Dynamic Graph CNN for Learning on Point Clouds

Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay Sarma, Michael Bronstein, Justin Solomon

Presenter: Xingchao Liu

2021.09.07

Problem: Deep 3D Perception on Point Clouds

- Point cloud is a natural form of 3D sensing, which is usually the output of many 3D sensors like LiDAR.
  Applications: indoor navigation, self-driving vehicles, etc..

- Deep point cloud processing outperforms traditional methods in various tasks, e.g., point cloud classification and semantic segmentation, but it is **non-trivial**.

- **Challenge:** Point clouds are fundamentally **irregular**

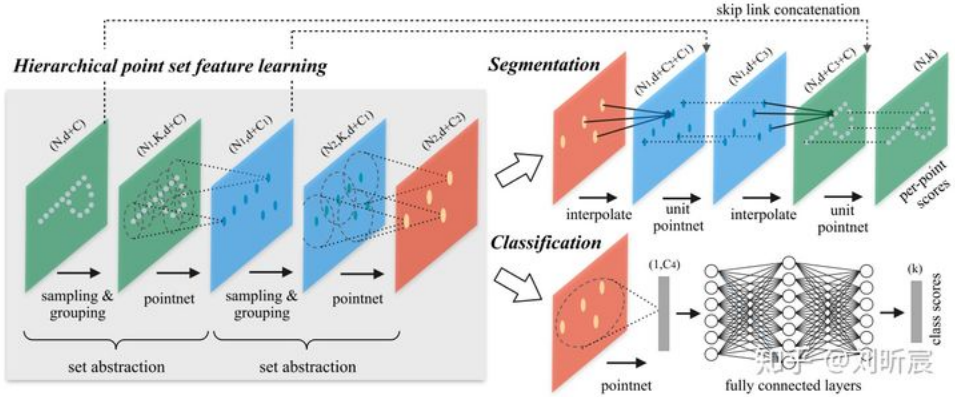
- **Requires neural networks specially designed for point clouds!**

Existing Method: PointNet

- **A pioneering work**: designed specifically for point cloud processing
- **Permutation invariant**: global feature will not change if we change the order of the input points
- **Drawback**: works on individual points, ignores the geometric cues; cannot capture local information

Qi C R, Su H, Mo K, et al. Pointnet: Deep learning on point sets for 3d classification and segmentation[C]

Existing Method: PointNet++



- **Improvement**: capture local information by sampling and grouping with furthest point sampling
  Recall previous talk
- **Still permutation invariant**
- **Drawback**: still treat points independently in local scale because of using PointNet

Qi C R, Yi L, Su H, et al. Pointnet++: Deep hierarchical feature learning on point sets in a metric space

Motivation of the Proposed Method

- **Proposed Method: EdgeConv**
  Will elaborate later
- **Motivation 1**: The local structure itself is informative. It should be considered when processing each point
- **Motivation 2**: Keep permutation invariance
- **Motivation 3**: Easy to implement

EdgeConv

$\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subseteq \mathbb{R}^F$: Input point cloud / features in the intermediate layers

$\mathcal{G} = (\mathcal{V}, \mathcal{E})$: A k-nearest neighbor graph (only nodes that are kNNs are connected)

$\boldsymbol{e}_{ij} = h_{\boldsymbol{\Theta}}(\mathbf{x}_i, \mathbf{x}_j)$: Edge features, where h is a nonlinear function with learnable parameters

EdgeConv: $\mathbf{x}'_i = \square_{j:(i,j)\in\mathcal{E}} h_{\boldsymbol{\Theta}}(\mathbf{x}_i, \mathbf{x}_j).$

$\square$: An aggregation function, can be sum or max

$h_{\boldsymbol{\Theta}}(\mathbf{x}_i, \mathbf{x}_j) = \bar{h}_{\boldsymbol{\Theta}}(\mathbf{x}_i, \mathbf{x}_j - \mathbf{x}_i).$: The authors adopt this form of nonlinear function

EdgeConv: An illustration

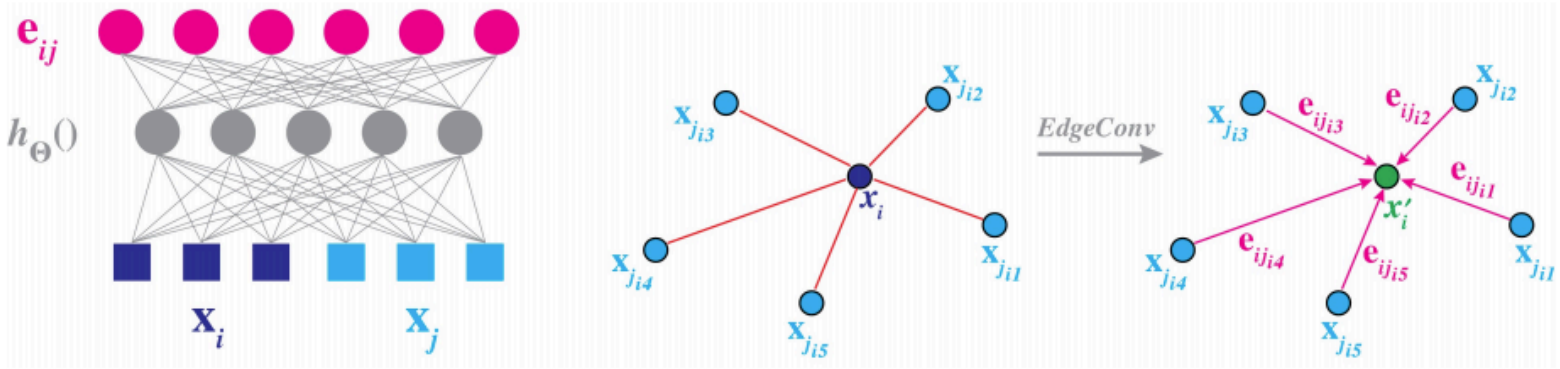


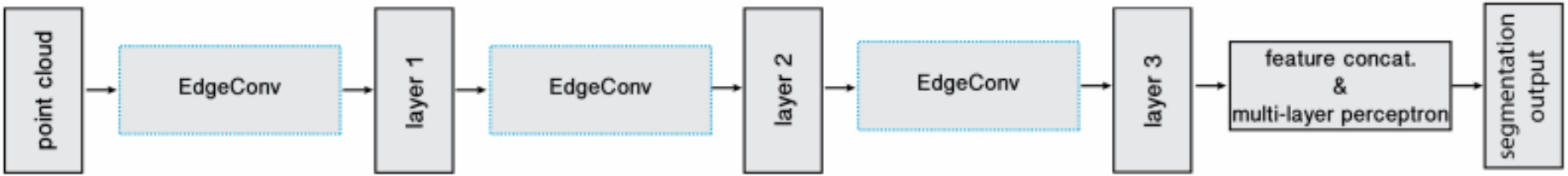
Fig. 2. **Left**: Computing an edge feature, $\mathbf{e}_{ij}$ (top), from a point pair, $\mathbf{x}_i$ and $\mathbf{x}_j$ (bottom). In this example, $h_\Theta()$ is instantiated using a fully connected layer, and the learnable parameters are its associated weights. **Right**: The EdgeConv operation. The output of EdgeConv is calculated by aggregating the edge features associated with all the edges emanating from each connected vertex.

- Concatenate the point features, pass the concatenated feature through a MLP to get edge features
- A symmetric aggregation function is applied upon the edge features to get new point feature

EdgeConv: Properties

- EdgeConv extracts local information for **every point**
- EdgeConv is beyond the coordinate space, instead, **it works in the feature space** so that it can draw connection between semantically close points (In contrast, PointNet++ only works in the original coordinate space).
- EdgeConv is **permutation invariant and easy to implement** because it only involves kNN and symmetric operation

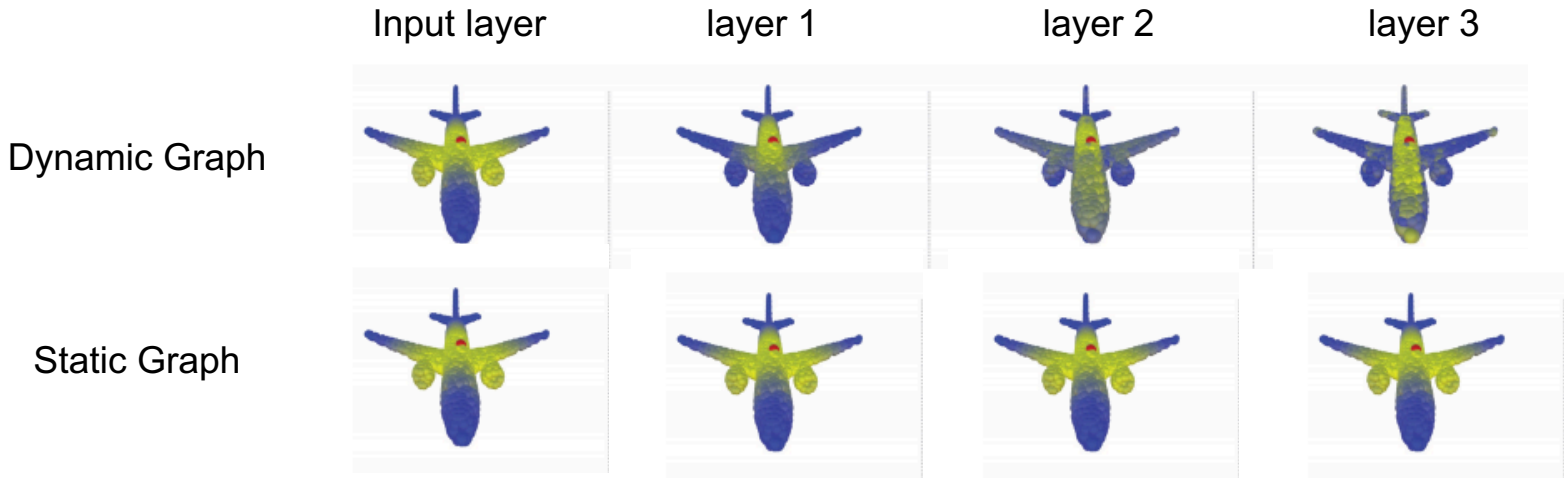
Dynamic Graph CNN



- By simply plug in EdgeConvs into a network, we get Dynamic Graph CNN (DGCNN).

- **Dynamic Graph!**
  The authors find it beneficial to recompute the graph using the nearest neighbors in the feature space produced by each layer.
  A **new graph, instead of a static graph** for each layer !
  A major difference from previous graph CNNs.

Why Dynamic Graph?



Dynamic graph enables the network to choose which point to use from the global point set.

Static graph is too restrictive, can be a suboptimal choice for extracting local information.

In later stages, DGCNN learns to gather semantically similar points, despite the large distance in the original coordinate space

Relationship to Transformers

	DGCNN	Transformers
Global Receptive Field	Yes	Yes
Distance Function	Euclidean Distance	Inner Product
Attention Score	Binary (kNN)	Softmax
Aggregation Function	Max Pooling	Linear Combination

A transformer-like network before transformer era.

Empirical Evaluation: Point Cloud Classification

- **Dataset**: ModelNet40, which consists 12,311 meshed CAD models with 40 categories.
  9,843 models are used for training and 2,468 models are used for testing.
  1024 points are uniformly sampled to create the point clouds.
- **Hyperparameters**: k = 20. See the paper for others.
- **Tested Methods**:

  Ours (Baseline): Static graph computed from the input.

  Ours: Dynamic graph.

  Ours (2048): Dynamic graph with 2048 input points and k = 40.

Outperforms previous SOTAs by a 0.6% on overall Acc!
0.7% improvement from dynamic graph.

	Mean Class Accuracy	Overall Accuracy
3DShapeNets [Wu et al. 2015]	77.3	84.7
VoxNet [Maturana and Scherer 2015]	83.0	85.9
Subvolume [Qi et al. 2016]	86.0	89.2
VRN (single view) [Brock et al. 2016]	88.98	-
VRN (multiple views) [Brock et al. 2016]	91.33	-
ECC [Simonovsky and Komodakis 2017]	83.2	87.4
PointNet [Qi et al. 2017b]	86.0	89.2
PointNet++ [Qi et al. 2017c]	-	90.7
Kd-net [Klokov and Lempitsky 2017]	-	90.6
PointCNN [Li et al. 2018a]	88.1	92.2
PCNN [Atzmon et al. 2018]	-	92.3
Ours (baseline)	88.9	91.7
Ours	**90.2**	**92.9**
Ours (2048 points)	**90.7**	**93.5**

Table 2. Classification results on ModelNet40.

Empirical Evaluation: Part Segmentation

- **Dataset**: ShapeNet part dataset.

  Each point from a point cloud set is classified into one of the 50 predefined part labels.

  The dataset contains 16,881 3D shapes from 16 object categories.

  2,048 points are sampled from each training shape.

- **Metric**: Intersection-over-Union (IoU)

	MEAN	AREO	BAG .	CAP	CAR	CHAIR	EAR PHONE	GUITAR	KNIFE	LAMP	LAPTOP	MOTOR	MUG	PISTOL	ROCKET	SKATE BOARD	TABLE
# SHAPES		2690	76	55	898	3758	69	787	392	1547	451	202	184	283	66	152	5271
POINTNET	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++	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	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
LOCALFEATURENET	84.3	**86.1**	73.0	54.9	77.4	88.8	55.0	90.6	86.5	75.2	**96.1**	57.3	91.7	83.1	53.9	72.5	**83.8**
PCNN	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	**86.1**	84.1	**86.45**	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
OURS	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

Table 6. Part segmentation results on ShapeNet part dataset. Metric is mIoU(%) on points.

Yields comparable performance as previous methods

Empirical Evaluation: Indoor Scene Segmentation

- **Dataset**: Stanford Large-Scale 3D Indoor Spaces Dataset (S3DIS)

  Includes 3D scan point clouds for 6 indoor areas including 272 rooms in total.

  Each point belongs to one of 13 semantic categories.

- **Evaluation Protocol**:

  Each room is split into blocks with area 1m × 1m, and each point is represented as a 9D vector (XYZ, RGB, and normalized spatial coordinates).

  4,096 points are sampled for each block during training. All points are used for testing.

  Use 6-fold cross validation over the 6 areas, and the average evaluation results are reported.

	Mean IoU	overall accuracy
PointNet (baseline) [Qi et al. 2017b]	20.1	53.2
PointNet [Qi et al. 2017b]	47.6	78.5
MS + CU(2) [Engelmann et al. 2017]	47.8	79.2
G + RCU [Engelmann et al. 2017]	49.7	81.1
PointCNN [Li et al. 2018a]	**65.39**	-
Ours	56.1	**84.1**

Beats all counterparts in overall accuracy
Ranks #2 in mean IoU

Qualitative Results

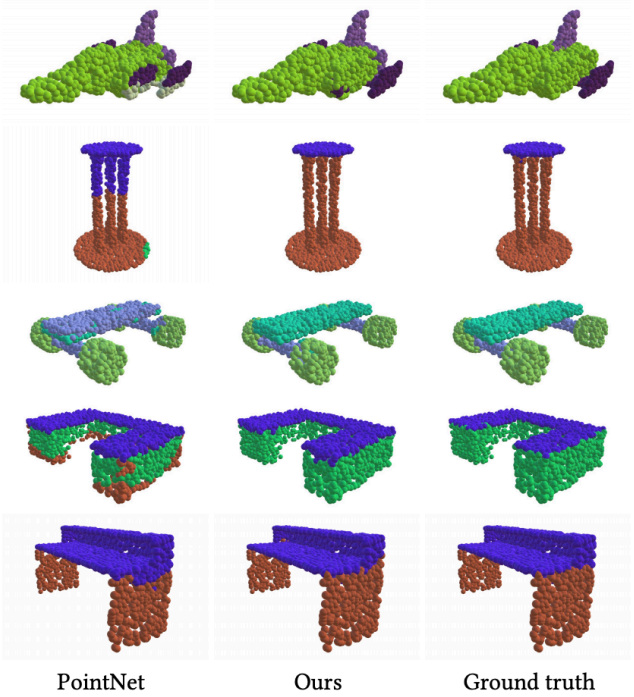
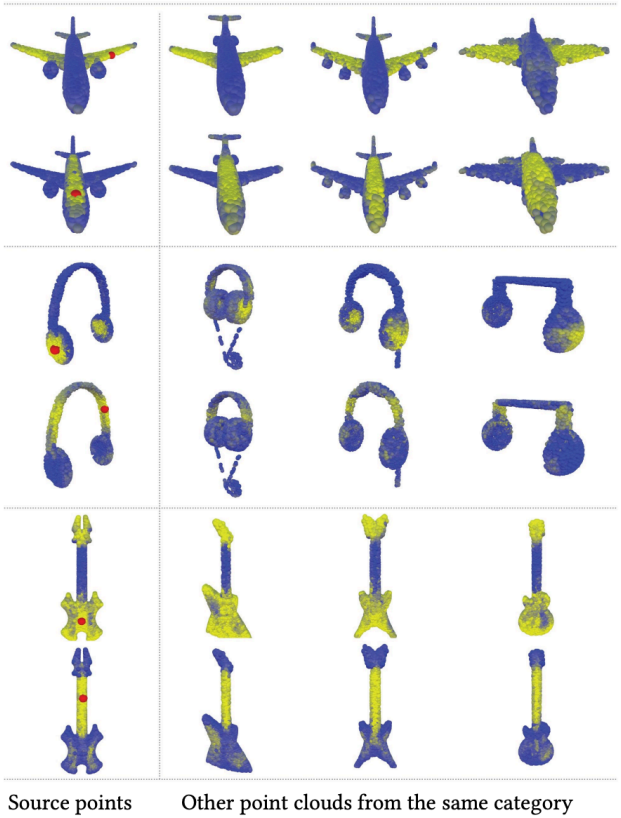


Fig. 7. Compare part segmentation results. For each set, from left to right: PointNet, ours and ground truth.

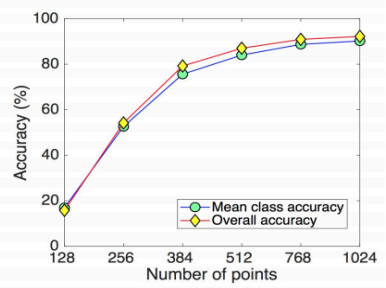


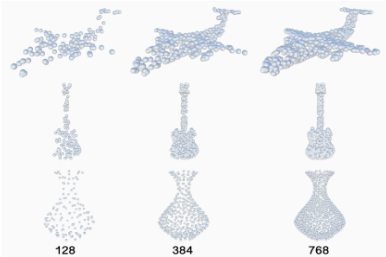
Conclusion from the Empirical Evidence

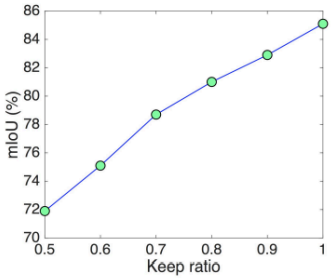
- **DGCNN is a powerful structure**, yielding state-of-the-art performance on popular benchmarks including ModelNet40, ShapeNet Part segmentation and S3DIS semantic segmentation.
- **Strengths**: Simple yet powerful

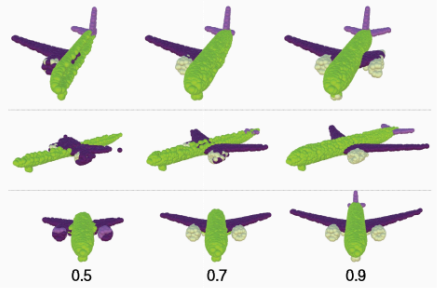
  Very high performance on ModelNet40
- **Weaknesses**: No clear advantage over PointNet++ on part segmentation (85.2 mIOU v.s. 85.1 mIOU)

  Lack comparison with baselines on the small accessory experiments







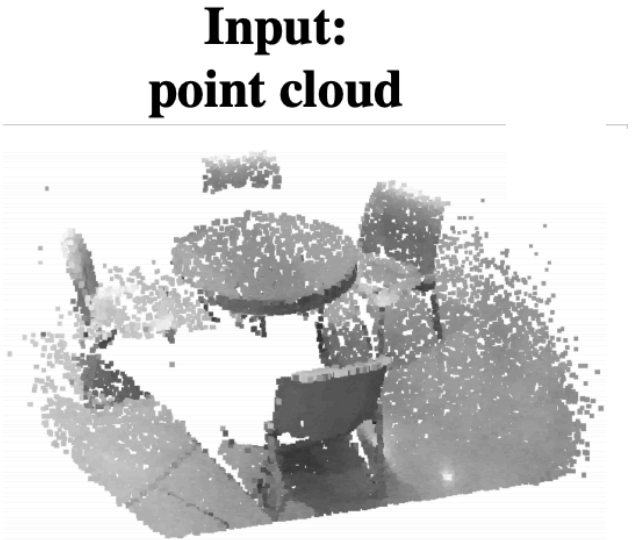


Limitations

- **GPU Memory Consumption**: Construct the graph requires to compute all pairwise distances between all the points --- very high GPU memory consumption

- **Varying Density**: Point clouds can have varying density at different positions;
  a fixed k may be not flexible enough

- **Large-scale Point Cloud**: DGCNN does not compress the number of points during forward pass



Possible Future Directions

- **From the ablation study, larger k do not always bring better performance**

  Possible direction: design a strategy (or by learning) to automatically find appropriate k for each point

Number of nearest neighbors (k)	Mean Class Accuracy(%)	Overall Accuracy(%)
5	88.0	90.5
10	88.9	91.4
20	90.2	92.9
40	89.4	92.4

Table 5. Results of our model with different numbers of nearest neighbors.

- **How to decrease the memory consumption?**

  Possible direction: Combing the grouping module in PointNet++ with DGCNN

- **Faster computation?**

  Possible direction: Maintain an octree for constructing graph

Existing Future Direction: KPConv

- DGCNN has high computational cost
- KPConv adopts convolution to gather local information
- Extracts local information for every point; saves computation by only considering a local ball
- Yields better results on segmentation tasks

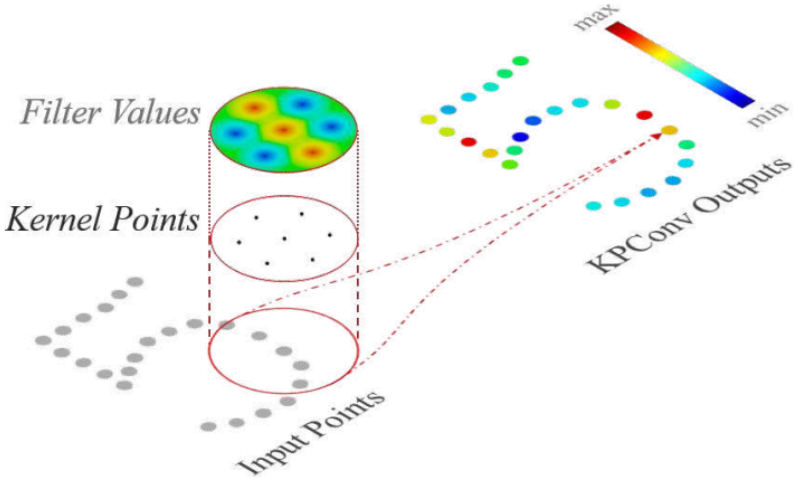


Figure 1. KPConv illustrated on 2D points. Input points with a constant scalar feature (in grey) are convolved through a KPConv that is defined by a set of kernel points (in black) with filter weights on each point.

Thomas H, Qi C R, Deschaud J E, et al. Kpconv: Flexible and deformable convolution for point clouds

Extended Readings: Point Cloud Processing

- **4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks, CVPR 2019**

  High performance point cloud processing with sparse convolution on voxelized point clouds

- **3D Semantic Segmentation with Submanifold Sparse Convolutional Networks, CVPR 2018**

  Another great work on point cloud processing with sparse convolution

- **RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds, CVPR 2020**

  Fast processing on large-scale point clouds by simple random sampling

Summary

- Problem: point cloud processing with deep networks
- A fundamental part of robot perception
- Point clouds are irregular, asking for specially designed neural networks
- Previous works fail to leverage the geometric structure in point clouds
- This paper proposes EdgeConv, and builds Dynamic Graph CNN with this module
- EdgeConv extracts local information for each point with dynamic graph
- DGCNN yields state-of-the-art performance on point cloud classification and segmentation

Thank You!

Questions?