Self-Supervised Graph Representation Learning via Topology Transformations

Xiang Gao, Student Member, IEEE, Wei Hu, Senior Member, IEEE, and Guo-Jun Qi, Fellow, IEEE

Abstract—We present the Topology Transformation Equivariant Representation learning, a general paradigm of self-supervised learning for node representations of graph data to enable the wide applicability of Graph Convolutional Neural Networks (GCNNs). We formalize the proposed model from an information-theoretic perspective, by maximizing the mutual information between topology transformations and node representations before and after the transformations. We derive that maximizing such mutual information can be relaxed to minimizing the cross entropy between the applied topology transformation and its estimation from node representations. In particular, we seek to sample a subset of node pairs from the original graph and flip the edge connectivity between each pair to transform the graph topology. Then, we self-train a representation encoder to learn node representations by reconstructing the topology transformations from the feature representations of the original and transformed graphs. In experiments, we apply the proposed model to the downstream node classification, graph classification and link prediction tasks, and results show that the proposed method outperforms the state-of-the-art unsupervised approaches.

Index Terms—Self-supervised learning, graph representation learning, topology transformation, transformation equivariant representation.

1 INTRODUCTION

Graphs provide a natural and efficient representation for non-Euclidean data, such as brain networks, social networks, citation networks, and 3D point clouds. Graph Convolutional Neural Networks (GCNNs) [1] have been proposed to generalize the CNNs to learn representations from non-Euclidean data, which has made significant advances in various applications such as node classification [2], [3], [4] and graph classification [5]. However, most existing GCNNs are trained in a supervised fashion, requiring a large amount of labeled data for network training. This limits the applications of the GCNNs since it is often costly to collect adequately labeled data, especially on large-scale graphs. Hence, self-supervised learning is required to learn graph feature representations by exploring the dependencies of unlabeled data in an unsupervised fashion, which enables the discovery of intrinsic graph structures and thus adapts to various downstream tasks.

Various attempts have been made to explore self-supervisory signals for representation learning. The self-supervised learning framework requires only unlabeled data in order to design a pretext learning task, where the target objective is optimized without any supervision [6]. Self-supervised learning models can be categorized into three classes [7]: generative, adversarial, and contrastive. Generative models are often based on Auto-regressive models [8], [9], [10], flow-based models [11], [12], and Auto-Encoding (AE) models [13], [14], [15] to generate or reconstruct data from latent representations. Adversarial models extract feature representations in an unsupervised fashion by generating data from input noises via a pair of generator and discriminator [16], [17]. Contrastive models aim to train an encoder to be contrastive between the representations of positive samples and negative samples [18], [19], [20], [21].

Recently, many approaches have sought to learn transformation equivariant representations (TERs) to further improve the quality of unsupervised representation learning. It assumes that the learned representations equivarying to transformations are able to encode the intrinsic structures of data such that the transformations can be reconstructed from the representations before and after transformations [22]. Learning TERs traces back to Hinton’s seminal work on learning transformation capsules [23], and embodies a variety of methods developed for Euclidean data [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34]. Further, Gao et al. [35] extend transformation equivariant representation learning to non-Euclidean domain, which formalizes Graph Transformation Equivariant Representation (Graph-TER) learning by auto-encoding node-wise transformations in an unsupervised fashion. Nevertheless, only transformations on node features are explored, while the underlying graph may vary implicitly. The graph topology has not been fully explored yet, which however is crucial in graph representation learning.

To this end, we propose a self-supervised Topology Transformation Equivariant Representation learning to infer expressive graph feature representations by estimating topology transformations. As the topology of graph data is critical in graph-based machine learning, this motivates us to study the properties of graph topology including especially the topology transformation equivariance. In par-
particular, the topology of some graph data such as citation networks and social networks may transform over time, leading to the change of node features. When the topology transforms, the node features transform equivariantly in the representation space. Hence, instead of transforming node features as in the GraphTER, the proposed method studies the transformation equivariant representation learning by transforming the graph topology, i.e., adding or removing edges to perturb the graph structure. Then the same input signals are attached to the resultant graph topologies, resulting in different graph representations. This provides an insight into how the same input signals associated with different graph topologies would lead to equivariant representations, enhancing the fusion of node feature and graph topology in GCNNs.

Formally, we formulate the proposed model from an information-theoretic perspective, aiming to maximize the mutual information between topology transformations and feature representations with respect to the original and transformed graphs. We derive that maximizing such mutual information can be relaxed to the cross entropy minimization between the applied topology transformations and the estimation from the learned representations of graph data under the topological transformations.

Specifically, given an input graph and its associated node features, we first sample a subset of node pairs from the graph and flip the edge connectivity between each pair at a perturbation rate, leading to a transformed graph with attached node features. Then, we design a graph-convolutional auto-encoder architecture, where the encoder learns the node-wise representations over the original and transformed graphs respectively, and the decoder predicts the topology transformations of edge connectivity from both representations by minimizing the cross entropy between the applied and estimated transformations. Experimental results demonstrate that the proposed method outperforms the state-of-the-art unsupervised methods in node classification, graph classification, and link prediction. The remainder of this paper is organized as follows. We first review related works in Sec. 2. Then we formalize our model in Sec. 3 and present the algorithm in Sec. 4. Finally, experimental results and conclusions are presented in Sec. 5 respectively.

2 RELATED WORK

We review previous works on relevant unsupervised/self-supervised feature representation learning, including graph auto-encoders, graph generative models, graph contrastive learning, as well as transformation equivariant representation learning.

2.1 Graph Auto-Encoders

Graph Auto-Encoders (GAEs) are the most representative unsupervised methods. GAEs encode graph data into feature space via an encoder and reconstruct the input graph data from the encoded feature representations via a decoder. Kipf et al. [15] first integrate the GCN [2] into an auto-encoder framework to learn graph representations in an unsupervised manner by reconstructing the adjacency matrix. Variational GAE (VGAE) [15] is a variational version of GAE to learn the distribution of data. Cao et al. [36] proposed to employ the stacked denoising auto-encoder [37] to reconstruct the positive pointwise mutual information (PPMI) matrix to capture the correlation of node pairs. Wang et al. [38] employ the stacked auto-encoders to preserve the first-order proximity and the second-order proximity of nodes jointly. Qu et al. [39] proposed the Graph Markov Neural Network (GMNN) to model the joint distribution of object labels with a conditional random field for semi-supervised and unsupervised graph representation learning. Jiang et al. [40] introduce a graph auto-encoder framework that can be used for unsupervised link prediction, where the encoder embeds both nodes and edges to a latent feature space simultaneously. Compared with these auto-encoder-based methods that aim to decode the original graph, our proposed model makes better use of the topology information of graphs and the transformation equivariance property.

Our main contributions are summarized as follows.

• We propose a self-supervised paradigm of the Topology Transformation Equivariant Representation learning to infer expressive node feature representations, which characterizes the intrinsic structures of graphs and the associated features by exploring the graph transformations of connectivity topology.

• We formulate the Topology Transformation Equivariant Representation learning from an information-theoretic perspective, by maximizing the mutual information between feature representations and topology transformations, which is proved to relax to the cross entropy minimization between the applied transformations and the prediction in an end-to-end graph-convolutional auto-encoder architecture.

• Experiments demonstrate that the proposed method outperforms the state-of-the-art unsupervised methods in node classification, graph classification, and link prediction.

1. For instance, with the development of research in different fields, more interdisciplinary citations will appear in citation networks; the personal relationships in social networks may change over time.
Graph Generative Networks aim to learn the generative distribution of graphs by encoding graphs into hidden representations and generate graph structures given hidden representations. The graph generative networks can be classified into two categories: sequential approaches and global approaches. Sequential approaches generate nodes and edges step by step. Deep Generative Model of Graphs (DeepGMG) assumes that the probability of a graph is the sum over all possible node permutations, and generates graphs by making a sequence of decisions. You et al. proposed GraphRNN model to generate nodes from a graph-level RNN and edges from an edge-level RNN. Global approaches generate an entire graph at once. Molecular GAN (MoGAN) combines Relational Graph Convolutional Networks (R-GCNs) and reinforcement learning objectives to generate graphs with desired properties. NetGAN combines LSTMs with the Wasserstein GANs to generate graphs from a random-walk-based approach.

### 2.3 Graph Contrastive Learning

An important paradigm called contrastive learning aims to train an encoder to be contrastive between the representations of positive samples and negative samples. Recent contrastive learning frameworks for graph data can be divided into two categories: context-instance contrast and context-context contrast. Context-instance contrast focuses on modeling the relationships between the local feature of a sample and its global context representation. Deep InfoMax (DIM) first maximizes the mutual information between a local patch and the representations of substructures of different granularity. Peng et al. proposed a Graphical Mutual Information (GMI) approach to maximize the mutual information of both features and edges between inputs and outputs. Compared with context-instance methods, context-context contrast studies the relationships between the representations of entire graphs and the representations of substructures of different granularity. Peng et al. proposed a Graphical Mutual Information (GMI) approach to maximize the mutual information of both features and edges between inputs and outputs.

#### 2.4 Transformation Equivariant Representations

Many approaches have sought to learn transformation equivariant representations, which has been advocated in Hinton’s seminal work on learning transformation capsules. Following this, a variety of approaches have been proposed to learn transformation equivariant representations. To generalize to generic transformations, Zhang et al. proposed to learn unsupervised feature representations via Auto-Encoding Transforms (AET) by estimating transformations from the learned feature representations of both the original and transformed images, while Qi et al. extend AET from an information-theoretic perspective by maximizing the lower bound of mutual information between transformations and representations. Wang et al. extend the AET to Generative Adversarial Networks (GANs) for unsupervised image synthesis and representation learning. Gao et al. introduce the GraphTER model that extends AET to graph-structured data, which is formalized by auto-encoding node-wise transformations in an unsupervised manner. De et al. proposed Gauge Equivariant Mesh CNNs which generalize GCNNs to apply anisotropic gauge equivariant kernels. Fuchs introduce a self-attention mechanism specifically for 3D point cloud data, which adheres to equivariance constraints, improving robustness to nuisance transformations. Haan et al. proposed a Natural Graph Network (NGN) that can be used to describe maximally flexible global and local equivariance. Satorras et al. present an E(n) equivariant graph neural network that is translation, rotation, and reflection equivariant. Gao et al. proposed to learn multi-view representations by decoding the 3D transformations of 3D objects from multiple 2D views. Our approach belongs to this family of methods. The key difference is that, we propose to learn topology transformation equivariant representations by transforming the graph topology.

#### 3 The Proposed Formulation

In this section, we first introduce the preliminaries in Sec. 3.1 and define the topology transformation in Sec. 3.2. Then we formulate the proposed method in Sec. 3.3. Further, some analysis of the proposed model are presented in Sec. 3.4.

### 3.1 Preliminary

We consider an undirected graph $\mathcal{G} = \{V, E, A\}$ composed of a node set $V$ of cardinality $|V| = N$, an edge set $E$ connecting nodes of cardinality $|E| = M$. $A$ is a real symmetric $N \times N$ matrix that encodes the graph structure, where $a_{i,j} = 1$ if there exists an edge $(i, j)$ between nodes $i$ and $j$, and $a_{i,j} = 0$ otherwise. Graph signal refers to data that reside on the nodes of a graph $\mathcal{G}$, denoted by $X \in \mathbb{R}^{N \times C}$ with the $i$-th row representing the $C$-dimensional graph signal on the $i$-th node of $V$.

### 3.2 Topology Transformation

We define the topology transformation $t$ as adding or removing edges from the original edge set $E$ in graph $\mathcal{G}$. This can be done by sampling, i.i.d., a switch parameter $\sigma_{i,j}$ as in...
which determines whether to modify edge \((i, j)\) in the adjacency matrix. Assuming a Bernoulli distribution \(B(p)\), where \(p\) denotes the probability of each edge being modified, we draw a random matrix \(\Sigma = \{\sigma_{i,j}\} \in \mathbb{R}^{N \times N}\). From an information-theoretic perspective, this transformation produces a transformed graph through the topology transformation \(t\), \(\tilde{A} = t(A)\). Here, the edge perturbation probability of \(p = 0\) corresponds to a non-transformed adjacency matrix, which is a special case of an identity transformation to \(A\).

The transformed adjacency matrix \(\tilde{A}\) can also be written as the sum of the original adjacency matrix \(A\) and a topology perturbation matrix \(\Delta A\):

\[
\tilde{A} = A + \Delta A,
\]

where \(\Delta A = \{\delta a_{i,j}\} \in \mathbb{R}^{N \times N}\) encodes the perturbation of edges, with \(\delta a_{i,j} \in \{-1, 0, 1\}\). As shown in Fig. 1 when \(\delta a_{i,j} = 0\), the edge between node \(i\) and node \(j\) keeps unchanged (i.e., black solid lines); when \(\delta a_{i,j} = 1\) or \(-1\), it means removing (i.e., orange dotted lines) or adding (i.e., blue solid lines) the edge between node \(i\) and node \(j\), respectively.

### 3.3 The Formulation

**Definition 1.** Given a pair of graph signal and adjacency matrix \((X, A)\), and a pair of graph signal and transformed adjacency matrix \((\tilde{X}, \tilde{A})\) by a topology transformation \(t(\cdot)\), a function \(E(\cdot)\) is transformation equivariant if it satisfies

\[
E(X, \tilde{A}) = E(X, t(A)) = \rho(t)[E(X, A)],
\]

where \(\rho(t)\) is a homomorphism of transformation \(t\) in the representation space.

Let us denote

\[
\begin{align*}
H &= E(X, A), \\
\tilde{H} &= E(X, \tilde{A}).
\end{align*}
\]

We seek to learn an encoder \(E : (X, A) \mapsto H, (X, \tilde{A}) \mapsto \tilde{H}\) that maps both the original and transformed samples to representations \(\{H, \tilde{H}\}\) equivariant to the sampled transformation \(t\), whose information can thus be inferred from the representations via a decoder \(D : (H, \tilde{H}) \mapsto \Delta A\) as much as possible. From an information-theoretic perspective, this requires \((H, \Delta A)\) should jointly contain all necessary information about \(\tilde{H}\).

Then a natural choice to formalize the topology transformation equivariance is the mutual information \(I(H, \Delta A; \tilde{H})\) between \((H, \Delta A)\) and \(\tilde{H}\). The larger the mutual information is, the more knowledge about \(\Delta A\) can be inferred from the representations \(\{H, \tilde{H}\}\). Hence, we propose to maximize the mutual information to learn the topology transformation equivariant representations as follows:

\[
\max_\theta I(H, \Delta A; \tilde{H}),
\]

where \(\theta\) denotes the parameters of the auto-encoder network.

Nevertheless, it is difficult to compute the mutual information directly. Instead, we derive that maximizing the mutual information can be relaxed to minimizing the cross entropy, as described in the following theorem.

**Theorem 1.** The maximization of the mutual information \(I(H, \Delta A; \tilde{H})\) can be relaxed to the minimization of the cross entropy \(H(p \parallel q)\) between the probability distributions \(p(\Delta A, H, \tilde{H})\) and \(q_\theta(\Delta A; H, \tilde{H})\):

\[
\min_\theta H(p(\Delta A, H, \tilde{H}) \parallel q_\theta(\Delta A; H, \tilde{H})).
\]

**Proof.** By using the chain rule of mutual information, we have

\[
I(H, \Delta A; \tilde{H}) = I(\Delta A; \tilde{H} | H) + I(H; \tilde{H}) \geq I(\Delta A; \tilde{H} | H).
\]

Thus the mutual information \(I(\Delta A; \tilde{H} | H)\) is the lower bound of the mutual information \(I(H, \Delta A; \tilde{H})\) that attains its minimum value when \(I(H; \tilde{H}) = 0\).

Therefore, we relax the objective to maximizing the lower bound mutual information \(I(\Delta A; \tilde{H} | H)\) between the transformed representation \(\tilde{H}\) and the topology transformation \(\Delta A\):

\[
I(\Delta A; \tilde{H} | H) = H(\Delta A | H) - H(\Delta A | \tilde{H}, H),
\]

where \(H(\cdot)\) denotes the conditional entropy. Since \(\Delta A\) and \(H\) are independent, we have \(H(\Delta A | H) = H(\Delta A)\). Hence, maximizing \(I(\Delta A; \tilde{H} | H)\) becomes

\[
\min_\theta H(\Delta A | \tilde{H}, H).
\]

However, this minimization problem requires to evaluate the posterior probability distribution \(q(\Delta A | H, \tilde{H})\), which
is often difficult to calculate directly. We next introduce a conditional probability distribution \( q_\theta(\Delta \tilde{A}|\tilde{H}, H) \) to approximate the intractable posterior \( q(\Delta \tilde{A}|H, H) \) with an estimated transformation \( \tilde{A} \), i.e.,

\[
H(\Delta \tilde{A}|\tilde{H}, H) = - \sum_{p(\Delta A,H, H)} \log q(\Delta A|\tilde{H}, H)
\]

\[
\geq - \sum_{p(\Delta A, H, H)} \log q_\theta(\Delta A|\tilde{H}, H)
\]

\[
\leq - \sum_{p(\Delta A, H, H)} \log q_\theta(\Delta A|\tilde{H}, H),
\]

where \( D_{KL}(q(\Delta A|\tilde{H}, H) \| q_\theta(\Delta A|\tilde{H}, H)) \) denotes the Kullback-Leibler divergence of \( q(\Delta A|\tilde{H}, H) \) and \( q_\theta(\Delta A|\tilde{H}, H) \) that is non-negative. Thus, the minimization problem in Eq. 6 is converted to minimizing the cross entropy as the upper bound:

\[
\min_{\theta} H(p(\Delta A, \tilde{H}, H) \| q_\theta(\Delta A|\tilde{H}, H)) \triangleq - \sum_{p(\Delta A, H, H)} \log q_\theta(\Delta A|\tilde{H}, H).
\]

Hence, we relax the maximization problem in Eq. 6 to the optimization in Eq. 7.

Based on Theorem 1, we train the decoder \( D \) to learn the distribution \( q_\theta(\Delta A|\tilde{H}, H) \) so as to estimate the topology transformation \( \Delta A \) from the encoded \( \{ \tilde{H}, H \} \), where the input pairs of original and transformed graph representations \( \{ \tilde{H}, H \} \) as well as the ground truth target \( \Delta A \) can be sampled tractably from the factorization of \( p(\Delta A, \tilde{H}, H) \) as \( p(\Delta A)p(\tilde{H}|H)p(H|A, H) \). This allows us to minimize the cross entropy between \( p(\Delta A) \) and \( q_\theta(\Delta A|\tilde{H}, H) \) with the training triplets \( \{ \tilde{H}, H, \Delta A \} \) drawn from the tractable factorization of \( p(\Delta A, \tilde{H}, H) \). Hence, we formulate the Topology Transformation Equivariant Representation learning as the joint optimization of the representation encoder \( E \) and the transformation decoder \( D \).

3.4 Analysis

Since we learn the topology transformation equivariant representations by maximizing the mutual information between representations and transformations as discussed in Sec. 3.3, the proposed model approximately learns equivariant representations for graph data as defined in Eq. 5 via the optimization in Eq. 6. This distinguishes from relevant attempts on equivariant kernels for graph data [59], [60], [61], [62], which design exact equivariant kernels for graph data.

Though equivariance is explicitly satisfied in [59], [60], [61], [62], the equivariant kernels depend on the modalities of the input data, and thus are not generalizable to various tasks. This is because they exploit representational information in different data modalities instead of their commonalities, e.g., 3D coordinates of point clouds [60] and the angles of two neighboring points in meshes [59]. For instance, Fuchs et al. [60] proposed an SE(3)-equivariant convolutional network by constructing an attention-based SE(3)-Transformer specifically for 3D point clouds. Haan et al. [59] presented an anisotropic gauge equivariant kernel for mesh data, which is applied into graph convolutional networks and results in equivalent outputs regardless of the arbitrary choice of kernel orientation. Also, Haan et al. [61] proposed Natural Graph Networks for isomorphic graphs that are equivariant to node permutations. Further, Satorras et al. [62] came up with a new graph convolution kernel, which makes inputs equivariant to parameterized orthogonal transformations (e.g., rotations, translations and reflections) and permutations for data such as molecules. In all, these methods design equivariant network kernels tailored for specific graph data, which have been applied in supervised graph representation learning.

In contrast, we propose an unsupervised model that generalizes to different downstream tasks of various graph data, without restrictions to transformations or types of graph data, while providing good approximations of equivariant representations via the proposed effective optimization.

4 The Proposed Algorithm

We design a graph-convolutional auto-encoder network for the proposed model, as illustrated in Fig. 2. Given a graph signal \( X \) associated with a graph \( G = \{ V, E, A \} \), the proposed unsupervised learning algorithm consists of three steps: 1) topology transformation, which samples and perturbs some edges from all node pairs to acquire a transformed adjacency matrix \( \tilde{A} \); 2) representation encoding, which extracts the feature representations of graph signals before and after the topology transformation; 3) transformation decoding, which estimates the topology transformation parameters from the learned feature representations. We elaborate on the three steps as follows.

4.1 Topology Transformation

We randomly sample a subset of node pairs from all node pairs for topology perturbation—adding or removing edges, which not only enables to characterize local graph structures at various scales, but also reduces the number of edge transformation parameters to estimate for computational efficiency. In practice, in each iteration of training, we sample all the node pairs with connected edges \( S_1 \) (i.e., \( S_1 = E \)) and randomly sample a subset of disconnected node pairs \( S_0 \), i.e.,

\[
S_0 = \{(i, j)|a_{i,j} = 0\}, S_1 = \{(i, j)|a_{i,j} = 1\},
\]

where \( |S_0| = |S_1| = M \). Next, we randomly split \( S_0 \) and \( S_1 \) into two disjoint sets, respectively, i.e.,

\[
S_i = \{S_i^{(1)}, S_i^{(2)}\}, S_i^{(1)} \cap S_i^{(2)} = \emptyset, S_i^{(1)} \cup S_i^{(2)} = S_i,
\]

\[
|S_i^{(1)}| = r \cdot |S_i|, i \in \{0, 1\},
\]

where \( r \) is the edge perturbation rate. Then, for each node pair \( (i, j) \) in \( S_0^{(1)} \) and \( S_1^{(1)} \), we flip the corresponding entry in the original graph adjacency matrix. That is, if \( a_{i,j} = 0 \), then we set \( \tilde{a}_{i,j} = 1 \); otherwise, we set \( \tilde{a}_{i,j} = 0 \). For each node pair \( (i, j) \) in \( S_0^{(2)} \) and \( S_1^{(2)} \), we keep the original connectivities unchanged, i.e., \( \tilde{a}_{i,j} = a_{i,j} \).

This leads to the transformed adjacency matrix \( \tilde{A} \), as well as the sampled transformation parameters by accessing...
\( \Delta A \) at position \((i, j)\) from \( S_0 \) and \( S_1 \). Also, we can categorize the sampled topology transformation parameters into four types:
1) add an edge to a disconnected node pair, i.e., \( \{ t : a_{i,j} = 0 \rightarrow \tilde{a}_{i,j} = 1, (i,j) \in S_0^{(1)} \} \);
2) delete the edge between a connected node pair, i.e., \( \{ t : a_{i,j} = 1 \rightarrow \tilde{a}_{i,j} = 0, (i,j) \in S_0^{(1)} \} \);
3) keep the disconnection between node pairs in \( S_0^{(2)} \), i.e., \( \{ t : a_{i,j} = 0 \rightarrow \tilde{a}_{i,j} = 0, (i,j) \in S_0^{(2)} \} \);
4) keep the connection between node pairs in \( S_1^{(2)} \), i.e., \( \{ t : a_{i,j} = 1 \rightarrow \tilde{a}_{i,j} = 1, (i,j) \in S_1^{(2)} \} \).

Thus, we cast the problem of estimating transformation parameters in \( \Delta A \) from \((\tilde{H}, H)\) as the classification problem of the transformation parameter types. The percentage of these four types is \( r : r' : (1 - r) : (1 - r') \).

### 4.2 Representation Encoder

We train an encoder \( E : (X, A) \rightarrow E(X, A) \) to encode the feature representations of each node in the graph. As demonstrated in Fig. 2, we leverage GCNNs with shared weights to extract feature representations of each node in the graph signal. Taking the GCN [2] as an example, the graph convolution in the GCN is defined as

\[
H = E(X, A) = D^{-\frac{1}{2}}(A + I)D^{-\frac{1}{2}}XW, \tag{9}
\]

where \( D \) is the degree matrix of \( A + I \), \( W \in \mathbb{R}^{C \times F} \) is a learnable parameter matrix, and \( H = [h_1, ..., h_N]^\top \in \mathbb{R}^{N \times F} \) denotes the node-wise feature matrix with \( F \) output channels.

From Eq. (9), we see that node-wise representations from GCNN are updated in two steps: 1) feature propagation and aggregation, and 2) linear transformation.

The first step aims to aggregate the features of each node \( v_i \) and its local neighborhood, e.g., 1-hop neighborhood,

\[
\tilde{h}_i = \frac{1}{d_i}x_i + \sum_{i \sim j} \frac{a_{i,j}}{\sqrt{d_i \cdot d_j}}x_j, \tag{10}
\]

where \( i \sim j \) represents node \( i \) and \( j \) are connected, and \( \tilde{h}_i \) denotes the aggregated features of node \( v_i \). Eq. (10) can also be expressed over the entire graph by matrix multiplication, i.e., \( \tilde{H} = D^{-\frac{1}{2}}(A + I)D^{-\frac{1}{2}}X \).

The second step performs a linear transformation with a learnable parameter matrix \( W \) on the aggregated features \( \tilde{H} \) to generate node embeddings for the GCN layer, i.e., \( H = \tilde{H}W \).

Similarly, the node feature of the transformed counterpart is as follows with the shared weights \( W \).

\[
\tilde{H} = E(X, \tilde{A}) = \tilde{D}^{-\frac{1}{2}}(\tilde{A} + I)\tilde{D}^{-\frac{1}{2}}XW \\
= \tilde{D}^{-\frac{1}{2}}(A + I)\tilde{D}^{-\frac{1}{2}}XW + \tilde{D}^{-\frac{1}{2}}\Delta AD^{-\frac{1}{2}}XW. \tag{11}
\]

We thus acquire the feature representations \( H \) and \( \tilde{H} \) of graph signals before and after topology transformations.

### 4.3 Transformation Decoder

Comparing Eq. (9) and Eq. (11), the prominent difference between \( \tilde{H} \) and \( H \) lies in the second term of Eq. (11) featuring \( \Delta A \). This enables us to train a decoder \( D : (\tilde{H}, H) \rightarrow \Delta A \) to estimate the topology transformation from the joint representations before and after transformation. We first take the difference between the extracted feature representations before and after transformations along the feature channel,

\[
\Delta H = \tilde{H} - H = [\delta h_1, ..., \delta h_N]^\top \in \mathbb{R}^{N \times F}. \tag{12}
\]

Thus, we can predict the topology transformation between node \( i \) and node \( j \) through the node-wise feature difference \( \Delta H \) by constructing the edge representation as

\[
e_{i,j} = \frac{\exp\{-(\delta h_i - \delta h_j) \odot (\delta h_i - \delta h_j)\}}{\| \exp\{-(\delta h_i - \delta h_j) \odot (\delta h_i - \delta h_j)\} \|_1}, \tag{13}
\]

\[
\forall (i,j) \in S_0 \cup S_1,
\]

where \( \odot \) denotes the Hadamard product of two vectors to capture the feature representation, and \( \| \cdot \|_1 \) is the \( \ell_1 \)-norm of a vector for normalization. The edge representation \( e_{i,j} \) of node \( i \) and \( j \) is then fed into several linear layers for the prediction of the topology transformation,

\[
\tilde{y}_{i,j} = \text{softmax}(\text{linear}(e_{i,j})), \quad \forall (i,j) \in S_0 \cup S_1, \tag{14}
\]

where \( \text{softmax}(\cdot) \) is an activation function.

According to Eq. (5), the entire auto-encoder network is trained by minimizing the cross entropy

\[
\mathcal{L} = -\mathbb{E}_{(i,j) \in S_0 \cup S_1} \sum_{f=1}^{4} y_{i,j}^{(f)} \log \tilde{y}_{i,j}^{(f)}, \tag{15}
\]

where \( f \) denotes the transformation type \( (f \in \{1, 2, 3, 4\}) \), and \( y \) is the ground-truth binary indicator \((0 \text{ or } 1)\) for each transformation parameter type.

### 5 Experiments

In this section, we evaluate the proposed model on three representative downstream tasks: node classification, graph classification, and link prediction.
5.1 Node Classification

5.1.1 Datasets
We adopt three citation networks to evaluate our model: Cora, Citeseer, and Pubmed. The dataset statistics are reported in Tab. 2. The three datasets contain sparse bag-of-words feature vectors for each document and a list of citation links between documents. We treat documents as nodes, and the citation links as (undirected) edges, leading to a binary and symmetric adjacency matrix $A$ as in Eq. (8). We follow the standard train/test split in [26] to conduct the experiments, where the label rate denotes the number of labeled nodes that are used for training.

5.1.2 Implementation Details
In this task, the auto-encoder network is trained via Adam optimizer, and the learning rate is set to $10^{-4}$. We use the same early stopping strategy as DGI [19] on the observed training loss, with a patience of 20 epochs. We deploy one Simple Graph Convolution (SGC) layer [65] as our encoder, and the order of the adjacency matrix is set to 2. The LeakyReLU activation function with a negative slope of 0.1 is employed after the SGC layer. Similar to DGI [19], we set the output channel $F = 512$ for Cora and Citeseer dataset, and 256 for Pubmed dataset due to memory limitations. After the encoder, we use one linear layer to classify the transformation types. We set the edge perturbation rate in Eq. (6) as $r = \{0.7, 0.4, 0.7\}$ for Cora, Citeseer, and Pubmed, respectively.

During the training procedure of the classifier, the SGC layer in the encoder is used to extract graph feature representations with the weights frozen. After the SGC layer, we apply one linear layer to map the features to the classification scores.

5.1.3 Experimental Results
We compare the proposed method with five unsupervised methods, including one node embedding method DeepWalk, two graph auto-encoders GAE and VGAE [15], and two contrastive learning methods DGI [19] and GMI [20]. Additionally, we report the results of Raw Features and DeepWalk+Features [68] under the same settings. For fair comparison, the results of all other unsupervised methods are reproduced by using the same encoder architecture of the proposed model except DeepWalk and Raw Features. We report the mean classification accuracy (with standard deviation) on the test nodes for all methods after 50 runs of training. As reported in Tab. 1, the proposed method outperforms all other competing unsupervised methods on three datasets. Further, the proposed unsupervised method also achieves comparable performance with semi-supervised methods. This significantly closes the gap between unsupervised approaches and the semi-supervised methods.

5.1.4 Experiments on Different Orders of The Adjacency Matrix
As presented in Sec. 3.2, we perturb the 1-hop neighborhoods via the proposed topology transformations, leading to possibly significant changes in the graph topology. This increases the difficulties of predicting the topology transformations when using one-layer GCN [2] by aggregating the 1-hop neighborhood information. Therefore, we employ one Simple Graph Convolution (SGC) layer [65] with order $k$ as our encoder $E(\cdot)$, where the output feature representations
aggregate multi-hop neighborhood information. Formally, the SGC layer is defined as

$$H = E(X, A) = \left(D^{-\frac{1}{2}}(A + I)D^{-\frac{1}{2}}\right)^k X W,$$  \quad (16)$$

where $D$ is the degree matrix of $A + I$, $W \in \mathbb{R}^{C \times F}$ is a learnable parameter matrix, and $k$ is the order of the normalized adjacency matrix.

To study the influence of different orders of the adjacency matrix, we adopt five orders from 1 to 5 to train five models on the node classification task. Fig. 3 presents the node classification accuracy under different orders of the adjacency matrix for the proposed method and DGI respectively. As we can see, the proposed method achieves best classification performance when $k = 1$, our model still achieves reasonable results on the three datasets; when the number of SGC layers increases, the performance of our model decreases quickly due to overfitting. This is consistent with the conclusion in [2] that overfitting can become an issue as the number of parameters increases with model depth in graph convolutional neural networks.

### 5.1.5 Experiments on Different Numbers of SGC Layers

We conduct ablation studies on the number of SGC layers to study the influence of deeper encoder architecture on the node classification task. In this experiment, we adopt an encoder with five numbers of SGC layers ranging from 1 to 5 to train five models on the Cora, Citeseer, and Pubmed datasets. To prevent memory issues during the training, the channel of each hidden layer is set to 64, and the output channel is 512. Fig. 4 presents the node classification accuracy under different numbers of SGC layers for the proposed method. We see that, when the number of layers is small (e.g., 1 or 2), our model achieves reasonable results on the three datasets; when the number of SGC layers increases, the performance of our model decreases quickly due to overfitting.

### 5.1.6 Robustness Test

To evaluate the robustness of our model on the node classification task, we jitter the original node features with an additive noise model, namely,

$$\tilde{X} = X + E,$$  \quad (17)$$

where $X \in \mathbb{R}^{N \times F}$ is the original node features, $E \in \mathbb{R}^{N \times F}$ is a random matrix which is sampled from a random distribution (e.g., Gaussian or Laplace), and $\tilde{X}$ denotes the noise-corrupted node features.

Specifically, we select the zero-mean Gaussian noise with a range of standard deviation $\sigma$ from 0.01 to 0.10 at an interval of 0.01, as well as the zero-location Laplace noise with a range of scale parameter $s$ from 0.01 to 0.10 at an interval of 0.01, for extensive classification performance comparison. We employ one SGC layer as the encoder $E(\cdot)$ of the proposed model and two representative self-supervised model DGI and GAE, where the order of the adjacency matrix is set to 1. We use the original node features $X$ to train the three models in the unsupervised training stage and the linear classifier in the supervised evaluation stage, and employ the corrupted node features $\tilde{X}$ to evaluate the classification accuracies.

The classification performance under Gaussian and Laplace noises are presented in Fig. 5 and Fig. 6 respectively. When the noise level is low, our model outperforms GAE by a large margin on the Cora and Citeseer datasets.
datasets, and achieves comparable results to DGI on the three datasets. When the noise level is high, our model significantly outperforms GAE and DGI on the three datasets. This is because DGI takes the original graph topology to aggregate node features. When the original node features are seriously corrupted (the noise level is high), the aggregated features will change significantly. GAE reconstructs the adjacency matrix from the feature representations of individual nodes. The high dependency of the graph topology and node features leads to bad performance when the node features suffer from serious noise perturbations. In contrast, our model aims to predict the topology transformations from the feature representations of nodes before and after transformation, which not only employs the original graph topology information, but also explores how node features would change by applying a topology transformation, thus enhancing the robustness.

5.1.7 Experiments On Different Edge Perturbation Rates

Further, we evaluate the influence of the edge perturbation rate in Eq. (8) on the node classification task. We choose 11 edge perturbation rates from 0.0 to 1.0 at an interval of 0.1 to train the proposed model. We use one SGC layer as our encoder $E(\cdot)$, where the order of the adjacency matrix is set to 1. As presented in Fig. 7, the blue solid line with error bars shows the classification accuracy of our method under different edge perturbation rates. We also provide the classification accuracy on feature representations of graphs from a randomly initialized encoder $E(\cdot)$, denoted as Random Init., which serves as the lower bound of the performance.

As we can see, the classification performance reaches the best when the graph is perturbed under a reasonable edge perturbation rate, e.g., $r = \{0.6, 0.5, 0.6 \}$ for the Cora, Citeseer, and Pubmed dataset, respectively. When the edge perturbation rate $r = 0.0$, the unsupervised training task of the proposed model becomes link prediction, which cannot take advantage of the proposed method by predicting the topology transformations; when the edge perturbation rate $r = 1.0$, our model still achieves reasonable classification results, which shows the stability of our model under high edge perturbation rates. At the same time, we observe that the proposed method outperforms Random Init. by a large margin, which validates the effectiveness of the proposed unsupervised training strategy.

5.2 Graph Classification

5.2.1 Datasets

We conduct graph classification on six well-known graph benchmark datasets [70], including two molecule datasets MUTAG and PTC, and four social network datasets REDDIT-BINARY, REDDIT-MULTI-5K, IMDB-BINARY, and IMDB-MULTI. In the two molecule datasets, graphs are molecules, where nodes represent atoms and edges represent chemical bonds, and the graph classification task is to classify the molecules. In the REDDIT dataset, a graph denotes a discussion thread, where nodes correspond to users, two of which are connected by an edge if one responded to a comment of the other. The graph classification task is to distinguish whether the subreddits is discussion-based or question/answer-based (REDDIT-BINARY), or predict the subreddit (REDDIT-MULTI-5K). The IMDB dataset consists of ego-networks derived from actor collaborations, and the graph classification task is to predict the genre, e.g., Action or Romance.

5.2.2 Implementation Details

In this task, the entire network is trained via Adam optimizer with a batch size of 64, and the learning rate is set to $10^{-3}$. For the encoder architecture, we follow the same encoder settings in the released code of InfoGraph [21], i.e., three Graph Isomorphism Network (GIN) layers [3], with batch normalization. We also use one linear layer to classify the transformation types. We set the sampling rate $r = 0.5$ for all datasets.

| Dataset | MUTAG | PTC-MR | RDT-B | RDT-M5K | IMDB-B | IMDB-M |
|---------|-------|--------|-------|---------|--------|--------|
| (#Graphs) | 188    | 344    | 2000  | 4999    | 1000   | 1500   |
| (#Classes) | 2      | 2      | 2     | 2       | 2      | 3      |
| (Avg. #Nodes) | 17.93  | 14.29  | 429.63| 508.52  | 19.77  | 13.00  |

**Table 4**

Graph classification accuracies (with standard deviation) in percentage on 6 datasets. *">1 Day" represents that the computation exceeds 24 hours. "OOM" is out of memory error.
During the evaluation stage, the entire encoder will be frozen to extract node-level feature representations, which will go through a global add pooling layer to acquire global features. We then use LIBSVM to classify these global features to classification scores. We adopt the same procedure of previous works [21] to make a fair comparison and use 10-fold cross validation accuracy to report the classification performance, and the experiments are repeated five times.

5.2.3 Experimental Results

We take six graph kernel approaches for comparison: Random Walk (RW) [71], Shortest Path Kernel (SP) [72], Graphlet Kernel (GK) [73], Weisfeiler-Lehman Sub-tree Kernel (WL) [74], Deep Graph Kernels (DGK) [70], and Multi-Scale Laplacian Kernel (MLG) [75]. Aside from graph kernel methods, we also compare with three unsupervised graph-level representation learning methods: node2vec [76], sub2vec [77], and graph2vec [78], and one contrastive learning method: InfoGraph [21]. The experimental results of unsupervised graph classification are presented in Tab. 4. The proposed method outperforms all unsupervised baseline methods on the first five datasets, and achieves comparable results on the other dataset. Also, the proposed approach reaches the performance of supervised methods at times, thus validating the superiority of the proposed method.
5.3 Link Prediction on Static Graphs

5.3.1 Implementation Details

Also, we evaluate our model on link prediction over the three citation network datasets Cora, Citeseer, and Pubmed. We follow the same experimental settings in [15], where 85% of citation links are used for training, and 10% and 5% of citation links are for testing and validating respectively. We employ the same number of randomly sampled pairs of disconnected nodes for testing.

We deploy two GCN [2] layers with 32 hidden channels and 16 output channels as the encoder as in [15]. The ReLU activation function is employed between the two GCN layers. Subsequent to the encoder, we also use one linear layer to classify the transformation types.

In this experiment, we demonstrate the benefits of pre-training with our method before training on link prediction. During the training stage, we first train our model by minimizing Eq. (15) on the training set. After that, we replace the transformation decoder with an inner product decoder to acquire the reconstructed adjacency matrix \( \hat{A} \), i.e.,

\[
\hat{A} = \text{sigmoid}(HH^\top), \quad \text{where } H = E(X, A),
\]

where \( A \) is the original adjacency matrix, and \( \text{sigmoid}(\cdot) \) is an activation function. The encoder is thus fine-tuned by minimizing the reconstruction error between \( A \) and \( \hat{A} \).

During the evaluation stage, we predict the edges in the testing set from the feature representations \( H \), and report area under the ROC curve (AUC) and average precision (AP) scores for each dataset.

5.3.2 Experimental Results

We take five approaches for comparison: Spectral Clustering [79], DeepWalk [68], GAE [15], VGAE [15], and CensNet-VAE [40]. We present the mean AUC and AP (with standard deviation) after 10 runs of training with random initialization on fixed dataset splits. As reported in Tab. 5 the proposed method outperforms all AE-based methods on the three datasets. This is because our method makes better use of the topology information of graphs by decoding the topology perturbation, thereby revealing not only static visual structures but also how they would change by applying different topology transformations. This again validates the effectiveness of our method.

5.4 Link Prediction on Temporal Graphs

5.4.1 Implementation Details

In order to validate the intuition of the benefit of our model for graph data changing over time, we further perform link prediction over temporal graph datasets. Since most existing methods focus on static graphs, few methods have been proposed to study link prediction on dynamic graphs. For example, DeepWalk [68] generates node representations by randomly walking on static graphs; graph auto-encoders [15] learn feature representations of graphs by decoding the input static graphs. Hence, we propose an experimental setup to make graph auto-encoder methods adaptable to link prediction on temporal graphs.

Considering two consecutive graphs \( G_{t-1} = \{V_{t-1}, \mathcal{E}_{t-1}, A_{t-1}\} \) and \( G_t = \{V_t, \mathcal{E}_t, A_t\} \) from a temporal graph sequence \( \{G_t\}_{t=1}^T \) with \( T \) graphs, our goal is to predict the edges in \( G_t \) given \( G_{t-1} \). For the graph auto-encoder method, we feed the adjacency matrix \( A_{t-1} \) and its corresponding node features \( X_{t-1} \) to the encoder \( E(\cdot) \) to learn the feature representations \( H_{t-1} \) at time \( t-1 \), and reconstruct the adjacency \( \hat{A}_t \) at time \( t \) through an inner product decoder, i.e.,

\[
\hat{A}_t = \text{sigmoid}(H_{t-1}H_{t-1}^\top),
\]

where \( H_{t-1} = E(X_{t-1}, A_{t-1}) \), \( \text{sigmoid}(\cdot) \) is an activation function. For our model, we treat \( A_{t-1} \) and \( A_t \) as the original and transformed graphs, respectively. The topology transformation is acquired by simply calculating \( \Delta A = A_t - A_{t-1} \). Since these datasets have no initial node features, we set all node features as 1 to make our model mainly learn feature representations from graph structures, i.e., \( X_{t-1} = X_0 = 1_{N \times 1} \). We thus feed the original and transformed graphs \( \{A_{t-1}, A_t\} \) to estimate the topology transformation \( \Delta A \). After the training of the graph auto-encoder model and our model, we feed the adjacency matrix \( A_{t-1} \) to the encoder with the frozen weights to acquire the feature representation \( H_{t-1} \), and predict the edges in \( A_t \) through an inner product decoder.

In this experiment, we evaluate our model and two baseline methods GAE and VGAE [15] on link prediction over three real-world temporal graph datasets [20]: Sparrow Social, Ant Colony, and Wildbird Network. We deploy two GCN [2] layers with 32 hidden channels and 256 output channels as the encoder. Subsequent to the encoder, we also use one linear layer to classify the transformation types. During the evaluation stage, we predict the edges in the testing set from the feature representations \( H_{t-1} \), and report area under the ROC curve (AUC) and average precision (AP) scores for each dataset.

5.4.2 Experimental Results

We present the mean AUC and AP (with standard deviation) after 10 runs of training with random initialization.
on fixed dataset splits. As reported in Tab. 6, the proposed method outperforms the two baseline methods on the three datasets. This verifies the effectiveness of our method on temporal graphs.

5.5 Visualization of the TopoTER Learning

We visualize the node features \( \{ \mathbf{H}, \hat{\mathbf{H}} \} \) of the original graph \( \mathbf{A} \) and transformed graph \( \hat{\mathbf{A}} \) to observe whether the learned feature representations meet the definition of the proposed topology transformation equivariance in Eq. (5). If our proposed model learns topology transformation equivariant representations, then the feature representation \( \Delta \mathbf{H} = \mathbf{D}^{-\frac{1}{2}} \Delta \mathbf{A} \mathbf{D}^{-\frac{1}{2}} \mathbf{XW} \) based on the estimated \( \Delta \mathbf{A} \) by our model satisfies \( \mathbf{H} + \Delta \mathbf{H} \approx \hat{\mathbf{H}} \) according to Eq. (11).

In the experiment, we adopt the real-scenario temporal graph dataset Sparrow Social [80] to evaluate our model. Since the dataset only contains graphs at two timestamps, we treat the first graph as the original graph, denoted as \( \mathbf{A} = \mathbf{A}_{t-1} \), and the second graph as the transformed graph over the original graph, denoted as \( \hat{\mathbf{A}} = \mathbf{A}_t \). We use a one-hot encoding of node ID as the initial node features. We then feed the original and transformed graphs into the proposed model to train our TopoTER for the estimation of the topology transformation \( \Delta \mathbf{A} \). Having trained our model, we feed both the original and transformed graphs with the same node features to the model to acquire the feature representations \( \mathbf{H} = \mathbf{H}_{t-1} \) and \( \hat{\mathbf{H}} = \mathbf{H}_t \) respectively. We further feed the estimated topology transformation \( \Delta \mathbf{A} \) and the node features \( \mathbf{X} \) to our model to infer the changed node features \( \Delta \mathbf{H} \).

Fig. 8 presents the heatmaps of the learned node features. On the one hand, Figs. 8(a) and 8(b) demonstrate that the difference between the node features \( \mathbf{H}_{t-1} \) of the original graph and \( \mathbf{H}_t \) of the transformed graph is dominant due to the transformation. On the other hand, Figs. 8(c) and 8(d) show that the difference between the node features \( \mathbf{H}_t + \Delta \mathbf{H} \) is trivial, which validates \( \mathbf{H}_{t-1} + \Delta \mathbf{H} \approx \mathbf{H}_t \) discussed above. This verifies that our model learns topology transformation equivariant representations.

6 Conclusion

We propose a self-supervised paradigm of Topology Transformation Equivariant Representation for graph representation learning. By maximizing the mutual information between topology transformations and feature representations before and after transformations, the proposed method enforces the encoder to learn intrinsic graph feature representations that contain sufficient information about structures under applied topology transformations. We apply our model to node classification, graph classification and link prediction tasks, and results demonstrate that the proposed method outperforms state-of-the-art unsupervised approaches and reaches the performance of supervised methods at times. We believe this model will have impact on applications such as social analysis, molecule property prediction, as well as applications in 3D computer vision.

REFERENCES

[1] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, “Geometric deep learning: going beyond euclidean data,” IEEE Signal Processing Magazine, vol. 34, no. 4, pp. 18–42, 2017.
[2] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in International Conference on Learning Representations (ICLR), 2017.
[3] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, “Graph attention networks,” in International Conference on Learning Representations (ICLR), 2018.
[4] B. Xu, H. Shen, Q. Cao, Y. Qiu, and X. Cheng, “Graph wavelet neural network,” in International Conference on Learning Representations (ICLR), 2019.
[5] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, “How powerful are graph neural networks?” in International Conference on Learning Representations (ICLR), 2019.
[6] A. Kolesnikov, X. Zhai, and L. Beyer, “Revisiting self-supervised visual representation learning,” in International Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 1920–1929.
[7] X. Liu, F. Zhang, Z. Hou, Z. Wang, L. Mian, J. Zhang, and J. Tang, “Self-supervised learning: Generative or contrastive,” arXiv, pp. arXiv–2006, 2020.
[8] A. Van Oord, N. Kalchbrenner, and K. Kavukcuoglu, “Pixel recurrent neural networks,” in International Conference on Machine Learning (ICML). PMLR, 2016, pp. 1747–1756.
[9] A. van den Oord, N. Kalchbrenner, L. Espeholt, K. Kavukcuoglu, O. Vinyals, and A. Graves, “Conditional image generation with pixelcnn decoders,” in Neural Information Processing Systems (NIPS), 2016, pp. 4790–4798.
[10] J. You, R. Ying, X. Ren, W. L. Hamilton, and J. Leskovec, “Graphrnn: Generating realistic graphs with deep auto-regressive models,” in International Conference on Machine Learning (ICML), 2018, pp. 5694–5703.
[11] L. Dinh, D. Krueger, and Y. Bengio, “Nice: Non-linear independent components estimation,” arXiv preprint arXiv:1410.8516, 2014.
[12] L. Dinh, J. Sohl-Dickstein, and S. Bengio, “Density estimation using real nvp,” arXiv preprint arXiv:1605.08803, 2016.
[13] D. H. Ballard, “Modular learning in neural networks,” in AAAI Conference on Artificial Intelligence (AAAI), 1987, pp. 279–284.
[14] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” in International Conference on Learning Representations (ICLR), 2014.
[15] T. N. Kipf and M. Welling, “Variational graph auto-encoders,” in Neural Information Processing Systems (NIPS) Workshop on Bayesian Deep Learning, 2016.
[16] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in Neural Information Processing Systems (NIPS), 2014, pp. 2672–2680.
[17] H. Wang, J. Wang, J. Wang, M. Zhao, W. Zhang, F. Zhang, X. Xie, and M. Guo, “Graphgan: Graph representation learning with generative adversarial nets,” in AAAI Conference on Artificial Intelligence (AAAI), vol. 32, no. 1, 2018.
[18] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio, “Learning deep representations by mutual information estimation and maximization,” in International Conference on Learning Representations (ICLR), 2018.

| Method | Sparrow Social | Ant Colony | Wildbird Network |
|--------|---------------|------------|-----------------|
|       | AUC           | AP         | AUC             | AP             | AUC             | AP             |
| GAE   | 56.5 ± 0.54   | 52.1 ± 0.26| 59.1 ± 0.87     | 51.9 ± 3.27    | 52.8 ± 0.30     | 47.9 ± 0.33    |
| VGAE  | 55.0 ± 0.17   | 52.1 ± 0.26| 66.0 ± 0.29     | 56.0 ± 0.44    | 52.3 ± 0.25     | 47.5 ± 0.38    |
| Ours  | 67.7 ± 4.19   | 59.6 ± 2.51| 70.0 ± 0.30     | 58.5 ± 0.30    | 60.8 ± 4.19     | 54.9 ± 2.68    |
Fig. 8. Visualization of node features on the real-scenario temporal graph dataset Sparrow Social [80]. a) The node features learned from the original graph; b) The node features learned from the transformed graph; c) The changed node features learned from the estimated topology transformation; d) The node features of $H_{t-1} + \hat{\Delta}H$. Note that we only visualize the added edges in $\hat{\Delta}A$ in (c) for easy observation, while the unchanged and removed edges are ignored.

[19] P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, “Deep graph infomax,” in International Conference on Learning Representations (ICLR), 2019.

[20] Z. Peng, W. Huang, M. Luo, Q. Zheng, Y. Rong, T. Xu, and J. Huang, “Graph representation learning via graphical mutual information maximization,” in WWW, 2020, pp. 259–270.

[21] F.-Y. Sun, J. Hoffmann, V. Verma, and J. Tang, “Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization,” in International Conference on Learning Representations (ICLR), 2020.

[22] G.-J. Qi, L. Zhang, and X. Wang, “Learning generalized transformation equivariant representations via autoencoding transformations,” arXiv preprint arXiv:1906.08629, 2019.

[23] G. E. Hinton, A. Krizhevsky, and S. D. Wang, “Transforming auto-encoders,” in International Conference on Artificial Neural Networks (ICANN). Springer, 2011, pp. 44–51.

[24] J. J. Kivinen and C. K. Williams, “Transformation equivariant boltzmann machines,” in International Conference on Artificial Neural Networks (ICANN). Springer, 2011, pp. 1–9.

[25] K. Sohn and H. Lee, “Learning invariant representations with local transformations,” in International Conference on Machine Learning (ICML), 2012, pp. 1339–1346.

[26] U. Schmidt and S. Roth, “Learning rotation-aware features: From invariant priors to equivariant descriptors,” in International Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2012, pp. 2050–2057.

[27] H. Skibbe, “Spherical tensor algebra for biomedical image analysis,” Ph.D. dissertation, Verlag nicht ermittelbar, 2013.

[28] R. Gens and P. M. Domingos, “Deep symmetry networks,” in Neural Information Processing Systems (NIPS), 2014, pp. 2537–2545.

[29] K. Lenc and A. Vedaldi, “Understanding image representations by measuring their equivariance and equivalence,” in International Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 991–999.

[30] S. Dieleman, K. W. Willett, and J. Dambre, “Rotation-invariant convolutional neural networks for galaxy morphology prediction,” Monthly Notices of the Royal Astronomical Society, vol. 450, no. 2, pp. 1441–1459, 2015.

[31] S. Dieleman, J. De Fauw, and K. Kavukcuoglu, “Exploiting cyclic symmetry in convolutional neural networks,” in International Conference on Machine Learning (ICML), 2016, pp. 1889–1898.

[32] L. Zhang, G.-J. Qi, L. Wang, and J. Luo, “AET vs. AED: Unsupervised representation learning by auto-encoding transformations rather than data,” in International Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 2547–2555.

[33] G.-J. Qi, L. Zhang, C. W. Chen, and Q. Tian, “AVT: Unsupervised learning of transformation equivariant representations by auto-encoding variational transformations,” in International Conference on Computer Vision (ICCV), 2019.

[34] J. Wang, W. Zhou, G.-J. Qi, Z. Fu, Q. Tian, and H. Li, “Transformation gan for unsupervised image synthesis and representation learning,” in International Conference on Computer Vision and Pattern
