Abstract

We introduce a self-supervised approach for learning node and graph level representations by contrasting structural views of graphs. We show that unlike visual representation learning, increasing the number of views to more than two or contrasting multi-scale encodings do not improve performance, and the best performance is achieved by contrasting encodings from first-order neighbors and a graph diffusion. We achieve new state-of-the-art results in self-supervised learning on 8 out of 8 node and graph classification benchmarks under the linear evaluation protocol. For example, on Cora (node) and Reddit-Binary (graph) classification benchmarks, we achieve 86.8% and 84.5% accuracy, which are 5.5% and 2.4% relative improvements over previous state-of-the-art. When compared to supervised baselines, our approach outperforms them in 4 out of 8 benchmarks.

1. Introduction

Graph neural networks (GNN) (Li et al., 2015; Gilmer et al., 2017; Kipf & Welling, 2017; Veličković et al., 2018; Xu et al., 2019b) reconcile the expressive power of graphs in modeling interactions with unparalleled capacity of deep models in learning representations. They process variable-size permutation-invariant graphs and learn low-dimensional representations through an iterative process of transferring, transforming, and aggregating the representations from topological neighbors. Each iteration expands the receptive field by one-hop and after $k$ iterations the nodes within $k$-hops influence one another (Khasahmadi et al., 2020). GNNs are applied to data with arbitrary topology such as point clouds (Hassani & Haley, 2019), meshes (Wang et al., 2018), robot designs (Wang et al., 2019), physical processes (Sanchez-Gonzalez et al., 2018), social networks (Kipf & Welling, 2017), molecules (Duvenaud et al., 2015), and knowledge graphs (Vivona & Hassani, 2019). For an overview see (Zhang et al., 2020; Wu et al., 2020).

GNNs mostly require task-dependent labels to learn rich representations. Nevertheless, annotating graphs is challenging compared to more common modalities such as video, image, text, and audio. This is partly because graphs are usually used to represent concepts in specialized domains, e.g., biology. Also, labeling graphs procedurally using domain knowledge is costly (Sun et al., 2020). To address this, unsupervised approaches such as reconstruction based methods (Kipf & Welling, 2016) and contrastive methods (Li et al., 2019) are coupled with GNNs to allow them learn representations without relying on supervisory data. The learned representations are then transferred to a priori unknown downstream tasks. Recent works on contrastive learning by maximizing mutual information (MI) between node and graph representations have achieved state-of-the-art results on both node classification (Veličković et al., 2019) and graph classification (Sun et al., 2020) tasks. Nonetheless, these methods require specialized encoders to learn graph or node level representations.

Recent advances in multi-view visual representation learning (Tian et al., 2019; Bachman et al., 2019; Chen et al., 2020), in which composition of data augmentations is used to generate multiple views of a same image for contrastive learning, has achieved state-of-the-art results on image classification benchmarks surpassing supervised baselines. However, it is not clear how to apply these techniques to data represented as graphs. To address this, we introduce a self-supervised approach to train graph encoders by maximizing MI between representations encoded from different structural views of graphs. We show that our approach outperforms previous self-supervised models with significant margin on both node and graph classification tasks without requiring specialized architectures. We also show that when compared to supervised baselines, it performs on par with or better than strong baselines on some benchmarks.

To further improve contrastive representation learning on node and graph classification tasks, we systematically study the major components of our framework and surprisingly show that unlike visual contrastive learning: (1) increasing the number of views, i.e., augmentations, to more than
two views does not improve the performance and the best performance is achieved by contrasting encodings from first-order neighbors and a general graph diffusion, (2) contrasting node and graph encodings across views achieves better results on both tasks compared to contrasting graph or multi-scale encodings, (3) a simple graph readout layer achieves better performance on both tasks compared to hierarchical graph pooling methods such as differentiable pooling (DiffPool) (Ying et al., 2018), and (4) applying regularization (except early-stopping) or normalization layers has a negative effect on the performance.

Using these findings, we achieve new state-of-the-art in self-supervised learning on 8 out of 8 node and graph classification benchmarks under the linear evaluation protocol. For example, on Cora node classification benchmark, our approach achieves 86.8% accuracy, which is a 5.5% relative improvement over previous state-of-the-art (Veličković et al., 2019), and on Reddit-Binary graph classification benchmark, it achieves 84.5% accuracy, i.e., a 2.4% relative improvement over previous state-of-the-art (Sun et al., 2020). When compared to supervised baselines, our approach performs on par with or better than strong supervised baselines, e.g., graph isomorphism network (GIN) (Xu et al., 2019b) and graph attention network (GAT) (Veličković et al., 2018), on 4 out of 8 benchmarks. As an instance, on Cora (node) and IMDB-Binary (graph) classification benchmarks, we observe 4.5% and 5.3% relative improvements over GAT, respectively.

2. Related Work

2.1. Unsupervised Representation Learning on Graphs

Random walks (Perozzi et al., 2014; Tang et al., 2015; Grover & Leskovec, 2016; Hamilton et al., 2017) flatten graphs into sequences by taking random walks across nodes and use language models to learn node representations. They are shown to over-emphasize proximity information at the expense of structural information (Veličković et al., 2019; Ribeiro et al., 2017). Also, they are limited to transductive settings and cannot use node features (You et al., 2019).

Graph kernels (Borgwardt & Kriege, 2005; Shervashidze et al., 2009; 2011; Yanardag & Vishwani, 2015; Kondor & Pan, 2016; Kriege et al., 2016) decompose graphs into substructures and use kernel functions to measure graph similarity between them. Nevertheless, they require non-trivial task of devising similarity measures between substructures.

Graph autoencoders (GAE) (Kipf & Welling, 2016; Garcia Duran & Niepert, 2017; Wang et al., 2017; Pan et al., 2018; Park et al., 2019) train encoders that impose the topological closeness of nodes in the graph structure on the latent space by predicting the first-order neighbors. GAEs overemphasize proximity information (Veličković et al., 2019) and suffer from unstructured predictions (Tian et al., 2019).

Contrastive methods (Li et al., 2019; Veličković et al., 2019; Sun et al., 2020) measure the loss in latent space by contrasting samples from a distribution that contains dependencies of interest and the distribution that does not. These methods are the current state-of-the-art in unsupervised node and graph classification tasks. Deep graph Infomax (DGI) (Veličković et al., 2019) extends deep InfoMax (Hjelm et al., 2019) to graphs and achieves state-of-the-art results in node classification benchmarks by learning node representations through contrasting node and graph encodings. InfoGraph (Sun et al., 2020), on the other hand, extends deep InfoMax to learn graph-level representations and outperforms previous models on unsupervised graph classification tasks. Although these two methods use the same contrastive learning approach, they utilize specialized encoders.

2.2. Graph Diffusion Networks

Graph diffusion networks (GDN) reconcile spatial message passing and generalized graph diffusion (Klicpera et al., 2019b) where diffusion as a denoising filter allows messages to pass through higher-order neighborhoods. GDNs can be categorized to early- and late- fusion models based on the stage the diffusion is used. Early-fusion models (Xu et al., 2019a; Jiang et al., 2019) use graph diffusion to decode the neighbors, e.g., graph diffusion convolution (GDC) replaces adjacency matrix in graph convolution with a sparsified diffusion matrix (Klicpera et al., 2019b), whereas, late-fusion models (Tsitsulin et al., 2018; Klicpera et al., 2019a) project the node features into a latent space and then propagate the learned representation based on a diffusion.

2.3. Learning by Mutual Information Maximization

InfoMax principle (Linsker, 1988) encourages an encoder to learn representations that maximizes the MI between the input and the learned representation. Recently, a few self-supervised models inspired by this principle are proposed which estimate the lower bound of the InfoMax objective, e.g., using noise contrastive estimation (Gutmann & Hyvärinen, 2010), across representations. Contrastive predictive coding (CPC) (Oord et al., 2018) contrasts a summary of ordered local features to predict a local feature in the future whereas deep InfoMax (DIM) (Hjelm et al., 2019) simultaneously contrasts a single summary feature, i.e., global feature, with all local features. Contrastive multiview coding (CMC) (Tian et al., 2019), augmented multi-scale DIM (AMDIM) (Bachman et al., 2019), and SimCLR (Chen et al., 2020) extend the InfoMax principle to multiple views and maximize the MI across views generated by composition of data augmentations. Nevertheless, it is shown that success of these models cannot only be attributed to the properties of MI alone, and the choice of encoder and MI estimators have a significant impact on the performance (Tschannen et al., 2020).
3. Method

Inspired by recent advances in multi-view contrastive learning for visual representation learning, our approach learns node and graph representations by maximizing MI between node representations of one view and graph representation of another view and vice versa which achieves better results compared to contrasting global or multi-scale encodings on both node and graph classification tasks (see section 4.4). As shown in Figure 1, our method consists of the following components:

- An augmentation mechanism that transforms a sample graph into a correlated view of the same graph. We only apply the augmentation to the structure of the graphs and not the initial node features. This is followed by a sampler that sub-samples identical nodes from both views, i.e., similar to cropping in visual domain.

- Two dedicated GNNs, i.e., graph encoders, one for each view, followed by a shared MLP, i.e., projection head, to learn node representations for both views.

- A graph pooling layer, i.e., readout function, followed by a shared MLP, i.e., projection head, to learn graph representations for both views.

- A discriminator that contrasts node representations from one view with graph representation from another view and vice versa, and scores the agreement between representations which is used as the training signal.

3.1. Augmentations

Recent works in self-supervised visual representation learning suggest that contrasting congruent and incongruent views of images allows encoders to learn rich representations (Tian et al., 2019; Bachman et al., 2019). Unlike images in which views are generated by standard augmentations, e.g., cropping, rotating, distorting colors, etc., defining views on graphs is not a trivial task. We can consider two types of augmentations on graphs: (1) feature-space augmentations operating on initial node features, e.g., masking or adding Gaussian noise, and (2) structure-space augmentations and corruptions operating on graph structure by adding or removing connectivities, sub-sampling, or generating global views using shortest distances or diffusion matrices. The former augmentation can be problematic as many benchmarks do not carry initial node features. Moreover, we observed that masking or adding noise on either spaces degrades the performance. Hence, we opted for generating a global view followed by sub-sampling.

We empirically show that in most cases the best results are achieved by transforming an adjacency matrix to a diffusion matrix and treating the two matrices as two congruent views of the same graph’s structure (see section 4.4). We speculate that because adjacency and diffusion matrices provide local and global views of a graph structure, respectively, maximizing agreement between representation learned from these two views allows the model to simultaneously encode rich local and global information.
Diffusion is formulated as Eq. (1) where \( T \in \mathbb{R}^{n \times n} \) is the generalized transition matrix and \( \Theta \) is the weighting coefficient which determines the ratio of global-local information. Imposing \( \sum_{k=0}^{\infty} \theta_k = 1 \), \( \theta_k \in [0, 1] \), and \( \lambda_i \in [0, 1] \) where \( \lambda_i \) are eigenvalues of \( T \), guarantees convergence. Diffusion is computed once using fast approximation and sparsification methods (Klicpera et al., 2019b).

\[
S = \sum_{k=0}^{\infty} \Theta_k T^k \in \mathbb{R}^{n \times n}
\]

Given an adjacency matrix \( A \in \mathbb{R}^{n \times n} \) and a diagonal degree matrix \( D \in \mathbb{R}^{n \times n} \), Personalized PageRank (PPR) (Page et al., 1999) and heat kernel (Kondor & Lafferty, 2002), i.e., two instantiations of the generalized graph diffusion, are defined by setting \( T = AD^{-1} \), and \( \theta_k = \alpha(1-\alpha)^k \) and \( \theta_k = e^{-t_k^2/k!} \), respectively, where \( \alpha \) denotes teleport probability in a random walk and \( t \) is diffusion time (Klicpera et al., 2019b). Closed-form solutions to heat and PPR diffusion are formulated in Eq. (2) and (3), respectively.

\[
S^{\text{heat}} = \exp \left( tAD^{-1} - t \right)
\]

\[
S^{\text{PPR}} = \alpha \left( I_n - (1-\alpha)D^{-1/2}AD^{-1/2} \right)^{-1}
\]

For sub-sampling, we randomly sample nodes and their edges from one view and select the exact nodes and edges from the other view. This procedure allows our approach to be applied to inductive tasks with graphs that do not fit into the GPU memory and also to transductive tasks by considering sub-samples as independent graphs.

### 3.2. Encoders

Our framework allows various choices of the network architecture without any constraints. We opt for simplicity and adopt the commonly used graph convolution network (GCN) (Kipf & Welling, 2017) as our base graph encoders. As shown in Figure 1, we use a dedicated graph encoder for each view, i.e., \( g_\theta(\cdot), g_\omega(\cdot) : \mathbb{R}^{n \times d_x} \times \mathbb{R}^{n \times n} \rightarrow \mathbb{R}^{n \times d_h} \). We consider adjacency and diffusion matrices as two congruent structural views and define GCN layers as \( \sigma(A\mathbf{X}\Theta) \) and \( \sigma(S\mathbf{X}\Theta) \) to learn two sets of node representations each corresponding to one of the views, respectively.

\( \hat{A} = \hat{D}^{-1/2} \hat{A} \hat{D}^{-1/2} \in \mathbb{R}^{n \times n} \) is symmetrically normalized adjacency matrix, \( \hat{D} \in \mathbb{R}^{n \times n} \) is the degree matrix of \( \hat{A} = A + I_n \) where \( I_n \) is the identity matrix, \( \mathbf{S} \in \mathbb{R}^{n \times n} \) is diffusion matrix, \( \mathbf{X} \in \mathbb{R}^{n \times d_x} \) is the initial node features, \( \Theta \in \mathbb{R}^{d_x \times d_h} \) is network parameters, and \( \sigma \) is a parametric ReLU (PReLU) non-linearity (He et al., 2015). The learned representations are then fed into a shared projection head \( f_\phi(\cdot) : \mathbb{R}^{n \times d_h} \rightarrow \mathbb{R}^{d_h \times d_h} \) which is an MLP with two hidden layers and PReLU non-linearity. This results in two sets of node representations \( \mathbf{H}_\alpha, \mathbf{H}_\beta \in \mathbb{R}^{n \times d_h} \) corresponding to two congruent views of a same graph.

For each view, we aggregate the node representations learned by GNNs, i.e., before the projection head, into a graph representation using a graph pooling (readout) function \( \mathcal{P}(\cdot) : \mathbb{R}^{n \times d_h} \rightarrow \mathbb{R}^{d_h} \). We use a readout function similar to jumping knowledge network (JK-Net) (Xu et al., 2018) where we concatenate the summation of the node representations in each GCN layer and then feed them to a single layer feed-forward network to have a consistent dimension size between node and graph representations:

\[
\tilde{h}_g = \sigma \left( L \left\| \sum_{i=1}^{n} \tilde{h}_i^{(l)} \right\| W \right) \in \mathbb{R}^{h_d}
\]

where \( \tilde{h}_i^{(l)} \) is the latent representation of node \( i \) in layer \( l \), \( \| \) is the concatenation operator, \( L \) is the number of GCN layers, \( W \in \mathbb{R}^{(L \times d_h) \times d_h} \) is the network parameters, and \( \sigma \) is a PReLU non-linearity. In section 4.4, we show that this pooling function achieves better results compared to more complicated graph pooling methods such as DiffPool (Ying et al., 2018). Applying the readout function to node representations results in two graph representations, each associated with one of the views. The representations are then fed into a shared projection head \( f_\phi(\cdot) : \mathbb{R}^{d_h} \rightarrow \mathbb{R}^{d_h} \), which is an MLP with two hidden layers and PReLU non-linearity, resulting in the final graph representations: \( \tilde{h}_\alpha, \tilde{h}_\beta \in \mathbb{R}^{d_h} \).

At inference time, we aggregate the representations from both views in both node and graph levels by summing them up: \( \tilde{h} = \tilde{h}_\alpha + \tilde{h}_\beta \in \mathbb{R}^n \) and \( \mathbf{H} = \mathbf{H}_\alpha + \mathbf{H}_\beta \in \mathbb{R}^{n \times d_h} \) and return them as graph and node representations, respectively, for the down-stream tasks.

### 3.3. Training

In order to train the encoders end-to-end and learn rich node and graph level representations that are agnostic to downstream tasks, we utilize the deep InfoMax (Hjelm et al., 2019) approach and maximize the MI between two views by contrasting node representations of one view with graph representation of the other view and vice versa. We empirically show that this approach consistently outperforms contrasting graph-graph or multi-scale encodings on both node and graph classifications benchmarks (see section 4.4). We define the objective as follows:

\[
\max_{\theta, \omega, \phi, \psi} \frac{1}{|G|} \sum_{g \in G} \left[ \frac{1}{|g|} \sum_{i=1}^{|g|} \left[ \text{MI} \left( \tilde{h}_i^{(g)}, \tilde{h}_g^{(\alpha)} \right) + \text{MI} \left( \tilde{h}_i^{(\alpha)}, \tilde{h}_g^{(\beta)} \right) \right] \right]
\]

where \( \theta, \omega, \phi, \psi \) are graph encoder and projection head parameters, \( |G| \) is the number of graphs in train set or number of sub-sampled graphs in transductive setting, \( |g| \) is the number of nodes in graph \( g \), and \( \tilde{h}_i^{(g)}, \tilde{h}_g^{(\alpha)}, \tilde{h}_g^{(\beta)} \) are representations of node \( i \) and graph \( g \) encoded from views \( \alpha, \beta \), respectively.
We provide the positive samples from joint distribution \( \mathcal{D}(\ldots) : \mathbb{R}^{d_h} \times \mathbb{R}^{d_h} \rightarrow \mathbb{R} \) that takes in a node representation from one view and a graph representation from another view, and scores the agreement between them. We simply implement the discriminator as the dot product between two representations: \( \mathcal{D}(h_{\alpha}, h_{\beta}) = h_{\alpha} \cdot h_{\beta}^T \). We observed a slight improvement in node classification benchmarks when discriminator and projection heads are integrated into a bilinear layer. In order to decide the MI estimator, we investigate four estimators and chose the best one for each benchmark (see section 4.4).

We provide the positive samples from joint distribution \( x_p \sim p([X, \tau_\alpha(A)], [X, \tau_\beta(A)]) \) and the negative samples from the product of marginals \( x_n \sim p([X, \tau_\alpha(A)])p([X, \tau_\beta(A)]) \). To generate negative samples in transductive tasks, we randomly shuffle the features \( \forall x \) in node classification benchmarks when discriminator and agreement between them. We simply implement the discriminator \( \text{MI is modeled as discriminator } \mathcal{D}(K, \beta) \in \mathbb{R} \) that takes in a node representation from one view and a graph representation from another view, and scores the agreement between them. We simply implement the discriminator as the dot product between two representations: \( \mathcal{D}(h_{\alpha}, h_{\beta}) = h_{\alpha} \cdot h_{\beta}^T \). We observed a slight improvement in node classification benchmarks when discriminator and projection heads are integrated into a bilinear layer. In order to decide the MI estimator, we investigate four estimators and chose the best one for each benchmark (see section 4.4).

We provide the positive samples from joint distribution \( x_p \sim p([X, \tau_\alpha(A)], [X, \tau_\beta(A)]) \) and the negative samples from the product of marginals \( x_n \sim p([X, \tau_\alpha(A)])p([X, \tau_\beta(A)]) \). To generate negative samples in transductive tasks, we randomly shuffle the features \( \forall x \) in node classification benchmarks when discriminator and agreement between them. We simply implement the discriminator \( \text{MI is modeled as discriminator } \mathcal{D}(K, \beta) \in \mathbb{R} \) that takes in a node representation from one view and a graph representation from another view, and scores the agreement between them. We simply implement the discriminator as the dot product between two representations: \( \mathcal{D}(h_{\alpha}, h_{\beta}) = h_{\alpha} \cdot h_{\beta}^T \). We observed a slight improvement in node classification benchmarks when discriminator and projection heads are integrated into a bilinear layer. In order to decide the MI estimator, we investigate four estimators and chose the best one for each benchmark (see section 4.4).

Algorithm 1 Contrastive multi-view graph representation learning algorithm.

**Input:** Augmentations \( \tau_\alpha \) and \( \tau_\beta \), sampler \( \Gamma \), pooling \( \mathcal{P} \), discriminator \( \mathcal{D} \), loss \( \mathcal{L} \), encoders \( g_\theta, g_\omega, f_\psi, f_\phi \), and training graphs \( \{G[g = (X, A) \in \mathcal{G}] \) for sampled batch \( \{g_k\}_{k=1}^K \in \mathcal{G} \) do

```plaintext
for \( k = 1 \) to \( N \) do
    \( X_k, A_k = \Gamma(g_k) \) // sub-sample graph
    \( V_k^\alpha = \tau_\alpha(A_k) \) // first view
    \( H_k^\alpha = f_\phi(Z_k^\alpha) \) // node rep.
    \( \tilde{H}_k^\alpha = f_\phi(\mathcal{P}(Z_k^\alpha)) \) // projected node rep.
    \( V_k^\beta = \tau_\beta(A_k) \) // second view
    \( H_k^\beta = f_\phi(Z_k^\beta) \) // node rep.
    \( \tilde{H}_k^\beta = f_\phi(\mathcal{P}(Z_k^\beta)) \) // projected node rep.
end
```

```plaintext
// compute pairwise similarity:
for \( i = 1 \) to \( N \) and \( j = 1 \) to \( N \) do
    \( s_{ij}^\alpha = \mathcal{D}(\tilde{h}_i^\alpha, H_j^\alpha), s_{ij}^\beta = \mathcal{D}(\tilde{h}_i^\beta, H_j^\beta) \)
end
```

```plaintext
// compute gradients:
\[ \nabla_{\theta, \omega, \psi} \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \left[ \mathcal{L}(s_{ij}^\alpha) + \mathcal{L}(s_{ij}^\beta) \right] \]
```

end

```plaintext
return \( [H_g^\alpha + H_g^\beta, \tilde{H}_g^\alpha + \tilde{H}_g^\beta] \), \( \forall g \in \mathcal{G} \)
```

4. Experimental Results

4.1. Benchmarks

We use three node classification and five graph classification benchmarks widely used in the literature (Kipf & Welling, 2017; Veličković et al., 2018; 2019; Sun et al., 2020). For node classification, we use Citeseer, Cora, and Pubmed citation networks (Sen et al., 2008) where documents (nodes) are connected through citations (edges). For graph classification, we use the following: MUTAG (Kriege & Mutzel, 2012) containing mutagenic compounds, PTC (Kriege & Mutzel, 2012) containing compounds tested for carcinogenicity, Reddit-Binary (Yanardag & Vishwanath, 2015) connecting users (nodes) through responses (edges) in Reddit online discussions, and IMDB-Binary and IMDB-Multi (Yanardag & Vishwanath, 2015) connecting actors/actresses (nodes) based on movie appearances (edges). The statistics are summarized in Table 1.

4.2. Evaluation Protocol

For both node and graph classification benchmarks, we evaluate the proposed approach under the linear evaluation protocol and for each task, we closely follow the experimental protocol of the previous state-of-the-art approaches. For node classification, we follow DGI and report the mean classification accuracy with standard deviation on the test nodes after 50 runs of training followed by a linear model. For graph classification, we follow InfoGraph and report the mean 10-fold cross validation accuracy with standard deviation after 5 runs followed by a linear SVM. The linear classifier is trained using cross validation on training folds of data and the best mean classification accuracy is reported. Moreover, for node classification benchmarks, we evaluate the proposed method under clustering evaluation protocol and cluster the learned representations using K-Means algorithm. Similar to (Park et al., 2019), we set the number of clusters to the number of ground-truth classes and report the average normalized MI (NMI) and adjusted rand index (ARI) for 50 runs.

We initialize the parameters using Xavier initialization (Glorot & Bengio, 2010) and train the model using Adam optimizer (Kingma & Ba, 2014) with an initial learning rate of 0.001. To have fair comparisons, we follow InfoGraph for graph classification and choose the number of GCN layers, number of epochs, batch size, and the C parameter of the SVM from \([2, 4, 8, 12], [10, 20, 40, 100], [32, 64, 128, 256], \) and \([10^{-3}, 10^{-2}, ..., 10^3]\), respectively. For node classification, we follow DGI and set the number of GCN layers and the number of epochs and to 1 and 2,000, respectively, and choose the batch size from \([2, 4, 8]\). We also use early stopping with a patience of 20. Finally, we set the size of hidden dimension of both node and graph representations to 512. For chosen hyper-parameters see Appendix.
### Table 1. Statistics of classification benchmarks.

<table>
<thead>
<tr>
<th>Method</th>
<th>CORA</th>
<th>CITESEER</th>
<th>PUBMED</th>
<th>MUTAG</th>
<th>PTC-MR</th>
<th>IMDB-BIN</th>
<th>IMDB-MULTI</th>
<th>REDDIT-BIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP</td>
<td>51.5</td>
<td>46.5</td>
<td>71.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICA</td>
<td>75.1</td>
<td>69.1</td>
<td>73.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LP</td>
<td>68.0</td>
<td>45.3</td>
<td>63.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MANIREG</td>
<td>59.5</td>
<td>60.1</td>
<td>70.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEMIEMB</td>
<td>59.6</td>
<td>56.9</td>
<td>71.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLANETOID</td>
<td>75.7</td>
<td>64.7</td>
<td>77.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHERBYCHEV</td>
<td>81.2</td>
<td>69.8</td>
<td>74.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GCN</td>
<td>81.5</td>
<td>70.3</td>
<td>79.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MoNET</td>
<td>81.7 ± 0.5</td>
<td>–</td>
<td>78.8 ± 0.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JKNET</td>
<td>82.7 ± 0.4</td>
<td>73.0 ± 0.5</td>
<td>77.9 ± 0.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GAT</td>
<td>83.0 ± 0.7</td>
<td>72.5 ± 0.7</td>
<td>79.0 ± 0.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 2. Mean node classification accuracy for supervised and unsupervised models. The input column highlights the data available to each model during training (X: features, A: adjacency matrix, S: diffusion matrix, Y: labels).

<table>
<thead>
<tr>
<th>Method</th>
<th>Input</th>
<th>CORA</th>
<th>CITESEER</th>
<th>PUBMED</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP</td>
<td>X, Y</td>
<td>51.5</td>
<td>46.5</td>
<td>71.4</td>
</tr>
<tr>
<td>ICA</td>
<td>A, Y</td>
<td>75.1</td>
<td>69.1</td>
<td>73.9</td>
</tr>
<tr>
<td>LP</td>
<td>A, Y</td>
<td>68.0</td>
<td>45.3</td>
<td>63.0</td>
</tr>
<tr>
<td>MANIREG</td>
<td>X, A, Y</td>
<td>59.5</td>
<td>60.1</td>
<td>70.7</td>
</tr>
<tr>
<td>SEMIEMB</td>
<td>X, Y</td>
<td>59.6</td>
<td>56.9</td>
<td>71.7</td>
</tr>
<tr>
<td>PLANETOID</td>
<td>X, Y</td>
<td>75.7</td>
<td>64.7</td>
<td>77.2</td>
</tr>
<tr>
<td>CHERBYCHEV</td>
<td>X, A, Y</td>
<td>81.2</td>
<td>69.8</td>
<td>74.4</td>
</tr>
<tr>
<td>GCN</td>
<td>X, A</td>
<td>81.5</td>
<td>70.3</td>
<td>79.0</td>
</tr>
<tr>
<td>MoNET</td>
<td>X, A, Y</td>
<td>81.7 ± 0.5</td>
<td>–</td>
<td>78.8 ± 0.3</td>
</tr>
<tr>
<td>JKNET</td>
<td>X, A, Y</td>
<td>82.7 ± 0.4</td>
<td>73.0 ± 0.5</td>
<td>77.9 ± 0.4</td>
</tr>
<tr>
<td>GAT</td>
<td>X, A</td>
<td>83.0 ± 0.7</td>
<td>72.5 ± 0.7</td>
<td>79.0 ± 0.3</td>
</tr>
</tbody>
</table>

### Table 3. Performance on node clustering task reported in normalized MI (NMI) and adjusted rand index (ARI) measures.

<table>
<thead>
<tr>
<th>Method</th>
<th>CORA</th>
<th>CITESEER</th>
<th>PUBMED</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGAE (KIPF &amp; WELLING, 2016)</td>
<td>0.3292</td>
<td>0.2547</td>
<td>0.2605</td>
</tr>
<tr>
<td>MGAE (WANG ET AL., 2017)</td>
<td>0.5111</td>
<td>0.4447</td>
<td>0.4122</td>
</tr>
<tr>
<td>ARGA (PAN ET AL., 2018)</td>
<td>0.4490</td>
<td>0.3520</td>
<td>0.3500</td>
</tr>
<tr>
<td>ARVGA (PAN ET AL., 2018)</td>
<td>0.4500</td>
<td>0.3740</td>
<td>0.2610</td>
</tr>
<tr>
<td>GALA (PARK ET AL., 2019)</td>
<td>0.5767</td>
<td>0.5315</td>
<td>0.4411</td>
</tr>
<tr>
<td>OURS</td>
<td>0.6291</td>
<td>0.5696</td>
<td>0.4696</td>
</tr>
</tbody>
</table>

### Table 4. Mean 10-fold cross validation accuracy on graphs for kernel, supervised, and unsupervised methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>CORA</th>
<th>CITESEER</th>
<th>PUBMED</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP (BORGWARTD &amp; KRIEGEL, 2005)</td>
<td>85.2 ± 2.4</td>
<td>58.2 ± 2.4</td>
<td>55.6 ± 0.2</td>
</tr>
<tr>
<td>GK (SHERVASHDIZADE ET AL., 2009)</td>
<td>81.7 ± 2.1</td>
<td>57.3 ± 1.4</td>
<td>65.9 ± 0.0</td>
</tr>
<tr>
<td>WL (SHERVASHDIZADE ET AL., 2011)</td>
<td>80.7 ± 3.0</td>
<td>58.0 ± 0.5</td>
<td>72.3 ± 3.4</td>
</tr>
<tr>
<td>DGK (YANARDAG &amp; VISHWANA, 2015)</td>
<td>87.4 ± 2.7</td>
<td>60.1 ± 2.6</td>
<td>67.0 ± 0.6</td>
</tr>
<tr>
<td>MLG (KONDOR &amp; PAN, 2016)</td>
<td>87.9 ± 1.6</td>
<td>63.3 ± 1.5</td>
<td>66.6 ± 0.3</td>
</tr>
<tr>
<td>GRAPH SAGE (HAMILTON ET AL., 2017)</td>
<td>85.1 ± 7.6</td>
<td>63.9 ± 7.7</td>
<td>72.3 ± 5.3</td>
</tr>
<tr>
<td>GCN (KIPF &amp; WELLING, 2017)</td>
<td>85.6 ± 5.8</td>
<td>64.2 ± 4.3</td>
<td>74.0 ± 3.4</td>
</tr>
<tr>
<td>GIN (XU ET AL., 2019)</td>
<td>89.4 ± 5.6</td>
<td>64.6 ± 7.0</td>
<td>75.1 ± 5.1</td>
</tr>
<tr>
<td>GAT (VELIČKOVIC ET AL., 2018)</td>
<td>89.4 ± 6.1</td>
<td>66.7 ± 5.1</td>
<td>70.5 ± 2.3</td>
</tr>
<tr>
<td>RANDOM WALK (GÄRTNER ET AL., 2003)</td>
<td>83.7 ± 1.5</td>
<td>57.9 ± 1.3</td>
<td>50.7 ± 0.3</td>
</tr>
<tr>
<td>NODE2VEC (GROVER &amp; LESKOVEC, 2016)</td>
<td>72.6 ± 10.2</td>
<td>58.6 ± 8.0</td>
<td>–</td>
</tr>
<tr>
<td>SUB2VEC (ADHIKARI ET AL., 2018)</td>
<td>61.1 ± 15.8</td>
<td>60.0 ± 6.4</td>
<td>55.3 ± 1.5</td>
</tr>
<tr>
<td>GRAPH2VEC (NARAYANAN ET AL., 2017)</td>
<td>83.2 ± 9.6</td>
<td>60.2 ± 6.9</td>
<td>71.1 ± 0.5</td>
</tr>
<tr>
<td>INFOGRAPH (SUN ET AL., 2020)</td>
<td>89.0 ± 1.1</td>
<td>61.7 ± 1.4</td>
<td>73.0 ± 0.9</td>
</tr>
<tr>
<td>OURS</td>
<td>89.7 ± 1.1</td>
<td>62.5 ± 1.7</td>
<td>74.2 ± 0.7</td>
</tr>
</tbody>
</table>
4.3. Comparison with State-of-the-Art

To evaluate node classification under the linear evaluation protocol, we compare our results with unsupervised models including DeepWalk (Perozzi et al., 2014) and DGI. We also train a GAE (Kipf & Welling, 2016), a variant of DGI with a GDC encoder, and a variant of VERSE (Tsitulin et al., 2018) by minimizing KL-divergence between node representations and diffusion matrix. Furthermore, we compare our results with supervised models including an MLP, iterative classification algorithm (ICA) (Lu & Getoor, 2003), label propagation (LP) (Zhu et al., 2003), manifold regularization (ManiReg) (Belkin et al., 2006), semi-supervised embedding (SemiEmb) (Weston et al., 2012), Planetoid (Yang et al., 2016), Chebyshev (Defferrard et al., 2016), mixture model networks (MoNet) (Monti et al., 2017), JK-Net, GCN, and GAT. The results reported in Table 2 show that we achieve state-of-the-art results with respect to previous unsupervised models. For example, on Cora, we achieve 86.8% accuracy, which is a 5.5% relative improvement over previous state-of-the-art. When compared to supervised baselines, we outperform strong supervised baselines: on Cora and PubMed benchmarks we observe 4.5% and 1.4% relative improvement over GAT, respectively.

To evaluate node classification under the clustering protocol, we compare our model with models reported in (Park et al., 2019) including: variational GAE (VGAE) (Kipf & Welling, 2016), marginalized GAE (MGAE) (Wang et al., 2017), adversarially regularized GAE (ARGA) and VGAE (ARVG) (Pan et al., 2018), and GALA (Park et al., 2019). The results shown in Table 3 suggest that our model achieves state-of-the-art NMI and ARI scores across all benchmarks.

To evaluate graph classification under the linear evaluation protocol, we compare our results with five graph kernel methods including shortest path kernel (SP) (Borgwardt & Kriegel, 2005), Graphlet kernel (GK) (Shervashidze et al., 2009), Weisfeiler-Lehman sub-tree kernel (WL) (Shervashidze et al., 2011), deep graph kernels (DGK) (Yanardag & Vishwan, 2015), and multi-scale Laplacian kernel (MLG) (Kondor & Pan, 2016) reported in (Sun et al., 2020). We also compare with five supervised GNNs reported in (Xu et al., 2019b) including GraphSAGE (Hamilton et al., 2017), GCN, GAT, and two variants of GIN; GIN-0 and GIN-ε. Finally, we compare the results with five unsupervised methods including random walk (Gärtner et al., 2003), node2vec (Grover & Leskovec, 2016), sub2vec (Adhikari et al., 2018), graph2vec (Narayanan et al., 2017), and InfoGraph. The results shown in Table 4 suggest that our approach achieves state-of-the-art results with respect to unsupervised models. For example, on Reddit-Binary (Yanardag & Vishwa, 2015), it achieves 84.5% accuracy, i.e., a 2.4% relative improvement over previous state-of-the-art. Our model also outperforms kerneb methods in 4 out of 5 datasets and also outperforms best supervised model in one of the datasets. When compared to supervised baselines individually, our model outperforms GCN and GAT models in 3 out of 5 datasets, e.g., a 5.3% relative improvement over GAT on IMDB-Binary dataset.

It is noteworthy that we achieve the state-of-the-art results on both node and graph classification benchmarks using a unified approach and unlike previous unsupervised models (Veličković et al., 2019; Sun et al., 2020), we do not devise a specialized encoder for each task.

4.4. Ablation Study

4.4.1. Effect of Mutual Information Estimator

We investigated four MI estimators including: noise-contrastive estimation (NCE) (Gutmann & Hyvärinen, 2010; Oord et al., 2018), Jensen-Shannon (JSD) estimator following formulation in (Nowozin et al., 2016), normalized temperature-scaled cross-entropy (NT-Xent) (Chen et al., 2020), and Donsker-Varadhan (DV) representation of the KL-divergence (Donsker & Varadhan, 1975). The results shown in Table 5 suggests that Jensen-Shannon estimator achieves better results across all graph classification benchmarks, whereas in node classification benchmarks, NCE achieves better results in 2 out of 3 datasets.

4.4.2. Effect of Contrastive Mode

We considered five contrastive modes including: local-global, global-global, multi-scale, hybrid, and ensemble modes. In local-global mode we extend deep InfoMax (Hjelm et al., 2019) and contrast node encodings from one view with graph encodings from another view and vice versa. Global-global mode is similar to (Li et al., 2019; Tian et al., 2019; Chen et al., 2020) where we contrast graph encodings from different views. In multi-scale mode, we contrast graph encodings from one view with intermediate encodings from another view and vice versa, and we also contrast intermediate encodings from one view with node encodings from another view and vice versa. We use two DiffPool layers to compute the intermediate encodings. The first layer projects nodes into a set of clusters where the number of clusters, i.e., motifs, is set as 25% of the number of nodes before applying DiffPool, whereas the second layer projects the learned cluster encodings into a graph encoding. In hybrid mode, we use both local-global and global-global modes. Finally, in ensemble mode, we contrast nodes and graph encodings from a same view for all views.

Results reported in Table 5 suggest that contrasting node and graph encodings consistently perform better across benchmarks. The results also reveal important differences between graph and visual representation learning: (1) in visual representation learning, contrasting global views achieves
We also followed (Chen et al., 2020) and randomly sampled Warshall algorithm, inversed in an element-wise fashion, We investigated four structural views including adjacency matrices carry similar information about the graph structure. We showed that unlike visual representation learning, increasing the number of views or contrasting multi-scale encodings does not improve performance, whereas, in node classification, it does not have a significant effect. Thus we opted for efficient smaller batch sizes. To generate negative samples in node classification, we considered two corruption functions: (1) random feature permutation, and (2) adjacency matrix corruption. We observed that applying the former achieves significantly better results compared to the latter or a combination of both.

Furthermore, we observed that applying normalization layers such as BatchNorm (Ioffe & Szegedy, 2015) or LayerNorm (Ba et al., 2016), or regularization methods such as adding Gaussian noise, L2 regularization, or dropout (Srivastava et al., 2014) during the pre-training degrades the performance on down-stream tasks (except early stopping).

### 5. Conclusion

We introduced a self-supervised approach for learning node and graph level representations by contrasting encodings from two structural views of graphs including first-order neighbors and a graph diffusion. We showed that unlike visual representation learning, increasing the number of views or contrasting multi-scale encodings do not improve the performance. Using these findings, we achieved new state-of-the-art in self-supervised learning on 8 out of 8 node and graph classification benchmarks under the linear evaluation protocol and outperformed strong supervised baselines in 4 out of 8 benchmarks. In future work, we are planning to investigate large pre-training and transfer learning capabilities of the proposed method.
Acknowledgements

We would like to thank Adamo Young, Adrian Butscher, and Tonya Custis for their feedback. We are also grateful for general support from Autodesk AI lab and Vector institute teams in Toronto.

References


Contrastive Multi-View Representation Learning on Graphs


