Point Cloud Classification Using Deep Neural Networks

Federica Di Lauro (829470), Andrea Premate (829777), Lidia Lucrezia Tonelli (813114)

Presentation Outline

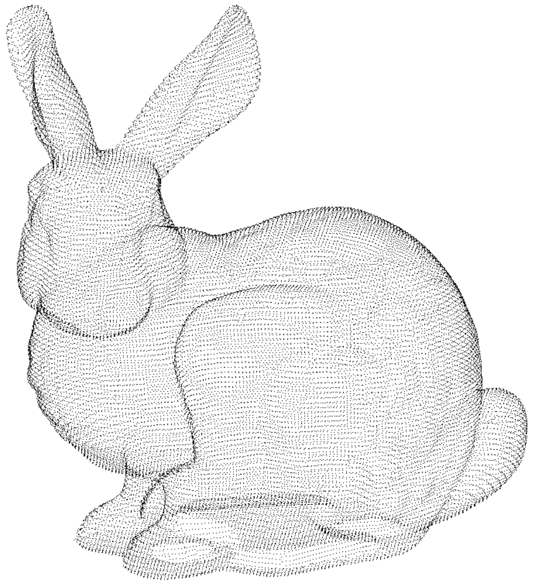
- **Introduction to Point Clouds**
- **Classification Methods**
  - Projection-based
  - Point-based
- **Comparison**
- **Conclusion**

Introduction to Point Clouds

Point Clouds → **Datasets** → **Metrics**

Point Clouds

- **3D data** acquired by sensors can make machines able to understand the surrounding environment
- **Areas of application:** autonomous driving, robotics, remote sensing, medical treatment etc. [1]
- **Point cloud understanding tasks:**
  - Classification
  - Segmentation
  - Object Detection
  - Tracking, flow estimation, matching and registration, reconstruction …
- **Points in a point cloud have Cartesian coordinates (x,y,z)**



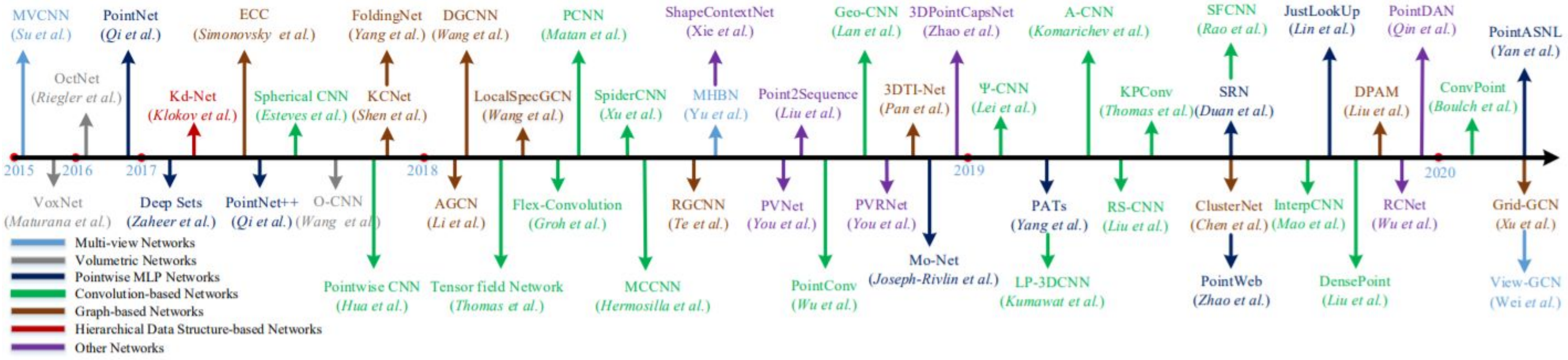
Point Clouds

- **Problems in point cloud understanding [2]:**
  - Find a dense representation from a sparse point cloud
  - Build a network satisfying size-invariance and permutation-invariance
  - Process large volumes of data in low time and computational resources

Point Clouds

Classification task

- **Classification of 3D shapes** is the first task described in literature
- Similar to image classification
- Methods learn the embedding of each point and then extract a global embedding
- **Projection-based methods** and **point-based methods**



Datasets

- 3D datasets are usually **smaller** than 2D image datasets
- 3D data acquired by sensors such as **LiDARs** and **RGB-D** cameras, complemented with 2D images
- For classification datasets can be **synthetic** (no occlusions, no background) or **real-word** (occlusions at different levels, background noise)
- Synthetic: **ModelNet40, ShapeNet,** real-word: **KITTI, S3DIS**

Point Clouds

ModelNet40 [3]

- 3D computer **graphics CAD models**: 151,128 models in 660 unique classes
- Downloaded models were labelled with Amazon Mechanical Turk and then manually checked
- ModelNet has 3 benchmarks: ModelNet10, ModelNet40, Aligned40
- Used to evaluate capacity of backbones before applying networks on more complicated tasks: usually point clouds are created sampling points on CAD models

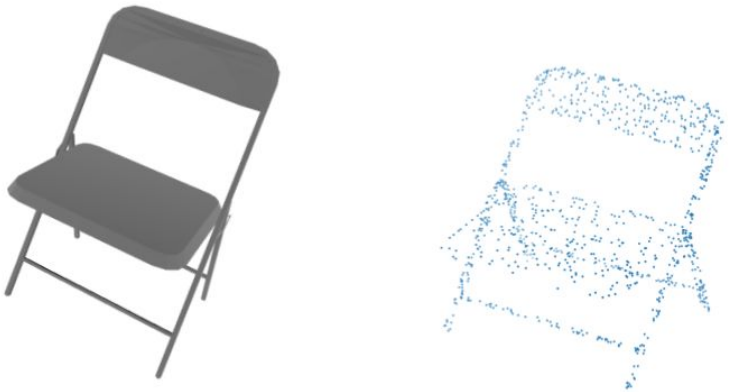
Point Clouds

ModelNet40

Left: word cloud visualization of ModelNet dataset. Right: examples of 3D chair models

Point Clouds

ModelNet40



Left: CAD model from ModelNet40. Right: point cloud sampled randomly from the CAD model, 1024 points used.

Metrics

- For 3D shape classification metrics are the **Overall Accuracy (O.A.)** and the **Mean Class Accuracy (mACC)** ([1], [2])

$$O.A. = \frac{TP+TN}{|dataset|} = \frac{TP+TN}{TP+TN+FP+FN}$$

$$mACC = \frac{1}{C}\sum_{c=1}^{C} Accuracy_c$$

where C is the number of classes in the dataset.

- Visualize the results is always recommended

Classification Methods

Projection-based

Point-based

Projection-based methods

- Methods that project unstructured 3D point cloud into specific presupposed modality and extract features from the target format
- PRO: benefit from previous research finding in corresponding direction (computer vision, graphics …)
- CON: while projecting in the target format we have information loss

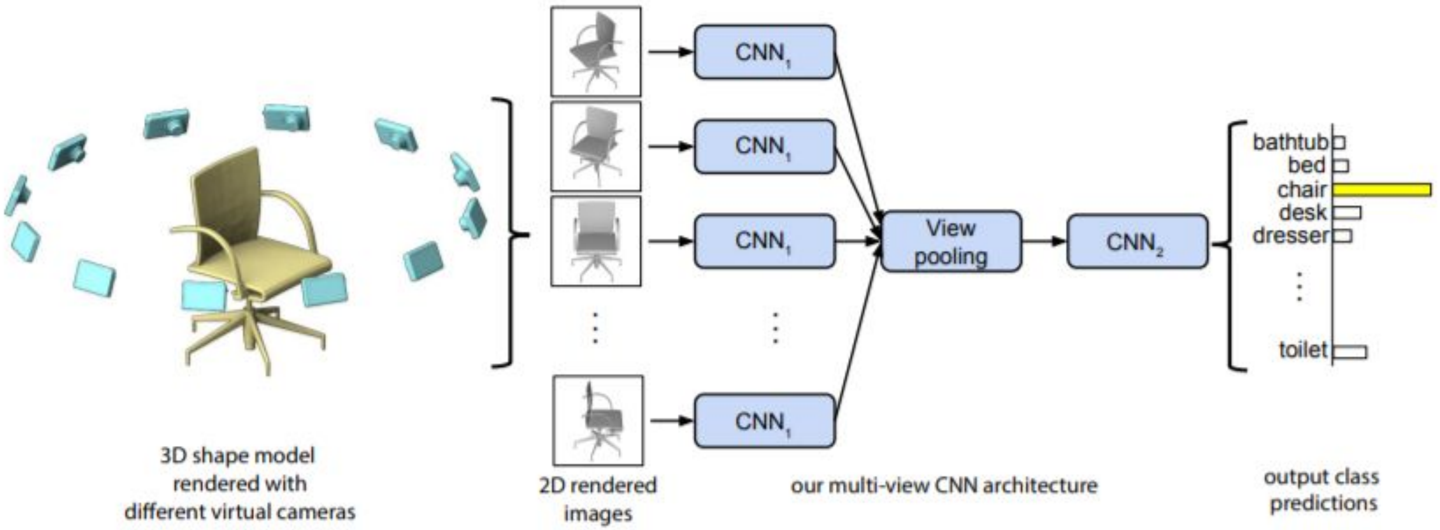
Multi-View

Volumetric

Projection-based methods

Multi-View [6][7]

- Main idea: train a model starting from **different 2D views of the 3D point cloud**. Since this approach works on images, we can take advantage of CNNs



Projection-based methods

Multi-View - Input

- We need to **connect cloud points**, obtaining edges forming faces and apply a reflection model (Phong). Next we **apply perspective projection**, obtaining some views of the 3D point cloud, as 2D images
- **Camera setup alternatives**:

  1) We assume input is upright oriented. 12 cameras, each every 30 degrees, elevated 30 degrees from the ground plane pointing towards the centroid of the mesh

  2) No assumptions are made about orientation. 20 cameras are placed at the 20 vertices of a dodecahedron enclosing the shape. For each camera 4 different rotated views are taken (on axis passing through the camera and the object centroid)

Projection-based methods

Multi-View - Architecture

- **Naive approach:**

  1) Fine-tune a pre-trained CNN using the different views, obtaining a descriptor for each view

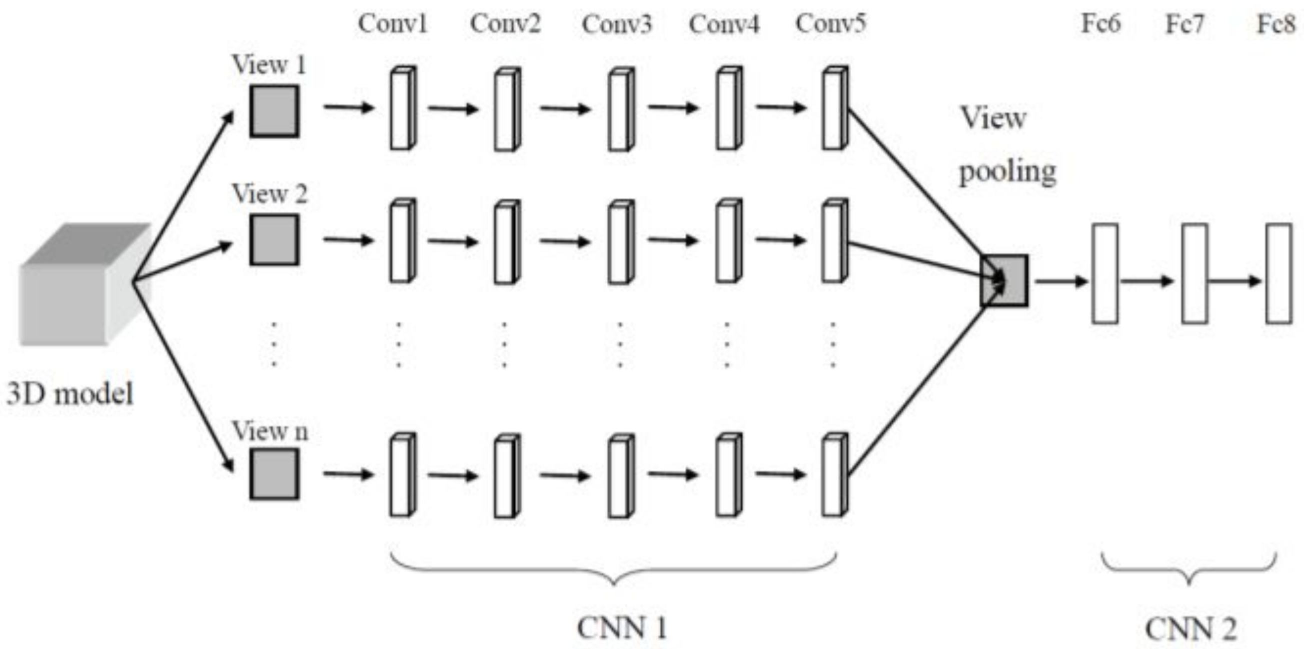
  2) Train one-vs-rest linear SVMs to classify shapes using their image features

  3) At test time sum up SVMs decision values over all 12 views and return the class with the highest sum

Projection-based methods

Multi-View - Architecture

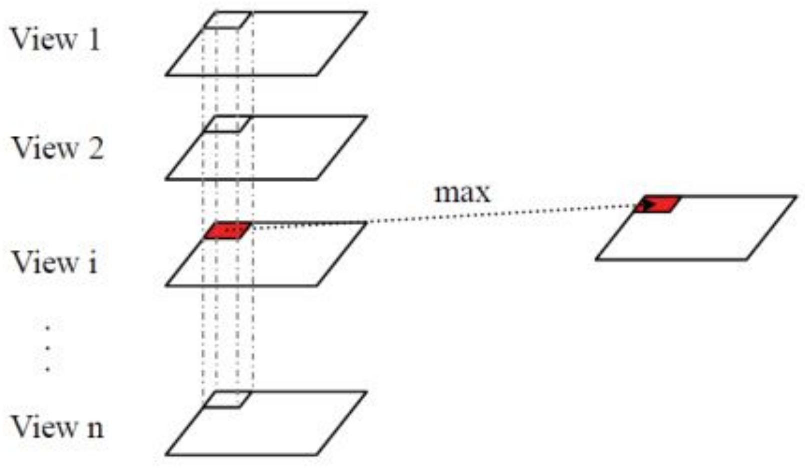
- **MVCNN approach** (learning to aggregate views)



Projection-based methods

Multi-View - Architecture

- All branches in the first part of the network **share the same parameters** in CNN1
- The **view-pooling** layer uses element-wise max pooling strategy to combine the discriminative information of multiple views and increase the computational efficiency. The viewpooling method is similar to the traditional max-pooling operation used in CNNs



Projection-based methods

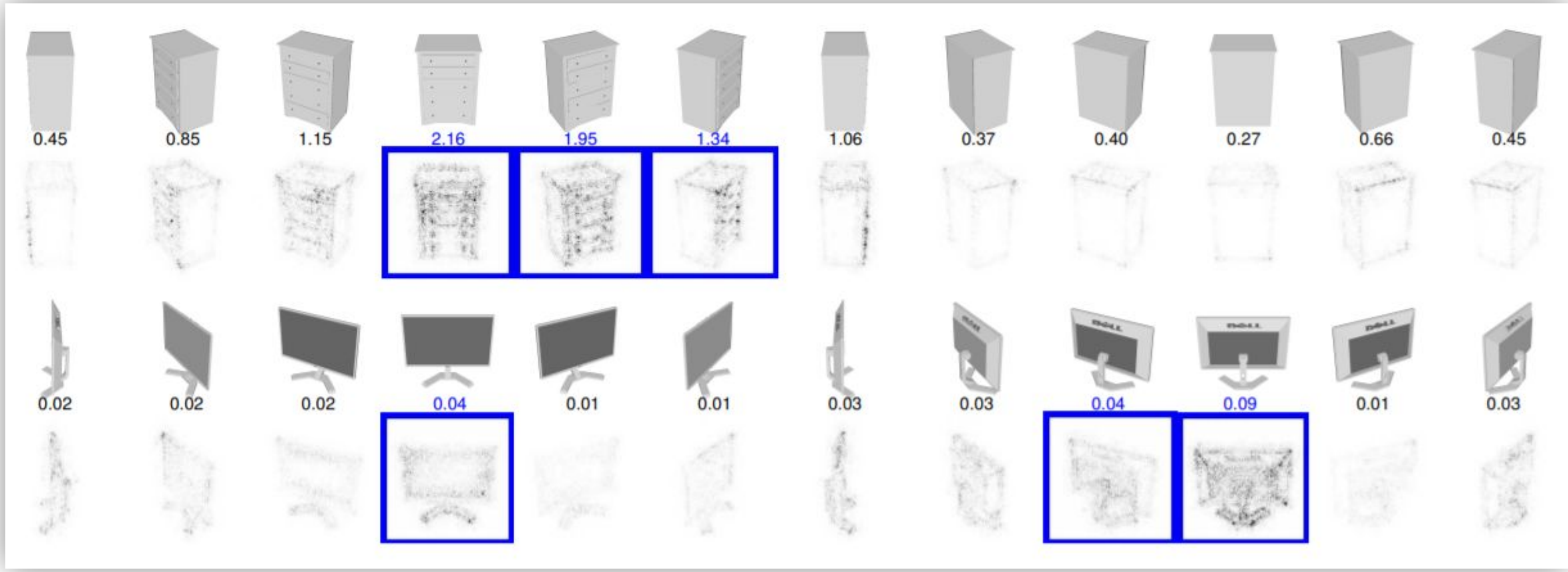
Multi-View - Saliency map among views

- It is possible to trace back to the influence of the different views on the MVCNN output score F_c for its ground truth class c. For each 3D shape S and its relative views {I1, I2, ...IK} we can compute w of the following equation using backpropagation and obtain **saliency maps** for individual views

$$[w_1, w_2, \dots w_K] = \left[\frac{\partial F_c}{\partial I_1}\Big|_S, \frac{\partial F_c}{\partial I_2}\Big|_S, \dots \frac{\partial F_c}{\partial I_K}\Big|_S \right]$$

Projection-based methods

Multi-View - Saliency map among views



Projection-based methods

Multi-View - Results

- Classification performance on **ModelNet40** dataset

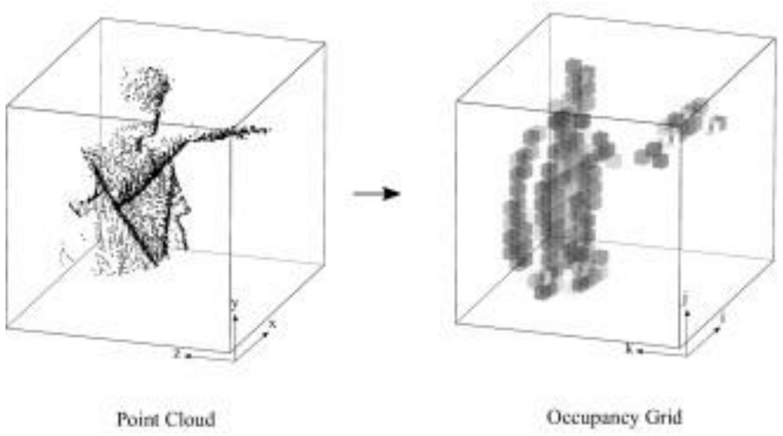
Method	Training Config. Pre-train	Fine-tune	# Views	Test Config. # Views	Classification (O.A.)
(1) FV	-	ModelNet40	12	1	78.8 %
(2) FV, 12×	-	ModelNet40	12	12	84.8 %
(3) CNN	ImageNet1K	-	-	1	83.0 %
(4) CNN, f.t.	ImageNet1K	ModelNet40	12	1	85.1 %
(5) CNN, 12×	ImageNet1K	-	-	12	87.5 %
(6) CNN, f.t., 12×	ImageNet1K	ModelNet40	12	12	88.6 %
(7) MVCNN, 12×	ImageNet1K	-	-	12	88.1 %
(8) MVCNN, f.t., 12×	ImageNet1K	ModelNet40	12	12	89.9 %
(9) MVCNN, f.t. + metric, 12×	ImageNet1K	ModelNet40	12	12	89.5 %
(10) MVCNN, 80×	ImageNet1K	-	80	80	84.3 %
(11) MVCNN, f.t., 80×	ImageNet1K	ModelNet40	80	80	**90.1 %**
(12) MVCNN, f.t. + metric, 80×	ImageNet1K	ModelNet40	80	80	**90.1 %**

Table 1: Classification results. FV is another simpler approach described in the same paper that describes MVCNNs [8] based on Fisher Vectors. f.t. = fine tuning, metric = low-rank Mahalanobis metric learning

Projection-based methods

Volumetric [8]

- Main idea: train a model starting from the **volumetric occupancy grid** obtained from the point cloud, thus maintaining a three-dimensional structure



Projection-based methods

Volumetric - Input

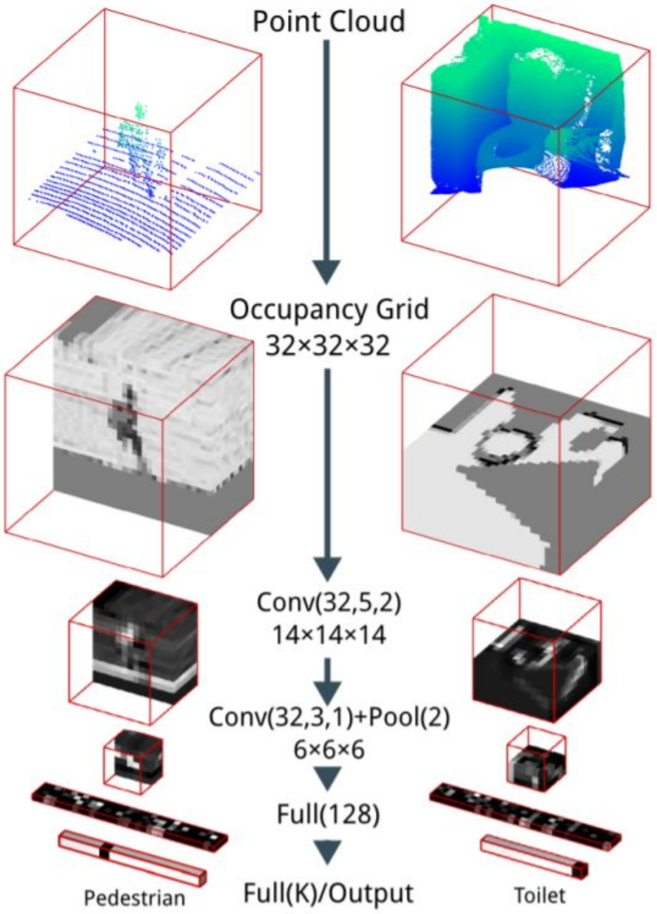
- Each point (x,y,z) of the point cloud is mapped to discrete voxel coordinates (i,j,k)
- This mapping depends on:
  1) **Grid model and resolution** (in experiments fixed to 32x32x32)
  2) **Origin**. It is supposed to be given as input
  3) **Orientation**. It is assumed that z axis is aligned with the gravity direction

  There is still a degree of freedom we have to deal with: the rotation around the z axis. SOLUTION: create n copies of each input instance, each rotated 360°/n around the z axis. At testing time we pool the activations of the output layer over all n copies
- Voxel entries are supposed to be normalized in the range (-1, 1)

Projection-based methods

Volumetric - Architecture

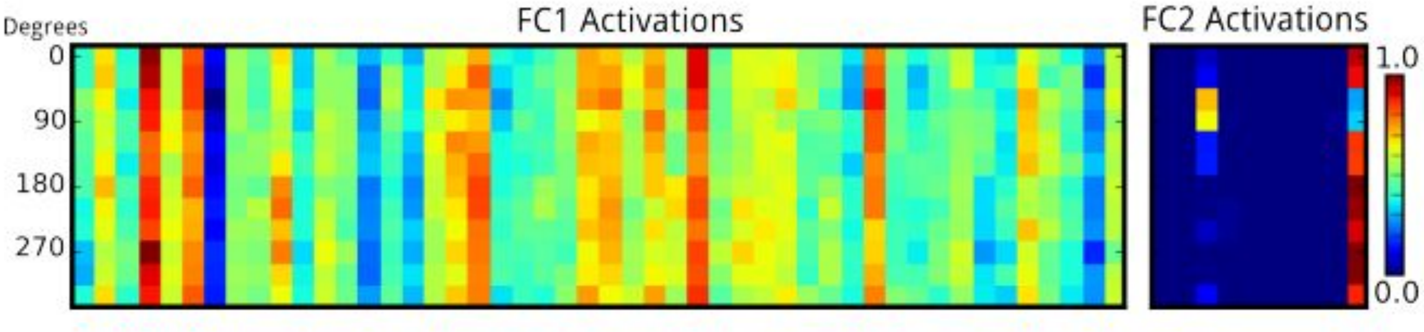
- **VoxNet architecture:**
  C(32,5,2) - C(32,3,1) - P(2) - FC(128) - FC(K),
  where K is the number of classes
- 921736 parameters, much less than a typical CNN. A
  simpler task? we don’t have perspective,
  illumination, …
- Option: **multi-resolution approach**. We use a
  network for each resolution grid and merge
  information at FC layers



Projection-based methods

Volumetric - Rotational invariance

- Qualitative result: neuron activation in FC layers are very similar for the same object rotated around z axis -> **approximate rotational invariance**



Projection-based methods

Volumetric - Results

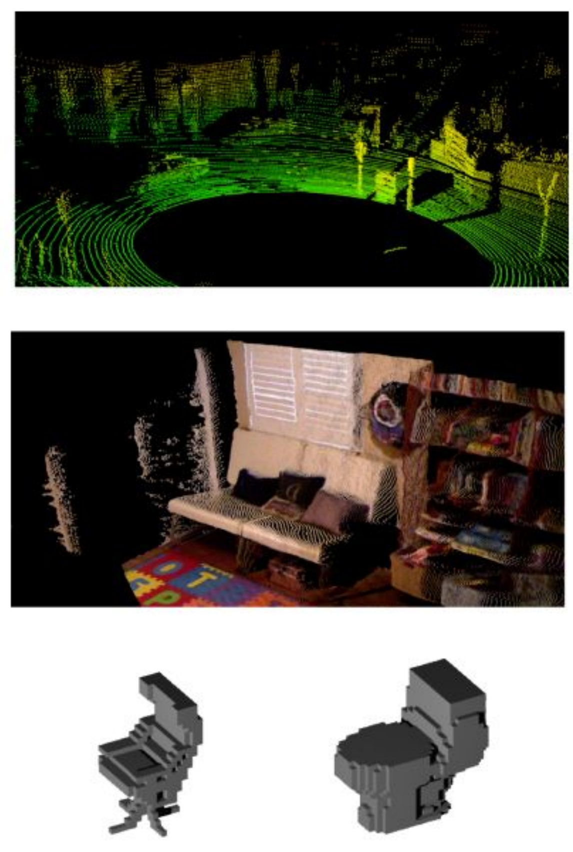
- VoxNet evaluated in **three different domains:** LiDAR point cloud, RGB-D point clouds and CAD models

EFFECT OF OCCUPANCY GRIDS

Occupancy	Sydney F1	NYUv2 Acc
Density grid	**0.72**	**0.71**
Binary grid	0.71	0.69
Hit grid	0.70	0.70

EFFECTS OF ROTATION AUGMENTATION AND VOTING

Training Augm.	Test Voting	Sydney F1	ModelNet40 Acc
Yes	Yes	**0.72**	**0.83**
Yes	No	0.71	0.82
No	Yes	0.69	0.69
No	No	0.69	0.61



Point-based methods

- The transformations of data made by projection-based methods make data voluminous and introduce quantization artifacts
- Point-based methods **learn features directly from the points** and not from their spatial arrangement

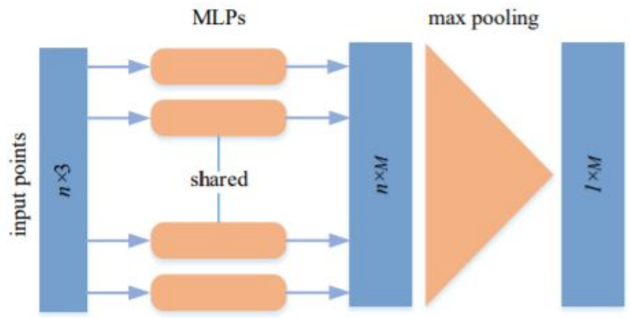
MLP

CNN

Graph inspired networks

Point-based methods

Multi-Layer Perceptron - PointNet [4]

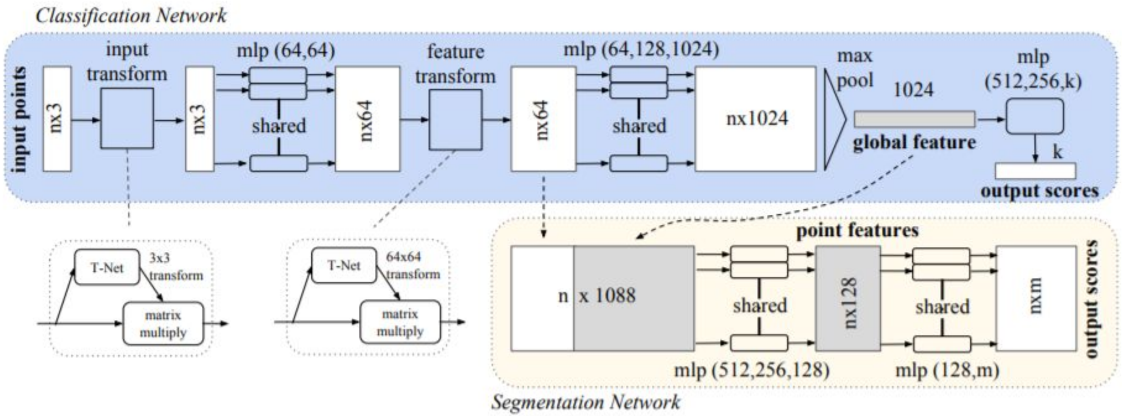


- Independent MLPs **extract features from each point** of point clouds, then features are **aggregated with a symmetric function** (needed for permutation invariance)
- PointNet is one of the early deep network which directly consumes point clouds
- Used for classification and segmentation
- Properties of a subset of points in Euclidean space:
  - **Unordered:** networks have to be invariant to n! permutations
  - **Points interact with each other:** for the properties of the metric space and its distance
  - **Invariant under transformations:** transformations don't change the class

Point-based methods

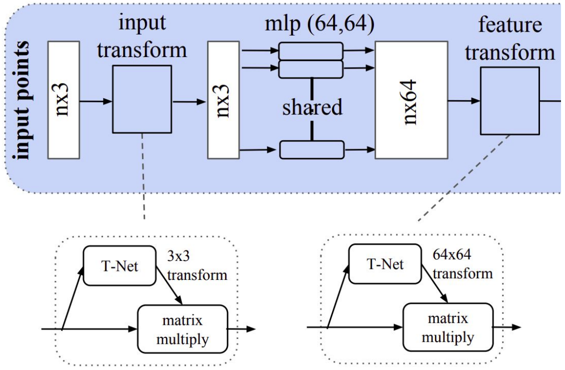
Multi-Layer Perceptron - PointNet [4]

- **Input:** set of n points
- Points pass through many layers
- Features are aggregated with **max pooling**
- **Global feature** tensor is passed through an MLP
- **Output:** k scores, one for each class



Point-based methods

Multi-Layer Perceptron - PointNet [4]



- **Joint Alignment Network:** achieve invariance to rigid geometric transformations
- Simple solution would be set models to a canonical space
- PointNet uses **T-net:** point independent feature extractions + max pooling + fully connected layers
- T-net predicts an affine transformation matrix: **pose normalization**
- A regularization term constrains the matrix to be orthogonal
- This reduces the need for data augmentation

Point-based methods

Multi-Layer Perceptron - PointNet [4]

- **Symmetric Function** for Unordered Input: achieve permutation invariance
- PointNet uses a symmetric function that produced an order-invariant vector
- Idea: a set function is approximated by a function applied on transformed points

$$f(\{x_1, \ldots, x_n\}) \approx g(h(x_1), \ldots, h(x_n))$$

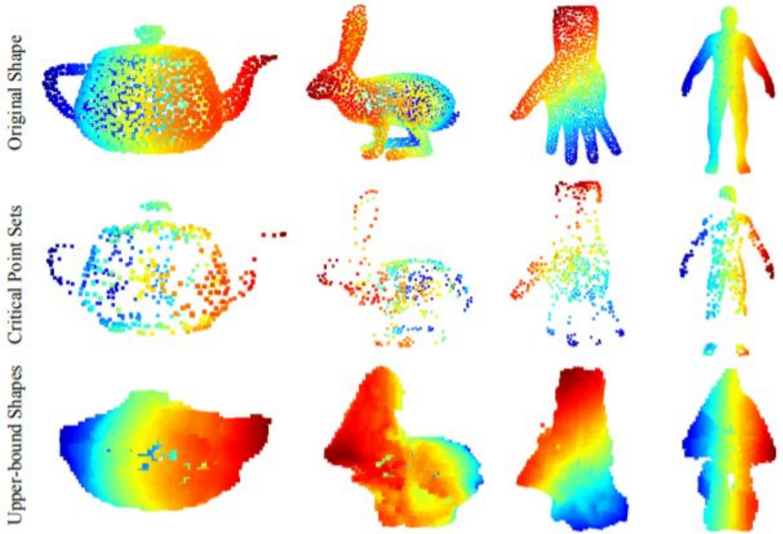
where h is an MLP and g the max pooling.

- The approximated functions (one for each class) are the **global signature** of the point cloud, invariant to permutation

Point-based methods

Multi-Layer Perceptron - PointNet [4]

- For every point cloud there exist a **critical point set** and an **upper-bound shape**
- Between these sets the features extracted by PointNet are the same
- **Robustness** of the network w.r.t. perturbation, corruption and extra noise points



Point-based methods

Multi-Layer Perceptron - PointNet [4]

- Validation experiments on **ModelNet40: 1024 points sampled from each CAD model**
- Normalization in a unit sphere and augmentation with rotation and Gaussian noise
- **mACC = 86.2%, O.A. = 89.2 %**
- Experiments were conducted on **T-net application**
- 3.5 millions parameters, 440 millions FLOPs/sample, linear space and time complexity w.r.t. the number of input points

Transform	**Accuracy (%)**
none	87.1
input (3 x 3)	87.9
feature (64 x 64)	86.9
feature (64 x 64) + reg.	87.4
both	89.2

Point-based methods

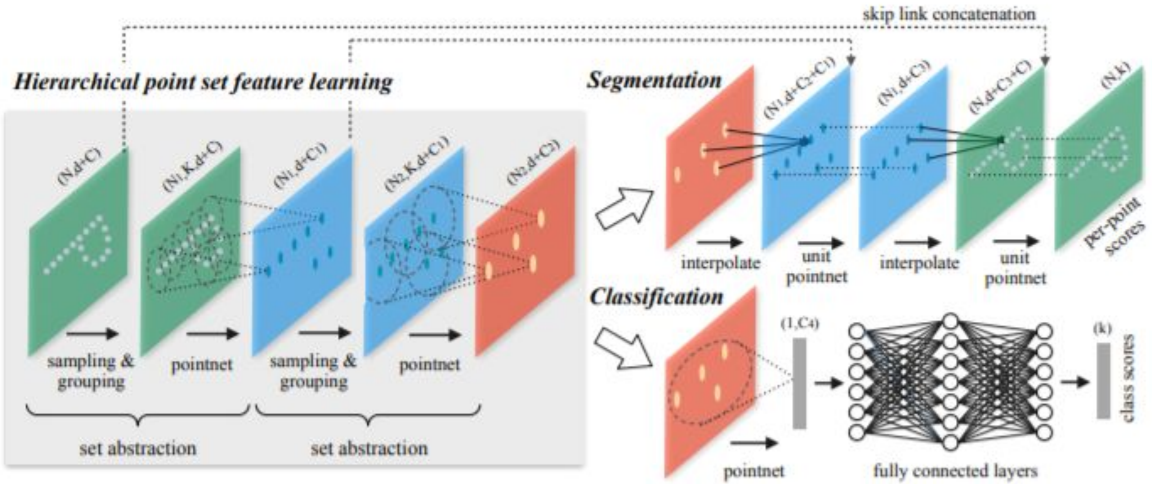
Multi-Layer Perceptron - PointNet++ [5]

- **Limitation** of PointNet: unable to capture local and fine geometric structures from the neighborhoods
- Inspired by CNNs, PointNet++ captures features at increasingly large scales (**hierarchical network**)
- Point clouds are divided in partitions (overlapping local regions)
- The **Local Feature Learner** is PointNet
- Weights of local features are shared across the partitions

Point-based methods

Multi-Layer Perceptron - PointNet++ [5]

- PointNet++ is composed by **Set Abstraction Levels** that produce smaller and smaller features
- Set abstraction = **Sampling + Grouping + PointNet** layers



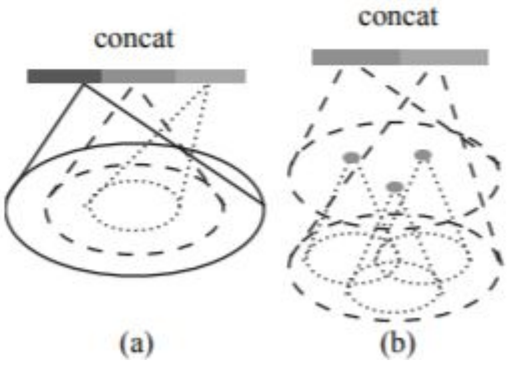
Point-based methods

Multi-Layer Perceptron - PointNet++ [5]

- **Sampling layer:**
  - Selects centroids from the input sets with FPS (Iterative Farthest Point Sampling)
  - A centroid is chosen if it's the most distant point w.r.t. the metric from all the other centroids previously chosen
- **Grouping layer:**
  - Builds local regions
  - Neighborhood is defined with a radius (possible use of different distances) or with k-NN search
- **PointNet layer:**
  - Extracts feature vectors from local regions (coordinates of points are translated in a local frame relative to the centroid)
  - Global feature vectors have fixed length even with different inputs
  - Each local region is abstracted by a centroid and a local feature that encodes the centroid's neighborhood

Point-based methods

Multi-Layer Perceptron - PointNet++ [5]

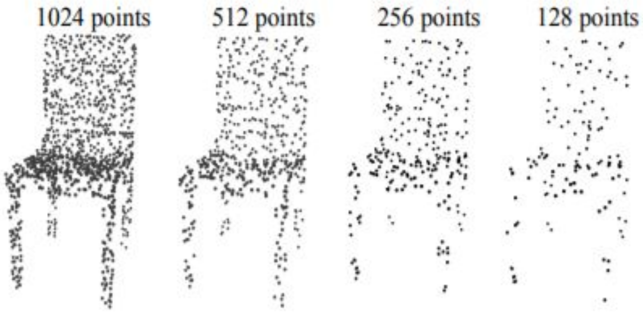


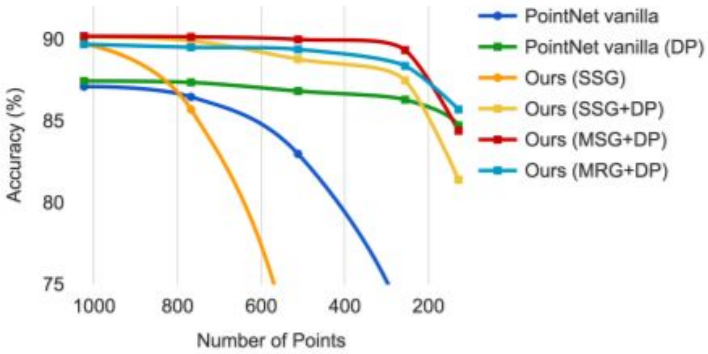
- Robust Feature Learning under Non-Uniform Sampling Density: PointNet++ adds some **density adaptive** PointNet layers
- These layers combine features from different scale regions when the density is different
- Two types: **MSG** and **MRG**
- MSG (Multi-scale grouping, fig. (a)):
  - Extract features at different scale and concatenate them in a multi-scale feature vector
  - Computationally expensive
- MRG (Multi-resolution grouping, fig. (b)):
  - At each layer builds a concatenation of two vectors
  - The former contains the features of every subregion at the previous layer
  - The latter contains the features of the local region at the current layer

Point-based methods

Multi-Layer Perceptron - PointNet++ [5]

- Validation experiments on different datasets, including ModelNet40
- Normalization with zero mean within a unit ball
- Architecture evaluated has 3 hierarchical levels and 3 FC layers
- **O.A. = 91.9 %**
- More robust to **density variation** w.r.t. PointNet





Point-based methods

Convolutional Neural Networks - PointConv [9]

$$\text{PointConv}(S, W, F)_{xyz} = \sum_{(\delta_x, \delta_y, \delta_z) \in G} S(\delta_x, \delta_y, \delta_z) W(\delta_x, \delta_y, \delta_z) F(x + \delta_x, y + \delta_y, z + \delta_z)$$

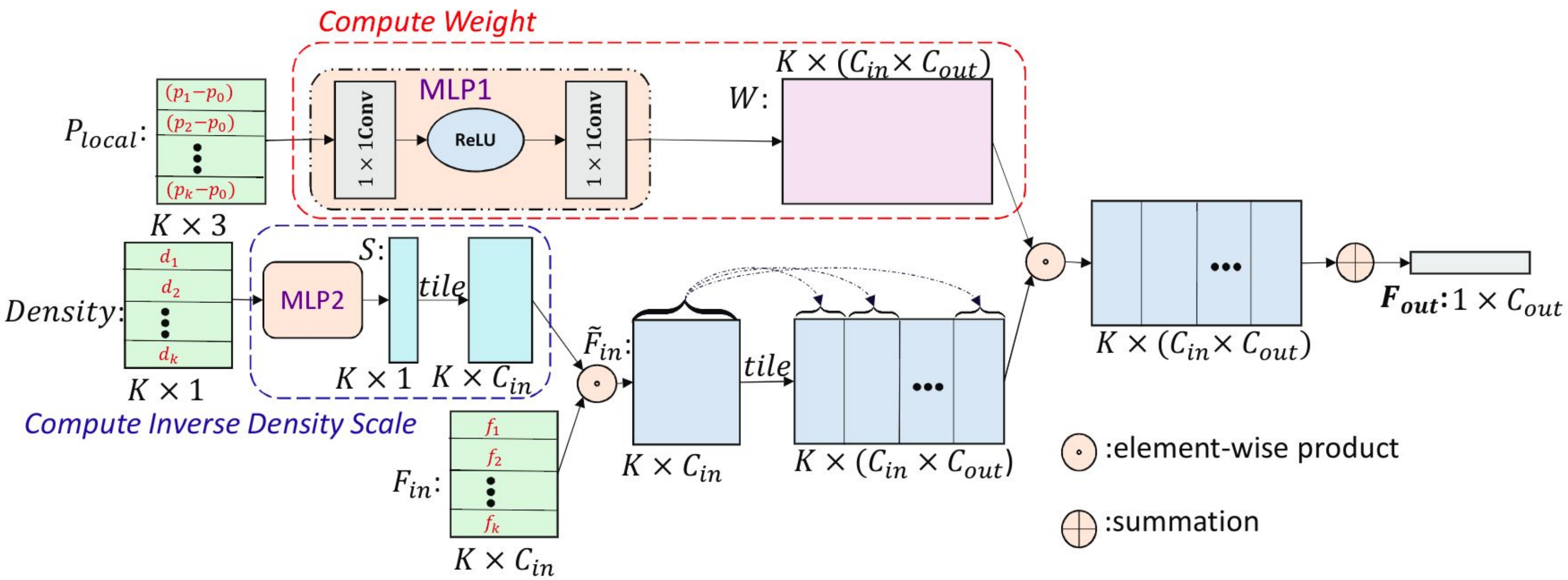
$S(\delta_x, \delta_y, \delta_z)$: **inverse sparsity** function, calculated using KDE

$W(\delta_x, \delta_y, \delta_z)$: the **weight** function

$F(x + \delta_x, y + \delta_y, z + \delta_z)$: the **feature** of a point in the local region G centered in p = (x,y,z)

Point-based methods

Convolutional Neural Networks - PointConv



Point-based methods

Convolutional Neural Networks - PointConv

Architecture for ModelNet40 classification:

- 3 feature encoding blocks, as seen in PointNet++
- 3 fully connected layers

Overall Accuracy: 92.5

https://github.com/DylanWusee/pointconv_pytorch/blob/master/model/pointconv.py

Point-based methods

Convolutional Neural Networks - PointConv

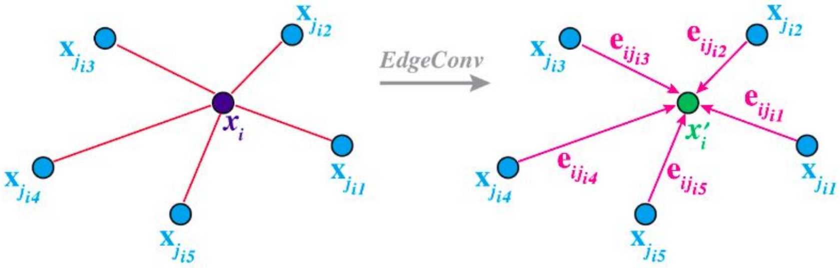
PointConv performances as a traditional convolution layer on CIFAR-10:

Network	Accuracy (%)
AlexNet	89.00
VGG19	93.60
PointConv (5-layers)	89.13
PointConv (VGG19)	93.19

Point-based methods

Graph inspired networks - DGCNN [10]

- Constructs a **local graph for each point**, using k-NN
- For each local graph apply EdgeConv



Point-based methods

Graph inspired networks - DGCNN

Edge feature: non-linear function on an edge, implemented by using MLP

$$e_{ij} = h_{\Theta}(x_i, x_j) : R^F \times R^F \rightarrow R^{F'}$$

Choice of edge function h:

$$h_{\mathbf{\Theta}}\left(\mathbf{x}_i, \mathbf{x}_j\right) = \bar{h}_{\mathbf{\Theta}}\left(\mathbf{x}_i, \mathbf{x}_j - \mathbf{x}_i\right)$$

global structure

local structure

Point-based methods

Graph inspired networks - DGCNN

EdgeConv: apply an aggregation function over all the edge features

$$\mathbf{x}_i' = \square_{j:(i,j)\in\mathcal{E}} h_{\Theta}\left(\mathbf{x}_i, \mathbf{x}_j\right)$$

Choice of the aggregation function: choosing a symmetric aggregation function makes EdgeConv **permutation invariant**

$$x_i' = \max_{j:(i,j)\in\mathcal{E}} e_{ij}',$$

Point-based methods

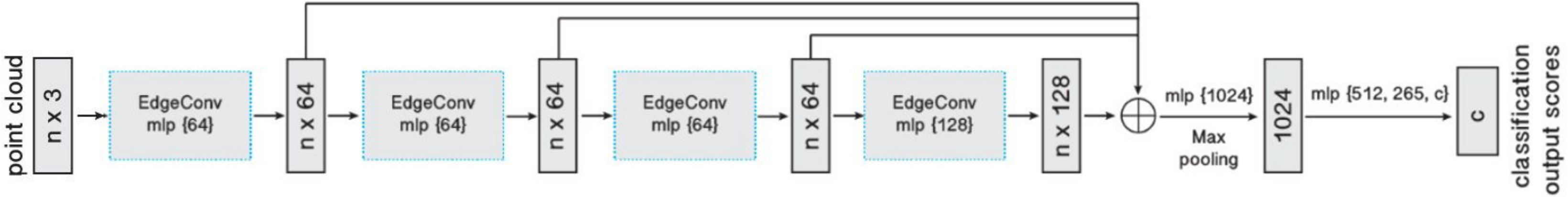
Graph inspired networks - DGCNN

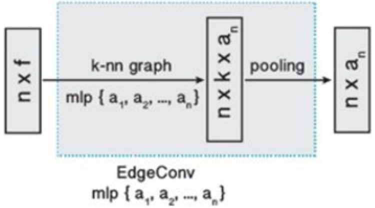
Dynamic Graph Update: recompute the k-nearest neighbors on the features after each EdgeConv operation

Point-based methods

Graph inspired networks - DGCNN

Architecture for **ModelNet40** classification





Point-based methods

Graph inspired networks - DGCNN

CENT	DYN	MPOINTS	MEAN CLASS ACCURACY(%)	OVERALL ACCURACY(%)
x			88.9	91.7
x	x		89.3	92.2
x	x	x	90.2	92.9

- CENT denotes centralization: the edge function $\bar{h}_{\Theta}(\mathbf{x}_i, \mathbf{x}_j - \mathbf{x}_i)$
- DYN denotes dynamical graph recomputation
- MPOINTS denotes experiments with 2,048 points

Comparison

Comparison

Input data

Projection based:

- project cloud points into a different structure like images or 3D grids
- information loss and possible artifacts

Point based:

- directly feed the classifier with the cloud points
- complications like points permutations and spatial information management

Comparison

Permutation invariance

Projection based methods:

- intrinsically invariant to permutation since the points are projected either on images or 3D grids

Point based methods:

- **PointNet and PointNet++:** symmetric function that takes a set of points, such as a max pooling function
- **PointConv:** weights are shared between all the points
- **DGCNN:** symmetric aggregation function over the edge features

Comparison

Transformation invariance

Projection based:

- Both approaches provide an approximate rotational invariance because the network is fed with different rotated copies for each object. Both approaches are invariant to translation because the projections depend on the centroid

Point based:

- **PointNet and PointNet++:** T-Nets used to achieve invariance to rigid transformations
- **PointConv:** full approximation of the convolution operator, thus is invariant to translation. It is not intrinsically invariant to rotation
- **DGCNN:** partially invariant to translation, depending on the edge function. Not intrinsically invariant to rotation

Comparison

Spatial information

Projection based methods:

- Project points into fixed grids and then use standard convolution so they make use of the relationship between points

Point based methods:

- **PointNet:** spatial information not take into account
- **PointNet++:** feature abstraction layer to find local neighbors and then apply PointNet locally to extract a hierarchy of features
- **PointConv:** same feature abstraction layer to find the neighbors and then apply a convolution operator
- **DGCNN:** local neighborhood graph and then apply a convolution-like operator on the edges of the graph

Comparison

Results

All methods have been tested on **ModelNet40**:

	MODEL SIZE (params)	ACCURACY (%)
MVCNN	128M	90.1
VoxNet	921K	83.0
PointNet	3.5 M	89.2
PointNet++	1.48 M	91.9
PointConv	11 M	92.5
DGCNN	1.84 M	92.9

Conclusion

Conclusion

Different approaches

Multiple neural networks were analyzed to understand:

- How point clouds can be fed into a neural network
- The methods used to deal with permutations of the points, geometric transformations
- How to make use of the spatial relationships between the points

Conclusion

Point clouds classification as a starting point

- It is also worth noting that while the classification networks here described, tested on an easy, synthetic dataset might seem not really useful in real case applications, they are used as **backbones for more complicated and interesting tasks**

- For example we can see from the survey by Zhang et al [23] some neural networks that do point cloud registration are based on the methods described: PointNetLK [24] is based on PointNet, DeepVCP [25] is based on PointNet++, DCP [26] is based on DGCNN

References

[1] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3dpoint clouds: A survey, 2020.

[2] Haoming Lu and Humphrey Shi. Deep learning for 3d point cloud understanding: A survey, 2021

[3] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3dshapenets: A deep representation for volumetric shapes. InIEEE Conference on Computer Vision and PatternRecognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015, pages 1912–1920. IEEE Computer Society, 2015

[4] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3dclassification and segmentation, 2017

[5] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on pointsets in a metric space, 2017

[6] Su, Hang, et al. "Multi-view convolutional neural networks for 3d shape recognition." Proceedings of the IEEE international conference on computer vision. 2015.

[7] C. Li H. Zeng, Q. Wang and W. Song.Learning-based multiple pooling fusion in multi-view convolutional neuralnetwork for 3d model classification and retrieval. Journal of Information Processing Systems, 15(5):1179–1191, 2019.

[8] Maturana, Daniel, and Sebastian Scherer. "Voxnet: A 3d convolutional neural network for real-time object recognition." 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2015.

[9] Wenxuan Wu, Zhongang Qi, and Fuxin Li. Pointconv: Deep convolutional networks on 3d point clouds. InIEEEConference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019,pages 9621–9630. Computer Vision Foundation / IEEE, 2019

[10] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. ACM Trans. Graph., 38(5), 2019

[11] Zhiyuan Zhang, Yuchao Dai, and Jiadai Sun. Deep learning based point cloud registration: an overview.VirtualReality & Intelligent Hardware, 2(3):222–246, 2020. 3D Visual Processing and Reconstruction Special Issue

[12] Yasuhiro Aoki, Hunter Goforth, Rangaprasad Arun Srivatsan, and Simon Lucey. Pointnetlk: Robust amp;efficient point cloud registration using pointnet. In2019 IEEE/CVF Conference on Computer Vision and PatternRecognition (CVPR), pages 7156–7165, 2019

[13] Weixin Lu, Guowei Wan, Yao Zhou, Xiangyu Fu, Pengfei Yuan, and Shiyu Song. Deepvcp: An end-to-end deepneural network for point cloud registration. In2019 IEEE/CVF International Conference on Computer Vision(ICCV), pages 12–21, 2019.

[14] Yue Wang and Justin Solomon. Deep closest point: Learning representations for point cloud registration. In2019IEEE/CVF International Conference on Computer Vision (ICCV), pages 3522–3531, 2019

Thank you for your attention

Backup

Point-based methods

Convolutional Neural Networks - PointConv