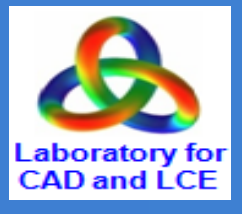
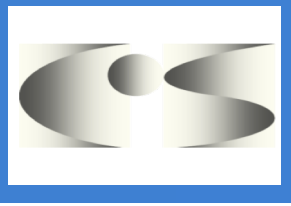




3D Point Cloud Classification, Segmentation, and Normal estimation using Modified Fisher Vector and CNNs

Yizhak (Itzik) Ben-Shabat[1]
Michael Lindenbaum[2]
Anath Fischer[1]

Technion - [1]Mechanical Engineering and [2]Computer Science

Outline

- Point clouds
- Point clouds and CNNs – why the connection is challenging?
- Fisher Vectors
- Representing Point clouds with Fisher vectors
- Deep learning with Fisher vectors input
- Three applications :
  - Classification
  - Semantic segmentation
  - Scale selection & Normal estimation

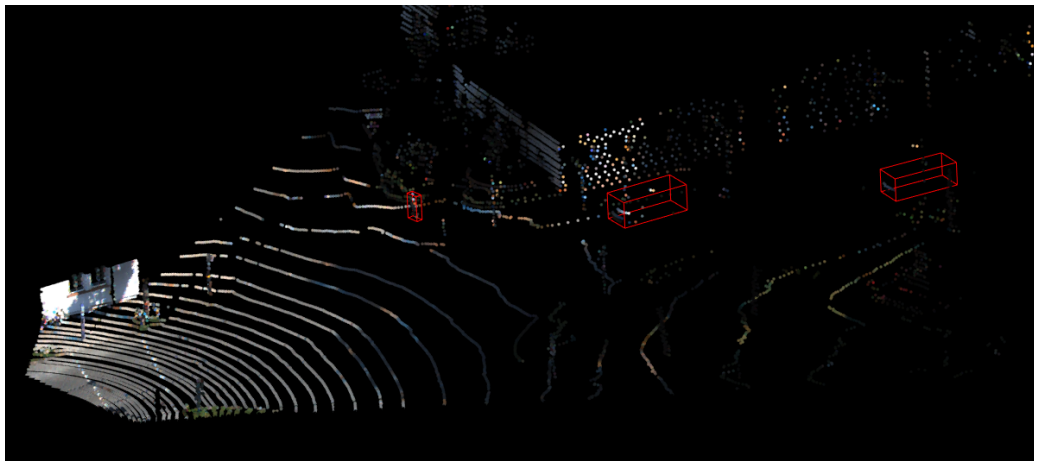
3D data acquisition

Direct 3D sensors are available:

- LiDAR
- RGBD Camera

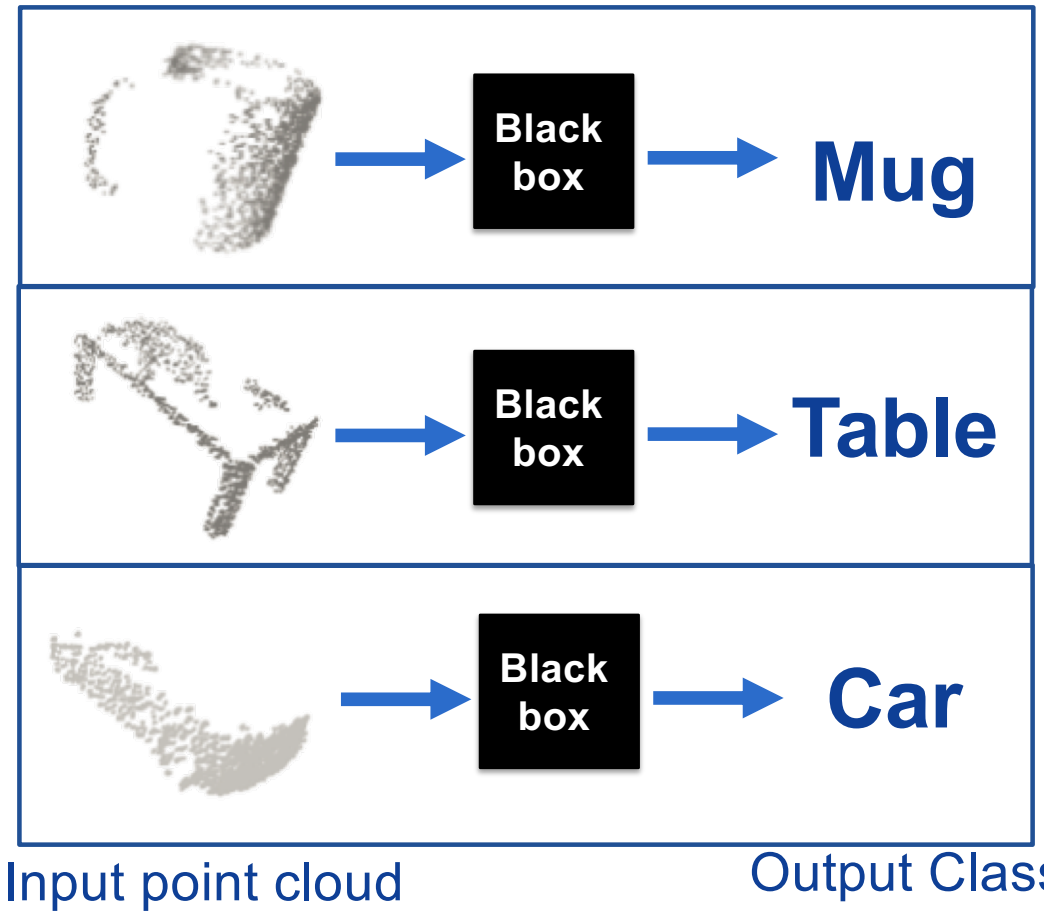
and provide a set of 3D points = point cloud

3D Point Cloud

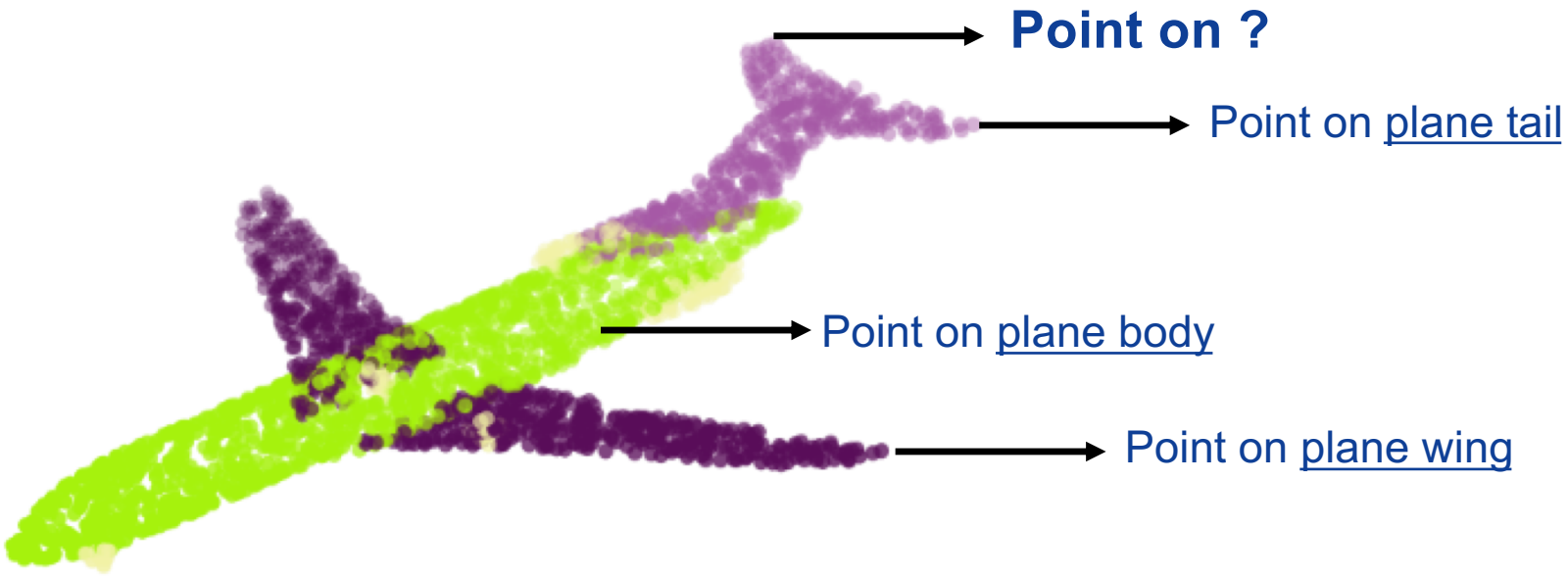


Point clouds from KITTI dataset and NYU Depth V2 dataset

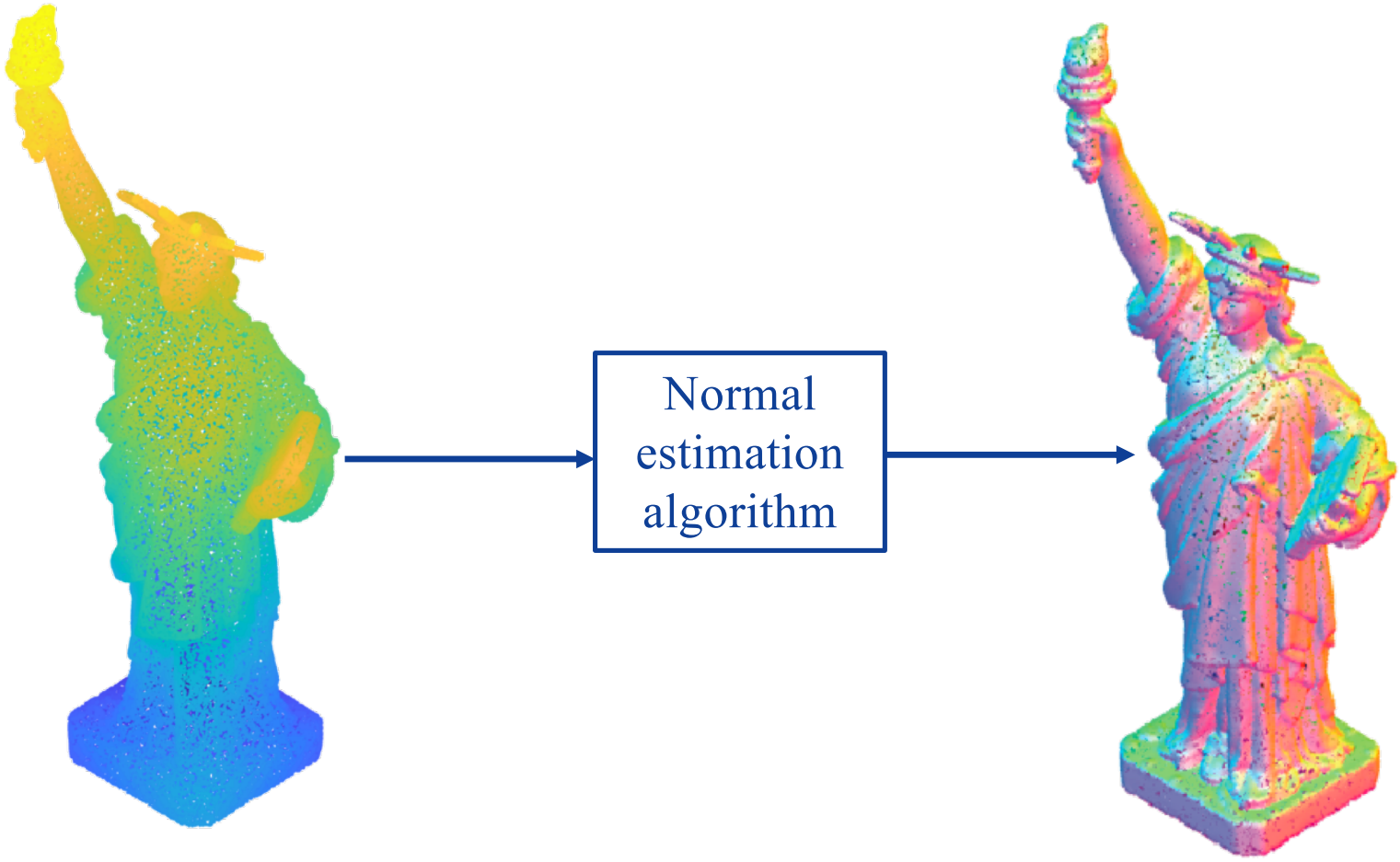
Task 1– Point Cloud Classification



Task 2– Point Cloud Part Segmentation

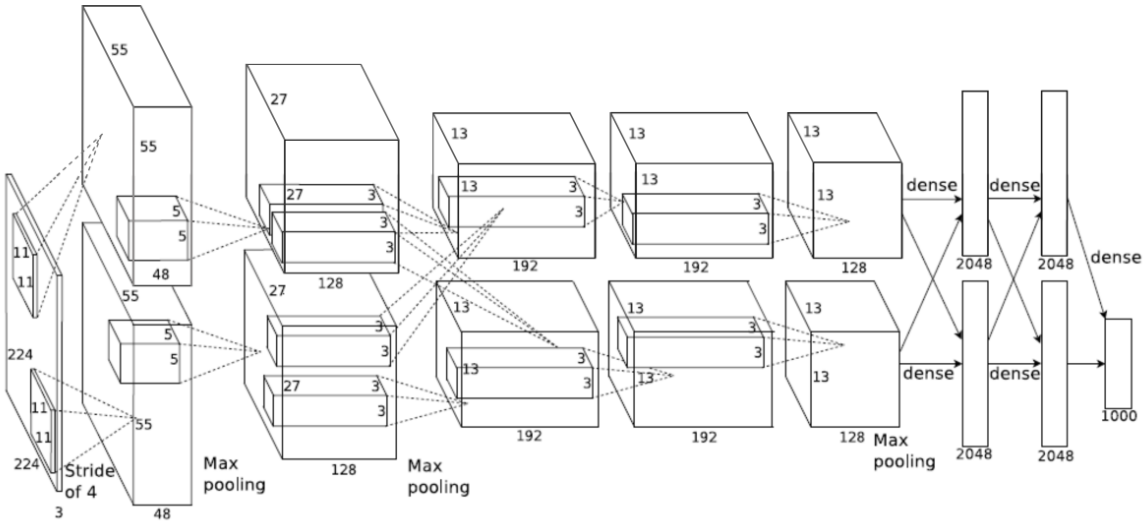


Task 3– Point Cloud Normal Estimation



The preferred tool: Convolutional Neural Networks

- In 2D : Deep CNNs revolutionized image analyzed
  - Convolutional neural nets learn shared weights filters
  - The input (Image) is specified on a grid structure
  - Number of pixels in the input image is fixed



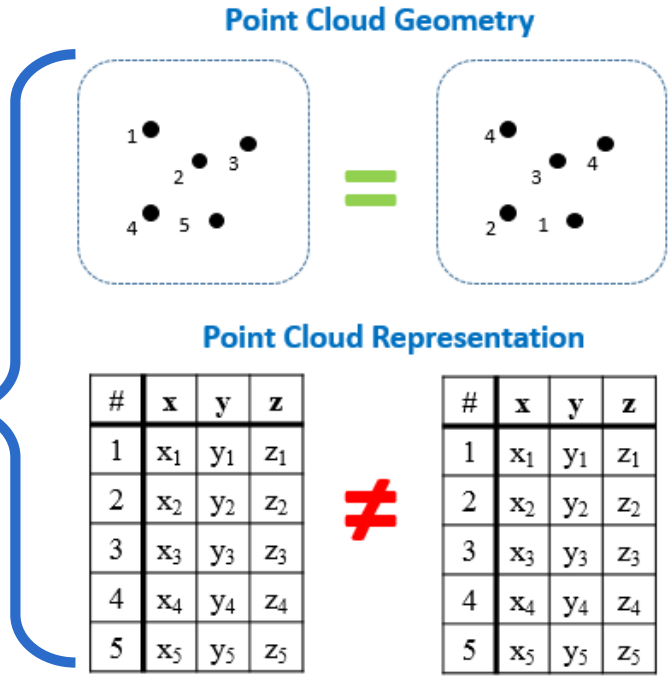
How can we use them for analyzing 3D point cloud?

Challenges

- How can we use the power of CNNs with 3D point cloud data?

- Representing the input is not trivial:
  - A point cloud is not a natural input to a CNN
    - Number of input points is not constant
    - Data is unstructured (no a signal on a grid)
    - Linear ordering cannot reflect spatial proximity
    - No way for unique ordering (permutations)

  - Other challenges with point clouds
    - Missing data, noise, rotations



Voxelization approach

The straightforward approach: transform the point cloud into a voxel grid by rasterizing and use 3D CNNs

A choice between

- Large memory cost and Slow processing time
- OR No Limited spatial resolution Quantization artifacts

A sparsely populated grid which seems un-natural

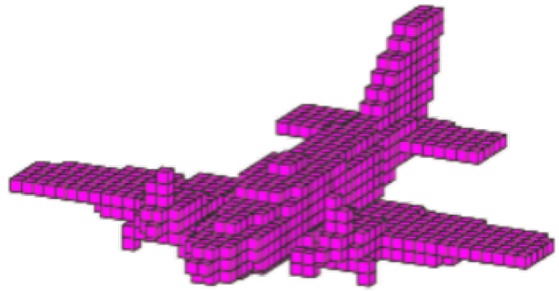


*Image source - AOI-Matlab Voxelizer

Multi-View approach

- The multi-view approach: project multiple views to 2D and use

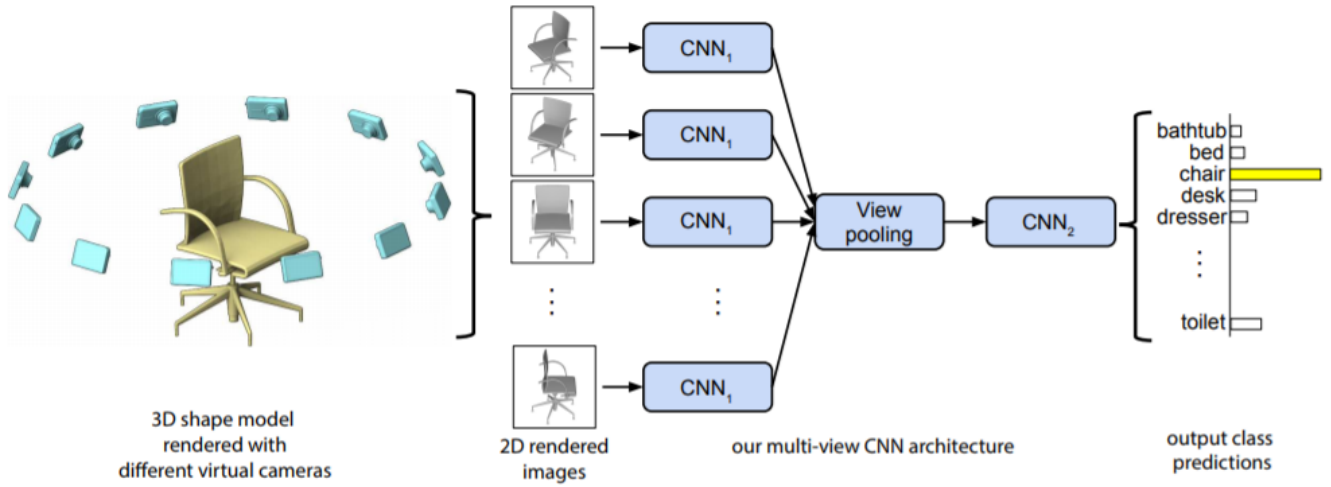
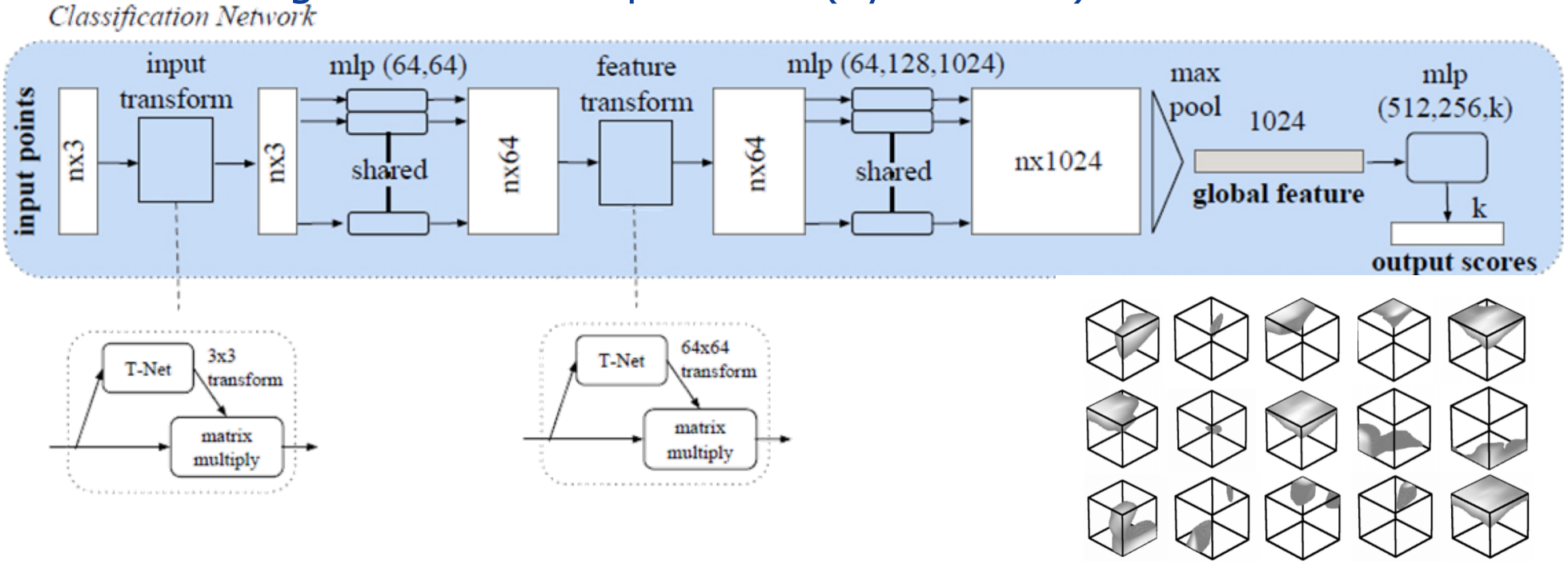


Image taken from H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. Multiviewconvolutional neural networks for 3d shape recognition. In Proceedings of the IEEE International Conference on Computer Vision (CVPR), pages 945–953, 2015.

Direct point cloud approach (PointNet)

Direct approach:

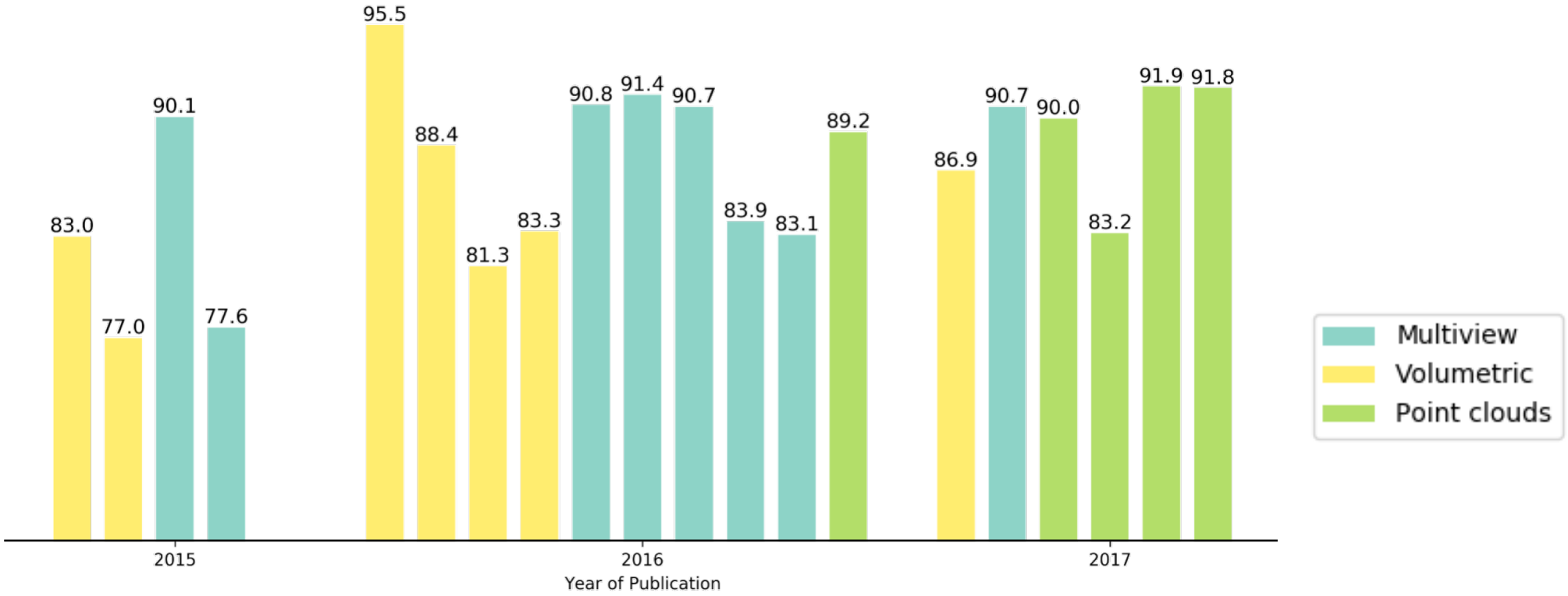
- Process each point separately
- Pool using an order independent (symmetric) function



*Images taken from Qi, Charles R., et al. "Pointnet: Deep learning on point sets for 3d classification and segmentation." , The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017

Previous work

Recent reported classification performance



*Accuracy is reported on the ModelNet40 Dataset

Representing Point clouds with Fisher Vectors

What are Fisher Vectors ?

Context – Kernel based learning & classification

- $\{(X_i, S_i)\}$ - examples = {vector description, class label}
- $K(X_i, X_j)$ - an affinity function (kernel)
- The classifier uses learned weight λ_i and is

$$\hat{S} = sign(\sum_i S_i \lambda_i K(X_i, X))$$

- Which kernel function is best ?
- Every valid kernel function may be written as an inner product between feature vectors $K(X_i, X_j) = \phi_{X_i}^T \phi_{X_j}$ (Mercer theorem)

Which feature vectors are best ?

T. Jaakkola and D. Haussler, “Exploiting generative models in discriminative classifiers,”, NIPS 1999.

Fisher Vectors

Deriving feature vectors using the class distributions

- Suppose you know the generative model (a distribution of the vector description) $P(X|\theta)$
- Then use the (simplified) Fisher score vector

$$\phi_X = \nabla_\theta \log P(X|\theta)$$

Theoretical justification:

- A differential extension of a discrimination task: consider two similar classes

$$P(X|\theta_1), P(X|\theta_{-1}) \quad s.t.\ \theta_1 \approx \theta_{-1} \approx \theta$$

- Then, use Taylor expansion

$$\log P(X|\theta_S) = \log P(X|\theta) + (\theta_S - \theta)^T \phi_X$$

- -> there is a linear classifier (in Fisher score space) which is consistent with maximum likelihood or MAP decision.
- Learning a linear, logistic regression, classifier gives a kernel classifier with

$$K(X_i, X_j) = \phi_{X_i}^T \phi_{X_j}$$

- -> using this kernel makes decisions that are as good as MAP, asymptotically

T. Jaakkola and D. Haussler, “Exploiting generative models in discriminative classifiers,”, NIPS 1999.

Fisher Vectors

For a set of independent observations $\bar{X} = \{X_1, X_2, \ldots, X_n\}$

$$\phi_{\bar{X}} = \nabla_\theta \log P(\bar{X}|\theta) = \nabla_\theta \log \Pi_i P(X_i|\theta) = \sum_i \phi_{X_i}$$

Fisher Vectors – Application to 2D object recognition

1. Characterize the image:
   a. Describe the image by a set of dense SIFT descriptors
   b. Assume the SIFTs are generated by a Gaussians mixture mode,
   c. Learn the GMM using EM (from a large image set).
   d. Re-describe the image by a single Fisher vector
2. Use the feature vectors for learning and classification (using, say, SVM).

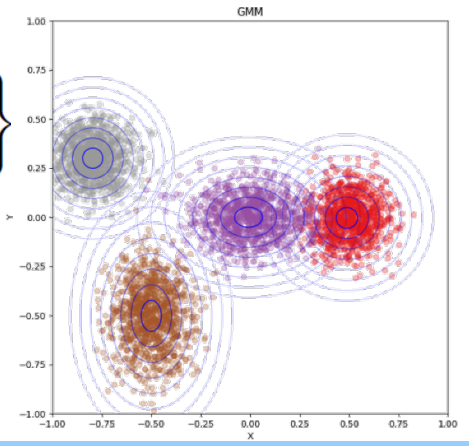
- The GMM

$$u_\lambda(\boldsymbol{p}) = \sum_{k=1}^{K} w_k u_k(\boldsymbol{p})$$

$$u_k(\boldsymbol{p}) = \frac{1}{(2\pi)^{D/2}|\Sigma_k|^{1/2}} \exp\left\{-\frac{1}{2}(\boldsymbol{p}-\mu_k)'\Sigma_k^{-1}(\boldsymbol{p}-\mu_k)\right\}$$

- The model parameters w_k, μ_k, Σ_k

Mixture weights Centers Covariance matrix



Perronnin et al. "Improving the fisher kernel for large-scale image classification." *ECCV 2010*

Here: A Gaussian Mixture Model (GMM) on a 3D grid

$$u_\lambda(\boldsymbol{p}) = \sum_{k=1}^{K} w_k u_k(\boldsymbol{p}) \qquad u_k(\boldsymbol{p}) = \frac{1}{(2\pi)^{D/2}|\Sigma_k|^{1/2}} \exp\left\{-\frac{1}{2}(\boldsymbol{p}-\mu_k)'\Sigma_k^{-1}(\boldsymbol{p}-\mu_k)\right\}$$

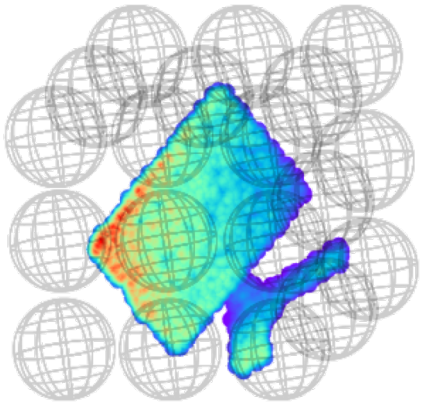
- The parameters w_k, μ_k, Σ_k

Mixture weights — Centers — Covariance matrix

- Here we use **spherical Gaussians** on a **coarse uniform grid**. (diagonal covariance matrix with equal values)

$$m \times m \times m \quad \sigma_k = \frac{1}{m} \quad m \in [3,16]$$

- Uniformity is enforced to achieve space invariance as input to CNNs



Describing a Point Cloud with Fisher Vectors

- Characterizes data samples by their deviation from a GMM generative model.
- Computes the gradients (FVs) of the log likelihood at the cloud points w.r.t model parameters
- Aggregates the gradients by averaging (invariant to point ordering)
- Constant size output
- Theoretically justified

In general $$\mathscr{G}_{\lambda}^{X} = \sum_{t=1}^{T} L_{\lambda} \nabla_{\lambda} \log u_{\lambda}(\boldsymbol{p}_t)$$ Vector of derivatives w.r.t model parameters

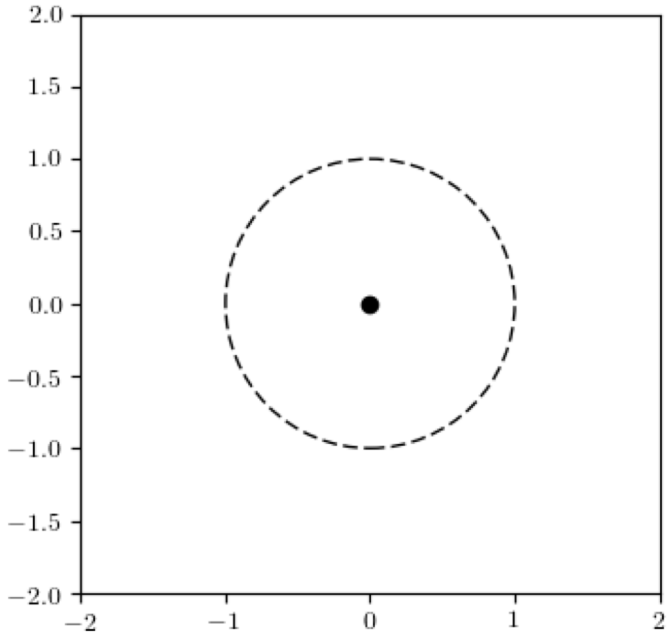
Here $$\mathscr{G}_{FV_{\lambda}}^{X} = \left(\mathscr{G}_{\alpha_1}^{X}, ..., \mathscr{G}_{\alpha_k}^{X}, \mathscr{G}_{\mu_1}^{X\prime}, ..., \mathscr{G}_{\mu_k}^{X\prime}, \mathscr{G}_{\sigma_1}^{X\prime}, ..., \mathscr{G}_{\sigma_k}^{X\prime} \right)$$

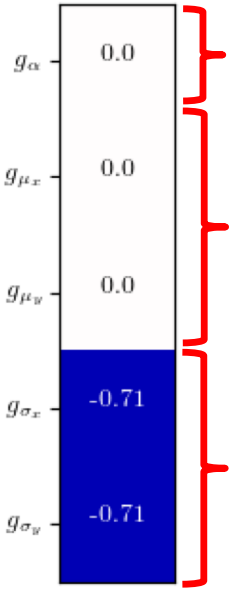
$$w_k = \frac{exp(\alpha_k)}{\sum_{j=1}^{K} exp(\alpha_j)}$$

$$\mathscr{G}_{FV_{\lambda}}^{X} \leftarrow \frac{1}{T} \mathscr{G}_{FV_{\lambda}}^{X}$$ - Normalize derivatives by sample size

Illustration: One Point, One Gaussian, FV

Each Gaussian "generates" a vector which represents all the data w.r.t it





$$\mathcal{G}^X_{\alpha_k} = \frac{1}{\sqrt{w_k}} \sum_{t=1}^{T} (\gamma_t(k) - w_k)$$

Derivative w.r.t Gaussian **weights**

$$\mathcal{G}^X_{\mu_k} = \frac{1}{\sqrt{w_k}} \sum_{t=1}^{T} \gamma_t(k) \left(\frac{\boldsymbol{p}_t - \mu_k}{\sigma_k} \right)$$

Derivative w.r.t Gaussian **expected value** (centers)

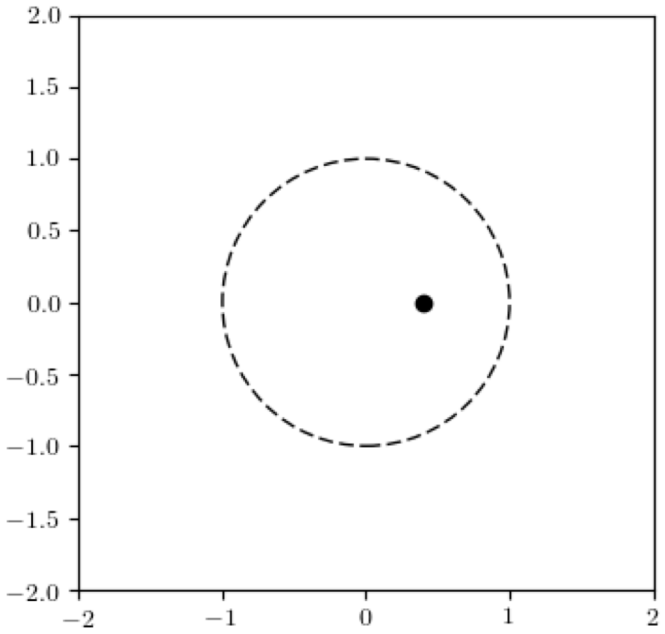
$$\mathcal{G}^X_{\sigma_k} = \frac{1}{\sqrt{2w_k}} \sum_{t=1}^{T} \gamma_t(k) \left[\frac{(\boldsymbol{p}_t - \mu_k)^2}{\sigma_k^2} - 1 \right]$$

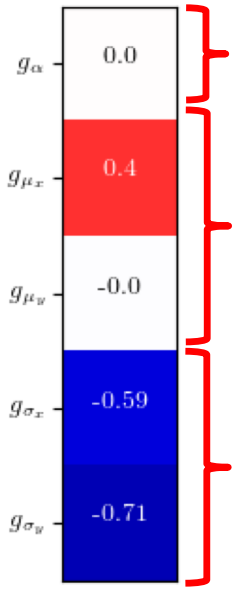
Derivative w.r.t Gaussian **stds**

$$\gamma_t(k) = \frac{w_k u_k(\boldsymbol{p}_t)}{\sum_{j=1}^{K} w_j u_j(\boldsymbol{p}_t)}$$

Illustration: One Point, One Gaussian, FV

Each Gaussian "generates" a vector which represents all the data w.r.t it





$$\mathcal{G}^X_{\alpha_k} = \frac{1}{\sqrt{w_k}} \sum_{t=1}^{T} (\gamma_t(k) - w_k)$$

Derivative w.r.t Gaussian **weights**

$$\mathcal{G}^X_{\mu_k} = \frac{1}{\sqrt{w_k}} \sum_{t=1}^{T} \gamma_t(k) \left(\frac{\boldsymbol{p}_t - \mu_k}{\sigma_k} \right)$$

Derivative w.r.t Gaussian **expected value** (centers)

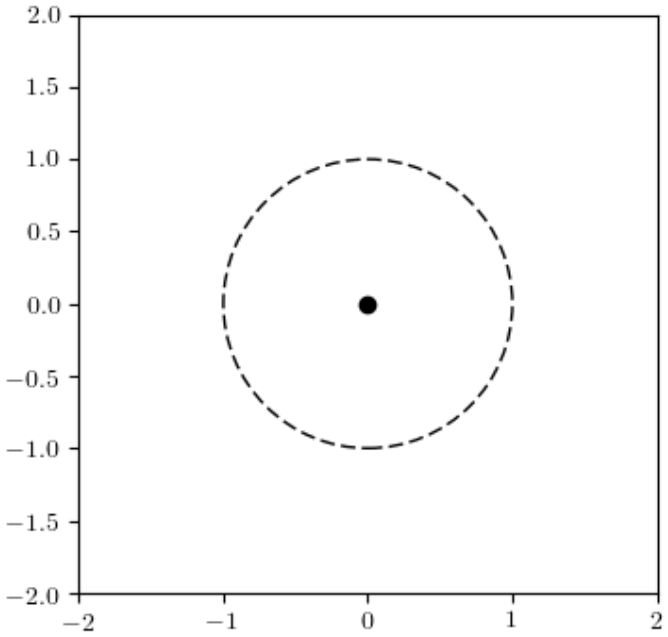
$$\mathcal{G}^X_{\sigma_k} = \frac{1}{\sqrt{2w_k}} \sum_{t=1}^{T} \gamma_t(k) \left[\frac{(\boldsymbol{p}_t - \mu_k)^2}{\sigma_k^2} - 1 \right]$$

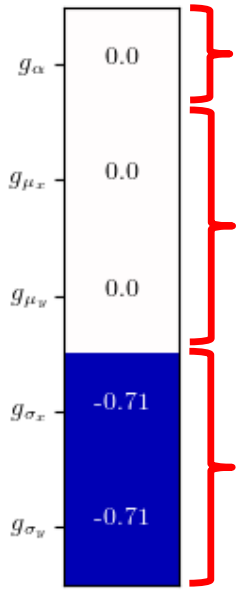
Derivative w.r.t Gaussian **stds**

$$\gamma_t(k) = \frac{w_k u_k(\boldsymbol{p}_t)}{\sum_{j=1}^{K} w_j u_j(\boldsymbol{p}_t)}$$

Illustration: One Point, One Gaussian, FV

Each Gaussian "generates" a vector which represents all the data w.r.t it





$$\mathcal{G}^X_{\alpha_k} = \frac{1}{\sqrt{w_k}} \sum_{t=1}^{T} (\gamma_t(k) - w_k)$$

Derivative w.r.t Gaussian **weights**

$$\mathcal{G}^X_{\mu_k} = \frac{1}{\sqrt{w_k}} \sum_{t=1}^{T} \gamma_t(k) \left(\frac{\boldsymbol{p}_t - \mu_k}{\sigma_k} \right)$$

Derivative w.r.t Gaussian **expected value** (centers)

$$\mathcal{G}^X_{\sigma_k} = \frac{1}{\sqrt{2w_k}} \sum_{t=1}^{T} \gamma_t(k) \left[\frac{(\boldsymbol{p}_t - \mu_k)^2}{\sigma_k^2} - 1 \right]$$

Derivative w.r.t Gaussian **stds**

$$\gamma_t(k) = \frac{w_k u_k(\boldsymbol{p}_t)}{\sum_{j=1}^{K} w_j u_j(\boldsymbol{p}_t)}$$

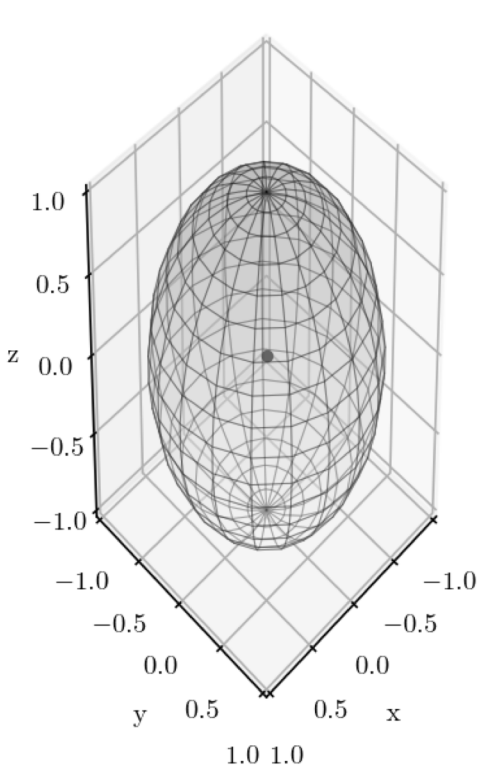
3DmFV Representation

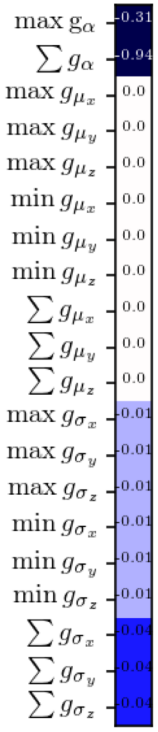
3D **modified** Fisher vector (3DmFV) representation

- Uniform grid GMM
- Additional permutation invariant ("symmetric") function (min, max)

$$\mathcal{G}^X_{3DmFV_\lambda} = \begin{bmatrix} \sum_{t=1}^{T} L_\lambda \nabla_\lambda \log u_\lambda(\boldsymbol{p}_t)\Big|_{\lambda=\alpha,\mu,\sigma} \\ \max_t (L_\lambda \nabla_\lambda \log u_\lambda(\boldsymbol{p}_t)|_{\lambda=\alpha,\mu,\sigma} \\ \min_t (L_\lambda \nabla_\lambda \log u_\lambda(\boldsymbol{p}_t))|_{\lambda=\mu,\sigma} \end{bmatrix}$$

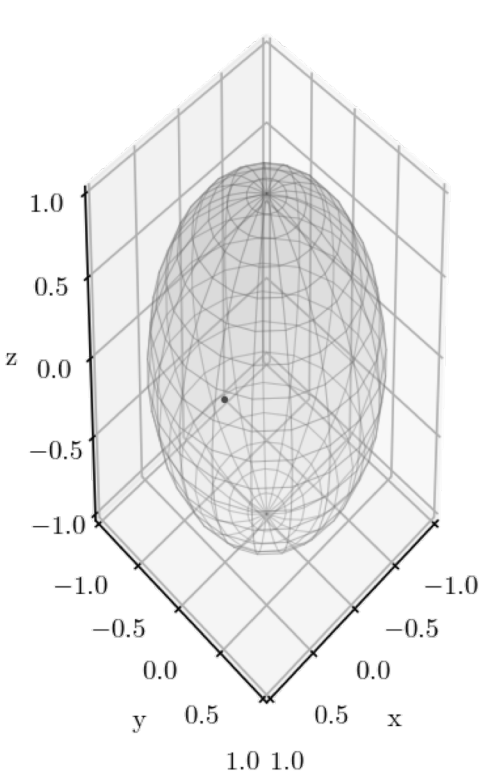
3DmFV Representation

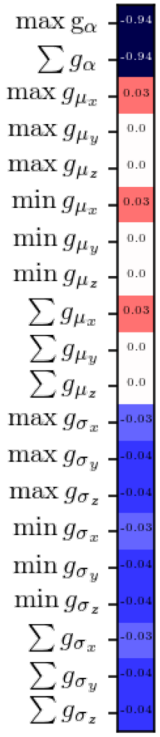




*No normalization for visualization purposes

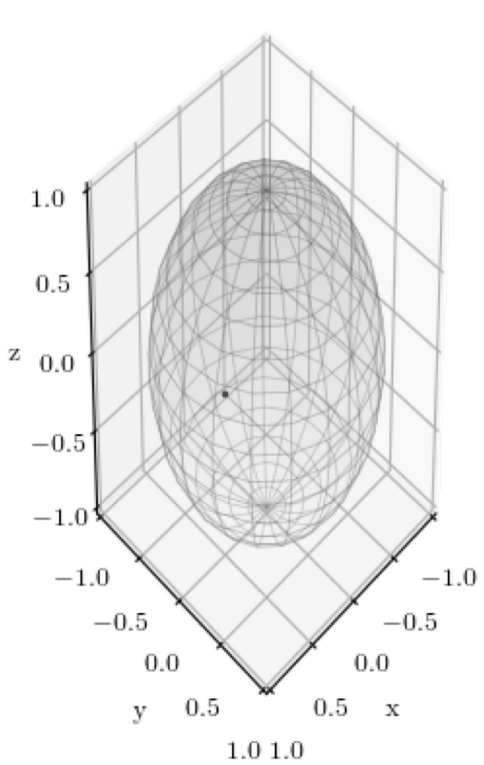
3DmFV Representation

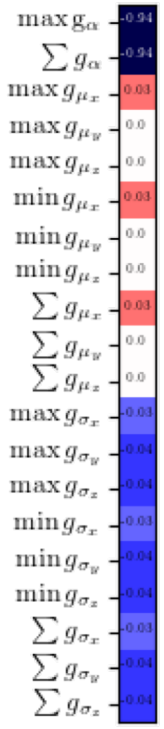




*No normalization for visualization purposes

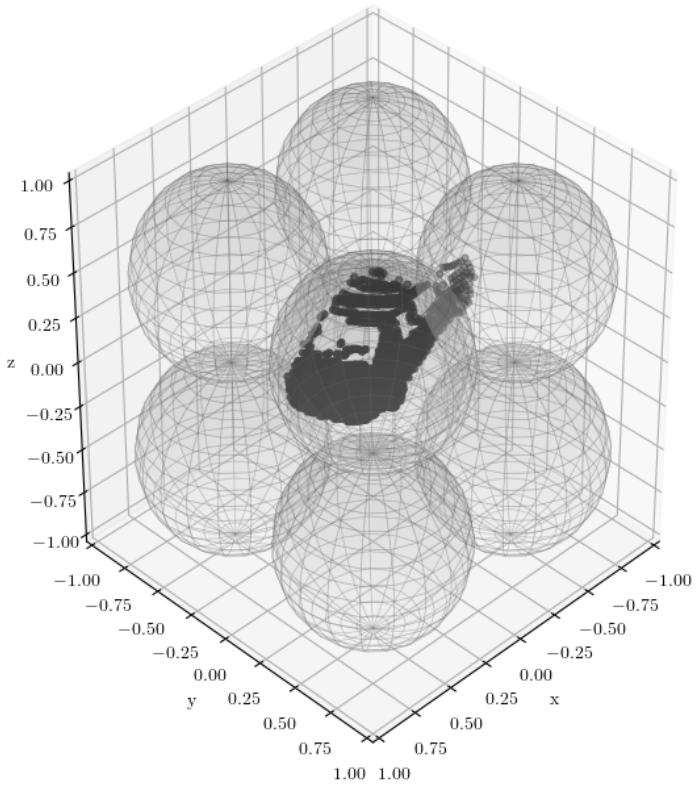
3DmFV Representation

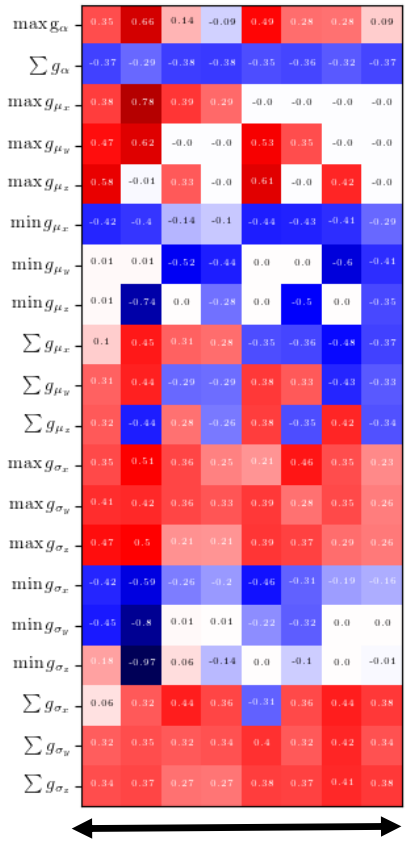




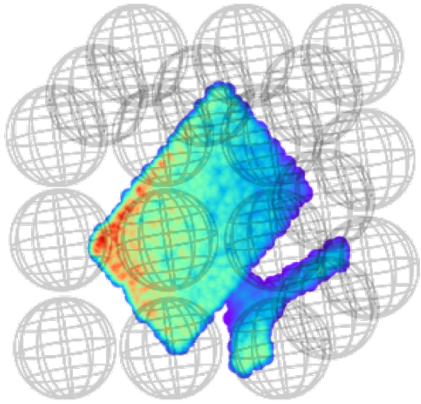
*No normalization for visualization purposes

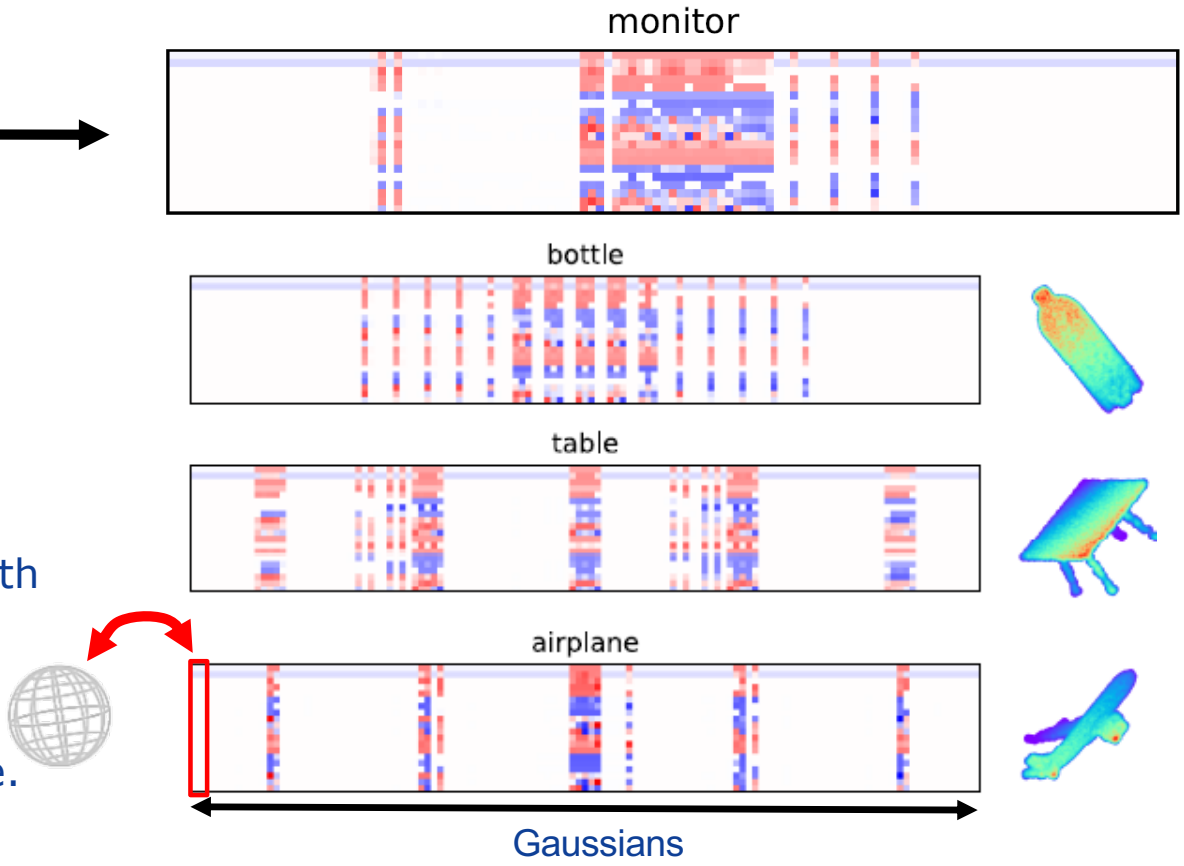
3DmFV Representation - Example





3DmFV Visualization





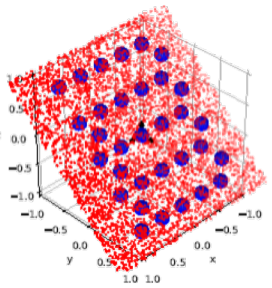
- Images are used for visualization purposes only.
- Every column corresponds to the gradient components associated with one Gaussian (and one 3D spatial position)
- The full descriptor is a 4D structure.

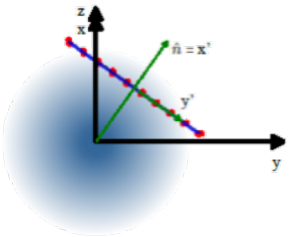
Y. Ben-Shabat, M. Lindenbaum, A. Fischer. "3DmFV: 3D Point Cloud Classification in Real-Time using Convolutional Neural Networks", IEEE Robotics and Automation Letters, and IROS 2018.

Point cloud reconstruction from FV

- FV is continuous on the point set (unlike voxels)
- Reconstructing from FV: simple cases
  - FV calculated relative to a single Gaussian representing a single point – analytic reconstruction of the p $\boldsymbol{p}_1 = \sigma_k \mathcal{G}^X_{\mu_k} + \mu_k$
  - FV calculated relative to a single Gaussian representing multiple points on one plane – analytic reconstruction of the plane

$$a = \frac{\mathcal{G}_{\mu_x}}{\|\mathcal{G}_\mu\|}, b = \frac{\mathcal{G}_{\mu_y}}{\|\mathcal{G}_\mu\|}, c = \frac{\mathcal{G}_{\mu_z}}{\|\mathcal{G}_\mu\|}, \rho = \sigma \|\mathcal{G}_\mu\| \sqrt{w}$$

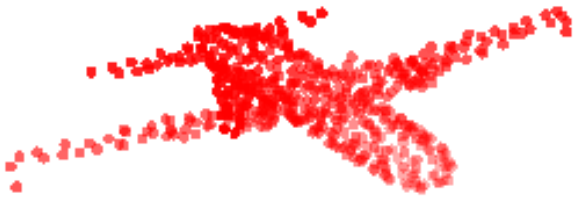




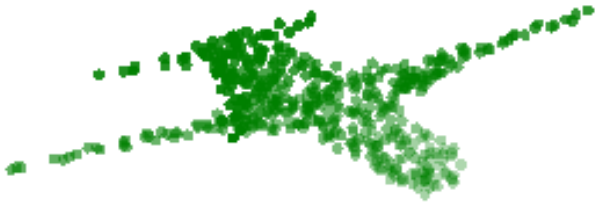
Under the assumption of sharply peaked $\gamma_t(k)$

Point cloud reconstruction from FV

- Reconstructing points from FV:
  - FV consisting of multiple Gaussians and multiple points – reconstruction using a Deep decoder network

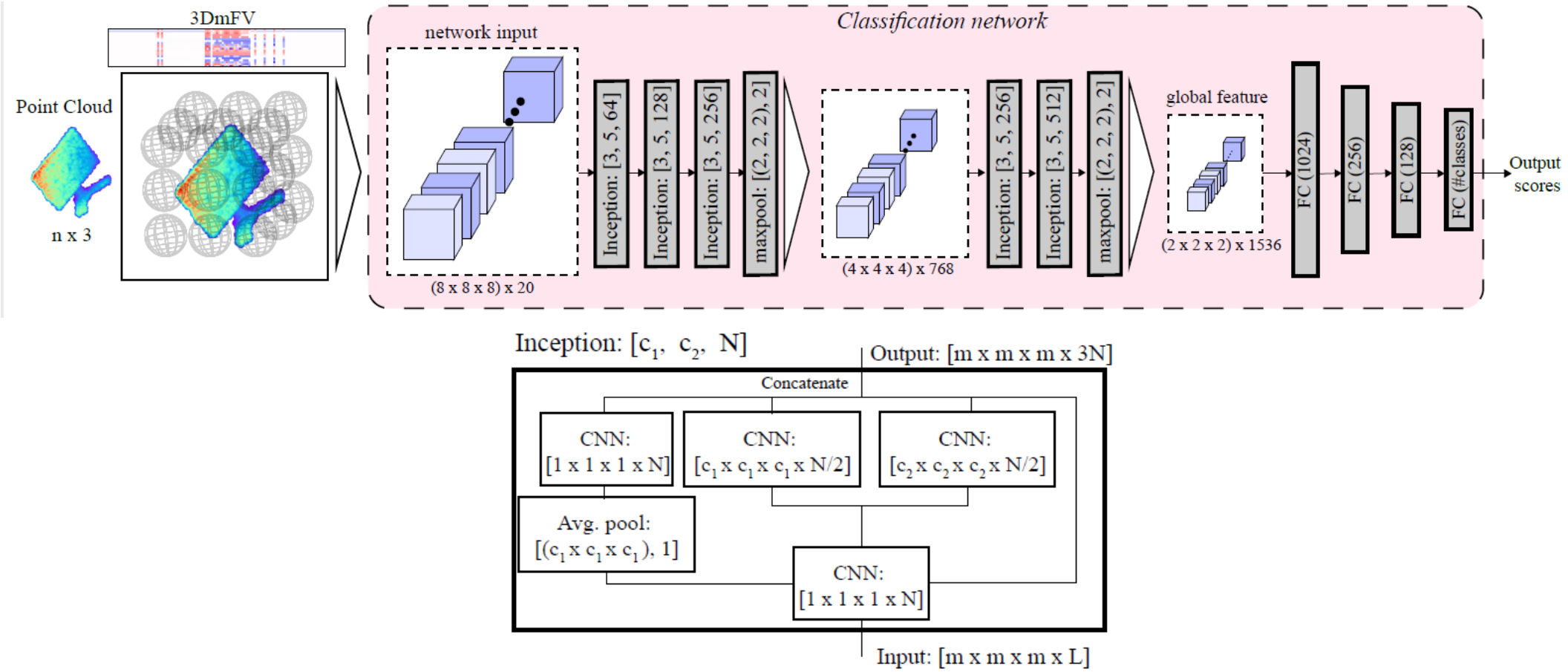


Original



Reconstruction

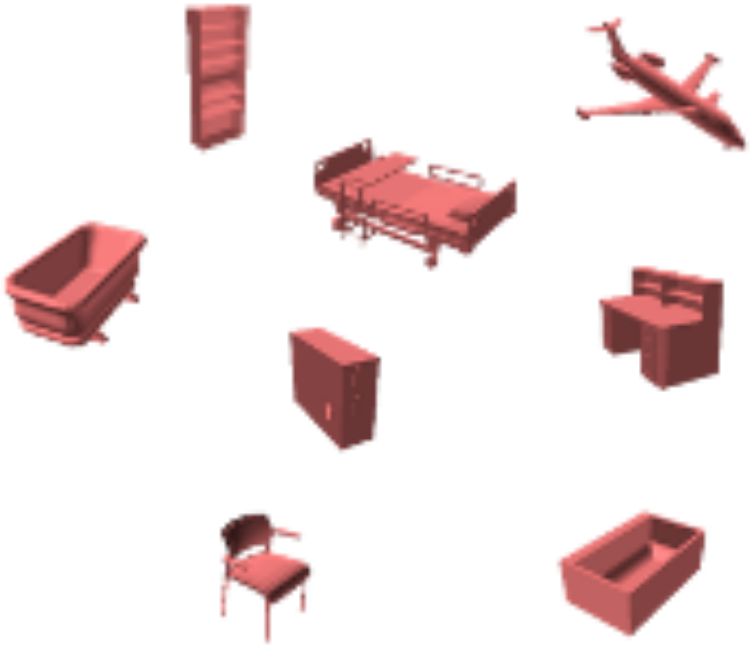
3DmFV-Net - classification



Y. Ben-Shabat, M. Lindenbaum, A. Fischer. "3DmFV: 3D Point Cloud Classification in Real-Time using Convolutional Neural Networks", IEEE Robotics and Automation Letters, and IROS 2018.

Benchmark Dataset

- **Modelnet40**
  - ~12.5K CAD models (triangle mesh)
  - 40 man-made object categories
  - ~10K for training
  - ~2.5K for testing
- **Modelnet10**
  - ~5K CAD models (triangle mesh)
  - 10 man-made object categories
  - ~4K for training
  - ~1K for testing



Wu, Zhirong, et al. "3d shapenets: A deep representation for volumetric shapes." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015. /http://modelnet.cs.princeton.edu

Training details

- **Number of points: 2048 (for best performance)**
- **Point cloud manipulation**: Centered around the origin and scaled to fit a cube of edge length 2.
- **Data augmentation**:
  - Random anisotropic scaling (range [0.66, 1.5])
  - Random translation (range: [-0.2, 0.2])
  - Gaussian noise (std of 0.01)
- Implemented in Tensorflow and trained on Nvidia Titan Xp GPU
- Time:– ~7h
- Optimizer: Adam
- Learning rate: 0.001
- Learning rate decay: 0.7 every 20 epochs
- Activation function: ReLU
- Dropout of 0.7 keep ratio between each FC layer
- Batch Size: 64

Classification Accuracy Results

Voxel and Multi-view methods

Method	ModelNet10	Modelnet40
MVCNN [28]	-	90.1
3DShapeNets [33]	83.5	77.32
VoxNet [18]	92.0	83.0
VRN [4]	93.6	91.33
FusionNet [11]	93.1	90.8
3DmFV+VoxNet	94.3	88.5
Our 3DmFV-Net	**95.2**	**91.4**

Point methods

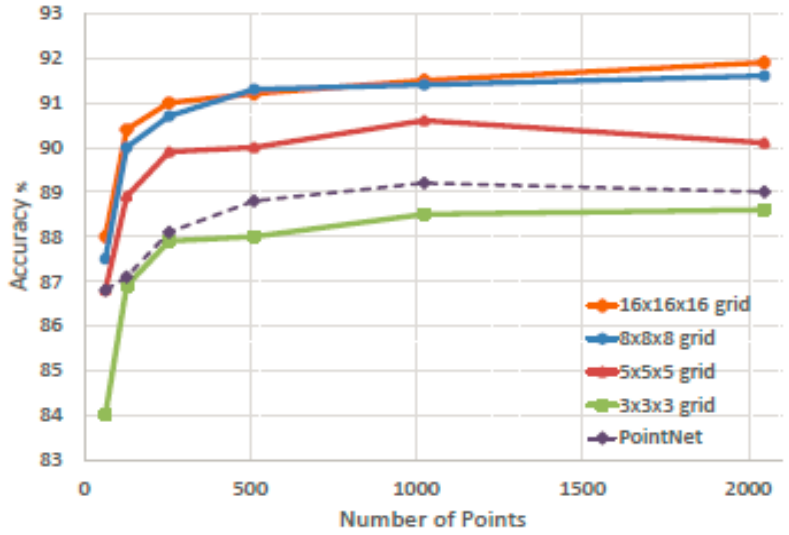
Method	ModelNet10	Modelnet40
PointNet [22]	-	89.2
PointNet++ [24]	-	90.7
Kd-network [14]	93.3	90.6
3DmFV+VoxNet	94.3	88.5
Our 3DmFV-Net	**95.2**	**91.4**

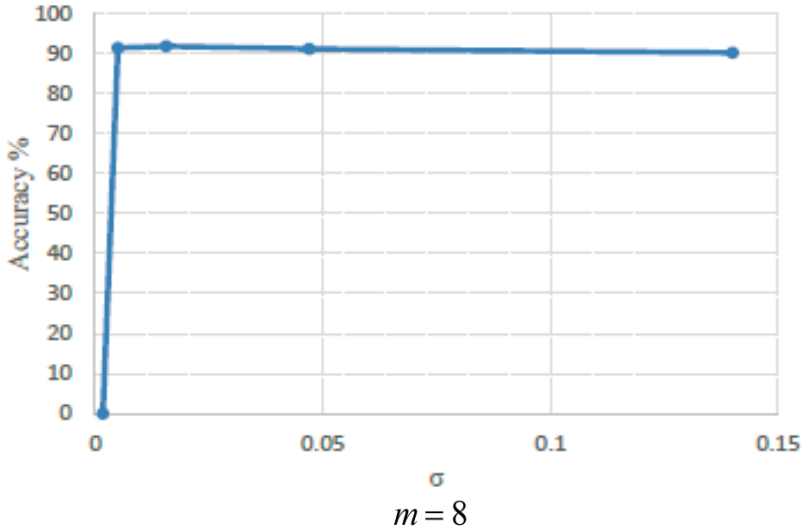
*Note: Performance is measured in equivalent experimental conditions i.e. single architecture, 1024 points

3DmFV parameter influence

- Grid or not ?
- Grid size
- Standard deviation (σ)
- Symmetric function

Rep.	ML + LinCls	ML + NonLinCls	Grid + NonLinCls
FV	58.4	82.8	84.5
3DmFV-ss	58.8	85.0	84.4
3DmFV-min	67.7	87.7	86.1
3DmFV-max	68.6	87.4	85.3
3DmFV	76.8	88.0	87.7



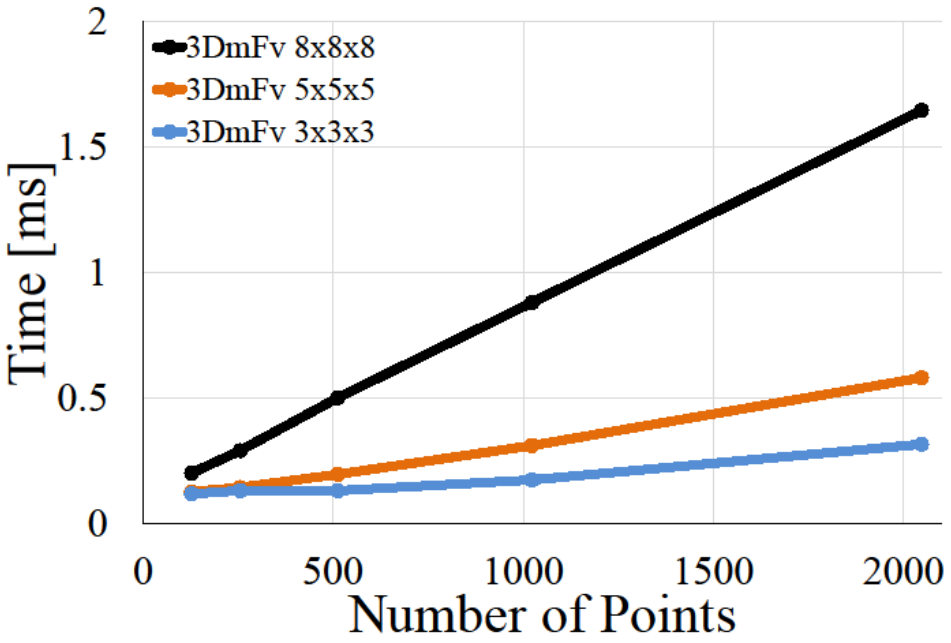


$m = 8$

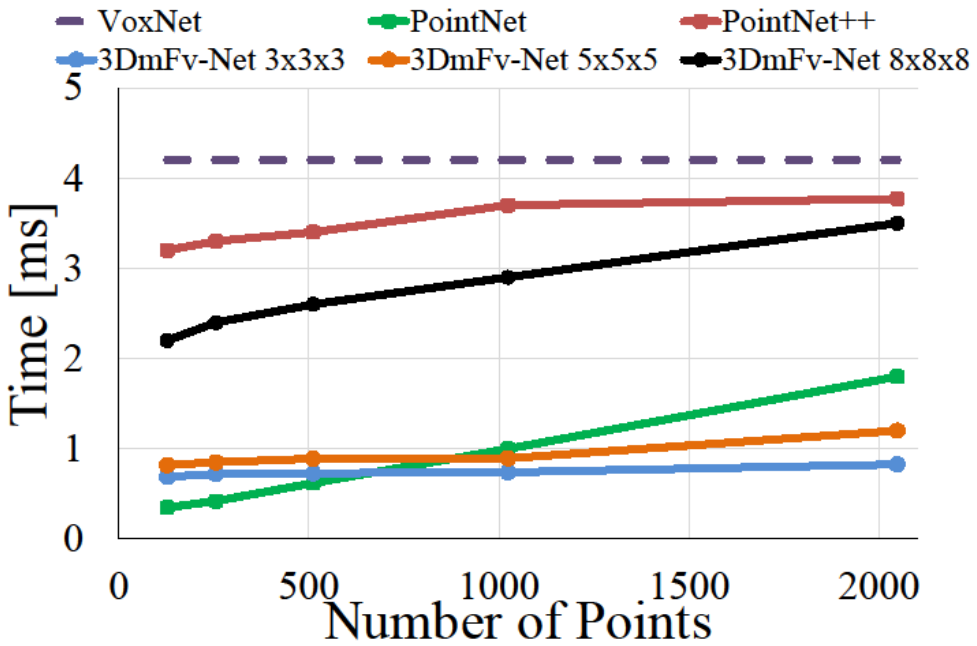
Run-time

Real-time performance

Theoretical time complexity of $O(TK)$ is validated empirically



Representation computation time



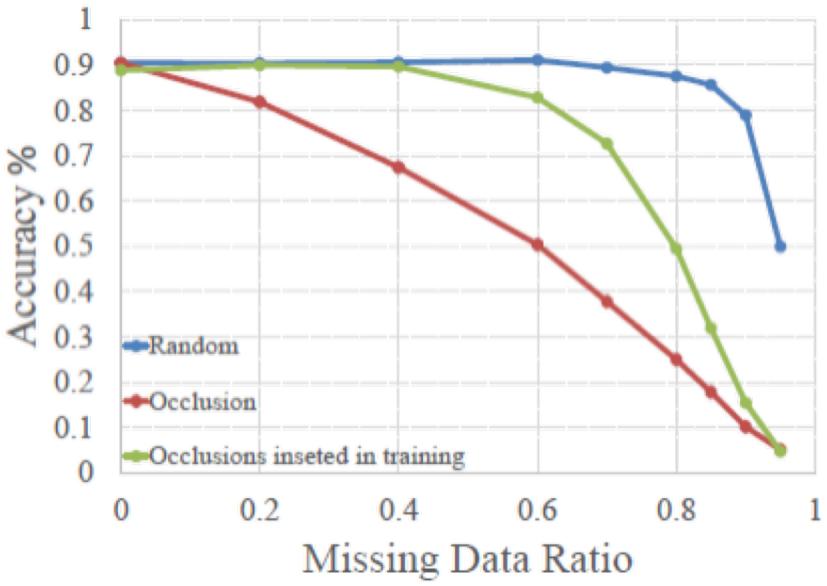
Total inference time

*Results are averaged over 2448 point clouds subdivided into batches of 16 on a Titan Xp GPU

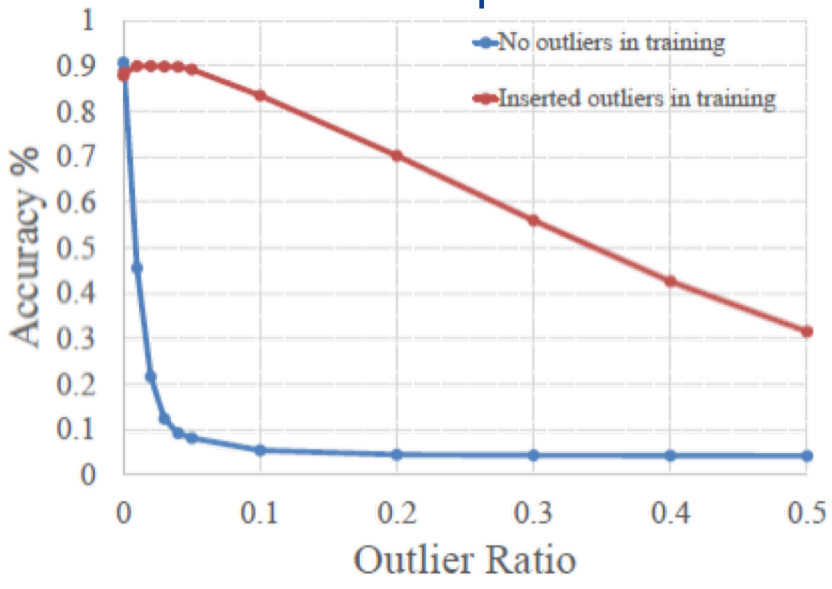
Robustness

- Point deletion: Uniform deletion , focused region deletion
- Outlier points: Random in space

Point deletion

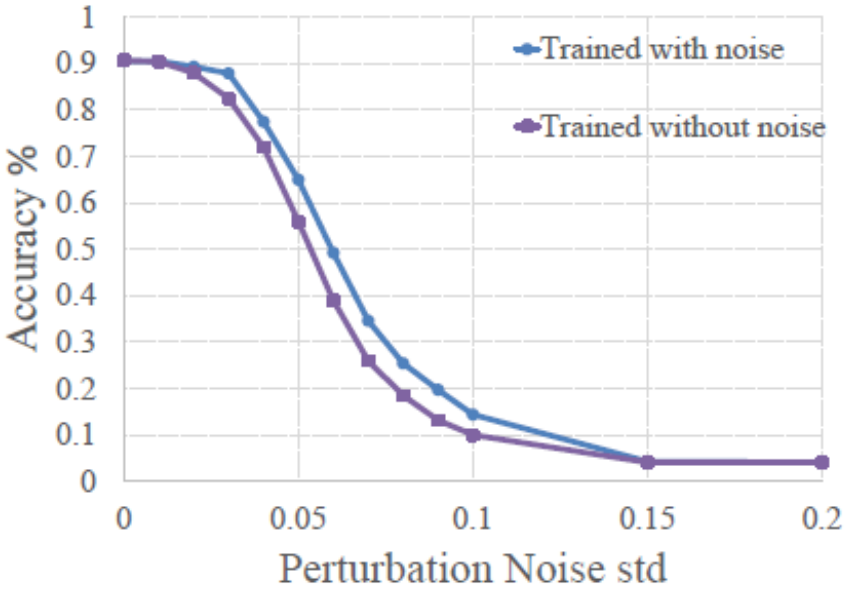


Outlier points

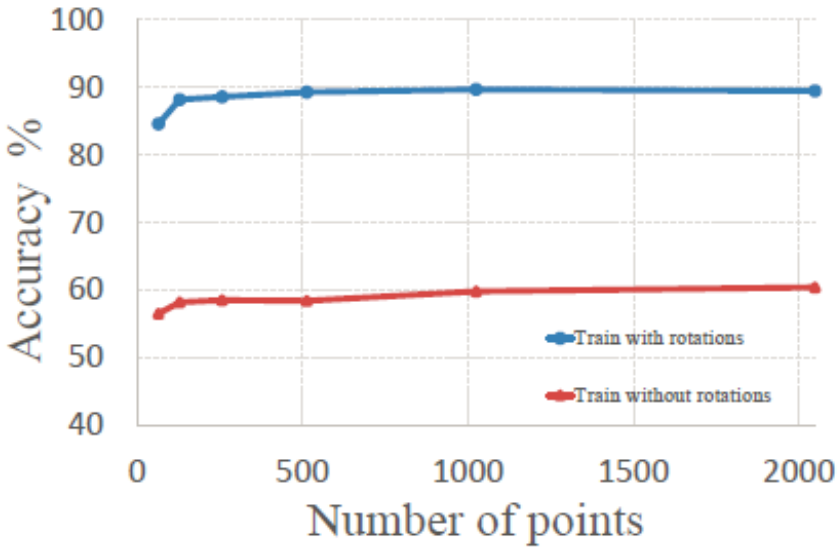


Robustness

Perturbation noise



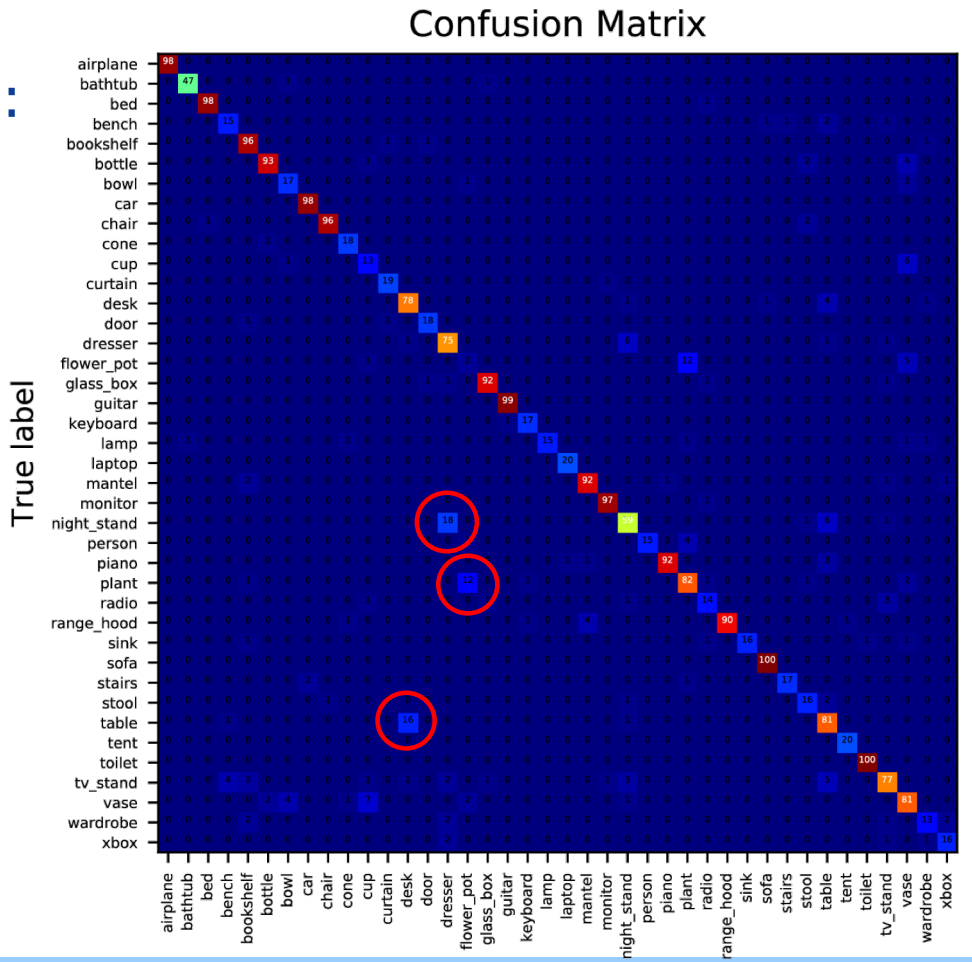
Random rotation



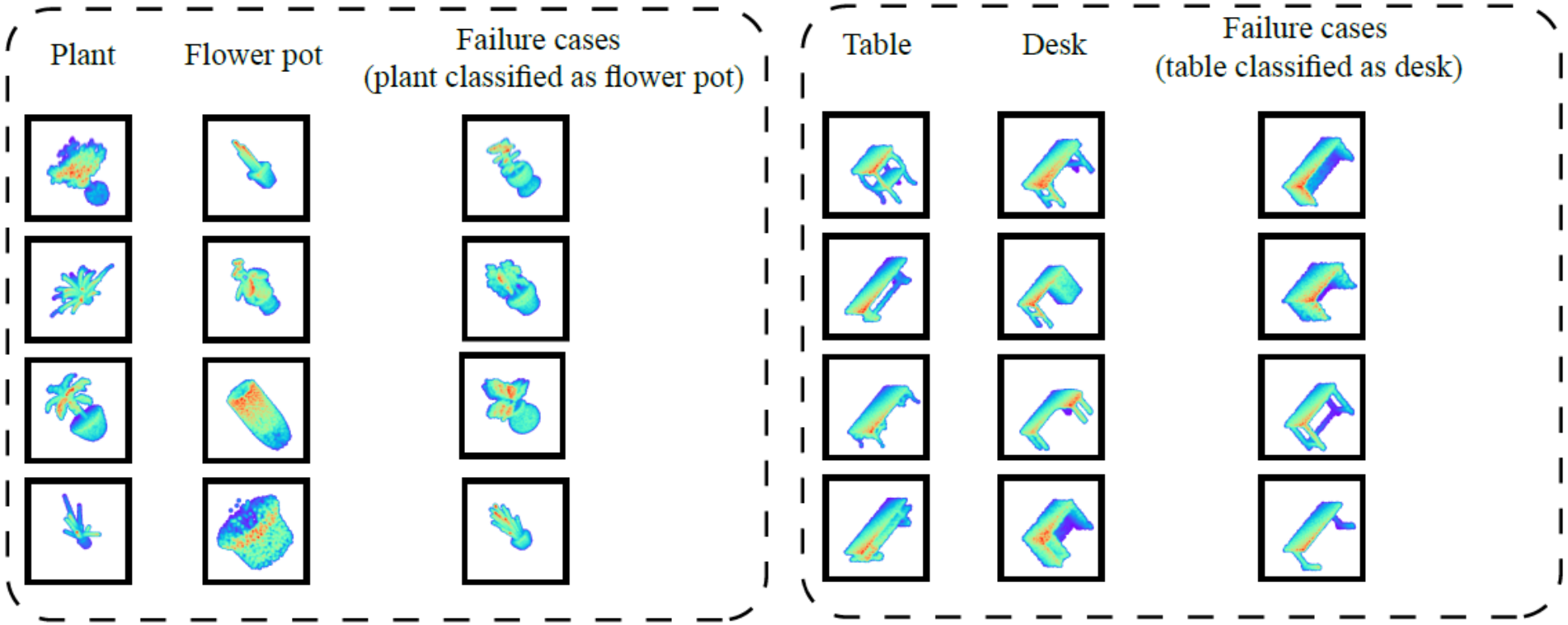
Classification Failure Cases

Many failures occur in specific pairs:

- Table – desk
- Dresser – night stand
- Flower pot - plant

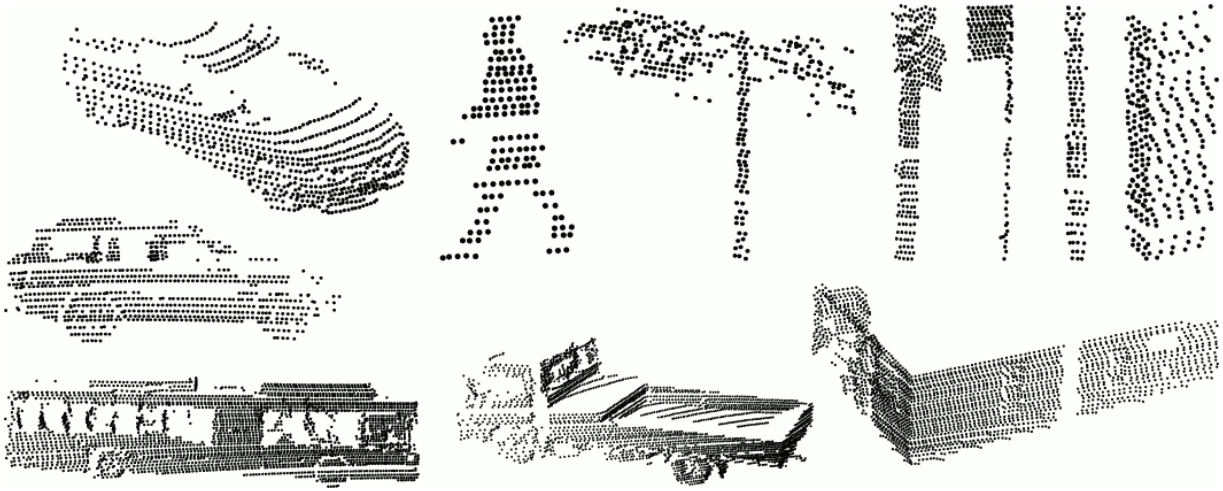


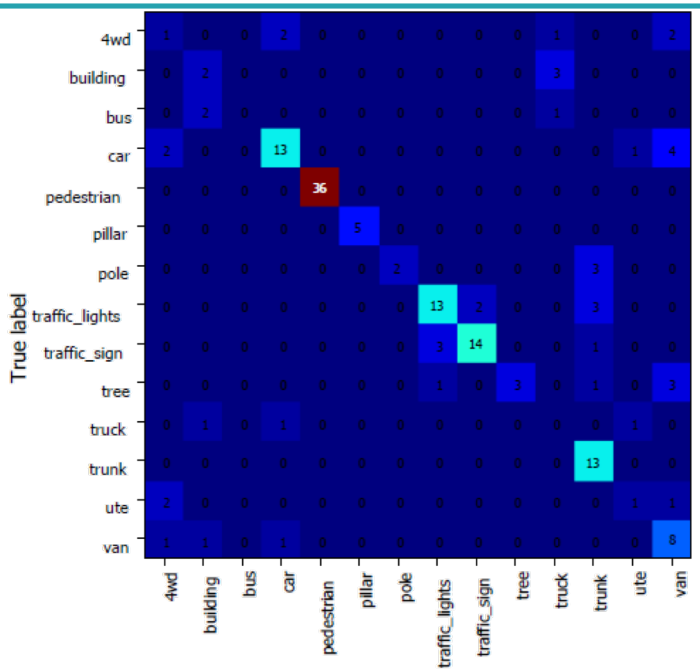
Classification Failure Cases



Results on Sydney Dataset - Outdoor

- LiDAR scans
- 14 object classes
- 588 total objects (subdivided into 4 folds)
- Imbalanced classes

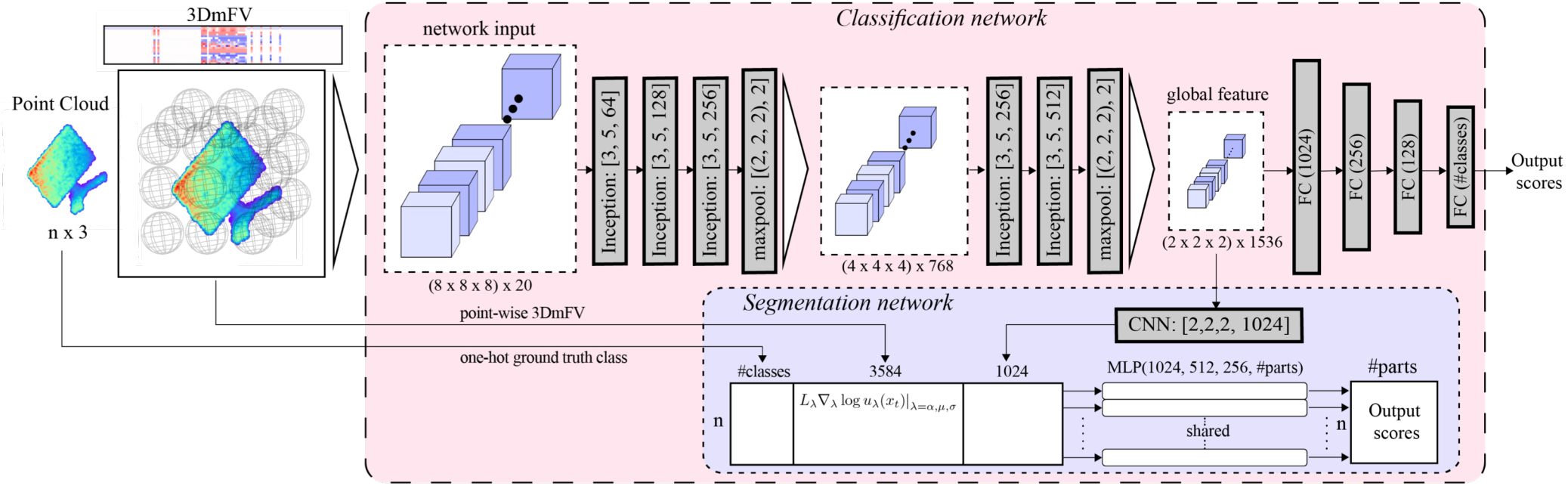




Method	F1 score
Triangle+SVM [16]	0.67
GBH+SVM [15]	0.71
VoxNet [3]	0.72
3DmFV-Net (no aug.)	0.73
3DmFV-Net (rotation aug.)	**0.76**

*http://www.acfr.usyd.edu.au/papers/SydneyUrbanObjectsDataset.shtml

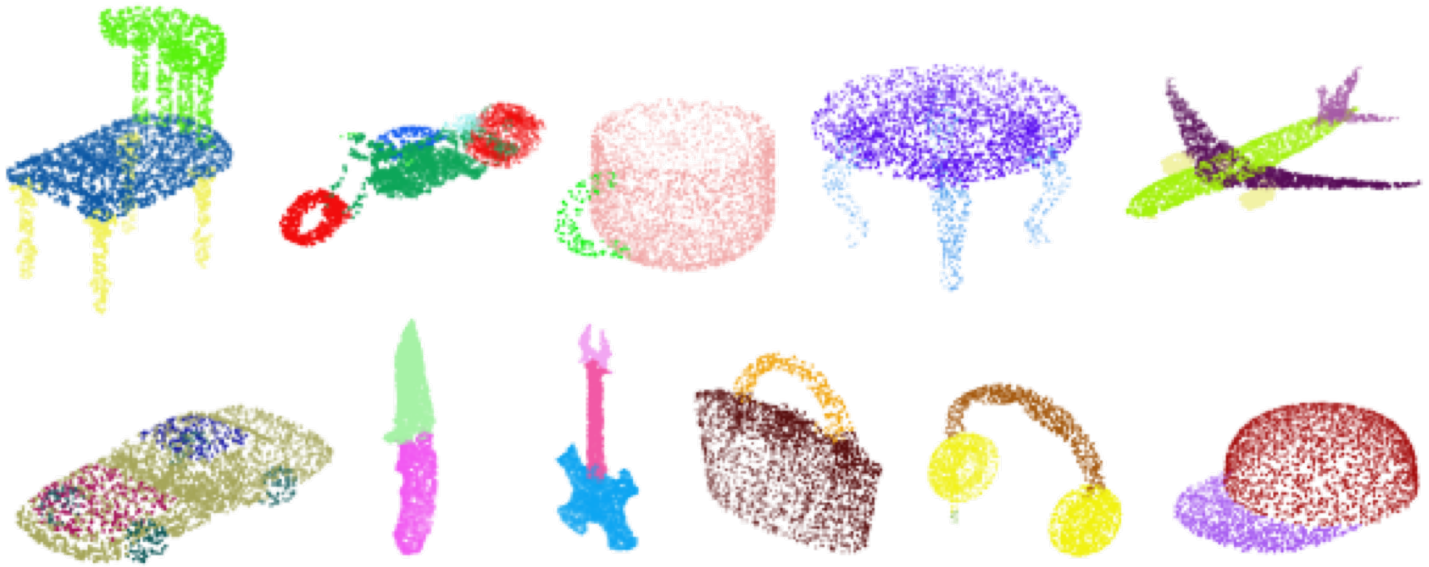
3DmFV-Net – Part segmentation



Ben-Shabat, Y., Lindenbaum, M. and Fischer, A., 3D Point Cloud Classification and Segmentation using 3D Modified Fisher Vector Representation for Convolutional Neural Networks. IROS. 2018.

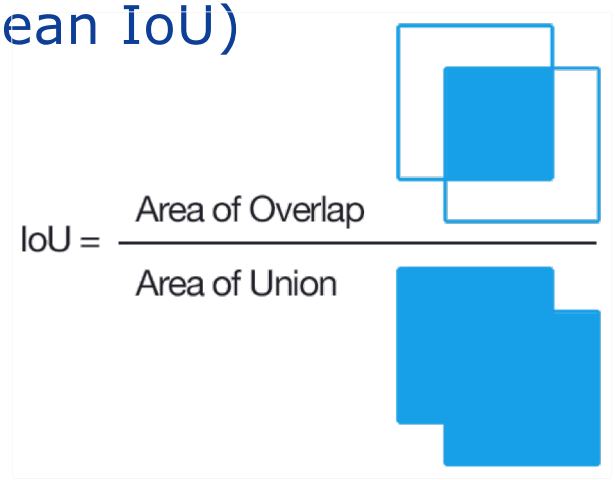
Part Segmentation Qualitative Results

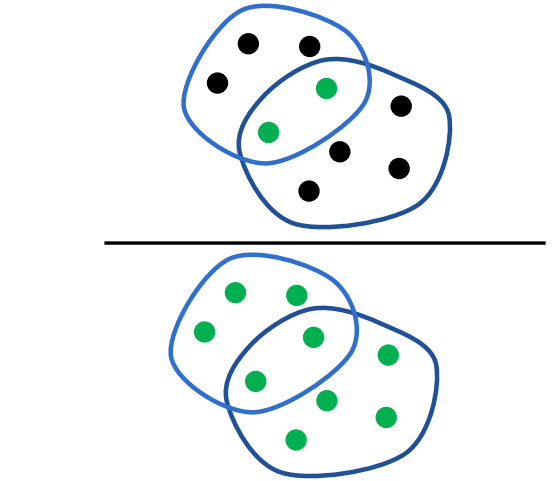
- ShapeNet part dataset
- Contains ~17K point clouds with 50 annotated parts from 16 categories.
- Imbalanced dataset



Part Segmentation Quantitative Results

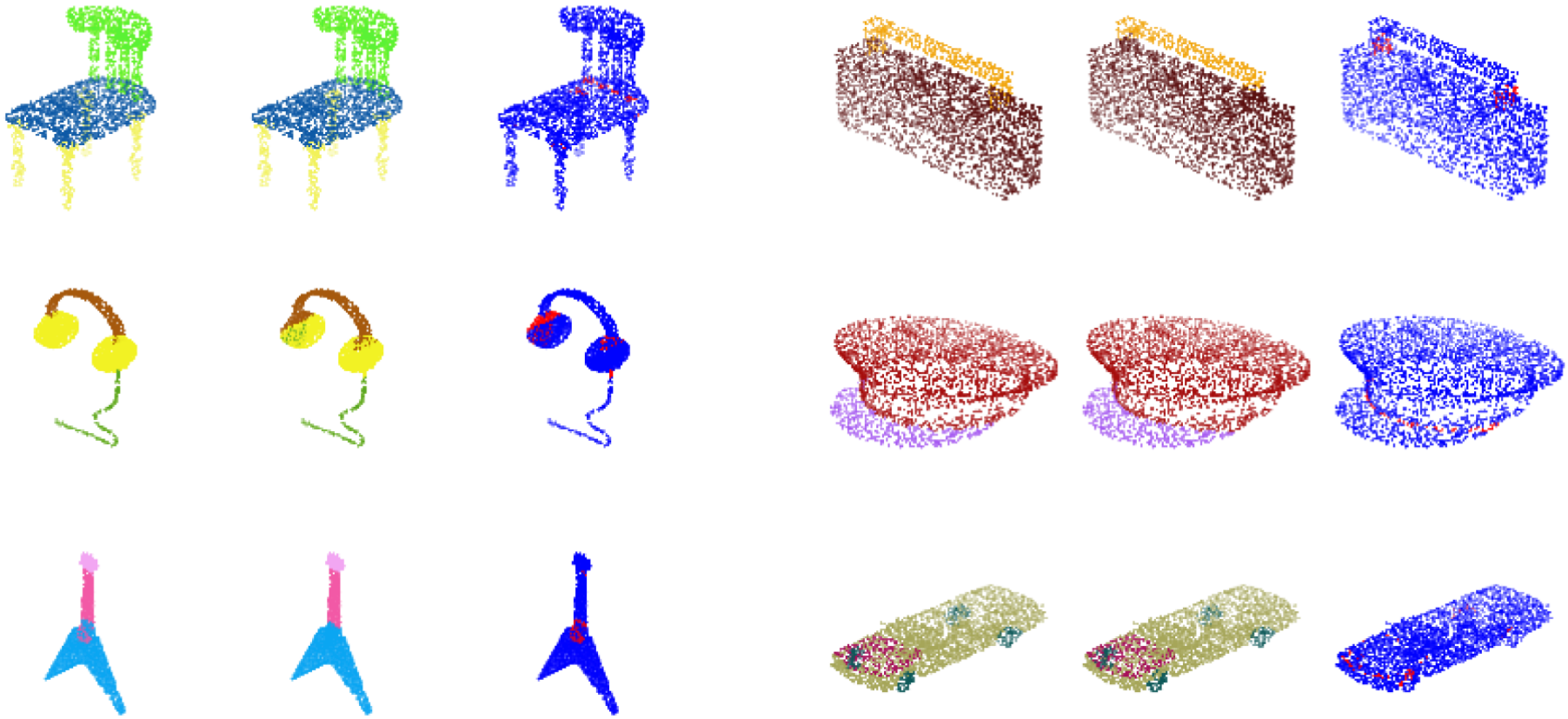
Evaluation metric (mean IoU)





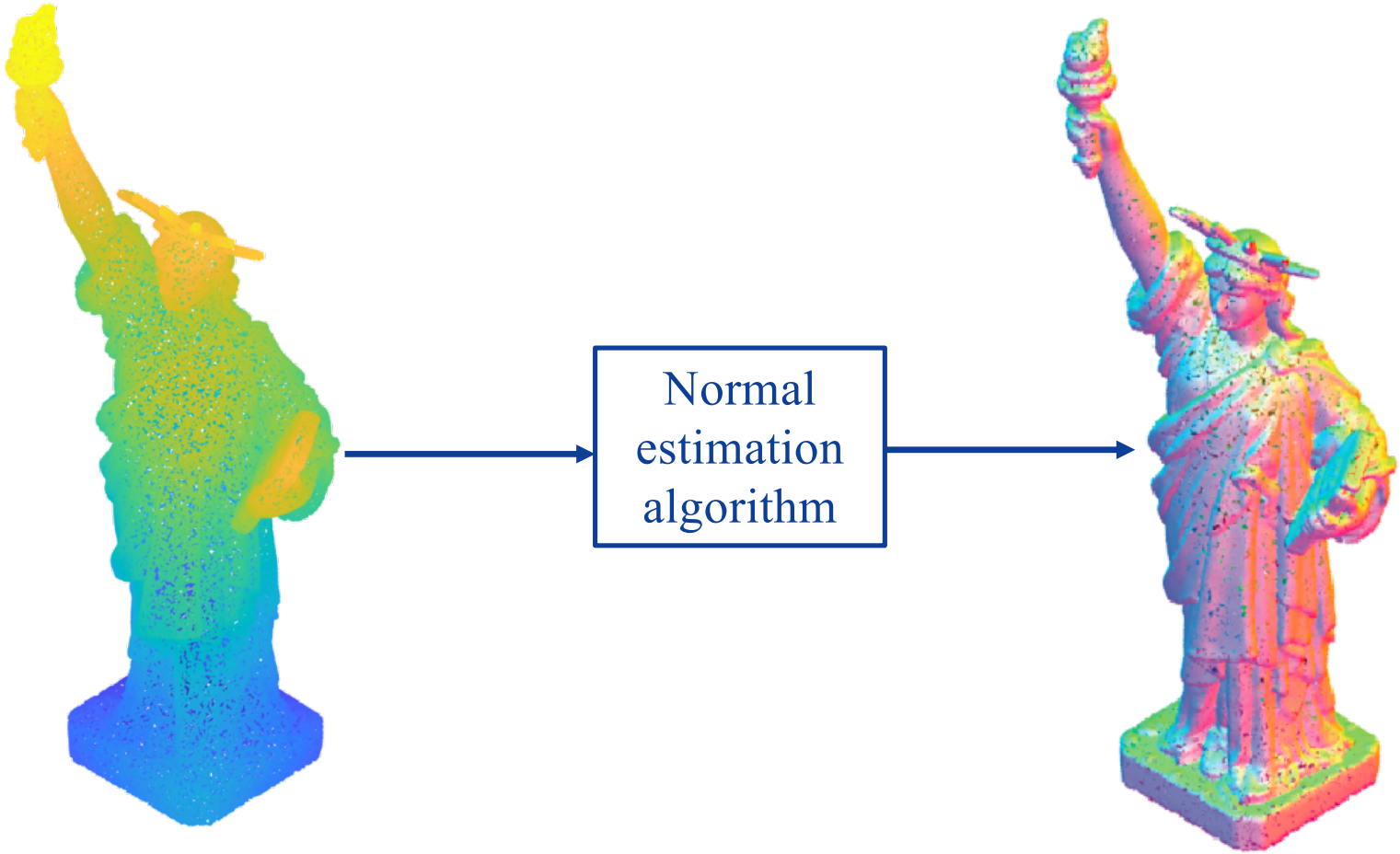
method	mean	aero	bag	cap	car	chair	ear phone	guitar	knife	lamp	laptop	motor bike	mug	pistol	rocket	skate board	table
Yi [34]	81.4	81.0	78.4	77.7	75.7	87.6	61.9	**92.0**	85.4	82.5	**95.7**	70.6	91.9	**85.9**	53.1	69.8	75.3
3DCNN [22]	79.4	75.1	72.8	73.3	70.0	87.2	63.5	88.4	79.6	74.4	93.9	58.7	91.8	76.4	51.2	65.3	77.1
PointNet [22]	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	81.2	57.9	72.8	80.6
Kd-Net [14]	77.2	79.9	71.2	80.9	68.8	88.0	72.4	88.9	**86.4**	79.8	94.9	55.8	86.5	79.3	50.4	71.1	80.2
PointNet++ [24]	**85.1**	82.4	79.0	**87.7**	**77.3**	**90.8**	71.8	91.0	85.9	**83.7**	95.3	**71.6**	**94.1**	81.3	**58.7**	**76.4**	**82.6**
Ours	84.3	82.0	**84.3**	86.0	76.9	89.9	**73.9**	90.8	85.7	82.6	95.2	66.0	94.0	82.6	51.5	73.5	81.8

Part Segmentation Results



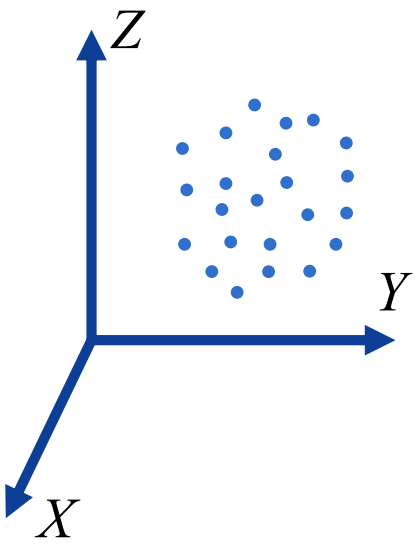
GT Prediction Difference GT Prediction Difference

Normal Estimation

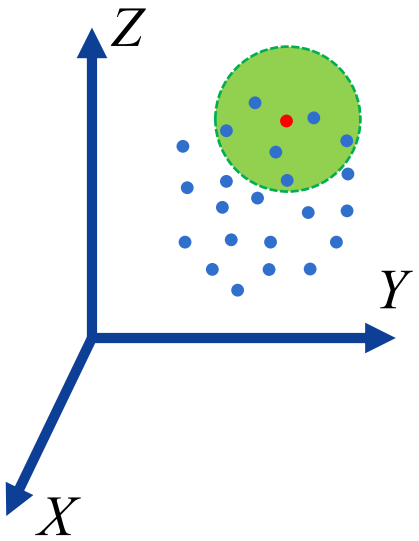


Previous work

Input point cloud

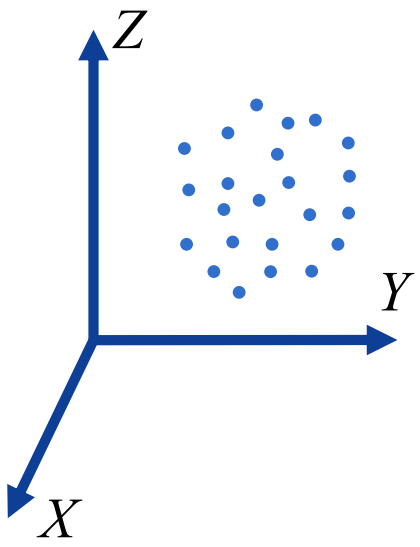


Extract subset

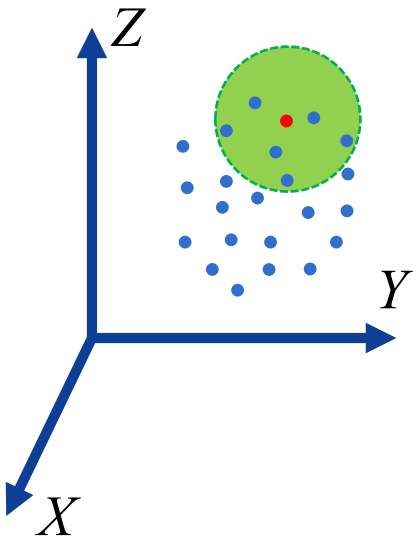


Previous work

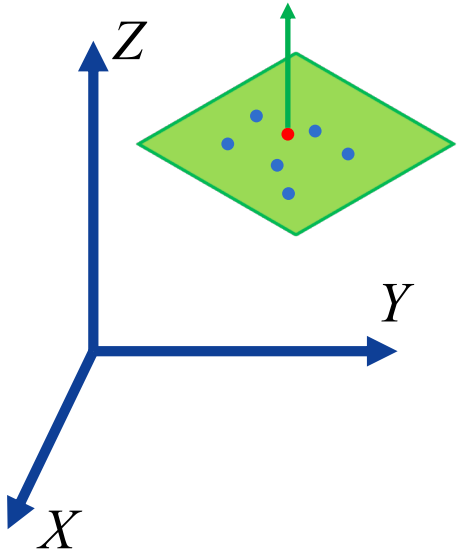
Input point cloud



Extract subset

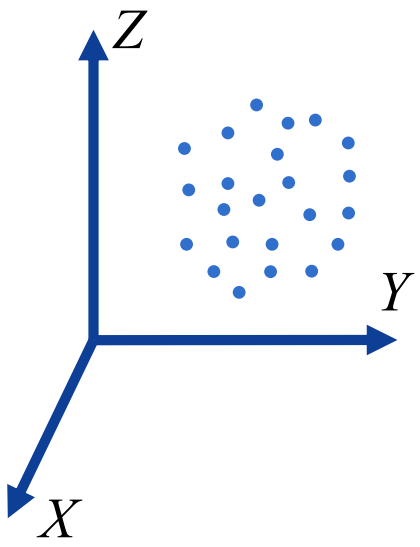


Fit a surface

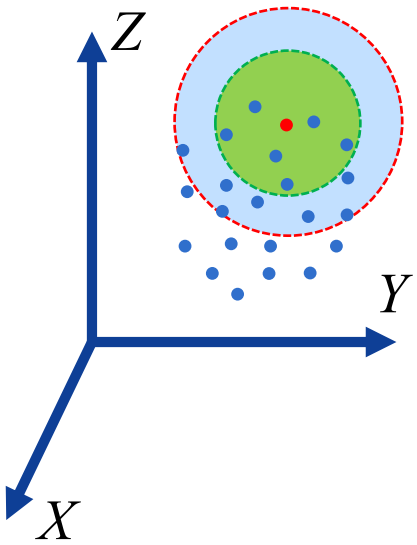


Previous work

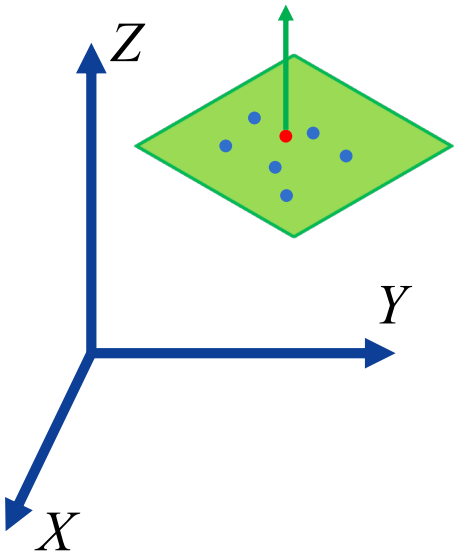
Input point cloud



Extract subset



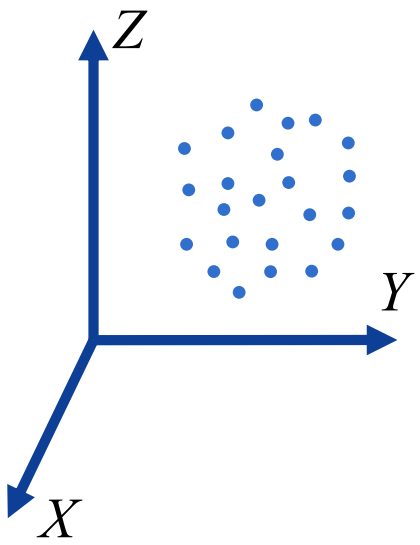
Fit a surface



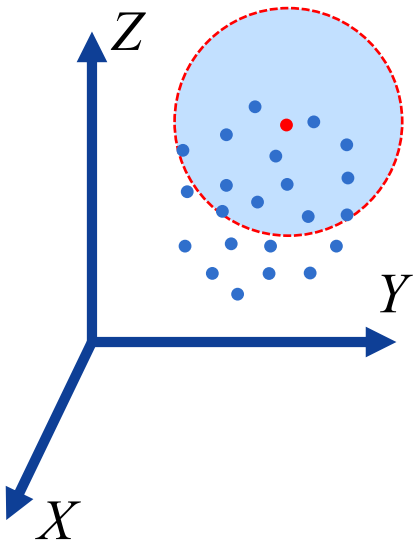
Previous work

Can we learn to select the best radius?

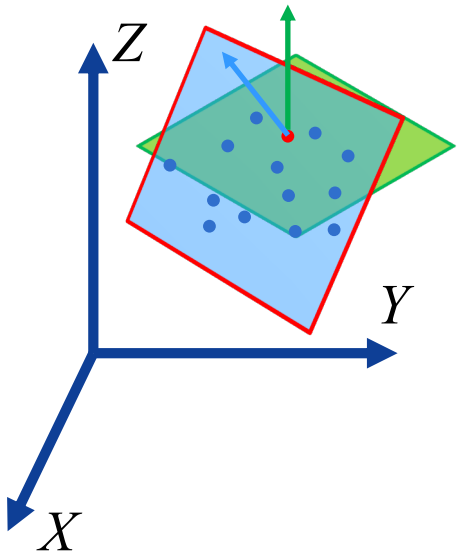
Input point cloud



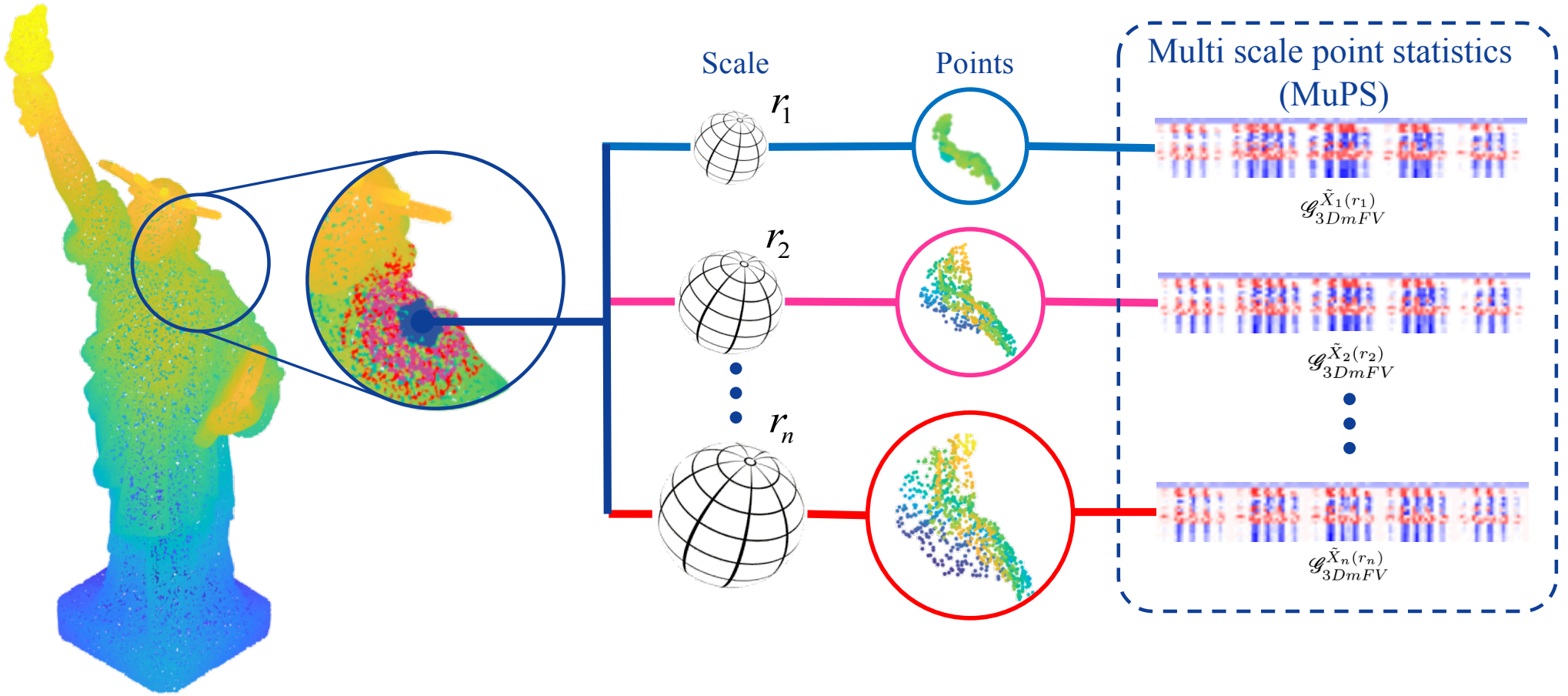
Extract subset



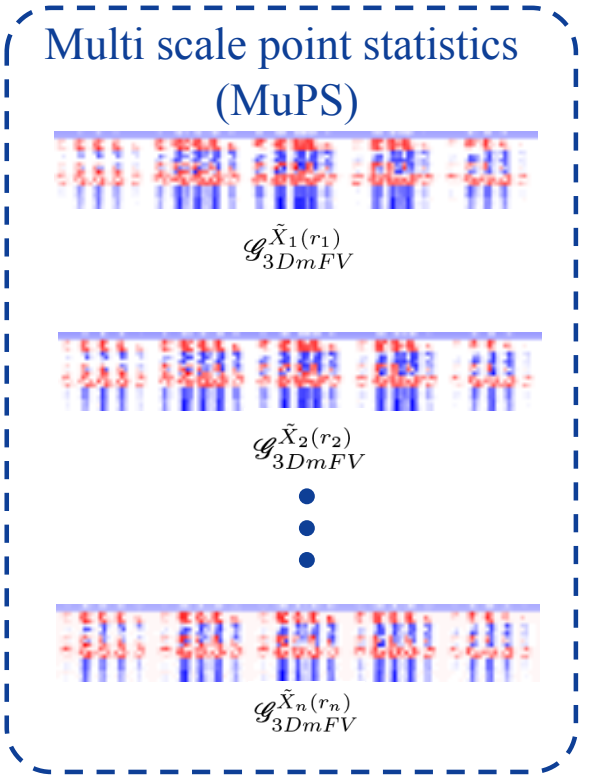
Fit a surface



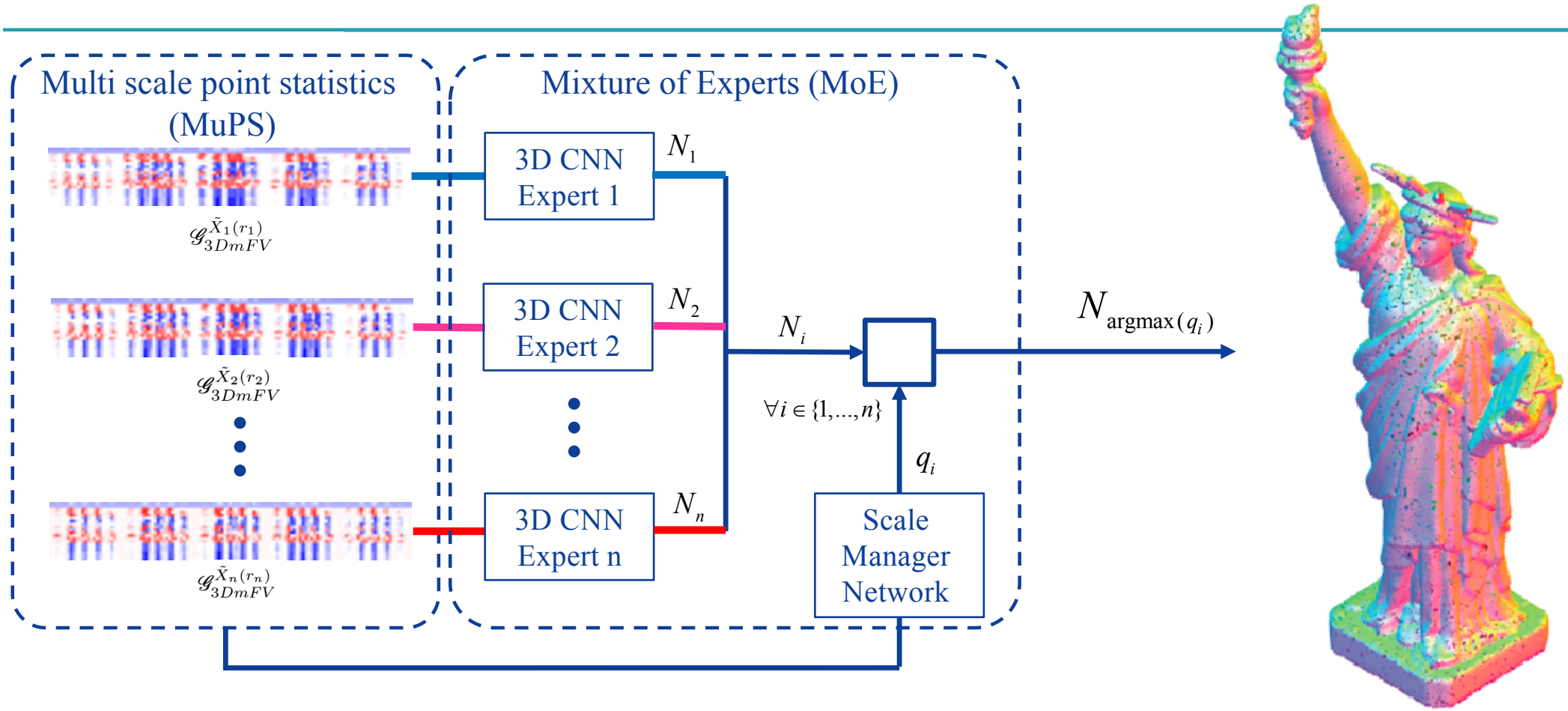
Nesti-Net pipeline



Nesti-Net pipeline

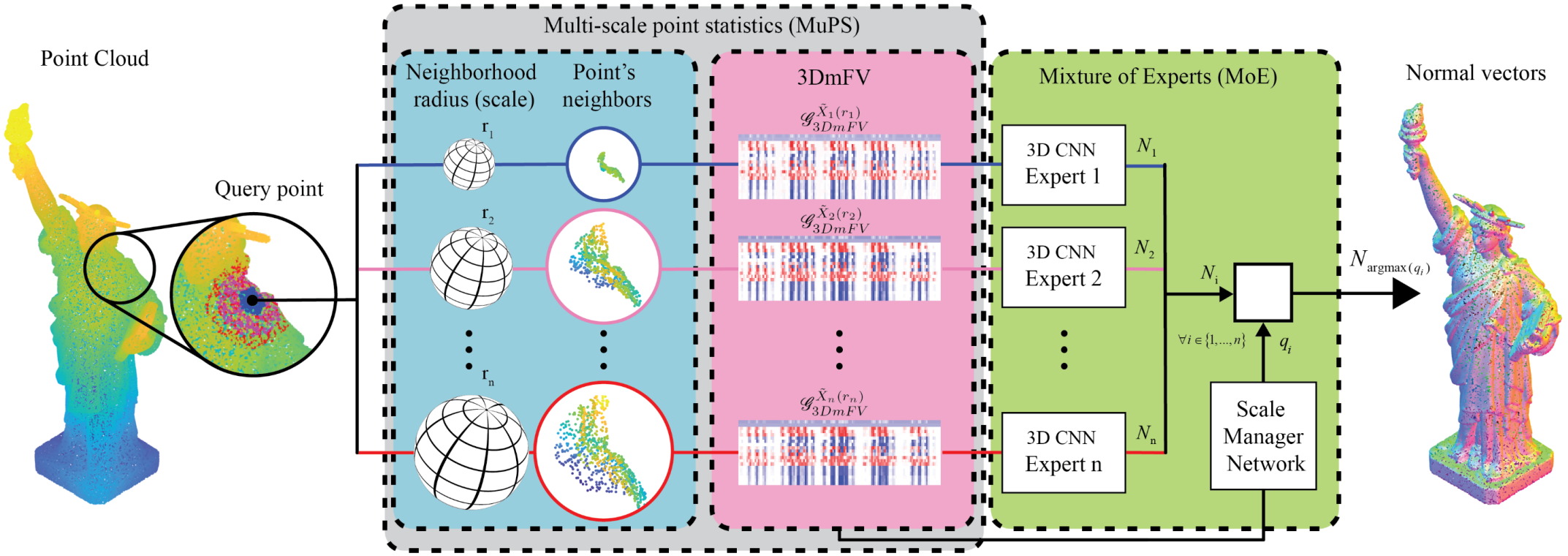


Nesti-Net pipeline



MoE: R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. Neural Computation, 3(1):79-87, 1991.

Nesti-Net pipeline



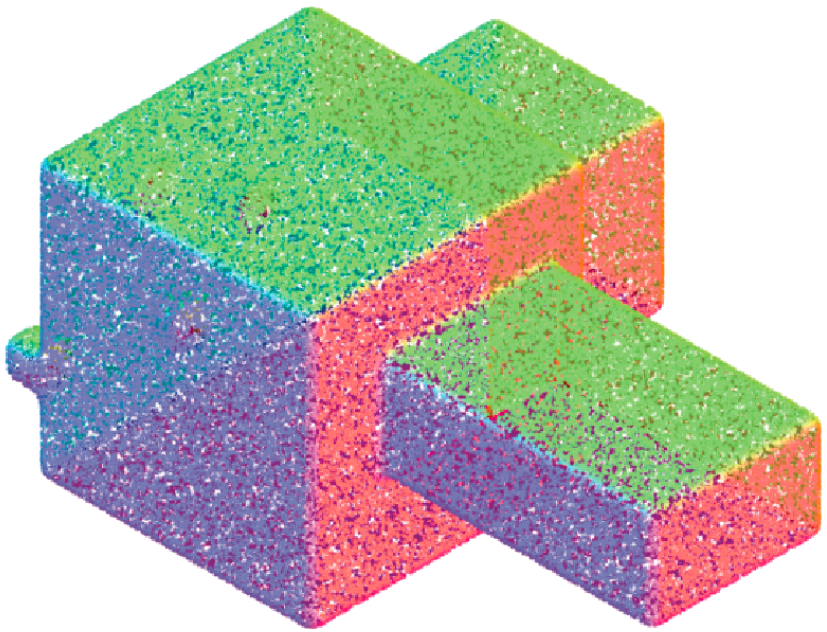
Loss: $$L_{MoE} = \sum_{i=1}^{n} q_i D_N = \sum_{i=1}^{n} q_i \frac{\|N_i \times N_{GT}\|}{\|N_i\| \cdot \|N_{GT}\|}$$

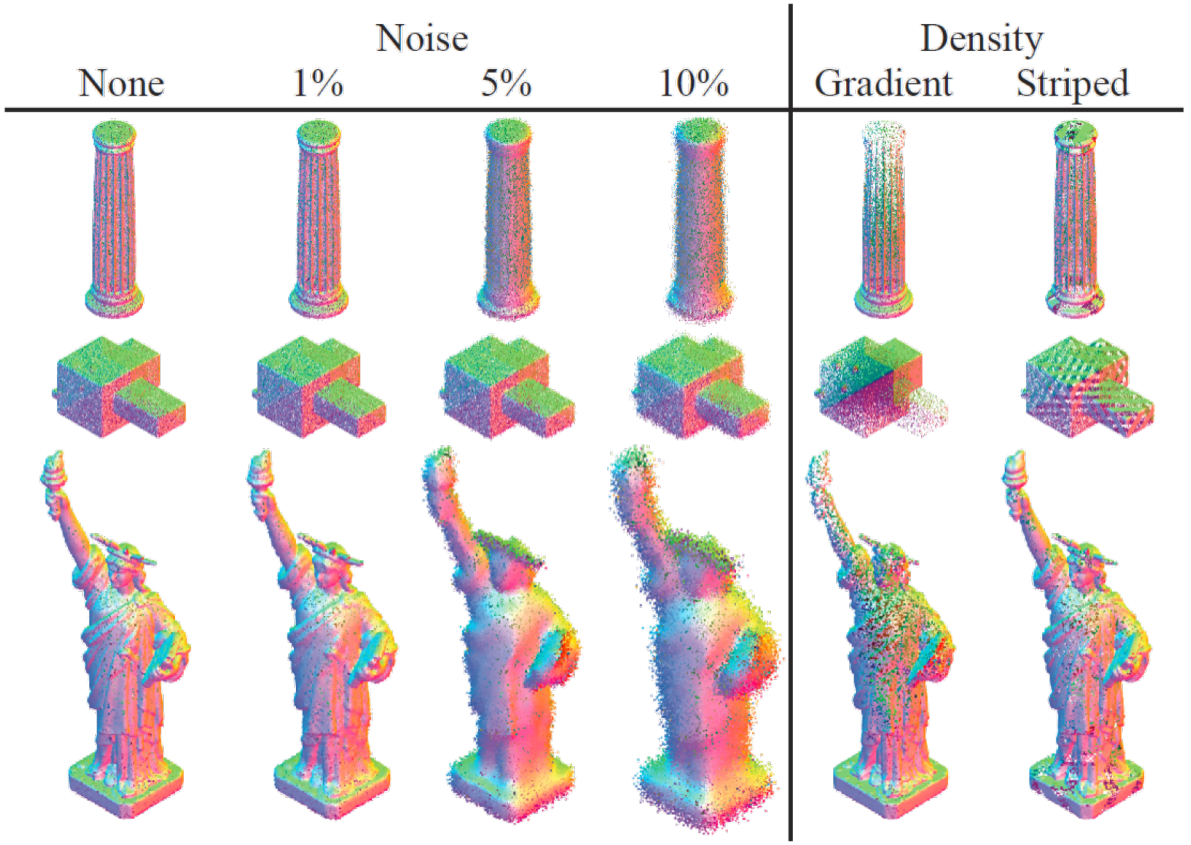
Quantitative results

Aug.	**Our Nesti-Net**	**PCA** [12]			**Jet** [7]			**PCPNet** [11]		**HoughCNN** [6]	
scale		small	med	large	small	med	large	ss	ms	ss	ms
None	**6.99**	8.31	12.29	16.77	7.60	12.35	17.35	9.68	9.62	10.23	10.02
Noise											
$\sigma = 0.00125$	**10.11**	12.00	12.87	16.87	12.36	12.84	17.42	11.46	11.37	11.62	11.51
$\sigma = 0.006$	**17.63**	40.36	18.38	18.94	41.39	18.33	18.85	18.26	18.87	22.66	23.36
$\sigma = 0.012$	**22.28**	52.63	27.5	23.5	53.21	27.68	23.41	22.8	23.28	33.39	36.7
Density											
Gradient	9.00	9.14	12.81	17.26	**8.49**	13.13	17.8	13.42	11.7	11.02	10.67
Stripes	**8.47**	9.42	13.66	19.87	8.61	13.39	19.29	11.74	11.16	12.47	11.95
average	**12.41**	21.97	16.25	18.87	21.95	16.29	19.02	14.56	14.34	16.9	17.37

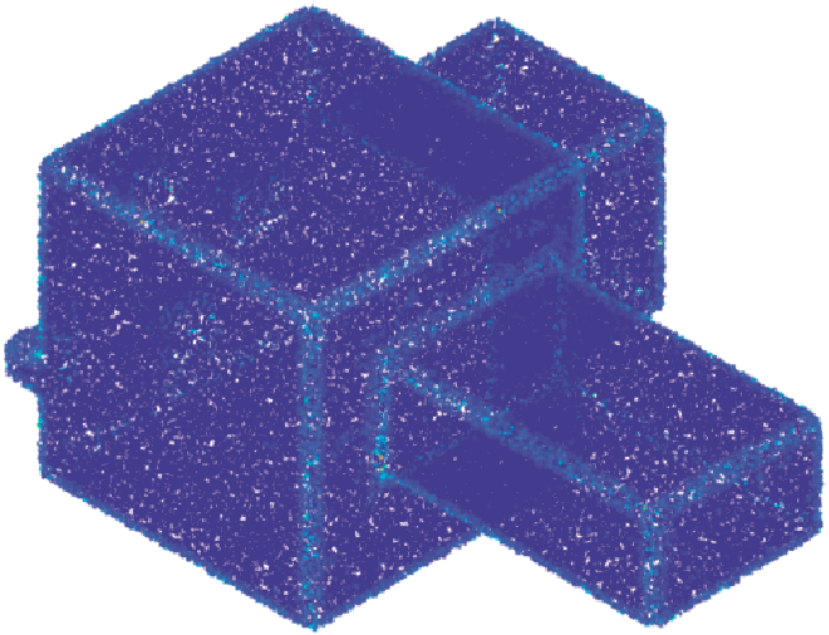
Table 1. Comparison of the RMS angle error for unoriented normal vector estimation of our Nesti-Net method to classic geometric methods (PCA [12] , Jet [7]) with three scales, and deep learning methods (PCPNet [11], HoughCNN [6])

Qualitative results

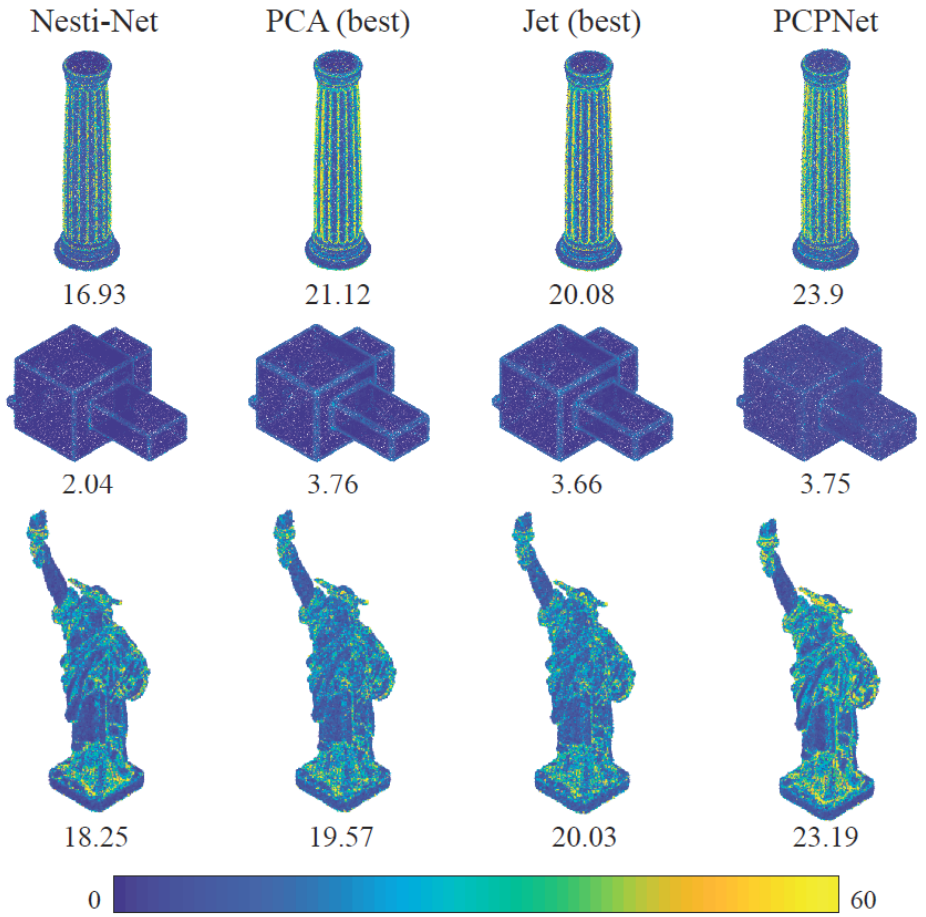




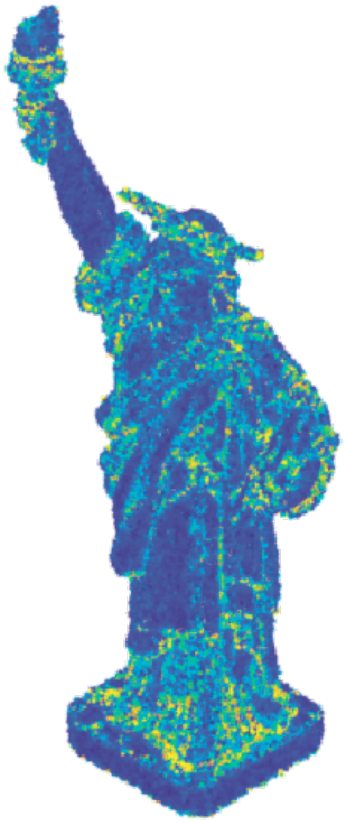
Error visualization



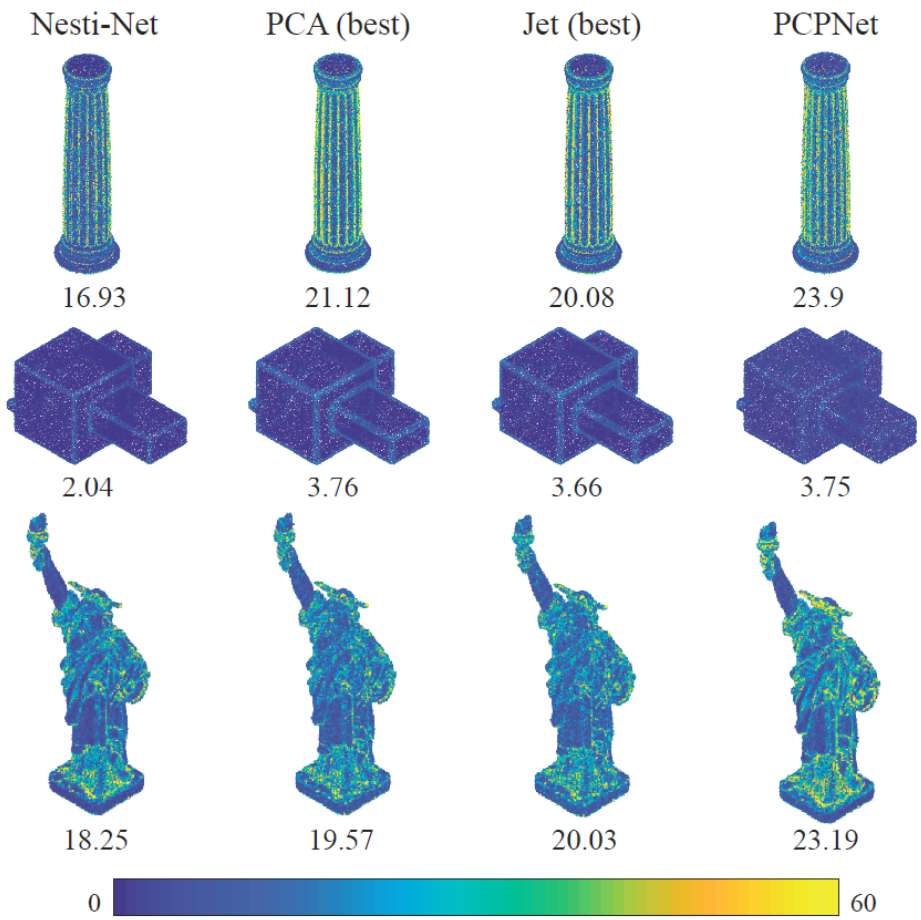
2.04



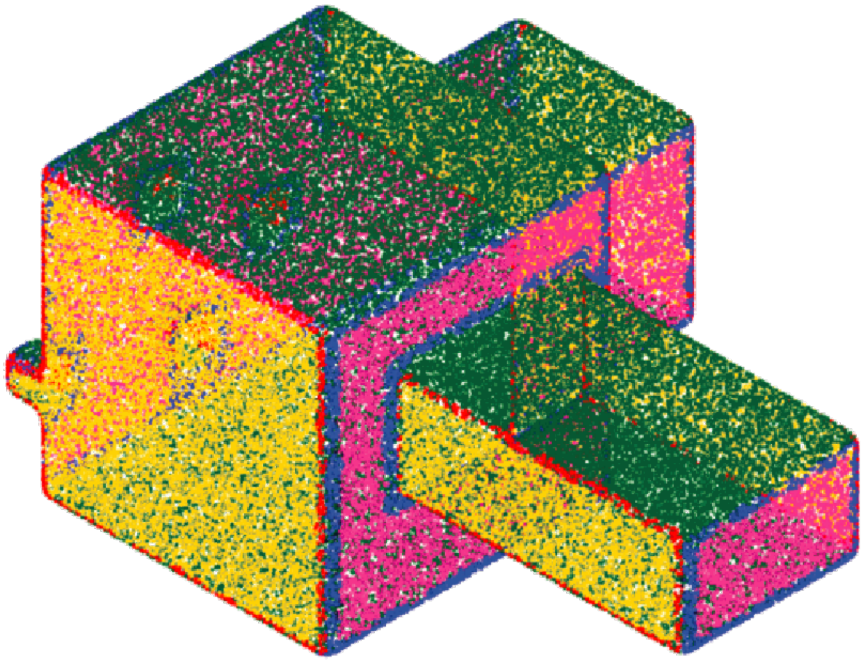
Error visualization

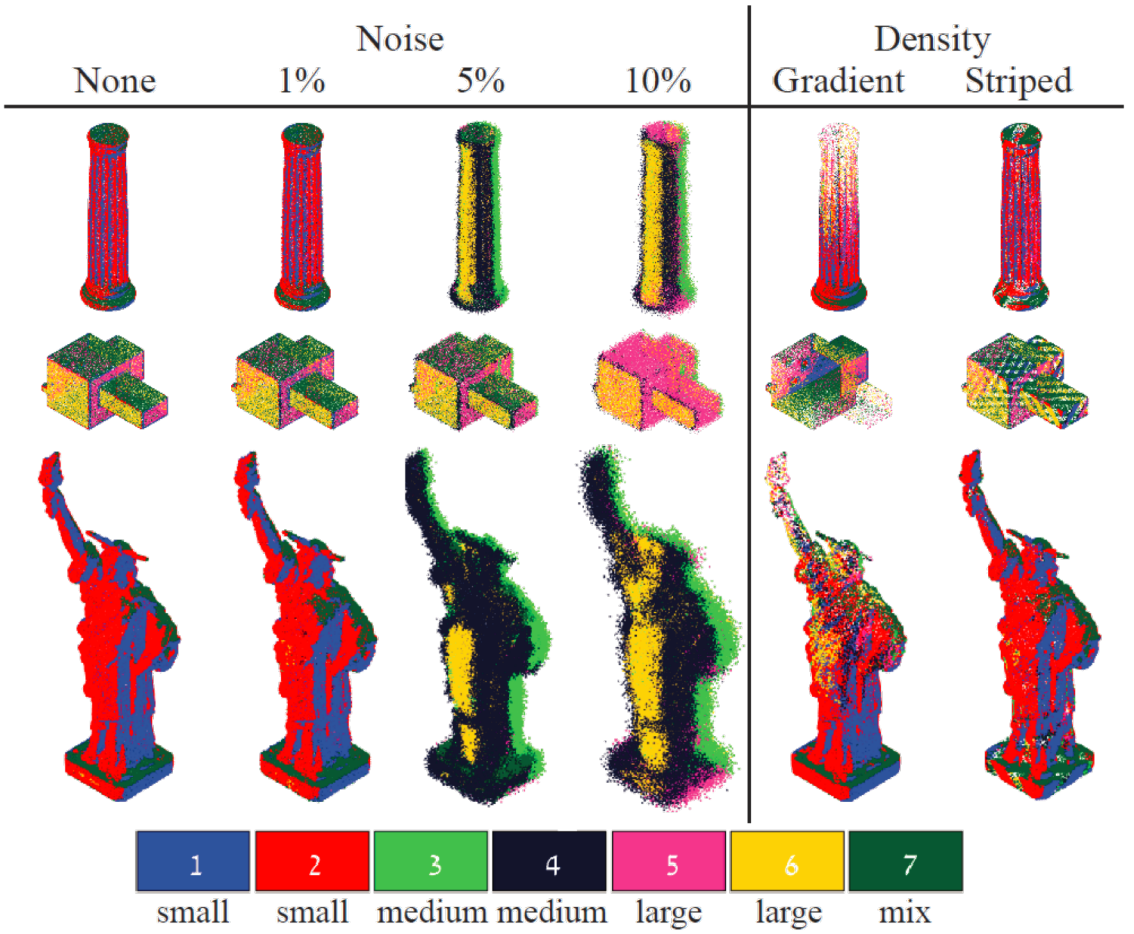


18.25

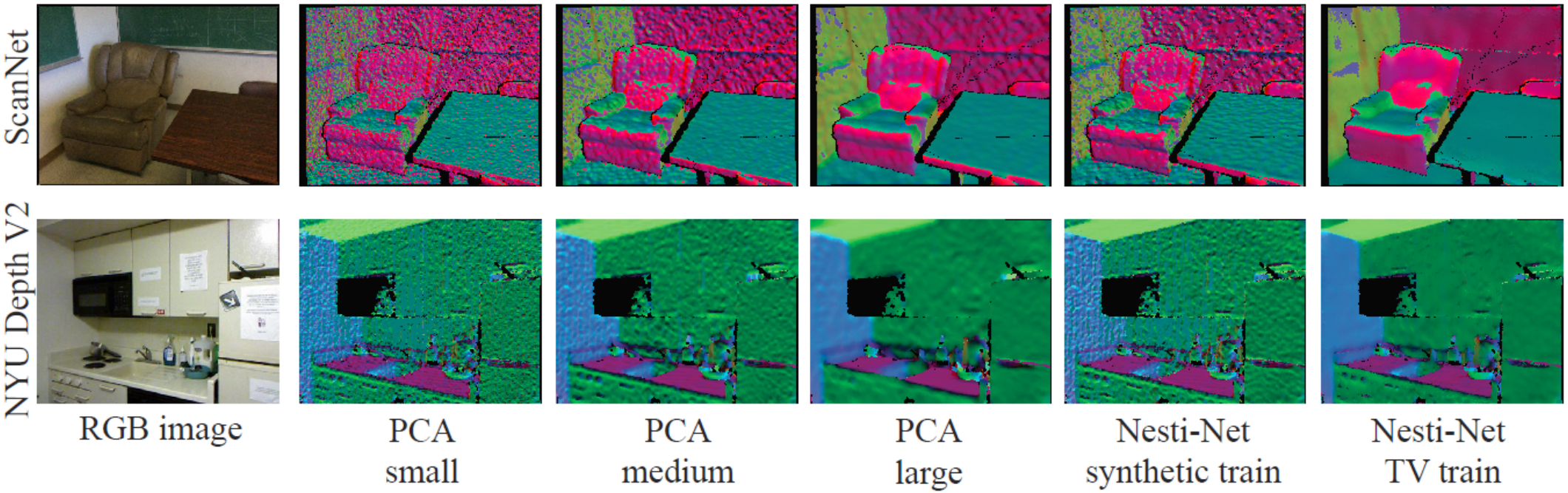


Scale prediction results





Normal estimation results on scanned data



Ben-Shabat, Y., Lindenbaum, M. and Fischer, A., Normal Estimation for Unstructured 3D Point Clouds using Convolutional Neural Networks. CVPR. 2019.

Summary

- We introduce a **new hybrid representation** for 3D point clouds (3DmFV) which is structured, order and sample size independent. It **enables the use of CNNs** with point cloud data.
- 3DmFV offers an efficient way for **encoding global and local spatial distributions.**
- We design a **new deep CNN architecture (3DmFVNet)** based on this representation and use it for point cloud **classification**, obtaining state of the art results in **real-time**.
- We extend the 3DmFV-Net to **part segmentation** of point clouds and to **Normal Estimation**.
- Note: These best results are obtained **without "end to end"** training.

Questions ?

For code and tutorials visit www.itzikbs.com