decoder [12] framework organizes various methods for representation learning on graphs [6,10] around two mapping functions: encoder and decoder. An encoder is a function that maps nodes to low-dimensional vector embeddings, and a decoder is a function that takes a set of node embeddings as input and decodes task-specified graph statistics for downstream machine learning tasks.

2.3. Relational graph convolutional network (RGCN)

The encoder of the RGCN model [10] considers message updates in a multi-relational graph, which regards relations as edge features. The forward propagation of RGCN can be described using the MPass framework with a massage function and an update function:

$$M_t(\mathbf{h}_i^t, \mathbf{h}_j^t, \mathbf{e}_{i,j}) = \sum_{r \in \mathcal{R}} \frac{1}{c_{i,r}} \mathbf{A}_{i,j,r} \mathbf{W}_r^t \mathbf{h}_j^t, \tag{3}$$

$$U_t(\mathbf{h}_i^t, \mathbf{m}_i^{t+1}) = \sigma(\mathbf{m}_i^{t+1} + \mathbf{W}_0^t \mathbf{h}_i^t), \tag{4}$$

where $\mathbf{A}_r$ is the real-valued adjacency matrix under relation $r \in \mathcal{R}$ for the graph, $c_{i,r}$ is a normalization constant that can be learned or chosen according to the task, and $\mathbf{m}_i^{t+1}$ is defined in Eq. (1), and $\sigma(\cdot)$ is an element-wise activate function such as ReLU. RGCN shares weights between different relation types based on basis decomposition to overcome the rapid growth in the number of parameters with the number of relation types increased in the graph. Given a set of basis vectors B, $\mathbf{W}_r$ can be regarded as a layer-wise linear combination of basis transformation $\mathbf{V}_b \in \mathbb{R}^{d^{(l+1)} \times d^{(l)}}$ with relation-associated coefficients $a_{r,b}$:

$$\mathbf{W}_r = \sum_{b \in B} a_{r,b} \mathbf{V}_b. \tag{5}$$

2.4. DistMult factorization

DistMult factorization is a simplified basic bilinear scoring function used as the decoder of the DistMult model [1]. Given node embeddings $\mathbf{z}_i$, $\mathbf{z}_j$ for nodes i, j in the graph G, the score on the edge (i, j) with the edge label l is:

$$f(i, j, l) = (\mathbf{z}_i)^\top \mathbf{M}_l (\mathbf{z}_j), \tag{6}$$

where $\mathbf{M}_l$ is a trainable diagonal matrix.

3. The proposed GripNet model

Representation learning for complex heterogeneous graphs is challenging. Existing approaches can not capture semantic relationships and leverage node and edge attributes at the same time, and those graph convolution-based methods are not scalable to large-scale, complex heterogeneous graphs. We need a different approach.

3.1. Segregate to learn

Our hypothesis is that because the semantics of nodes can vary greatly for a complex heterogeneous graph, it would be beneficial to learn representations of nodes with large semantic differences separately and then propagate the learned information to serve the need of a particular downstream task of interest. This can lead to different embedding spaces for semantically different nodes, and subsequently improve both predictive accuracy and scalability due to the more compact and effective representations. To realize this idea, we need a new framework to support our segregate-to-learn scheme (see Fig. 3). Therefore, we propose a novel graph structure, named as *supergraph*. Let us consider the following mathematical definition of a heterogeneous graph.

Definition 1. A *heterogeneous* graph $H = (V, E, \mathcal{T}, \mathcal{L}, \tau)$ is a graph with multiple types of nodes and whose edges are labeled. V is the set of nodes and $\mathcal{T}$ is the set of node types. $E = \{(i, j, l) | i, j \in V,\ l \in \mathcal{L}\}$ is the set of labeled edges, where $\mathcal{L}$ is the set of edge labels. The function $\tau : V \to \mathcal{T}$ is defined as $\tau(i) = t \in \mathcal{T}$ if node i is of type t.

With this definition, we are ready to introduce the concept of *supergraph*.

3.2. Supergraph structure

Firstly, we define a *categorical partition* of node types $\mathcal{T} = \bigcup_c \mathcal{T}_c$ such that types within each $\mathcal{T}_c$ are *semantically coherent*, i.e., belonging to the same broad *category* $c \in \mathcal{C}$ ($\mathcal{C}$ is the set of all categories). This partition is user-defined and there can be a trade-off between prediction performance and memory cost. In our studies, larger $|\mathcal{C}|$ gives lower memory cost. A naive choice is to assign each type a unique category such that $|\mathcal{C}| = |\mathcal{T}|$, but this choice can lead to worse performance if the semantic coherence between node types is ignored. See detailed studies in Section 4.5.

Now, we are ready to define the *supergraph* for a given heterogeneous graph H to describe our proposed information propagation process. To learn node features of different categories in different feature spaces, we segregate the original graph H into several semantically-coherent subgraphs, named as *supervertices*, interconnected by bipartite subgraphs, named as *superedges*. An example of supergraph construction is given in Steps 1–3 in Fig. 3. Formal definition follows below.

Definition 2. *Supergraph, supervertex*, and *superedge*. Given a heterogeneous graph $H = (V, E, \mathcal{T}, \mathcal{L}, \tau)$ and a categorical partition of its node types $\mathcal{T} = \bigcup_c \mathcal{T}_c$, a supervertex v_c^S for a category $c \in \mathcal{C}$ is

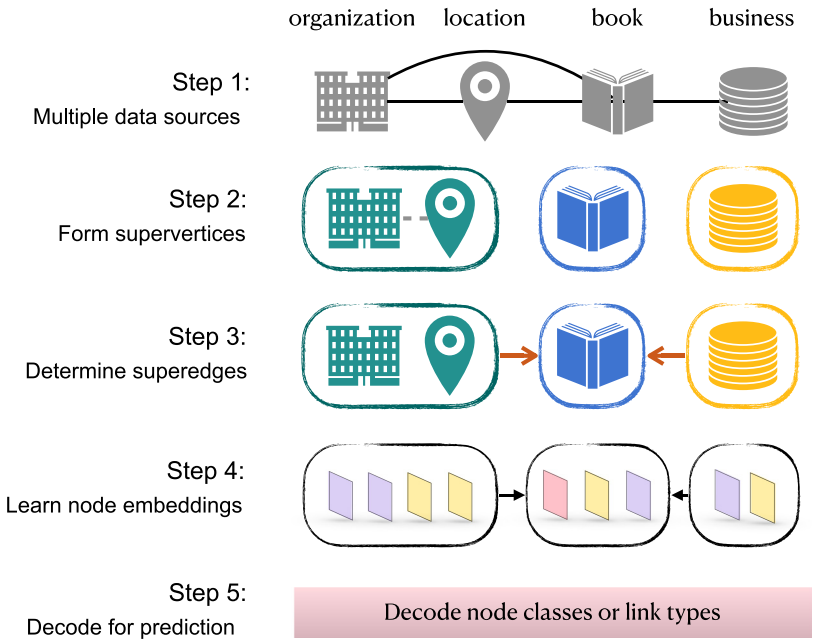

Fig. 3. Five main steps involved in the GripNet model design for a given (integrated) graph-like dataset with heterophily (best viewed in color). The symbol usages are the same as those in Fig. 2. **Step 1**: List all the involved node types and show whether interactions exist among them. **Step 2**: Group the node types to form supervertices. **Step 3**: Determine the information propagation paths among the supervertices to form superedges. **Step 4**: Decide the encoder architecture to learn the node embeddings for each supervertex according to the supergraph's topological ordering. **Step 5**: Input the task-related learned node embeddings to the decoder to make predictions.

defined to be the induced subgraph of H from the set of nodes $V_c = \{v \in V | \tau(v) \in c\}$. A superedge $e^S_{cc'}$ connects two supervertices v^S_c and $v^S_{c'}$. Thus, a superedge is H's bipartite subgraph $e^S_{cc'} = (V_c, V_{c'}, E_{cc'})$, where $E_{cc'} = \{(v, v', l) \in E | v \in V_c, v' \in V_{c'}\}$. We say a superedge $e_{cc'}$ exists if $E_{cc'}$ is non-empty. Finally, a supergragh of H is a directed acyclic graph $G^S(H) = (V^S, E^S)$ where $V^S = \{v^S_c | c \in \mathcal{C}\}$ and $E^S = \{e^S_{cc'} | E_{cc'} \neq \emptyset;\ c, c' \in \mathcal{C}\}$.

Depending on whether the ML task of interest is conducted on a supervertex, we divide the supervertices into a task supervertex and non-task supervertices. In general, valid supervertices should meet the following requirements: 1) Each supervertex does not contain any isolated node or subgraph, and 2) on the task supervertex, the node types are only task-related. These requirements suggest only grouping node types with interactions to form supervertices when performing Step 2 in Fig. 3.

The directions of superedges are defined by users according to two criteria: **1)** the supergraph G^S must be directed acyclic (otherwise we will encounter loopy information propagation) and **2)** the task supervertex should be a leaf vertex of G^S and all information from other supervertices should finally be propagated into it. Step 3 in Fig. 3 shows an example of superedge determination. In addition, as long as these two criteria are satisfied, the direction of information propagation between non-task supervertices can be chosen manually. In practice, the direction of these superedges does not significantly affect the model performance or the memory and time costs (see detailed studies in Section 4.5).

This new supergraph model is generic and can be used for many problems in heterogeneous graphs, e.g., we can use it as an intuitive and flexible data integration model by constructing a supergraph from multiple datasets with each dataset as a supervertex and their relationships defined by the directions of superedges. In the following, we propose our supergraph-based GripNet for heterogeneous graph representation learning. There can be many ways to realize GripNet. Here we present one simple, basic realization that can be extended to many other architectures by varying individual components. Figure 2C shows an example of this simple GripNet implementation.

3.3. GripNet architecture

GripNet learns the embeddings of nodes in supervertices sequentially according to the topological ordering of the supervertices they belong to, which is made possible by designing the supergraph to be a directed acyclic graph [23]. For each supervertex v^S_c, GripNet has a module for learning node embeddings within v^S_c from the information given by v^S_c and its parent supervertices $\mathcal{N}^S_c = \{v^S_{c'} | e^S_{c'c} \in E^S\}$. Conceptually the module is composed of three layers:

The external aggregation feature layer of supervertex v^S_c takes the learned node embeddings from its parent supervertices as input, and transforms them into features for nodes within v^S_c, which are referred to as *external features*. This layer is realized as a set of mutually parallel RGCN encoder sublayers, one for each parent supervertex $v^S_{c'} \in \mathcal{N}^S_c$. The supervertex $v^S_{c'}$'s contribution to the external features of node i in v^S is:

$$\mathbf{r}^{c' \to c}_{i,\text{ex}} = \sum_{l \in \mathcal{L}_{c'c}} \sum_{j \in \mathcal{N}^{cc'}_{i,l}} \frac{1}{|\mathcal{N}^{cc'}_{i,l}|} \mathbf{W}^{(\text{ex})}_l \mathbf{z}^{c'}_j, \tag{7}$$

where $\mathbf{z}^c_j$ is the embedding of node j in $v^S_{c'}$, $\mathcal{L}_{c'c}$ denotes the set of possible edge labels in superedge $e^S_{cc'}$ (a bipartite subgraph), $\mathcal{N}^{cc'}_{i,l}$ is the set of node i's neighbors in $e^S_{cc'}$ linked by an edge of type l, and $\mathbf{W}^{(\text{ex})}$s are learnable parameters. The total external features of node i in v^S_c is obtained by summing over all supervertices' contributions and applying an activation function, such as ReLU:

$$\mathbf{r}^c_{i,\text{ex}} = \text{ReLU}\left(\frac{1}{|\mathcal{N}^S_c|} \sum_{c' \in \mathcal{N}^S_c} \mathbf{r}^{c' \to c}_{i,\text{ex}} \right). \tag{8}$$

Note that if a supervertex doesn't have any parent supervertex, i.e., if it's a root supervertex in the supergraph, then the external feature layer becomes degenerated and we can simply choose the external feature vectors as zero vectors: $\mathbf{r}^c_{i,\text{ex}} = \mathbf{0}$.

The internal feature layer of supervertex v_c^S maps the original node features, typically one-hot vectors, into the same space that external features $\{\mathbf{r}_{i,\text{ex}}^c\}$ live in. This layer is constructed as a linear transformation $\mathbf{W}^{(\text{in})}$ followed by a nonlinear activation function, such as ReLU. Its output on node i in v_c^S, referred to as *internal features*, is:

$$\mathbf{r}_{i,\text{in}}^c = \text{ReLU}(\mathbf{W}^{(\text{in})}\mathbf{x}_i^c), \tag{9}$$

where $\mathbf{x}_i^c$ is the original feature vector of node i. We combine the internal and external features to obtain the *total features* of node i, e.g., by concatenation or summation as

$$\mathbf{r}_i^c = (\mathbf{r}_{i,\text{ex}}^c \oplus \mathbf{r}_{i,\text{in}}^c) \text{ or } (\mathbf{r}_{i,\text{ex}}^c + \mathbf{r}_{i,\text{in}}^c), \tag{10}$$

where $\oplus$ denotes concatenation, and + denotes summation. We set the default in GripNet to be concatenation $\oplus$.

The internal aggregation layer takes the total features $\{\mathbf{r}_i^c\}$ as the input $\mathbf{u}_i^0 = \mathbf{r}_{i,in}^c$, updates them within the supervertex v_c^S, and obtains the final embeddings for nodes in v_c^S. This layer is designed to be composed of T RGCN sublayers concatenated together, and the update rule between the tth and the $(t+1)$th sublayer is:

$$\mathbf{u}_i^{t+1} = \mathbf{ReLU}\left(\sum_{l\in\mathcal{L}_c, j\in\mathcal{N}_{i,l}^c} \frac{1}{|\mathcal{N}_{i,l}^c|}\mathbf{W}_{l,t}^{(\mathbf{ia})}\mathbf{u}_j^t + \mathbf{W}_{0,t}^{(\mathbf{ia})}\mathbf{u}_i^t\right), \tag{11}$$

where $\mathcal{L}_c$ is the set of possible edge labels in v_c^S, $\mathcal{N}_{i,l}^c$ denotes the neighbors of node i linked by edges of type l, and **W**s are learnable parameters. The output of the last RGCN layer $\mathbf{z}_i^c = \mathbf{u}_i^T$ is the final learned embedding of node i in v_c^S. These embeddings can then be used for computing external features of v_c^S 's child supervertices or downstream ML tasks.

Algorithm 1 summarizes the steps in GripNet node embedding.

Algorithm 1 GripNet node embedding.

Input: Heterogeneous graph H and its corresponding user-defined directed acyclic supergraph $G^S(H) = (V^S, E^S)$
Output: Embedding vectors $\{\mathbf{z}_i\}$ for each node i in H.
while V^S is not empty **do**
  $v_c^S \leftarrow$ supervertex with the highest topological order in V^S
  remove v_c^S from V^S
  for each vertex i within v_c^S **do**
    if v_c^S has a parent in G^S **then**
      $\mathbf{r}_{i,ex}^c \leftarrow$ external features calculated using Eqs. (7) and (8)
    else
      $\mathbf{r}_{i,ex}^c \leftarrow 0$
    end if
    $\mathbf{r}_{i,in}^c \leftarrow$ internal features calculated using Eq. (9)
    $\mathbf{r}_i^c \leftarrow$ total features calculated using Eq. (10)
  end for
  $\{\mathbf{u}_i^0\}_{i\in v_c^S} \leftarrow \{\mathbf{r}_i^c\}_{i\in v_c^S}$
  $\{\mathbf{u}_i^n\}_{i\in v_c^S} \leftarrow$ update $\{\mathbf{u}_i^0\}_{i\in v_c^S}$ iteratively T times using Eq. (11)
  $\{\mathbf{z}_i\}_{i\in v_c^S} \leftarrow \{\mathbf{u}_i^n\}_{i\in v_c^S}$
end while
$\{\mathbf{z}_i\} \leftarrow \bigcup_{v_c^S\in V^S}\{\mathbf{z}_i^n\}_{i\in v_c^S}$

Here we consider two popular tasks: link prediction and node classification. We follow the popular encoder-decoder framework in graph representation learning [12]. The GripNet representation learning introduced above learns a new vectorial embedding for each node, which is the GripNet encoder. For downstream tasks, the learned GripNet representation is fed into a GripNet decoder to infer graph properties, such as the existence probabilities of node attributes and edges (links).

3.4. Link prediction

We focus on the case where the edges (links) to be predicted are all located in a single supervertex v_c^S. The method presented below can be slightly modified to predict edges located in a superedge. We formulate link prediction as a binary classification of exist vs non-exist and adopt the DistMult factorization [1] activated by the sigmoid function as the GripNet decoder for link prediction. Given nodes i and j in supervertex v_c^S and a possible edge label $l \in \mathcal{L}_c$, the probability that the edge (i, j, l) exists is computed as:

$$f(i, j, l) = \text{sigmoid}((\mathbf{z}_i^c)^\top \mathbf{M}_l(\mathbf{z}_j^c)), \tag{12}$$

where $\mathbf{M}_l$ is a trainable diagonal matrix associated with the edge label l, and $\mathbf{z}_i$ and $\mathbf{z}_j$ are the embeddings for nodes i and j. To train the model, we need both positive and negative instances of edges. Positive edges are the observed edges E_c, while negative ones require sampling. Traditional negative sampling methods [1,6,10,24] do not take the edge heterogeneity of graphs into account. Thus, we propose a *categorized negative sampling* strategy below for GripNet decoder in link prediction.

Categorized negative sampling (CNS). To sample a negative edge with label l in supervertex $v_c^S = (V_c, E_c)$, we propose a sampler to randomly choose a source-target node pair from the set $\{(i, j)|i, j \in V_c,\ \ (i, j, l) \notin E_c\}$. In contrast, in previous methods negative edges are chosen from the whole heterogeneous graph, and are constructed by corrupting (redirecting) one end of each positive instance. For each edge label l, we enforce the number of negative samples to be the same as that of positive instances in v_c^S.

We use the *cross-entropy loss* to optimize the model, aiming to assign higher probabilities to positive edges and lower probabilities to negative ones,

$$L^{(\text{lp})} = -\sum_{(i,j,l)\in E_c} \log f(i, j, l) - \sum_{(i,j,l)\in N(E_c)} \log(1 - f(i, j, l)), \tag{13}$$

where E_c is the set of positive edges in v_c^S, and $N(E_c)$ is the set of negative edges sampled by categorized negative sampler according to E_c.

3.5. Node classification

In node classification, we consider the task of predicting a specific node attribute with possible values from a set $\mathcal{A}$ for nodes in a given supervertex v_c^S (their types or categories are assumed to be known). The task can be viewed as a multiclass classification problem. Our GripNet decoder for node classification applies a linear transformation $\mathbf{W}^{(\text{nc})}$ followed by a softmax activation function to ith node's embedding vector $\mathbf{z}_i^c$ to obtain the probability distribution vector $\mathbf{p}_i$ over $\mathcal{A}$:

$$\mathbf{p}_i = \text{softmax}(\mathbf{W}^{(\text{nc})}\mathbf{z}_i^c). \tag{14}$$

We minimize the following *cross-entropy* loss during training:

$$L^{(\text{nc})} = -\sum_{i\in V_c,\ a\in\mathcal{A}} s_{ia}\ln(p_{ia}), \tag{15}$$

where $\mathbf{s}$ is the one hot representation of true attributes: $s_{ia} = \mathbb{1}[\text{node } i\text{'s true attribute is } a]$, V_c is the set of nodes in supervertex v_c^S and p_{ia} is the ath entry of $\mathbf{p}_i$.

4. Experiments

We evaluate GripNet on *large-scale*, complex heterogeneous graphs.

Table 1
Dataset statistics. $|V|$: #nodes, $|E|$: #edges, $|V_p|$ ($|E_p|$): #nodes (#edges) to be predicted, $|\mathcal{C}_p|$: #candidate labels, $|\mathcal{T}|$: #node types, $|\mathcal{L}|$: #edge labels. **Items**: average #items to be predicted per candidate label. Node types: the types of nodes contained in datasets, drug(d), gene(g), paper(p), author(a), book(bo), business(bu), organization(o) and location(l).

Dataset	$\|V\|$	$\|\mathcal{T}\|$	$\|E\|$	$\|\mathcal{L}\|$	$\|V_p\|$	$\|E_p\|$	$\|\mathcal{C}_p\|$	Items	Node types
PoSE-0	4,285	2	4,725,690	1100	-	4,625,608	1,097	4,217	d, g
PoSE-1	19,365	2	2,621,423	864	-	1,171,603	861	1,361	d, g
PoSE-2	19,726	2	6,075,428	1100	-	4,625,608	1,097	4,217	d, g
Aminer	397,477	2	1,265,593	3	124,806	-	7	17,829	a, p
Freebase-a	14,989	1	12,556	1	14,989	-	8	1,873	bo
Freebase-b	354,961	2	848,032	3	14,989	-	8	1,873	bo, bu
Freebase-c	457,504	3	998,663	5	14,989	-	8	1,873	bo, bu, o
Freebase-d	1,100,400	4	3,354,079	8	14,989	-	8	1,873	bo, bu, o, l

4.1. Experimental design

4.1.1. Dataset

For link prediction with a large number of different relations, we consider the BioSNAP-Decagon dataset with 6M edges and 1K different edge labels in total [6] by constructing a series of three sub-datasets named as **PoSE-0, -1**, and **-2**, with varying proportions of task-related nodes. For node classification with a large number of labeled nodes, we construct the **Aminer** dataset with 124K labeled nodes and the Freebase-series datasets with four sub-datasets, **Freebase-a, -b, -c** and **-d**, with increasing scale and heterogeneity. Table 1 reports their statistics.

- The **PoSE-series** are three datasets from the BioSNAP dataset collection [25] for **Po**lypharmacy **S**ide **E**ffect (POSE) prediction [6]: PP-Decagon (PP), ChG-InterDecagon (ChG) and ChChSe-Decagon (ChChSe) contain relations between genes, gene and drug, and drugs respectively. We integrate the datasets by GeneID [26] and PubChem CID [27]. The key difference among PoSE-0, 1 and 2 is the number of gene/drug nodes that are not directly linked with any drug/gene.
- **Aminer** is constructed from the public Aminer Academic Network dataset [28]. Following the top venue classification in Google Scholar[1], we select seven popular categories in computer science (CS) and 20 top venues for each category. Following [16], we assign each author node a CS-category label with the most publications.
- The **Freebase-series** are subsets of the Freebase dataset [18] with books classified into 8 categories. As the last column of Table 1 shows, Freebase-a contains book nodes and their relations. Business, organization, and location information are integrated cumulatively to construct the Freebase-b, -c, and -d datasets respectively.

4.1.2. Methods compared

In the PoSE link prediction task, **DECAGON** [6] is a pioneering work but with high computational requirements that we were not able to satisfy. A more recent work [21] showed **DistMult** [1] has better performance and lower cost of memory and time than DECAGON so we choose it for comparison. Three KG embedding models **TransE** [29], **ComplEx** [30] and **RotatE** [31] are compared as they are considered as baselines for multi-relation link prediction by Open Graph Benchmark [32]. We also compare with **RGCN** [10], which shows good performance on standard datasets for heterogeneous graphs. In node classification, we choose one MPass/RL-based method **RGCN** [10], two popular GRL methods **GCN** [19] and **GAT** [33], as well as two recent methods **GCNII** [34] and **Cluster-GCN** [35].

4.1.3. Evaluation metrics

We use the area under the precision-recall curve (AUPRC), the receiver-operating characteristic (AUROC), and average precision at 50 (AP@50) to evaluate link prediction performance, for each edge label first and then take their average. We use the micro-averaged F1 scores (Micro-F1) and macro-averaged F1 scores (Macro-F1) to evaluate node classification. We also evaluate the GPU memory usage during model training using tools in the *pytorch_memlab* package[2], and the computational time.

4.1.4. Experimental configuration

We train and test all considered methods on NVIDIA GV100GL (Tesla V100 PCIe 32 GB VRAM). For fair comparison and to avoid costly tuning on large-scale data, we use the following configurations for all methods:

- set the learnable embeddings' dimension for the nodes of interest to 32 in node classification
- choose the embedding dimension for each model such that training can be performed on the GPU memory budget [32] in link prediction
- initialize weights using *Xavier initialization* [36]
- optimize models end-to-end by full-batch with the *Adam optimizer* [37] with learning rates of 0.01
- train a fixed 100 epochs for all the experiments.

In addition, when implementing methods involving graph convolution operations, an additional embedding layer with trainable parameters $\mathbf{W} \in \mathbb{R}^{|\mathcal{V}|\times 256}$ is added before convolution layers for computational efficiency, as the datasets contain up to millions of nodes. We do a stratified split into 90% for training and 10% for testing for both link prediction and node classification. With the above configuration, we set the embedding dimension of GripNet hidden layers to either 64 or 128 according to the size of the supervertices (Section 4.2), and report the best results on the test sets in node classification. Each experiment is repeated 10 times with different random seeds, and all test results are reported with the mean and unbiased standard deviation.

4.1.5. Model implementation

We implement GripNet models with *PyTorch* [38] and *PyTorch-Geometric* packages [39]. We use the model implementation of GCN, RGCN, and GAT model provided by *PyTorch-Geometric* [39]. The implementation of TransE, RotatE, ComplEx, and DistMult model are provided by the *Open Graph Benchmark* [32]. We use the model implementation of GCNII[3] and Cluster-GCN[4] provided by the authors. Due to the large data size, we improved the RGCN implementation to reduce its memory requirements for the link prediction task.

[1] https://scholar.google.com/citations?view_op=top_venues&hl=en&vq=eng_enggeneral.

[2] https://github.com/Stonesjtu/pytorch_memlab.

[3] https://github.com/chennnM/GCNII.

[4] https://github.com/google-research/google-research/tree/master/cluster_gcn.

Table 2
Model settings for supergraphs with two supervertices. Node type (**NT**) symbols: g, d, a, p, bo and bu are the same as those in Table 1.

	Root Supervertex				Task-related supervertex				
Model	NT	$\mathbf{W}^{(in)}$	$\mathbf{W}_1^{(ia)}$	$\mathbf{W}_2^{(ia)}$	NT	$\mathbf{W}^{(ex)}$	$\mathbf{W}^{(in)}$	$\mathbf{W}_1^{(ia)}$	$\mathbf{W}_2^{(ia)}$
GripNet-0/1/2	g	$\mathbb{R}^{\lvert p\rvert\times 32}$	$\mathbb{R}^{32\times 16}$	$\mathbb{R}^{16\times 16}$	d	$\mathbb{R}^{16\times 16}$	$\mathbb{R}^{\lvert d\rvert\times 32}$	$\mathbb{R}^{(32+16)\times 16}$	-
GripNet-b	bu	$\mathbb{R}^{\lvert bu\rvert\times 128}$	$\mathbb{R}^{128\times 64}$	$\mathbb{R}^{64\times 64}$	bo	$\mathbb{R}^{64\times 64}$	$\mathbb{R}^{\lvert bo\rvert\times 128}$	$\mathbb{R}^{(128+64)\times 64}$	$\mathbb{R}^{64\times 32}$
GripNet-am	p	$\mathbb{R}^{\lvert p\rvert\times 128}$	$\mathbb{R}^{128\times 64}$	$\mathbb{R}^{64\times 64}$	a	$\mathbb{R}^{64\times 64}$	$\mathbb{R}^{\lvert a\rvert\times 128}$	$\mathbb{R}^{(128+64)\times 128}$	$\mathbb{R}^{128\times 32}$

4.2. GripNet supergraph construction

Supergraph construction involves Steps 2–4 in Fig. 3.

Freebase-c and -d datasets. Figure 2 gives an example on supergraph construction for book classification on Freebase-d. In the Freebase-d supergraph, there are **three supervertices**: book v_0^S, location & organization v_1^S, and business v_2^S. The **information propagation paths** are: location& organization → book, and business → book. In each supervertex, the input node features of all the entities use one-hot encoding features, and the outputs are the node embeddings. The location, organization and business node embeddings are learned with 1) an internal feature layer with $\mathbf{W}^{(in)} \in \mathbb{R}^{|V_i|\times 256}$, where V_i are nodes in the ith supervertex, and 2) two internal aggregation layers with $\mathbf{W}_1^{(ia)} \in \mathbb{R}^{256\times 128}$, and $\mathbf{W}_2^{(ia)} \in \mathbb{R}^{128\times 128}$. On the book supervertex, 1) the business, location and organization embeddings are transformed into book features by an external aggregation feature layer with $\mathbf{W}_1^{(ex)}, \mathbf{W}_2^{(ex)} \in \mathbb{R}^{128\times 128}$, 2) the input feature is extracted in an internal feature layer with $\mathbf{W}^{(in)} \in \mathbb{R}^{|V_0|\times 128}$, and 3) the final book embeddings are learned by an internal aggregation layer with $\mathbf{W}_1^{(ia)} \in \mathbb{R}^{256\times 32}$, whose input is the concatenation of the output of external and internal feature layers. The supergraph construction for Freebase-c is similar to that for Freebase-d. It also contains **three supervertices**: book v_0^S, organization v_1^S, and business v_2^S. The **information propagation paths** are: organization → book, and business → book. GripNet implementations for Freebase-c have the same layer settings and dimensions of all the trainable parameters as that for Freebase-d.

PoSE-0, -1, -2, Freebase-b and Aminer datasets. For the datasets with two node types, PoSE-series, Freebase-b, and Aminer, their supergraphs only contain two supervertices and a superedge directed from the root supervertex to the task-related supervertex. See layer settings and trainable parameter dimensions in Table 2. GripNet-0, -1, and -2 are for the polypharmacy side-effect prediction task on datasets PoSE-0, -1, and -2, respectively. Usually, the prediction is made directly on the drug-drug interaction graph in the ChChSe dataset. Assuming that drug-drug interactions are caused by interactions between genes affected by them, integrating the PP, and ChG graph may improve the prediction performance. GripNet-0,1,2 models 1) learn the gene embeddings in the gene (root) supervertex, 2) propagate the information to the drug (task-related) supervertex via an external aggregation feature layer, and 3) learn the drug embeddings and predict side effects on the drug supervertex. For GripNet-b on Freebase-b dataset, the business node embeddings are learned on the business supervertex and are used to learn book embeddings on the book supervertex. For GripNet-am on Aminer dataset, the paper node embeddings are learned on the paper supervertex, then they are propagated to the author supervertex.

Freebase-a dataset. Freebase-a is represented as a homogeneous graph. Its supergraph only contains a book supervertex. On this supervertex, the embeddings are learned without any external aggregation feature layer. The parameter settings are $\mathbf{W}^{(in)} \in \mathbb{R}^{|Bo|\times 128}$, $\mathbf{W}_1^{(ia)} \in \mathbb{R}^{(128)\times 64}$, and $\mathbf{W}_2^{(ia)} \in \mathbb{R}^{64\times 32}$.

4.3. Performance comparison

Tables 3 and 4 report the experimental results in terms of accuracy (AUROC / Micro-F1), time (training and testing time per epoch) for link prediction and node classification respectively, with the best result in bold and the second-best underlined. With effective information propagation between supervertices, GripNet achieves the best AUROC/Micro-F1 scores on all seven datasets.

4.3.1. Link prediction

For the PoSE-series supergraphs, we have two GripNet implementations, **GripNet-l** and **GripNet-r**, to emphasize capturing information on the leaf and root supervertex by using less layer on the root and leaf supervertex respectively. Typically, the internal aggregation layer (see Eq. (11)) on each supervertex contains two RGCN sublayers. Here we use a one-sublayer internal aggregation layer on the root supervertex for GripNet-l and on the leaf supervertex for GripNet-r. The hyperparameter settings for GripNet-r are the same as GripNet-0/1/2 in Table 2, while GripNet-l has slightly different settings: it does not have $\mathbf{W}_2^{ia}$ for the root (gene) supervertex but set that for the leaf (drug) supervertex to $\mathbb{R}^{16\times 16}$. From the link prediction results in Table 3 and Figs. 4A and B, GripNet has the best prediction accuracy under a maximum GPU memory of 30GB and varying embedding dimensions, and baseline models are more sensitive to embedding dimension than GripNet. From Fig. 4A, the AUROC of GripNet does not increase with increasing embedding dimension, while at the lowest embedding dimension 8, GripNet greatly outperformed the RL-based methods (TransE,

Table 3
Multi-relational link prediction results in **AUROC**. The best results are in bold and the second best ones are underlined. Analogous trends hold for AUPRC and AP@50. **Dim**: the embedding dimension for each model such that the training peak GPU memory usage is around 30GB. **TpE**: computational time per epoch (including training and testing score computation).

	PoSE-0			PoSE-1			PoSE-2		
Model	AUROC	Dim	TpE(s)	AUROC	Dim	TpE(s)	AUROC	Dim	TpE(s)
TransE	0.718 ± 0.006	134	30.0	0.826 ± 0.007	170	21.5	0.751 ± 0.006	106	30.8
RotatE	0.891 ± 0.003	32	29.7	0.857 ± 0.006	38	20.7	0.851 ± 0.004	24	30.6
ComplEx	0.595 ± 0.004	34	<u>29.2</u>	0.844 ± 0.003	48	**20.3**	0.560 ± 0.004	26	<u>30.4</u>
DistMult	0.843 ± 0.008	84	**29.0**	0.859 ± 0.008	108	<u>20.4</u>	0.877 ± 0.009	66	**29.7**
RGCN	<u>0.905 ± 0.006</u>	76	47.5	0.909 ± 0.005	94	37.1	0.850 ± 0.003	58	60.3
GripNet-l	$\mathbf{0.920 \pm 0.003}$	32	39.9	$\mathbf{0.914 \pm 0.006}$	120	24.2	<u>0.918 ± 0.004</u>	32	40.3
GripNet-r	$\mathbf{0.920 \pm 0.003}$	16	41.1	<u>0.912 ± 0.003</u>	58	25.0	$\mathbf{0.920 \pm 0.050}$	16	41.4

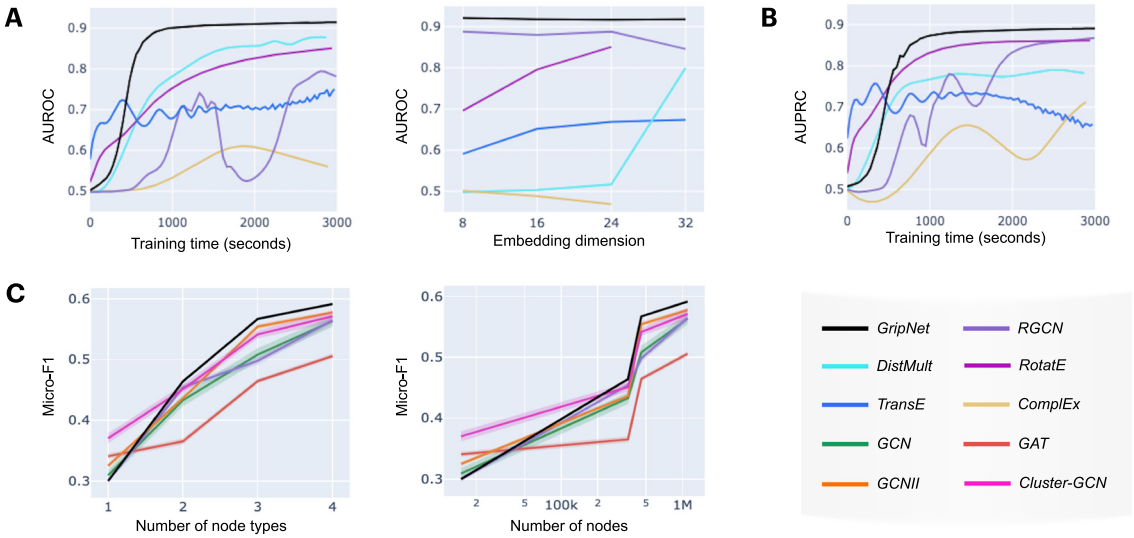

Fig. 4. **A**: AUROC over training time and at different embedding dimensions for different models on PoSE-2. **B**: AUROC over training time for different models on PoSE-0. **C**: Micro-F1 on the four Freebase datasets plotted with respect to the number of node types (graph heterogeneity) and the number of nodes, with standard deviations as shades (only visible for GCN).

RotatE, ComplEx, and DistMult). Although RL-based methods have a shorter per-epoch running time than GripNet (from Table 3), GripNet converges much faster (from Figs. 4A and B) and fewer epochs are needed in practice.

4.3.2. Node classification

In node classification (Table 4), the F1 scores for different models have small differences on Aminer but more variance for the Freebase series. This indicates that the information between supervertices of Aminer is less complementary and GripNet leads to more improvement when the dataset heterogeneity increases. The **GripNet-cat** and **GripNet-add** implementations correspond to two operations combining the internal and external features (see Eq. (10)). We will discuss them more in Section 4.6 on ablation studies.

From Fig. 4C, we can see that GripNet has the biggest improvement in book classification accuracy by 99% on Freebase-d over on Freebase-a, while GCN, GCNII, RGCN, GAT, and Cluster-GCN have a smaller improvement of 86%, 78%, 69%, 65%, and 54%, respectively. In addition, compared with other MPass-based methods (GCN, RGCN, and GAT), GripNet is much faster (from Table 4) and consumes much less GPU memory because of its segregated architecture that reduces unnecessary expensive graph convolution operations.

4.4. Data integration studies

Heterogeneous graphs are useful for integrating multiple datasets. Thus, GripNet can be used for data integration, where each supervertex represents a dataset and superedges define the relationships between datasets as information propagation paths. For example, Freebase-b has two supervertices (book and business). To add/integrate an "organization dataset" into Freebase-b, we add a supervertex of organization first, and then create a superedge from the organization supervertex to the book supervertex because of the close association between organization and book (based on existing knowledge).

In Fig. 4C, Freebase-a contains only one node type (book) while Freebase-d contains four node types because it has integrated three additional types of data (business, organization, and location). Figure 4C also shows that the improvement by GripNet increases as the heterogeneity of the graph increases. We visualize book embedding learned by GripNet in the four Freebase-series datasets in Fig. 5 to show that as we integrate data of different types, the boundaries between book categories become clearer. For the polypharmacy side effect prediction task, we get decent prediction accuracy with 0.743 in AUROC when using the PP & ChG dataset, while the AUROC trained with the ChChSe dataset is 0.908. The integration of these datasets leads to an AUROC with 0.922, which improves the performance by 24.1% and 1.5% respectively.

4.5. Studies on supergraph construction

In supergraph construction, the number of supervertices $|\mathcal{C}|$ and the directions of superedges are manually determined. Knowing their impact on both the model performance and the memory and time costs are critical to designing GripNet models.

Table 4
Node classification results in **Micro-F1**. The best results are in **bold**, the second best ones are underlined. Analogous trends hold for Macro-F1. Note that the time cost of Cluster-GCN is not comparable against other models because it is implemented in TensorFlow while other models are implemented in PyTorch.

	Aminer		Freebase-b		Freebase-c		Freebase-d	
Model	Micro-F1	TpE(s)	Micro-F1	TpE(s)	Micro-F1	TpE(s)	Micro-F1	TpE(s)
Cluster-GCN	0.915 ± 0.008	-	0.451 ± 0.008	-	0.541 ± 0.007	-	0.571 ± 0.009	-
GAT	0.899 ± 0.003	0.36	0.454 ± 0.005	0.43	0.498 ± 0.004	0.50	0.564 ± 0.004	0.94
GCN	0.907 ± 0.010	0.26	0.433 ± 0.010	0.17	0.508 ± 0.011	0.22	0.563 ± 0.009	0.51
GCNII	0.827 ± 0.004	2.15	0.437 ± 0.002	2.07	0.554 ± 0.004	2.58	0.577 ± 0.004	3.92
RGCN	0.882 ± 0.005	2.62	0.365 ± 0.005	1.09	0.464 ± 0.006	1.31	0.506 ± 0.006	2.67
GripNet-cat	0.920 ± 0.002	**0.25**	0.464 ± 0.002	**0.08**	$\mathbf{0.567 \pm 0.002}$	**0.16**	0.591 ± 0.002	**0.36**
GripNet-add	$\mathbf{0.921 \pm 0.002}$	**0.25**	$\mathbf{0.476 \pm 0.001}$	0.13	0.556 ± 0.002	0.17	$\mathbf{0.594 \pm 0.002}$	**0.36**

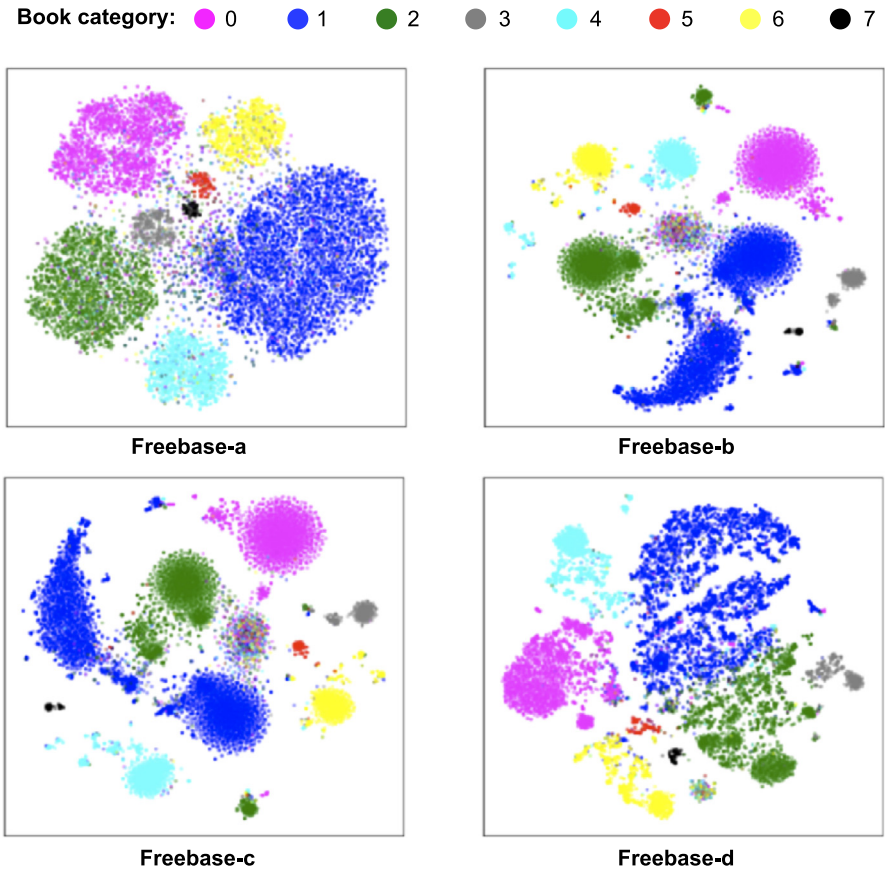

Fig. 5. Visualization of GripNet embedding for eight book categories in Freebase-series datasets using t-SNE [40] to study the data integration performance. Colors represent different book categories.

We study supergraph construction on Freebase-d dataset, which has four types of nodes. According to the criteria of supergraph construction in Section 3.2, there is one supergraph with three supervertices and two supergraphs with four supervertices, and we build GripNet models, called GripNet-cat$_3$, GripNet-cat$_{4a}$, and GripNet-cat$_{4b}$, on these supergraphs respectively (see Table 5). Please note that GripNet-cat$_3$ is the same as GripNet-cat in Table 3. There is no valid supergraph with two supervertices in this case, because any such supergraph either violates the first requirement or the second requirement of supervertices (Section 3.2). Because GripNet is equivalent to RGCN when $|\mathcal{C}| = 1$, we also compare them with RGCN. We use the same experimental settings as GripNet-cat, and set the dimensions of all the hidden layers and final embeddings for each node type to be the same.

Firstly, the results in Table 5 show that a large $|\mathcal{C}|$ gives lower memory and time costs, which is consistent with our expectation that GripNet will reduce the memory and time costs by reducing the amount of information propagation operations as the number of supervertices increases. Generally, when $|\mathcal{C}| = |\mathcal{T}|$, we expect the memory and time costs to be the lowest. For Freebase-d, $|\mathcal{T}| = 4$. As expected, the GripNet models with $|\mathcal{C}| = 4$ show the lowest memory and time costs in Table 5.

Secondly, comparing the results of two GripNet models with four supervertices, we can see that the direction of the superedge between non-task supervertices does not significantly affect the model performance and the memory and time costs. One possible reason why GripNet-cat$_{4a}$ is better than GripNet-cat$_{4b}$ is that there are more location-book edges ($\sim 58k$) than organization-

Table 5
Results of supergraph construction studies on Freebase-d dataset in **Micro-F1**. Analogous trends hold for Macro-F1. $|\mathcal{C}|$: #supervertices, **GMem**: averaged peak GPU memory usage (in GB), **TpE**: the same as in Table 3. Supergraphs use the same legends as in Fig. 2. The best results are in bold.

$\|\mathcal{C}\|$	1	3	4	
Supergraph				
Model	RGCN	GripNet-cat$_3$	GripNet-cat$_{4a}$	GripNet-cat$_{4b}$
Micro-F1	0.506 ± 0.006	$\mathbf{0.591 \pm 0.002}$	0.582 ± 0.002	0.576 ± 0.002
GMem	23.46	13.82	**11.64**	11.65
TpE(s)	2.67	0.34	**0.23**	**0.23**

Table 6
Averaged peak GPU memory usages (in GB) for feature integration operation comparison on node classification. The better results are in bold.

Model	Aminer	Freebase-b	Freebase-c	Freebase-d
GripNet-cat	5.89	**2.29**	**4.24**	**13.82**
GripNet-add	**5.74**	4.63	4.87	15.74

book edges ($\sim 22k$) and information tends to propagate better from less-connected supervertex to more-connected supervertex (than vice versa).

Thirdly, choosing the number of supervertices as three gives the best prediction performance but slightly higher computing cost than the two GripNet models with four supervertices. This also indicates that considering the semantic coherence between location and organization nodes and assigning them to the same supervertex leads to better prediction results. In addition, learning on valid supergraphs with GripNet is better than learning with RGCN.

4.6. Ablation studies

We perform ablation studies to understand the effects of different design choices to our results.

Feature Integration Operation. GripNet provides two operations to obtain the total features of node in the internal feature layer within a supervertex: concatenation and sum, and we set the concatenation to be the default setting. As shown in the last two rows of Table 4 and in Table 6, these two operations lead to similar prediction accuracy and have similar memory and time costs except on Freebase-b. The reason why the memory and time costs for GripNet-add are higher than GripNet-cat on Freebase-b is that in GripNet-cat, the external aggregation feature layer has an output dimension of 64 and the internal feature layer has an output dimension of 128 but in GripNet-add, these two layers need to have the same output dimension 128. Therefore, the difference in peak GPU memory usages suggests that the concatenation operation is a more flexible choice that allows latent features to have different dimensions for more efficient information propagation.

Categorized Negative Sampling (CNS). GripNet only needs CNS when modeling relations within the drug supervertex. When we replaced it with the popular negative sampling used in comparative methods for link prediction (RGCN, DistMult, and DECAGON), the accuracy on the PoSE-series datasets was decreased by no more than 1%, but the convergence speed slowed down by 9.2%.

5. Conclusion

This paper proposed a new supergraph data structure and a new Graph Information Propagation (GripNet) framework for learning node representations on heterogeneous graphs. We evaluated GripNet on seven large-scale datasets in link prediction, node classification, and data integration. GripNet achieved the best overall performance on these datasets.

The idea of supergraph gives GripNet the ability to efficiently learn embeddings on highly heterogeneous graphs and allows GripNet to be deep (4–8 graph convolutional layers in our experiments). Here, we demonstrated GripNet using semantically-coherent segregation as an intuitive approach for assigning different types of nodes to different supervertices; however, this is not the only way. Thus, future work is needed 1) to test GripNet performance for graphs with higher heterogeneity, 2) to explore different node type segregation strategies, and 3) to discover the trade-off between the number of node types and supervertices. In addition, while the tasks covered in this paper only occur in one supervertex, GripNet can be adapted for tasks spanning multiple supervertices in the future.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work was partially supported by The University of Sheffield, Queen's University (at Kingston, Canada), and Natural Sciences and Engineering Research Council of Canada (NSERC) grant RGPIN-2020-06325. This research was enabled in part by support provided by Compute Canada (www.computecanada.ca).

References

[1] B. Yang, W.-t. Yih, X. He, J. Gao, L. Deng, Embedding entities and relations for learning and inference in knowledge bases, in: The International Conference on Learning Representations, 2015.
[2] S. Yun, M. Jeong, R. Kim, J. Kang, H.J. Kim, Graph transformer networks, in: Advances in Neural Information Processing Systems, 2019, pp. 11960–11970.
[3] S. Xue, J. Lu, G. Zhang, Cross-domain network representations, Pattern Recognit. 94 (2019) 135–148.
[4] Y. Wu, D. Lian, S. Jin, E. Chen, Graph convolutional networks on user mobility heterogeneous graphs for social relationship inference, in: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, 2019.
[5] J.-D. Zhang, C.-Y. Chow, GeoSoCa: exploiting geographical, social and categorical correlations for point-of-interest recommendations, in: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2015, pp. 443–452.
[6] M. Zitnik, M. Agrawal, J. Leskovec, Modeling polypharmacy side effects with graph convolutional networks, Bioinformatics 34 (13) (2018) i457–i466.
[7] J. Gilmer, S.S. Schoenholz, P.F. Riley, O. Vinyals, G.E. Dahl, Neural message passing for quantum chemistry, in: Proceedings of the 34th International Conference on Machine Learning, vol. 70, JMLR. org, 2017, pp. 1263–1272.
[8] L. Zhang, H. Song, N. Aletras, H. Lu, Node-feature convolution for graph convolutional networks, Pattern Recognit. 128 (2022) 108661.
[9] F.-Y. Sun, M. Qu, J. Hoffmann, C.-W. Huang, J. Tang, vGraph: a generative model for joint community detection and node representation learning, in: Advances in Neural Information Processing Systems, 2019, pp. 512–522.
[10] M. Schlichtkrull, T.N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, M. Welling, Modeling relational data with graph convolutional networks, in: European Semantic Web Conference, Springer, 2018, pp. 593–607.
[11] S. Feng, C. Xu, Y. Zuo, G. Chen, F. Lin, J. XiaHou, Relation-aware dynamic attributed graph attention network for stocks recommendation, Pattern Recognit. 121 (2022) 108119.
[12] W.L. Hamilton, R. Ying, J. Leskovec, Representation learning on graphs: methods and applications, Bull. Inst. Electr.Electron. Eng. Comput. Soc. Tech. Committee Data Eng. (2017).
[13] Y. Sun, J. Han, X. Yan, P.S. Yu, T. Wu, PathSim: meta path-based top-k similarity search in heterogeneous information networks, Proc. VLDB Endowment 4 (11) (2011) 992–1003.
[14] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, P.S. Yu, Heterogeneous graph attention network, in: The World Wide Web Conference, 2019, pp. 2022–2032.
[15] Y. Zhang, Y. Xiong, X. Kong, S. Li, J. Mi, Y. Zhu, Deep collective classification in heterogeneous information networks, in: Proceedings of the 2018 World Wide Web Conference, 2018, pp. 399–408.
[16] Y. Dong, N.V. Chawla, A. Swami, metapath2vec: scalable representation learning for heterogeneous networks, in: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining, 2017, pp. 135–144.
[17] B. Yu, J. Hu, Y. Xie, C. Zhang, Z. Tang, Rich heterogeneous information preserving network representation learning, Pattern Recognit. 108 (2020) 107564.
[18] C. Yang, Y. Xiao, Y. Zhang, Y. Sun, J. Han, Heterogeneous network representation learning: aunified framework with survey and benchmark, IEEE Trans. Knowl. Data Eng. (2020), doi:10.1109/TKDE.2020.3045924. 1–1.
[19] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: International Conference on Learning Representations, 2017.
[20] A. Garcia-Duran, M. Niepert, KBLRN: end-to-end learning of knowledge base representations with latent, relational, and numerical features, in: The Conference on Uncertainty in Artificial Intelligence, 2018.
[21] B. Malone, A. García-Durán, M. Niepert, Knowledge graph completion to predict polypharmacy side effects, in: International Conference on Data Integration in the Life Sciences, Springer, 2018, pp. 144–149.
[22] T.N. Kipf, M. Welling, Variational graph auto-encoders, NIPS Workshop on Bayesian Deep Learning, 2016.
[23] K. Thulasiraman, M. Swamy, Acyclic directed graphs, in: J. Wiley, Son (Eds.), Graphs: Theory and Algorithms, 1992, p. 118.

[24] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Advances in Neural Information Processing Systems, 2013, pp. 3111–3119.
[25] S.M. Marinka Zitnik, R. Sosic, J. Leskovec, BioSNAP Datasets: stanford biomedical network dataset collection, 2018, (http://snap.stanford.edu/biodata).
[26] D. Maglott, J. Ostell, K.D. Pruitt, T. Tatusova, Entrez gene: gene-centered information at NCBI, Nucleic Acids Res. 33 (suppl_1) (2005) D54–D58.
[27] S. Kim, P.A. Thiessen, E.E. Bolton, J. Chen, G. Fu, A. Gindulyte, L. Han, J. He, S. He, B.A. Shoemaker, et al., Pubchem substance and compound databases, Nucleic Acids Res. 44 (D1) (2016) D1202–D1213.
[28] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, Z. Su, ArnetMiner: extraction and mining of academic social networks, in: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2008, pp. 990–998.
[29] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, in: Advances in Neural Information Processing Systems, 2013, pp. 2787–2795.
[30] T. Trouillon, J. Welbl, S. Riedel, E. Gaussier, G. Bouchard, Complex embeddings for simple link prediction, in: International Conference on Machine Learning, 2016.
[31] Z. Sun, Z.-H. Deng, J.-Y. Nie, J. Tang, RotatE: knowledge graph embedding by relational rotation in complex space, in: International Conference on Machine Learning, 2019.
[32] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, J. Leskovec, Open graph benchmark: datasets for machine learning on graphs, Advances in Neural Information Processing Systems, 2020.
[33] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph attention networks, in: International Conference on Learning Representations, 2018.
[34] M. Chen, Z. Wei, Z. Huang, B. Ding, Y. Li, Simple and deep graph convolutional networks, in: International Conference on Machine Learning, 2020, pp. 1725–1735.
[35] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, C.-J. Hsieh, Cluster-GCN: an efficient algorithm for training deep and large graph convolutional networks, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 257–266.
[36] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the Thirteenth International Conference on Artificial intelligence and statistics, 2010, pp. 249–256.
[37] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, in: International Conference on Learning Representations, 2015.
[38] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic differentiation in PyTorch, NIPS Autodiff Workshop, 2017.
[39] M. Fey, J.E. Lenssen, Fast graph representation learning with pytorch geometric, ICLR Workshop on Representation Learning on Graphs and Manifolds, 2019.
[40] L. Van Der Maaten, Accelerating t-SNE using tree-based algorithms, J. Mach. Learn. Res. 15 (1) (2014) 3221–3245.

Hao Xu received the BEng degree from Central South University, China in 2016, and MSc degree with distinction from Department of Computer Science, University of Sheffield, UK in 2019. She is currently a Research Assistant at Queen's University, Canada. Her research interests include graph representation learning and explainable machine learning for biology.

Shengqi Sang received his BSc degree from Fudan University, China in 2017, and his MSc degree from University of Chicago, United State in 2019. Currently, he is a PhD student at Perimeter Institute for Theoretical Physics and University of Waterloo in Canada. His research interests are condensed matter physics and machine learning methods for physical science.

Peizhen Bai received the BSc degree from Sun Yat-sen University, China in 2016, and MSc degree with distinction from Department of Computer Science, University of Sheffield, UK in 2017, where he is now a PhD student. His research interests include graph machine learning, recommendation systems and drug discovery.

Ruike Li received the BEng degree from Central China Normal University in 2018. She is currently an MSc student at Department of Computer Science, University of Sheffield, UK. Her research interests include graph representation learning and its applications.

Laurence Yang is an Assistant Professor of Chemical Engineering and Queen's National Scholar in Systems Biology at Queen's University. He received his PhD degree in Chemical Engineering from the University of Toronto. His research interests include computational systems biology, machine learning, and applying them for human health, environmental sustainability, and industrial biotechnology.

Haiping Lu received the BEng and MEng degrees in electrical and electronics engineering from Nanyang Technological University, Singapore, in 2001 and 2004, respectively, and the PhD degree in electrical and computer engineering from University of Toronto, Canada, in 2008. Currently, he is a Senior Lecturer in Machine Learning at the Department of Computer Science, University of Sheffield, UK.