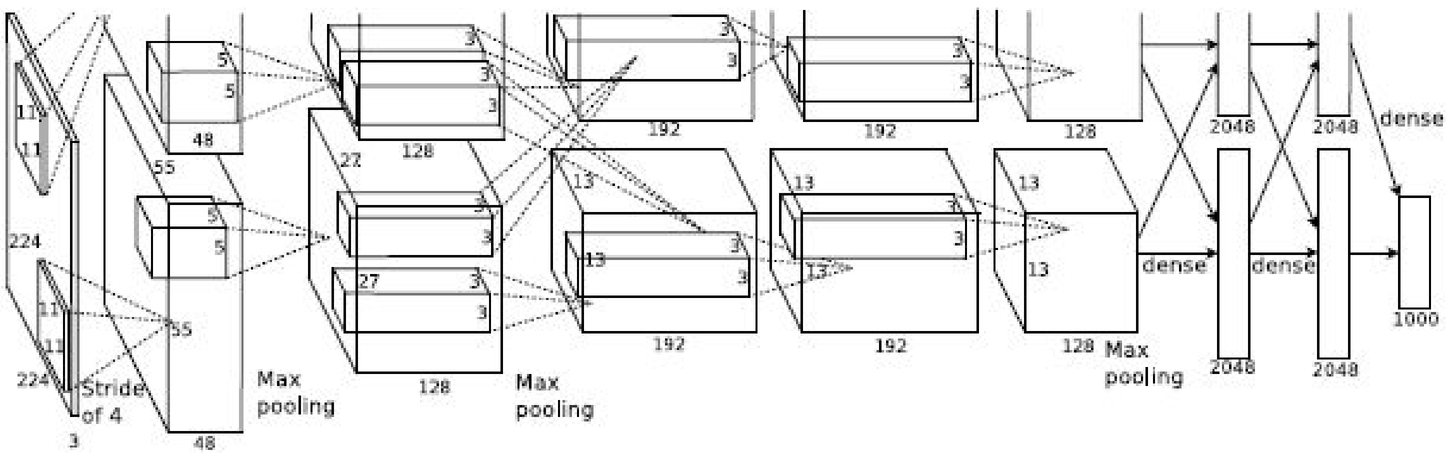
Numerical Optimization for Graphics/AI
3D Deep Learning I

Qixing Huang
Nov. 19th 2018

AlexNet


224
224
11
11
3
Stride of 4
55
55
5
5
48
48
Max pooling
27
128
128
3
3
Max pooling
13
192
192
192
192
128
128
Max pooling
dense
dense
dense
2048
2048
2048
2048
1000

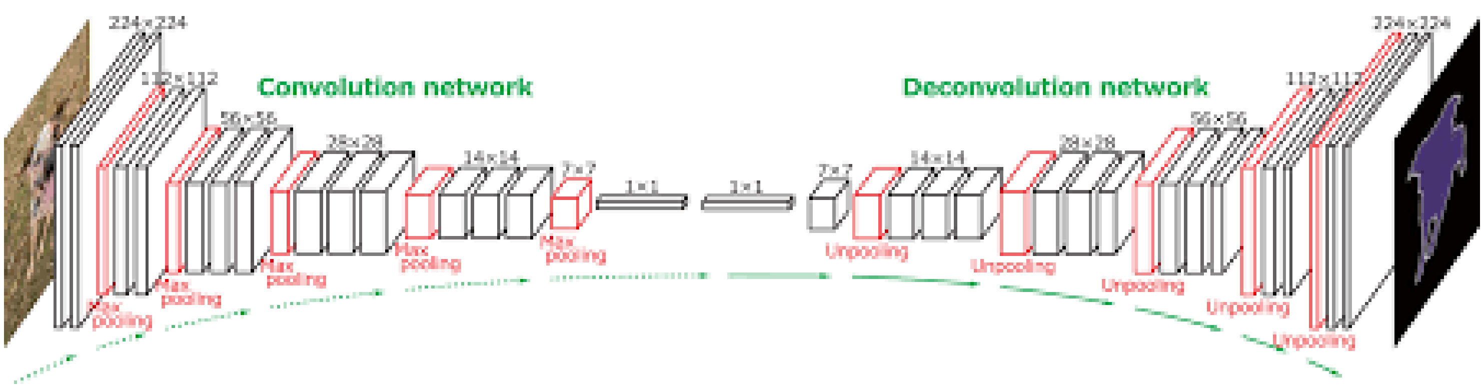
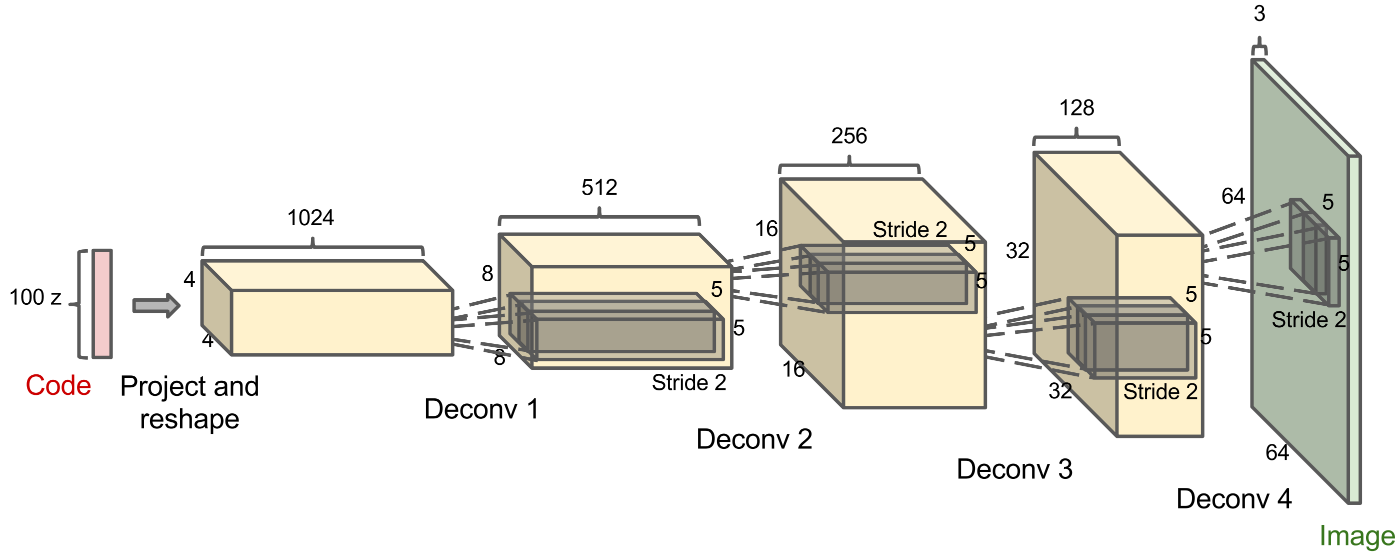
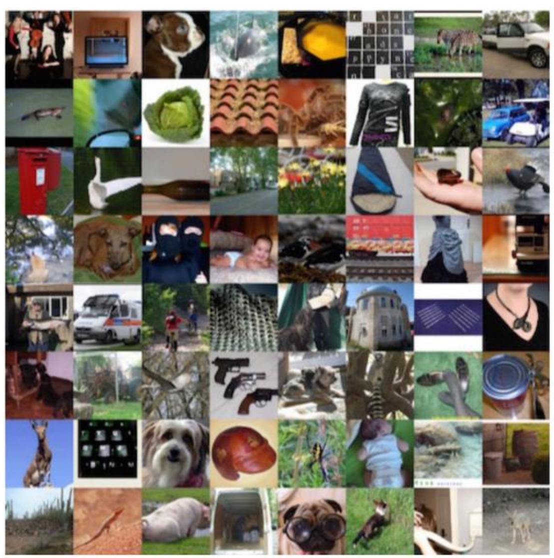

Convolution network
Deconvolution network
224×224
112×112
56×56
28×28
14×14
7×7
1×1
1×1
7×7
14×14
28×28
56×56
112×112
224×224
Max pooling
Unpooling

Image Generation

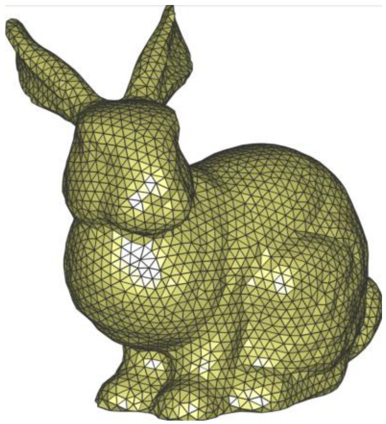




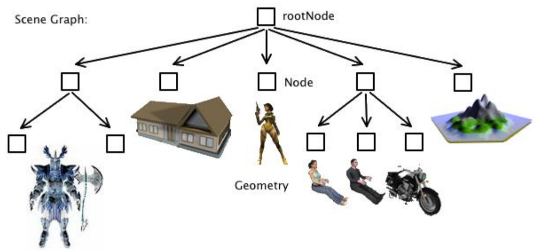
Real images (ImageNet)

Generated images

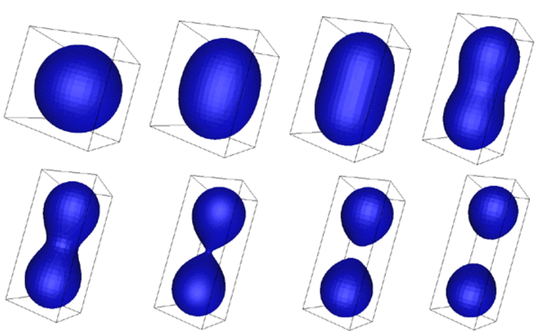
3D Surface Representations



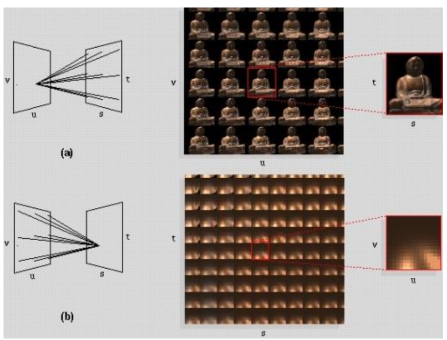
Triangular mesh



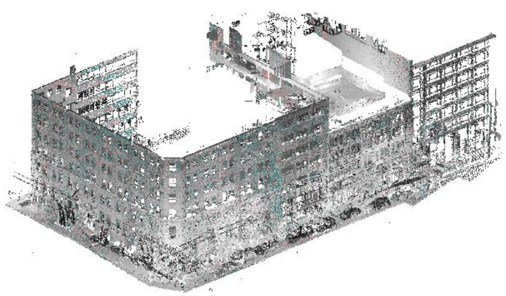
Part-based models



Implicit surface



Light Field Representation



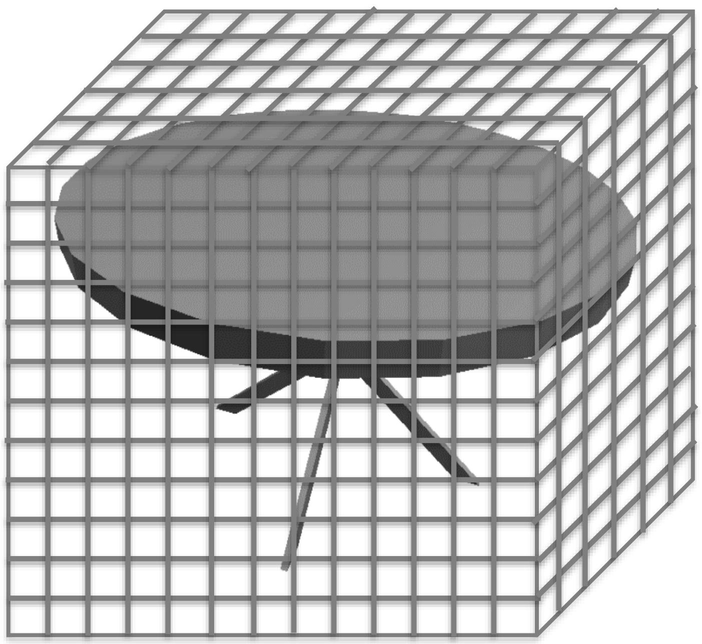
Point cloud

3D Voxel Grids

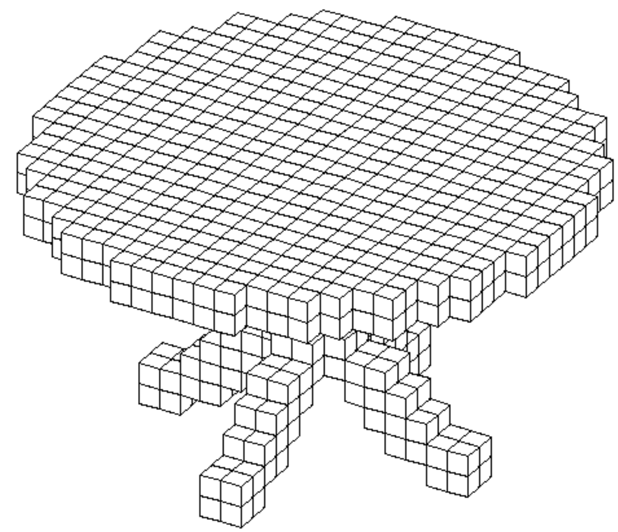
[Wu et al. 15]

3D Deep Learning

3D Shape as Volumetric Representation



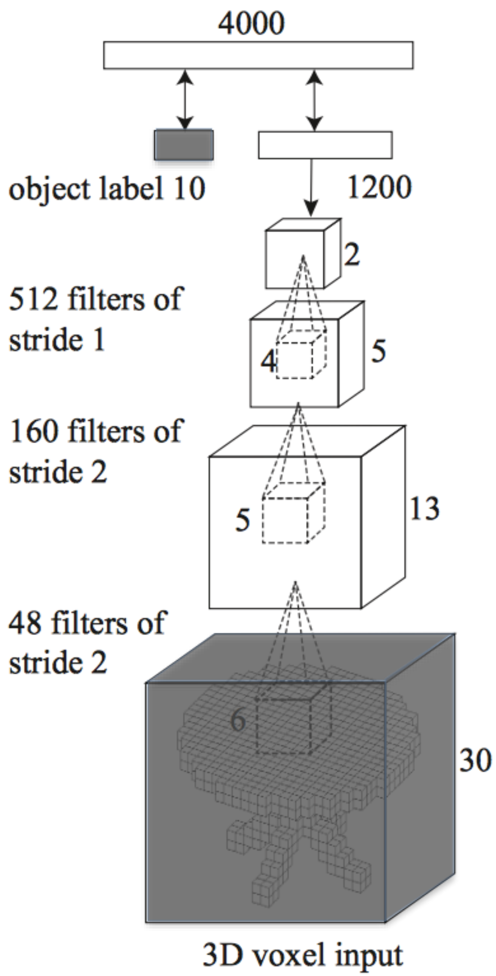
mesh



binary voxel

[Wu et al. 15]

3D ShapeNets



Convolutional Deep Belief Network $p(\mathbf{x}, y)$

A **Deep Belief Network** is a generative graphical model that describes the distribution of input x over class y.

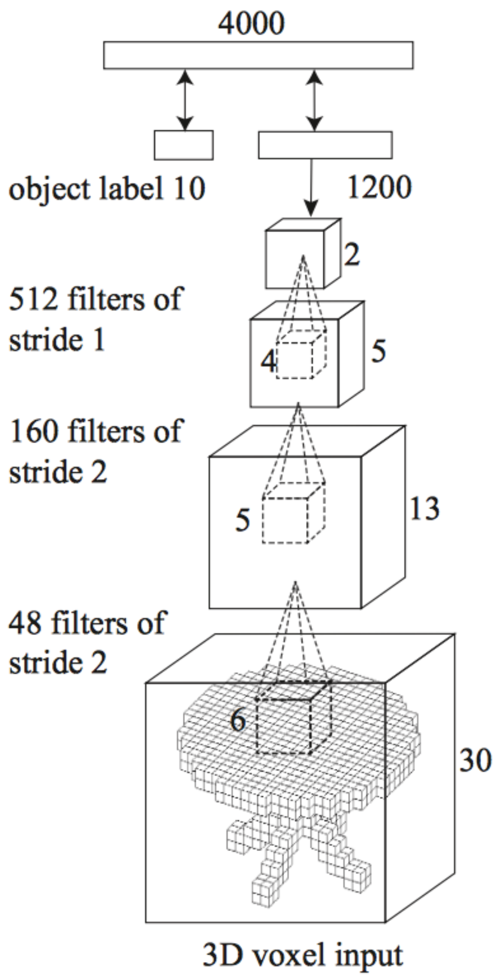
- Convolution to enable compositionality
- No pooling to reduce reconstruction error

configurations

Layer 1-3	convolutional RBM
Layer 4	fully connected RBM
Layer 5	multinomial label + Bernoulli feature form an associate memory

[Wu et al. 15]

3D ShapeNets



Convolutional Deep Belief Network $p(\mathbf{x}, y)$

3D ShapeNets ≠ CNNs

$p(x, y)$ $\quad$ $p(y|x)$

$$\downarrow$$

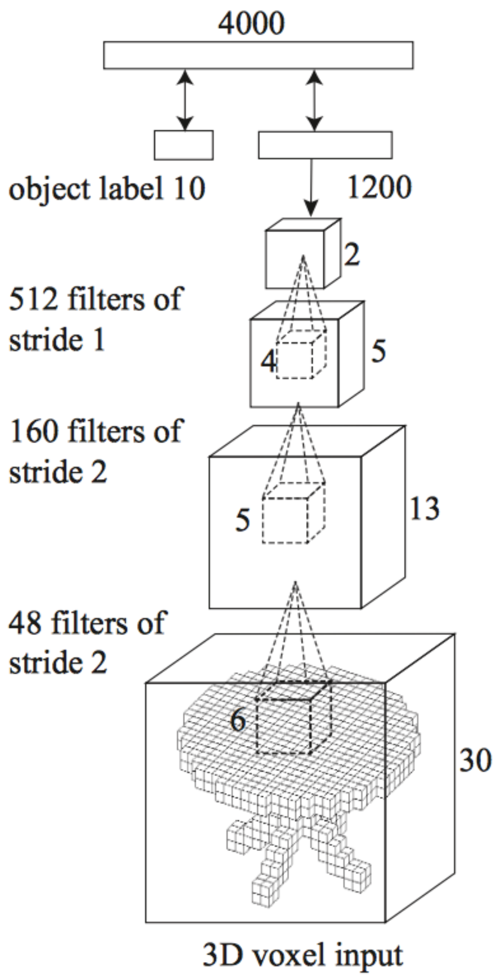
$p(y|x)$ discriminative process

$p(x|y)$ generative process

* 3D ShapeNets can be converted into a CNN,
and discriminatively trained with back-propagation.

[Wu et al. 15]

Training



Convolutional Deep Belief Network $p(\mathbf{x}, y)$

Maximum Likelihood Learning

Layer-wise pre-training:

Lower four layers are trained by CD

Last layer is trained by FPCD[1]

Fine-tuning:

Wake sleep but keep weights tied

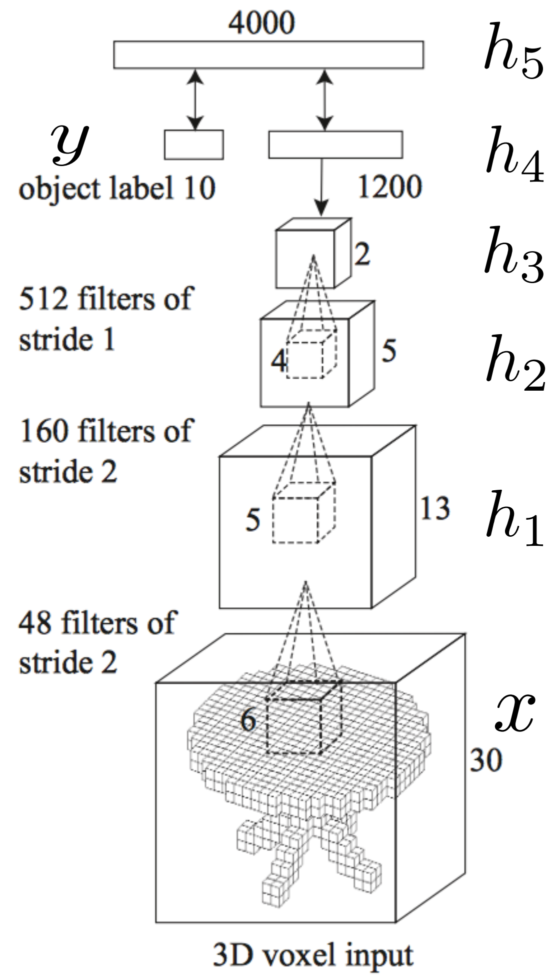
[Wu et al. 15]

Sampling

generation process:

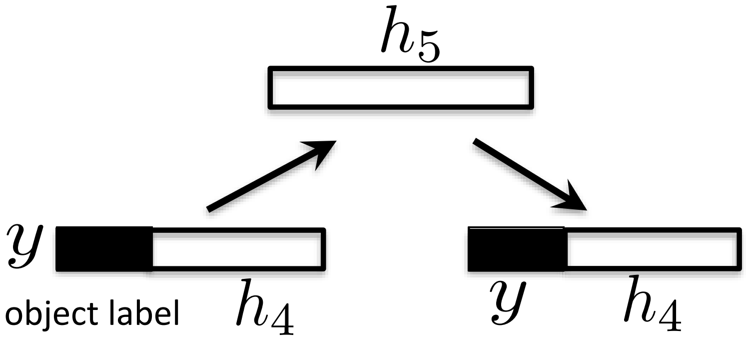
Gibbs Sampling

$$p(x, h_1 ... h_5 | y) = p(h_4, h_5 | y) \cdot \prod_{i=1}^{4} p(h_{i-1} | h_i)$$

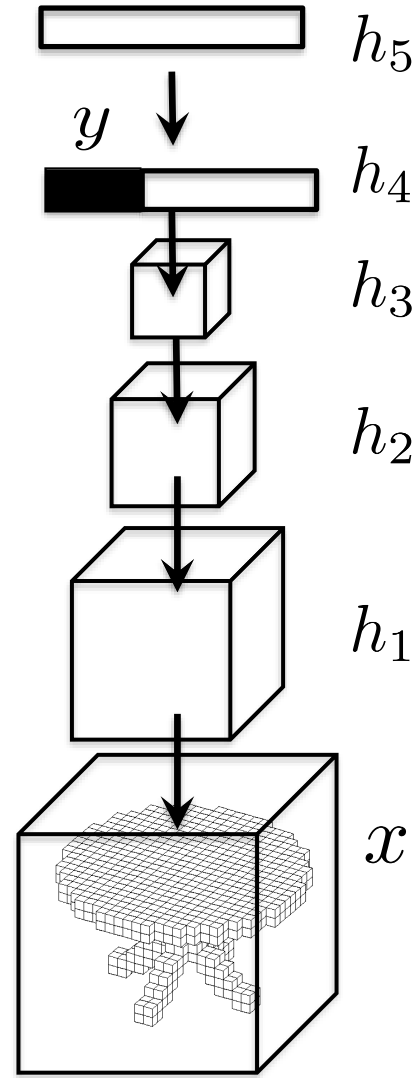


Convolutional Deep Belief Network $p(\mathbf{x}, y)$

Sampling



....



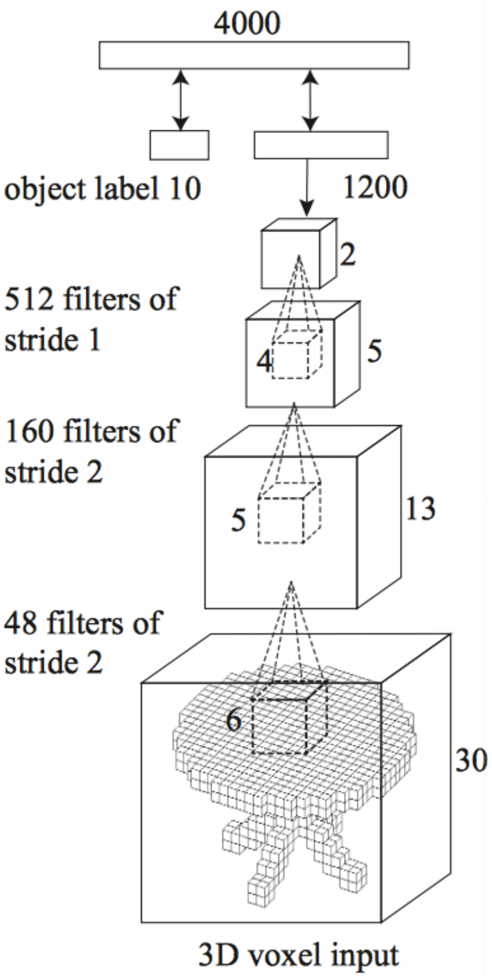
generation process:

Gibbs Sampling

$$p(x, h_1 ... h_5 | y) = p(h_4, h_5 | y) \cdot \prod_{i=1}^{4} p(h_{i-1} | h_i)$$

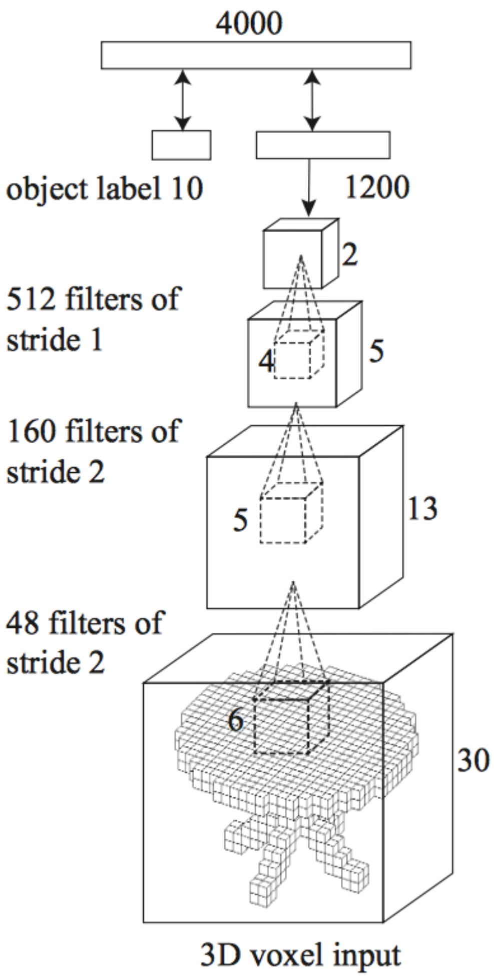
[Wu et al. 15]

3D ShapeNets

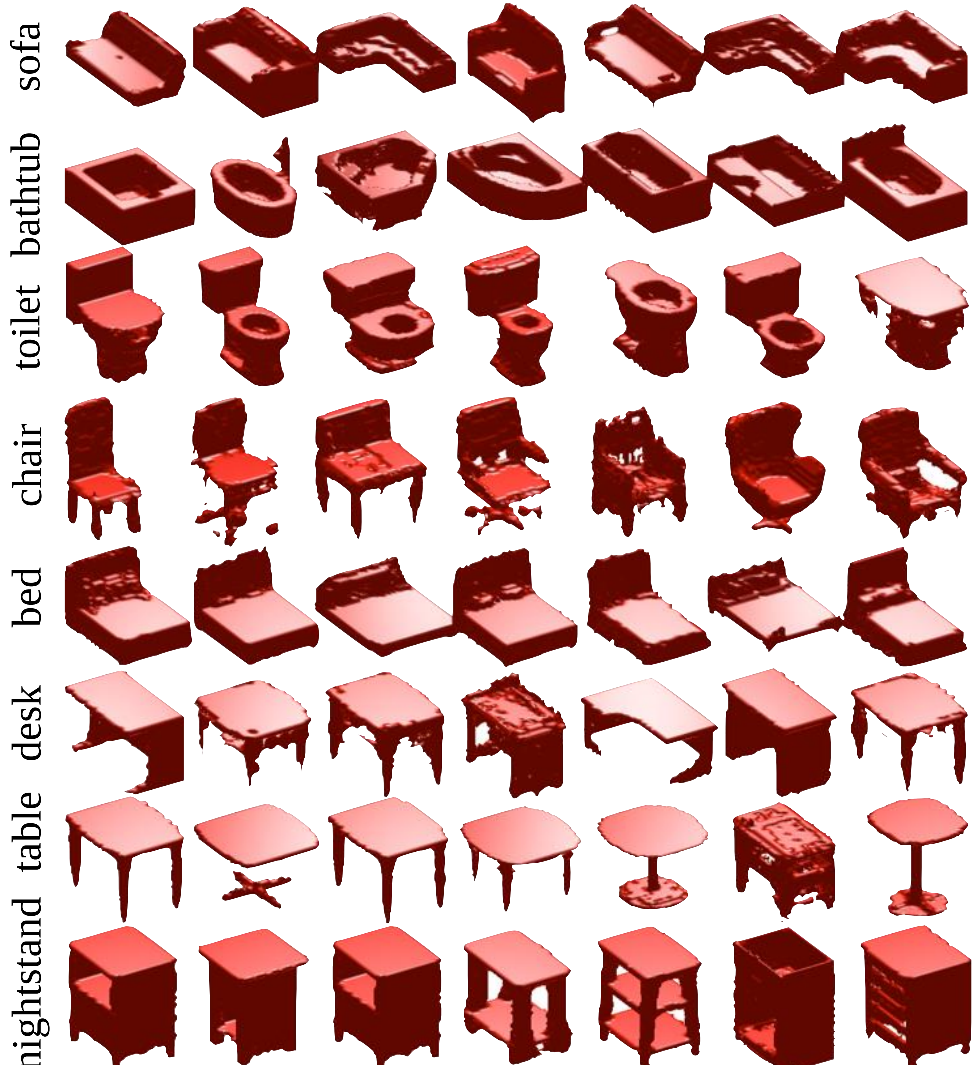


Convolutional Deep Belief Network $p(\mathbf{x}, y)$

As a 3D Shape Prior

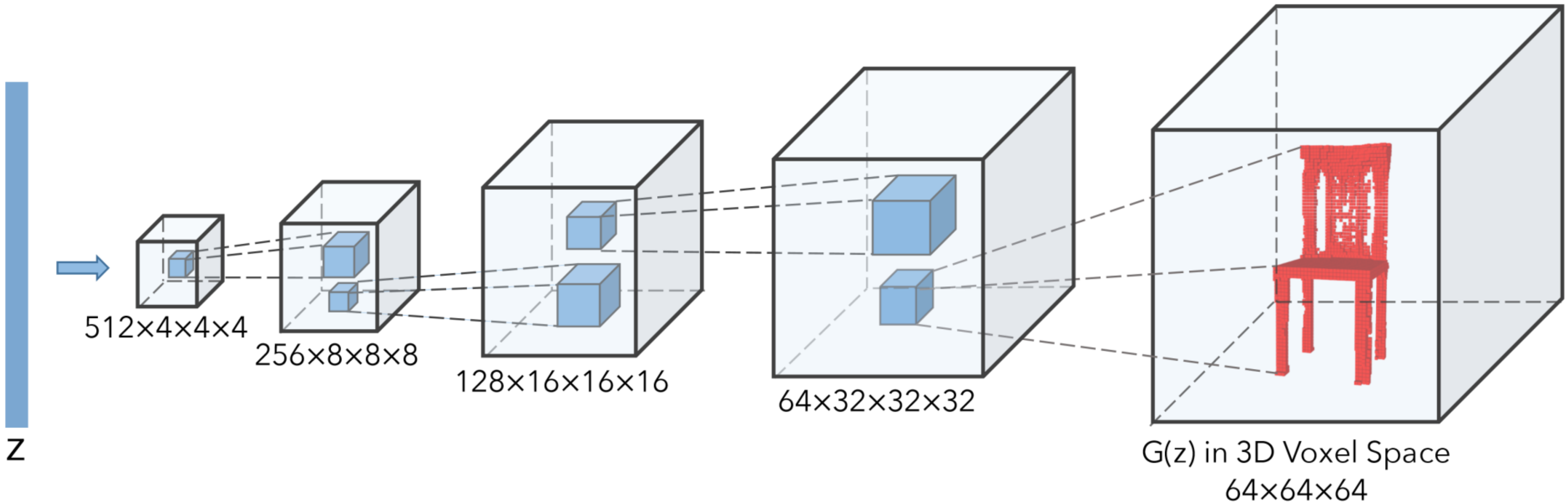


Convolutional Deep Belief Network $p(\mathbf{x}, y)$



Sampled Models

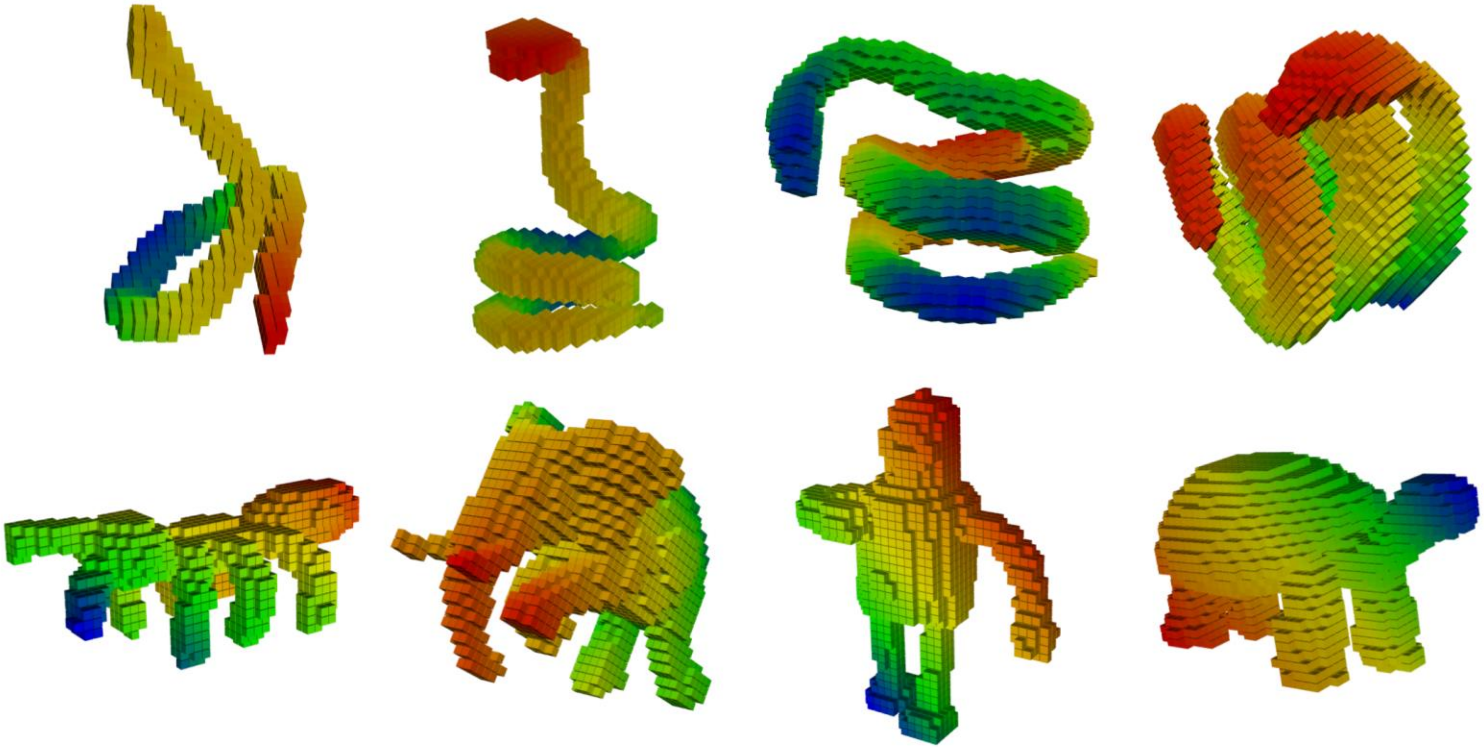
3D Generative Adversarial Network [Wu et al. 16]



Objects generated by Wu et al. [2015] (30 × 30 × 30)

Objects generated by a volumetric autoencoder (64 × 64 × 64)

Sparse 3D Convolutional Networks
[Ben Graham 2016]



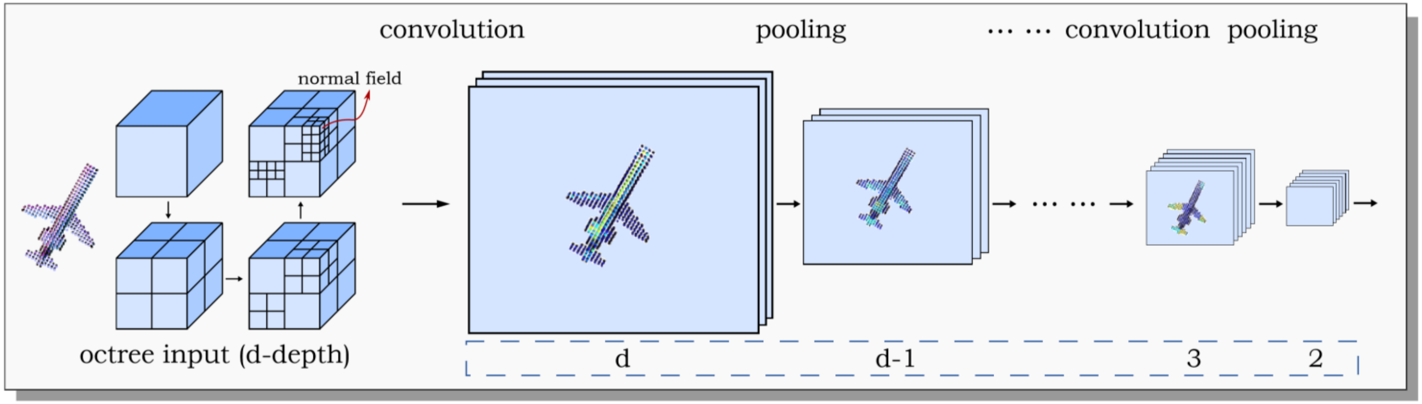
40x40x40 Grid

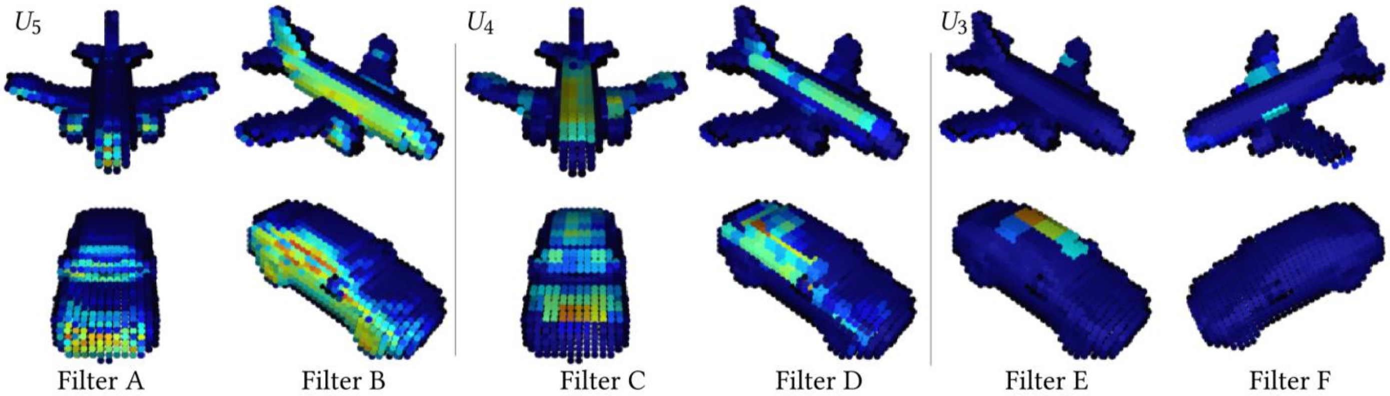
Sparsity for lower layers

Low resolution for upper layers

Octree classification networks

[Wang et al. 18]





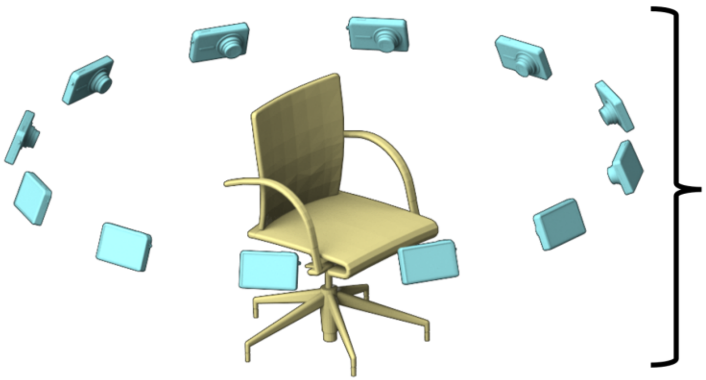
The responses of some convolutional filters at different levels on two models are rendered. Red represents a large response and blue a low response.

Discussion

+ Easy to implement
+ Hardware friendly

- Low resolution
- No structural information
- Cannot utilize 2D training data

Light Field Representation



3D shape model
rendered with
different virtual cameras

Method	**Training Config.**			**Test Config.**	**Classification (Accuracy)**	**Retrieval (mAP)**
	Pre-train	**Fine-tune**	**#Views**	**#Views**		
(1) SPH [16]	-	-	-	-	68.2%	33.3%
(2) LFD [5]	-	-	-	-	75.5%	40.9%
(3) 3D ShapeNets [37]	ModelNet40	ModelNet40	-	-	77.3%	49.2%
(4) FV	-	ModelNet40	12	1	78.8%	37.5%
(5) FV, 12×	-	ModelNet40	12	12	84.8%	43.9%
(6) CNN	ImageNet1K	-	-	1	83.0%	44.1%
(7) CNN, f.t.	ImageNet1K	ModelNet40	12	1	85.1%	61.7%
(8) CNN, 12×	ImageNet1K	-	-	12	87.5%	49.6%
(9) CNN, f.t.,12×	ImageNet1K	ModelNet40	12	12	88.6%	62.8%
(10) MVCNN, 12×	ImageNet1K	-	-	12	88.1%	49.4%
(11) MVCNN, f.t., 12×	ImageNet1K	ModelNet40	12	12	89.9%	70.1%
(12) MVCNN, f.t.+metric, 12×	ImageNet1K	ModelNet40	12	12	89.5%	**80.2**%
(13) MVCNN, 80×	ImageNet1K	-	80	80	84.3%	36.8%
(14) MVCNN, f.t., 80×	ImageNet1K	ModelNet40	80	80	**90.1**%	70.4%
(15) MVCNN, f.t.+metric, 80×	ImageNet1K	ModelNet40	80	80	**90.1**%	79.5%

* f.t.=fine-tuning, metric=low-rank Mahalanobis metric learning

Discussion

+ Can utilize 2D training data
+ Efficient since using 2D convolutions
+ Top-performing algorithms

-- Redundancy
-- Loss of information per view
-- How to pick views?

? Convolutions on Spheres

Point cloud Representation

[Su et al. 17a, Su et al. 17b]

Object Classification on Partial Scans

Input:
Partial scan
(XYZ)





Output:
Category
classification

Car

Car

Airplan
e

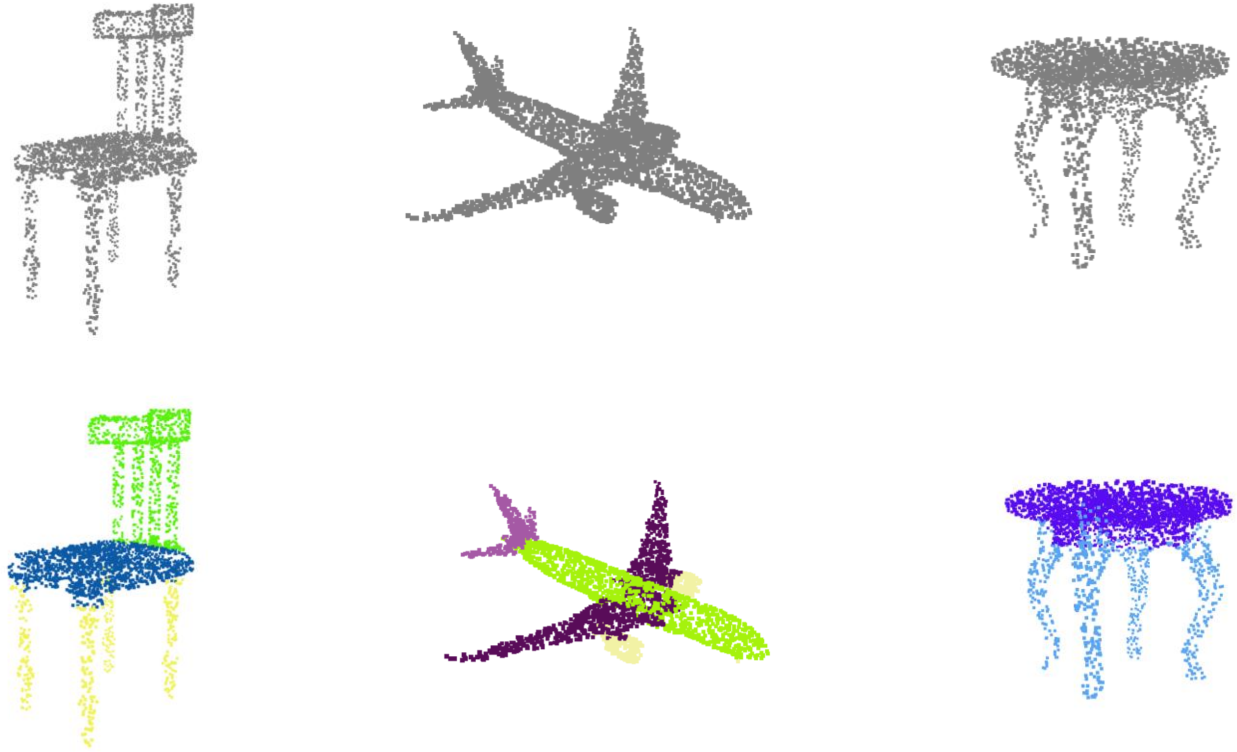
Tabl
e

Mu
g

Object Part Segmentation

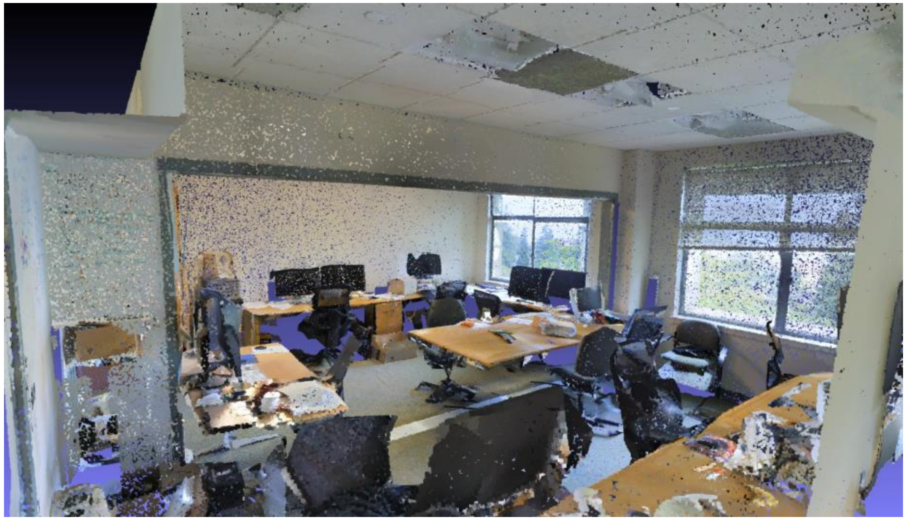
Input:
Point cloud (XYZ)

Output:
Per point label

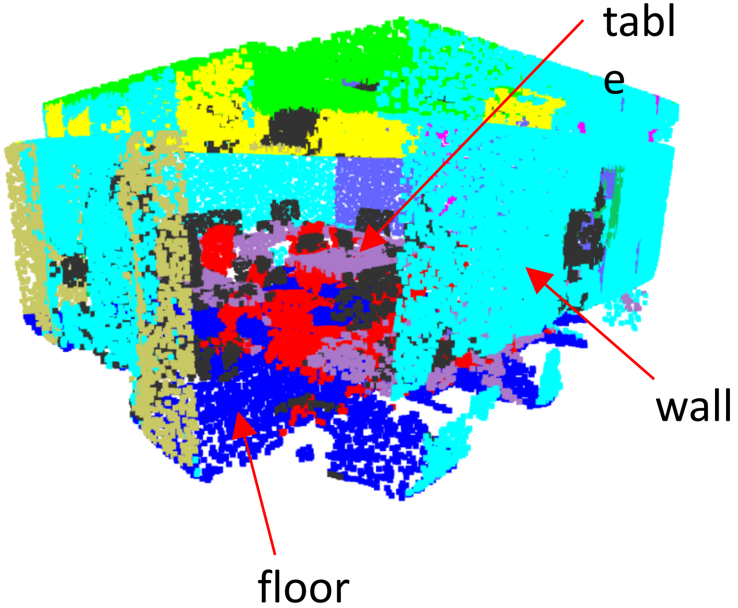


Semantic Segmentation for Indoor Scenes

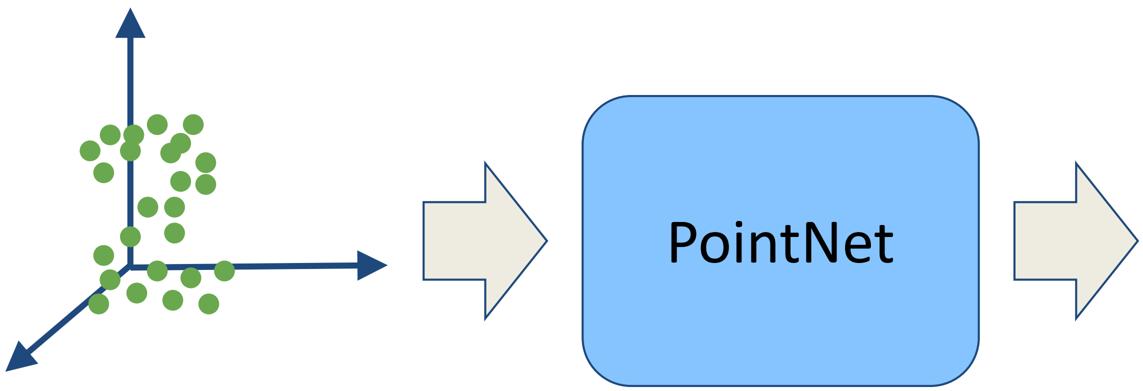
Input:
Point cloud (XYZRGB) of a room



Output (*current performance*):
Semantic segmentation of the room



Uniform Framework: PointNet



Point Set

Object Classification

Part Segmentation

Semantic Segmentation in scenes

Point Feature Learning

...

Theorem 1. *Suppose* $f : \mathcal{X} \rightarrow \mathbb{R}$ *is a continuous set function w.r.t Hausdorff distance* $d_H(\cdot, \cdot)$. $\forall \epsilon > 0$, $\exists$ *a continuous function* h *and a symmetric function* $g(x_1, \ldots, x_n) = \gamma \circ MAX$, *such that for any* $S \in \mathcal{X}$,

$$\left| f(S) - \gamma \left(\underset{x_i \in S}{MAX} \{h(x_i)\} \right) \right| < \epsilon$$

where $x_1, \ldots, x_n$ *is the full list of elements in* S *ordered arbitrarily,* γ *is a continuous function, and MAX is a vector max operator that takes* n *vectors as input and returns a new vector of the element-wise maximum.*

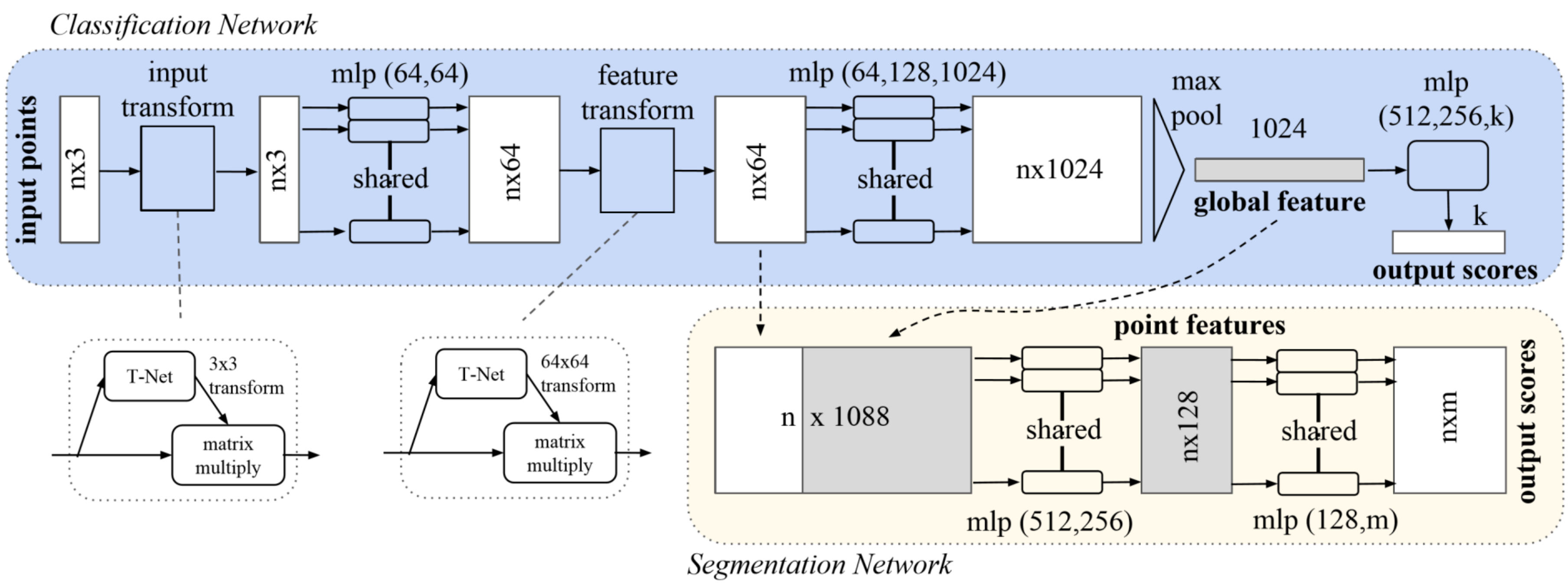


Figure 2. PointNet Architecture. The classification network takes n points as input, applies input and feature transformations, and then aggregates point features by max pooling. The output is classification score for k classes. The segmentation network is an extension to the classification net. It concatenates global and local features and outputs per point scores. "mlp" stands for multi-layer perceptron, the numbers in brackets are its layer sizes. Batchnorm is used for all layers with ReLU. Dropout layers are used for the last mlp in classification net.

ModelNet shape 40-class classification

Model	Accuracy
MLP	40%
LSTM	75%
Conv-Max-FC (1 max)	84%
Conv-Max-FC (2 max)	86%
Conv-Max-FC (2 max) + Input Transform	87.8%
Conv-Max-FC (2 max) + Feature Transform	86.8%
Conv-Max-FC (2 max) + Feature Transform + orthogonal regularization	87.4%
Conv-Max-FC (2 max) + Input Transform + Feature Transform + orthogonal regularization	**88.9%**

Best Volumetric CNN: 89.1%
However, PointNet is around 5x - 10x faster than Volumetric CNN