Deep learning for 3D data

Levente Tamas

With slides adapted from H. Su and IB Shabat

Content

Content

1. **Preliminary ideas on DL**
2. Why 3D data with deep learning?
3. Classification
4. Segmentation
5. Perspectives

1. Short introduction to DL

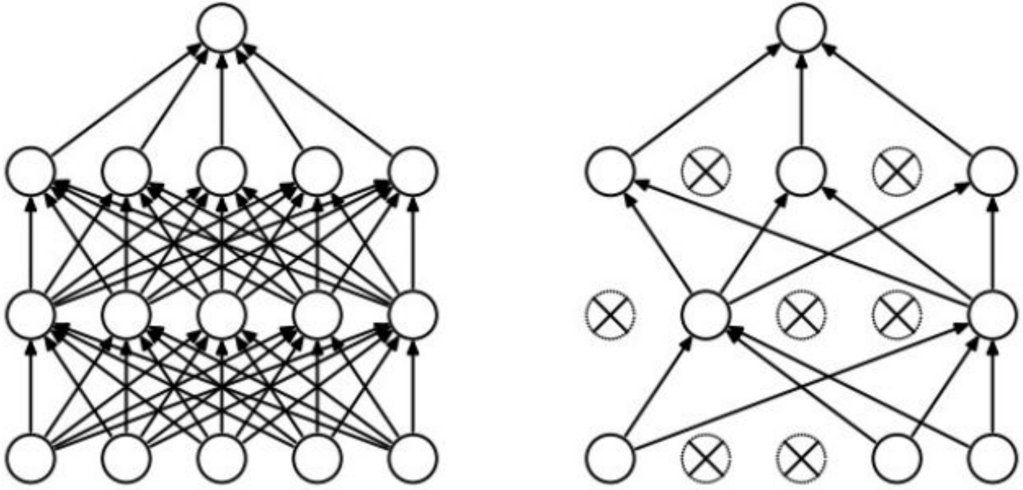
1.1. Motivation

1.2. Neuronal networks

1.3. Optimization details

1.4. Convolutional neural networks

1.5. Recurrent neural networks

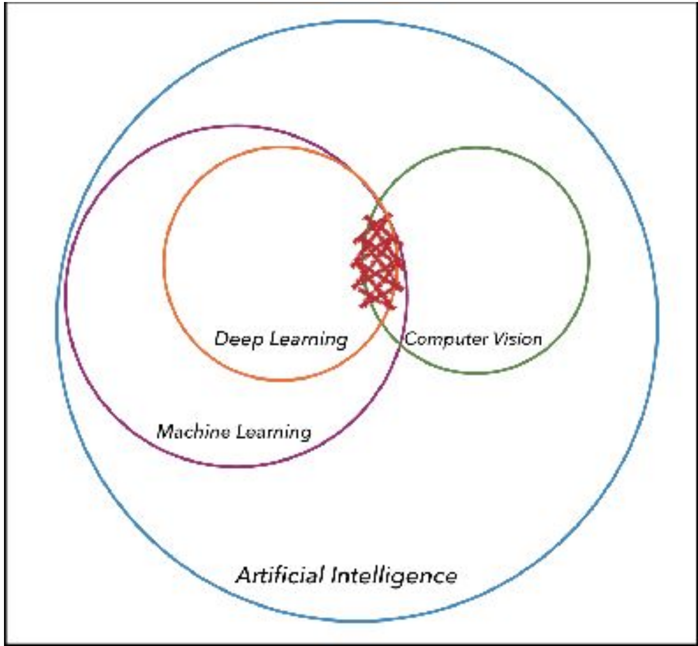


(a) Standard Neural Net (b) After applying dropout.

1.1 Motivation

- Subset of AI (see fig)
- Universal approx. of nonlinear functions (Hornik 1991)
- Significant results in
  - Image processing
  - Speech reco
  - NLP
- Different types:
  - Multilayer perceptrons
  - Convolutional Neural Networks (CNN)
  - Recurrent Neural Networks (RNN)
- Main reference book:

  http://www.deeplearningbook.org/



1.1 Motivation

- Significant results in
  - Image processing
  - Speech reco
  - NLP
- Different types:
  - Multilayer perceptrons
  - Convolutional Neural Networks (CNN)
  - Recurrent Neural Networks (RNN)
- Main reference book:

  http://www.deeplearningbook.org/

1.1 Motivation

Demystification:

What is not?

Black box	• Interpretable ML • Visualize gradients and activations
Needs too much data	• Transfer learning • Share pre-trained nets
Needs ML PhD	• No longer true • fastai & keras libs, MOOCs, etc
Only for vision	• No longer true • SoTA for speech, structured data, time series…
Needs lots of GPUs	• Was never true • …except for some research projects
“Not really AI”	• Who cares? • Do you really want to build a brain?

1.1 Motivation

Evolution of the state of the art performance in **2D classification**:

.	Shape	Appearance		Classification Accuracy (%)				
		layout type	using ground truth	family	breed (S. 4.2)		both (S. 4.3)	
				(S. 4.1)	cat	dog	hierarchical	flat
1	✓	–	–	94.21	NA	NA	NA	NA
2	–	Image	–	82.56	52.01	40.59	NA	39.64
3	–	Image+Head	–	85.06	60.37	52.10	NA	51.23
4	–	Image+Head+Body	–	87.78	64.27	54.31	NA	54.05
5	–	Image+Head+Body	✓	88.68	66.12	57.29	NA	56.60
6	✓	Image	–	94.88	50.27	42.94	42.29	43.30
7	✓	Image+Head	–	95.07	59.11	54.56	52.78	54.03
8	✓	Image+Head+Body	–	94.89	63.48	55.68	55.26	56.68
9	✓	Image+Head+Body	✓	95.37	66.07	59.18	57.77	59.21

1.2 Neuronal networks

Artificial **neuron** with params θ, input x: $y = f(x, \theta)$ → **nonlinear** optimization

More generally: $y_j = f_j(x) = \phi(w_j, x_i + b_j)$

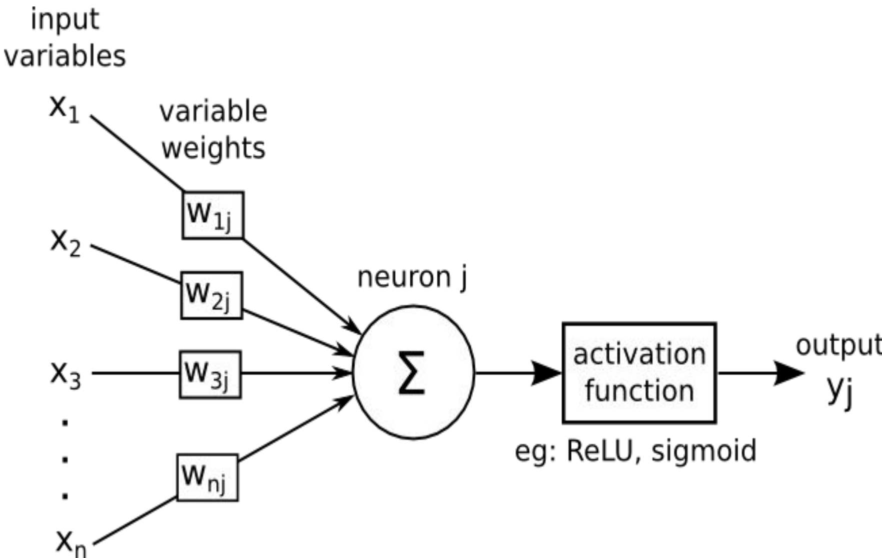
With ϕ being the **activation** function, e.g

- Identity
- Rectified Linear Unit (ReLU)
- Sigmoid
- Softmax

$$\phi(x) = x$$
$$\phi(x) = \max(0, x)$$
$$\phi(x) = \frac{1}{1 + exp(-x)}$$
$$softmax(z) = \frac{exp(z)}{\sum_j exp(x_j)}$$

Schematic representation of:

$$\sum = (w_j, x_i) + b_j$$



1.3 Multilayer perceptron (MLP)

Several interconnected **hidden** layers + I/O

No connection between neurons on the same layer

The activation Ψ at the O layer:

- **Regression**
- **Classification** (e. softmax)

Summary of L hidden layers, with

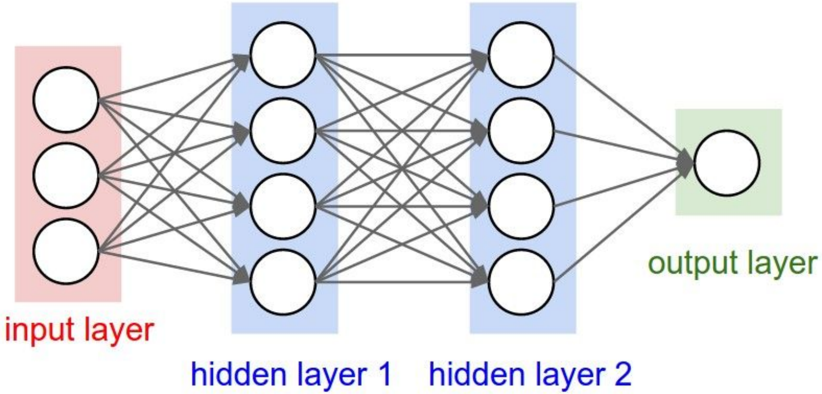
$$h^{(0)}(x) = x$$

For $k = 1, \ldots, L$ (hidden layers),

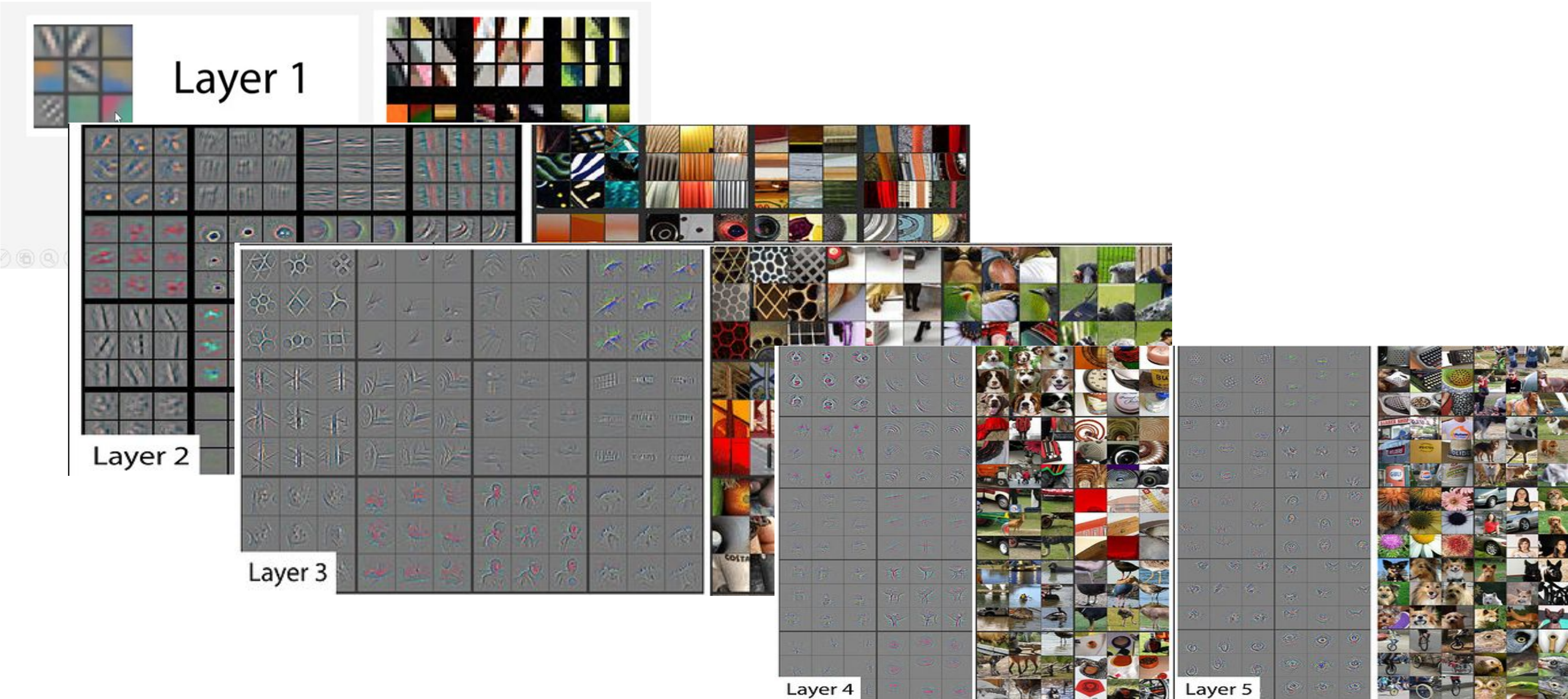
$$\begin{aligned} a^{(k)}(x) &= b^{(k)} + W^{(k)} h^{(k-1)}(x) \\ h^{(k)}(x) &= \phi(a^{(k)}(x)) \end{aligned}$$

For $k = L + 1$ (output layer),

$$\begin{aligned} a^{(L+1)}(x) &= b^{(L+1)} + W^{(L+1)} h^{(L)}(x) \\ h^{(L+1)}(x) &= \psi(a^{(L+1)}(x)) := f(x, \theta). \end{aligned}$$



1.3 Multilayer perceptron (MLP)



1.3 Optimization details

Loss function - how to minimize?

- Expected loss $$L(\theta) = -\mathbb{E}_{(X,Y)\sim P}(\log(p_\theta(Y/X)).$$
- For Gaussian models, $p_\theta(Y/X = x) \sim \mathcal{N}(f(x,\theta), I)$, $$L(\theta) = \mathbb{E}_{(X,Y)\sim P}(\|Y - f(X,\theta)\|^2).$$
- For binary cae $Y \in \{0,1\}$, $$L(\theta) = -\mathbb{E}_{(X,Y)\sim P}[Y \log(f(X,\theta)) + (1-Y)\log(1-f(X,\theta))].$$
- For K classes $$L(\theta) = -\mathbb{E}_{(X,Y)\sim P}[\sum_{j=1}^{k} \mathbf{1}_{Y=j} \log p_\theta(Y = j/X)].$$

1.3 Optimization details

Stochastic gradient descent

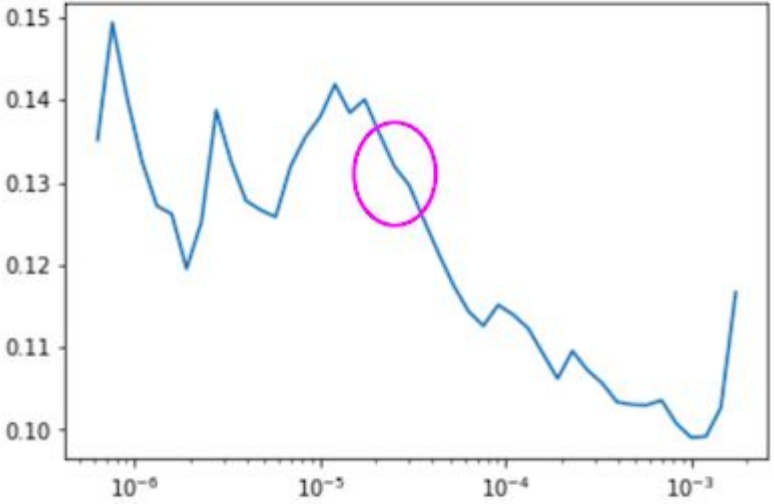
- Initialization of $\theta = (W^{(1)}, b^{(1)}, \dots, W^{(L+1)}, b^{(L+1)})$.
- For N iterations :
  - For each training data (X_i, Y_i),

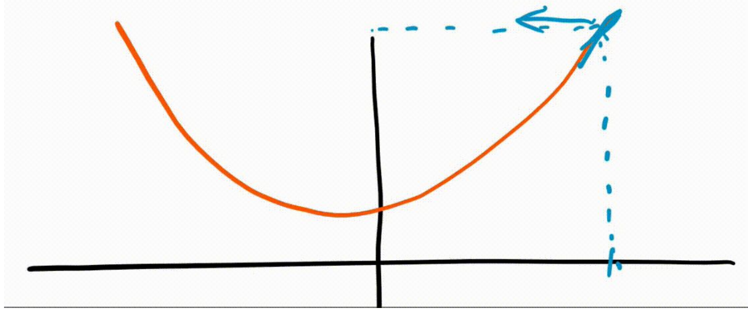
$$\theta = \theta - \varepsilon \frac{1}{m} \sum_{i \in B} [\nabla_\theta \ell(f(X_i, \theta), Y_i) + \lambda \nabla_\theta \Omega(\theta)].$$

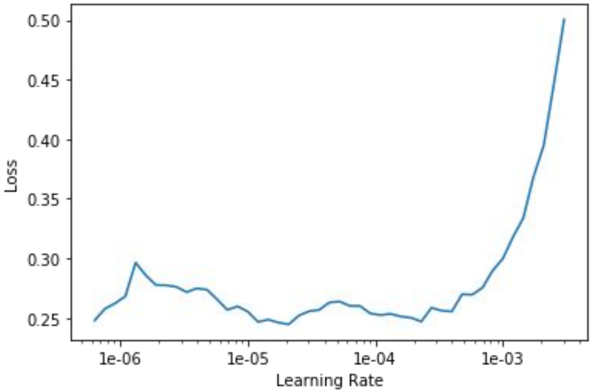
B is a subset of cardinality *m* called **batch.** An iteration over the dataset is called **epoch** and with a **learning rate** ϵ

1.3 Optimization details

About the **learning rate**:







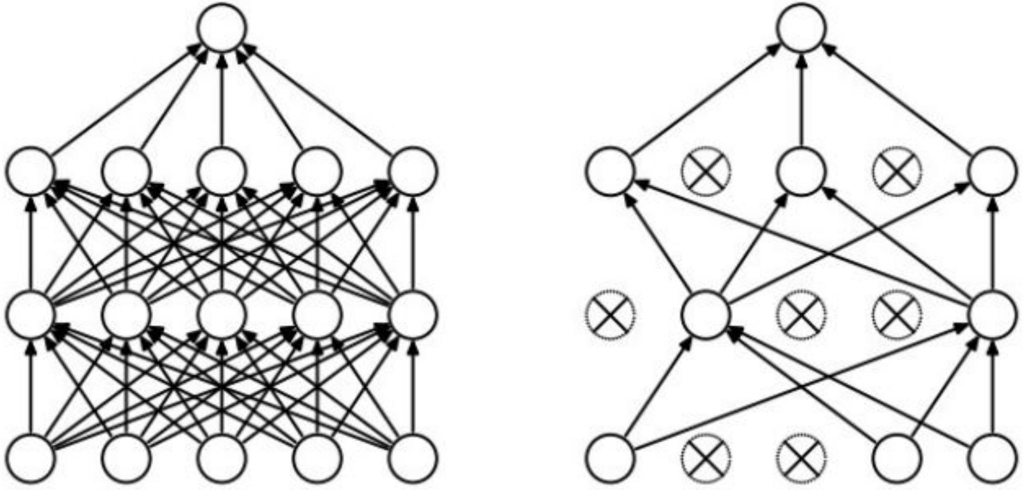
1.3 Optimization details

Dropout (Hinton, 2012)

Increase generalization

Used with

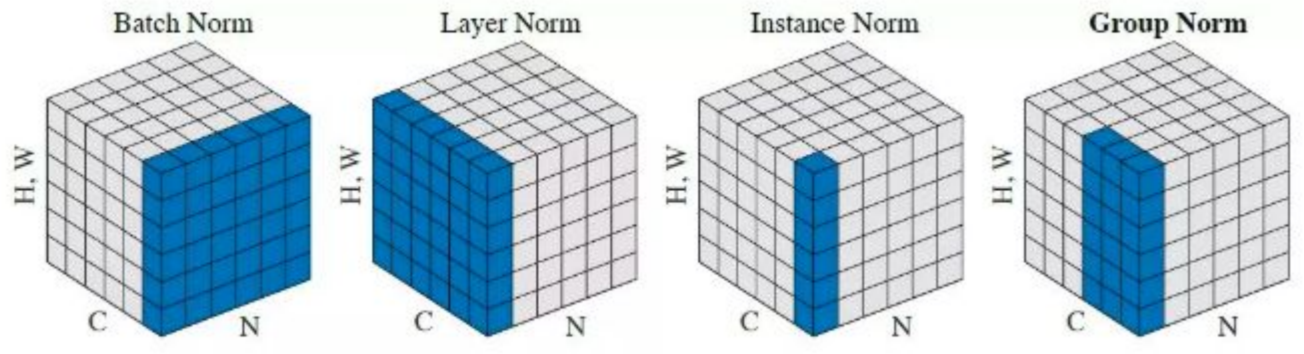
- Data augmentation
- Adversarial examples



(a) Standard Neural Net (b) After applying dropout.

1.3 Optimization details

About data **normalization**

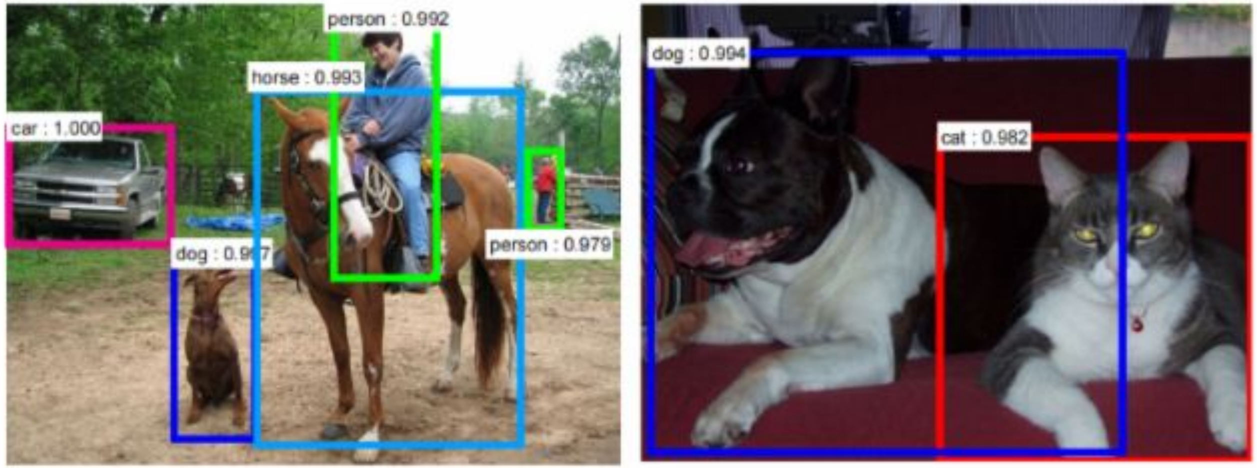


1.4 CNN

MLP not easy to adopt for 2D(3D) data → transform it into some feature vectors

Vectors → how to conserve the spatial relations?

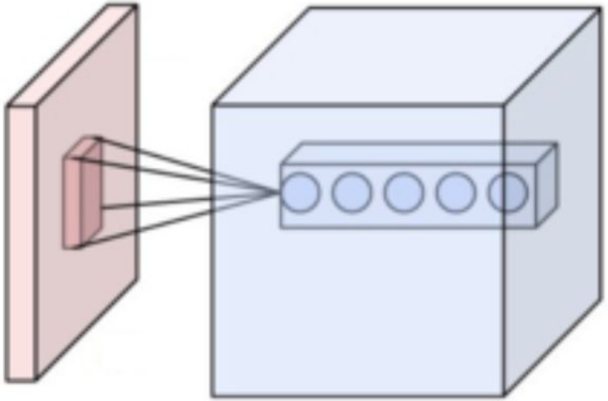
CNN (LeCun, 1998) → automatic vector extraction



1.4 CNN - convolution operator

Definition for * on function *f* and *g*: $$(f * g)(x) = \sum_t f(t)g(x+t).$$

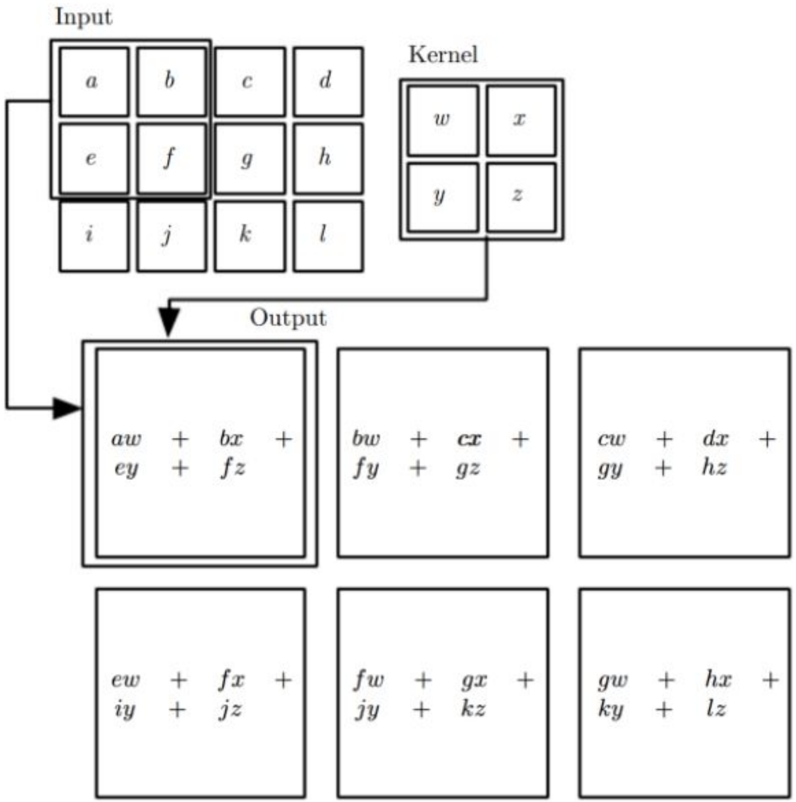
For 2D(*I*) data **kernel** *K* is used: $$(K * I)(i,j) = \sum_{m,n} K(m,n)I(i+n, j+m).$$



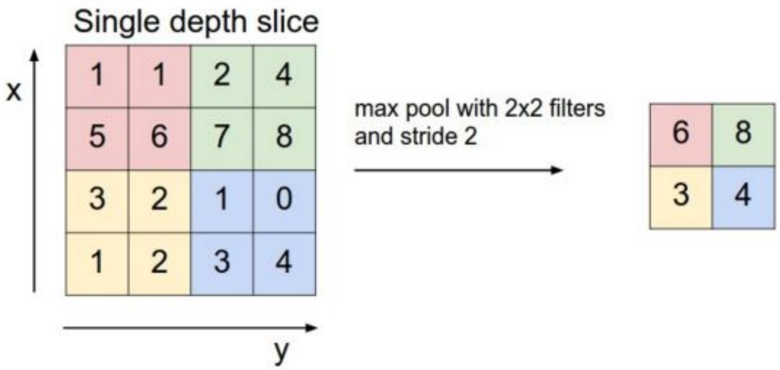
1.4 CNN - convolution operator

Example

$$K_l * I(i,j) = \sum_{c=0}^{2}\sum_{n=0}^{4}\sum_{m=0}^{4} K_l(n,m,c)I(i+n-2, i+m-2, c).$$

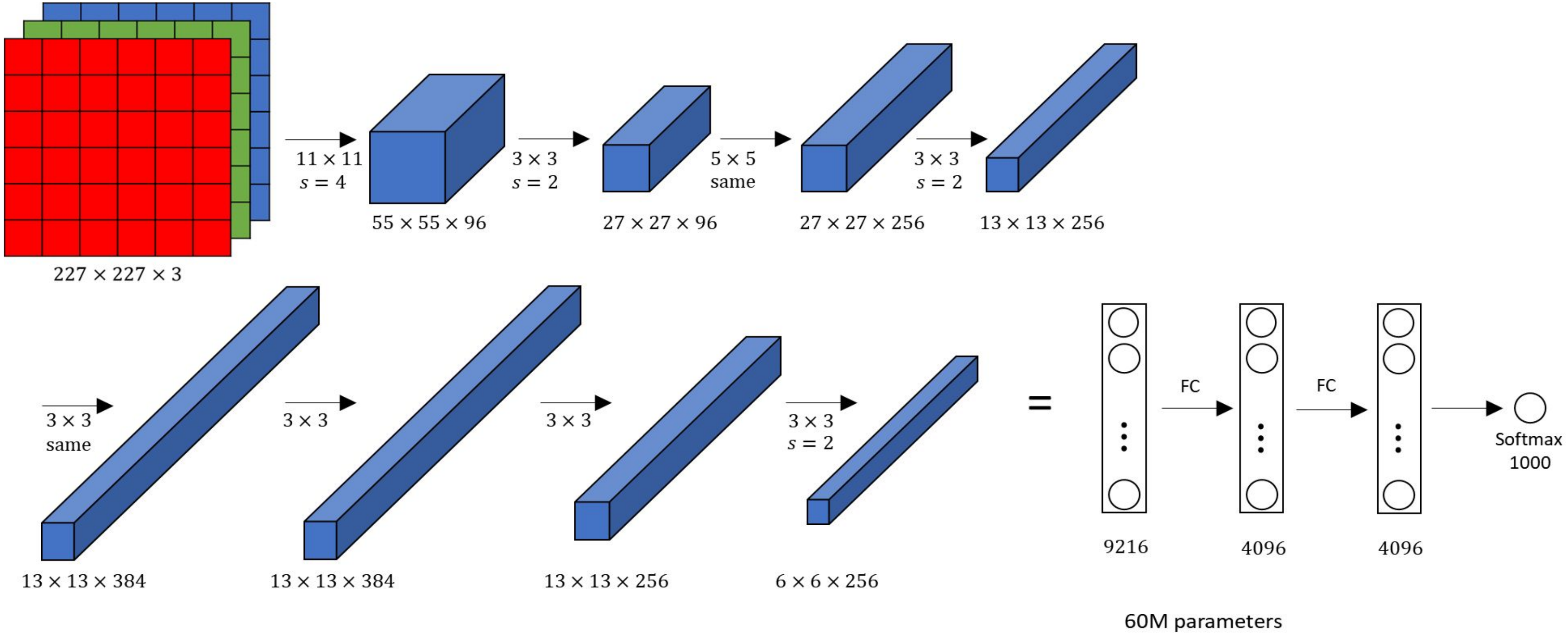


Pooling (no padding)



1.4 CNN - common architectures

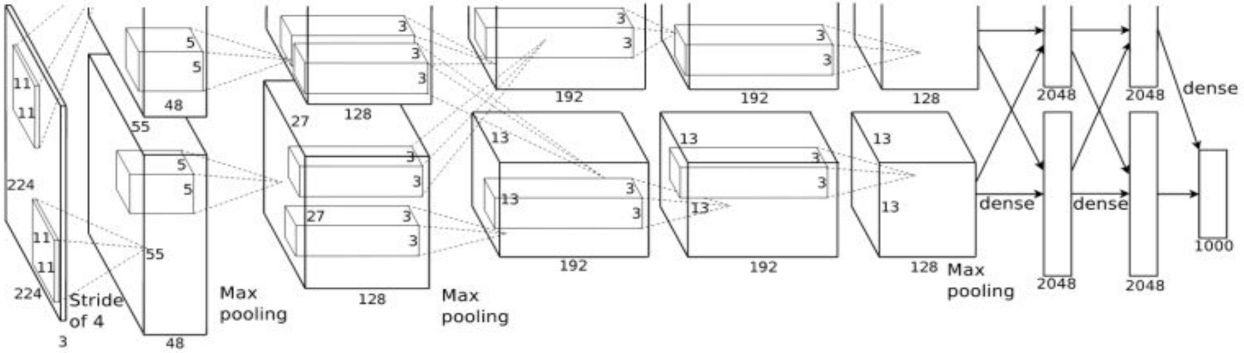
E.g Alexnet



1.4 CNN - common architectures - description

Alexnet

Code:

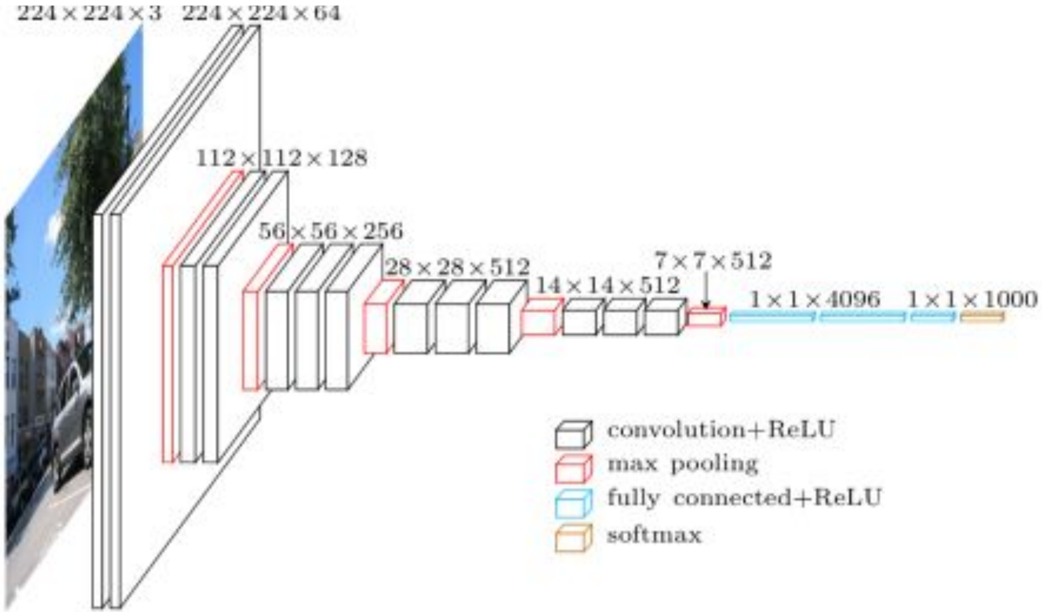


Input	227 * 227 * 3			
Conv 1	55*55*96	96	11 *11	filters at stride 4, pad 0
Max Pool 1	27*27*96		3 *3	filters at stride 2
Conv 2	27*27*256	256	5*5	filters at stride 1, pad 2
Max Pool 2	13*13*256		3 *3	filters at stride 2
Conv 3	13*13*384	384	3*3	filters at stride 1, pad 1
Conv 4	13*13*384	384	3*3	filters at stride 1, pad 1
Conv 5	13*13*256	256	3*3	filters at stride 1, pad 1
Max Pool 3	6*6*256		3 *3	filters at stride 2
FC1	4096	4096	neurons	
FC2	4096	4096	neurons	
FC3	1000	1000	neurons	(softmax logits)

1.4 CNN - common architectures

VGG

More to be found on ModelZoo from Nvidi



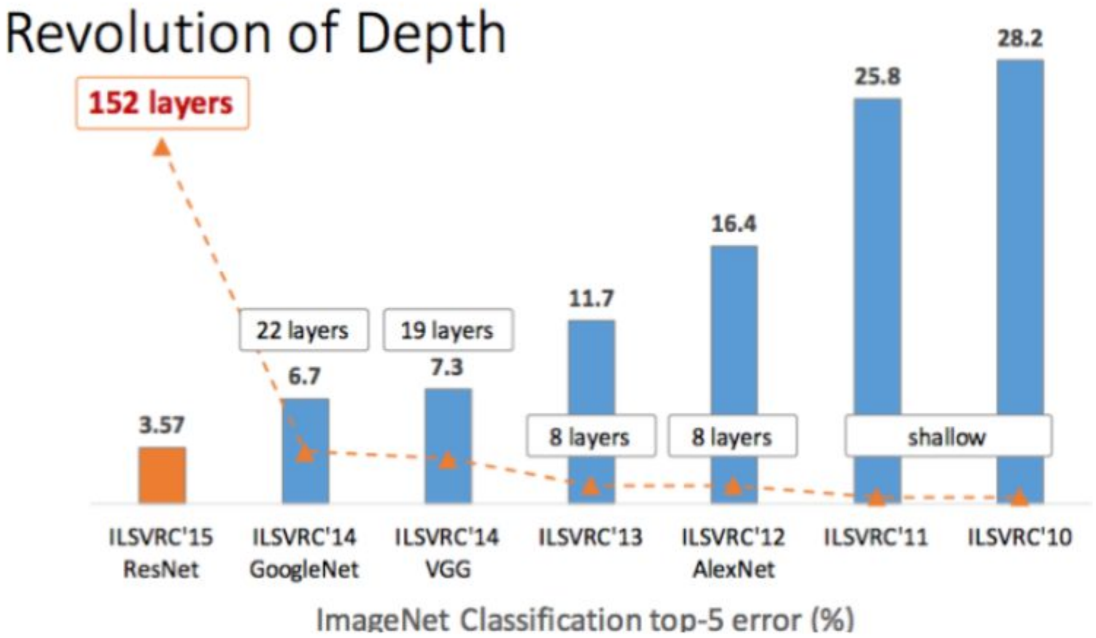
1.4 CNN - evolution of depth

Reduction of params

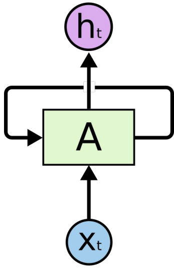
Increase of depth

Pruning networks

Targeted devices (TRT)

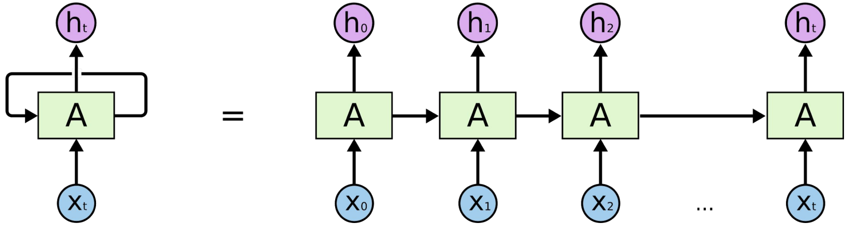


1.5 RNN & LSTM



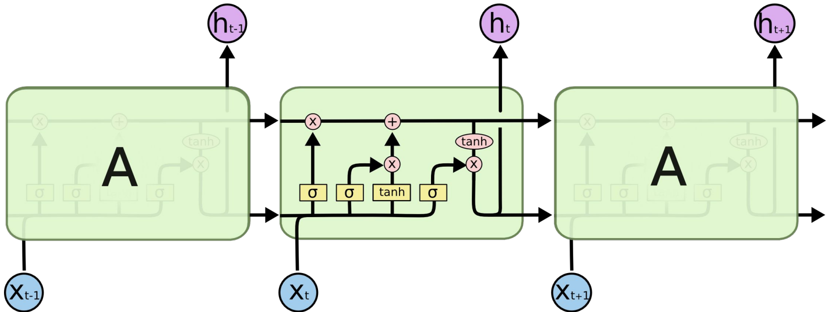
RNN (Jordan, 1990): information from past as well

LSTM (Schmidhuber, 1997): special form of RNN



Slightly more complex internal structure as in case of RNN:

C Olah.

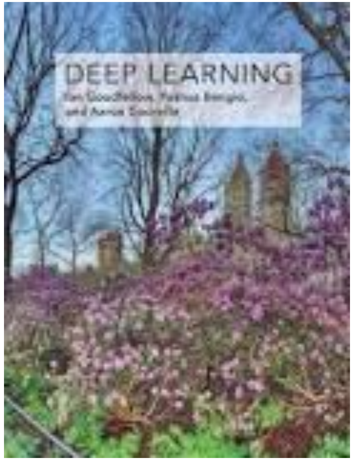


1. Summary

Promising research direction

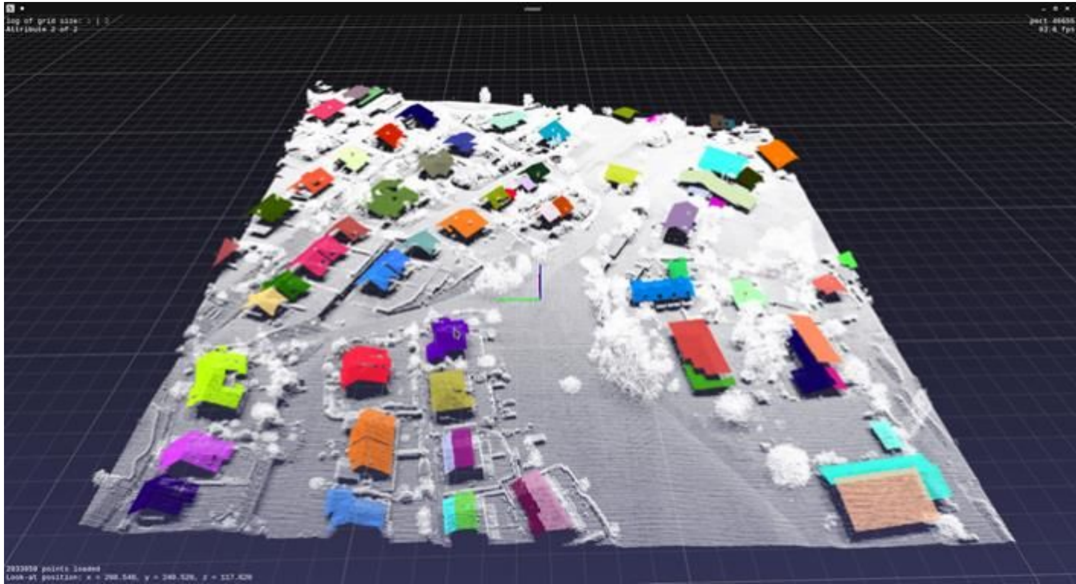
Stay tuned!

Good starting point:



Content

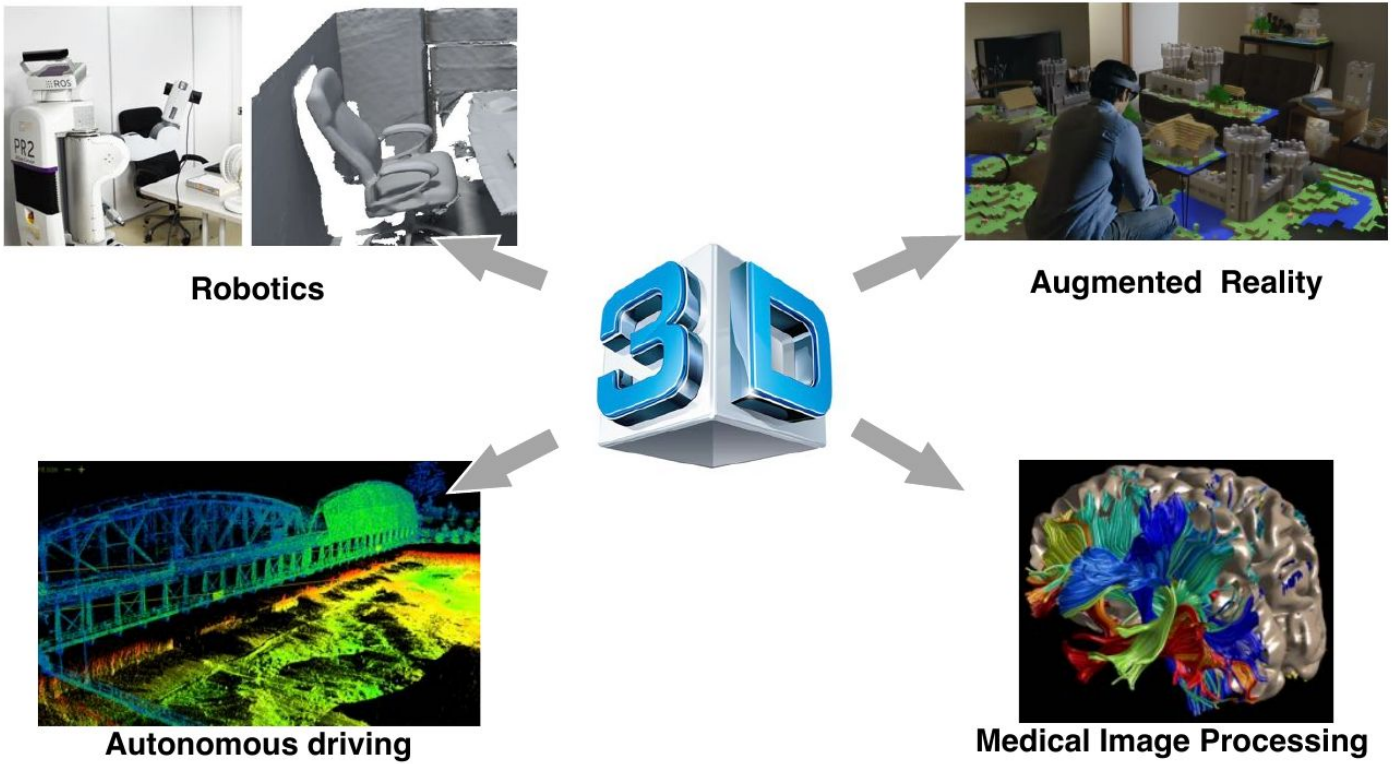
1. Preliminary ideas on DL
2. **Why 3D data with deep learning?**
3. Classification
4. Segmentation
5. Perspectives



3D DL

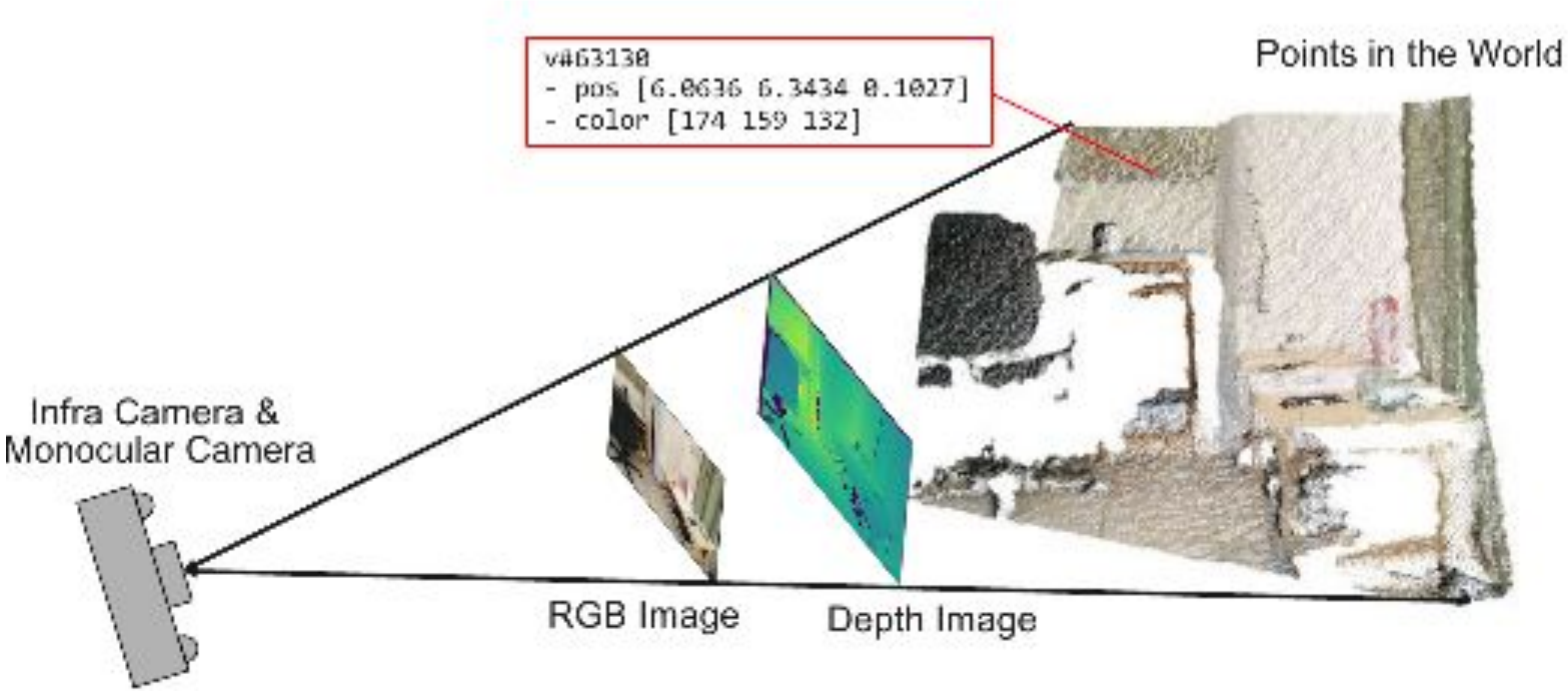
World is in 3D...

Motivation


Robotics
Augmented Reality
Autonomous driving
Medical Image Processing

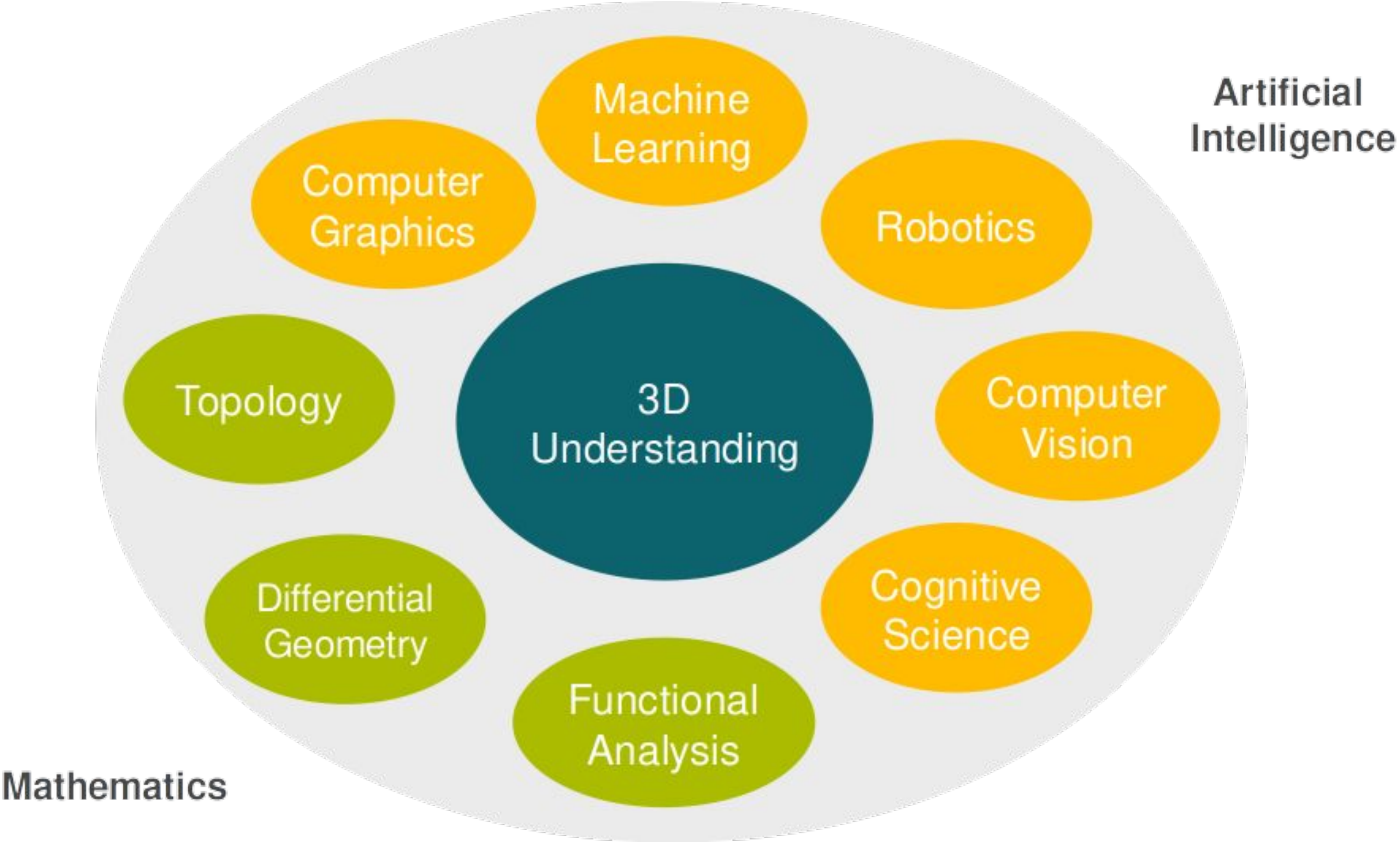
2D, 2.5D, 3D ?

From 2D to 3D



DL for 3D?

Now happening

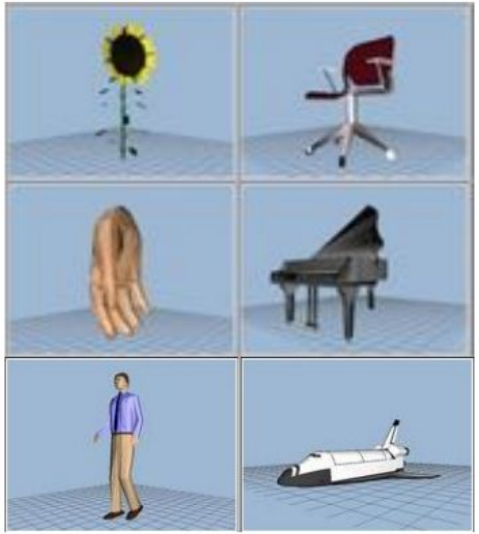


Motivation - lack of data/model ~10 years ago

Stanford bunny

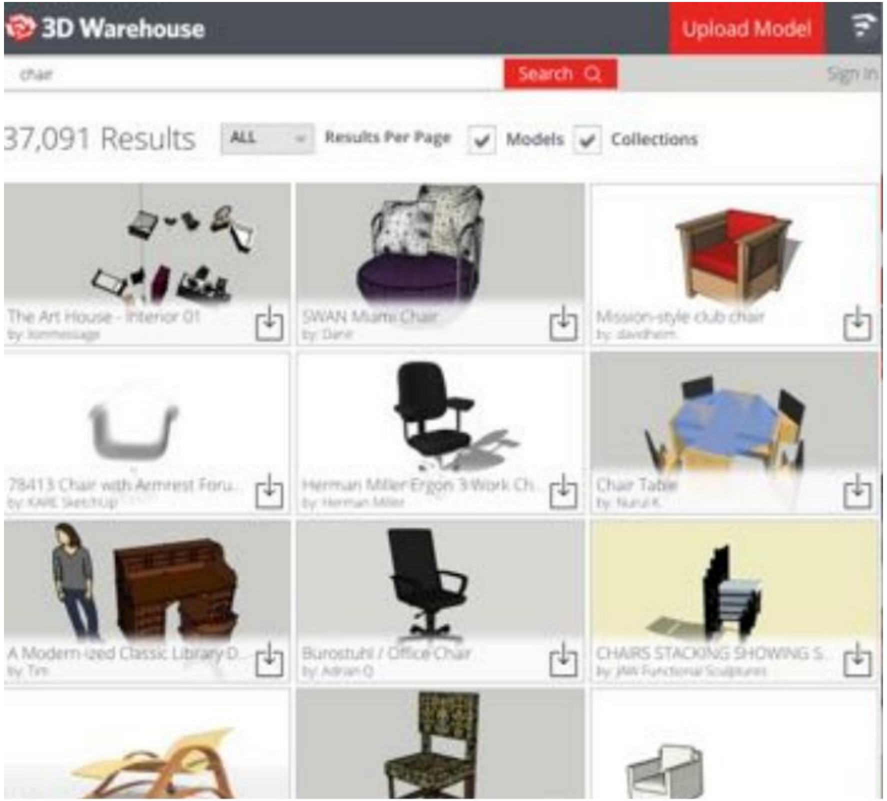
Utah teapot

1800 models in **90** categories



Princeton shape benchmark
[Shilane et al. 04]

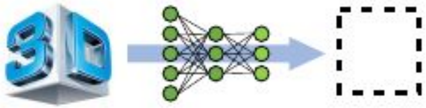
Motivation - plenty of data/model today



Motivation - plenty of data/model today

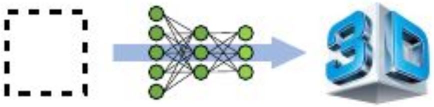
3D deep learning tasks

3D geometric analysis



3D assisted image analysis

3D synthesis

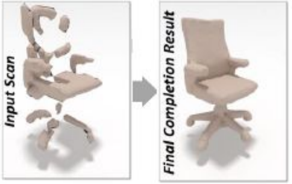


Classification

Parsing (object/scene)

Correspondence

Monocular 3D reconstruction



Shape completion

Shape modeling

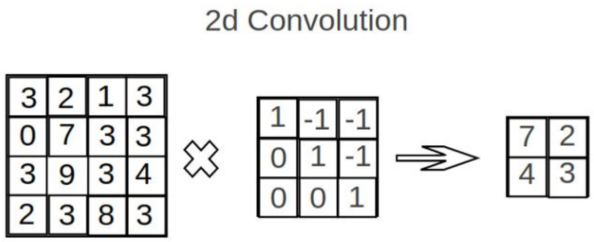
3D representation for DL

2D images: uniqueness in representation, plays well with * operator

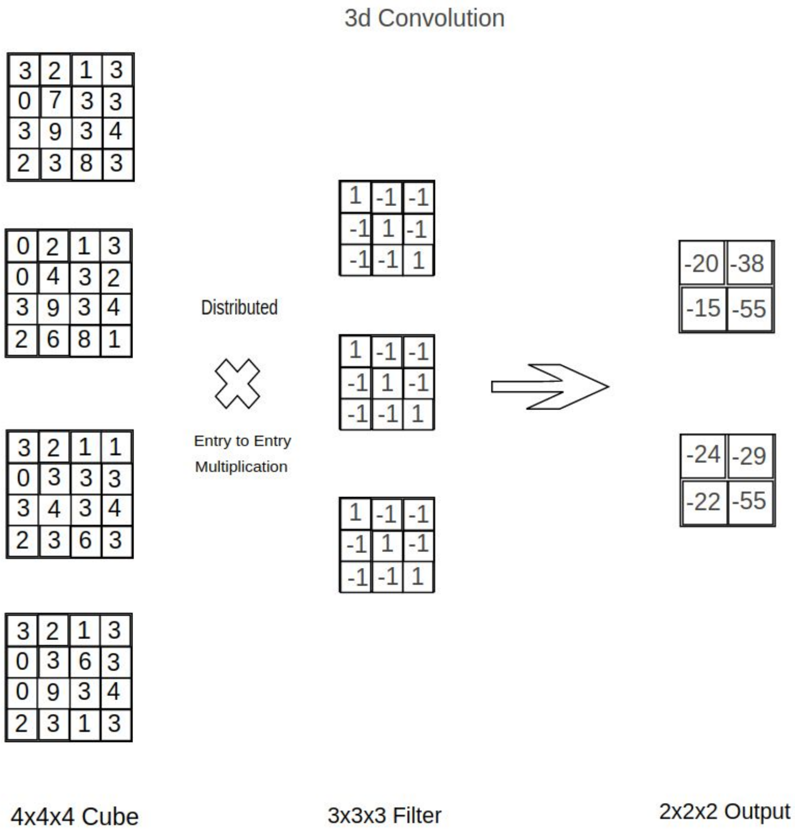
1	44	33	12	20	23	35	14
51	16	40	32	46	48	28	17
29	60	3	63	49	55	36	7
52	22	26	41	38	10	61	53
2	24	19	11	34	43	5	8
57	9	37	42	25	21	27	18
30	56	50	64	4	59	6	13
58	47	45	31	39	15	62	54

Unordered point clouds → not that easy!

3D representation for DL - some 2D analogy



Order is still critical!



3D representation for DL

3D representation

- Multiview 2D images
- Volumetric
- Poly Mesh
- Point cloud
- Primite based

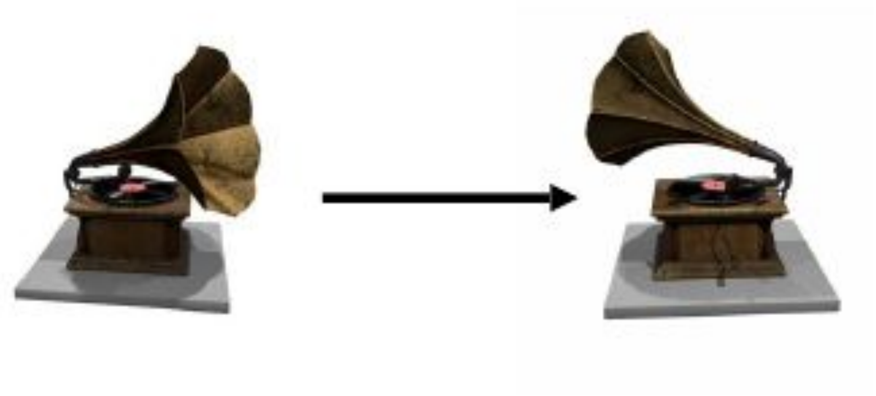
Rasterized (grid)→ direct2D, with challenges

Geometric relation (irregular) → directly CNN

3D representation for DL

3D representation

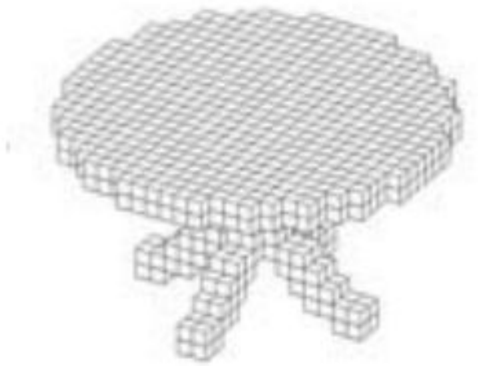
- **Multiview 2D images**
- Volumetric
- Poly Mesh
- Point cloud
- Primite based



3D representation for DL

3D representation

- Multiview 2D images
- **Volumetric**
- Poly Mesh
- Point cloud
- Primite based



3D representation for DL

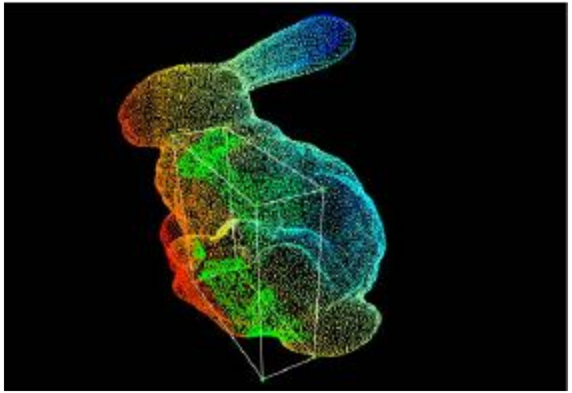
3D representation

- Multiview 2D images
- Volumetric
- **Poly Mesh**
- Point cloud
- Primite based

3D representation for DL

3D representation

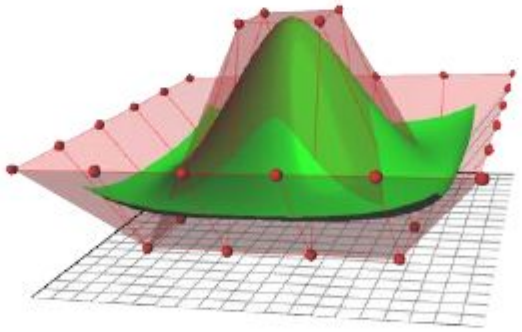
- Multiview 2D images
- Volumetric
- Poly Mesh
- **Point cloud**
- Primite based



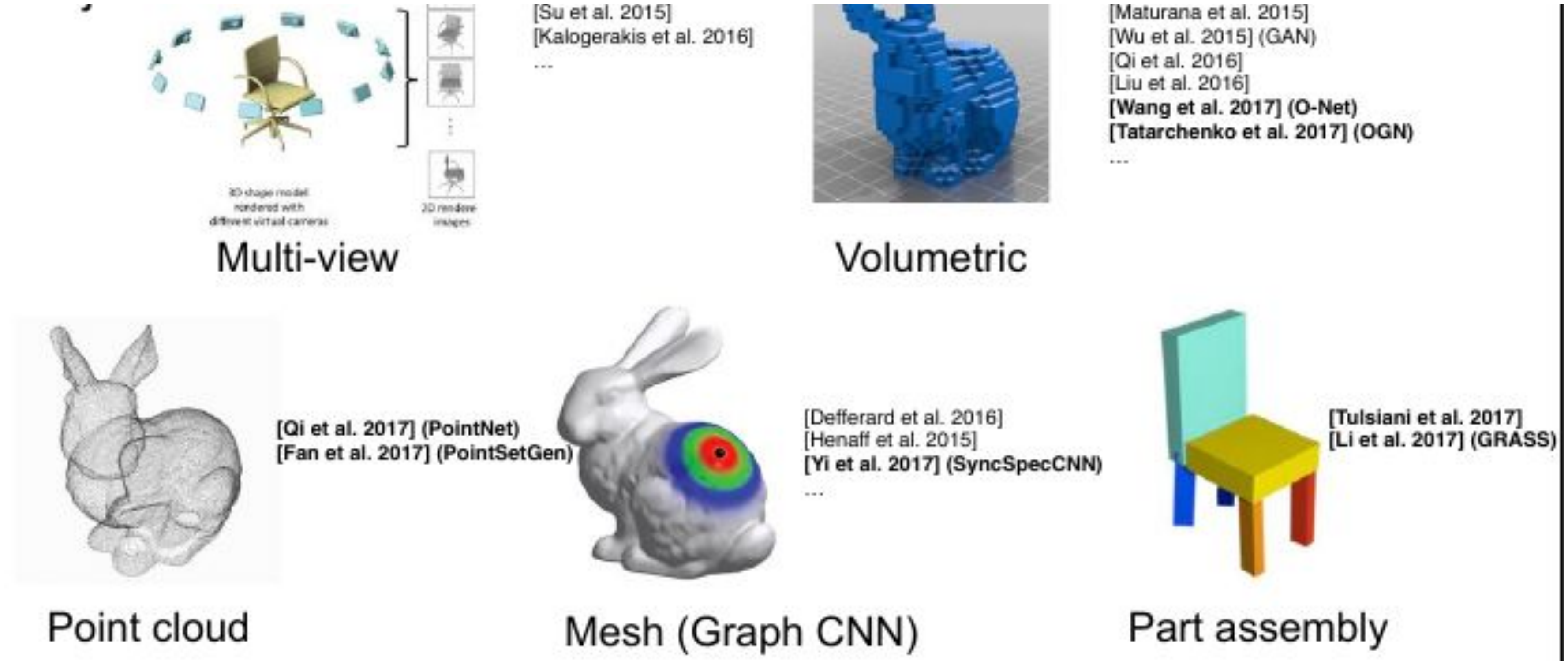
3D representation for DL

3D representation

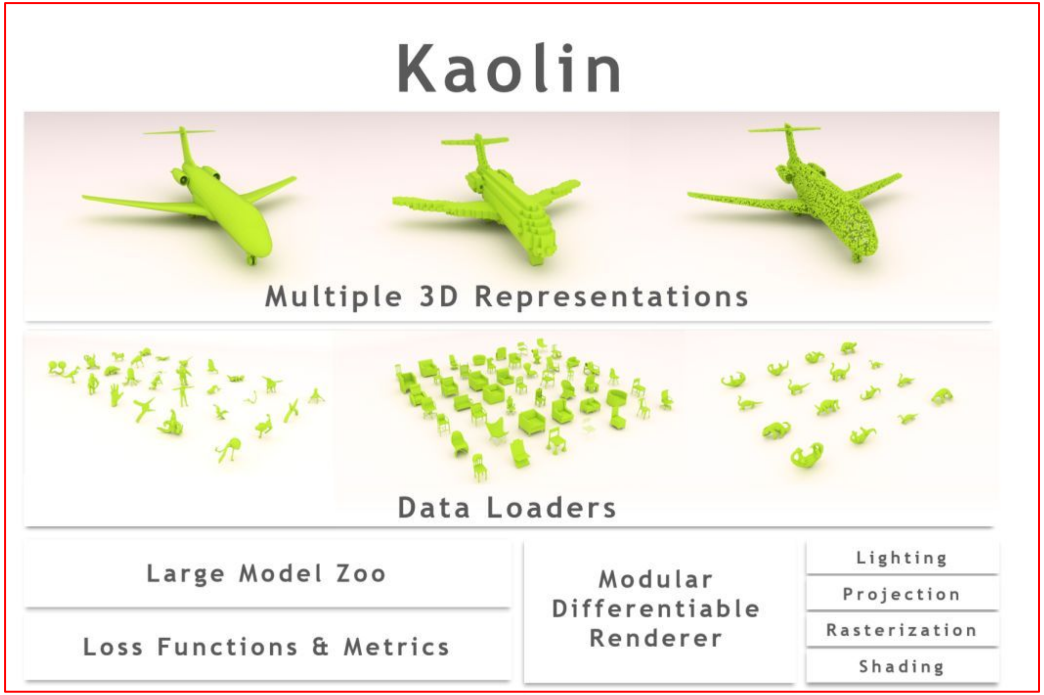
- Multiview 2D images
- Volumetric
- Poly Mesh
- Point cloud
- **Primite based**



3D representation for DL - references



3D representation for DL - tools

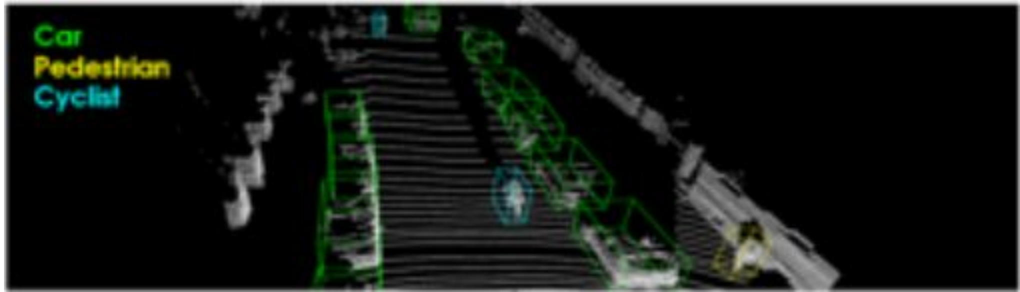


PyTorch3D

OPEN3D

Content

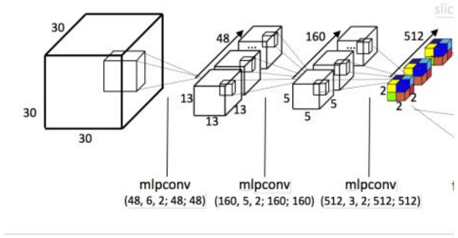
1. Preliminary ideas on DL
2. Why 3D data with deep learning?
3. **Classification**
4. Segmentation
5. Perspectives



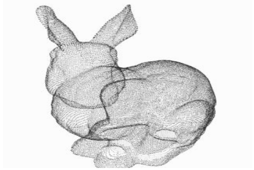
Classification

Multi-view CNN

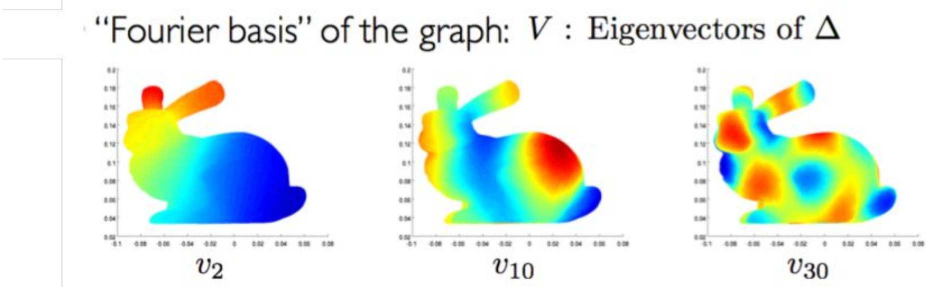
Volumetric CNN



Point nets

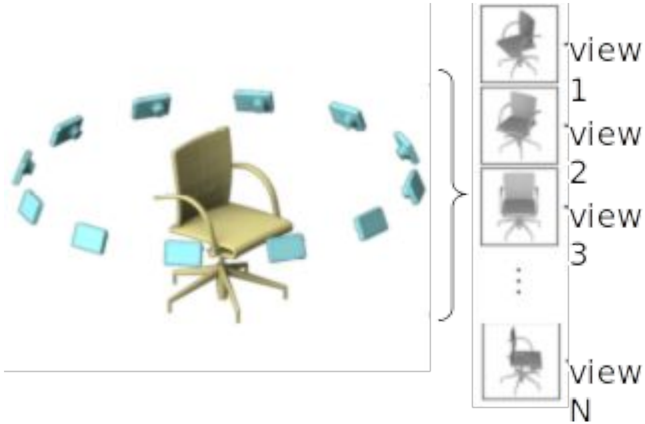


Spectral convolution



Classification: **Multi-view CNN**

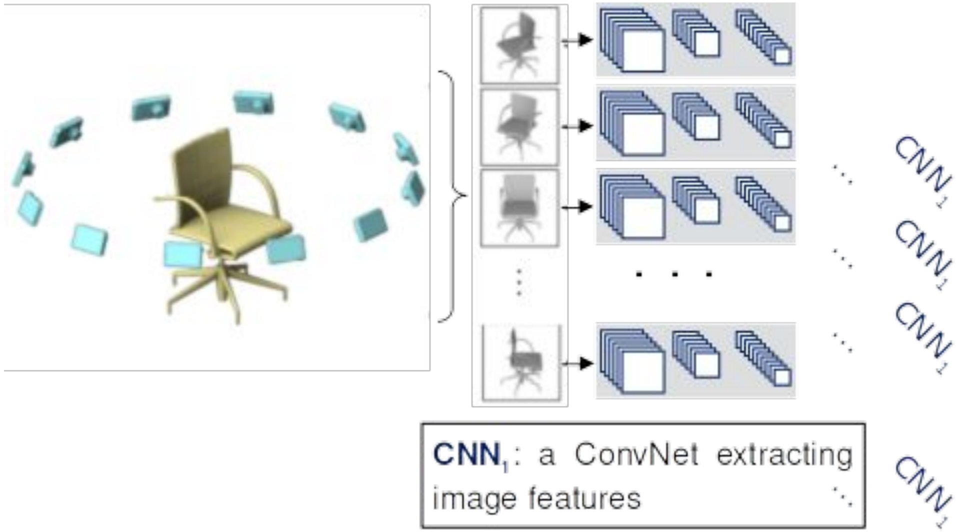
Render with Multiple Virtual Cameras



Su et al., "Multi-view Convolutional Neural Networks for 3D Shape Recognition”, ICCV 2015

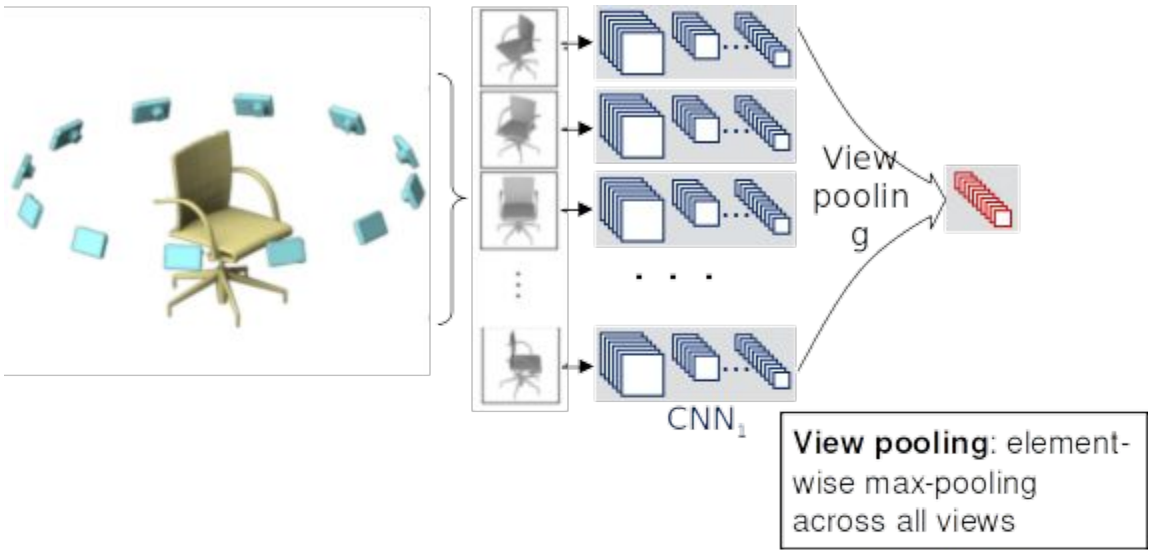
Classification: Multi-view CNN

Images are Passed through CNN1 for Image Features



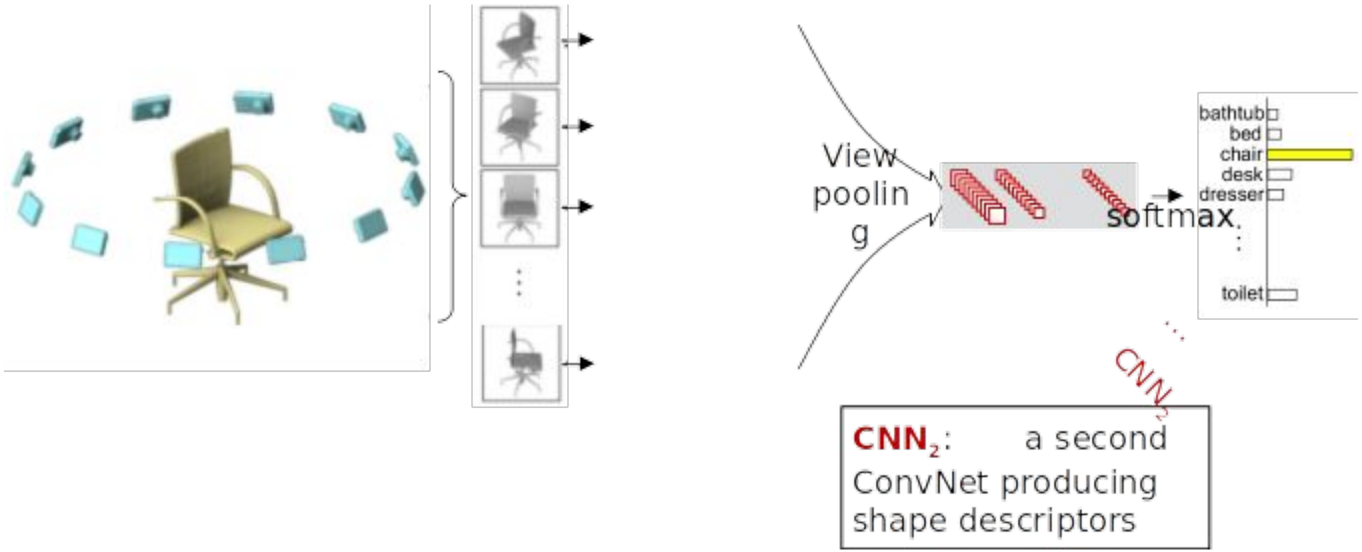
Classification: Multi-view CNN

All Image Features are Combined by View Pooling



Classification: Multi-view CNN

and then Passed through CNN2 and to Generate Final Predictions



Classification: Multi-view CNN

Experiments – Classification & Retrieval

Method	Classification (Accuracy)	Retrieval (mAP)
SPH [16]	68.2%	33.3%
LFD [5]	75.5%	40.9%
3D ShapeNets [37]	77.3%	49.2%
FV, 12 views	84.8%	43.9%
CNN, 12 views	88.6%	62.8%
MVCNN, 12 views	**89.9%**	70.1%
MVCNN+metric, 12 views	89.5%	**80.2%**
MVCNN, 80 views	90.1%	70.4%
MVCNN+metric, 80 views	**90.1%**	**79.5%**

Classification: Multi-view CNN

Summary

- Gives good performance
- Can leverage vast literature of image classification
- Can use pertained features
-
- What if the input is noisy and/or incomplete? **e.g., point cloud**

Classification: **Volumetric CNN**

Main ideas:

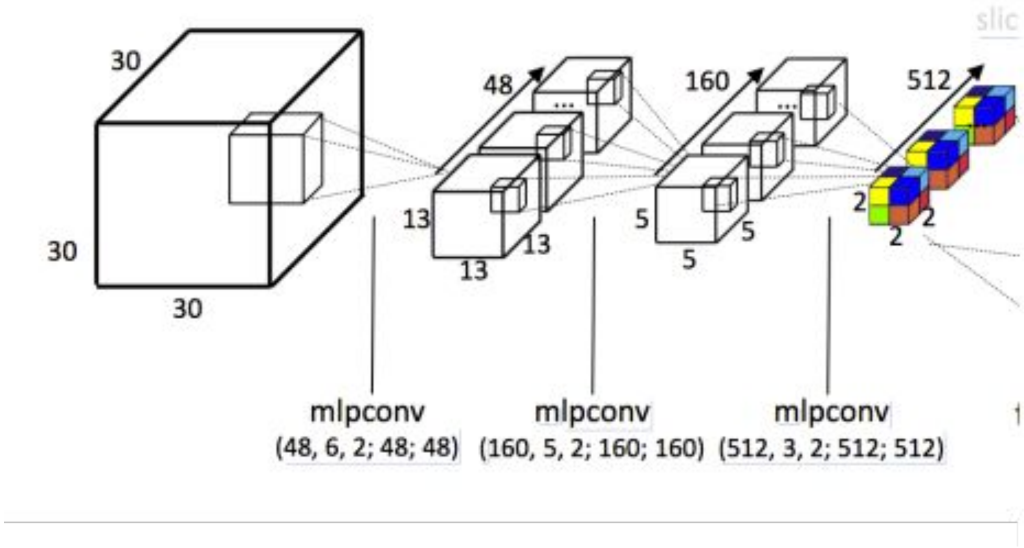
- Use CNN without explicit 3D-2D projection
- Make use of 3D native convolution (aka 4D CNN)
- Represent the occupied space with voxel grids

Classification: **Voxelization**

Represent the occupancy of regular 3D grids

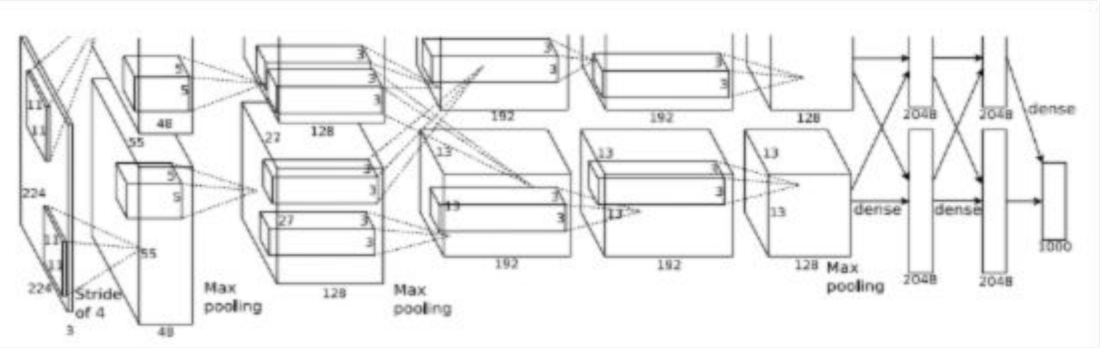
3D CNN on Volumetric Data

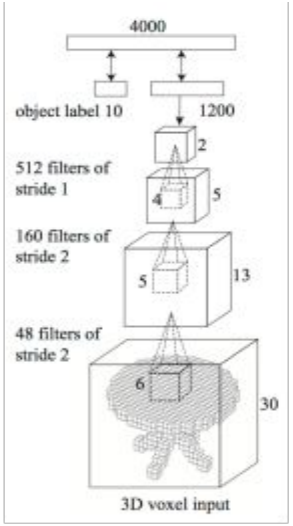
3D convolution uses 4D kernels



Complexity issues

Compared with 2D cases





AlexNet, 2012

Input resolution: 224x224

224x224=50176

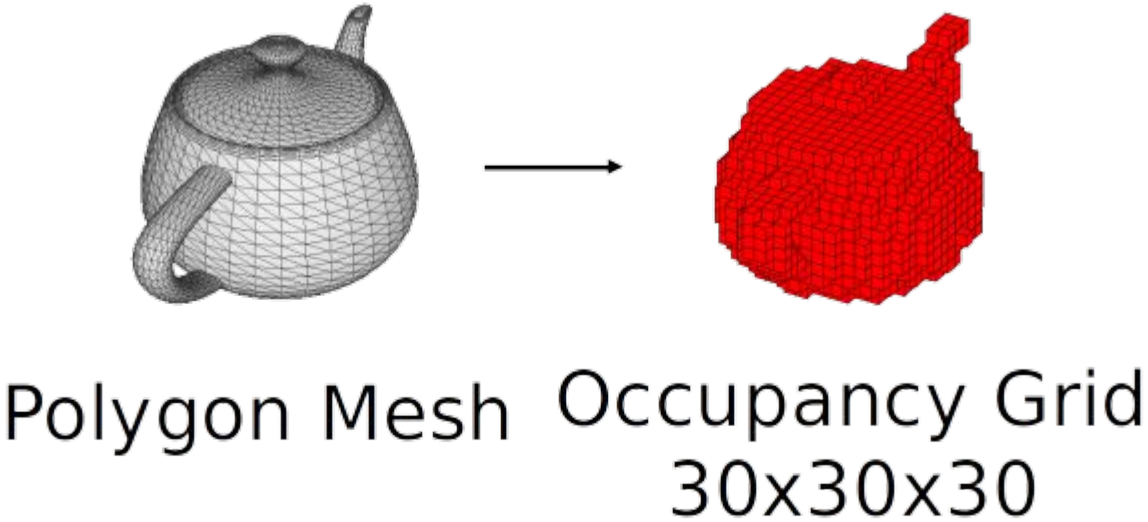
3DShapeNets, 2015

Input resolution: 30x30x30

224x224=27000

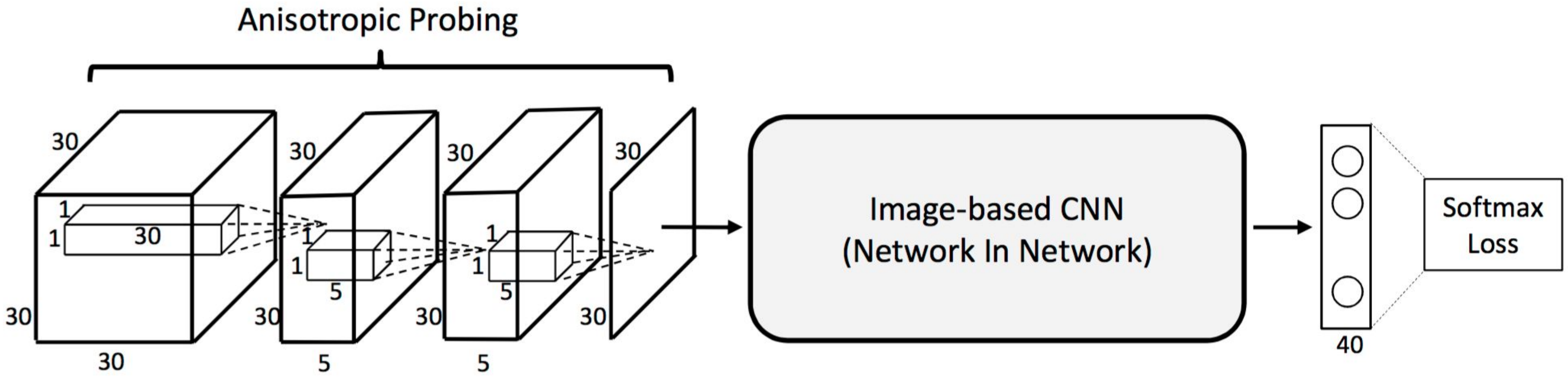
Complexity issues

What about information loss?



Basic idea: learn to project

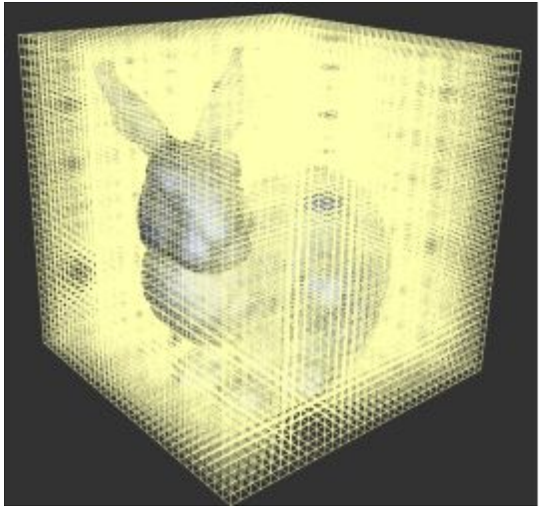
By **ray tracing** and 2D CNN low param number/low runtime is obtained

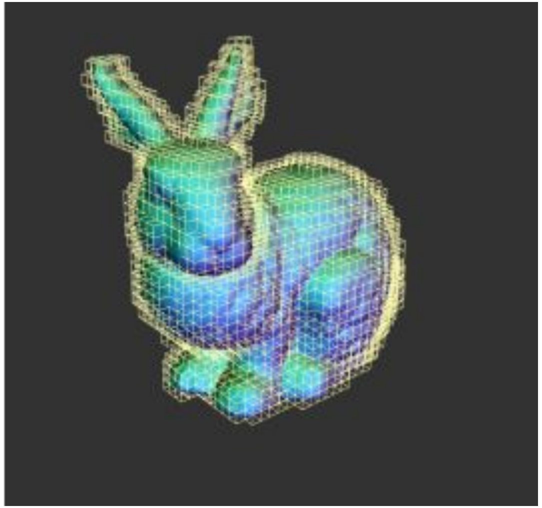


Su et al., "**Volumetric and Multi-View CNNs for Object Classification on 3D Data**", *CVPR 2016*

Voxel vs occupancy grids

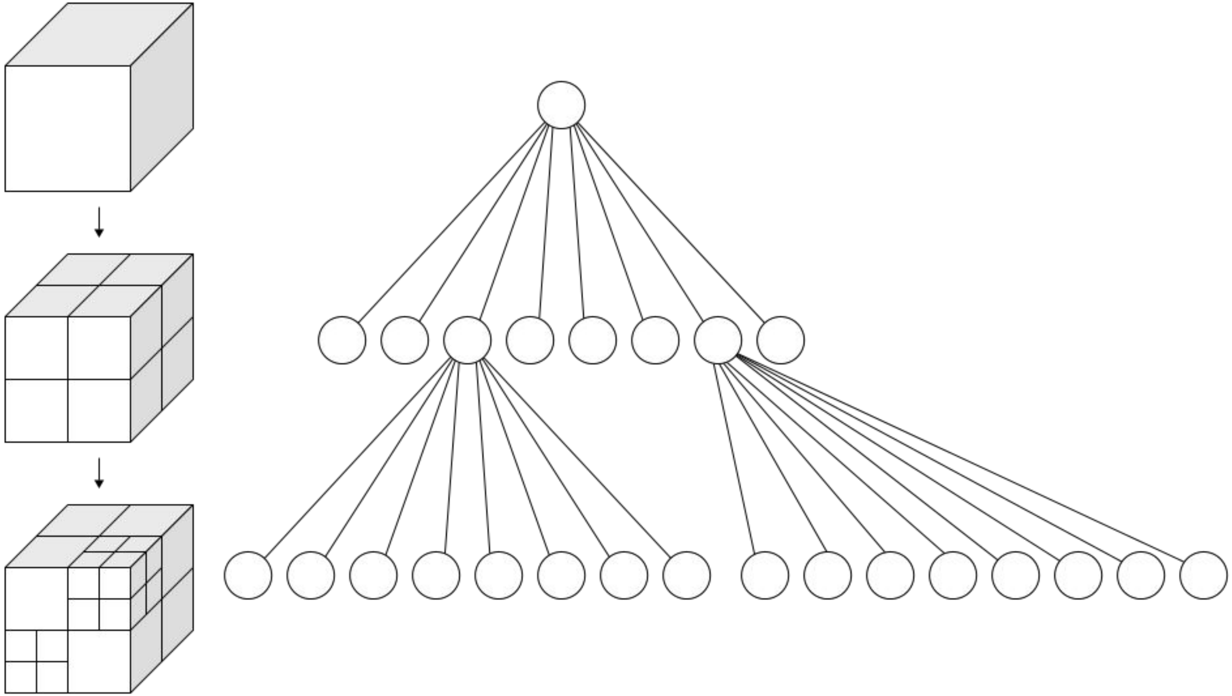
- Store the sparse surface signals
- Constrain the computation near the surface





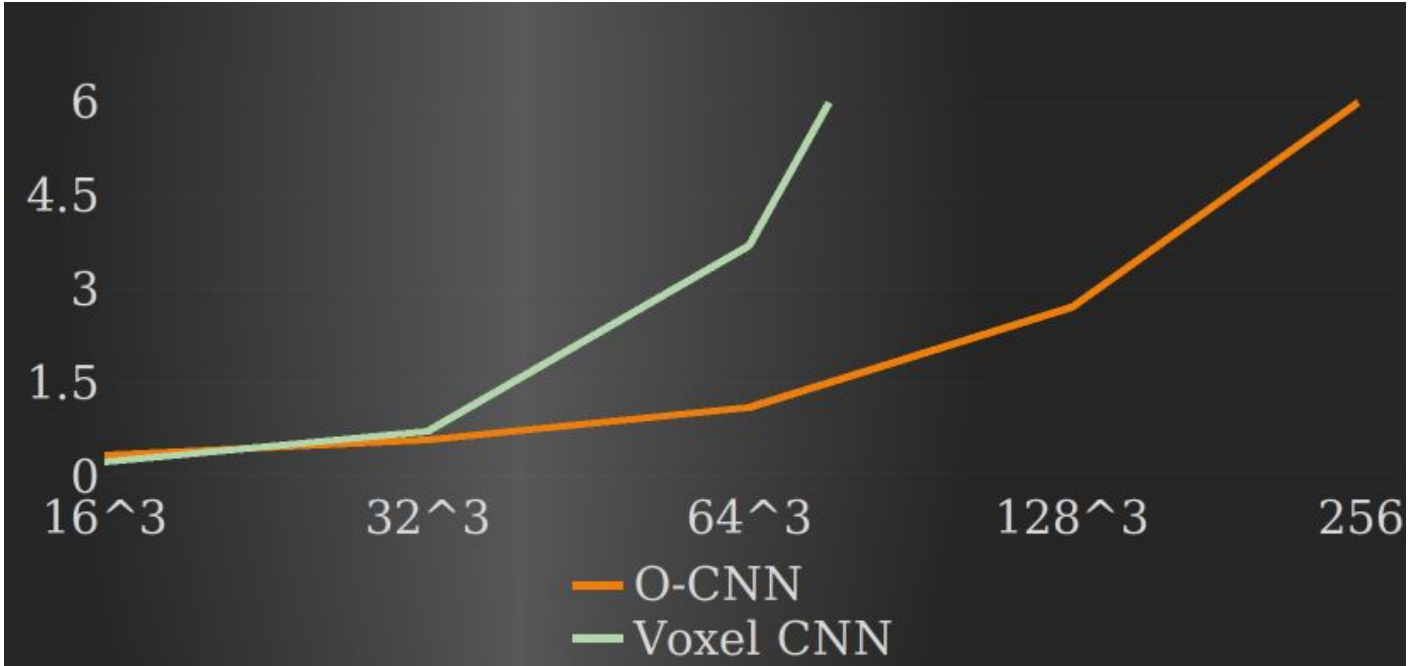
Optimized variant: octree

8 (oct) leaves for each node. Searching very efficient.



Memory efficiency

SparsconvNet → designed for octree representation



Classification: **Voxel CNN**

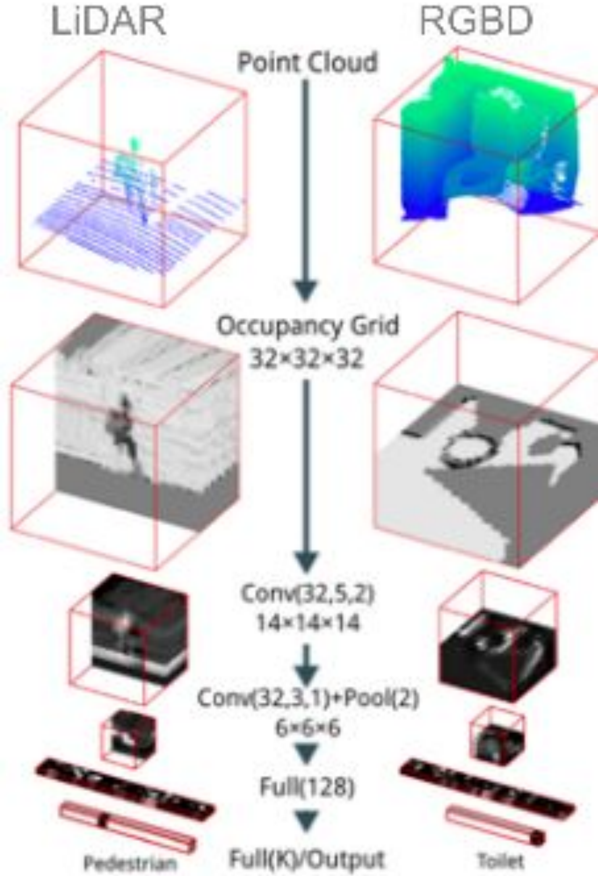
Voxnet

Reco with occupancy grid → prior in robotics

R invariant features + data augmentation

Efficiency

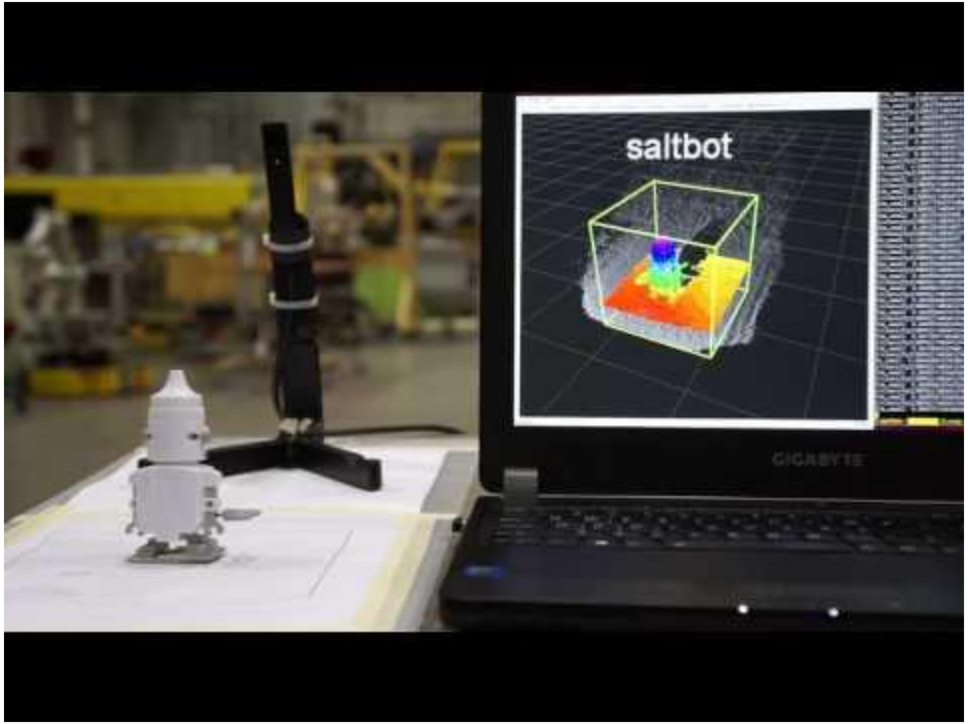
Drawback: only small grids/voxels



Maturana et al. VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition. IROS. 2015.

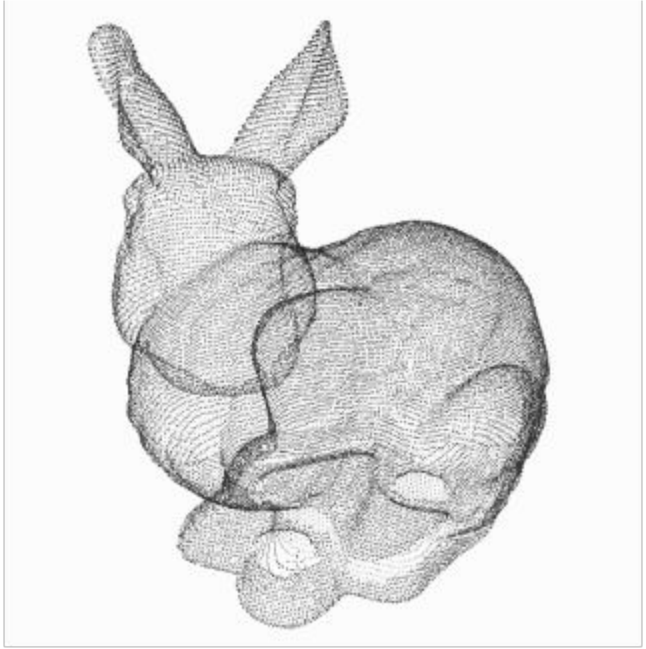
Classification: **Voxel CNN**

Voxnet - demo



Maturana et al. VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition. IROS. 2015.

Classification: **Point networks**



Point cloud

(The most common 3D sensor data)

Directly Process Point Cloud Data

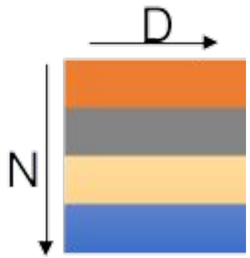
End2end learning for:

- Unstructured
- Unordered

Qi, Charles R., et al. "Pointnet: Deep learning on point sets for 3d classification and segmentation”, CVPR 2017

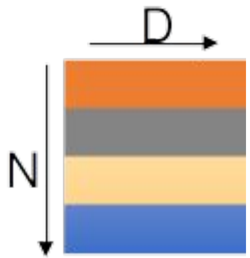
Ensure permutation invariance

Point cloud: **N odorless** points, each represented by a D dim coordinate

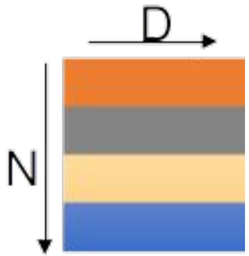


Ensure permutation invariance

Point cloud: **N odorless** points, each represented by a D dim coordinate



represents the same set as

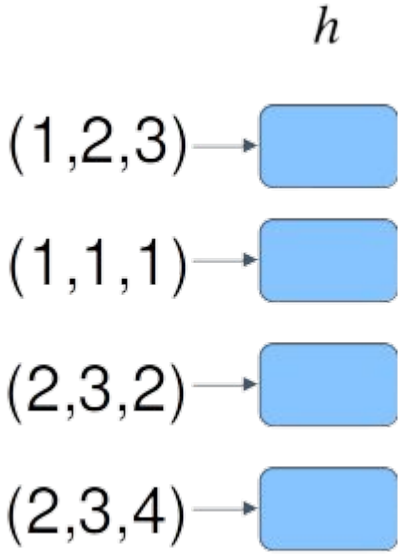


2D array representation

How to cope with this?

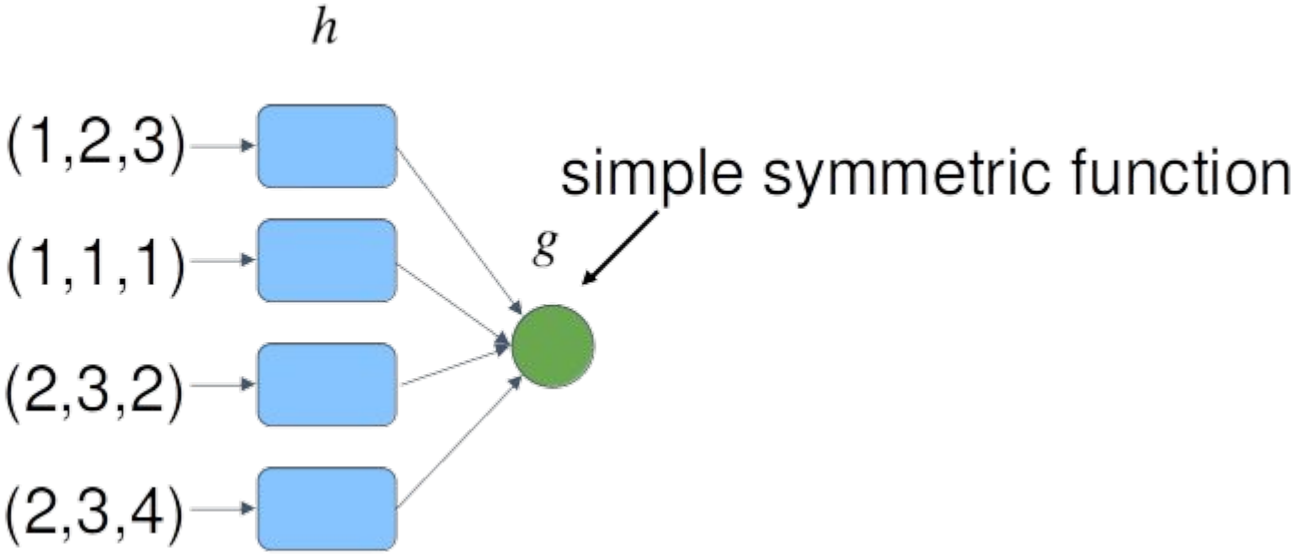
Construct a Symmetric Function

$f(x_1, x_2, \ldots, x_n) = \gamma \circ g(h(x_1), \ldots, h(x_n))$ Is symmetric if **g** is symmetric



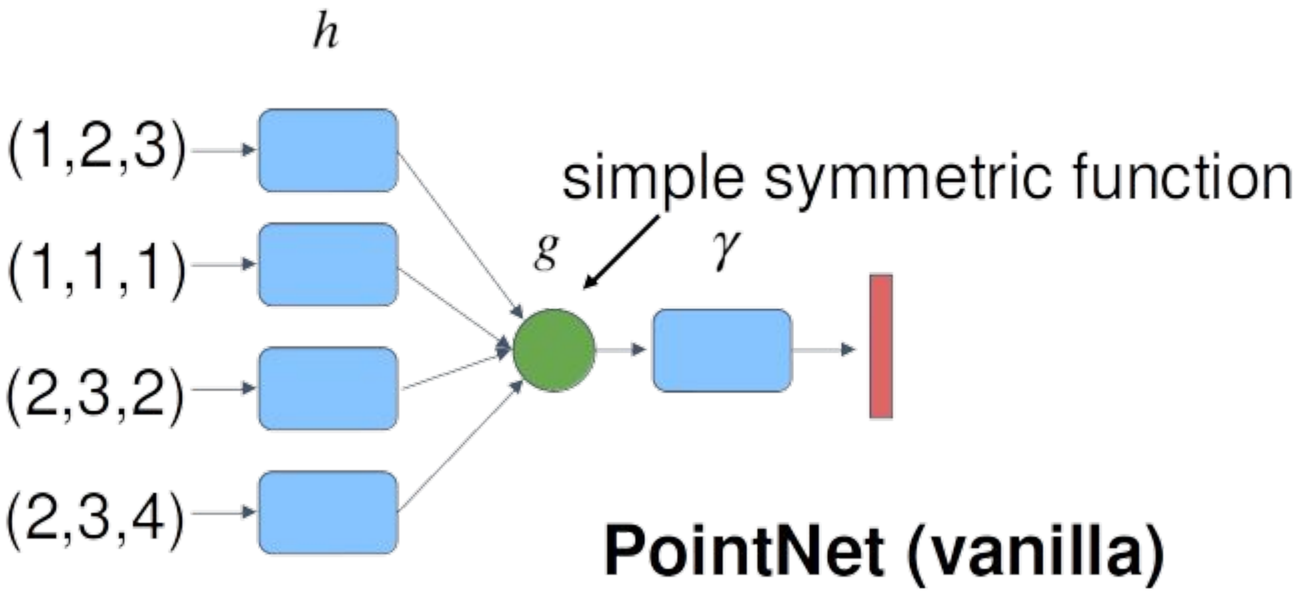
Construct a Symmetric Function

$$f(x_1, x_2, \ldots, x_n) = \gamma \circ g(h(x_1), \ldots, h(x_n))$$ Is symmetric if **g** is symmetric



Construct a Symmetric Function

$$f(x_1, x_2, \ldots, x_n) = \gamma \circ g(h(x_1), \ldots, h(x_n))$$ Is symmetric if **g** is symmetric

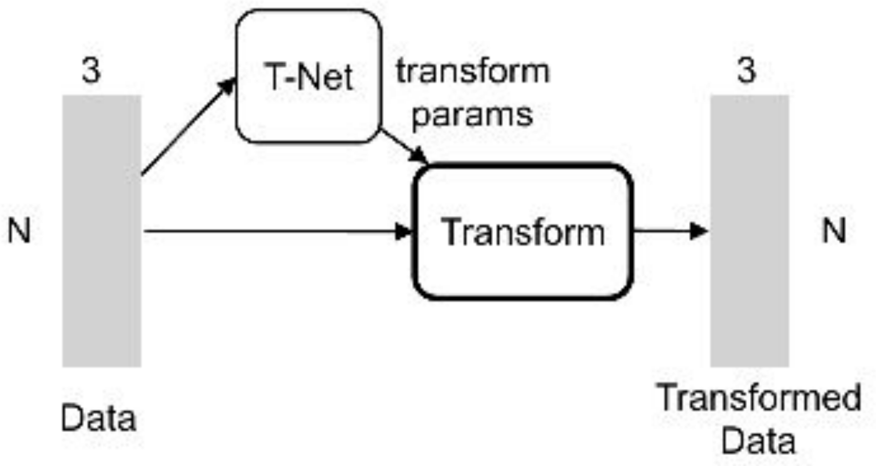


PointNet: geometric transform invariance

Solution: use some simple transform nets (T-Net)

Transform: → matrix multiplication

Dimension (e.g. 3) can be arbitrary for data

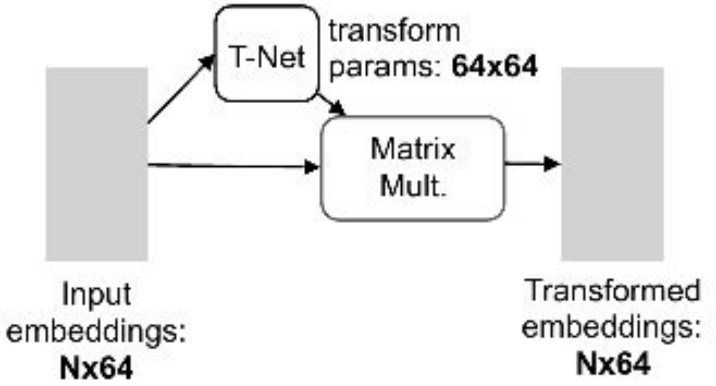


PointNet: geometric transform invariance

Solution: use some simple transform nets (T-Net)

Transform: → matrix multiplication

Dimension (e.g. 3) can be arbitrary for data



Regularization:

Transform matrix A 64x64
close to orthogonal:

$$L_{reg} = \|I - AA^T\|_F^2$$

PointNet: architecture

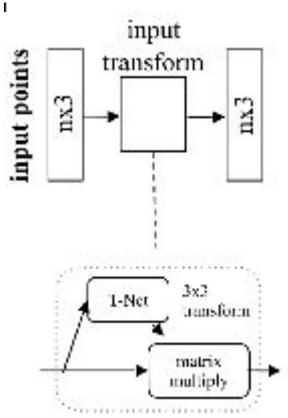
Composition of T-Nets

PointNet: architecture

Composition of T-Nets

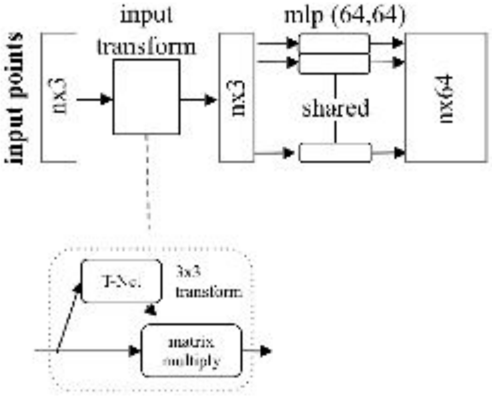
PointNet: architecture

Composition of T-Nets



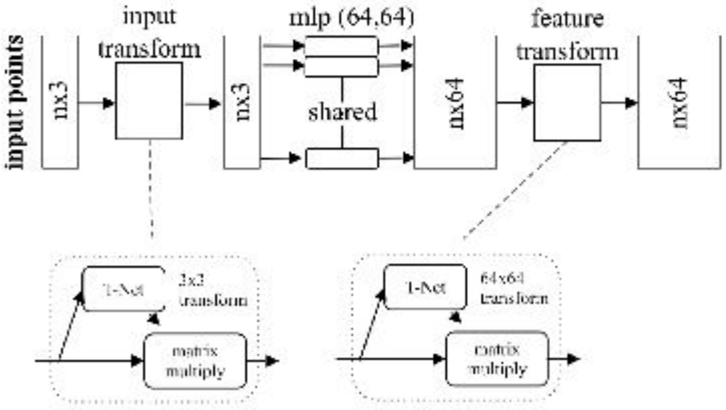
PointNet: architecture

Composition of T-Nets



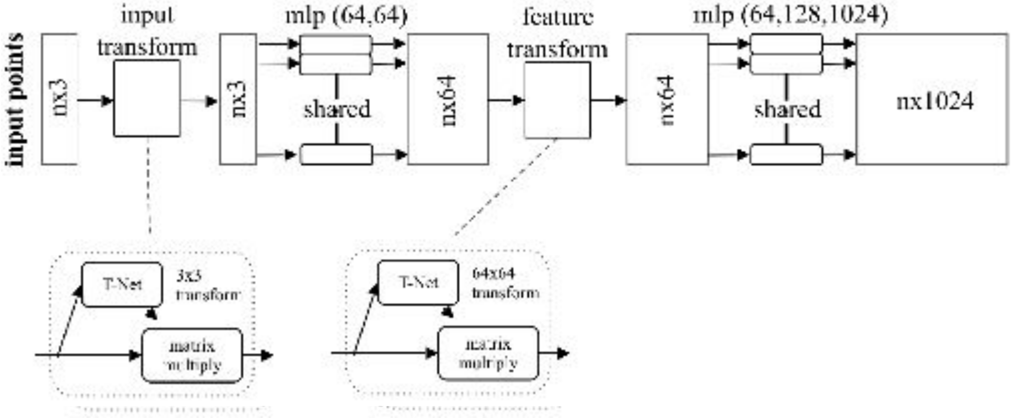
PointNet: architecture

Composition of T-Nets



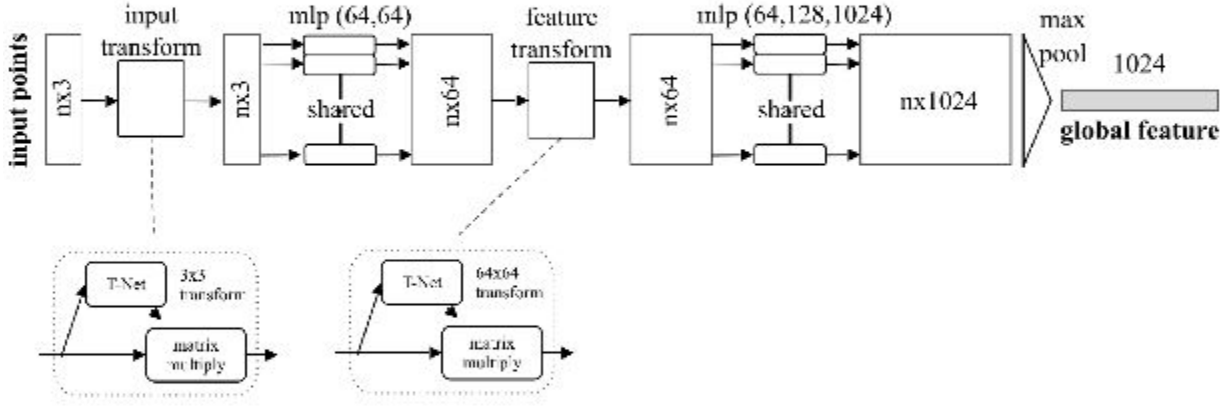
PointNet: architecture

Composition of T-Nets



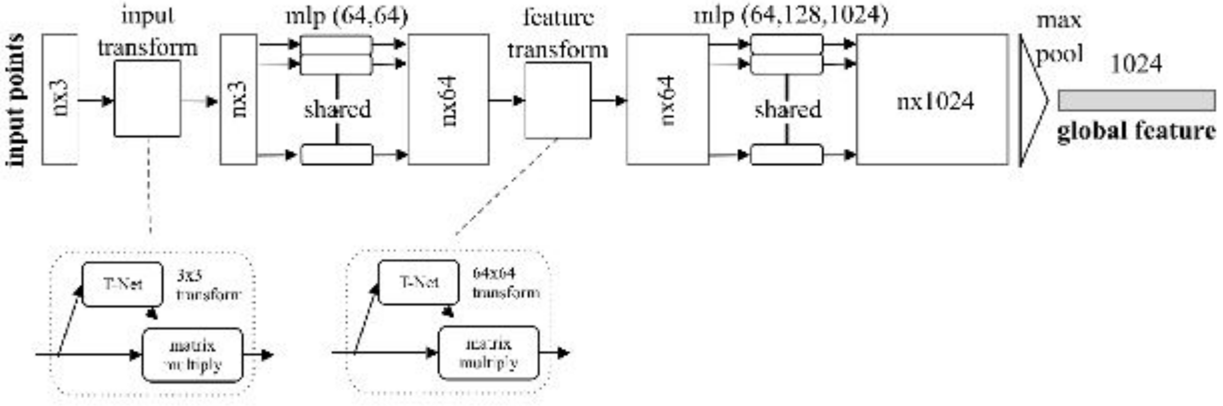
PointNet: architecture

Composition of T-Nets



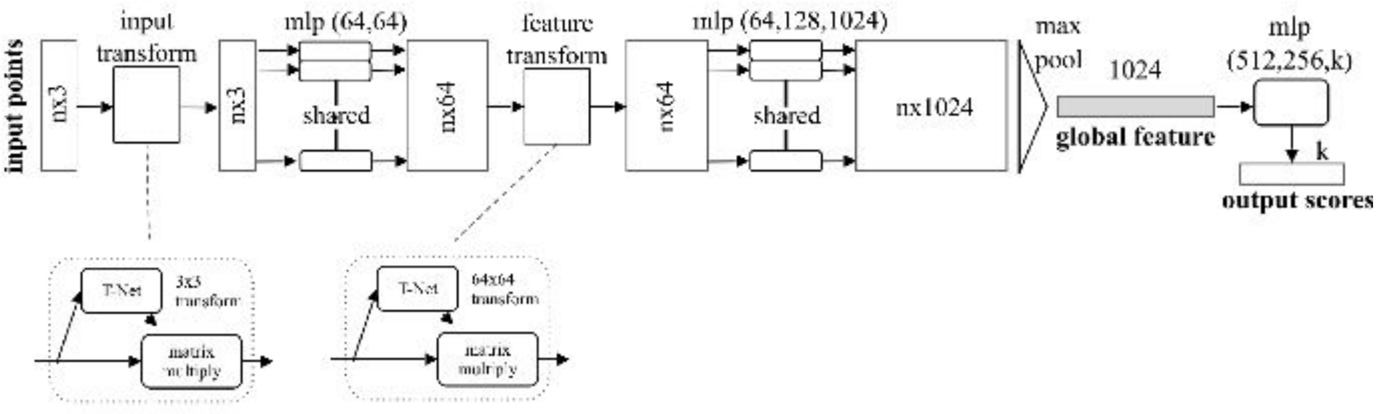
PointNet: architecture

Composition of T-Nets



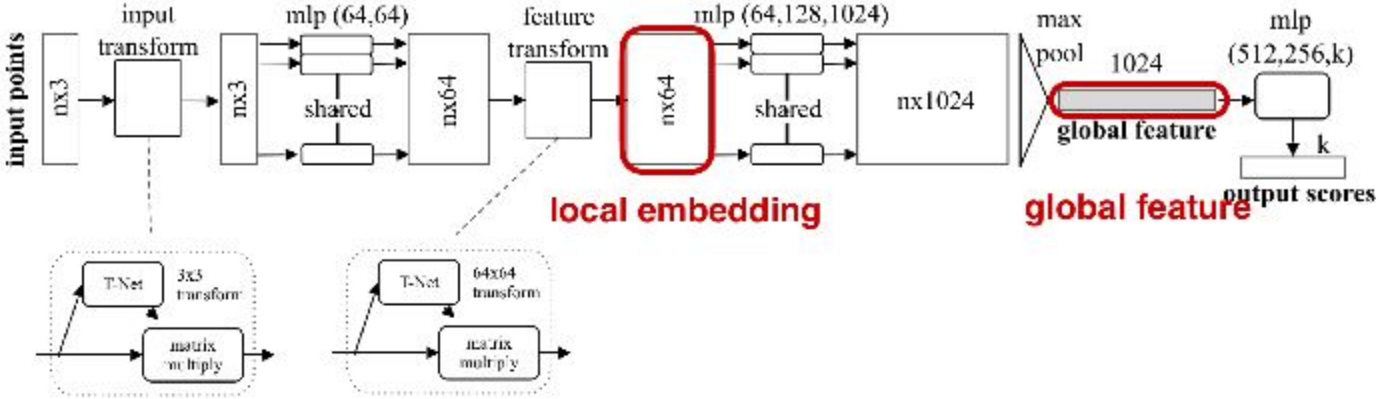
PointNet: architecture

Composition of T-Nets



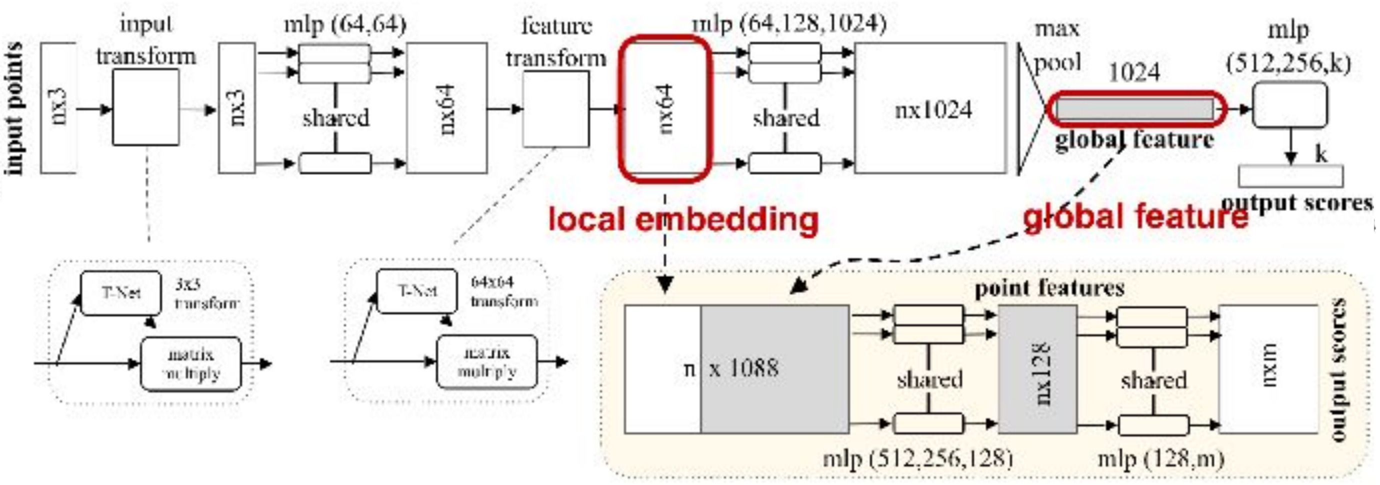
PointNet: architecture

Composition of T-Nets



PointNet: architecture

Composition of T-Nets



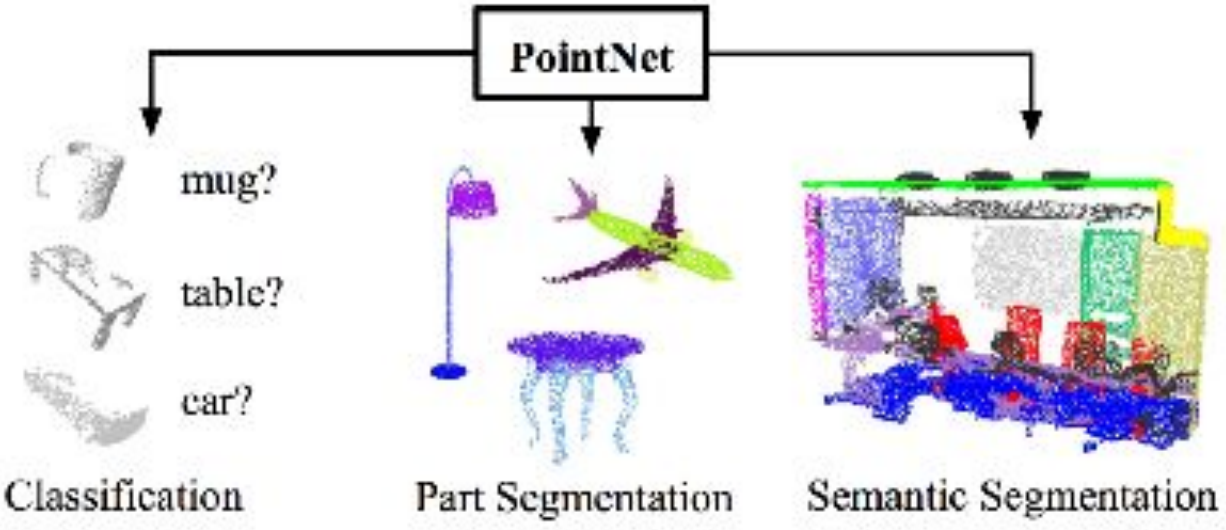
PointNet: results

3D CNNs

	input	#views	accuracy avg. class	accuracy overall
SPH [12]	mesh	-	68.2	
3DShapeNets [29]	volume	1	77.3	84.7
VoxNet [18]	volume	12	83.0	85.9
Subvolume [19]	volume	20	86.0	**89.2**
LFD [29]	image	10	75.5	-
MVCNN [24]	image	80	**90.1**	-
Ours baseline	point	-	72.6	77.4
Ours PointNet	point	1	86.2	**89.2**

dataset: ModelNet40; metric: 40-class classification accuracy (%)

PointNet: results



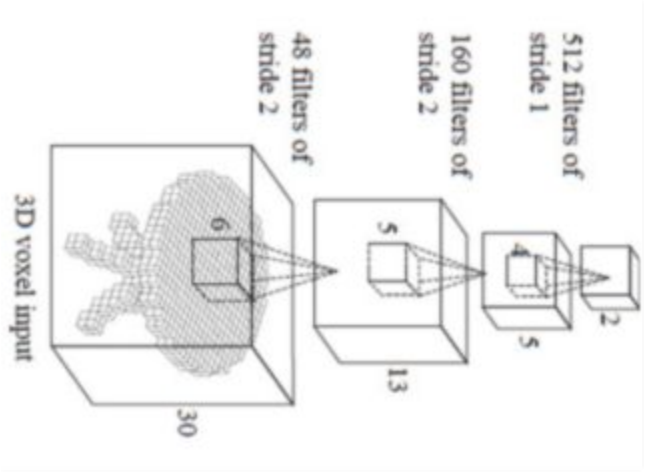
PointNet: results

Scene parsing

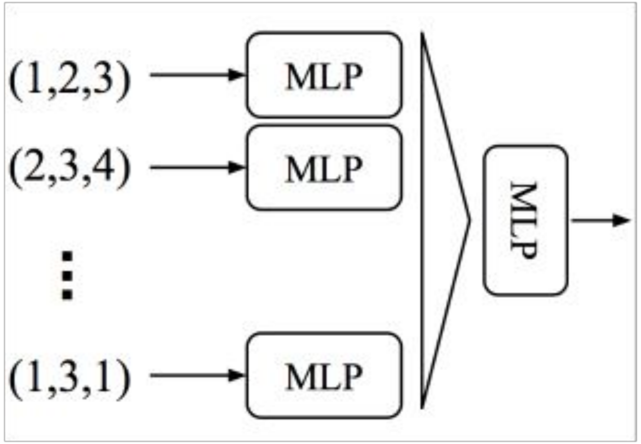
dataset: Stanford 2D-3D-S (Matterport scans)

Limitations of Pointnet

- No local context for each point!
- Global feature depends on absolute coordinate.
- Hard to generalize to unseen scene configurations!



3D CNN (Wu et al.)

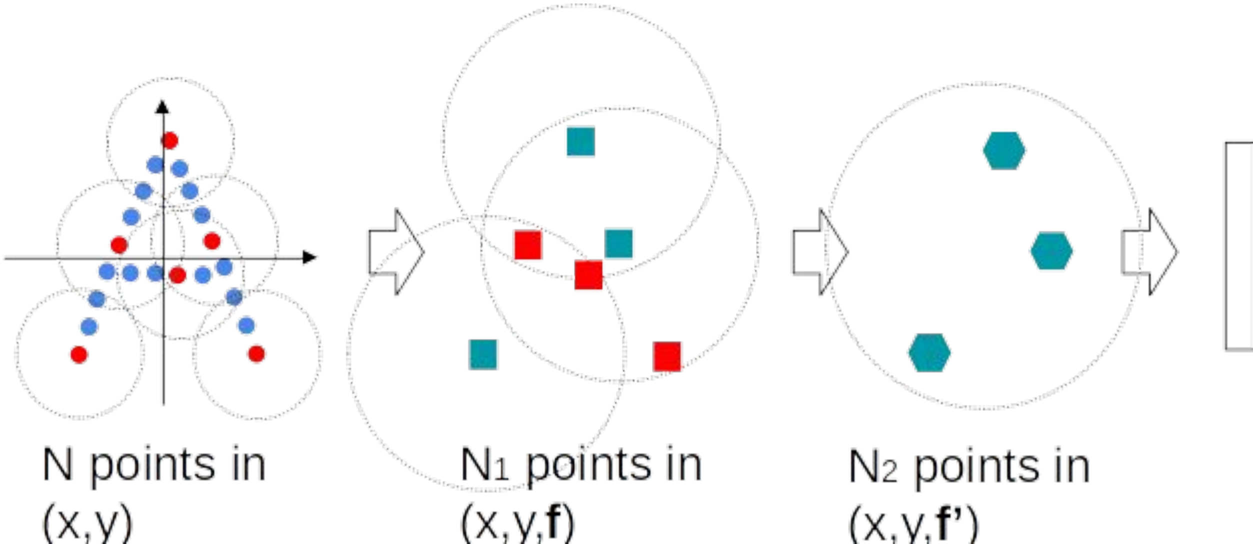


PointNet (vanilla) (Qi et al.)

PointNet v2.0: Multi-Scale PointNet

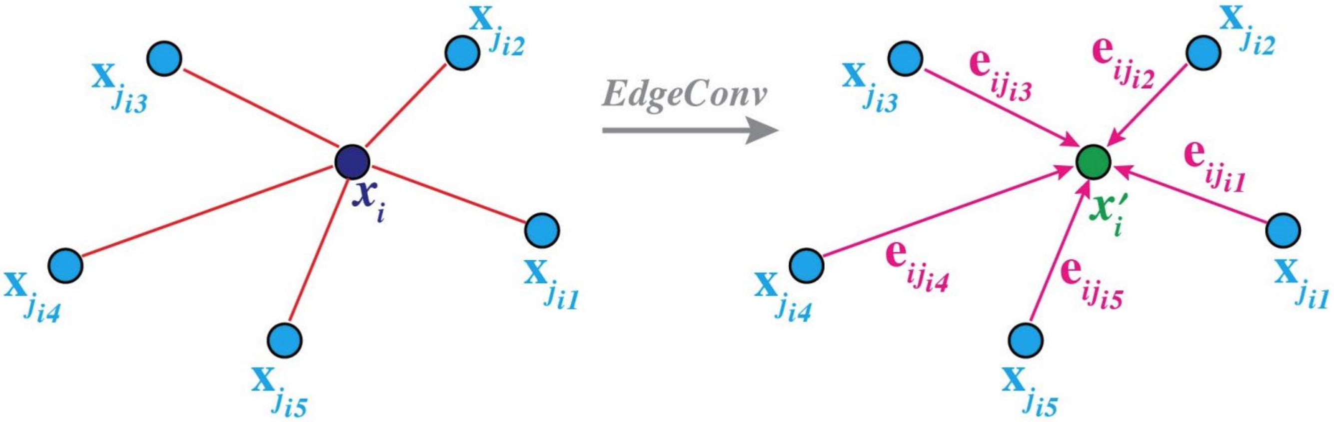
Repeat

- Sample anchor points
- Find neighborhood of anchor points
- Apply PointNet in each neighborhood to mimic convolution



Point Convolution As **Graph Convolution**

- Points -> Nodes
- Neighborhood -> Edges
- Graph CNN for point cloud processing



Wang et al., “Dynamic Graph CNN for Learning on Point Clouds”, Transactions on Graphics, 2019

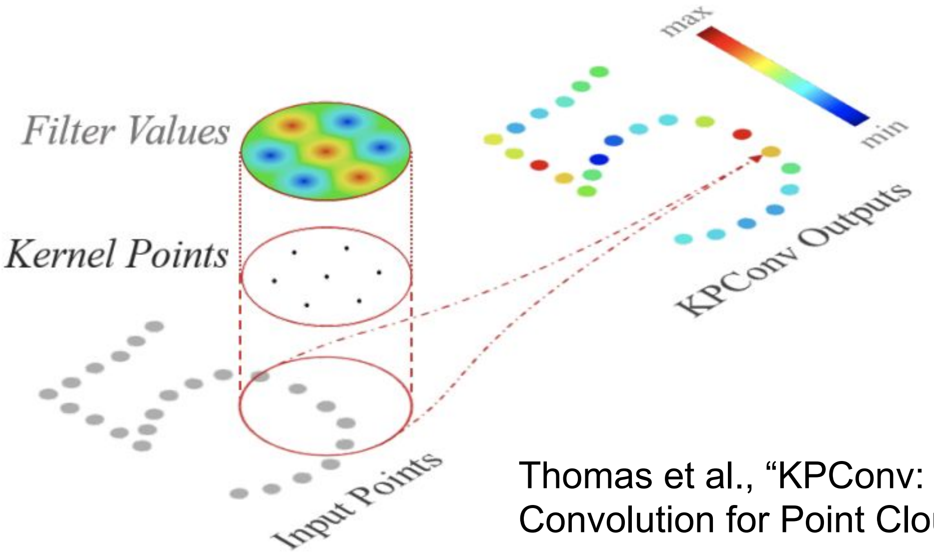
CNN are not aware of geometry

- Points sampled from surfaces
- Lack of sample invariance addressing (e.g. Lidar data)

Solution: Estimate the continuous kernel and point density for continuous convolution

Mathematically Proper Conv. Discretization

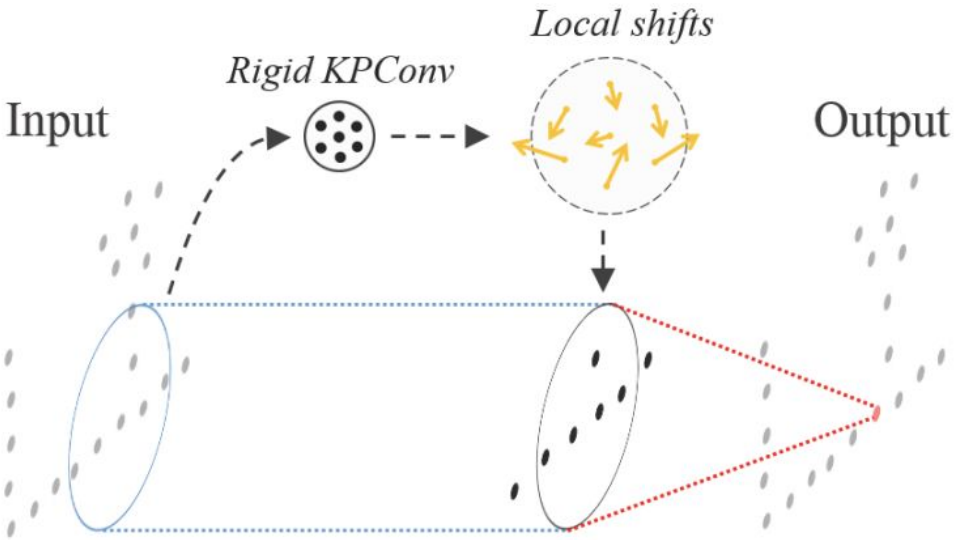
- Continuous conv: $$(\mathcal{F} * g)(x) = \int g(y-x) f(y) dy$$
- Empirical conv: $$(\mathcal{F} * g)(x) = \sum_{x_i \in \mathcal{N}_x} g(x_i - x) f_i$$



Thomas et al., “KPConv: Flexible and Deformable Convolution for Point Clouds”, ICCV 2019

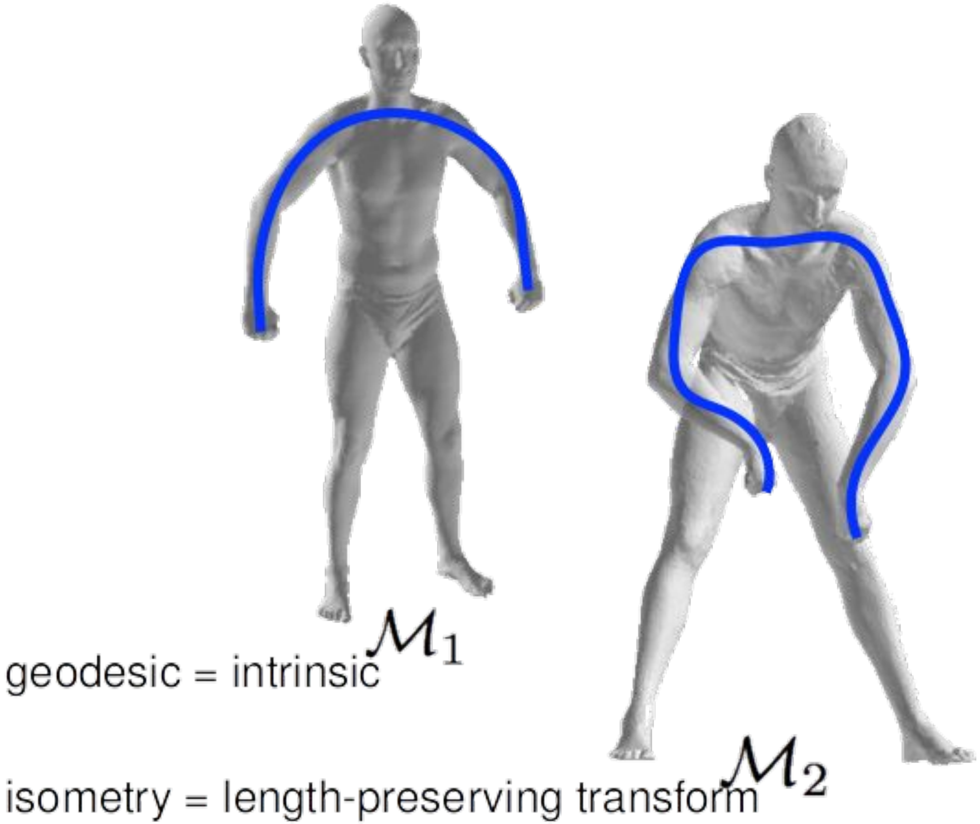
Deformable Kernel for Deformable Objects

- Deformable point-based kernel
- The 3D version of 2D deformable convolution



Thomas et al., “KPConv: Flexible and Deformable Convolution for Point Clouds”, ICCV 2019

Classification: **Spectral CNN**



Classification: **Spectral CNN**

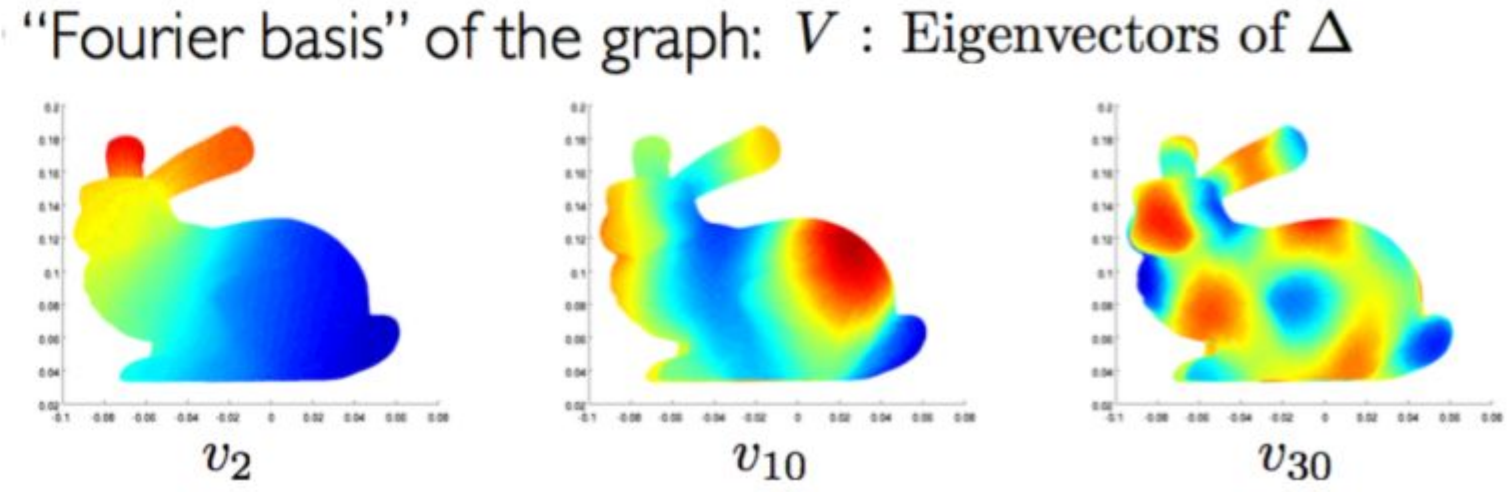
- Convolution done in the spectral domain
- Kernels are also built in spectral domain
- Activation done in the spatial domain

Needs to be a differentiable manifold!

Masci et al., "Geometric deep learning on graphs and manifolds using mixture model CNNs", CVPR 2017

Spectral CNN: obtain fourier basis

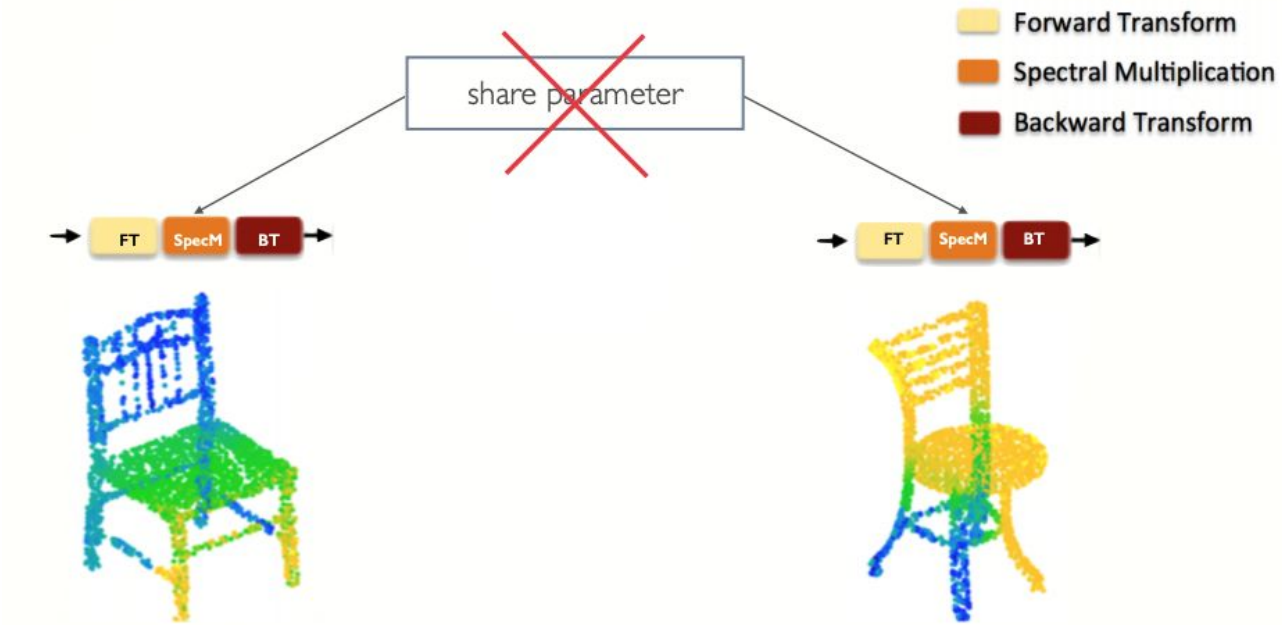
Derived by eigenfunctions of self-adjoint operators, e.g. Laplacian-Beltrami



Masci et al., "Geometric deep learning on graphs and manifolds using mixture model CNNs", CVPR 2017

Fundamental Challenge of Spectral CNN

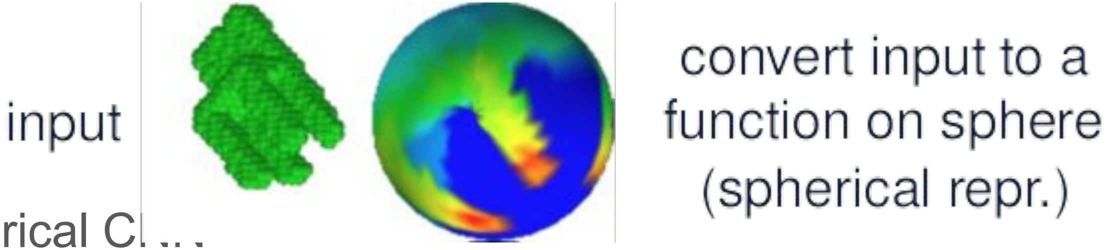
- If the shapes to compare are not isometric, their spectral domains are not aligned
- Function bases are derived by Laplacian operator, which is geometry dependent



Yi et al., “SyncSpecCNN: Synchronized Spectral CNN for 3D Shape Segmentation”, CVPR 2017

A Special Case: Spherical CNN

- If the surface is always a SPHERE, no worry about the functional space alignment anymore
- Generate a spherical representation



- Do Spherical C
    - Has numerical tricks exploiting the symmetry of sphere

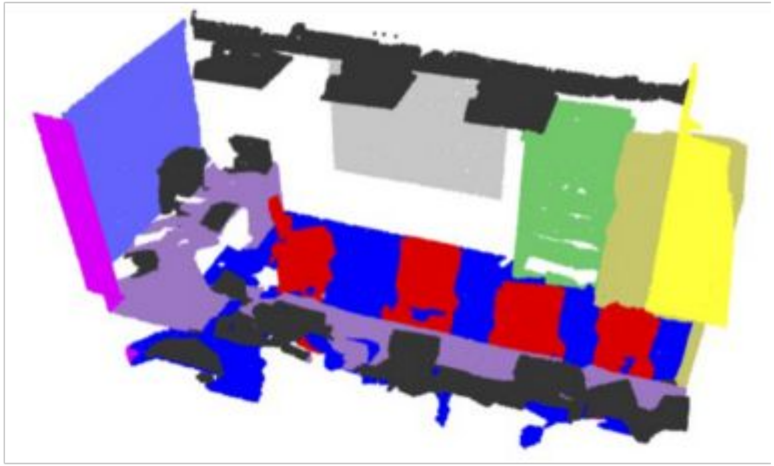
Cohen et al., “Spherical CNN”, ICLR 2018

Content

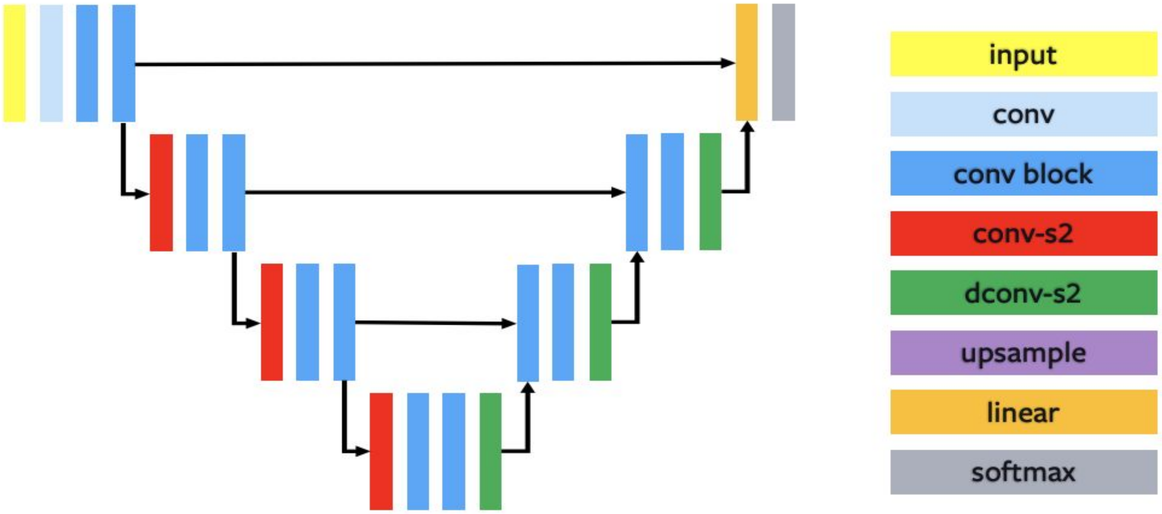
1. Preliminary ideas on DL
2. Why 3D data with deep learning?
3. Classification
4. **Segmentation**
5. Perspectives

Segmentation Detection

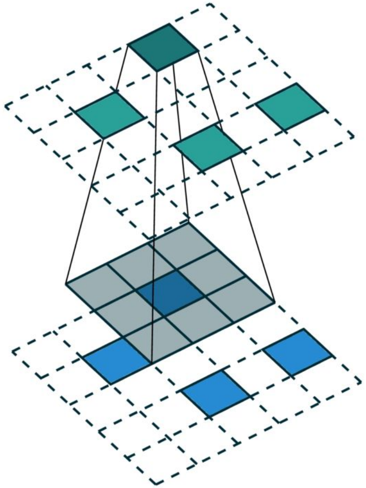
Semantic segmentation



Encoder-Decoder: sparse conv



(b) Submanifold sparse U-Net.



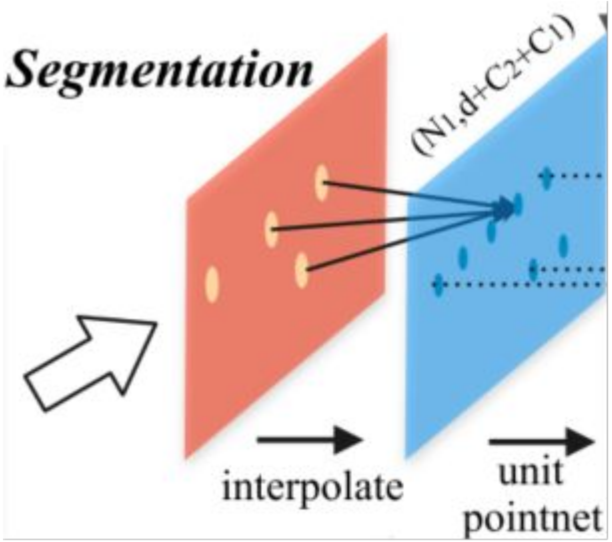
Graham, Benjamin, Martin Engelcke, and Laurens van der Maaten. "3d semantic segmentation with submanifold sparse convolutional networks." CVPR 2018.

Choy, Christopher, et al. “4D Spatio-Temporal Convnets: Minkowski Convolutional Neural Networks.” CVPR 2019

Encoder-Decoder: upsampled pointnet

Upsampled pcd features interpolating

from 3 nn poins



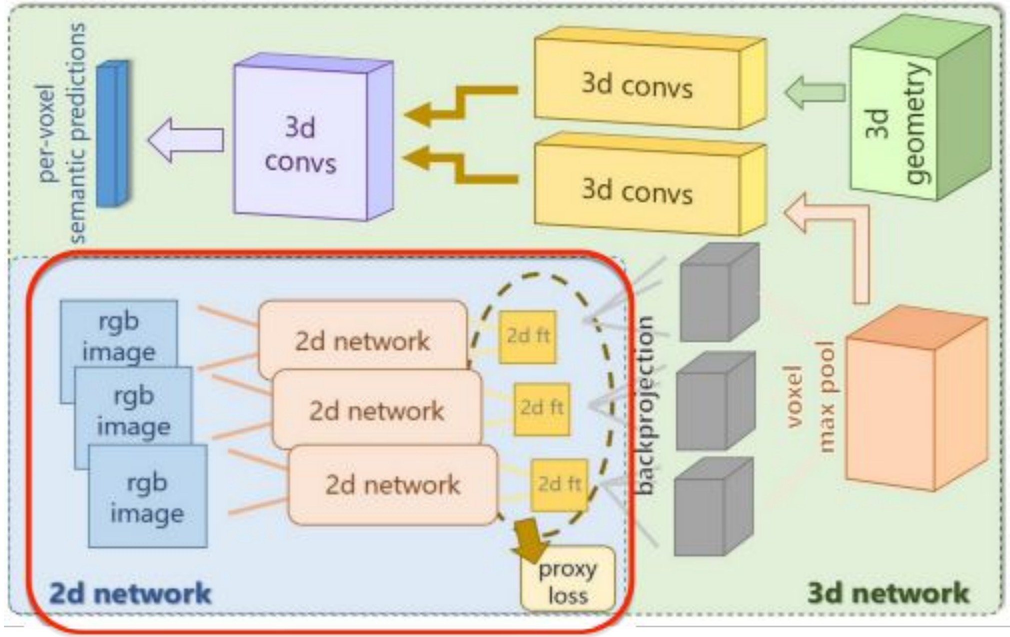
Lower resolution
Fewer points

Higher resolution
More points

Qi, Charles R., et al. “PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space.”, NeurIPS 2017

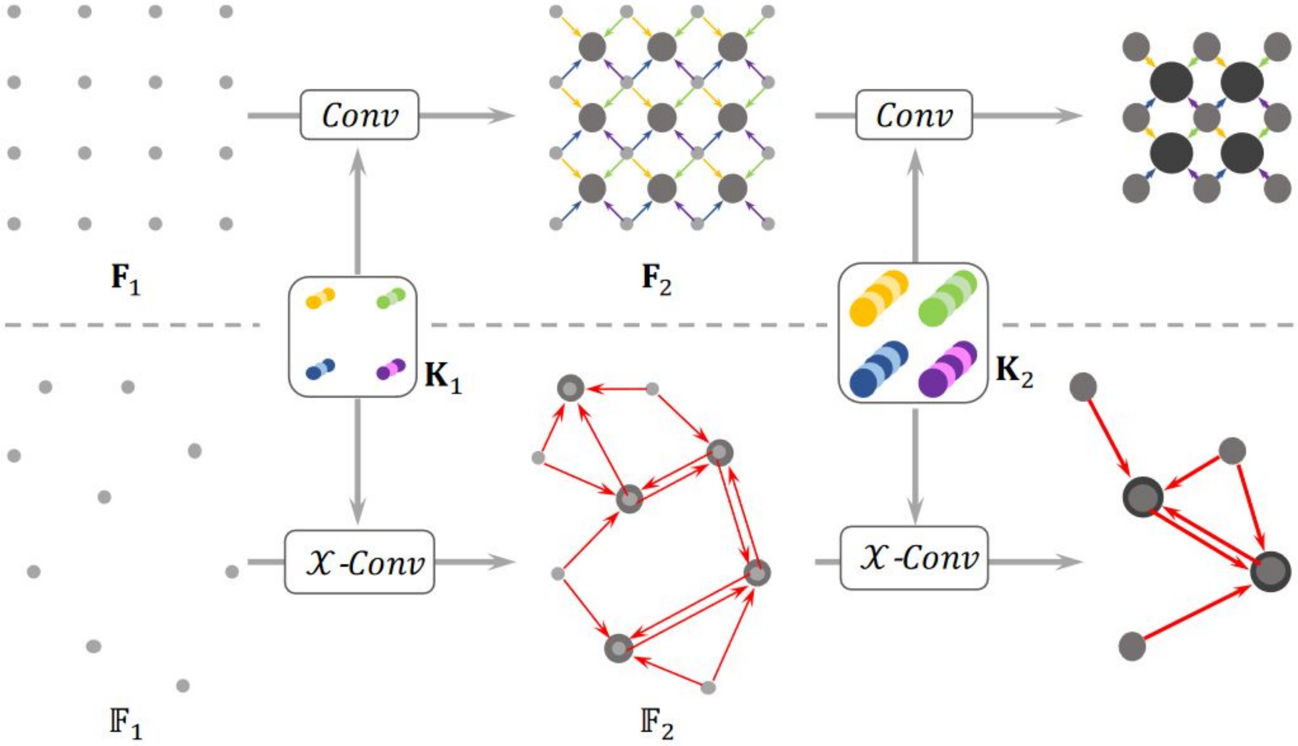
Multimodal approach

- Backproject 2D features to 3D voxels
- Apply voxel-wise max-pooling across multiple views
- Fuse 2D and 3D features at the intermediate level

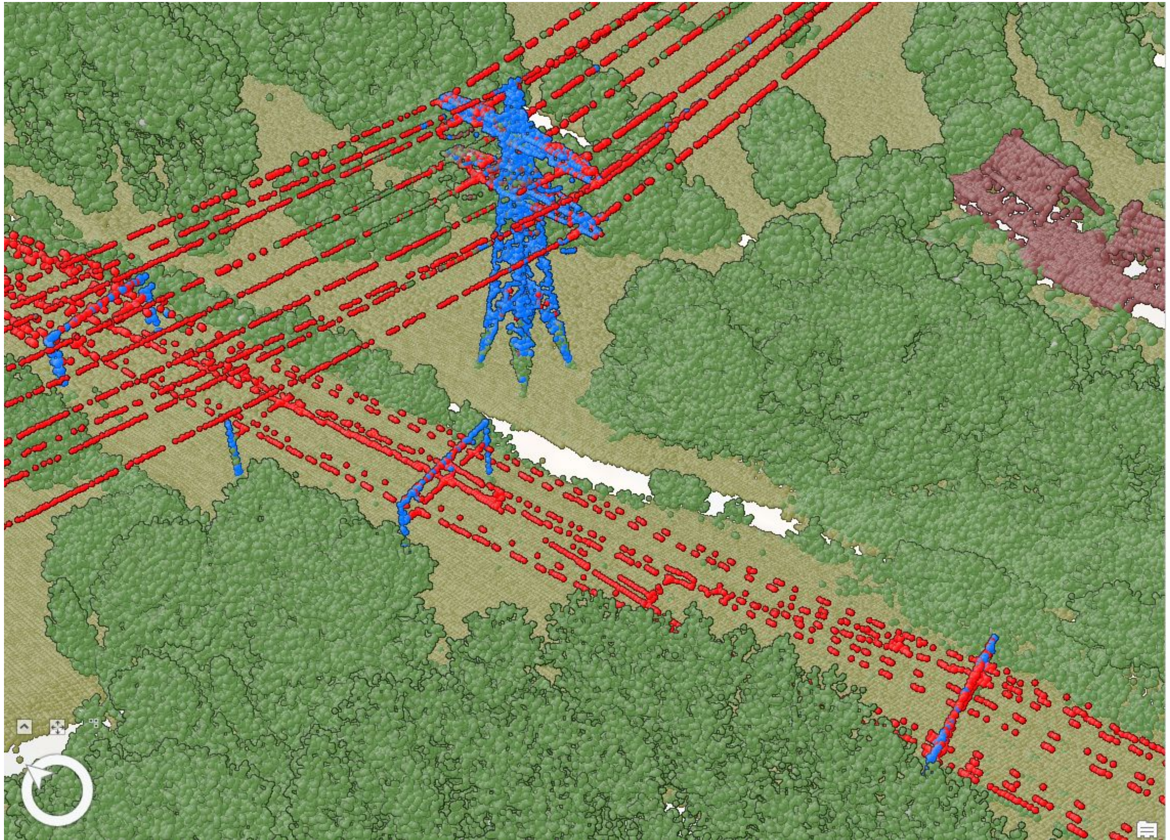


Dai, Angela, and Matthias Nießner. "3dmv: Joint 3d-multi-view prediction for 3d semantic scene segmentation.", ECCV 2018

Segmentation with **X-conv**



Segmentation with **X-conv**

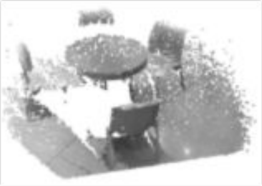


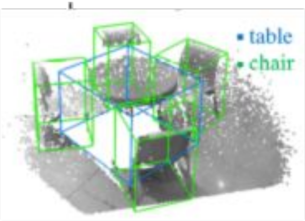
Segmentation: Instance-level Understanding

Input

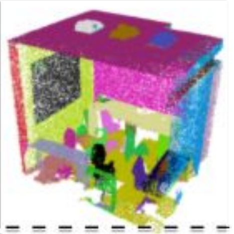
Outpu

Object Detection





Object Segmentation



Part Segmentation

Task: 3D Detection & Instance Segmentation

- General
  - Sliding shape
  - 3D-SIS
  - Frustum PointNet
  - Point R-CNN
  - VoteNet
  - GSPN
  - SGPN
  - JSIS3D
- BEV
  - ContFuse
  - PointPillar
  - Second

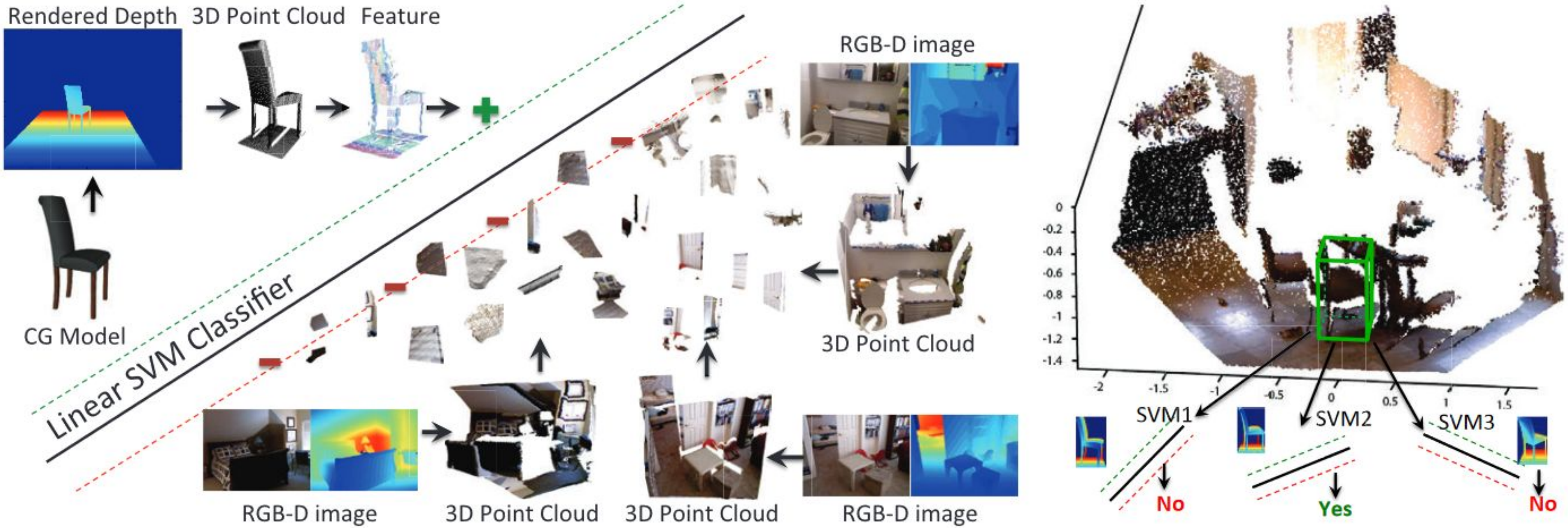
T

op-bottom approach
What object is in the pcd?

Bottom-up approach
To wich object belongs this point?

Segmentation: top-down approaches

Sliding Shapes



Song et al., “Sliding Shapes for 3D Object Detection in Depth Images”, ECCV 2014

Segmentation: top-down approaches

From Box to Instance Segmentation

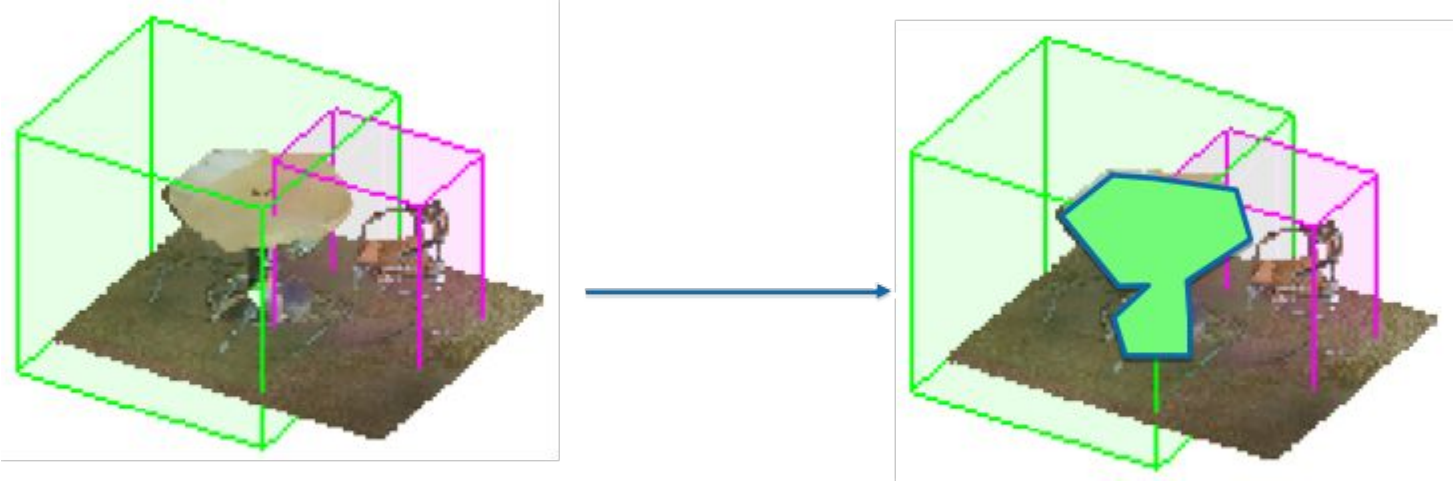


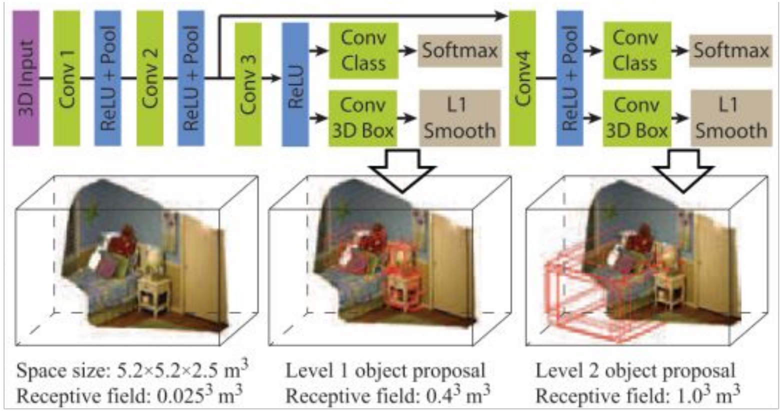
Figure from “Learning Object Bounding Boxes for 3D Instance Segmentation on Point Clouds”, NIPS 2019

Segmentation: top-down approaches

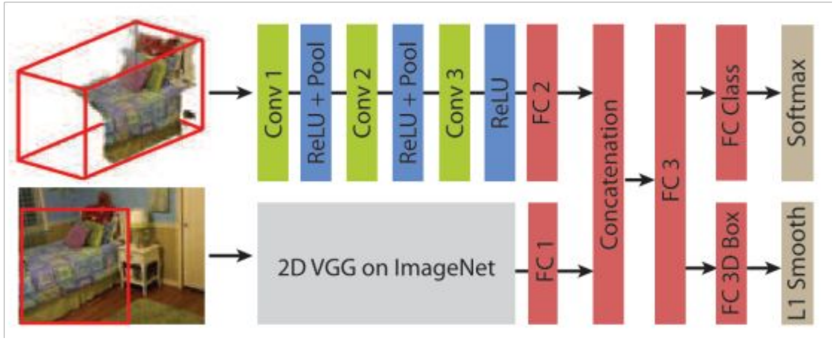
Volumetric R-CNN

Stage1: 3D Region Proposal Network

TSDF



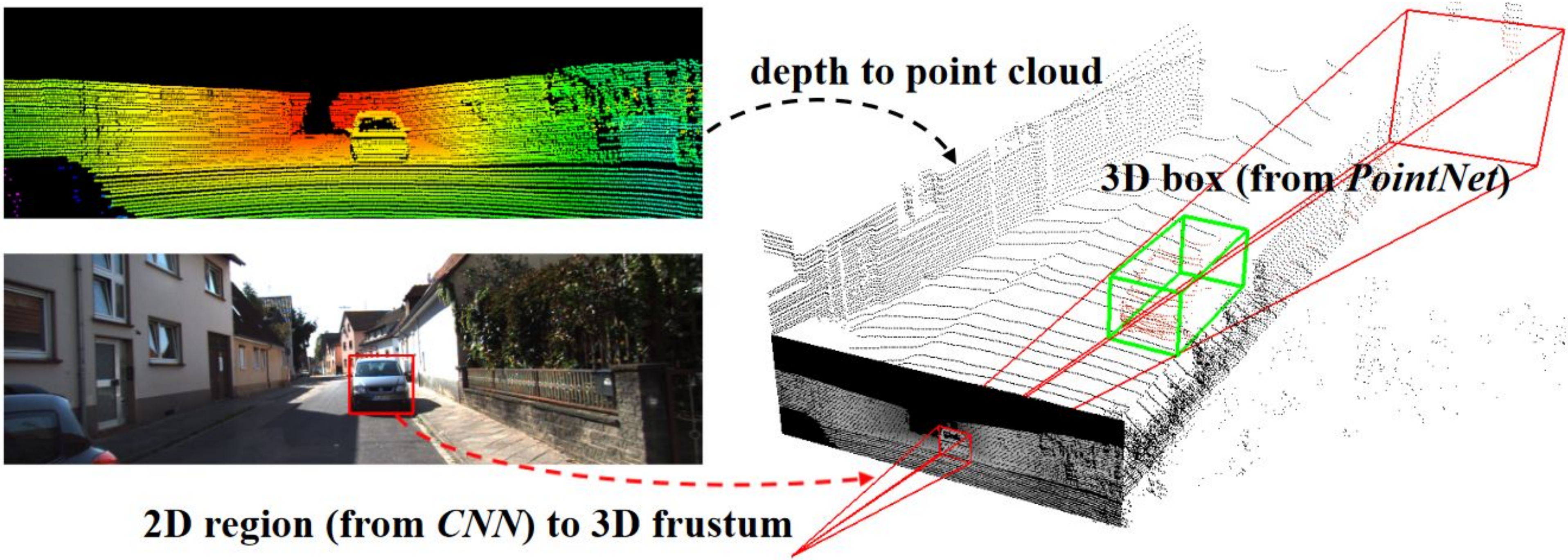
Stage2: Joint Object Recognition Network



Song et al., "Deep Sliding Shapes for Amodal 3D Object Detection in RGB-D Images", CVPR 2016

Segmentation: top-down approaches

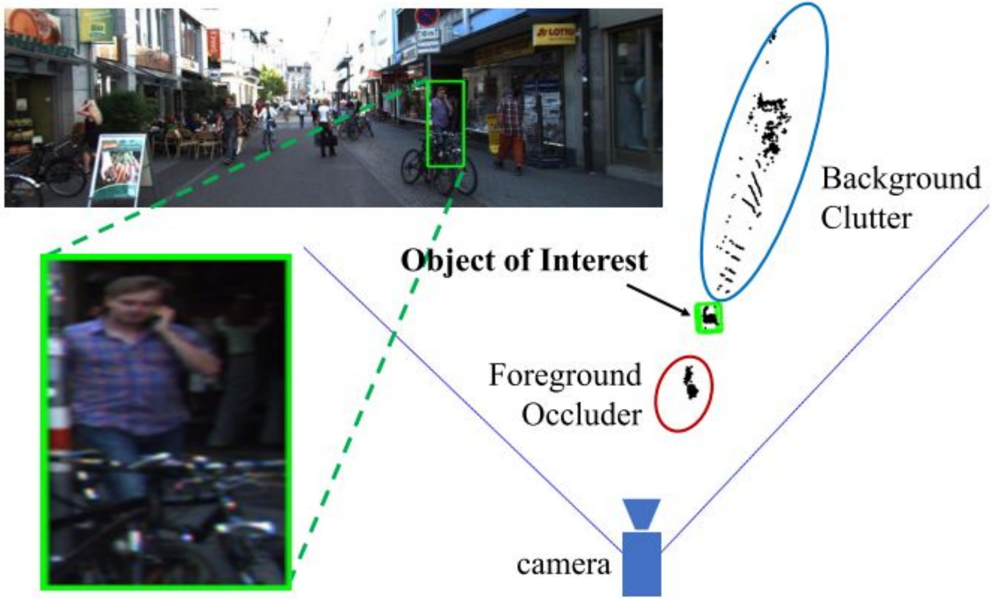
View-based: Generate object proposals from a view (e.g., using SSD)



Qi et al., “Frustum PointNets for 3D Object Detection from RGB-D Data”, CVPR 2018

Segmentation: top-down approaches

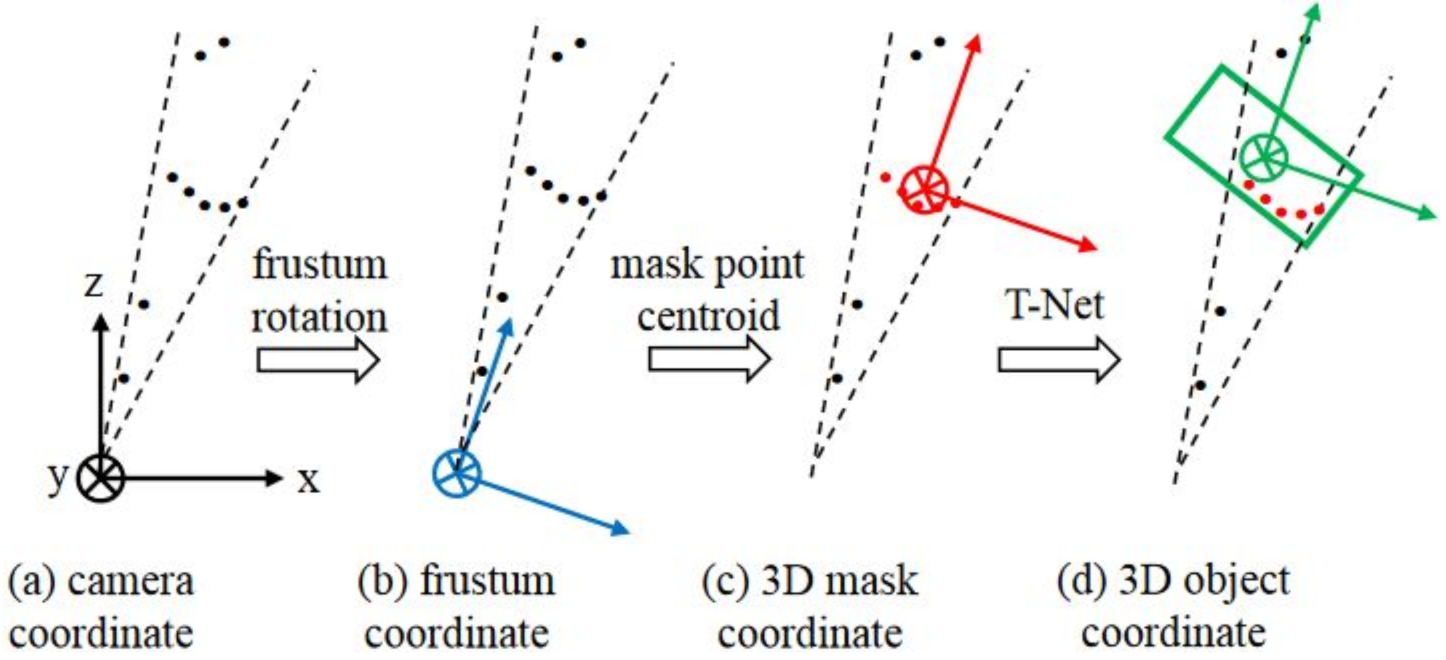
View-based: FG/BG segmentation



Qi et al., "Frustum PointNets for 3D Object Detection from RGB-D Data", CVPR 2018

Segmentation: top-down approaches

View-based: Perspective variation → normalization



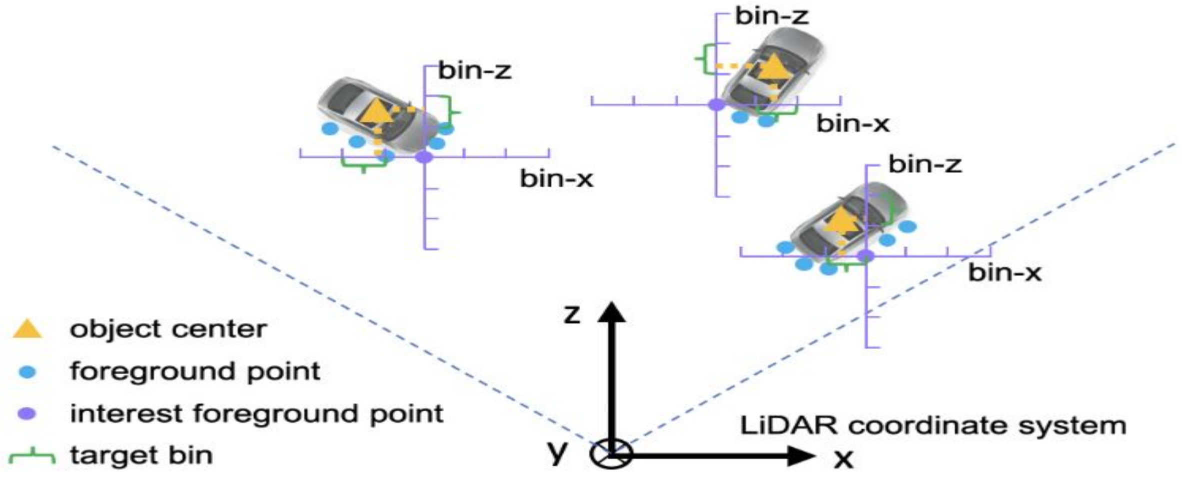
Qi et al., “Frustum PointNets for 3D Object Detection from RGB-D Data”, CVPR 2018

Segmentation: top-down approaches

Point-based Proposal:

Stage 1: FB/BG seg to generate 3D proposal

Stage 2: Refine



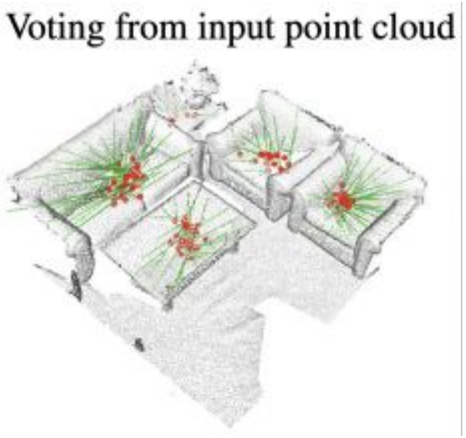
Shi et al., “PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud”, CVPR 2019

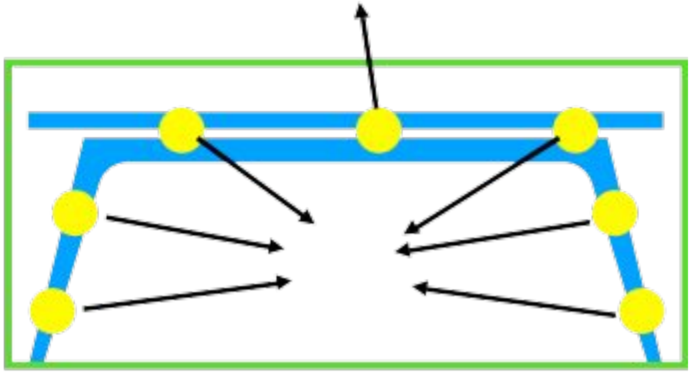
Segmentation: top-down approaches

Proposal from Voting:

How to get 3D object centroid can be far be from any surface point?

Sample a set of seed points and generate votes





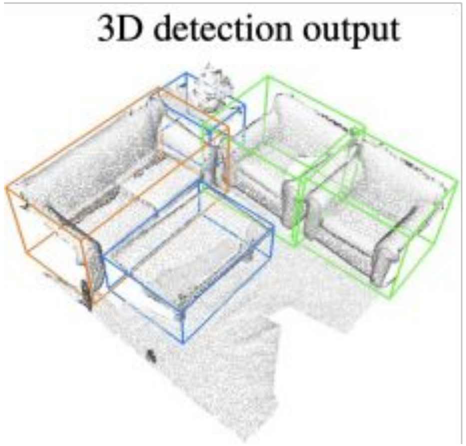
Qi et al., “Deep Hough Voting for 3D Object Detection in Point Clouds”, ICCV 2019

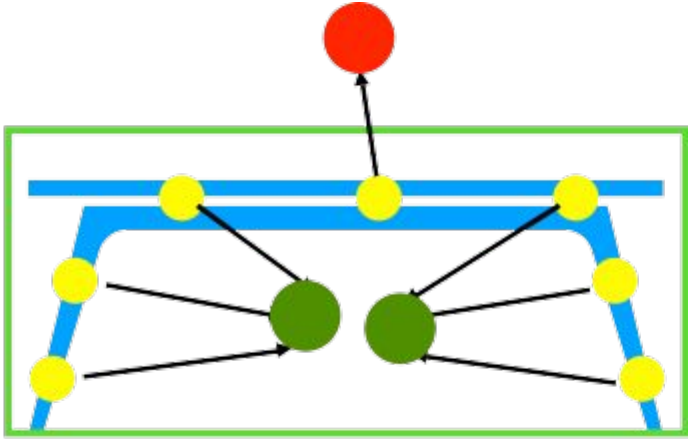
Segmentation: top-down approaches

Proposal from Voting:

How to get 3D object centroid can be far be from any surface point?

Sample a set of seed points and generate votes



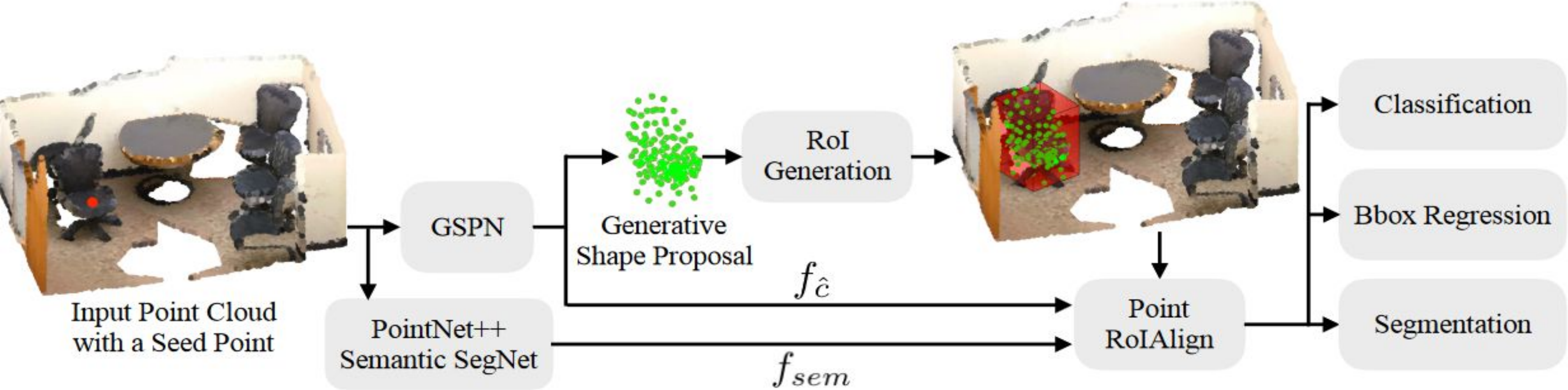


Qi et al., “Deep Hough Voting for 3D Object Detection in Point Clouds”, ICCV 2019

Segmentation: top-down approaches

Proposal from Generative Network:

- Randomly sample seeds points
- Use conditional VAE to generate a point cloud as proposal
- Convert the proposal to an ROI box

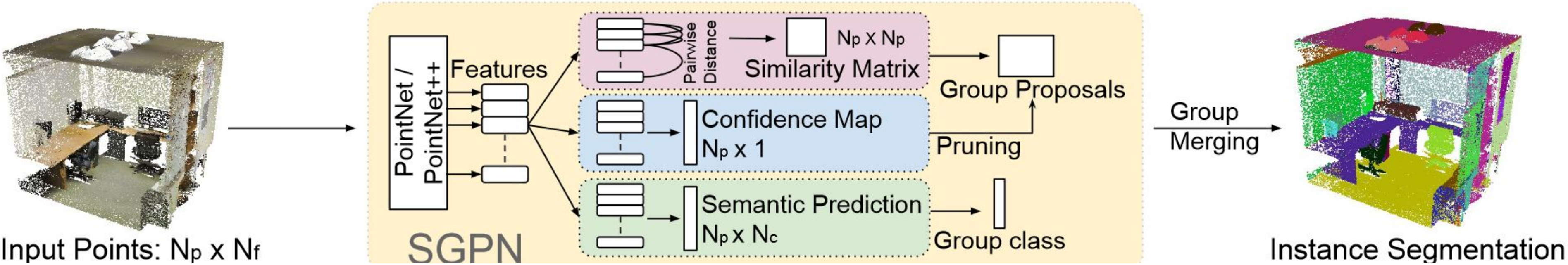


Yi et al., “GSPN: Generative Shape Proposal Network for 3D Instance Segmentation in Point Cloud”, CVPR 2019

Segmentation: bottom-up approaches

Associative Embedding:

- Learn a per-point embedding
- Points from the same instance have similar embeddings
- Clustering gives proposals



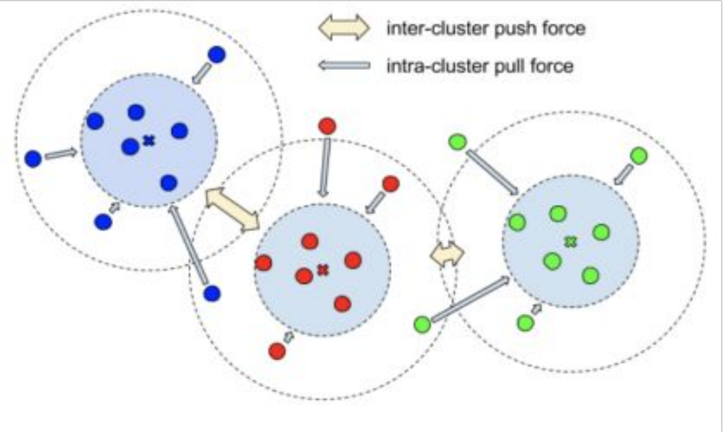
Wang et al., “SGPN: Similarity Group Proposal Network for 3D Point Cloud Instance Segmentation”, CVPR 2018

Segmentation: bottom-up approaches

JSIS3D:

- A discriminative function to present the embedding loss

$$\mathcal{L}_{embedding} = \alpha \cdot \mathcal{L}_{pull} + \beta \cdot \mathcal{L}_{push} + \gamma \cdot \mathcal{L}_{reg}$$

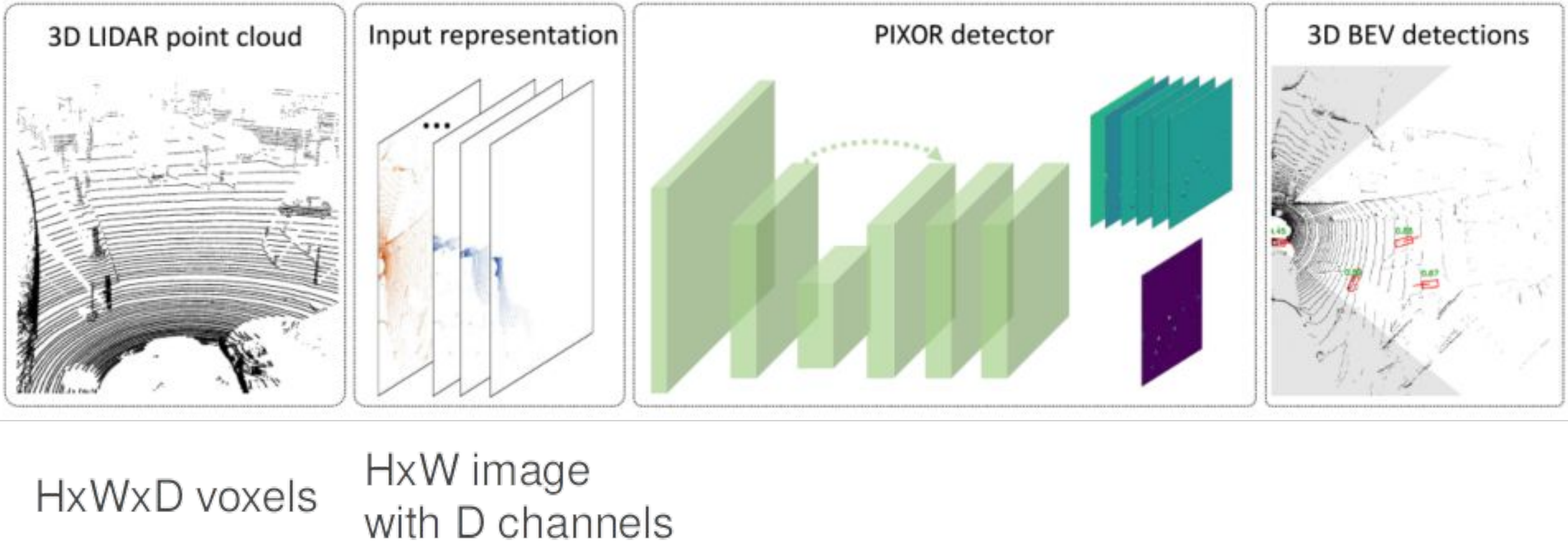


$$\mathcal{L}_{pull} = \frac{1}{K} \sum_{k=1}^{K} \frac{1}{N_k} \sum_{j=1}^{N_k} \left[\|\boldsymbol{\mu}_k - \mathbf{e}_j\|_2 - \delta_v \right]_+^2$$

Pham, Q Hieu, et al. “JSIS3D: Joint Semantic-Instance Segmentation of 3D Point Clouds with CRF.”, CVPR 2019

Segmentation: bottom-up approaches

BEV (bird-eye view):



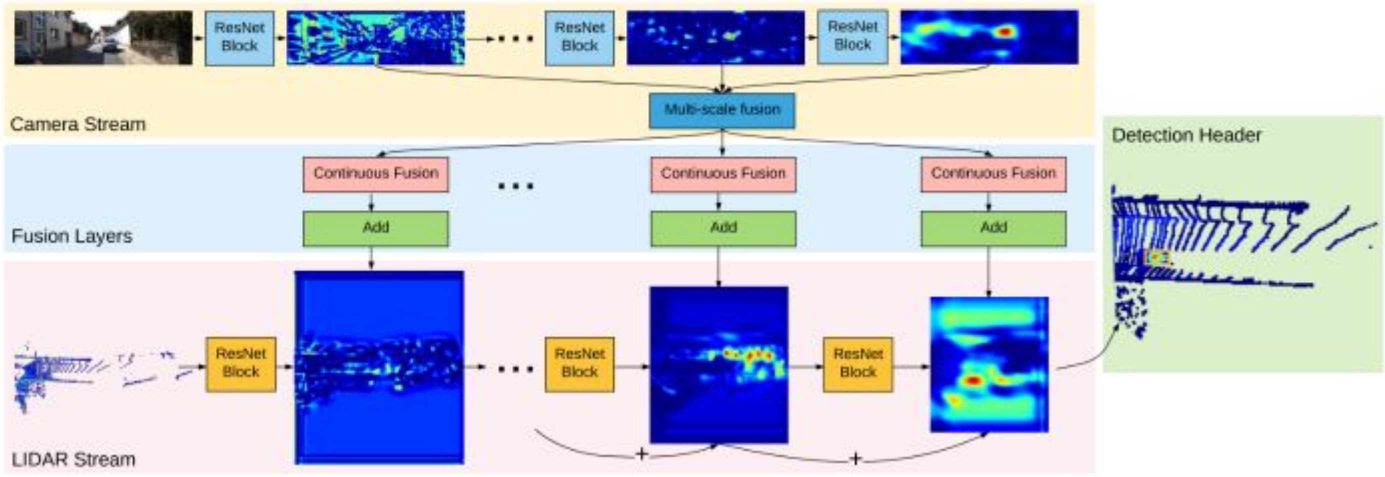
Yang, Bin, Wenjie Luo, and Raquel Urtasun. "Pixor: Real-time 3d object detection from point clouds." CVPR 2018

Segmentation: bottom-up approaches

BEV (bird-eye view): ContFuse

Image feature network: 2D image

BEV network: 3D voxel (HxWxC)

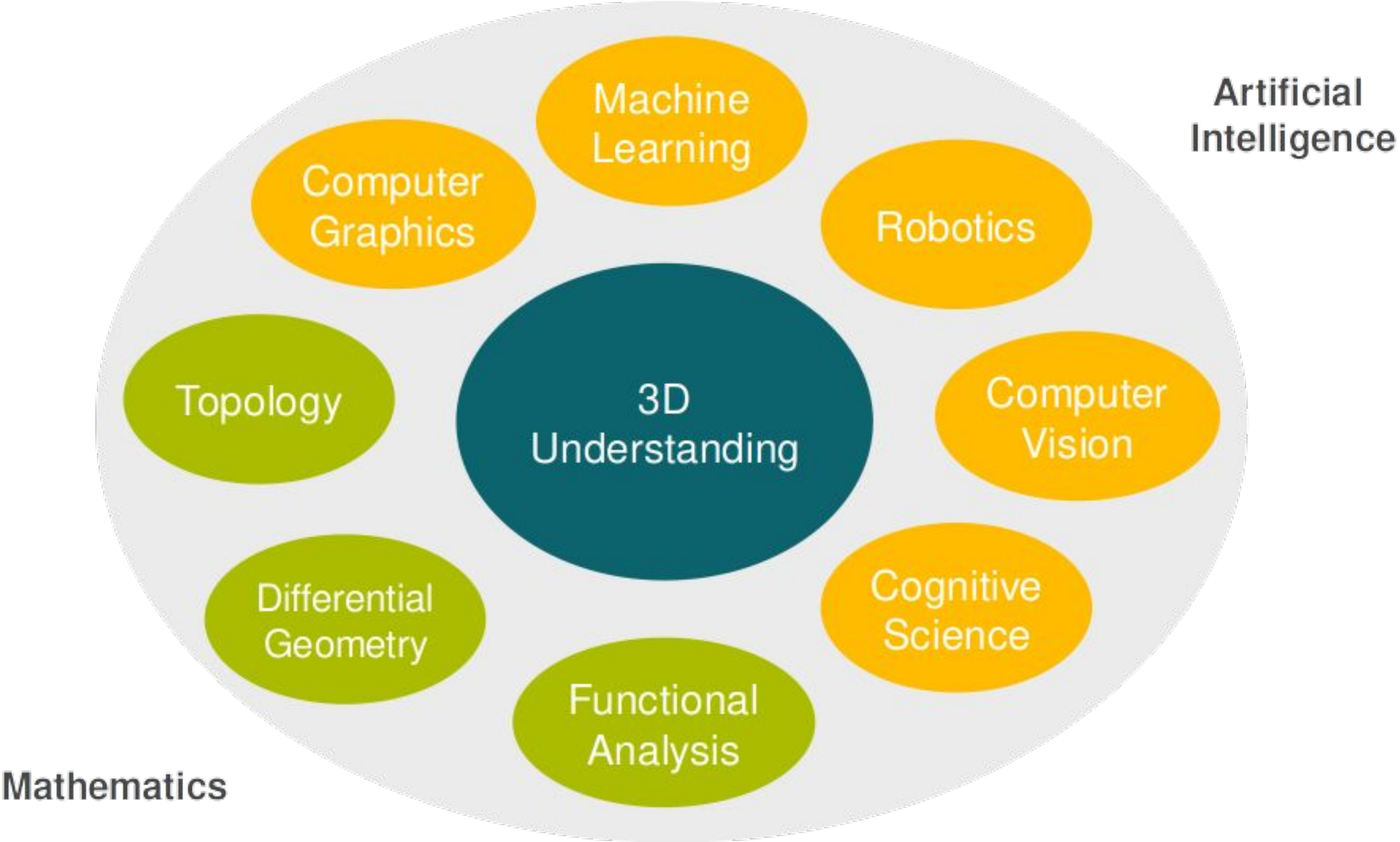


Liang, Ming, et al. "Deep Continuous Fusion for Multi-Sensor 3D Object Detection." ECCV 2018

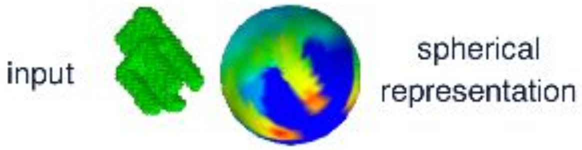
Content

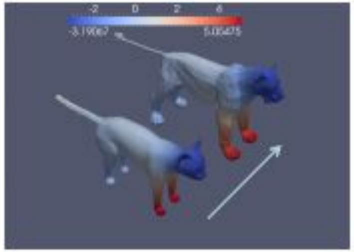
1. Preliminary ideas on DL
2. Why 3D data with deep learning?
3. Classification
4. Segmentation
5. **Perspectives**

Now happening!

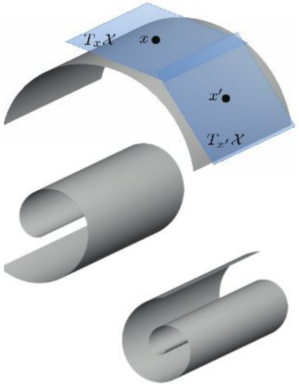


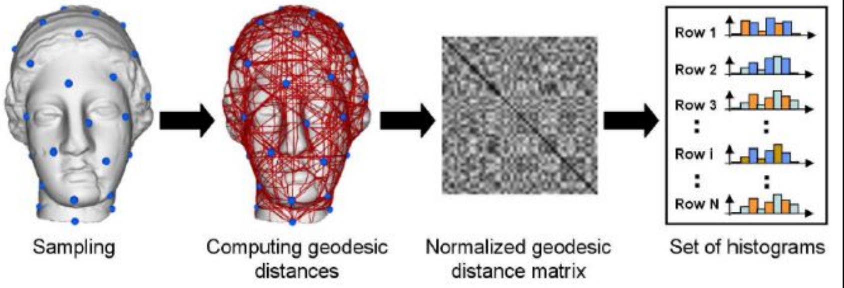
GeometricDL

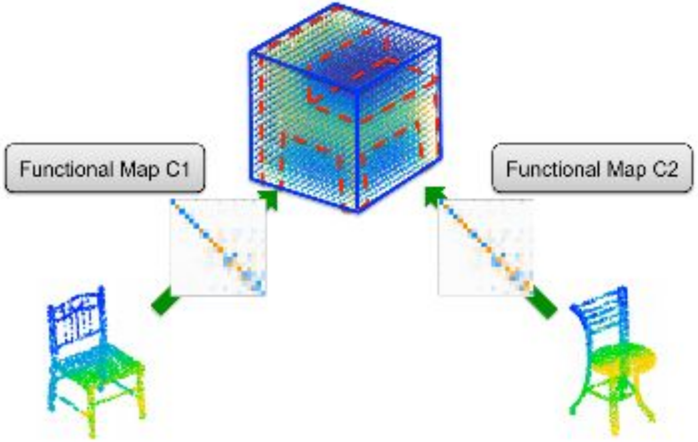


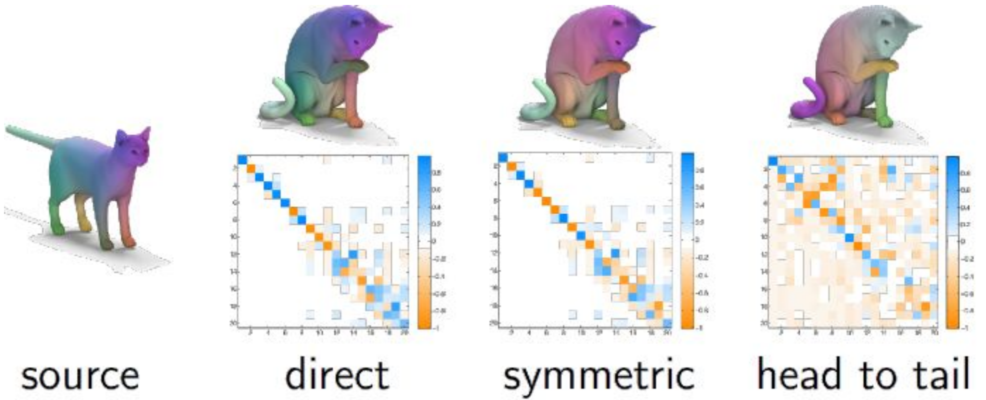


- Intrinsic/geod/R shape feature
- Heat kernel maps
- Laplacian map
- Spectral CNN
- Spectral synch









References

1. http://deeplearning.org
2. https://distill.pub/
3. http://3ddl.stanford.edu/
4. https://github.com/NVIDIAGameWorks/kaolin
5. http://ai.ucsd.edu/~haosu
6. https://geoml.github.io/
7. https://pytorch3d.org/
8. https://github.com/intel-isl/Open3D-ML
9. https://www.shapenet.org/
10. https://modelnet.cs.princeton.edu/

Thank you!

Levente.Tamas@robotics-ai.org