DUAL GRAPH REPRESENTATION LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph representation learning embeds nodes in large graphs as low-dimensional vectors and benefit to many downstream applications. Most embedding frameworks, however, are inherently transductive and unable to generalize to unseen nodes or learn representations across different graphs. Inductive approaches, such as GraphSAGE, neglect different contexts of nodes and cannot learn node embeddings dually. In this paper, we present a context-aware unsupervised dual encoding framework, **CADE**, to generate representation of nodes by combining real-time neighborhood structure with neighbor-attentioned representation, and preserving extra memory of known nodes. Experimently, we exhibit that our approach is effective by comparing to state-of-the-art methods.

1 INTRODUCTION

The study of real world graphs, such as social network analysis (Hamilton et al. (2017a)), molecule screening (Duvenaud et al. (2015)), knowledge base reasoning (Trivedi et al. (2017)), and biological protein-protein networks (Zitnik & Leskovec (2017)), evolves with the development of computing technologies. Learning vector representations of graphs is effective for a variety of prediction and graph analysis tasks (Grover & Leskovec (2016); Tang et al. (2015)). High-dimensional information about neighbors of nodes are represented by dense vectors, which can be fed to off-the-shelf approaches to tasks, such as node classification (Wang et al. (2017); Bhagat et al. (2011)), link prediction (Perozzi et al. (2014); Wei et al. (2017)), node clustering (Nie et al. (2017); Ding et al. (2001)), recommender systems (Ying et al. (2018a)) and visualization (Maaten & Hinton (2008)).

There are mainly two types of models for graph representation learning. Transductive approaches (Perozzi et al. (2014); Grover & Leskovec (2016); Tang et al. (2015)) are able to learn representations of existing nodes but unable to generalize to new nodes. However, in real-world evolving graphs such as social networks, new users will join and must be represented. Inductive approaches were proposed to address this issue. GraphSAGE (Hamilton et al. (2017b)), a hierarchical sampling and aggregating framework, successfully leverages feature information to generate embeddings of the new nodes. However, GraphSAGE has its own faults. Firstly, it samples all neighborhood nodes randomly and uniformly; secondly, it treats the output of encoder as the final representation of node.

Based on the hierarchical framework of GraphSAGE, GAT (Velickovic et al. (2017)) uses given class labels to guide attention over neighborhood so as to aggregate useful feature information. However, without knowledge of ground-truth class labels, it is difficult for unsupervised approaches to apply attention. To address this issue, we introduce a **dual encoding** framework for unsupervised inductive representation learning of graphs. Instead of learning self-attention over neighborhoods of nodes, we exploit the bi-attention between representations of two nodes that co-occur in a short random-walk (which we call a **positive pair**).

In Figure 1, we illustrate how nodes are embedded into low-dimensional vectors, where each node v has an optimal embeddings o_v. Yet the direct output of encoder z_v of GraphSAGE could be located anywhere. Specifically, given feature input from both sides of a positive pair (v, v_p), a neural network is trained to encode the pair into K different embeddings z_v^k and $z_{v_p}^k$ through different sampled neighborhoods or different encoding functions. Then, a bi-attention layer is applied to generate the most adjacent matches $z_{v|v_p}$ and $z_{v_p|v}$, which will be referred as dual-representations. By putting most attention on the pair of embeddings with smallest difference, dual representation of nodes with less deviation will be generated, which can be visualized as $z_{v|\cdot}$ in Figure 1.

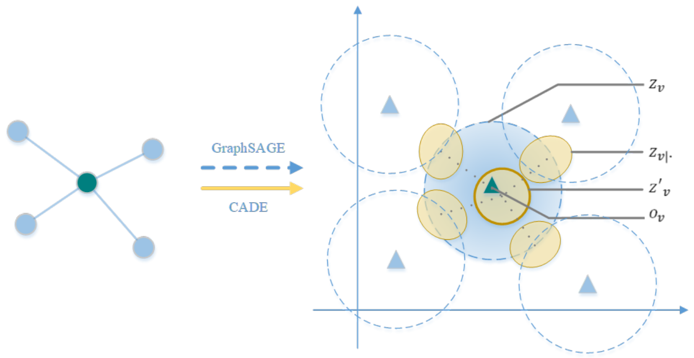


Figure 1: Visual comparison between representations learnt by current methods and dual encoding.

GraphSAGE naively assumes that unseen graph structure should be (easily) represented by known graphs data. We combine the ground truth structure and the learned dual-encoder to generate final representation. Unseen nodes can be represented based on their neighborhood structure. Current inductive approaches have no direct memory of the training nodes. We combine the idea of both transductive and inductive approaches via associating an additive global embedding bias to each node, which can be seen as a memorable global identification of each node in training sets.

Our contributions can be summarized as follows:

- we introduce a dual encoding framework to produce context-aware representation for nodes, and conduct experiments to demonstrate its efficiency and effectiveness;
- we apply bi-attention mechanism for graph representation dual learning, managing to learn dual representation of nodes more precisely;
- we combine the training of transductive global bias with inductive encoding process, as memory of nodes that are already used for training.

2 Related Work

Following (Cai et al. (2018), Kinderkhedia (2019) and Goyal & Ferrara (2018)), there are mainly two types of approaches:

2.1 Network embedding

For unsupervised embedding learning, DeepWalk (Perozzi et al. (2014)) and node2vec (Grover & Leskovec (2016)) are based on random-walks extending the Skip-Gram model; LINE (Tang et al. (2015))seeks to preserve first- and second-order proximity and trains the embedding via negative sampling; SDNE (Wang et al. (2016)) jointly uses unsupervised components to preserve second-order proximity and expolit first-order proximity in its supervised components; TRIDNR (Pan et al. (2016)), CENE(Sun et al. (2016)), TADW (Yang et al. (2015)),GraphSAGE (Hamilton et al. (2017b)) utilize node attributes and potentially node labels. Convolutional neural networks are also applied to graph-structured data. For instance, GCN (Kipf & Welling (2017)) proposed an simplified graph convolutional network. These graph convolutional network based approaches are (semi-)supervised. Recently, inductive graph embedding learning (Hamilton et al. (2017b) Velickovic et al. (2017) Bojchevski & Günnemann (2017) Derr et al. (2018) Gao et al. (2018), Li et al. (2018), Wang et al. (2018) and Ying et al. (2018b)) produce impressive performance across several large-scale benchmarks.

2.2 Attention

Attention mechanism in neural processes have been extensively studied in neuroscience and computational neuroscience (Itti et al. (1998); Desimone & Duncan (1995)) and frequently applied in deep learning for speech recognition (Chorowski et al. (2015)), translation (Luong et al. (2015)), question answering (Seo et al. (2016)) and visual identification of objects (Xu et al. (2015)). Inspired by (Seo et al. (2016) and Abu-El-Haija et al. (2018)), we construct a bi-attention layer upon aggregators to capture useful parts of the neighborhood.

3 Model

Let $\mathcal{G} = \{V, E, \boldsymbol{X}\}$ be an undirected graph, where a set of nodes V are connected by a set of edges E, and $\boldsymbol{X} \in \mathbb{R}^{|V| \times f}$ is the attribute matrix of nodes. A global embedding bias matrix is denoted by $\boldsymbol{B} \in R^{|V| \times d}$, where a row of $\boldsymbol{B}$ represents the d-dimensional global embedding bias of a node. The hierarchical layer number, the embedding output of the l-th layer and the final output embedding are denoted by L, $\boldsymbol{h}^l$ and $\boldsymbol{z}$, respectively.

3.1 Context-aware inductive embedding encoding

The embedding generation process is described in Algorithm 1. Assume that the dual encoder is trained and parameters are fixed.

Algorithm 1 Context-Aware Dual-Encoding (**CADE**)

```
input: the whole graph $\mathcal{G} = (V, E)$; the feature matrix $\boldsymbol{X}$; the trained $\mathrm{DualENC}$
output: learned embeddings $\boldsymbol{z}$;
 1: Run random walks on $\mathcal{G}$ to gain a set of positive pair $\mathbb{P}$;
 2: $\mathbb{Z}_v \leftarrow \emptyset, \forall v \in V$
 3: for $(v, v_p) \in \mathbb{P}$ do
 4:     $z_v, z_{v_p} = \mathrm{DualENC(v, v_p, \mathcal{G}, X)}$;
 5:     $\mathbb{Z}_v \leftarrow \mathbb{Z}_v \cup z_v$
 6:     $\mathbb{Z}_{v_p} \leftarrow \mathbb{Z}_{v_p} \cup z_{v_p}$
 7: end for
 8: for $v \in V$ do
 9:     $\boldsymbol{z}_v = \mathrm{Mean(\mathbb{Z}_v)}$;
10: end for
```

After training, positive pairs are collected by random walks on the whole dataset. The features of each positive pair are passed through a dual-encoder. Embeddings of nodes are generated so that the components of a pair are related and adjacent to each other.

3.2 Dual-encoder with multi-sampling

In this subsection, we explain the dual encoder.

Algorithm 2 DualENC

```
input: Training graph $\mathcal{G}(V, E)$; node attributes $\boldsymbol{X}$; global embedding bias matrix $\boldsymbol{B}$; sampling times $K$; positive node pair $(v, v_p)$;
output: adjacent embeddings $z_v$ and $z_{v_p}$;
1: For node $v$ and $v_p$, generate $K$ embeddings, $\boldsymbol{h}_v$, $\boldsymbol{h}_{v_p}$, using a base encoder SAGB
2: for $i, j \in \{1, ..., K\}$ do
3:     $S_{i,j} \leftarrow \alpha(\boldsymbol{h}_{vi}, \boldsymbol{h}_{v_p j})$
4: end for
5: softmax on flattened similarity matrix $S$: $S_{i,j} \leftarrow \frac{e^{S_{i,j}}}{\sum_{0,0}^{K,K} e^{S_{i,j}}}$
6: calculate attention $a_v$ and $a_{v_p}$:
   $\boldsymbol{a}_{vi} \leftarrow \sum_{j=1}^{K} S_{i,j}, \boldsymbol{a}_{v_p j} \leftarrow \sum_{i=1}^{K} S_{i,j}$
7: $\boldsymbol{z}_v \leftarrow \sum_{t=1}^{K} \boldsymbol{a}_{vk} \boldsymbol{h}_{vk}^L$
8: $\boldsymbol{z}_{v_p} \leftarrow \sum_{t=1}^{K} \boldsymbol{a}_{v_p k} \boldsymbol{h}_{v_p k}^L$
```

In the hierarchical sampling and aggregating framework (Hamilton et al. (2017b)), it is challenging and vital to select relevant neighbor with the layer goes deeper and deeper. For example, given word ”mouse” and its positive node ”PC”, it is better to sample ”keyboard”, instead of ”cat”, as a neighbor node. However, to sample the satisfying node according to heuristic rules layer by layer is very time consuming, and it is difficult to learn attention over neighborhood for unsupervised embedding learning.

As a matter of fact, these neighbor nodes are considered to be useful because they are more welcome to be sampled as input so as to produce more relevant output of the dual-encoder. Therefore, instead of physically sampling these neighbor nodes, in Step 2 to Step 6 in Algorithm 2, we directly apply a **bi-attention layer** on the two sets of embedding outputs with different sampled neighborhood feature as input, so as to locate the most relevant representation match, as a more efficient approach to exploring the most useful neighborhood.

We use the hierarchical sampling and aggregating framework as a base encoder in our experiments, but it can also be designed in many other ways.

Given node v, and node v_p as a positive node pair, after sampling and aggregating for K times, we have K different representations h_{vk}/h_{v_pk} corresponding to K different sampled neighborhoods, their similarity matrix can be calculated by

$$S_{i,j} = \alpha(\boldsymbol{h}_{vi}, \boldsymbol{h}_{v_pj}), \quad i, j = 1, ..., K \tag{1}$$

where α represents an dot-product operation.

our goal is to find the closest neighborhood match between v and v_p within $K \times K$ possibilities, so we apply softmax on the flattened similarity matrix, and sum up by row(column). In this way, we manage to gain different attention over K neighborhoods of $v(v_p)$ with respect to $v_p(v)$,

$$\boldsymbol{S}_{ij} \leftarrow \frac{\exp(\boldsymbol{S}_{ij})}{\sum_{0,0}^{K,K} \exp(\boldsymbol{S}_{ij})} \tag{2}$$

$$\boldsymbol{a}_{vi} = \sum_{j=1}^{K} \boldsymbol{S}_{ij} \tag{3}$$

$$\boldsymbol{a}_{v_pj} = \sum_{i=1}^{K} \boldsymbol{S}_{ij} \tag{4}$$

and sum up the lth layer representations with attention as the dual-encoder outputs,

$$\boldsymbol{z}_v = \sum_{k=1}^{K} \boldsymbol{a}_{vk} \boldsymbol{h}_{vk} \tag{5}$$

$$\boldsymbol{z}_{v_p} = \sum_{k=1}^{K} \boldsymbol{a}_{v_pk} \boldsymbol{h}_{v_pk} \tag{6}$$

To train our encoder before using it to generate final representations of nodes, we apply a typical pair reconstruction loss function with negative sampling Hamilton et al. (2017b):

$$J_{\mathcal{G}}(z_v) = -\log(\sigma(\boldsymbol{z}_v^T \boldsymbol{z}_{v_p})) - Q \cdot \mathbb{E}_{v_n \sim P_n(v)} \log(\sigma(-\boldsymbol{z}_v^T \boldsymbol{z}_{v_n})) \tag{7}$$

where node v_p co-occurs with v on fixed-length random walk (Perozzi et al. (2014)), σ is the sigmoid function, P_n is a negative sampling distribution, Q defines the number of negative samples. Note that z_v and z_{v_p} are dual representation to each other while z_{v_n} represents the direct encoder output of negative sample v_n.

3.3 Dual-encoder with multi-aggregating

Besides learning dual representation with multiple sampling, we introduce another version of our dual encoder with multiple aggregator function.The intuition is that through different perspective, a node can be represented differently corresponding to different kinds of positive nodes.

In Step 1 in Algorithm 2, for a node v, we sample neighborhood once and aggregate feature with K sets of parameters, gaining K different representations h_{vk} corresponding to K different character of v. Given a positive node pair, v and v_p, their dual representation are calculated by applying bi-attention as we described in the last section. There is but one difference that we use a weigh vector $\boldsymbol{A} \in R^{2d}$ as parameter instead of dot-product, to calculate the $K \times K$ attention matrix between node v and node v_p,

$$S_{ij} \leftarrow \frac{\exp(\boldsymbol{A}^{\mathsf{T}}[\boldsymbol{h}_{vi}||\boldsymbol{h}_{v_p j}])}{\sum_{0,0}^{K,K} \exp(\boldsymbol{A}^{\mathsf{T}}[\boldsymbol{h}_{vi}||\boldsymbol{h}_{v_p j}])} \tag{8}$$

where $\cdot^T$ represents transposition and $||$ is the concatenation operation. The rest of calculation of dual representation is the same as section 3.2.

Another difference is during training. With K sets of parameter for aggregating, negative sample v_n is now also represented by K different embeddings. As shown in Figure 2, we set $K = 5$ and use different shape to represent the embeddings of the positive node pair and the negative sampled nodes.

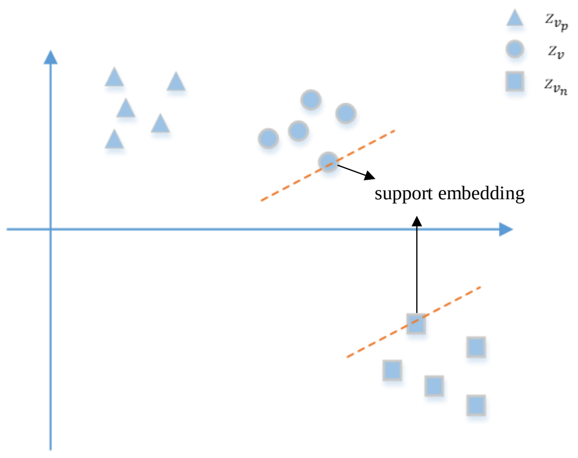


Figure 2: training support embeddings of node v and its negative sample node v_n.

As we can see in Figure 2, to make sure that any embeddings of node v_n as far away from any of node v as possible, it is equal to maximizing the distance between their **support embeddings**, which is the closest pair of embeddings of v and v_n. The support embedding can be calculated by the learned dual encoder. In conclusion, our loss function can be modified as follows,

$$J_{\mathcal{G}}(z_v) = -\log(\sigma(\boldsymbol{z}_v^T \boldsymbol{z}_{v_p})) - Q \cdot \mathbb{E}_{v_n \sim P_n(v)} \log(\sigma(-\boldsymbol{z}_v^{'\mathsf{T}} \boldsymbol{z}_{v_n})) \tag{9}$$

$$\boldsymbol{z}_v, \boldsymbol{z}_{v_p} = \text{DualENC}(v, v_p, \boldsymbol{A}) \tag{10}$$

$$\boldsymbol{z}'_v, \boldsymbol{z}_{v_n} = \text{DualENC}(v, v_p, \boldsymbol{A}^*) \tag{11}$$

where $\boldsymbol{A}^*$ representing that we stop the back-propagation through $\boldsymbol{A}$ in dual encoding for negative sample node, since $\boldsymbol{A}$ are supposed to learn bi-attention between the positive node pair and be reused only to capture the support embedding of v and its negative sample nodes.

3.4 Memorable global bias in hierarchical encoding

In this section, we first explain the base encoder used in our proposed dual encoding framework, and then we introduce how we apply memorable global bias within this framework.

The general intuition of GraphSAGE is that at each iteration, nodes aggregate information from their local neighbors, and as this process iterates, nodes incrementally gather more and more information from further reaches of the graph. For generating embedding for one specific node u, we describe the process below. First, we construct a neighborhood tree with node u as the root, $\mathbb{N}_u$, by iteratively sampling immediate neighborhood of nodes of the last layer as children. Nodes at the lth layer are represented by symbol $\mathbb{N}_u^l$, $\mathbb{N}_u^0 = \{u\}$. Then, at each iteration, each node i aggregates the representations of its children j, $\{h_j^{l-1}\}$, and of itself, h_i^{l-1}, into a single vector h_i^l, as representation of the next layer. After L iterations, we gain the Lth layer representation of v, as the final output.

While this framework generates good representation for nodes, it cannot preserve sufficient embedding informations for known nodes. More specifically, for nodes that are known but trained less than average, the learned model would have treated them like nodes unmet before. Therefore, we intuitively apply distinctive and trainable global bias to each node, as follows:

$$\boldsymbol{h}_{\mathcal{S}(i)}^{l-1} \leftarrow \text{AGGREGATE}_l(\{\boldsymbol{h}_j^{l-1}, \forall j \in \mathcal{S}(i)\}) \tag{12}$$

$$\boldsymbol{h}_i^l \leftarrow \sigma(\boldsymbol{W}^l \cdot [\boldsymbol{h}_i^{l-1} || \boldsymbol{h}_{\mathcal{S}(i)}^{l-1}]) \tag{13}$$

$$\boldsymbol{h}_i^l \leftarrow \boldsymbol{h}_i^l + \boldsymbol{b}_i, l < L \tag{14}$$

$$\boldsymbol{b}_i \leftarrow one\_hot(i)^\intercal \boldsymbol{B} \tag{15}$$

$$\tag{16}$$

where $\boldsymbol{B} \in R^{|V| \times d}$ is the trainable global bias matrix, $\mathcal{S}(i)$ represents the sampled neighborhood and also the children nodes of node i in the neighborhood tree, AGGREGATE represents the neighborhood aggregator function, and $||$ is a operator of concatenating vectors.

On one hand, $\boldsymbol{B}$ can be reused to produce embeddings for the known nodes or the unknown connected with the known, as supplement to the neural network encoder. On another hand, the global bias vectors can partially offset the uncertainty of the encoding brought by the random sampling not only during the training but also the final generation. Lastly but not least, we use only one set of global bias for all nodes, which means for any node, its representations of hidden layers are all added by the same bias vector. As a result of that, we are able to update parameters of aggregator function in the lowest layer with the global updated bias of nodes, highly increasing the training efficiency.

It is important for us to apply no global bias to the last layer of output, which is also the candidate of the dual-encoder output of nodes before applied with attention. The reason is that applying extra bias onto the last layer would directly change the embedding distribution of known nodes, making it unequal to the embedding distribution of unseen nodes. In general, the implementation of the base encoder with global bias is shown in Algorithm 3. The * in Step 8 means the children of node i in the neighborhood tree $\mathbb{N}_u$.

Algorithm 3 SAGB:sampling and aggregating with global bias

```
input: node $u$; hierarchical depth $L$; weight matrices $\boldsymbol{W}^l$; non-linearity $\sigma$; differentiable neighbor
aggregator $\text{AGGREGATE}_l$; fixed-size uniform sampler $\mathcal{S} : v \to 2^V$
output: embedding $\boldsymbol{z}_u$;
 1: $\mathbb{N}_u^0 = \{u\}$;
 2: for $l = 1...L$ do
 3:     $\mathbb{N}_u^l \leftarrow \{\mathcal{S}(i),\ \forall i \in \mathbb{N}_u^{l-1}\}$;
 4: end for
 5: for l=1...L do
 6:     for $i \in \mathbb{N}_u^0 \bigcup \mathbb{N}_u^1 \bigcup ... \bigcup \mathbb{N}_u^{L-l}$ do
 7:         $\boldsymbol{h}_{\mathcal{S}^*(i)}^{l-1} \leftarrow \text{AGGREGATE}_l(\{\boldsymbol{h}_j^{l-1}, \forall j \in \mathcal{S}^*(i)\})$
 8:         $\boldsymbol{h}_i^l \leftarrow \sigma(\boldsymbol{W}^l \cdot [\boldsymbol{h}_i^{l-1} || \boldsymbol{h}_{\mathcal{S}^*(i)}^{l-1}])$
 9:         if $l < L$: $\boldsymbol{h}_i^l \leftarrow \boldsymbol{h}_i^l + \text{one\_hot}(i)^\intercal \boldsymbol{B}$
10:     end for
11: end for
12: return $\boldsymbol{z}_u \leftarrow \boldsymbol{h}_u^L$
```

4 Experiments

In this section, we compare CADE against two strong baselines in an inductive and unsupervised setting, on challenging benchmark tasks of node classification and link prediction. We also perform further studies of the proposed model in section 4.5.

4.1 Datasets

The following graph datasets are used in experiments and statistics are summarized in Table1:

- **Pubmed**: The PubMed Diabetes (Sen et al. (2008))[1] dataset is a citation dataset which consists of scientific publications from Pubemd database pertaining to diabetes classified into one of three classes. Each publication in the dataset is described by a TF/IDF (Salton & Yu (1973)) weighted word vector from a dictionary.
- **Blogcatalog**: BlogCatalog[2] is a social blog directory which manages bloggers and their blogs, where bloggers following each others forms the network dataset.
- **Reddit**: Reddit[3] is an internet forum where users can post or comment on any content. We use the exact dataset conducted by (Hamilton et al. (2017b)), where each link connects two posts if the same user comments on both of them.
- **PPI**: The protein-protein-interaction (PPI) networks dataset contains 24 graphs corresponding to different human tissues(Zitnik & Leskovec (2017)). We use the preprocessed data also provided by (Hamilton et al. (2017b)).

Table 1: Dataset Statistics

Dataset	Nodes	Edges	Classes	Features	Avg Degree
Pubmed	19717	44324	3	500	4.47
Blogcatalog	5196	171743	6	8189	66.11
Reddit	232,965	11,606,919	41	602	100.30
PPI	56944	818716	121[4]	50	28.76

4.2 Experimental settings

We compare CADE against the following approaches in a fully unsupervised and inductive setting:

- GraphSAGE: In our proposed model, CADE, the base encoder mainly originates from GraphSAGE, a hierarchical neighbor sampling and aggregating encoder for inductive learning. Three alternative aggregators are used in Graphsage and CADE: (1) Mean aggregator, which simply takes the elementwise mean of the vectors in $h^{k-1}_{u\in N(v)}$; (2) LSTM aggregator, which adapts LSTMs to encode a random permutation of a node's neighbors' h^{k-1}; (3) Maxpool aggregator, which apply an elementwise maxpooling operation to aggregate information across the neighbor nodes.
- Graph2Gauss (Bojchevski & Günnemann (2017)): Unlike GraphSAGE and my method, G2G only uses the attributes of nodes to learn their representations, with no need for link information. Here we compare against G2G to prove that certain trade-off between sampling granularity control and embedding effectiveness does exists in inductive learning scenario.

Beside the above two models, we also include experiment results of raw features as baselines. In comparison, we call the version of dual-encoder with multiple sampling as **CADE-MS**, while the version with multiple aggregator function as **CADE-MA**.

[1]Available at https://linqs.soe.ucsc.edu/data.

[2]http://www.blogcatalog.com/

[3]http://www.reddit.com/

[4]PPI is a multi-label dataset.

For CADE-MS, CADE-MA and GraphSAGE, we set the depth of hierarchical aggregating as $L = 2$, the neighbor sampling sizes as $s_1 = 20, s_2 = 10$, and the number of random-walks for each node as 100 and the walk length as 4. The sampling time in CADE-MS or the number of aggregator in CADE-MA is set as $K = 10$. And for all emedding learning models, the dimension of embeddings is set to 256, as for raw feature, we use all the dimensions. Our approach is impemented in Tensorflow (Abadi et al. (2016)) and trained with the Adam optimizer (Kingma & Ba (2014)) at an initial learning rate of 0.0001.

4.3 Inductive node classification

We evaluate the node classification performance of methods on the four datasets. On Reddit and PPI, we follow the same training/validation/testing split used in GraphSAGE. On Pubmed and Blogcatalog, we randomly selected 10%/20%/30% nodes for training while the rest remain unseen. We report the averaged results over 10 random split.

After spliting the graph dataset, the model is trained in an unsupervised manner, then the learnt model computes the embeddings for all nodes, a node classifier is trained with the embeddings of training nodes and finally the learnt classifier is evaluated with the learnt embeddings of the testing nodes, i.e the unseen nodes.

Table 2: Prediction results for Pubmed/Blogcatalog w.r.t different unseen ratio

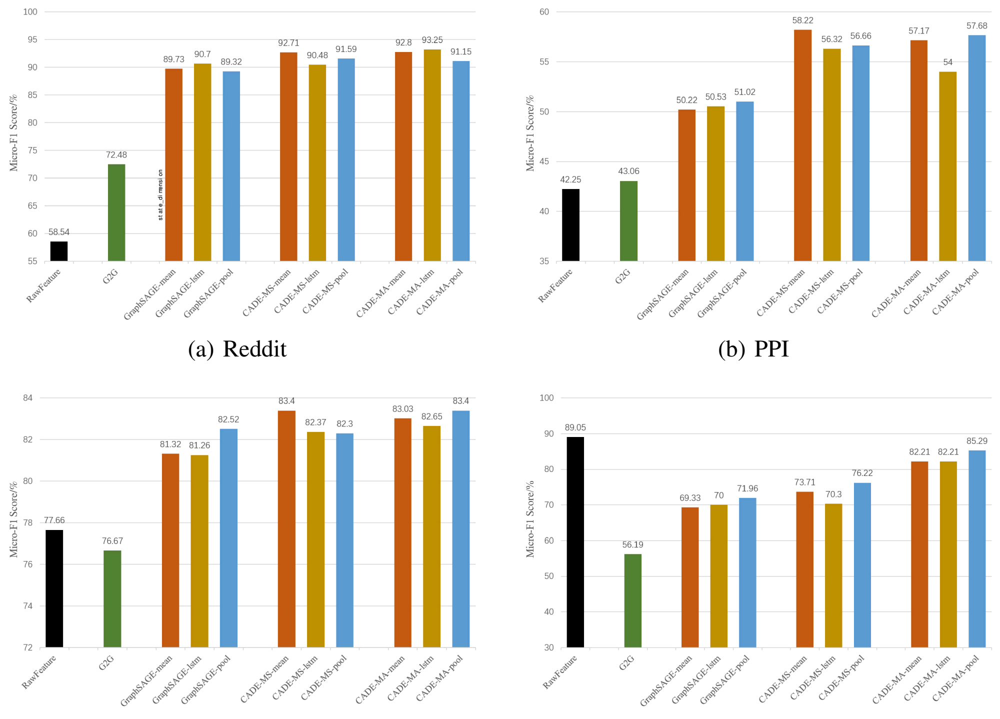
Methods	Pubmed			Blogcatalog		
unseen-ratio	10%	30%	50%	10%	30%	50%
RawFeats	79.22	77.66	77.74	**90.00**	**89.05**	**87.08**
G2G	80.70	76.67	76.31	62.35	56.19	48.46
GraphSAGE	82.05	81.32	79.68	71.48	69.33	64.92
CADE-MS	**84.25**	**83.40**	**81.74**	77.35	73.71	70.88
CADE-MA	**84.56**	**83.03**	**82.40**	84.33	82.21	79.04

Comparation on node classification performance on Pubmed and Blogcatalog dataset with respect to varying ratios of unseen nodes, are reported in Table 2. CADE-MS and CADE-MA outperform other approaches on Pubmed. On Blogcatalog dataset, however, RawFeats performs best mainly because that, in Blogcatalog dataset, node features are not only directly extracted from a set of user-defined tags, but also are of very high dimensionality (up to 8,189). Hence extra neighborhood information is not needed. As shown in Table 2, CADE-MA performs better than CADE-MS, and both outperform GraphSAGE and G2G. CADE-MA is capable of reducing high dimensionality while losing less information than CADE-MS, CADE-MA is more likely to search for the best aggregator function that can focus on those important features of nodes. As a result, the 256-dimensional embedding learnt by CADE-MA shows the cloest node classification performance to the 8k-dimensional raw features.

Comparasion among GraphSAGE, CADE and other aggregator functions is reported in Figure3. Each dataset contains 30% unseen nodes. In general, the model CADE shows significant advance to the other two state-of-art embedding learning models in node classification on four different challenging graph datasets.

4.4 Inductive link prediction

Link prediction task evaluates how much network structural information is preserved by embeddings. We preform the following steps: (1) mark some nodes as unseen from the training of embedding learning models. For Pubmed 20% nodes are marked as unseen; (2) randomly hide certain percentage of edges and equal-number of non-edges as testing edge set for link prediction, and make sure not to produced any dangling node; (3) the rest of edges are then used to form the input graph for embedding learning and with equal number of non-edges form the training edge set for link predictor; (4) after training and inductively generation of embeddings, the training edge set and their corresponding embeddings will help to train a link predictor; (5) finally evaluate the performance on the testing edges by the area under the ROC curve (AUC) and the average precision (AP) scores.



(a) Reddit (b) PPI

(c) Pubmed (d) Blogcatalog

Figure 3: Classification results (micro-averaged F1 scores) w.r.t different aggregators on four datasets

Table 3: Link prediction results for Pubmed/PPI w.r.t different percentage of hidden-edges

Dataset	Methods	90%:10%		80%:20%		60%:40%		40%:60%	
		AUC	AP	AUC	AP	AUC	AP	AUC	AP
Pubmed	RawFeats	57.61	54.72	58.51	56.19	54.47	52.82	52.41	50.77
	G2G	64.13	68.60	63.52	65.15	60.03	66.16	58.97	61.17
	GraphSAGE	85.49	82.79	87.64	83.35	81.07	77.47	79.34	74.92
	CADE-MS	**89.95**	**88.79**	**90.36**	**86.67**	**87.14**	**83.77**	**84.76**	**79.53**
	CADE-MA	**89.73**	**89.76**	**90.94**	**88.90**	**90.54**	**87.89**	**85.27**	**80.15**
PPI	RawFeats	57.46	56.99	57.34	56.86	57.35	56.75	56.83	56.36
	G2G	60.62	58.98	60.99	59.38	61.05	59.54	60.93	59.49
	GraphSAGE	82.74	81.20	82.21	80.66	82.11	80.51	82.07	80.70
	CADE-MS	**85.87**	**85.08**	**84.21**	**83.48**	**84.46**	**82.84**	**83.61**	**82.39**
	CADE-MA	**86.33**	**85.32**	**85.85**	**84.63**	**84.15**	**82.15**	81.98	79.54

Comparation on performance with respect to varying percentage of hidden edges are reported in Table3. CADE shows best link prediction performance on both datasets.

4.5 Model study

4.5.1 Sampling complexity in CADE-MS

Our proposed CADE-MS requires multiple neighborhood sampling, which increases the complexity of embedding learning. Yet by comparing CADE-MS against GraphSAGE with the same quantity of sampled neighborhood per node, the superiority of CADE model over existing models is still vast. In practice, we set the sampling layer $L = 2$ and the first-layer sampling size as 20. For the second layer, denote by s_2' the sampling size in GraphSAGE, and by s_2 and T the sampling size and sampling time in CADE-MS. We compare the two methods with $s_2' = s_2 * K$. A variant of CADE, called CADE-gb, applies only memorable global bias and no dual-encoding framework,

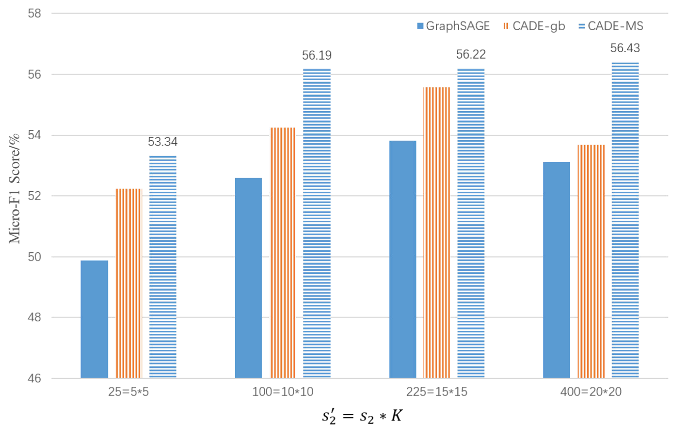


Figure 4: node classification performance (micro-f1 score) w.r.t varying sampling sizes

has the same sampling complexity as GraphSAGE. For the efficiency of experiment, we conduct experiments of node classification on a small subset of PPI, denoted by **subPPI**, which includes 3 training graphs plus one validation graph and one test graph. Results are reported in Figure 4. With much smaller sampling width, CADE-MS still outperforms the original framework significantly.

It implicates that searching for the best representation match through multiple sampling and bi-attention is efficient to filtering userful neighbor nodes without supervision from any node labels, and that the context-aware dual-encoding framework is capable of improving the inductive embedding learning ability without increasing sampling complexity. We aslo observe that CADE-gb, the variant simply adding the memorable global bias, continually shows advance in different sampling sizes.

4.5.2 THE NECESSARITY OF HIDDEN MEMORY

It is necessary not to apply global bias to the encoder output. We compare the original framework GraphSAGE, CADE with the variants of our method: applying global bias only to the last layer, and only to the former layers. As Table 4 shows, CADE-gl performs poorly, while CADE-gb demonstrates the effect of keeping memory of the hidden representation for each node.

Table 4: node classification performance (f1-micro score) with different way of applying global bias

method	F1-micro
GraphSAGE	50.22
CADE-gl	39.24
CADE-gb	54.13
CADE	58.22

5 CONCLUSION

In this paper, we proposed CADE, an unsupervised and inductive network embedding approach which is capable of preserving local connectiveness distinctively , as well as learning and memorizing global identities for seen nodes while generalizing to unseen nodes. We applied a bi-attention architeture upon hierarchical aggregating layers to capture the most relevant representations dually for any positive pair. We also present an effective way of combining inductive and transductive ideas by allowing trainable global embedding bias to be retrieved in hidden layers within the hierarchical aggregating framework. Experiments demonstrate the superiority of CADE over the state-of-art baselines on unsupervised and inductive tasks. In the future, we would explore several possibilities, such as expanding dual encoding from pair-wise to n-wise, or using dual encoding framework in supervised embedding learning, or combing dual encoding with G2G by learning distribution representations dually for positive pairs.

REFERENCES

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Gregory S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian J. Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Józefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Gordon Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul A. Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda B. Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *CoRR*, abs/1603.04467, 2016.

Sami Abu-El-Haija, Bryan Perozzi, Rami Al-Rfou, and Alexander A. Alemi. Watch your step: Learning node embeddings via graph attention. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pp. 9198–9208, 2018.

Smriti Bhagat, Graham Cormode, and S. Muthukrishnan. Node classification in social networks. In *Social Network Data Analytics*, pp. 115–148. Springer, 2011.

Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of attributed graphs: Unsupervised inductive learning via ranking. *CoRR*, abs/1707.03815, 2017.

HongYun Cai, Vincent W. Zheng, and Kevin Chen-Chuan Chang. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Trans. Knowl. Data Eng.*, 30(9):1616–1637, 2018.

Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-based models for speech recognition. In *Advances in neural information processing systems*, pp. 577–585, 2015.

Tyler Derr, Yao Ma, and Jiliang Tang. Signed graph convolutional networks. In *IEEE International Conference on Data Mining, ICDM 2018, Singapore, November 17-20, 2018*, pp. 929–934, 2018.

Robert Desimone and John Duncan. Neural mechanisms of selective visual attention. *Annual review of neuroscience*, 18(1):193–222, 1995.

Chris H. Q. Ding, Xiaofeng He, Hongyuan Zha, Ming Gu, and Horst D. Simon. A min-max cut algorithm for graph partitioning and data clustering. In *Proceedings of the 2001 IEEE International Conference on Data Mining, 29 November - 2 December 2001, San Jose, California, USA*, pp. 107–114, 2001.

David K. Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P. Adams. Convolutional networks on graphs for learning molecular fingerprints. In *NIPS*, pp. 2224–2232, 2015.

Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. Large-scale learnable graph convolutional networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pp. 1416–1424, 2018.

Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowl.-Based Syst.*, 151:78–94, 2018.

Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *KDD*, pp. 855–864, 2016.

William L. Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *IEEE Data Eng. Bull.*, 40(3):52–74, 2017a.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NIPS*, 2017b.

Laurent Itti, Christof Koch, and Ernst Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 20(11):1254–1259, 1998.

Mital Kinderkhedia. Learning representations of graph data - A survey. *CoRR*, abs/1906.02989, 2019.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 3538–3545, 2018.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

Feiping Nie, Wei Zhu, and Xuelong Li. Unsupervised large graph embedding. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pp. 2422–2428, 2017.

Shirui Pan, Jia Wu, Xingquan Zhu, Chengqi Zhang, and Yang Wang. Tri-party deep network representation. In *IJCAI*, pp. 1895–1901, 2016.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: online learning of social representations. In *KDD*, pp. 701–710, 2014.

Gerard Salton and Clement T. Yu. On the construction of effective vocabularies for information retrieval. In *Proceedings of the 1973 meeting on Programming languages and information retrieval, Gaithersburg, Maryland, USA, November 4-6, 1973*, pp. 48–60, 1973.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603, 2016.

Xiaofei Sun, Jiang Guo, Xiao Ding, and Ting Liu. A general framework for content-enhanced network representation learning. *CoRR*, abs/1610.02906, 2016.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. LINE: large-scale information network embedding. In *WWW*, pp. 1067–1077, 2015.

Rakshit Trivedi, Hanjun Dai, Yichen Wang, and Le Song. Know-evolve: Deep temporal reasoning for dynamic knowledge graphs. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pp. 3462–3471, 2017.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. *CoRR*, abs/1710.10903, 2017.

Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *KDD*, pp. 1225–1234, 2016.

Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. Graphgan: Graph representation learning with generative adversarial nets. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 2508–2515, 2018.

Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. Community preserving network embedding. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pp. 203–209, 2017.

Xiaokai Wei, Linchuan Xu, Bokai Cao, and Philip S. Yu. Cross view link prediction by learning noise-resilient representation consensus. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*, pp. 1611–1619, 2017.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, pp. 2048–2057, 2015.

Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y. Chang. Network representation learning with rich text information. In *IJCAI*, pp. 2111–2117, 2015.

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pp. 974–983, 2018a.

Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pp. 4805–4815, 2018b.

Marinka Zitnik and Jure Leskovec. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics*, 33(14):i190–i198, 2017.