Point-Voxel CNN for Efficient 3D Deep Learning

Zhijian Liu*, Haotian Tang*, Yujun Lin, and Song Han

NeurIPS 2019 (Spotlight)
Applications of 3D Deep Learning

3D Part Segmentation
(for Object)

3D Semantic Segmentation
(for Indoor Scene)

3D Object Detection
(for Autonomous Driving)
Memory Operations are Expensive

Off-chip DRAM access is much more expensive than arithmetic operation!

Random memory access is inefficient due to the potential bank conflicts!

Efficient 3D deep learning should have small memory footprints and avoid random memory access.
Voxel-Based Models are Non-Scalable

Low resolutions lead to significant information loss; high resolutions lead to large GPU memory consumption.

3D ShapeNets [CVPR’15]
VoxNet [IROS’15]
3D U-Net [MICCAI’16]
Point-Based Models Introduce Sparse Overheads

PointNet [CVPR’17]
PointCNN [NeurIPS’18]
PointNet++ [NIPS’17]
DGCNN [SIGGRAPH’19]

Up to 80% of the time is wasted on structuring the irregular data, not on the actual feature extraction.
Our Solution: Point-Voxel Convolution

Point-Based Feature Transformation (Fine-Grained)

Voxel-Based Feature Aggregation (Coarse-Grained)

We combine the advantages of point-based methods (small memory footprint) and voxel-based methods (regularity).
## Results on Object Part Segmentation

<table>
<thead>
<tr>
<th></th>
<th>Input Data</th>
<th>Convolution</th>
<th>Mean IoU</th>
<th>Latency</th>
<th>GPU Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>PointNet [32]</td>
<td>points (8×2048)</td>
<td>none</td>
<td>83.7</td>
<td>21.7 ms</td>
<td>1.5 GB</td>
</tr>
<tr>
<td>3D-UNet [52]</td>
<td>voxels (8×96³)</td>
<td>volumetric</td>
<td>84.6</td>
<td>682.1 ms</td>
<td>8.8 GB</td>
</tr>
<tr>
<td>RSNet [14]</td>
<td>points (8×2048)</td>
<td>point-based</td>
<td>84.9</td>
<td>74.6 ms</td>
<td>0.8 GB</td>
</tr>
<tr>
<td>PointNet++ [34]</td>
<td>points (8×2048)</td>
<td>point-based</td>
<td>85.1</td>
<td>77.9 ms</td>
<td>2.0 GB</td>
</tr>
<tr>
<td>DGCNN [44]</td>
<td>points (8×2048)</td>
<td>point-based</td>
<td>85.1</td>
<td>87.8 ms</td>
<td>2.4 GB</td>
</tr>
<tr>
<td><strong>PVCNN (Ours, 0.25×C)</strong></td>
<td>points (8×2048)</td>
<td>volumetric</td>
<td><strong>85.2</strong></td>
<td><strong>11.6 ms</strong></td>
<td><strong>0.8 GB</strong></td>
</tr>
<tr>
<td>SpiderCNN [47]</td>
<td>points (8×2048)</td>
<td>point-based</td>
<td>85.3</td>
<td>170.7 ms</td>
<td>6.5 GB</td>
</tr>
<tr>
<td><strong>PVCNN (Ours, 0.5×C)</strong></td>
<td>points (8×2048)</td>
<td>volumetric</td>
<td><strong>85.5</strong></td>
<td><strong>21.7 ms</strong></td>
<td><strong>1.0 GB</strong></td>
</tr>
<tr>
<td>PointCNN [24]</td>
<td>points (8×2048)</td>
<td>point-based</td>
<td>86.1</td>
<td>135.8 ms</td>
<td>2.5 GB</td>
</tr>
<tr>
<td><strong>PVCNN (Ours, 1×C)</strong></td>
<td>points (8×2048)</td>
<td>volumetric</td>
<td><strong>86.2</strong></td>
<td><strong>50.7 ms</strong></td>
<td><strong>1.6 GB</strong></td>
</tr>
</tbody>
</table>

Table 1: Results of object part segmentation on ShapeNet Part [4]. On average, PVCNN outperforms the point-based models with 5.5× measured speedup and 3× memory reduction, and outperforms the voxel-based baseline with 59× measured speedup and 11× memory reduction.
Results on Object Part Segmentation

![Graph showing performance vs latency and GPU memory usage for different models.]

- PVCNN
- PointCNN
- DGCNN
- RSNet
- 3D-UNet
- SpiderCNN
- PointNet++
- PointNet

2.7x speedup
1.5x reduction
Results on Object Part Segmentation

- PointCNN (86.1 mIoU) - 1.0 PVCNN (86.2 mIoU)
- PointNet (83.7 mIoU) - 0.25 PVCNN (85.2 mIoU)

Jetson Nano
- Objects per Second: 1.4, 2.5, 1.4

Jetson TX2
- Objects per Second: 3.3, 7.7, 19.9

Jetson AGX Xavier
- Objects per Second: 9.5, 20.2, 42.6

Jetson AGX Xavier
- Objects per Second: 76.0, 139.9
Results on Object Part Segmentation

Features from Point Branch

Features from Voxel Branch

The point branch focuses on the fine details; while the voxel branch focuses on the large contiguous region.
## Results on Indoor Semantic Segmentation

<table>
<thead>
<tr>
<th>Model</th>
<th>Input Data</th>
<th>Convolution Type</th>
<th>mAcc</th>
<th>mIoU</th>
<th>Latency</th>
<th>GPU Mem.</th>
</tr>
</thead>
<tbody>
<tr>
<td>PointNet [32]</td>
<td>points (8×4096)</td>
<td>none</td>
<td>82.54</td>
<td>42.97</td>
<td>20.9 ms</td>
<td>1.0 GB</td>
</tr>
<tr>
<td>PVCNN (Ours, 0.125×C)</td>
<td>points (8×4096)</td>
<td>volumetric</td>
<td><strong>82.60</strong></td>
<td><strong>46.94</strong></td>
<td><strong>8.5 ms</strong></td>
<td><strong>0.6 GB</strong></td>
</tr>
<tr>
<td>DGCNN [44]</td>
<td>points (8×4096)</td>
<td>point-based</td>
<td>83.64</td>
<td>47.94</td>
<td>178.1 ms</td>
<td>2.4 GB</td>
</tr>
<tr>
<td>RSNet [14]</td>
<td>points (8×4096)</td>
<td>point-based</td>
<td>–</td>
<td>51.93</td>
<td>111.5 ms</td>
<td>1.1 GB</td>
</tr>
<tr>
<td>PVCNN (Ours, 0.25×C)</td>
<td>points (8×4096)</td>
<td>volumetric</td>
<td><strong>85.25</strong></td>
<td><strong>52.25</strong></td>
<td><strong>11.9 ms</strong></td>
<td><strong>0.7 GB</strong></td>
</tr>
<tr>
<td>3D-UNet [52]</td>
<td>voxels (8×96³)</td>
<td>volumetric</td>
<td>86.12</td>
<td>54.93</td>
<td>574.7 ms</td>
<td>6.8 GB</td>
</tr>
<tr>
<td>PVCNN (Ours, 1×C)</td>
<td>points (8×4096)</td>
<td>volumetric</td>
<td>86.66</td>
<td>56.12</td>
<td>47.3 ms</td>
<td>1.3 GB</td>
</tr>
<tr>
<td>PVCNN++ (Ours, 0.5×C)</td>
<td>points (4×8192)</td>
<td>volumetric</td>
<td><strong>86.87</strong></td>
<td><strong>57.63</strong></td>
<td><strong>41.1 ms</strong></td>
<td><strong>0.7 GB</strong></td>
</tr>
<tr>
<td>PointCNN [24]</td>
<td>points (16×2048)</td>
<td>point-based</td>
<td>85.91</td>
<td>57.26</td>
<td>282.3 ms</td>
<td>4.6 GB</td>
</tr>
<tr>
<td>PVCNN++ (Ours, 1×C)</td>
<td>points (4×8192)</td>
<td>volumetric</td>
<td><strong>87.12</strong></td>
<td><strong>58.98</strong></td>
<td><strong>69.5 ms</strong></td>
<td><strong>0.8 GB</strong></td>
</tr>
</tbody>
</table>

Table 3: Results of indoor scene segmentation on S3DIS [1, 2]. On average, PVCNN and PVCNN++ outperforms the point-based models with 8× measured speedup and 3× memory reduction, and outperforms the voxel-based baseline with 14× measured speedup and 10× memory reduction.
Results on Indoor Semantic Segmentation

- PVCNN
- 3D-UNet
- RSNet
- PointNet
- PVCNN++
- PointCNN
- DGCNN

Graph 1:
- X-axis: Latency (ms)
- Y-axis: Mean IoU
- 6.9x speedup

Graph 2:
- X-axis: GPU Memory (GB)
- Y-axis: Mean IoU
- 5.7x reduction
Input Scene

PointNet
Time: 1.9 sec per scene
GPU Memory: 1.9 GB

PVCNN (Ours)
Time: 1.0 sec per scene
GPU Memory: 1.2 GB

Ground Truth

Time: 1.9 sec per scene
GPU Memory: 1.9 GB

Time: 1.0 sec per scene
GPU Memory: 1.2 GB
Input Scene

Ground Truth

PointNet
Time: **1.9 sec** per scene
GPU Memory: **1.9 GB**

PVCNN (Ours)
Time: **1.0 sec** per scene
GPU Memory: **1.2 GB**
Input Scene

Ground Truth

PointNet
Time: 1.9 sec per scene
GPU Memory: 1.9 GB

PVCNN (Ours)
Time: 1.0 sec per scene
GPU Memory: 1.2 GB
## Results on Object Detection for Driving

<table>
<thead>
<tr>
<th></th>
<th>Efficiency</th>
<th>Car</th>
<th>Pedestrian</th>
<th>Cyclist</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Latency</td>
<td>GPU Mem.</td>
<td>Easy</td>
<td>Mod.</td>
</tr>
<tr>
<td>F-PointNet [31]</td>
<td>29.1 ms</td>
<td>1.3 GB</td>
<td>83.26</td>
<td>69.28</td>
</tr>
<tr>
<td>F-PointNet++ [31]</td>
<td>105.2 ms</td>
<td>2.0 GB</td>
<td>83.76</td>
<td>70.92</td>
</tr>
<tr>
<td>PVCNN (efficient)</td>
<td>58.9 ms</td>
<td>1.4 GB</td>
<td><strong>84.22</strong></td>
<td>71.11</td>
</tr>
<tr>
<td>PVCNN (complete)</td>
<td>69.6 ms</td>
<td>1.4 GB</td>
<td>84.02</td>
<td><strong>71.54</strong></td>
</tr>
</tbody>
</table>

Table 4: Results of 3D object detection on the *val* set of KITTI [6]. The *complete* PVCNN outperforms F-PointNet++ in all categories significantly with $1.5 \times$ measured speedup and memory reduction.
F-PointNet++

Time: 105.2 ms per video
GPU Memory: 9.2 GB
PVCNN (Ours)

Time: **58.9 ms** per video

GPU Memory: **4.0 GB**
Summary

• We systemically analyze the computational bottlenecks of point and voxel-based models.

• We propose an efficient convolution primitive for 3D deep learning
  
  • It takes advantage the **sparse representation** of point cloud to reduce the **memory consumption**.

  • It makes use of the **regularity** of voxel convolution to avoid the **random memory access**.

• Efficient 3D deep learning enables the deployment of 3D models on the edge devices (e.g., AR/VR headsets, mobile phones, self-driving cars).
Thank You!