PointNet:
Deep Learning on Point Sets for 3D Classification and Segmentation

CVPR 2017, Stanford

http://stanford.edu/~rqi/pointnet/

https://github.com/fxia22/pointnet.pytorch (not official)

Point cloud


Selection Box Editing Mode
Max coordinates: (0.47053, 0.148309, 0.366711)
Min coordinates: (-0.250932, -0.810572, 0.138992)

Point cloud

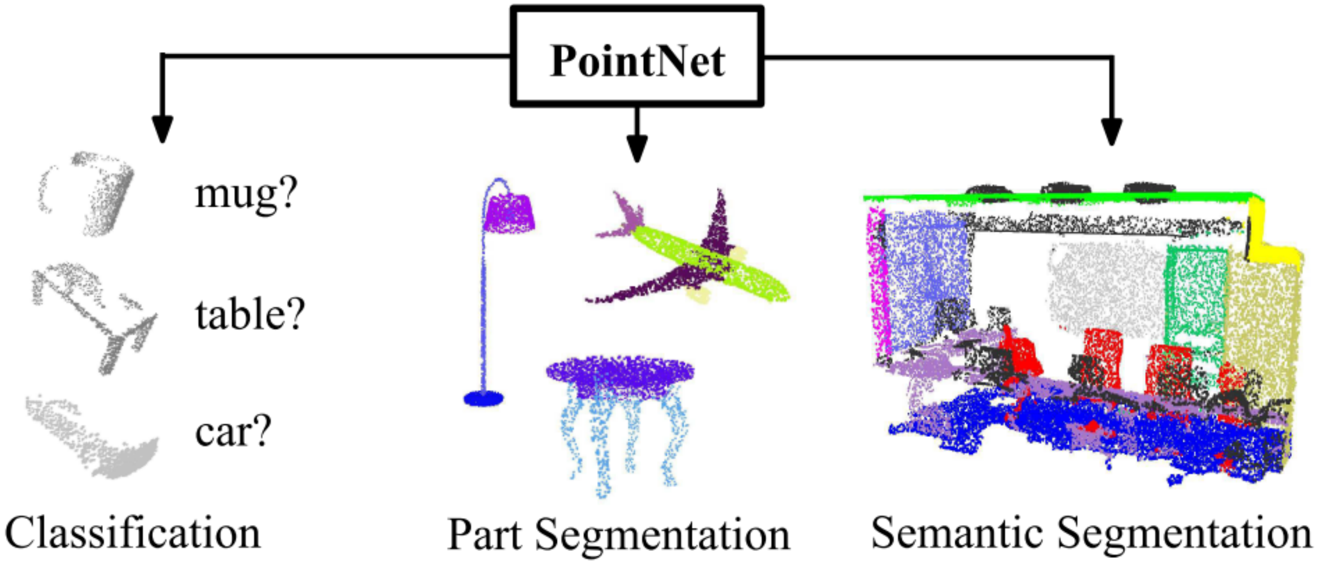


Figure 1. **Applications of PointNet.** We propose a novel deep net architecture that consumes raw point cloud (set of points) without voxelization or rendering. It is a unified architecture that learns both global and local point features, providing a simple, efficient and effective approach for a number of 3D recognition tasks.

3D featurization

- Voxelization
  - Pixel → Voxel (마인크래프트 생각하면 됨)
  - 3D convolution: computation cost, sparsity, symmetry, …
  - Voxelization 자체도 쉽지가 않다 (bleeding, …)
- Rendering
  - Differentiable renderer / ray tracer
  - 3D object as multi-view 2D images
  - 얘기하기 시작하면 책 몇 권이 뚝딱 나온다
- Point cloud
  - Canonical form

Dataset: ShapeNet

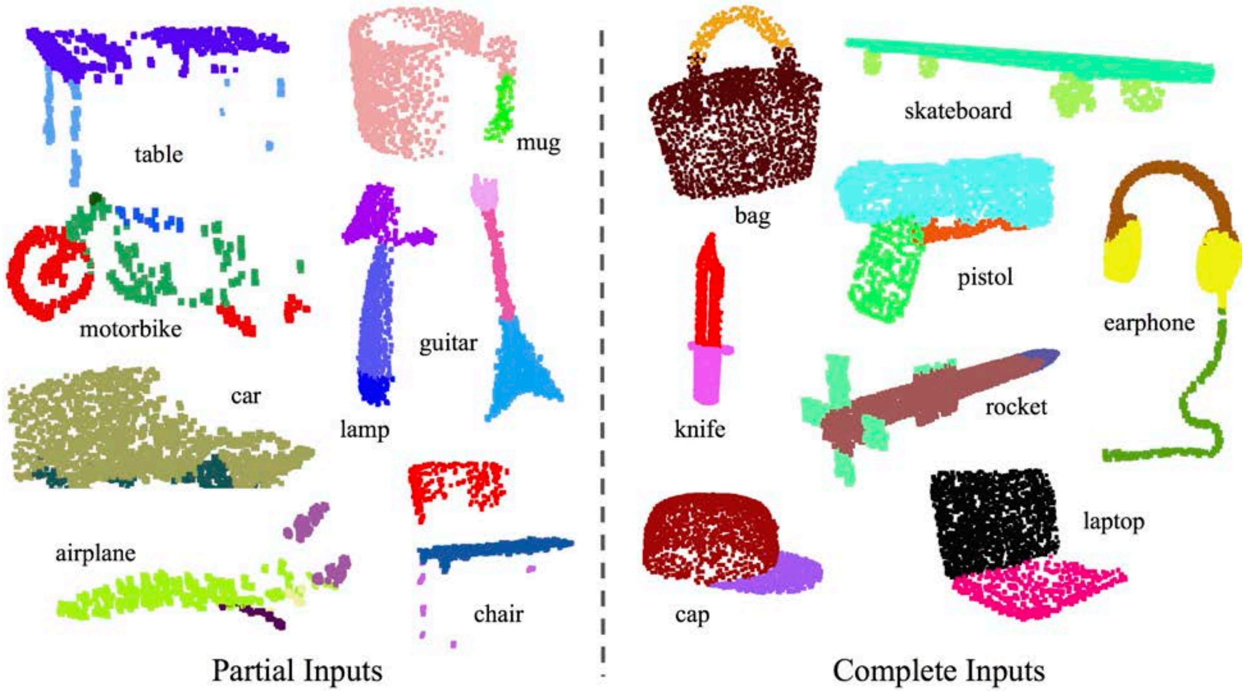
- ShapeNet
  - https://www.shapenet.org

- PointNet provides preprocessed ShapeNet data
  - ShapeNetPart dataset
  - https://github.com/charlesq34/pointnet
  - http://web.stanford.edu/~ericyi/project_page/part_annotation/index.html

Dataset: ShapeNet (part)

- ShapeNetPart
  - Airplane, Bag, Cap, …
  - 16 types of objects

- Each object has
  - Coordinates
  - Classes

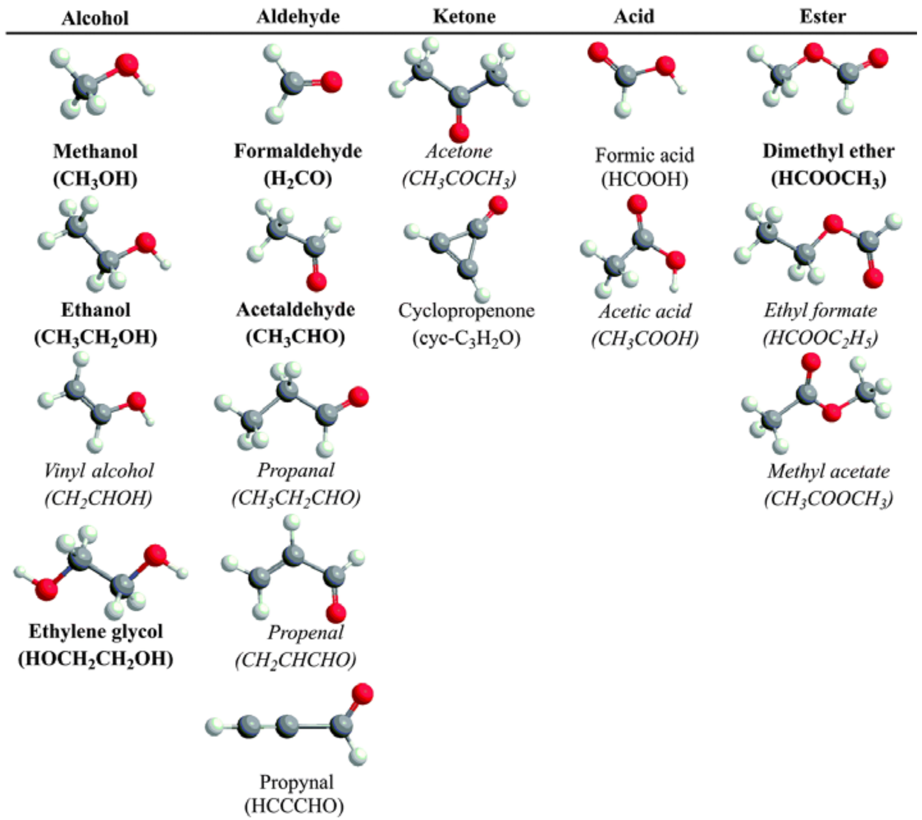


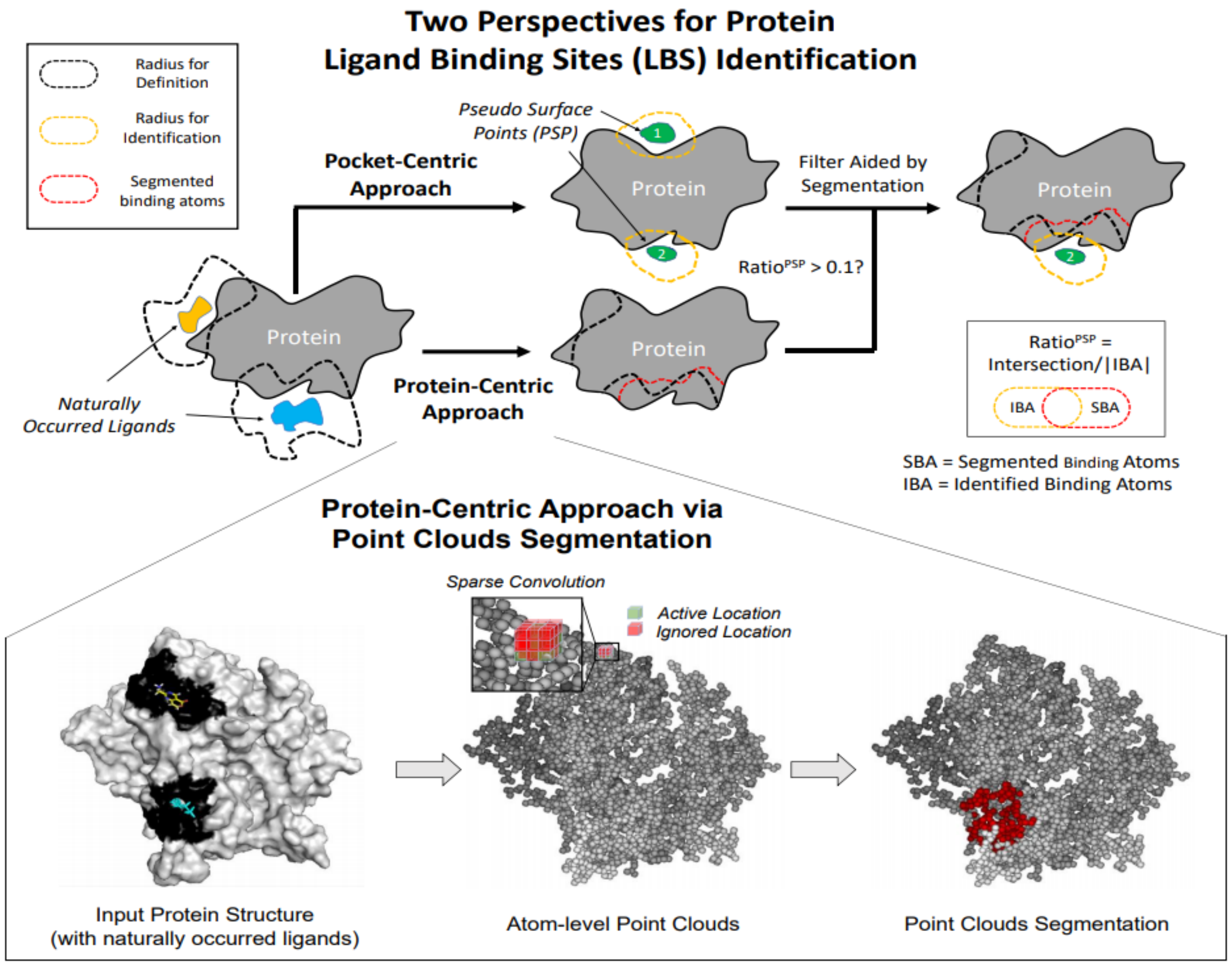
Challenges

- Point cloud features
  - Rotation invariant
  - Translation invariant
  - Order invariant (unordered set)
- Further reading
  - Set transformer (ICML 2019)
  - 3D steerable CNNs (NIPS 2018)
  - Generalizing Convolutional Neural Networks for Equivariance to Lie Groups on Arbitrary Continuous Data (Arxiv)
  - Deep parametric continuous convolutional neural networks (CVPR 2018)
  - 아무튼 엄청 많음

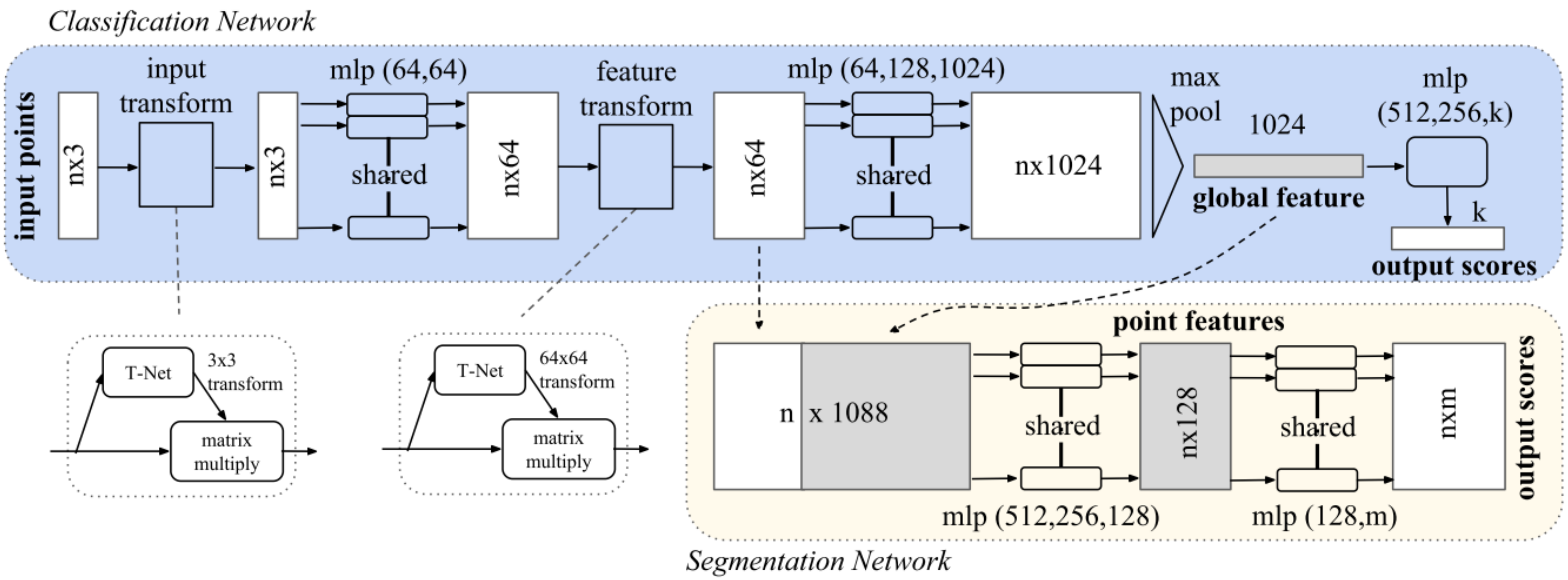
Motivation

- My motivation





Method



Classification Network
input points
nx3
input transform
nx3
mlp (64,64)
shared
nx64
feature transform
nx64
mlp (64,128,1024)
shared
nx1024
max pool
1024
global feature
mlp (512,256,k)
k
output scores
T-Net
3x3 transform
matrix multiply
T-Net
64x64 transform
matrix multiply
point features
n x 1088
shared
nx128
shared
nxm
output scores
mlp (512,256,128)
mlp (128,m)
Segmentation Network

Method: Implementation

- Main
  - x: (b, 3, n)
  - t1 = Tnet3d(x): (3, 3)
  - x = bmm(x, t1): (b, 3, n) → transpose 생략
  - x = ReLU(BatchNorm(Conv1d(x))): (b, 64, n)
  - …

Method: Implementation

- Tnet3d
  - x: (b, 3, n)
  - x = ReLU(BatchNorm(Conv1d(x))): (b, 64, n)
  - x = ReLU(BatchNorm(Conv1d(x))): (b, 128, n)
  - x = ReLU(BatchNorm(Conv1d(x))): (b, 1024, n)
  - x = max(x, dim = 2): (b, 1024)
  - x = MLP(x): (b, 9) → 3 layers
  - x = x.view(b, 3, 3) + Identity: (b, 3, 3)

  - return x

Method: Preprocessing

- Batch processing
  - Randomly select N points
- Normalization
  - Centering
  - Scaling (by dividing max distance)
- Data augmentation (optional)
  - Rotation (with fixed y-axis)
  - Jittering $\sim N(0, 0.02)$

Method: T-net

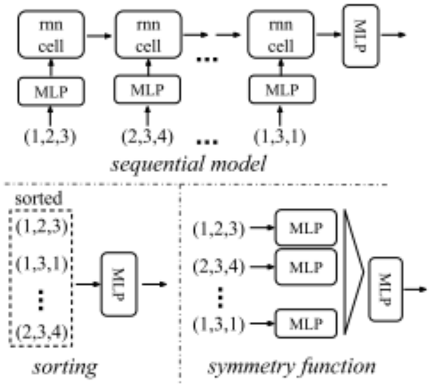
- T-net predict transformation matrix A
  - Output: 3 by 3 (input), 64 by 64 (feature) matrix
- Orthogonal regularization
  - @ feature transformation matrix
  - $L_{reg} = \|I - AA^T\|_F^2$

Transform	accuracy
none	87.1
input (3x3)	87.9
feature (64x64)	86.9
feature (64x64) + reg.	87.4
both	**89.2**

Table 5. **Effects of input feature transforms.** Metric is overall classification accuracy on ModelNet40 test set.

Result

Figure 4. **Qualitative results for semantic segmentation.** Top row is input point cloud with color. Bottom row is output semantic segmentation result (on points) displayed in the same camera viewpoint as input.



	accuracy
MLP (unsorted input)	24.2
MLP (sorted input)	45.0
LSTM	78.5
Attention sum	83.0
Average pooling	83.8
Max pooling	**87.1**

Figure 5. **Three approaches to achieve order invariance.** Multi-layer perceptron (MLP) applied on points consists of 5 hidden layers with neuron sizes 64,64,64,128,1024, all points share a single copy of MLP. The MLP close to the output consists of two layers with sizes 512,256.

Result

- Baseline: Yi et al. (SIGGRAPH Asia 2016)
  - No deep learning (CRF based)

	mean	aero	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
Wu [24]	-	63.2	-	-	-	73.5	-	-	-	74.4	-	-	-	-	-	-	74.8
Yi [26]	81.4	81.0	78.4	77.7	**75.7**	87.6	61.9	**92.0**	85.4	**82.5**	**95.7**	**70.6**	91.9	**85.9**	53.1	69.8	75.3
3DCNN	79.4	75.1	72.8	73.3	70.0	87.2	63.5	88.4	79.6	74.4	93.9	58.7	91.8	76.4	51.2	65.3	77.1
Ours	**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**

Table 2. **Segmentation results on ShapeNet part dataset.** Metric is mIoU(%) on points. We compare with two traditional methods [24] and [26] and a 3D fully convolutional network baseline proposed by us. Our PointNet method achieved the state-of-the-art in mIoU.

Result

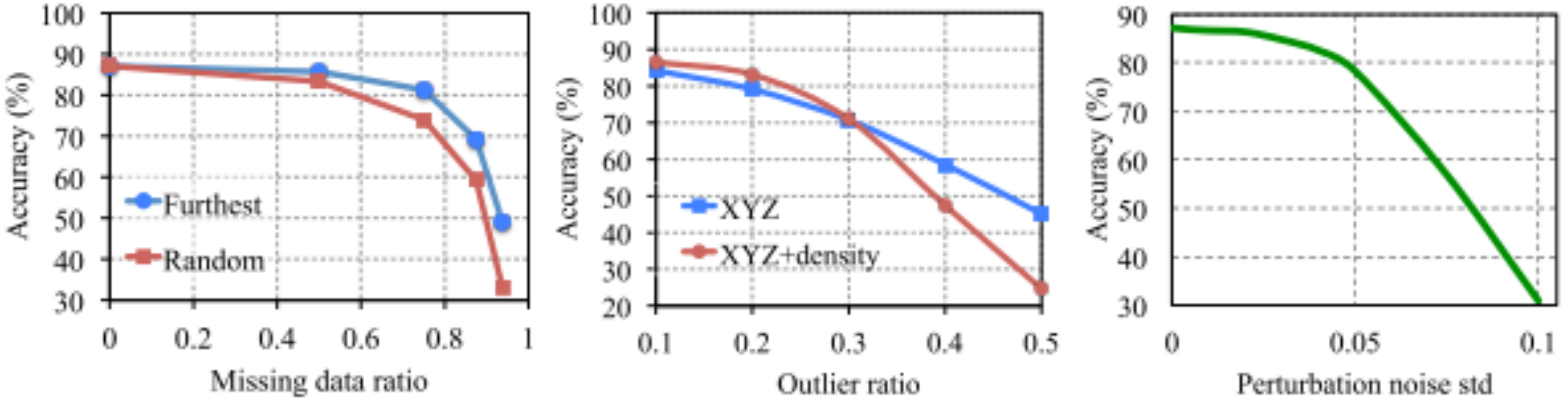


Figure 6. **PointNet robustness test.** The metric is overall classification accuracy on ModelNet40 test set. Left: Delete points. Furthest means the original 1024 points are sampled with furthest sampling. Middle: Insertion. Outliers uniformly scattered in the unit sphere. Right: Perturbation. Add Gaussian noise to each point independently.

Result

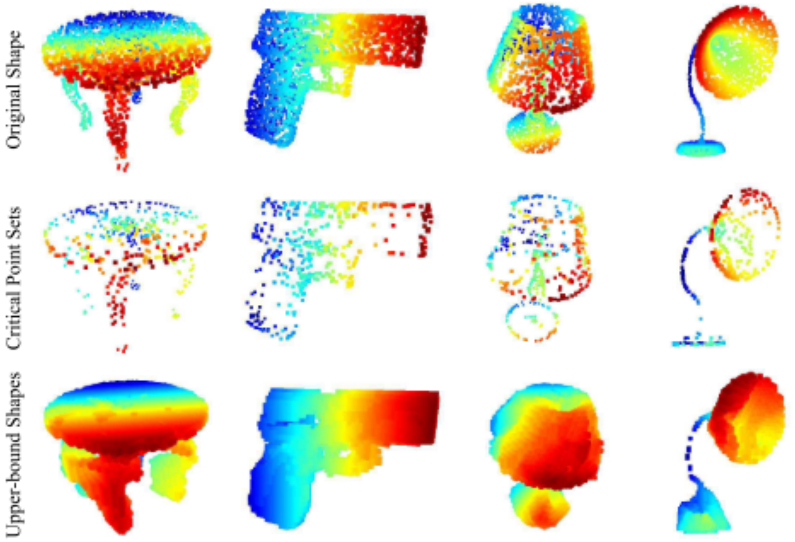


Figure 7. **Critical points and upper bound shape.** While critical points jointly determine the global shape feature for a given shape, any point cloud that falls between the critical points set and the upper bound shape gives exactly the same feature. We color-code all figures to show the depth information.

	#params	FLOPs/sample
PointNet (vanilla)	0.8M	148M
PointNet	3.5M	440M
Subvolume [16]	16.6M	3633M
MVCNN [20]	60.0M	62057M

Table 6. Time and space complexity of deep architectures for 3D data classification. PointNet (vanilla) is the classification PointNet without input and feature transformations. FLOP stands for floating-point operation. The "M" stands for million. Subvolume and MVCNN used pooling on input data from multiple rotations or views, without which they have much inferior performance.