Deep Learning (CNN) in 3D Point Cloud Processing

Yongcheng Liu
2019.04
Introduction
Introduction

**tasks**

-  lamp
-  shape classification
-  shape retrieval
-  keypoint detection
-  shape correspondence
-  semantic segmentation
-  object detection
-  normal estimation
Introduction datasets

Princeton ModelNet: 1k

ShapeNet Part: 2k

PartNet models

Hierarchical Semantic Segmentation


Wu et al. 3D ShapeNets: A Deep Representation for Volumetric Shapes. CVPR 2015.
Introduction

datasets

Stanford 3D indoor scene: 8k

Semantic 3D: 4 billion in total

ScanNet: seg + det

KITTI: det

Armeni et al. 3d semantic parsing of large-scale indoor spaces. CVPR 2016.
Introduction

**some challenges**

**Irregular (unordered): permutation invariance**

<table>
<thead>
<tr>
<th>f_a</th>
<th>f_b</th>
<th>f_c</th>
<th>f_d</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>i</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>f_a</th>
<th>f_b</th>
<th>f_c</th>
<th>f_d</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>ii</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>f_a</th>
<th>f_b</th>
<th>f_c</th>
<th>f_d</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>iii</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>f_a</th>
<th>f_b</th>
<th>f_c</th>
<th>f_d</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>iv</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Robustness to rigid transformations**

- **scale**
- **translation**
- **rotation**

**Robustness to corruption, outlier, noise; partial data**
Introduction 3D representations

- multi-view images + 2D CNN
- volumetric data + 3D CNN
- mesh data + DL (GNN)?
- image depth + CNN
- point cloud + DL (CNN)?

CAD model → Occupancy Grid 30x30x30
Related work – PointNet family
Related Work

PointNet: permutation invariance

Shared MLP + max pool (symmetric function)

No local patterns capturing

Related Work  \textit{PointNet++: local to global}

Hierarchical point set feature learning

Sampling + Grouping + PointNet

Related work – regularization
Related Work \textit{SEGCloud: voxelization}

Related Work

**SPLATNet: high-dimensional lattice**

Su et al. SPLATNet: Sparse Lattice Networks for Point Cloud Processing. CVPR 2018.
Related Work  

**Pointwise CNN: k-NN binned kernel**

\[
x^l_i = \sum_k w_k \frac{1}{|\Omega_i(k)|} \sum_{p_j \in \Omega_i(k)} x^{l-1}_j
\]

$s_{ik} = \text{kNN}(p_i \mid s_j, j = 0, \cdots, M - 1)$. 

$p_{ik} = p_i - s_{ik}$. 

$p_{ik}^{l+1} = \phi(W^l p_{ik}^l + b^l)$. 

$s_j^0 = \max\{p_{ik}^l, \forall s_{ik} = s_j\}$.
In this paper, we propose to learn a $K \times K \mathcal{X}$-transformation for the coordinates of $K$ input points $(p_1, p_2, ..., p_K)$, with a multilayer perceptron [39], i.e., $\mathcal{X} = \text{MLP}(p_1, p_2, ..., p_K)$. Our aim is to use it to simultaneously weight and permute the input features, and subsequently apply a typical convolution on the transformed features. We refer to this process as $\mathcal{X}$-Conv, and it is the Basic

**Algorithm 1: $\mathcal{X}$-Conv Operator**

<table>
<thead>
<tr>
<th>Input</th>
<th>$K$, $p$, $P$, $F$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output</td>
<td>$F_p$</td>
</tr>
<tr>
<td>1</td>
<td>$P' \leftarrow P - p$</td>
</tr>
<tr>
<td>2</td>
<td>$F_\delta \leftarrow \text{MLP}_\delta(P')$</td>
</tr>
<tr>
<td>3</td>
<td>$F_* \leftarrow [F_\delta, F]$</td>
</tr>
<tr>
<td>4</td>
<td>$\mathcal{X} \leftarrow \text{MLP}(P')$</td>
</tr>
<tr>
<td>5</td>
<td>$F_\mathcal{X} \leftarrow \mathcal{X} \times F_*$</td>
</tr>
<tr>
<td>6</td>
<td>$F_p \leftarrow \text{Conv}(K, F_\mathcal{X})$</td>
</tr>
</tbody>
</table>

- Features “projected”, or “aggregated”, into representative point $p$
- Move $P$ to local coordinate system of $p$
- **Individually** lift each point into $C_\delta$ dimensional space
- Concatenate $F_\delta$ and $F$, $F_*$ is a $K \times (C_\delta + C_1)$ matrix
- Learn the $K \times K \mathcal{X}$-transformation matrix
- Weight and permute $F_*$ with the learnt $\mathcal{X}$
- Finally, typical convolution between $K$ and $F_\mathcal{X}$
Related Work  **PointSift: SIFT-like network**

Figure 4: Illustration of the details in Orientation-encoding Point Convolutional layer. (a): point clouds in 3D space. (b) neighbors in eight directions. (c) three stages convolution combines all the features.

**SIFT:**
- orientation-encoding
- scale-awareness (shortcut connections)
Related work – robustness to rigid transformation
Related Work

Normalization:

✓ Translation
✓ Scale
✓ Rotation

\[ X \cdot R, (N \times 3) \cdot (3 \times 3) \]
Related Work

**SFCNN: Spherical Fractal CNN**

Related work – relation modeling


**Related Work**  

**DGCNN**

Points in high-level feature space captures semantically similar structures.

Despite a large distance between them in the original 3D space.
Related Work  **DGCNN**

$$h_\Theta(x_i, x_j - x_i)$$

- Neighbors are found in feature space
- Learn from semantically similar structures

• Relation modeling: self-attention
• Gumbel Subset Sampling VS. Farthest Point Sampling
  — permutation-invariant
  — high-dimension embedding space
  — differentiable
Related Work

**self-attention**

Embedding: PointNet

\[ X_p' = \{(x_p, x_i - x_p) \mid i \neq p\} \]

Self-attention:

group convolution + channel shuffle + pre-activation

Related Work \textit{self-attention}

\[ X_i \in \mathbb{R}^{N_i \times c} \]
\[ X_{i+1} \in \mathbb{R}^{N_{i+1} \times c} \subseteq X_i \]

Gumbel Subset Sampling:

\[ y = \text{softmax}(wX_i^T) \cdot X_i, \quad w \in \mathbb{R}^c. \]

\[ y_{\text{gumbel}} = \text{gumbel}_\text{softmax}(wX_i^T) \cdot X_i, \quad w \in \mathbb{R}^c. \]

\[ GSS(X_i) = \text{gumbel}_\text{softmax}(W X_i^T) \cdot X_i, \quad W \in \mathbb{R}^{N_{i+1} \times c}. \]

Related work – convolution on point cloud
$$(\mathcal{F} * g)(x) = \sum_{x_i \in \mathcal{N}_x} g(x_i - x) f_i$$

$$y_i = x_i - x$$

$$\mathcal{B}_r^3 = \{ y \in \mathbb{R}^3 \mid \|y\| \leq r \}$$

$$g(y_i) = \sum_{k < K} h(y_i, \tilde{x}_k) W_k$$

**Kernel Points**

$$h(y_i, \tilde{x}_k) = \max \left( 0, 1 - \frac{\|y_i - \tilde{x}_k\|}{\sigma} \right)$$

**Related Work**  

**Kernel Point Convolution**

repulsive potential:

\[
\forall x \in \mathbb{R}^3, \quad E_{k}^{rep}(x) = \frac{1}{\|x - \tilde{x}_k\|}
\]

attractive potential:

\[
\forall x \in \mathbb{R}^3, \quad E^{att}(x) = \|x\|^2
\]

\[
E^{tot} = \sum_{k<K} \left( E^{att}(\tilde{x}_k) + \sum_{l \neq k} E_{k}^{rep}(\tilde{x}_l) \right)
\]

Related Work  ** Kernel Point Convolution**

Rigid: \( (\mathcal{F} * g)(x) = \sum_{x_i \in \mathcal{N}_x} g(x_i - x) f_i \)

\[ g(y_i) = \sum_{k < K} h(y_i, \tilde{x}_k) W_k \]

Deformable: fit the local geometry

\( (\mathcal{F} * g)(x) = \sum_{x_i \in \mathcal{N}_x} g_{\text{deform}}(x - x_i, \Delta(x)) f_i \)

\[ g_{\text{deform}}(y_i, \Delta(x)) = \sum_{k < K} h(y_i, \tilde{x}_k + \Delta_k(x)) W_k \]

Github: awesome-point-cloud-analysis

- Recent papers (from 2017)

Keywords

det.: detection  |  tra.: tracking  |  pos.: pose  |  dep.: depth
reg.: registration  |  recon.: reconstruction  |  aut.: autonomous driving
oth.: other, including normal-related, correspondence, mapping, matching, alignment, compression...

2017

- [CVPR] PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. [tensorflow][pytorch] [cls., seg., det.]
- [CVPR] Dynamic Edge-Conditioned Filters in Convolutional Neural Networks on Graphs. [cls.]
- [CVPR] SyncSpecCNN: Synchronized Spectral CNN for 3D Shape Segmentation. [torch] [seg., oth.]
- [CVPR] ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes. [project][git] [dat., cls., rel., seg., oth.]
- [CVPR] Scalable Surface Reconstruction from Point Clouds with Extreme Scale and Density Diversity. [oth.]
- [CVPR] Efficient Global Point Cloud Alignment using Bayesian Nonparametric Mixtures. [code] [oth.]
- [CVPR] Discriminative Optimization: Theory and Applications to Point Cloud Registration. [reg.]
- [CVPR] 3D Point Cloud Registration for Localization using a Deep Neural Network Auto-Encoder. [git] [reg.]
Relation-Shape Convolutional Neural Network for Point Cloud Analysis (RS-CNN)

Yongcheng Liu, Bin Fan, Shiming Xiang, Chunhong Pan

CVPR 2019  Oral Presentation

Project Page: https://yochengliu.github.io/Relation-Shape-CNN/
RS-CNN  

Motivation

2D image  

3D point cloud

3D Shape Learning

Relation Learning

Deep Learning (CNN)
**RS-CNN Method**

Relation-Shape Convolution (RS-Conv)

- Local point subset $P_{\text{sub}} \subset \mathbb{R}^3$ → spherical neighborhood: $x_i + x_j \in \mathcal{N}(x_i)$

$$f_{P_{\text{sub}}} = \sigma(A(\{\mathcal{T}(f_{x_j}), \forall x_j\}))^1, \quad d_{ij} < r \quad \forall x_j \in \mathcal{N}(x_i) \quad y = \sigma(\sum W \ast X)$$

$\mathcal{T}$: feature transformation  \hspace{1cm} $A$: feature aggregation

- Permutation invariance: only when $A$ is symmetric and $\mathcal{T}$ is shared over each point
- Limitations of CNN: weight is not shared  
  \hspace{1cm} gradient only w.r.t single point - implicit

$$\mathcal{T}(f_{x_j}) = w_j \cdot f_{x_j}$$

- Conversion: learn from relation  
  \hspace{1cm} $\mathcal{T}(f_{x_j}) = w_{ij} \cdot f_{x_j} = \mathcal{M}(h_{ij}) \cdot f_{x_j}$

$h_{ij}$: predefined geometric priors → low-level relation

$$f_{P_{\text{sub}}} = \sigma(A(\{\mathcal{M}(h_{ij}) \cdot f_{x_j}, \forall x_j\}))) \quad \mathcal{M}$: mapping function(shared MLP) → high-level relation
**RS-CNN Method**

High-level relation encoding $+$ channel raising mapping

Low-level relation $h_{ij}$: (3D Euclidean distance, $x_i - x_j$, $x_i$, $x_j$) 10 channels
RS-CNN  \textbf{RS-Conv: Properties}

\[ f_{P_{\text{sub}}} = \sigma \left( A(\{ M(h_{i,j}) \cdot f_{x_j}, \forall x_j \}) \right) \]

- Permutation invariance
- Robustness to rigid transformation in Relation Learning, e.g., 3D Euclidean distance
- Points’ interaction
- Weight sharing

Revisiting 2D Conv:

RS-Conv with relation learning is more general and can be applied to model 2D grid spatial relationship.
Farthest Point Sampling + Sphere Neighborhood + RS-Conv
RS-CNN  **Shape classification**

ModelNet40 benchmark

Robustness to sampling density

<table>
<thead>
<tr>
<th>method</th>
<th>input</th>
<th>#points</th>
<th>acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pointwise-CNN [10]</td>
<td>xyz</td>
<td>1k</td>
<td>86.1</td>
</tr>
<tr>
<td>Deep Sets [48]</td>
<td>xyz</td>
<td>1k</td>
<td>87.1</td>
</tr>
<tr>
<td>ECC [31]</td>
<td>xyz</td>
<td>1k</td>
<td>87.4</td>
</tr>
<tr>
<td>PointNet [24]</td>
<td>xyz</td>
<td>1k</td>
<td>89.2</td>
</tr>
<tr>
<td>SCN [44]</td>
<td>xyz</td>
<td>1k</td>
<td>90.0</td>
</tr>
<tr>
<td>Kd-Net(depth=10) [16]</td>
<td>xyz</td>
<td>1k</td>
<td>90.6</td>
</tr>
<tr>
<td>PointNet++ [26]</td>
<td>xyz</td>
<td>1k</td>
<td>90.7</td>
</tr>
<tr>
<td>KCNet [30]</td>
<td>xyz</td>
<td>1k</td>
<td>91.0</td>
</tr>
<tr>
<td>MRTNet [3]</td>
<td>xyz</td>
<td>1k</td>
<td>91.2</td>
</tr>
<tr>
<td>Spec-GCN [38]</td>
<td>xyz</td>
<td>1k</td>
<td>91.5</td>
</tr>
<tr>
<td>PointCNN [21]</td>
<td>xyz</td>
<td>1k</td>
<td>91.7</td>
</tr>
<tr>
<td>DGCNN [41]</td>
<td>xyz</td>
<td>1k</td>
<td>92.2</td>
</tr>
<tr>
<td>PCNN [1]</td>
<td>xyz</td>
<td>1k</td>
<td>92.3</td>
</tr>
<tr>
<td><strong>Ours</strong></td>
<td>xyz</td>
<td>1k</td>
<td><strong>93.6</strong></td>
</tr>
<tr>
<td>SO-Net [19]</td>
<td>xyz</td>
<td>2k</td>
<td>90.9</td>
</tr>
<tr>
<td>Kd-Net(depth=15) [16]</td>
<td>xyz</td>
<td>32k</td>
<td>91.8</td>
</tr>
<tr>
<td>O-CNN [39]</td>
<td>xyz, nor</td>
<td>-</td>
<td>90.6</td>
</tr>
<tr>
<td>Spec-GCN [38]</td>
<td>xyz, nor</td>
<td>1k</td>
<td>91.8</td>
</tr>
<tr>
<td>PointNet++ [26]</td>
<td>xyz, nor</td>
<td>5k</td>
<td>91.9</td>
</tr>
<tr>
<td>SpiderCNN [45]</td>
<td>xyz, nor</td>
<td>5k</td>
<td>92.4</td>
</tr>
<tr>
<td>SO-Net [19]</td>
<td>xyz, nor</td>
<td>5k</td>
<td>93.4</td>
</tr>
</tbody>
</table>
RS-CNN  
*ShapePart Segmentation*

<table>
<thead>
<tr>
<th>method</th>
<th>input</th>
<th>class mIoU</th>
<th>instance mIoU</th>
<th>air</th>
<th>bag</th>
<th>cap</th>
<th>car</th>
<th>chair</th>
<th>ear</th>
<th>guitar</th>
<th>knife</th>
<th>lamp</th>
<th>laptop</th>
<th>motor</th>
<th>mug</th>
<th>pistol</th>
<th>rocket</th>
<th>skate</th>
<th>table</th>
<th>board</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kd-Net [16]</td>
<td>4k</td>
<td>77.4</td>
<td>82.3</td>
<td>80.1</td>
<td>74.6</td>
<td>74.3</td>
<td>70.3</td>
<td>88.6</td>
<td>73.5</td>
<td>90.2</td>
<td>87.2</td>
<td>81.0</td>
<td>94.9</td>
<td>57.4</td>
<td>86.7</td>
<td>78.1</td>
<td>51.8</td>
<td>69.9</td>
<td>80.3</td>
<td></td>
</tr>
<tr>
<td>PointNet [24]</td>
<td>2k</td>
<td>80.4</td>
<td>83.7</td>
<td>83.4</td>
<td>78.7</td>
<td>82.5</td>
<td>74.9</td>
<td>89.6</td>
<td>73.0</td>
<td>91.5</td>
<td>85.9</td>
<td>80.8</td>
<td>95.3</td>
<td>65.2</td>
<td>93.0</td>
<td>81.2</td>
<td>57.9</td>
<td>72.8</td>
<td>80.6</td>
<td></td>
</tr>
<tr>
<td>RS-Net [11]</td>
<td>-</td>
<td>81.4</td>
<td>84.9</td>
<td>82.7</td>
<td><strong>86.4</strong></td>
<td>84.1</td>
<td>78.2</td>
<td>90.4</td>
<td>69.3</td>
<td>91.4</td>
<td>87.0</td>
<td>83.5</td>
<td>95.4</td>
<td>66.0</td>
<td>92.6</td>
<td>81.8</td>
<td>56.1</td>
<td>75.8</td>
<td>82.2</td>
<td></td>
</tr>
<tr>
<td>SCN [44]</td>
<td>1k</td>
<td>81.8</td>
<td>84.6</td>
<td>83.8</td>
<td>80.8</td>
<td>83.5</td>
<td>79.3</td>
<td>90.5</td>
<td>69.8</td>
<td><strong>91.7</strong></td>
<td>86.5</td>
<td>82.9</td>
<td>96.0</td>
<td>69.2</td>
<td>93.8</td>
<td>82.5</td>
<td>62.9</td>
<td>74.4</td>
<td>80.8</td>
<td></td>
</tr>
<tr>
<td>PCNN [1]</td>
<td>2k</td>
<td>81.8</td>
<td>85.1</td>
<td>82.4</td>
<td>80.1</td>
<td>85.5</td>
<td>79.5</td>
<td>90.8</td>
<td>73.2</td>
<td>91.3</td>
<td>86.0</td>
<td>85.0</td>
<td>95.7</td>
<td>73.2</td>
<td>94.8</td>
<td>83.3</td>
<td>51.0</td>
<td>75.0</td>
<td>81.8</td>
<td></td>
</tr>
<tr>
<td>SPLATNet [34]</td>
<td>-</td>
<td>82.0</td>
<td>84.6</td>
<td>81.9</td>
<td>83.9</td>
<td>88.6</td>
<td>79.5</td>
<td>90.1</td>
<td>73.5</td>
<td>91.3</td>
<td>84.7</td>
<td>84.5</td>
<td><strong>96.3</strong></td>
<td>69.7</td>
<td>95.0</td>
<td>81.7</td>
<td>59.2</td>
<td>70.4</td>
<td>81.3</td>
<td></td>
</tr>
<tr>
<td>KCNet [30]</td>
<td>2k</td>
<td>82.2</td>
<td>84.7</td>
<td>82.8</td>
<td>81.5</td>
<td>86.4</td>
<td>77.6</td>
<td>90.3</td>
<td>76.8</td>
<td>91.0</td>
<td>87.2</td>
<td>84.5</td>
<td>95.5</td>
<td>69.2</td>
<td>94.4</td>
<td>81.6</td>
<td>60.1</td>
<td>75.2</td>
<td>81.3</td>
<td></td>
</tr>
<tr>
<td>DGCNN [41]</td>
<td>2k</td>
<td>82.3</td>
<td>85.1</td>
<td><strong>84.2</strong></td>
<td>83.7</td>
<td>84.4</td>
<td>77.1</td>
<td>90.9</td>
<td>78.5</td>
<td>91.5</td>
<td>87.3</td>
<td>82.9</td>
<td>96.0</td>
<td>67.8</td>
<td>93.3</td>
<td>82.6</td>
<td>59.7</td>
<td>75.5</td>
<td>82.0</td>
<td></td>
</tr>
<tr>
<td><strong>Ours</strong></td>
<td><strong>2k</strong></td>
<td><strong>84.0</strong></td>
<td><strong>86.2</strong></td>
<td>83.5</td>
<td>84.8</td>
<td><strong>88.8</strong></td>
<td><strong>79.6</strong></td>
<td><strong>91.2</strong></td>
<td><strong>81.1</strong></td>
<td>91.6</td>
<td><strong>88.4</strong></td>
<td><strong>86.0</strong></td>
<td>96.0</td>
<td><strong>73.7</strong></td>
<td>94.1</td>
<td><strong>83.4</strong></td>
<td>60.5</td>
<td><strong>77.7</strong></td>
<td><strong>83.6</strong></td>
<td></td>
</tr>
<tr>
<td>PointNet++ [26]</td>
<td>2k,nor</td>
<td>81.9</td>
<td>85.1</td>
<td>82.4</td>
<td>79.0</td>
<td>87.7</td>
<td>77.3</td>
<td>90.8</td>
<td>71.8</td>
<td>91.0</td>
<td>85.9</td>
<td>83.7</td>
<td>95.3</td>
<td>71.6</td>
<td>94.1</td>
<td>81.3</td>
<td>58.7</td>
<td>76.4</td>
<td>82.6</td>
<td></td>
</tr>
<tr>
<td>SyncCNN [47]</td>
<td>mesh</td>
<td>82.0</td>
<td>84.7</td>
<td>81.6</td>
<td>81.7</td>
<td>81.9</td>
<td>75.2</td>
<td>90.2</td>
<td>74.9</td>
<td>93.0</td>
<td>86.1</td>
<td>84.7</td>
<td>95.6</td>
<td>66.7</td>
<td>92.7</td>
<td>81.6</td>
<td>60.6</td>
<td>82.9</td>
<td>82.1</td>
<td></td>
</tr>
<tr>
<td>SO-Net [19]</td>
<td>1k,nor</td>
<td>80.8</td>
<td>84.6</td>
<td>81.9</td>
<td>83.5</td>
<td>84.8</td>
<td>78.1</td>
<td>90.8</td>
<td>72.2</td>
<td>90.1</td>
<td>83.6</td>
<td>82.3</td>
<td>95.2</td>
<td>69.3</td>
<td>94.2</td>
<td>80.0</td>
<td>51.6</td>
<td>72.1</td>
<td>82.6</td>
<td></td>
</tr>
<tr>
<td>SpiderCNN [45]</td>
<td>2k,nor</td>
<td>82.4</td>
<td>85.3</td>
<td>83.5</td>
<td>81.0</td>
<td>87.2</td>
<td>77.5</td>
<td>90.7</td>
<td>76.8</td>
<td>91.1</td>
<td>87.3</td>
<td>83.3</td>
<td>95.8</td>
<td>70.2</td>
<td>93.5</td>
<td>82.7</td>
<td>59.7</td>
<td>75.8</td>
<td>82.8</td>
<td></td>
</tr>
</tbody>
</table>

**class mIoU 1.7↑  instance mIoU 1.1↑**

Best results over 10 categories
RS-CNN  ShapePart Segmentation

Diverse, confusing shapes
RS-CNN  

Normal estimation

Table 3. Normal estimation error on ModelNet40 dataset.

<table>
<thead>
<tr>
<th>dataset</th>
<th>method</th>
<th>#points</th>
<th>error</th>
</tr>
</thead>
<tbody>
<tr>
<td>ModelNet40</td>
<td>PointNet [1]</td>
<td>1k</td>
<td>0.47</td>
</tr>
<tr>
<td></td>
<td>PointNet++ [1]</td>
<td>1k</td>
<td>0.29</td>
</tr>
<tr>
<td></td>
<td>PCNN [1]</td>
<td>1k</td>
<td>0.19</td>
</tr>
<tr>
<td></td>
<td>Ours</td>
<td>1k</td>
<td>0.15</td>
</tr>
</tbody>
</table>

less effective for some intractable shapes, such as spiral stairs and intricate plants
**RS-CNN**  
*Geometric priors*

\[ f_{P_{sub}} = \sigma (A(\{ \mathcal{M}(h_{ij}) \cdot f_{x_j}, \forall x_j \})) \]

<table>
<thead>
<tr>
<th>model</th>
<th>low-level relation ( h )</th>
<th>channels</th>
<th>acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>(3D-Ed)</td>
<td>1</td>
<td>92.5</td>
</tr>
<tr>
<td>B</td>
<td>(3D-Ed, ( x_i - x_j ))</td>
<td>4</td>
<td>93.0</td>
</tr>
<tr>
<td>C</td>
<td>(3D-Ed, ( x_i - x_j, x_i, x_j ))</td>
<td>10</td>
<td><strong>93.6</strong></td>
</tr>
<tr>
<td>D</td>
<td>(3D-cosd, ( x_i^{nor}, x_j^{nor} ))</td>
<td>7</td>
<td>92.8</td>
</tr>
<tr>
<td>E</td>
<td>(2D-Ed, ( x_i' - x_j', x_i, x_j' ))</td>
<td>10</td>
<td>( \approx 92.2 )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>low-level relation ( h )</th>
<th>channels</th>
<th>acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>(XY-Ed, ( x_{i}^{xy} - x_{j}^{xy}, x_{i}^{xy}, x_{j}^{xy} ))</td>
<td>10</td>
<td>92.1</td>
</tr>
<tr>
<td>(XY-Ed, ( x_{i}^{xz} - x_{j}^{xz}, x_{i}^{xz}, x_{j}^{xz} ))</td>
<td>10</td>
<td>92.1</td>
</tr>
<tr>
<td>(XY-Ed, ( x_{i}^{yz} - x_{j}^{yz}, x_{i}^{yz}, x_{j}^{yz} ))</td>
<td>10</td>
<td>92.2</td>
</tr>
</tbody>
</table>

**fusion of above three views**  
92.5
Robustness to point permutation and rigid transformation

<table>
<thead>
<tr>
<th>method</th>
<th>acc.</th>
<th>perm.</th>
<th>+0.2</th>
<th>-0.2</th>
<th>90°</th>
<th>180°</th>
</tr>
</thead>
<tbody>
<tr>
<td>PointNet [24]</td>
<td>88.7</td>
<td>88.7</td>
<td>70.8</td>
<td>70.6</td>
<td>42.5</td>
<td>38.6</td>
</tr>
<tr>
<td>PointNet++ [26]</td>
<td>88.2</td>
<td>88.2</td>
<td>88.2</td>
<td>88.2</td>
<td>47.9</td>
<td>39.7</td>
</tr>
<tr>
<td><strong>Ours</strong></td>
<td><strong>90.3</strong></td>
<td><strong>90.3</strong></td>
<td><strong>90.3</strong></td>
<td><strong>90.3</strong></td>
<td><strong>90.3</strong></td>
<td><strong>90.3</strong></td>
</tr>
</tbody>
</table>

\[
f_{P_{sub}} = \sigma(\mathcal{A}(\mathcal{M}(h_{ij}) \cdot f_{x_j}, \forall x_j))
\]

Model complexity

<table>
<thead>
<tr>
<th>method</th>
<th>#params</th>
<th>#FLOPs/sample</th>
</tr>
</thead>
<tbody>
<tr>
<td>PointNet [24]</td>
<td>3.50M</td>
<td>440M</td>
</tr>
<tr>
<td>PointNet++ [21]</td>
<td>1.48M</td>
<td>1684M</td>
</tr>
<tr>
<td>PCNN [21]</td>
<td>8.20M</td>
<td><strong>294M</strong></td>
</tr>
<tr>
<td><strong>Ours</strong></td>
<td><strong>1.41M</strong></td>
<td><strong>295M</strong></td>
</tr>
</tbody>
</table>
Relation-Shape Convolutional Neural Network for Point Cloud Analysis

We propose a learn-from-relation convolution operator, which extends 2D CNN to irregular configuration for point cloud analysis.
Thanks for your attention !