# 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
[2]Futurewei Technologies

2020/5/14

# Outline

- Introduction to GraphTER
- Graph Signal Transformation
- Problem Formulation
- Proposed Algorithm
- Experiments
- Conclusion

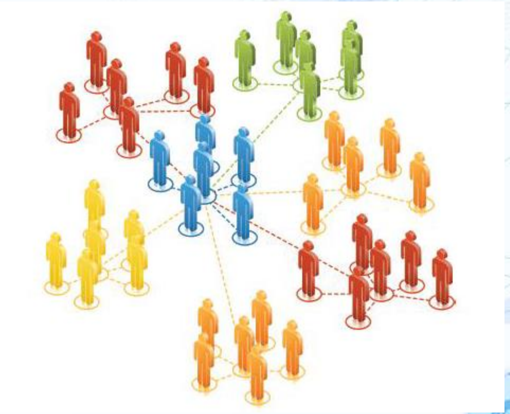
# 01 Introduction to GraphTER

# Introduction

- Graphs serve as a natural representation of irregular data, such as social networks and 3D point clouds

- Graph Convolutional Neural Networks (GCNNs) have shown their efficiency in learning representations of irregular data

- Existing GCNNs are mostly trained in a (semi-)supervised fashion, requiring a large amount of labeled data



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.

# Related Works in TER

- Two representative unsupervised methods:
  Auto-Encoders (AEs) & Generative Adversarial Networks (GANs)

- Transformation equivariant representations (TER) learning: further improvement
  - Assumption:
  Representations equivarying to transformations are able to encode the intrinsic structures of data
  ➡ the transformations can be reconstructed from the representations of data before and after transformations
  - Hinton' s seminal work on learning transformation capsules


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.

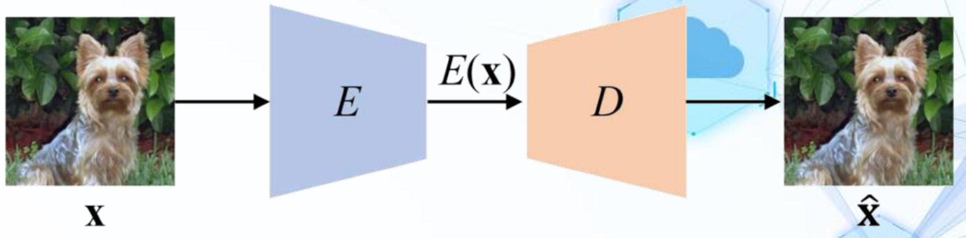
Guo-Jun Qi. Learning generalized transformation equivariant representations via autoencoding transformations. arXiv preprint arXiv:1906.08628, 2019.

# Related Works in TER

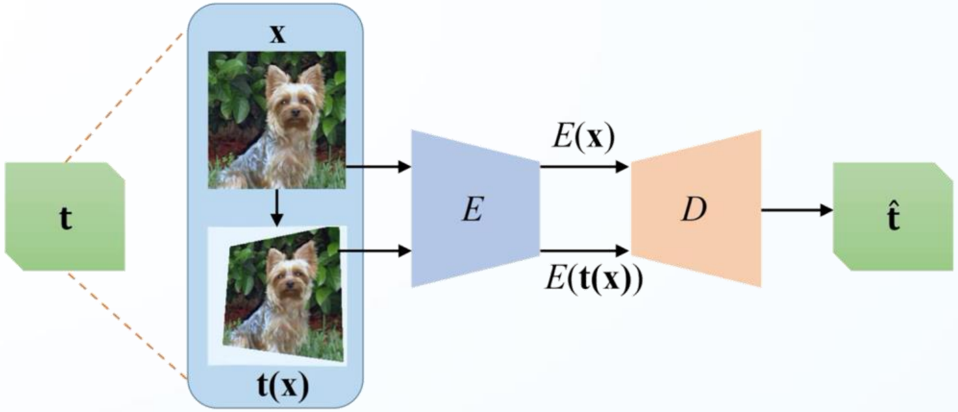
- Auto-Encoding Transformation (AET) [1] learns unsupervised representations by estimating the input transformations rather than data (AED)

- Auto-Encoding Variational Transformations (AVT) [2] extends AET from an information-theoretic perspective by maximizing the lower bound of mutual information

- **Limitation:** focus on ***Euclidean data*** such as images, which cannot be directly extended to graphs due to the irregular data structures




(a) Auto-Encoding Data (AED)

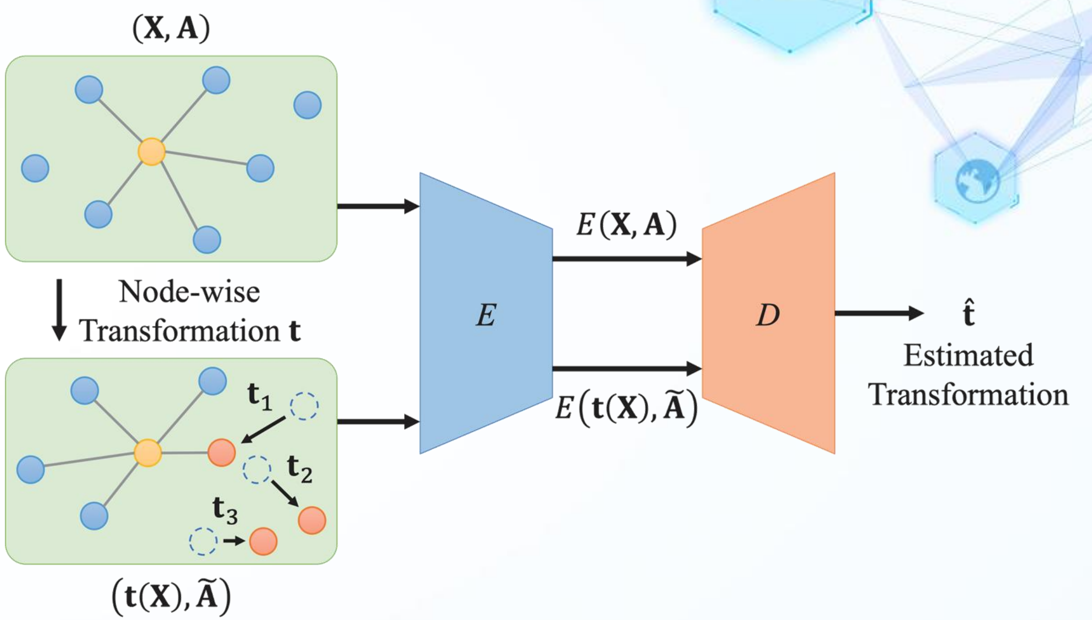



(b) Auto-Encoding Transformation (AET)


[1] L. Zhang, G.-J. Qi, L. Wang, and J. Luo, "AET vs. AED: Unsupervised Representation Learning by Auto-Encoding Transformations Rather Than Data," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.
[2] G.-J. Qi, L. Zhang, C. W. Chen, and Q. Tian, "AVT: Unsupervised Learning of Transformation Equivariant Representations by Autoencoding Variational Transformations," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 8129-8138.

# Contributions

- Propose Graph Transformation Equivariant Representation (GraphTER) learning to extract adequate graph representations in an unsupervised fashion
- Define generic graph signal transformations
- Formalize the GraphTER by decoding node-wise transformations end-to-end in a graph-convolution auto-encoder architecture



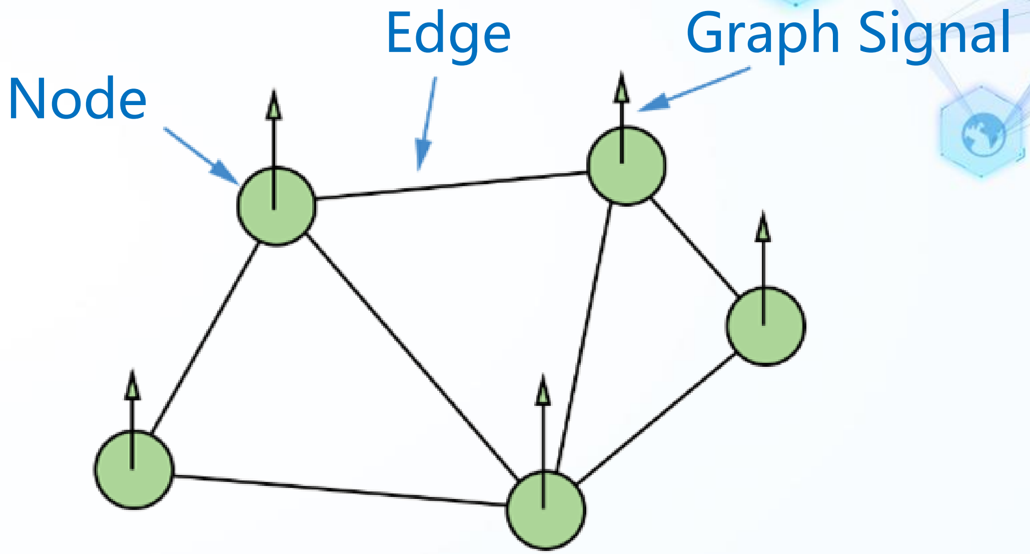
02

# Graph Signal Transformation

# Preliminaries

- A graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{A}\}$
  - a node set $\mathcal{V}$ of cardinality $|\mathcal{V}| = N$
  - an edge set $\mathcal{E}$ connecting nodes
  - weighted adjacency matrix $\mathbf{A}$

- **Graph signal** refers to data/features associated with the nodes of $\mathcal{G}$
  - denoted by $\mathbf{X} \in \mathbb{R}^{N \times C}$



# Preliminaries

- The adjacency matrix is constructed from graph signals

$$\mathbf{A} = f(\mathbf{X})$$

$f(\cdot)$ is often a non-linear function applied to nodes to compute the pair-wise similarity

e.g., a K-nearest neighbor graph

# Graph Signal Transformation

- We define a graph signal transformation on the signals $\mathbf{X}$ as node-wise filtering on $\mathbf{X}$
  - Low-pass graph filtering: e.g., $\mathbf{AX}$
  - High-pass graph filtering: e.g., $(\boldsymbol{I} - \mathbf{A})\mathbf{X}$
  - Node-independent graph filtering

- Applying the graph signal transformation $\mathbf{t}$ to graph signals $\mathbf{X} \sim \mathcal{X}_g$

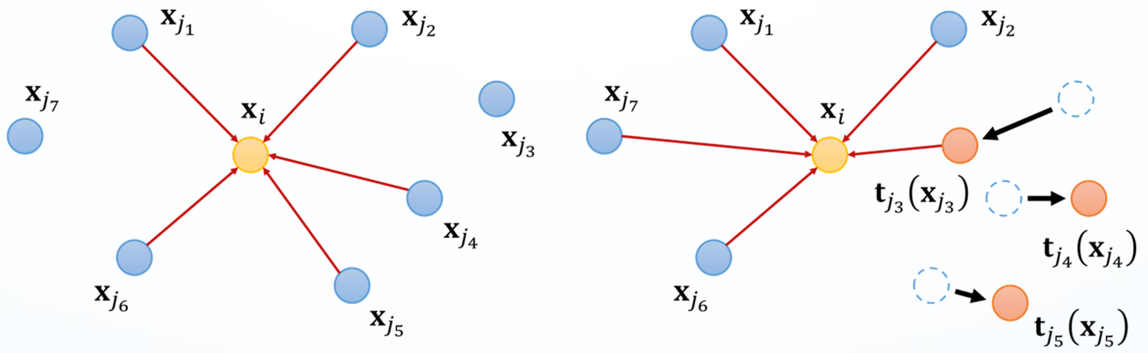
$$\widetilde{\mathbf{X}} = \mathbf{t}(\mathbf{X})$$

# Graph Signal Transformation

- The adjacency matrix of the transformed graph signal $\tilde{\mathbf{X}}$ equivaries implicitly

$$\tilde{\mathbf{A}} = f\left(\tilde{\mathbf{X}}\right) = f\left(\mathbf{t}(\mathbf{X})\right)$$

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




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

# Node-wise Transformation

- 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
  - capture the local graph structures under these transformations
  - contain global information about the graph when these nodes are sampled into different groups over iterations during training

# Node-wise Transformation

- The filter $\mathbf{t}$ is applied to each node individually, which can be either node-invariant (isotropic) or node-variant (anisotropic)

- We take affine transformations (e.g., translation, rotation and shear) on points of 3D point clouds as the straightforward node-wise transformations

Original

Isotropic Rotation

Anisotropic Rotation

# 03 Problem Formulation

# Transformation Equivariant

- A function $E(\cdot)$ is transformation equivariant if

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

node-wise transformation

homomorphism transformation of t

- The function $E(\cdot)$ extracts equivariant representations of graph signals $\mathbf{X}$

# Transformation Equivariant

- A function $E(\cdot)$ is transformation equivariant if

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

node-wise transformation

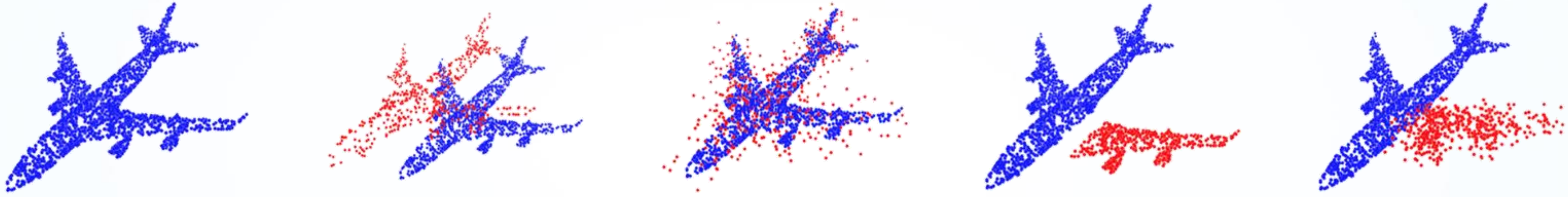
homomorphism transformation of t

- We learn a graph encoder $E: (\mathbf{X}, \mathbf{A}) \mapsto E(\mathbf{X}, \mathbf{A})$ to encode the feature representations of individual nodes from the graph

- 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

# Network Optimization

- We sample a subset of nodes $\mathbf{S}$ from the original graph signal $\mathbf{X}$, locally or globally to reveal graph structures at various scales



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

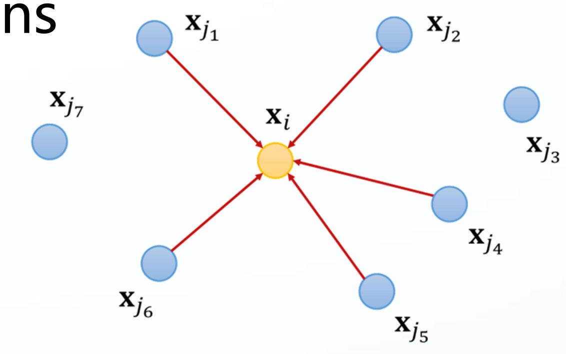
- The network is trained by minimizing the loss

$$\min_{E,D} \mathop{\mathbb{E}}_{\mathbf{S}\sim\mathcal{S}_g} \mathop{\mathbb{E}}_{\substack{\mathbf{t}\sim\mathcal{T}_g \\ \mathbf{X}\sim\mathcal{X}_g}} \ell_{\mathbf{S}}(\mathbf{t}, \hat{\mathbf{t}}) \quad \longleftarrow \quad \hat{\mathbf{t}} = D\left(E(\mathbf{X}, \mathbf{A}), E(\widetilde{\mathbf{X}}, \widetilde{\mathbf{A}})\right)$$

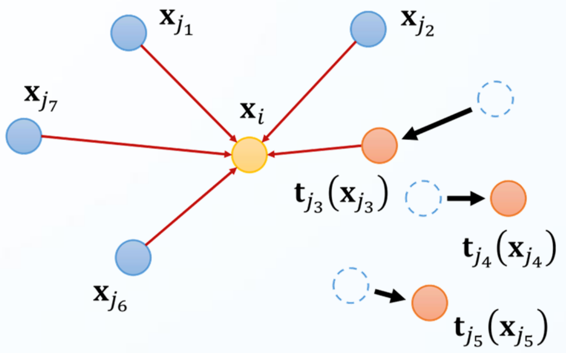
# 04 Proposed Algorithm

# Unsupervised Graph Feature Learning

1. Construct a $k$-nearest-neighbor ($k$NN) graph $\mathbf{A}$ over graph signals $\mathbf{X}$
2. Sample a subset of nodes $\mathbf{S} = \{\mathbf{x}_1, \ldots \mathbf{x}_m\}^\top$ from $\mathbf{X}$ locally or globally
3. Apply node-wise transformation $\mathbf{t}_i$ to each sample $\mathbf{x}_i$ in $\mathbf{S}$: the transformed signal $\tilde{\mathbf{X}}$
4. Update the adjacency matrix $\tilde{\mathbf{A}}$ of the transformed signal $\tilde{\mathbf{X}}$
5. Feed $(\mathbf{X}, \mathbf{A})$ and $(\tilde{\mathbf{X}}, \tilde{\mathbf{A}})$ into the graph-convolutional auto-encoder network to learn transformation equivariant representations




(a) Before transformation.

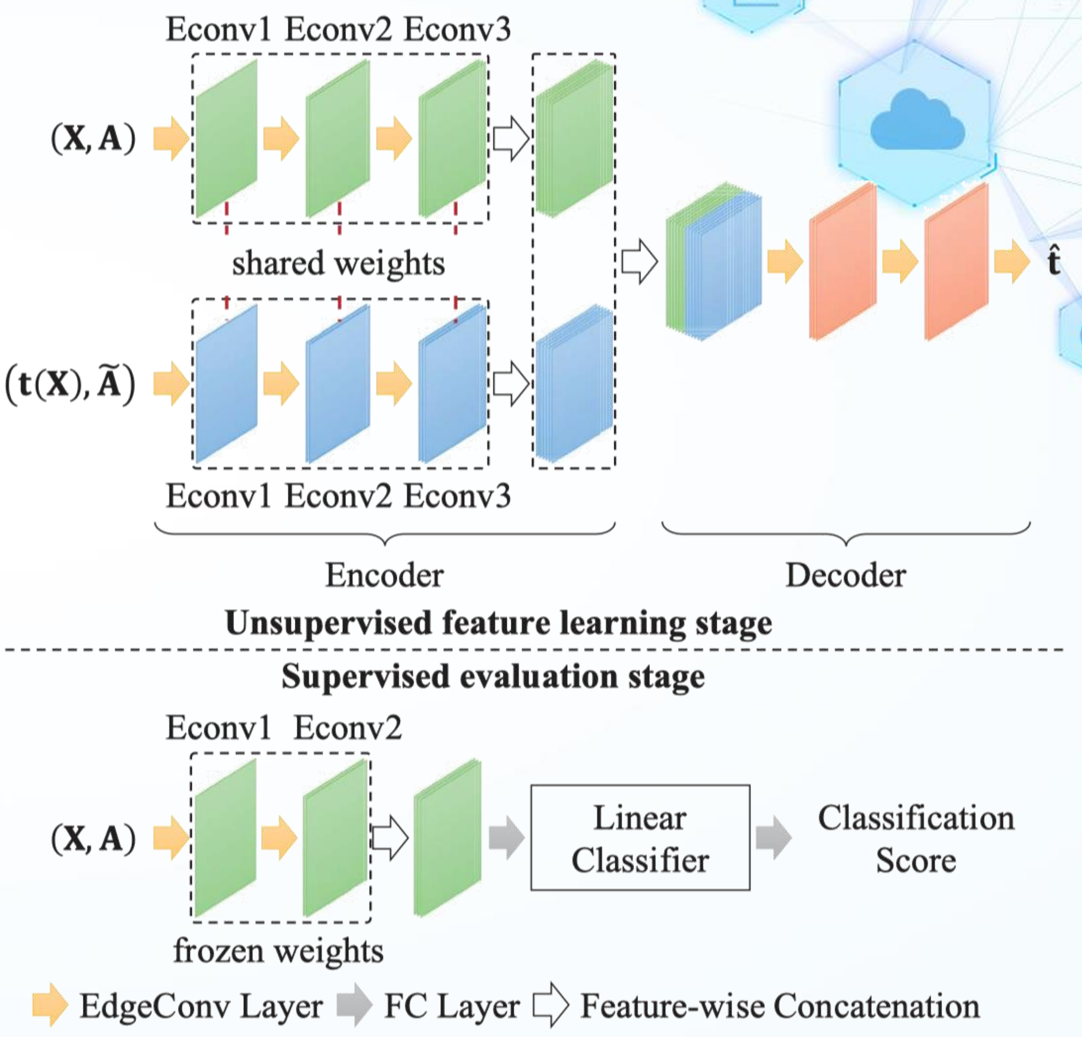



(b) After transformation.

# Graph-convolutional Auto-encoder Network

- Representation Encoder
  - Learn the representations of $(\mathbf{X}, \mathbf{A})$ and the transformed counterparts $\left(\mathbf{t}(\mathbf{X}), \widetilde{\mathbf{A}}\right)$
  - A Siamese Network, with EdgeConv [1] as the building block

- Transformation Decoder
  - Aggregate the representations of both $(\mathbf{X}, \mathbf{A})$ and $\left(\mathbf{t}(\mathbf{X}), \widetilde{\mathbf{A}}\right)$ to predict the node-wise transformation $\mathbf{t}$




[1] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic Graph CNN for Learning on Point Clouds," *ACM Transactions on Graphics*, 2019.

# 05 Experiments

# 3D Point Cloud Classification

- Dataset: ModelNet40
- Metric: Accuracy (%)

| 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** |

# 3D Point Cloud Classification

- **Ablation Study I**

  different sampling and transformation strategies (at sampling rate 25%):

| | 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 | | |

# 3D Point Cloud Classification

- **Ablation Study II**

  different node sampling rates (on Translation transformation):

| Sampling Rate | Global Sampling | | Local Sampling | | Mean |
|---|---|---|---|---|---|
| | 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 |

# 3D Point Cloud Segmentation

- Dataset: ShapeNet Part
- Metric: mIoU (%)

| | 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** |

# 3D Point Cloud Segmentation

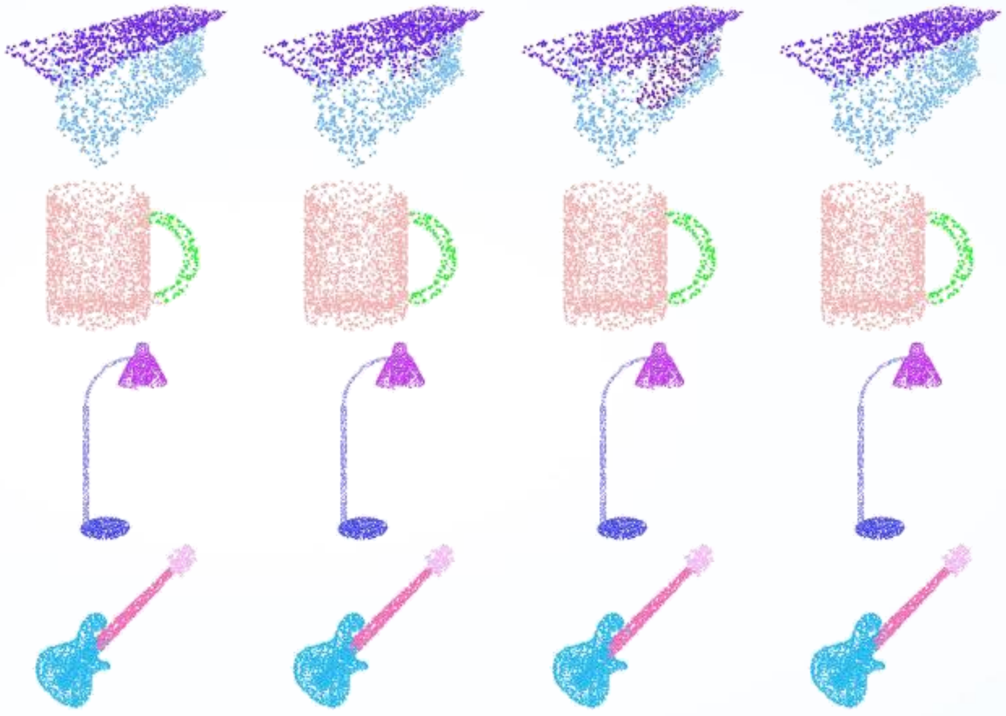
- **Ablation Study**

  different sampling and transformation strategies (at sampling rate 25%):

| | 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** | | |

# 3D Point Cloud Segmentation

Visual comparison with the **supervised** methods



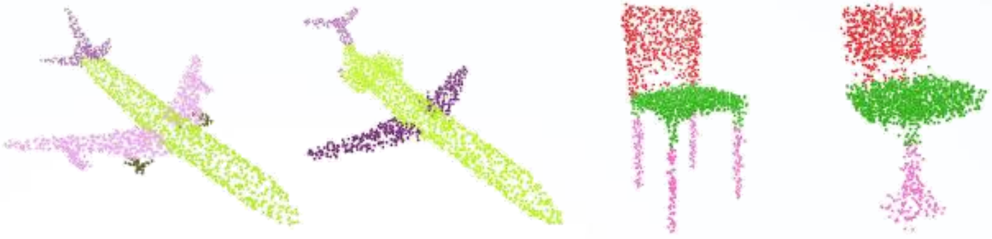
(a) Ground-truth (b) DGCNN (c) RS-CNN (d) Ours

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.

Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-shape convolutional neural network for point cloud analysis. In Proceedings of the IEEE CVPR, pages 8895–8904, 2019.

# 3D Point Cloud Segmentation

Visual comparison with
the SOTA **unsupervised** method



(a) MAP-VAE

(b) Ours

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. ICCV, October 2019.

# 06 Conclusion

# Conclusion

- Propose Graph Transformation Equivariant Representation (GraphTER) learning via auto-encoding node-wise transformations in an unsupervised fashion

- To characterize morphable structures of graphs at various scales:
  - sampling (globally or locally)
  - node-wise transformations (isotropically or anisotropically)

- By decoding node-wise transformations, GraphTER enforces the encoder to learn **intrinsic representations** under applied transformations.

- The general GraphTER model is applicable to various applications, e.g., point cloud learning, node classification of citation networks, etc.

# 谢谢观看

# THANK YOU

arXiv: https://arxiv.org/abs/1911.08142
Code: https://github.com/gyshgx868/graph-ter
Web: http://www.wict.pku.edu.cn/huwei/