# GraphTER: Unsupervised Learning of Graph Transformation Equivariant Representations via Auto-Encoding Node-wise Transformations

Xiang Gao[1], Wei Hu[1,*], and Guo-Jun Qi[2]
[1]Wangxuan Institute of Computer Technology, Peking University, Beijing
[2]Futurewei Technologies
{gyshgx868, forhuwei}@pku.edu.cn, guojunq@gmail.com

## Abstract

*Recent advances in Graph Convolutional Neural Networks (GCNNs) have shown their efficiency for non-Euclidean data on graphs, which often require a large amount of labeled data with high cost. It it thus critical to learn graph feature representations in an unsupervised manner in practice. To this end, we propose a novel unsupervised learning of Graph Transformation Equivariant Representations (GraphTER), aiming to capture intrinsic patterns of graph structure under both global and local transformations. Specifically, we allow to sample different groups of nodes from a graph and then transform them node-wise isotropically or anisotropically. Then, we self-train a representation encoder to capture the graph structures by reconstructing these node-wise transformations from the feature representations of the original and transformed graphs. In experiments, we apply the learned GraphTER to graphs of 3D point cloud data, and results on point cloud segmentation/classification show that GraphTER significantly outperforms state-of-the-art unsupervised approaches and pushes greatly closer towards the upper bound set by the fully supervised counterparts. The code is available at: https://github.com/gyshgx868/graph-ter.*

## 1. Introduction

Graphs are a natural representation of irregular data, such as 3D geometric points, social networks, citation networks and brain networks. Recent advances in Graph Convolutional Neural Networks (GCNNs) have shown their efficiency in learning representations of such data [4, 54, 53, 46], which generalize the celebrated CNN models. Existing GCNNs are mostly trained in a supervised or semi-supervised fashion, requiring a large amount of labeled data to learn graph representations. This prevents the wide applicability of GCNNs due to the high labeling cost especially for large-scale graphs in many real scenarios. Hence, it is demanded to train graph feature representations in an unsupervised fashion, which can adapt to downstream learning tasks on graphs.

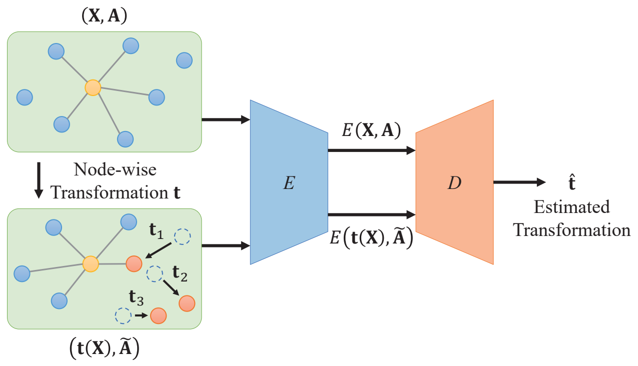

$(\mathbf{X}, \mathbf{A})$
Node-wise Transformation $\mathbf{t}$
$\mathbf{t}_1$
$\mathbf{t}_2$
$\mathbf{t}_3$
$(\mathbf{t}(\mathbf{X}), \tilde{\mathbf{A}})$
$E$
$E(\mathbf{X}, \mathbf{A})$
$E(\mathbf{t}(\mathbf{X}), \tilde{\mathbf{A}})$
$D$
$\hat{\mathbf{t}}$
Estimated Transformation

Figure 1. **An illustration of the proposed GraphTER model for unsupervised feature learning.** The encoder learns representations of the original graph data $\mathbf{X}$ associated with adjacency matrix $\mathbf{A}$ and its transformed counterpart $\mathbf{t}(\mathbf{X})$ associated with $\tilde{\mathbf{A}}$ respectively. By decoding node-wise transformations $\mathbf{t}$ from both representations, the auto-encoder is able to learn intrinsically morphable structures of graphs.

Auto-Encoders (AEs) and Generative Adversarial Networks (GANs) are two most representative methods for unsupervised learning. Auto-encoders aim to train an encoder to learn feature representations by reconstructing input data via a decoder [42, 36, 16, 18]. Most of auto-encoders stick to the idea of reconstructing *input data* at the output end (*e.g.*, images [16], graphs [19], 3D point clouds [48]), and thus can be classified into the Auto-Encoding Data (AED) [51]. In contrast, GANs [14, 10, 11, 50, 26, 7, 3] extract feature representations in an unsupervised fashion by generating data from input noises via a pair of generator and discriminator, where the noises are viewed as the data representations, and the generator is trained to generate data from the "noise" feature representations.

Based on AEs and GANs, many approaches have sought

*Corresponding author: Wei Hu (forhuwei@pku.edu.cn). This work was supported by National Natural Science Foundation of China [61972009] and Beijing Natural Science Foundation [4194080].

to learn *transformation equivariant representations* (TER) to further improve the quality of unsupervised representation learning. It assumes that 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 [34]. Learning transformation equivariant representations has been advocated in Hinton's seminal work on learning transformation capsules [16]. Following this, a variety of approaches have been proposed to learn transformation equivariant representations [20, 41, 38, 40, 23, 12, 9, 8]. Among them are the group equivariant convolutions [6] and group equivariant capsule networks [24] that generalize the CNNs and capsule nets to equivary to various transformations. However, these models are restricted to discrete transformations, and they should be trained in a supervised fashion. This limits their ability to learn unsupervised representations equivariant to a generic composition of continuous transformations [34].

To generalize to generic transformations, Zhang *et al.* [51] propose to learn unsupervised feature representations via Auto-Encoding Transformations (AET) rather than AED. By randomly transforming images, they seek to train auto-encoders by directly reconstructing these transformations from the learned representations of both the original and transformed images. A variational AET [35] is also introduced from an information-theoretic perspective by maximizing the lower bound of mutual information between transformations and representations. Moreover, it has also been theoretically proved [34, 35] that the unsupervised representations by the AET are equivariant to the applied transformations. Unfortunately, these works focus on transformation equivariant representation learning of images that are Euclidean data, which cannot be extended to graphs due to the irregular data structures.

In this paper, we take a first step towards this goal – we formalize Graph Transformation Equivariant Representation (GraphTER) learning by auto-encoding node-wise transformations in an unsupervised manner. The proposed method is novel in twofold aspects. On one hand, we define *graph signal transformations* and present a graph-based auto-encoder architecture, which encodes the representations of the original and transformed graphs so that the graph transformations can be reconstructed from both representations. On the other hand, in contrast to the AET where global spatial transformations are applied to the *entire* input image, we perform node-wise transformations on graphs, where each node can have its own transformation. Representations of individual nodes are thus learned by decoding node-wise transformations to reveal the graph structures around it. These representations will not only capture the local graph structures under node-wise transformations, but also reveal global information about the graph as we randomly sample nodes into different groups over training iterations. Different groups of nodes model different parts of graphs, allowing the learned representations to capture various scales of graph structures under isotropic and/or anisotropic node-wise transformations. This results in transformation equivariant representations to characterize the *intrinsically* morphable structures of graphs.

Specifically, given an input graph signal and its associated graph, we sample a subset of nodes from the graph (globally or locally) and perform transformations on individual nodes, either isotropically or anisotropically. Then we design a full graph-convolution auto-encoder architecture, where the encoder learns the representations of individual nodes in the original and transformed graphs respectively, and the decoder predicts the applied node-wise transformations from both representations. Experimental results demonstrate that the learned GraphTER significantly outperforms the state-of-the-art unsupervised models, and achieves comparable results to the fully supervised approaches in the 3D point cloud classification and segmentation tasks.

Our main contributions are summarized as follows.

- We are the first to propose Graph Transformation Equivariant Representation (GraphTER) learning to extract adequate graph signal feature representations in an unsupervised fashion.
- We define generic graph transformations and formalize the GraphTER to learn feature representations of graphs by decoding node-wise transformations end-to-end in a full graph-convolution auto-encoder architecture.
- Experiments demonstrate the GraphTER model outperforms the state-of-the-art methods in unsupervised graph feature learning as well as greatly pushes closer to the upper bounded performance by the fully supervised counterparts.

The remainder of this paper is organized as follows. We first review related works in Sec. 2. Then we define graph transformations in Sec. 3 and formalize the GraphTER model in Sec. 4. Finally, experimental results and conclusions are presented in Sec. 5 and Sec. 6, respectively.

## 2. Related Works

**Transformation Equivariant Representations.** Many approaches have been proposed to learn equivariant representations, including transforming auto-encoders [16], equivariant Boltzmann machines [20, 41], equivariant descriptors [38], and equivariant filtering [40]. Lenc et al. [23] prove that the AlexNet [22] trained on ImageNet learns representations that are equivariant to flip, scaling and rotation transformations. Gens *et al.* [12] propose an approximately equivariant convolutional architecture, which utilizes sparse and high-dimensional feature maps to deal with groups of transformations. Dieleman *et al.* [9] show that rotation symmetry can be exploited in convolutional networks for effectively learning an equivariant representation. This work is later extended in [8] to evaluate on other computer vision tasks that have cyclic symmetry. Cohen *et al.* [6] propose group equivariant convolutions that have been developed to equivary to more types of transformations. The idea of group equivariance has also been introduced to the capsule nets [24] by ensuring the equivariance of output pose vectors to a group of transformations. Zhang *et al.* [51] propose to learn unsupervised feature representations via Auto-Encoding Transformations (AET) by estimating transformations from the learned feature representations of both the original and transformed images. This work is later extended in [35] by introducing a variational transformation decoder, where the AET model is trained from an information-theoretic perspective by maximizing the lower bound of mutual information.

**Auto-Encoders and GANs.** Auto-encoders (AEs) have been widely adopted to learn unsupervised representations [17], which employ an encoder to extract feature representations and a decoder to reconstruct the input data from the representations. The idea is based on good feature representations should contain sufficient information to reconstruct the input data. A large number of approaches have been proposed following this paradigm of Auto-Encoding Data (AED), including variational AEs (VAEs) [18], denoising AEs [42], contrastive AEs [36], transforming AEs [16], *etc.* Based on the above approaches, graph AEs have been proposed to learn latent representations for graphs. These approaches basically learn graph embeddings for plain graphs [5, 43] and attributed graphs [19, 31], which are still trained in the AED fashion. In addition to AEs, Generative Adversarial Networks (GANs) [14] become popular for learning unsupervised representations of data, which tend to generate data from noises sampled from a random distribution. The basic idea of these models is to treat the sampled noise as the feature of the output data, and an encoder can be trained to obtain the "noise" feature representations for input data, while the generator is treated as the decoder to generate data from the "noise" feature representations [10, 11]. Recently, several approaches have been proposed to build graph GANs. For instance, [50] and [26] propose to generate nodes and edges alternately, while [7] and [3] propose to integrate GCNNs with LSTMs and GANs respectively to generate graphs.

## 3. Graph Transformations

### 3.1. Preliminaries

Consider an undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ composed of a node set $\mathcal{V}$ of cardinality $|\mathcal{V}| = N$, and an edge set $\mathcal{E}$ connecting nodes. *Graph signal* refers to data/features associated with the nodes of $\mathcal{G}$, denoted by $\mathbf{X} \in \mathbb{R}^{N \times C}$ with $i$th row representing the $C$-dimensional graph signal at the $i$th node of $\mathcal{V}$.

To characterize the similarities (and thus the graph structure) among node signals, an *adjacency matrix* $\mathbf{A}$ is defined on $\mathcal{G}$, which is a real-valued symmetric $N \times N$ matrix with $a_{i,j}$ as the weight assigned to the edge $(i, j)$ connecting nodes $i$ and $j$. Formally, the adjacency matrix is constructed from graph signals as follows,

$$\mathbf{A} = f(\mathbf{X}), \tag{1}$$

where $f(\cdot)$ is a linear or non-linear function applied to each pair of nodes to get the pair-wise similarity. For example, a widely adopted function is to nonlinearly construct a $k$-nearest-neighbor ($k$NN) graph from node features [44, 52].

### 3.2. Graph Signal Transformation

Unlike Euclidean data like images, graph signals are irregularly sampled, whose transformations are thus nontrivial to define. To this end, we define a graph transformation on the signals $\mathbf{X}$ as node-wise *filtering* on $\mathbf{X}$.

Formally, suppose we sample a graph transformation $\mathbf{t}$ from a transformation distribution $\mathcal{T}_g$, *i.e.*, $\mathbf{t} \sim \mathcal{T}_g$. Applying the transformation to graph signals $\mathbf{X}$ that are sampled from data distribution $\mathcal{X}_g$, *i.e.*, $\mathbf{X} \sim \mathcal{X}_g$, leads to the filtered graph signal

$$\tilde{\mathbf{X}} = \mathbf{t}(\mathbf{X}). \tag{2}$$

The filter $\mathbf{t}$ is applied to each node individually, which can be either node-invariant or node-variant. In other words, the transformation of each node signal associated with $\mathbf{t}$ can be different from each other. For example, for a translation $\mathbf{t}$, a distinctive translation can be applied to each node. We will call the graph transformation isotropic (anisotropic) if it is node-invariant (variant).

Consequently, the adjacency matrix of the transformed graph signal $\tilde{\mathbf{X}}$ equivaries according to (1):

$$\tilde{\mathbf{A}} = f(\tilde{\mathbf{X}}) = f(\mathbf{t}(\mathbf{X})), \tag{3}$$

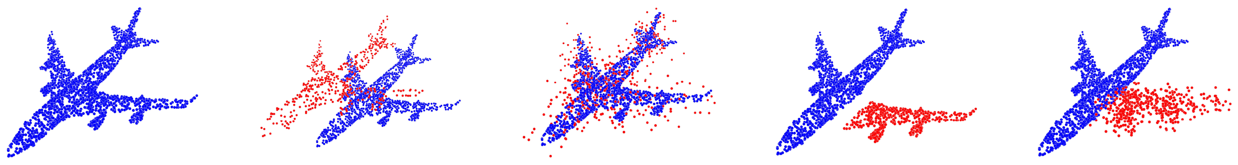
which transforms the *graph structures*, as edge weights are also filtered by $\mathbf{t}(\cdot)$.

Under this definition, there exist a wide spectrum of graph signal transformations. Examples include affine transformations (translation, rotation and shearing) on the location of nodes (*e.g.*, 3D coordinates in point clouds), and graph filters such as low-pass filtering on graph signals by the adjacency matrix [37].

### 3.3. Node-wise Graph Signal Transformation

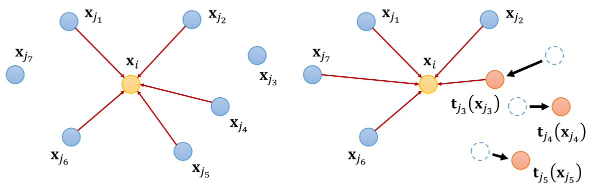
As aforementioned, in this paper, we focus on *node-wise* graph signal transformation, *i.e.*, each node has its own transformation, either isotropically or anisotropically. We seek to learn graph representations through the node-wise transformations by revealing how different parts of graph structures would change globally and locally.

Specifically, here are two distinct advantages.



(a) Original model (b) Global+Isotropic (c) Global+Anisotropic (d) Local+Isotropic (e) Local+Anisotropic

Figure 2. **Demonstration of different sampling (Global or Local) and node-wise *translation* (Isotropic or Anisotropic) methods on 3D point clouds.** Red and blue points represent transformed and non-transformed points, respectively. Note that we adopt the wing as a sampled local point set for clear visualization.



(a) Before transformation. (b) After transformation.

Figure 3. **An example of $k$NN graphs before and after node-wise transformations.** We first construct a $k$NN ($k = 5$) graph for the yellow node (other connections are omitted). Then we perform node-wise transformations on some blue nodes, which alters the graph structure around the yellow node.

- The node-wise transformations allow us to use node sampling to study different parts of graphs under various transformations.
- By decoding the node-wise transformations, we will be able to learn the representations of individual nodes. Moreover, these node-wise representations will not only capture the local graph structures under these transformations, but also contain global information about the graph when these nodes are sampled into different groups over iterations during training.

Next, we discuss the formulation of learning graph transformation equivariant representations by decoding the node-wise transformations via a graph-convolutional encoder and decoder.

## 4. GraphTER: The Proposed Approach

### 4.1. The Formulation

Given a pair of graph signal and adjacency matrix $(\mathbf{X}, \mathbf{A})$, and a pair of *transformed* graph signal and adjacency matrix $(\tilde{\mathbf{X}}, \tilde{\mathbf{A}})$ by a node-wise graph transformation $\mathbf{t}$, a function $E(\cdot)$ is *transformation equivariant* if it satisfies

$$E(\tilde{\mathbf{X}}, \tilde{\mathbf{A}}) = E\left(\mathbf{t}(\mathbf{X}), f\left(\mathbf{t}(\mathbf{X})\right)\right) = \rho(\mathbf{t})\left[E(\mathbf{X}, \mathbf{A})\right], \quad (4)$$

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

Our goal is to learn a function $E(\cdot)$, which extracts equivariant representations of graph signals $\mathbf{X}$. For this purpose, we employ an encoder-decoder network: we learn a graph encoder $E : (\mathbf{X}, \mathbf{A}) \mapsto E(\mathbf{X}, \mathbf{A})$, which encodes the feature representations of individual nodes from the graph. To ensure the transformation equivariance of representations, we train a decoder $D : \left(E(\mathbf{X}, \mathbf{A}), E(\tilde{\mathbf{X}}, \tilde{\mathbf{A}})\right) \mapsto \hat{\mathbf{t}}$ to estimate the node-wise transformation $\hat{\mathbf{t}}$ from the representations of the original and transformed graph signals. Hence, we cast the learning problem of transformation equivariant representations as the joint training of the representation encoder $E$ and the transformation decoder $D$. It has been proved that the learned representations in this way satisfy the generalized transformation equivariance without relying on a linear representation of graph structures [35].

Further, we sample a subset of nodes $\mathbf{S}$ following a sampling distribution $\mathcal{S}_g$ from the original graph signal $\mathbf{X}$, locally or globally in order to reveal graph structures at various scales. Node-wise transformations are then performed on the subset $\mathbf{S}$ isotropically or anisotropically, as demonstrated in Fig. 2. In order to predict the node-wise transformation $\mathbf{t}$, we choose a loss function $\ell_{\mathbf{S}}(\mathbf{t}, \hat{\mathbf{t}})$ that quantifies the distance between $\mathbf{t}$ and its estimate $\hat{\mathbf{t}}$ in terms of their parameters. Then the entire network is trained end-to-end by minimizing the loss

$$\min_{E,D} \underset{\mathbf{S}\sim\mathcal{S}_g}{\mathbb{E}} \underset{\substack{\mathbf{t}\sim\mathcal{T}_g \\ \mathbf{X}\sim\mathcal{X}_g}}{\mathbb{E}} \ell_{\mathbf{S}}(\mathbf{t}, \hat{\mathbf{t}}), \quad (5)$$

where the expectation $\mathbb{E}$ is taken over the sampled graph signals and transformations, and the loss is taken over the (locally or globally) sampled subset $\mathbf{S}$ of nodes in each iteration of training.

In (5), the node-wise transformation $\hat{\mathbf{t}}$ is estimated from the decoder

$$\hat{\mathbf{t}} = D\left(E(\mathbf{X}, \mathbf{A}), E(\tilde{\mathbf{X}}, \tilde{\mathbf{A}})\right). \quad (6)$$

Thus, we update the parameters in encoder $E$ and decoder $D$ iteratively by backward propagation of the loss.

### 4.2. The Algorithm

Given graph signals $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}^\top$ over $N$ nodes, in each iteration of training, we *randomly sample* a subset of nodes $\mathbf{S}$ from the graph, either globally or locally. Global sampling refers to random sampling over the entire nodes globally, while local sampling is limited to a local set of nodes in the graph. Node sampling not only enables us to characterize global and local graph structures at various scales, but also reduces the number of node-wise transformation parameters to estimate for computational efficiency.

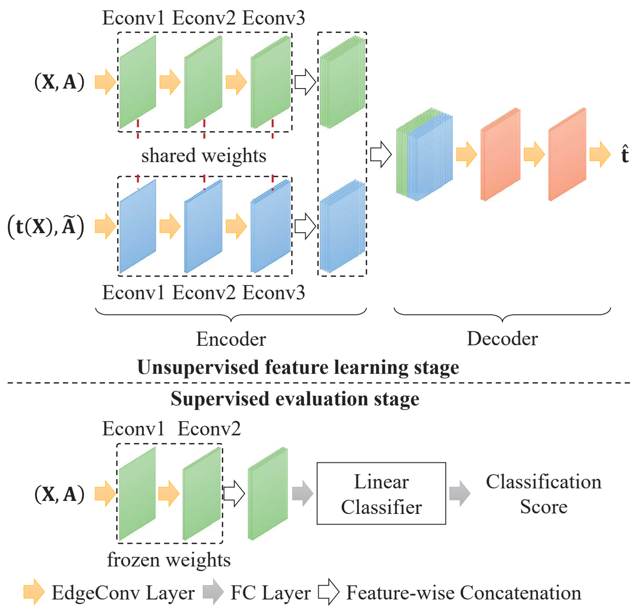


Figure 4. **The architecture of the proposed GraphTER.** In the unsupervised feature learning stage, the representation encoder and transformation decoder are jointly trained by minimizing (5). In the supervised evaluation stage, the first several blocks of the encoder are fixed with frozen weights and a linear classifier is trained with labeled samples.

Then we draw a node-wise transformation $\mathbf{t}_i$ corresponding to each sample $\mathbf{x}_i$ of nodes in $\mathbf{S}$, either isotropically or anisotropically. Accordingly, the graph $\tilde{\mathbf{A}}$ associated with the transformed graph also transforms equivariantly from the original $\mathbf{A}$ under $\mathbf{t}$. Specifically, as illustrated in Fig. 3, we construct a $k$NN graph to make use of the connectivity between the nodes, whose matrix representation in $\mathbf{A}$ changes after applying the sampled node-wise transformations.

To learn the applied node-wise transformations, we design a full graph-convolutional auto-encoder network as illustrated in Fig. 4. Among various paradigms of GCNNs, we choose EdgeConv [44] as a basic building block of the auto-encoder network, which efficiently learns node-wise representations by aggregating features along all the edges emanating from each connected node. Below we will explain the representation encoder and the transformation decoder in detail.

#### 4.2.1 Representation Encoder

The representation encoder $E$ takes the signals of an original graph $\mathbf{X}$ and the transformed counterparts $\tilde{\mathbf{X}}$ as input, along with their corresponding graphs. $E$ encodes node-wise features of $\mathbf{X}$ and $\tilde{\mathbf{X}}$ through a Siamese encoder network with shared weights, where EdgeConv layers are used as basic feature extraction blocks. As shown in Fig. 3, given a non-transformed central node $\mathbf{x}_i$ and its transformed neighbors $\mathbf{t}_j(\mathbf{x}_j)$, the input layer of encoded feature of $\mathbf{x}_i$ is

$$\begin{aligned} E_{\text{in}}(\tilde{\mathbf{X}}, \tilde{\mathbf{A}})_i &= \max_{j \in \mathcal{N}(i)} \tilde{a}_{i,j} \\ &= \max_{j \in \mathcal{N}(i)} \text{ReLU}(\theta(\mathbf{t}_j(\mathbf{x}_j) - \mathbf{x}_i) + \phi \mathbf{x}_i), \end{aligned} \tag{7}$$

where $\tilde{a}_{i,j}$ denotes the edge feature, *i.e.*, edge weight in $\tilde{\mathbf{A}}$. $\theta$ and $\phi$ are two weighting parameters, and $j \in \mathcal{N}(i)$ denotes node $j$ is in the $k$-nearest neighborhood of node $i$. Then, multiple layers of regular edge convolutions [44] are stacked to form the final encoder.

Edge convolution in (7) over each node essentially aggregates features from neighboring nodes via *edge weights* $\tilde{a}_{i,j}$. Since the edge information of the underlying graph transforms with the transformations of individual nodes as demonstrated in Fig. 3, edge convolution is able to extract higher-level features from the original and transformed edge information. Also, as features of each node are learned via propagation from transformed and non-transformed nodes isotropically or anisotropically by both local or global sampling, the learned representation is able to capture intrinsic graph structures at multiple scales.

#### 4.2.2 Transformation Decoder

Node-wise features of the original and transformed graphs are then concatenated at each node, which are then fed into the transformation decoder. The decoder consists of several EdgeConv blocks to aggregate the representations of both the original and transformed graphs to predict the node-wise transformations $\mathbf{t}$. Based on the loss in (5), $\mathbf{t}$ is decoded by minimizing the mean squared error (MSE) between the ground truth and estimated transformation parameters at each sampled node. Fig. 4 illustrates the architecture of learning the proposed GraphTER in such an auto-encoder structure.

## 5. Experiments

In this section, we evaluate the GraphTER model by applying it to graphs of 3D point cloud data on two representative downstream tasks: point cloud classification and segmentation. We compare the proposed method with state-of-the-art supervised and unsupervised approaches.

### 5.1. Datasets and Experimental Setup

**ModelNet40** [47]. This dataset contains $12,311$ meshed CAD models from $40$ categories, where $9,843$ models are used for training and $2,468$ models are for testing. For each model, $1,024$ points are sampled from the original mesh. We train the unsupervised auto-encoder and the classifier under the training set, and evaluate the classifier under the testing set.

**ShapeNet part** [49]. This dataset contains $16,881$ 3D point clouds from $16$ object categories, annotated with $50$ parts. Each 3D point cloud contains $2,048$ points, most of

Table 1. Classification accuracy (%) on ModelNet40 dataset.

| Method | Year | Unsupervised | Accuracy |
|---|---|---|---|
| 3D ShapeNets [47] | 2015 | No | 84.7 |
| VoxNet [30] | 2015 | No | 85.9 |
| PointNet [32] | 2017 | No | 89.2 |
| PointNet++ [33] | 2017 | No | 90.7 |
| KD-Net [21] | 2017 | No | 90.6 |
| PointCNN [25] | 2018 | No | 92.2 |
| PCNN [2] | 2018 | No | 92.3 |
| DGCNN [44] | 2019 | No | 92.9 |
| RS-CNN [28] | 2019 | No | 93.6 |
| T-L Network [13] | 2016 | Yes | 74.4 |
| VConv-DAE [39] | 2016 | Yes | 75.5 |
| 3D-GAN [45] | 2016 | Yes | 83.3 |
| LGAN [1] | 2018 | Yes | 85.7 |
| FoldingNet [48] | 2018 | Yes | 88.4 |
| MAP-VAE [15] | 2019 | Yes | 90.2 |
| L2G-AE [27] | 2019 | Yes | 90.6 |
| GraphTER |  | Yes | **92.0** |

which are labeled with fewer than six parts. We employ $12,137$ models for training the auto-encoder and the classifier, and $2,874$ models for testing.

We treat points in each point cloud as nodes in a graph, and the $(x, y, z)$ coordinates of points as graph signals. A $k$NN graph is then constructed on the graph signals to guide graph convolution.

Next, we introduce our node-wise graph signal transformation. In experiments, we sample a portion of nodes with a sampling rate $r$ from the entire graph to perform node-wise transformations, including 1) **Global sampling:** randomly sample $r\%$ of points from all the points in a 3D point cloud; 2) **Local sampling:** randomly choose a point and search its $k$ nearest neighbors in terms of Euclidean distance, forming a local set of $r\%$ of points.

Then, we apply three types of node-wise transformations to the coordinates of point clouds, including 1) **Translation:** randomly translate each of three coordinates of a point by three parameters in the range $[-0.2, 0.2]$; 2) **Rotation:** randomly rotate each point with three rotation parameters all in the range $[-5°, 5°]$; 3) **Shearing**: randomly shear the $x$-, $y$-, $z$-coordinates of each point with the six parameters of a shearing matrix in the range $[-0.2, 0.2]$. We consider two strategies to transform the sampled nodes: **Isotropically** or **Anisotropically**, which applies transformations with node-invariant or node-variant parameters.

## 5.2. Point Cloud Classification

First, we evaluate the GraphTER model on the ModelNet40 [47] dataset for point cloud classification.

### 5.2.1 Implementation Details

In this task, the auto-encoder network is trained via the SGD optimizer with a batch size of 32. The momentum and weight decay rate are set to $0.9$ and $10^{-4}$, respectively. The initial learning rate is $0.1$, and then decayed using a cosine annealing schedule [29] for $512$ training epochs. We adopt the cross entropy loss to train the classifier.

We deploy eight EdgeConv layers as the encoder, and the number $k$ of nearest neighbors is set to 20 for all EdgeConv layers. Similar to [44], we use shortcut connections for the first five layers to extract multi-scale features, where we concatenate features from these layers to acquire a $1,024$-dimensional node-wise feature vector. After the encoder, we employ three consecutive EdgeConv layers as the decoder – the output feature representations of the Siamese encoder first go through a channel-wise concatenation, which are then fed into the decoder to estimate node-wise transformations. The batch normalization layer and LeakyReLU activation function with a negative slope of $0.2$ is employed after each convolutional layer.

During the training procedure of the classifier, the first five EdgeConv layers in the encoder are used to represent input cloud data by node-wise concatenating their output features with the weights frozen. After the five EdgeConv layers, we apply three fully-connected layers node-wise to the aggregated features. Then, global max pooling and average pooling are deployed to acquire the global features, after which three fully-connected layers are used to map the global features to the classification scores. Dropout with a rate of $0.5$ is adopted in the last two fully-connected layers.

### 5.2.2 Experimental Results

Tab. 1 shows the results for 3D point cloud classification, where the proposed model applies isotropic node-wise shearing transformation with a global sampling rate of $r = 25\%$. We compare with two classes of methods: unsupervised approaches and supervised approaches. The GraphTER model achieves 92.0% of classification accuracy on the ModelNet40 dataset, which outperforms the state-of-the-art unsupervised methods. In particular, most of the compared unsupervised models combine the ideas of both GAN and AED, and map 3D point clouds to unsupervised representations by auto-encoding data, such as FoldingNet [48], MAP-VAE [15] and L2G-AE [27]. Results show that the GraphTER model achieves significant improvement over these methods, showing the superiority of the proposed node-wise AET over both the GAN and AED paradigms.

Moreover, the unsupervised GraphTER model also achieves comparable performance with the state-of-the-art fully supervised results. This significantly closes the gap between unsupervised approaches and the fully supervised counterparts in literature.

### 5.2.3 Ablation Studies

Further, we conduct ablation studies under various experimental settings of sampling and transformation strategies

Table 2. Unsupervised classification accuracy (%) on ModelNet40 dataset with different sampling and transformation strategies.

| | Global Sampling | | Local Sampling | | Mean |
|---|---|---|---|---|---|
| | Iso. | Aniso. | Iso. | Aniso. | |
| Translation | 90.15 | 90.15 | 89.91 | 89.55 | 89.94 |
| Rotation | 91.29 | 90.24 | 90.48 | 89.87 | 90.47 |
| Shearing | **92.02** | **90.32** | **91.65** | **89.99** | **90.99** |
| Mean | **91.15** | 90.24 | **90.68** | 89.80 | |
| | **90.70** | | 90.24 | | |

Table 3. Unsupervised classification accuracy (%) on ModelNet40 dataset applying translation at different node sampling rates.

| Sampling | Global Sampling | | Local Sampling | | Mean |
|---|---|---|---|---|---|
| Rate | Iso. | Aniso. | Iso. | Aniso. | |
| 25% | 90.15 | 90.15 | 89.91 | 89.55 | 89.94 |
| 50% | 90.03 | 89.63 | 89.95 | 89.47 | 89.77 |
| 75% | 91.00 | 89.67 | 91.41 | 89.75 | 90.46 |
| 100% | 89.67 | 89.99 | 89.67 | 89.99 | 89.83 |

on the ModelNet40 dataset.

First, we analyze the effectiveness of different node-wise transformations under global or local sampling. Tab. 2 presents the classification accuracy with three types of node-wise transformation methods. We see that the shearing transformation achieves the best performance, improving by 1.05% on average over translation, and 0.52% over rotation. This shows that the proposed GraphTER model is able to learn better feature representations under more complex transformations.

Moreover, we see that the proposed model achieves an accuracy of 90.70% on average under global sampling, which outperforms local sampling by 0.46%. This is because global sampling better captures the global structure of graphs, which is crucial in such a graph-level task of classifying 3D point clouds. Meanwhile, under the two sampling strategies, the classification accuracy from isotropic transformations is higher than that from the anisotropic one. The reason lies in the intrinsic difficulty of training the transformation decoder with increased complexity of more parameters when applying anisotropic transformations.

Moreover, we evaluate the effectiveness of different sampling rates $r$ under the translations as reported in Tab. 3. The classification accuracies under various sampling rates are almost the same, and the result under $r = 25\%$ is comparable to that under $r = 100\%$. This shows that the performance of the proposed model is insensitive to the variation of sampling rates, *i.e.*, applying node-wise transformations to a small number of nodes in the graph is sufficient to learn intrinsic graph structures.

## 5.3. Point Cloud Segmentation

We also apply the GraphTER model to 3D point cloud part segmentation on ShapeNet part dataset [49].

### 5.3.1 Implementation Details

We also use SGD optimizer to train the auto-encoding transformation network. The hyper-parameters are the same as in 3D point cloud classification except that we train for 256 epochs. We adopt the negative log likelihood loss to train the node-wise classifier for segmenting each point in the clouds.

The auto-encoding architecture is similar to that of the classification task, where we employ five EdgeConv layers as the encoder. However, the first two EdgeConv blocks consist of two MLP layers with the number of neurons $\{64, 64\}$ in each layer. We use shortcut connections to concatenate features from the first four layers to a 512-dimensional node-wise feature vector.

As for the node-wise classifier, we deploy the same architecture as in [44]. The output features from the encoder are concatenated node-wise with globally max-pooled features, followed by four fully-connected layers to classify each node. During the training procedure, the weights of the first four EdgeConv blocks in the encoder are kept frozen.

### 5.3.2 Experimental Results

We adopt the Intersection-over-Union (IoU) metric to evaluate the performance. We follow the same evaluation protocol as in the PointNet [32]: the IoU of a shape is computed by averaging the IoUs of different parts occurring in that shape, and the IoU of a category is obtained by averaging the IoUs of all the shapes belonging to that category. The mean IoU (mIoU) is finally calculated by averaging the IoUs of all the test shapes.

We also compare the proposed model with unsupervised approaches and supervised approaches in this task, as listed in Tab. 4. We achieve a mIoU of 81.9%, which significantly outperforms the state-of-the-art unsupervised method MAP-VAE [15] by 13.9%.

Moreover, the unsupervised GraphTER model also achieves the comparable performance to the state-of-the-art fully supervised approaches, greatly pushing closer towards the upper bound set by the fully supervised counterparts.

### 5.3.3 Visualization Results

Fig. 5 visualizes the results of the proposed unsupervised model and two state-of-the-art fully supervised methods: DGCNN [44] and RS-CNN [28]. The proposed model produces better segmentation on the "table" model in the first row, and achieves comparable results on the other models. Further, we qualitatively compare the proposed method with the state-of-the-art unsupervised method MAP-VAE [15], as illustrate in Fig. 6. The proposed model leads to more accurate segmentation results than MAP-VAE, *e.g.*, the engines of planes and the legs of chairs.

Table 4. Part segmentation results on ShapeNet part dataset. Metric is mIoU(%) on points.

| | Unsup. | Mean | Aero | Bag | Cap | Car | Chair | Ear Phone | Guitar | Knife | Lamp | Laptop | Motor | Mug | Pistol | Rocket | Skate Board | Table |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Samples | | | 2690 | 76 | 55 | 898 | 3758 | 69 | 787 | 392 | 1547 | 451 | 202 | 184 | 283 | 66 | 152 | 5271 |
| PointNet [32] | No | 83.7 | 83.4 | 78.7 | 82.5 | 74.9 | 89.6 | 73.0 | 91.5 | 85.9 | 80.8 | 95.3 | 65.2 | 93.0 | 81.2 | 57.9 | 72.8 | 80.6 |
| PointNet++ [33] | No | 85.1 | 82.4 | 79.0 | 87.7 | 77.3 | 90.8 | 71.8 | 91.0 | 85.9 | 83.7 | 95.3 | 71.6 | 94.1 | 81.3 | 58.7 | 76.4 | 82.6 |
| KD-Net [21] | No | 82.3 | 80.1 | 74.6 | 74.3 | 70.3 | 88.6 | 73.5 | 90.2 | 87.2 | 81.0 | 94.9 | 57.4 | 86.7 | 78.1 | 51.8 | 69.9 | 80.3 |
| PCNN [2] | No | 85.1 | 82.4 | 80.1 | 85.5 | 79.5 | 90.8 | 73.2 | 91.3 | 86.0 | 85.0 | 95.7 | 73.2 | 94.8 | 83.3 | 51.0 | 75.0 | 81.8 |
| PointCNN [25] | No | 86.1 | 84.1 | 86.5 | 86.0 | 80.8 | 90.6 | 79.7 | 92.3 | 88.4 | 85.3 | 96.1 | 77.2 | 95.3 | 84.2 | 64.2 | 80.0 | 83.0 |
| DGCNN [44] | No | 85.2 | 84.0 | 83.4 | 86.7 | 77.8 | 90.6 | 74.7 | 91.2 | 87.5 | 82.8 | 95.7 | 66.3 | 94.9 | 81.1 | 63.5 | 74.5 | 82.6 |
| RS-CNN [28] | No | 86.2 | 83.5 | 84.8 | 88.8 | 79.6 | 91.2 | 81.1 | 91.6 | 88.4 | 86.0 | 96.0 | 73.7 | 94.1 | 83.4 | 60.5 | 77.7 | 83.6 |
| LGAN [1] | Yes | 57.0 | 54.1 | 48.7 | 62.6 | 43.2 | 68.4 | 58.3 | 74.3 | 68.4 | 53.4 | 82.6 | 18.6 | 75.1 | 54.7 | 37.2 | 46.7 | 66.4 |
| MAP-VAE [15] | Yes | 68.0 | 62.7 | 67.1 | 73.0 | 58.5 | 77.1 | 67.3 | 84.8 | 77.1 | 60.9 | 90.8 | 35.8 | 87.7 | 64.2 | 45.0 | 60.4 | 74.8 |
| GraphTER | Yes | **81.9** | **81.7** | **68.1** | **83.7** | **74.6** | **88.1** | **68.9** | **90.6** | **86.6** | **80.0** | **95.6** | **56.3** | **90.0** | **80.8** | **55.2** | **70.7** | **79.1** |

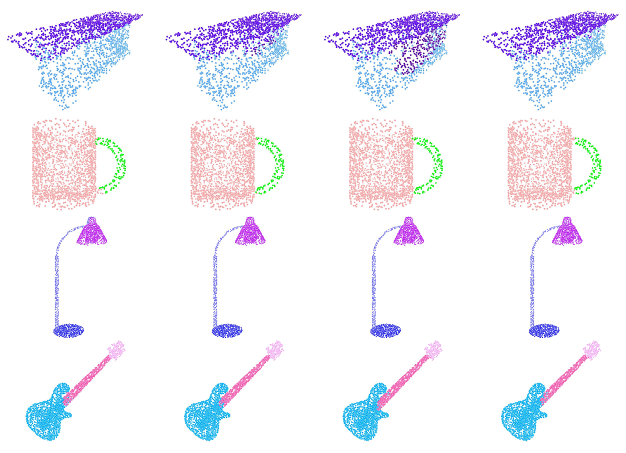
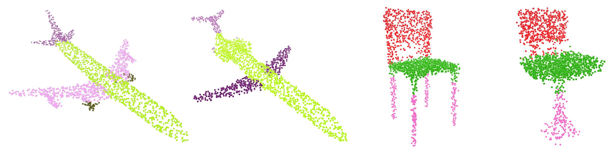

(a) Ground-truth (b) DGCNN (c) RS-CNN (d) GraphTER

Figure 5. **Visual comparison of point cloud part segmentation with supervised methods.** Our unsupervised GraphTER learning achieves comparable results with the state-of-the art fully supervised approaches.


(a) MAP-VAE

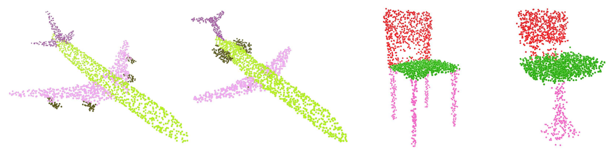

(b) GraphTER

Figure 6. **Visual comparison of point cloud part segmentation with the state-of-the-art unsupervised method MAP-VAE.** We achieve more accurate segmentation even in tiny parts and transition regions.

#### 5.3.4 Ablation Studies

Similar to the classification task, we analyze the effectiveness of different node-wise transformations under global or local sampling, as presented in Tab. 5. The proposed model achieves the best performance under the shearing transformation, improving by 1.23% on average over translation, and 0.78% over rotation, which demonstrates the benefits of GraphTER learning under complex transformations.

Table 5. Unsupervised segmentation results on ShapeNet part dataset with different transformation strategies. Metric is mIoU (%) on points.

| | Global Sampling | | Local Sampling | | Mean |
|---|---|---|---|---|---|
| | Iso. | Aniso. | Iso. | Aniso. | |
| Translation | 79.83 | 79.88 | 80.05 | 79.85 | 79.90 |
| Rotation | 80.20 | **80.29** | 80.87 | 80.02 | 80.35 |
| Shearing | **81.88** | 80.28 | **81.89** | **80.48** | **81.13** |
| Mean | **80.64** | 80.15 | **80.94** | 80.12 | |
| | 80.39 | | **80.53** | | |

Further, the proposed model achieves a mIoU of 80.53% on average under local sampling, which outperforms global sampling by 0.14%. This is because local sampling of nodes captures the local structure of graphs better, which is crucial in node-level 3D point cloud segmentation task.

## 6. Conclusion

In this paper, we propose a novel paradigm of learning graph transformation equivariant representation (GraphTER) via auto-encoding node-wise transformations in an unsupervised fashion. We allow it to sample different groups of nodes from a graph globally or locally and then perform node-wise transformations isotropically or anisotropically, which enables it to characterize morphable structures of graphs at various scales. By decoding these node-wise transformations, GraphTER enforces the encoder to learn intrinsic representations that contain sufficient information about structures under applied transformations. We apply the GraphTER model to classification and segmentation of graphs of 3D point cloud data, and experimental results demonstrate the superiority of GraphTER over the state-of-the-art unsupervised approaches, significantly closing the gap with the fully supervised counterparts. We will apply the general GraphTER model to more applications as future works, such as node classification of citation networks.

## References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 40–49, 2018.

[2] Matan Atzmon, Haggai Maron, and Yaron Lipman. Point convolutional neural networks by extension operators. *ACM Transactions on Graphics (TOG)*, 37(4):1–12, July 2018.

[3] Aleksandar Bojchevski, Oleksandr Shchur, Daniel Zügner, and Stephan Günnemann. Netgan: Generating graphs via random walks. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 609–618, 2018.

[4] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[5] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Deep neural networks for learning graph representations. In *Thirtieth AAAI Conference on Artificial Intelligence (AAAI)*, pages 1145–1152, 2016.

[6] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pages 2990–2999, 2016.

[7] Nicola De Cao and Thomas Kipf. Molgan: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973*, 2018.

[8] Sander Dieleman, Jeffrey De Fauw, and Koray Kavukcuoglu. Exploiting cyclic symmetry in convolutional neural networks. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pages 1889–1898, 2016.

[9] Sander Dieleman, Kyle W Willett, and Joni Dambre. Rotation-invariant convolutional neural networks for galaxy morphology prediction. *Monthly Notices of the Royal Astronomical Society*, 450(2):1441–1459, 2015.

[10] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. In *International Conference on Learning Representations (ICLR)*, 2017.

[11] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially learned inference. In *International Conference on Learning Representations (ICLR)*, 2017.

[12] Robert Gens and Pedro M Domingos. Deep symmetry networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2537–2545, 2014.

[13] Rohit Girdhar, David F Fouhey, Mikel Rodriguez, and Abhinav Gupta. Learning a predictable and generative vector representation for objects. In *European Conference on Computer Vision (ECCV)*, pages 484–499. Springer, 2016.

[14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2672–2680, 2014.

[15] Zhizhong Han, Xiyang Wang, Yu-Shen Liu, and Matthias Zwicker. Multi-angle point cloud-vae: Unsupervised feature learning for 3d point clouds from multiple angles by joint self-reconstruction and half-to-half prediction. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[16] Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. Transforming auto-encoders. In *International Conference on Artificial Neural Networks (ICANN)*, pages 44–51. Springer, 2011.

[17] Geoffrey E Hinton and Richard S Zemel. Autoencoders, minimum description length and helmholtz free energy. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3–10, 1994.

[18] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

[19] Thomas N Kipf and Max Welling. Variational graph auto-encoders. In *Proceedings of the NIPS Workshop on Bayesian Deep Learning*, 2016.

[20] Jyri J Kivinen and Christopher KI Williams. Transformation equivariant boltzmann machines. In *International Conference on Artificial Neural Networks (ICANN)*, pages 1–9. Springer, 2011.

[21] Roman Klokov and Victor Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 863–872, 2017.

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012.

[23] Karel Lenc and Andrea Vedaldi. Understanding image representations by measuring their equivariance and equivalence. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 991–999, 2015.

[24] Jan Eric Lenssen, Matthias Fey, and Pascal Libuschewski. Group equivariant capsule networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 8844–8853, 2018.

[25] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In *Advances in Neural Information Processing Systems (NIPS)*, pages 820–830, 2018.

[26] Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*, 2018.

[27] Xinhai Liu, Zhizhong Han, Xin Wen, Yu-Shen Liu, and Matthias Zwicker. L2g auto-encoder: Understanding point clouds by local-to-global reconstruction with hierarchical self-attention. In *Proceedings of the 27th ACM International Conference on Multimedia (ACM MM)*, pages 989–997. ACM, 2019.

[28] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-shape convolutional neural network for point cloud analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8895–8904, 2019.

[29] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations (ICLR)*, 2017.

[30] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928. IEEE, 2015.

[31] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. Adversarially regularized graph autoencoder for graph embedding. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2609–2615, 2018.

[32] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 652–660, 2017.

[33] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5099–5108, 2017.

[34] Guo-Jun Qi. Learning generalized transformation equivariant representations via autoencoding transformations. *arXiv preprint arXiv:1906.08628*, 2019.

[35] Guo-Jun Qi, Liheng Zhang, Chang Wen Chen, and Qi Tian. Avt: Unsupervised learning of transformation equivariant representations by autoencoding variational transformations. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019.

[36] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 833–840. Omnipress, 2011.

[37] Aliaksei Sandryhaila and José MF Moura. Discrete signal processing on graphs. *IEEE transactions on Signal Processing*, 61(7):1644–1656, 2013.

[38] Uwe Schmidt and Stefan Roth. Learning rotation-aware features: From invariant priors to equivariant descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2050–2057. IEEE, 2012.

[39] Abhishek Sharma, Oliver Grau, and Mario Fritz. Vconv-dae: Deep volumetric shape learning without object labels. In *European Conference on Computer Vision (ECCV)*, pages 236–250. Springer, 2016.

[40] Henrik Skibbe. *Spherical Tensor Algebra for Biomedical Image Analysis*. PhD thesis, Verlag nicht ermittelbar, 2013.

[41] Kihyuk Sohn and Honglak Lee. Learning invariant representations with local transformations. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, pages 1339–1346, 2012.

[42] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pages 1096–1103. ACM, 2008.

[43] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1225–1234. ACM, 2016.

[44] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 38(5):146, 2019.

[45] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in Neural Information Processing Systems (NIPS)*, pages 82–90, 2016.

[46] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596*, 2019.

[47] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, 2015.

[48] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 206–215, 2018.

[49] Li Yi, Vladimir G Kim, Duygu Ceylan, I Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, Leonidas Guibas, et al. A scalable active framework for region annotation in 3d shape collections. *ACM Transactions on Graphics (TOG)*, 35(6):210, 2016.

[50] Jiaxuan You, Rex Ying, Xiang Ren, William L Hamilton, and Jure Leskovec. GraphRNN: A deep generative model for graphs. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 5694–5703, 2018.

[51] Liheng Zhang, Guo-Jun Qi, Liqiang Wang, and Jiebo Luo. Aet vs. aed: Unsupervised representation learning by autoencoding transformations rather than data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2547–2555, 2019.

[52] Yingxue Zhang and Michael Rabbat. A graph-cnn for 3d point cloud classification. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6279–6283. IEEE, 2018.

[53] Ziwei Zhang, Peng Cui, and Wenwu Zhu. Deep learning on graphs: A survey. *arXiv preprint arXiv:1812.04202*, 2018.

[54] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018.