Deep Learning-based 3D Point Cloud Processing

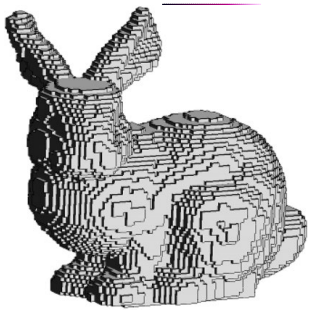
CS5182 Computer Graphics

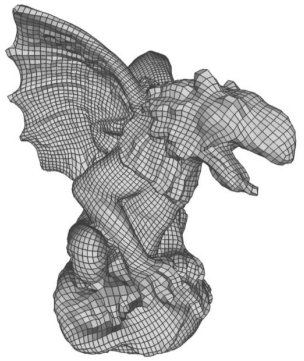
City University of Hong Kong (DG)

3D Data Representation

Regular Domain

- Multi-view Images:
  - A set of 2D images rendered from different viewpoints.
  - Format: [num_views, num_color, height, width].
- Voxels:
  - 3D Volumetric Grids.
  - Format: [depth, height, width].





Irregular Domain

- Polygon Meshes:
  - A collection of vertices, edges, and faces.
  - Format: [num_vertices, coordinates], edges [num_edges, vertex_index].
- Point Clouds:
  - The most direct representation modality of 3D data captured by 3D scanners.
  - Format: [number of points, coordinates].

3D Point Cloud Data

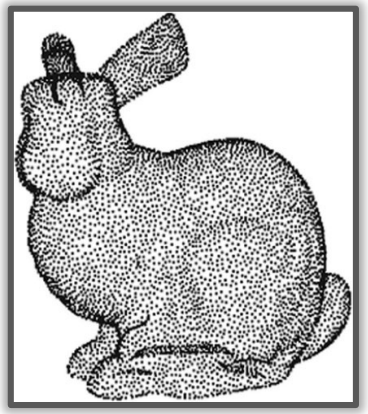
- 2D Image vs. 3D Point Cloud

2D Image

$\{(r, g, b, u, v)\}$ *vs.* $\{(x, y, z)\}$

signals index

both signals and index cues



3D Point Cloud

Illumination information regularly sampled on the Euclidean space (a plane)

Geometric information (3D coordinates) irregularly sampled on the non-Euclidean space (surface)

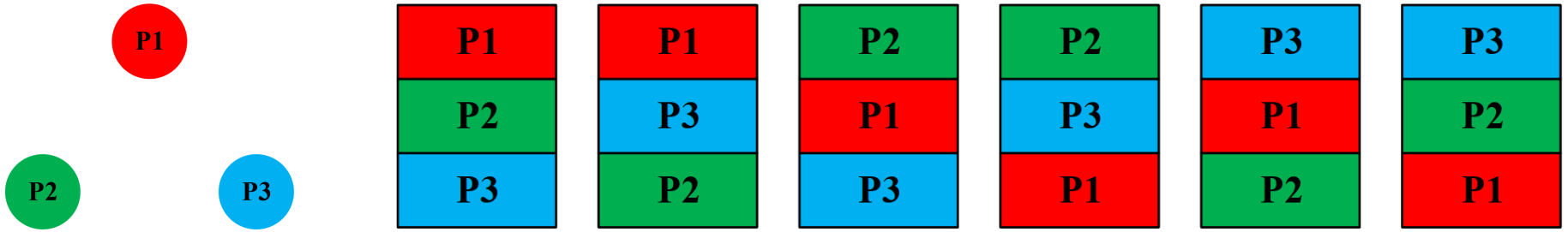
The unstructured nature challenges the development of deep modeling of 3D point cloud data.

Challenges in Point Cloud Processing

- **Irregularity**
  - Unlike 2D images/videos that are densely represented in completely regular grids, there is no canonical representations for 3D point clouds.
  - Typically, we store a 3D point cloud in a 2D ($N \times 3$) array.
- **Un-orderedness**
  - Point clouds are characterized by permutation-invariance.
  - When we arbitrarily change the order of points stored in the 2D array, the resulting point cloud still describes exactly the same 3D shape.
- **Rotation-invariance**
  - Point cloud rotations should not alter classification results

Illustration of Un-orderedness

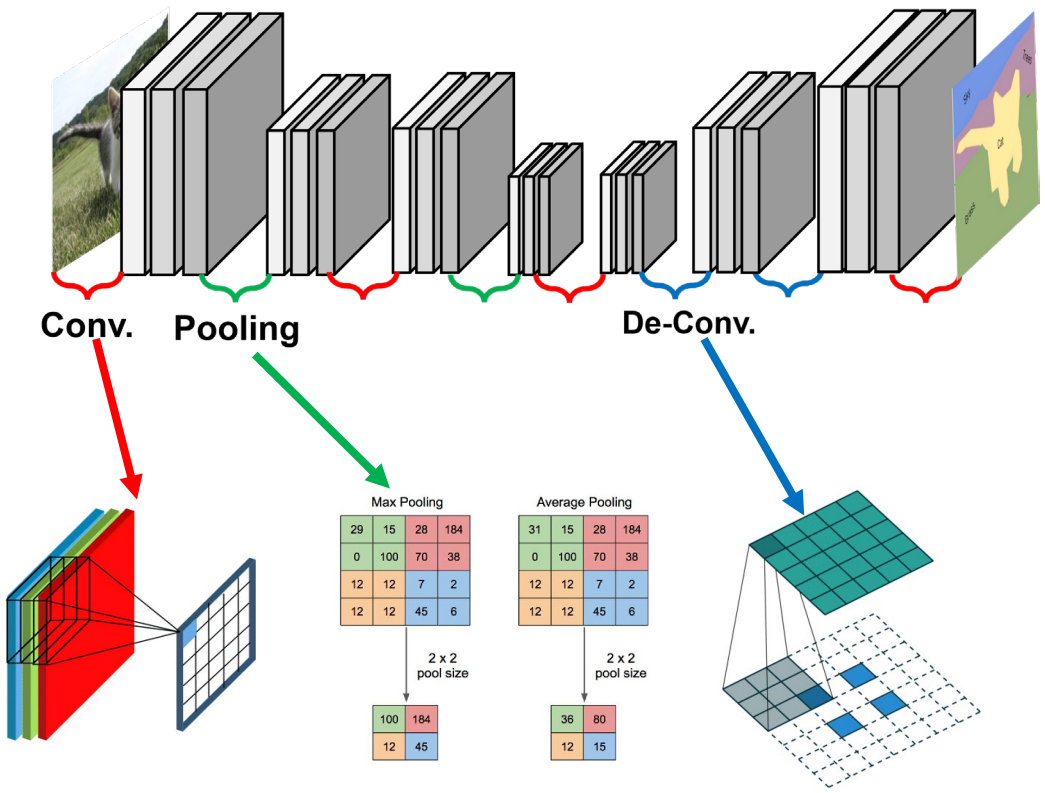
- Consider a toy example with 3 spatial points, which we denote as P1, P2, and P3 (left):



- There exist 6 different arrays (right) that represent the same point set.
- For a point cloud with N points, there existing N! different orders, all of which represent the same shape.

Deep Modeling of 3D Point Cloud Data

- Analogy to Convolutional Neural Network (CNN)



2D Images	3D Point Clouds
Convolution for local information embedding and aggregation	?
Pooling for the reduction of feature dimensions	?
De-convolution/Transposed Conv. for the increase of dimensions	?
Loss function (e.g., pixel-wise distance metric)	?
…	…

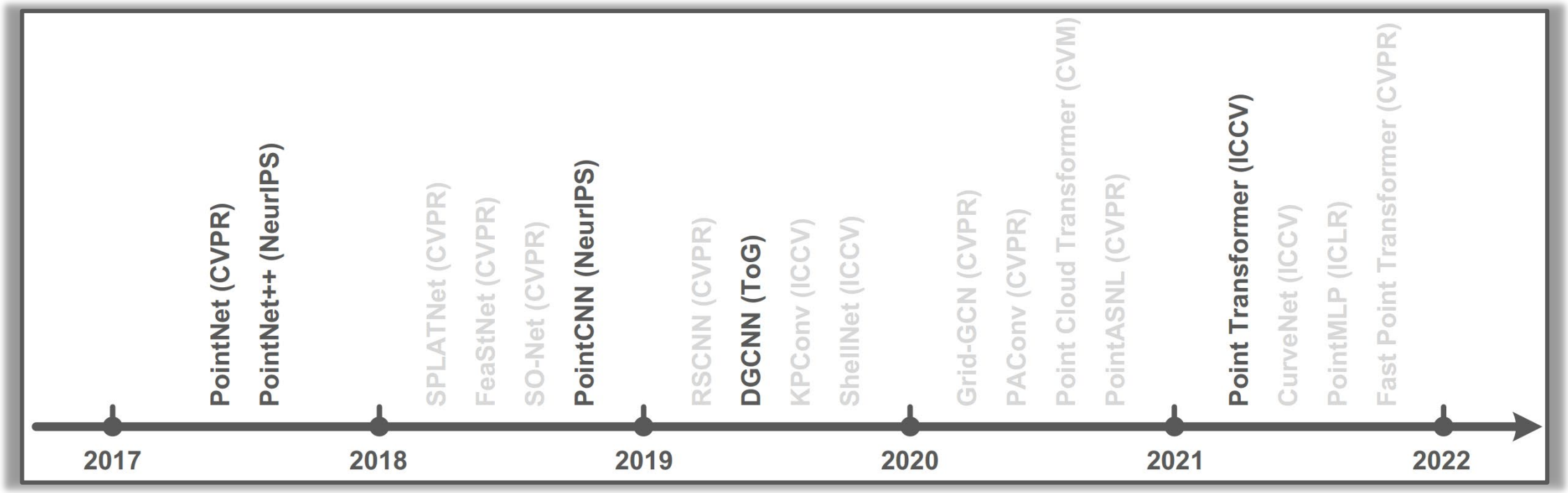
Outline

- **Learning Architectures:**
  - Deep Learning-based Image Analysis
  - Deep Learning-based Point Cloud Analysis

- **Point Cloud Applications:**
  - Classification
  - Segmentation
  - Up-sampling
  - Down-sampling
  - Shape Generation
  - Completion

Deep Modeling of 3D Point Cloud Data

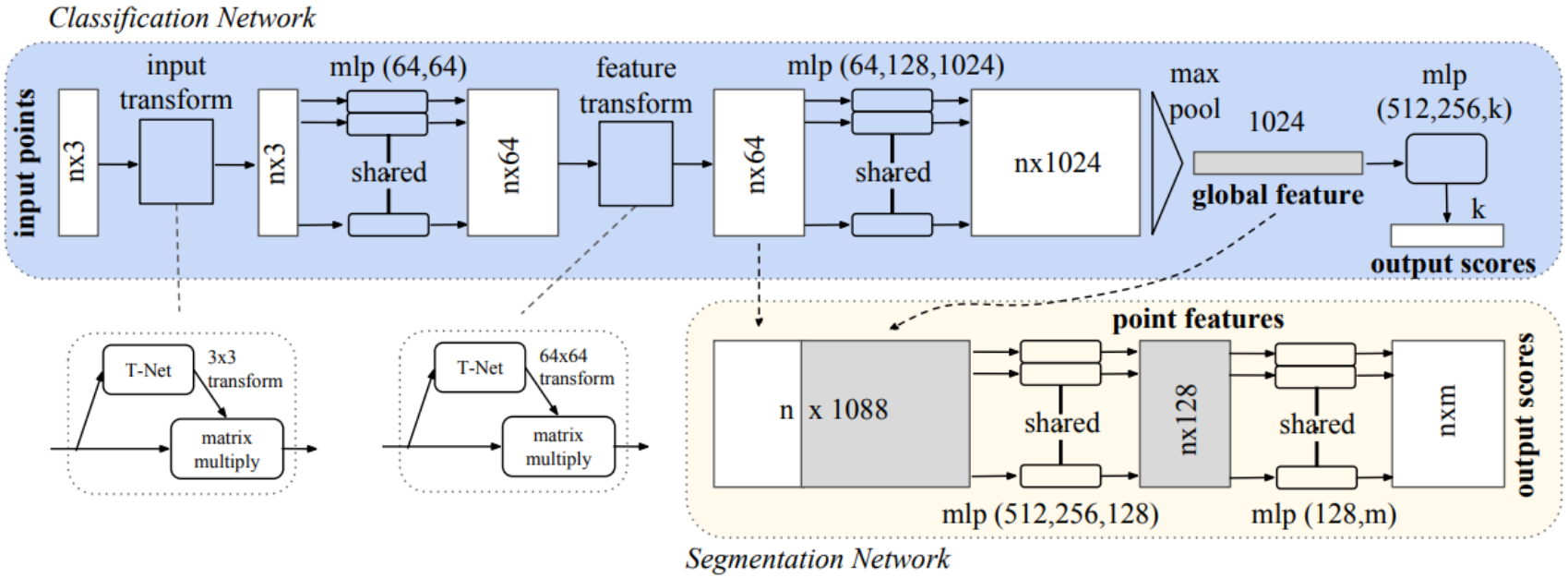
- **Representative Deep Architectures for 3D Point Clouds**
  - Point-based frameworks



Exemplar Point Cloud Learning Architectures: PointNet

[2017-CVPR] PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation

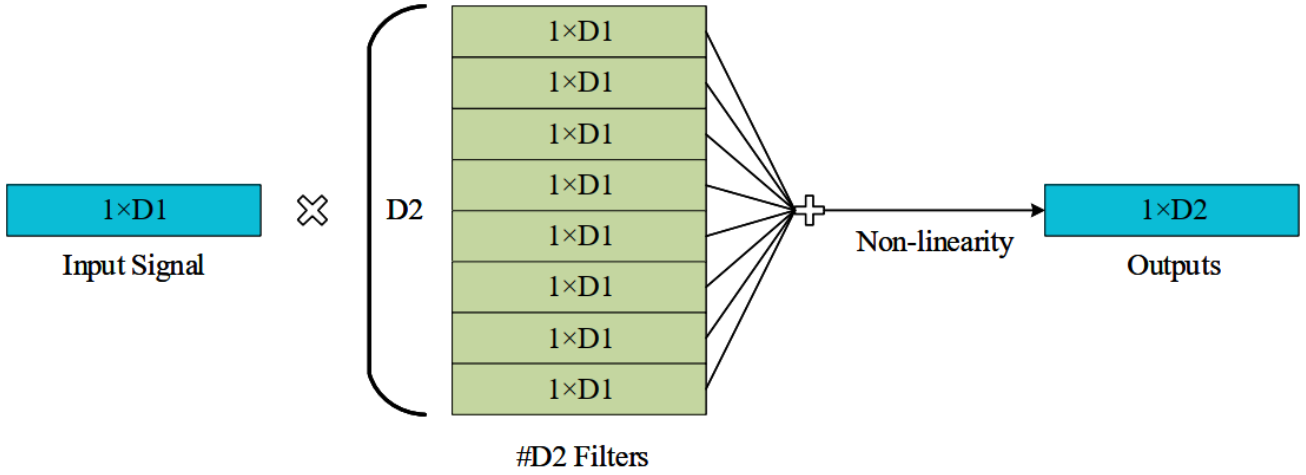
- The first point-based architecture for point cloud feature extraction, consuming raw point sets as input.
- Apply shared multi-layer perceptrons (MLPs) for point-wise feature embedding.
- Achieve permutation-invariance by the symmetric max-pooling operation to generate global features.
- Without considering neighborhood information, which greatly limited its representation ability.



Exemplar Point Cloud Learning Architectures: PointNet

[2017-CVPR] PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation

- **Learning on a single point:**

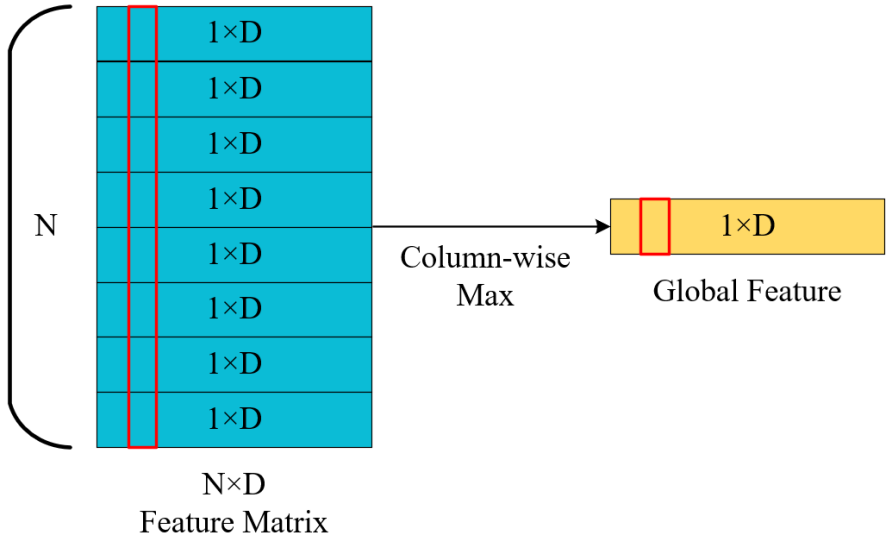


- **Learning on a point set:**
  - Repeatedly apply the same operation, as shown above, to every single point in the given set.
  - For an input N×D1 point set, the output is of N×D2. (For raw point coordinates, D1=3)

Exemplar Point Cloud Learning Architectures: PointNet

[2017-CVPR] PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation

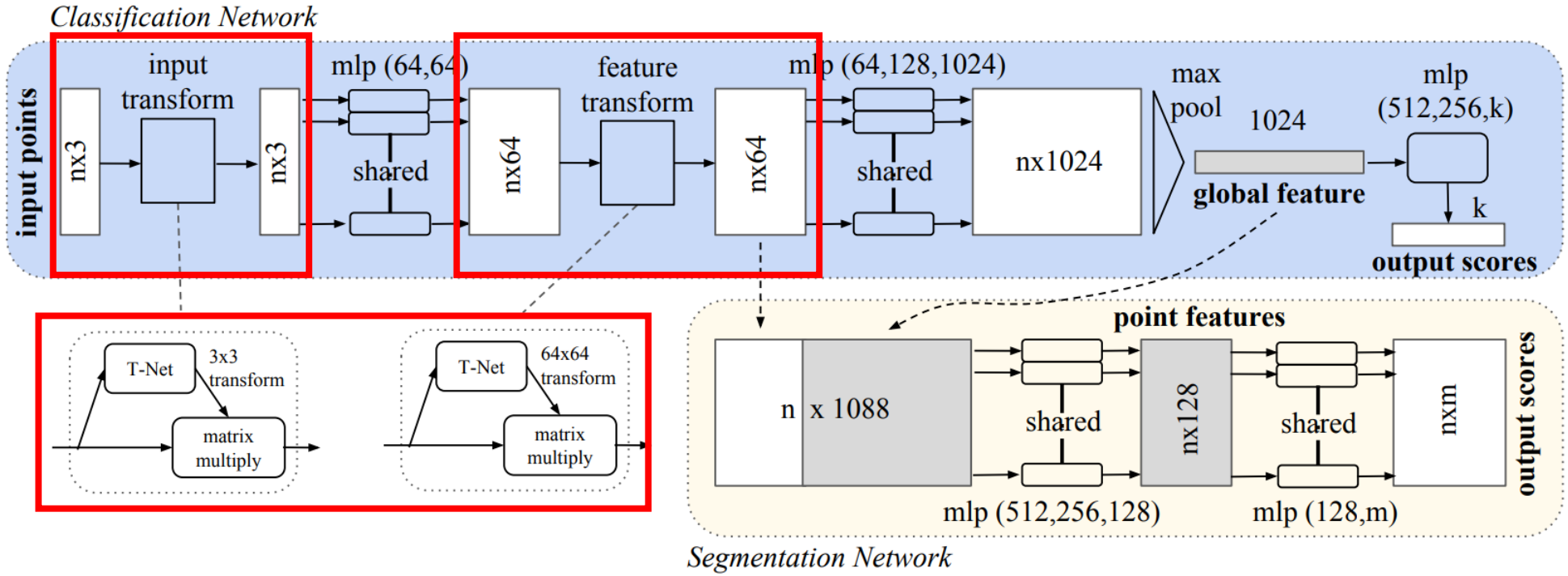
- **Max-pooling for generating permutation-invariant global feature:**



Exemplar Point Cloud Learning Architectures: PointNet

[2017-CVPR] PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation

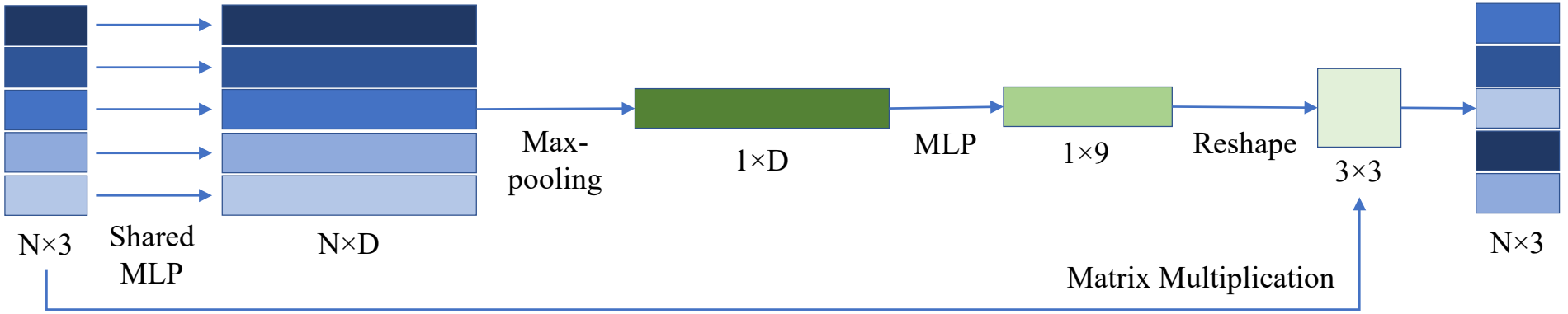
- **T-Net:**
  - The feature extraction process should be robust to geometric transformation, *e.g.*, rigid transformation.
  - T-Net is designed to adaptively learn 3×3 / D×D transformation matrix to adjust the points/features.



Exemplar Point Cloud Learning Architectures: PointNet

[2017-CVPR] PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation

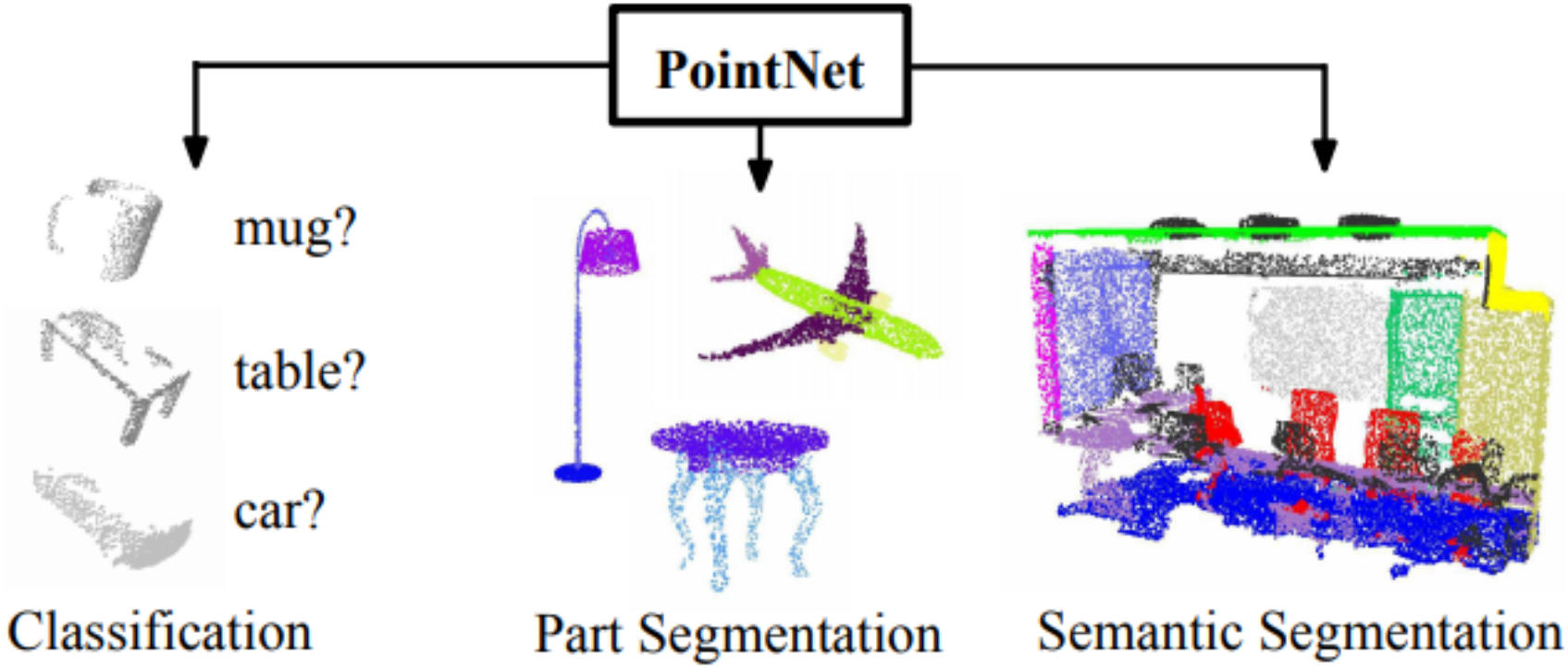
- **Details of T-Net:**
  - Share the same architectural design with the feature extraction process.



Exemplar Point Cloud Learning Architectures: PointNet

[2017-CVPR] PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation

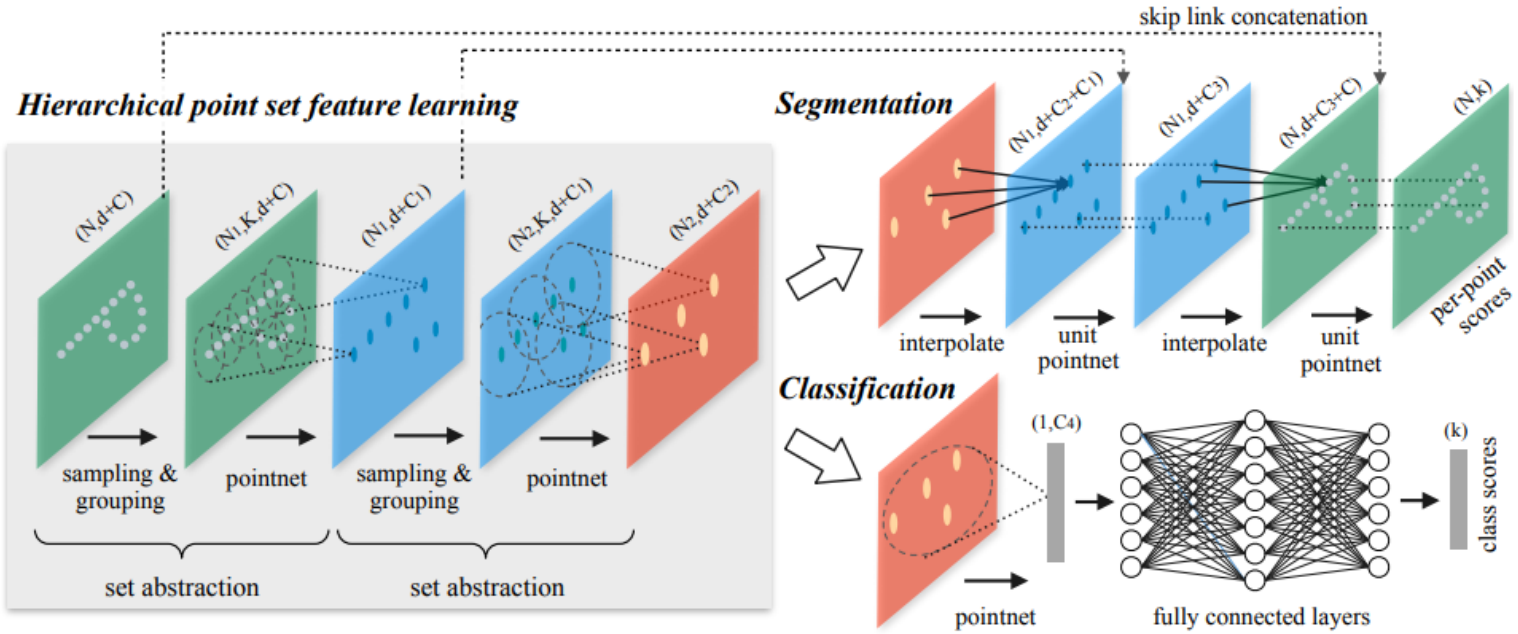
- **PointNet for Classification and Segmentation:**
  - Classification: feed global feature into fully-connected layers to predict a class score vector.
  - Segmentation: duplicate global feature and concatenate with point-wise features to predict class scores for every point. (segmentation can be considered as point-wise classification)



Exemplar Point Cloud Learning Architectures: PointNet++

[2017-NeurIPS] PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space

- Motivation: The previous PointNet architecture ignores modeling local neighborhood information.
- Apply PointNet as the basic point-wise feature extraction module.
- Progressively down-scale the input point set by FPS and adopted ball query to search K-nearest-neighbors of each centroid point.
- Aggregate features within KNN patches while considering relative distances between points as features.



Point Cloud Feature Extraction: Pooling

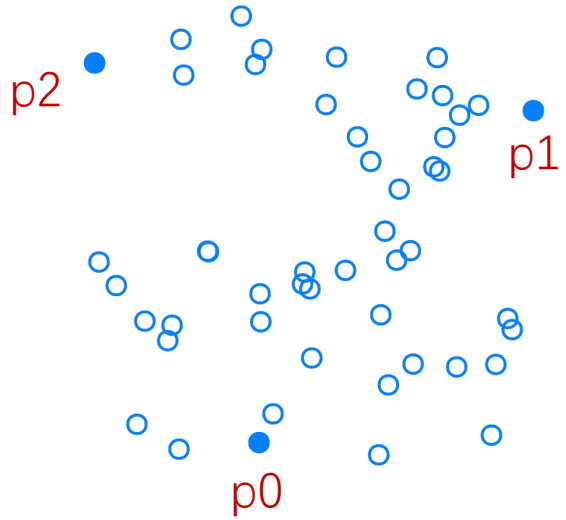
- Random Sampling
  - Randomly select a subset of original points.
  - Cannot guarantee coverage and uniformity.
  - Time cost: O(1), very fast.

- Farthest Point Sampling
  - Iteratively select the point that is farthest from currently selected points.
  - Can produce more uniform subset points with good coverage.
  - Time cost: O(N^2), very slow especially for large-scale point clouds.

Input: a point set $S \subset D$
Output: a sample S of D satisfying $U_S(p) \leqslant \varepsilon,\ \forall\, p \in D$

1) Select the farthest point of D from S : $p_{max} \leftarrow \underset{p \in D}{argmax}\ U_S(p)$

2) If $U_S(p_{max}) < \varepsilon$ then exit, else set $S \leftarrow S \cup \{p_{max}\}$ and goto 1.

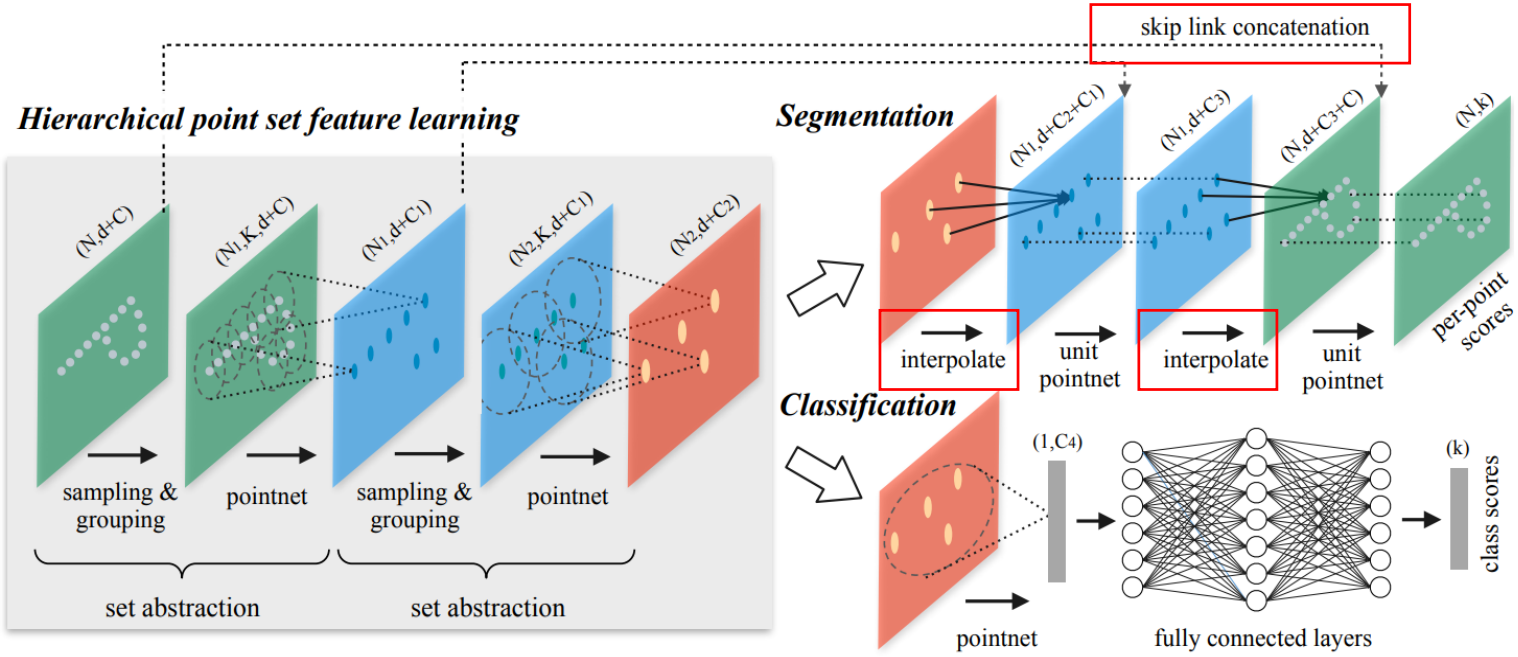


Exemplar Point Cloud Learning Architectures: PointNet++

[2017-NeurIPS] PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space

- Add skip links to concatenate low-level features for segmentation-style tasks.
- Interpolate original point feature from KNN sparse features:

$$f^{(j)}(x) = \frac{\sum_{i=1}^{k} w_i(x) f_i^{(j)}}{\sum_{i=1}^{k} w_i(x)} \qquad \text{where} \quad w_i(x) = \frac{1}{d(x, x_i)^p}, \; j = 1, ..., C$$



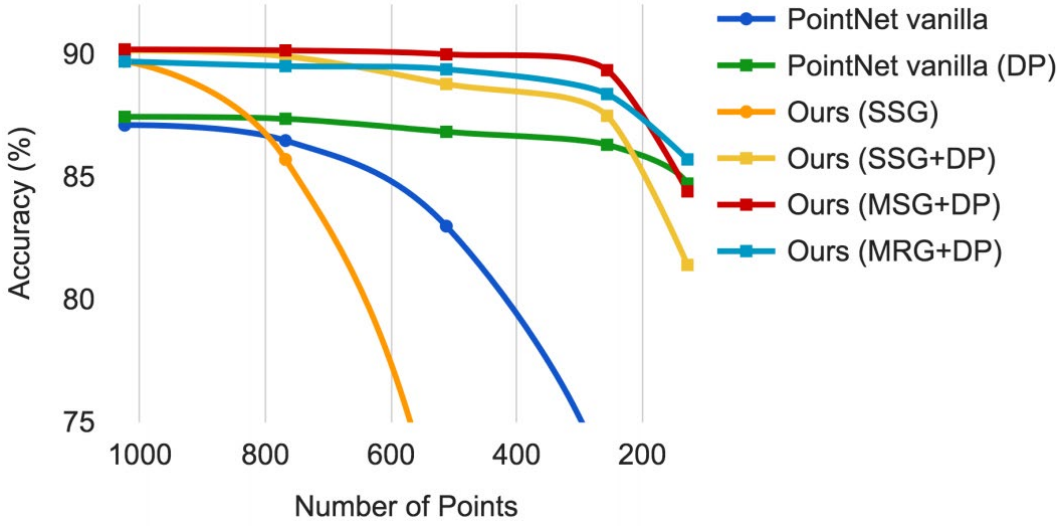
Exemplar Point Cloud Learning Architectures: PointNet++

[2017-NeurIPS] PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space

- **Performance**:

Method	Input	Accuracy (%)
Subvolume [21]	vox	89.2
MVCNN [26]	img	90.1
PointNet (vanilla) [20]	pc	87.2
PointNet [20]	pc	89.2
Ours	pc	90.7
Ours (with normal)	pc	**91.9**

Table 2: ModelNet40 shape classification.

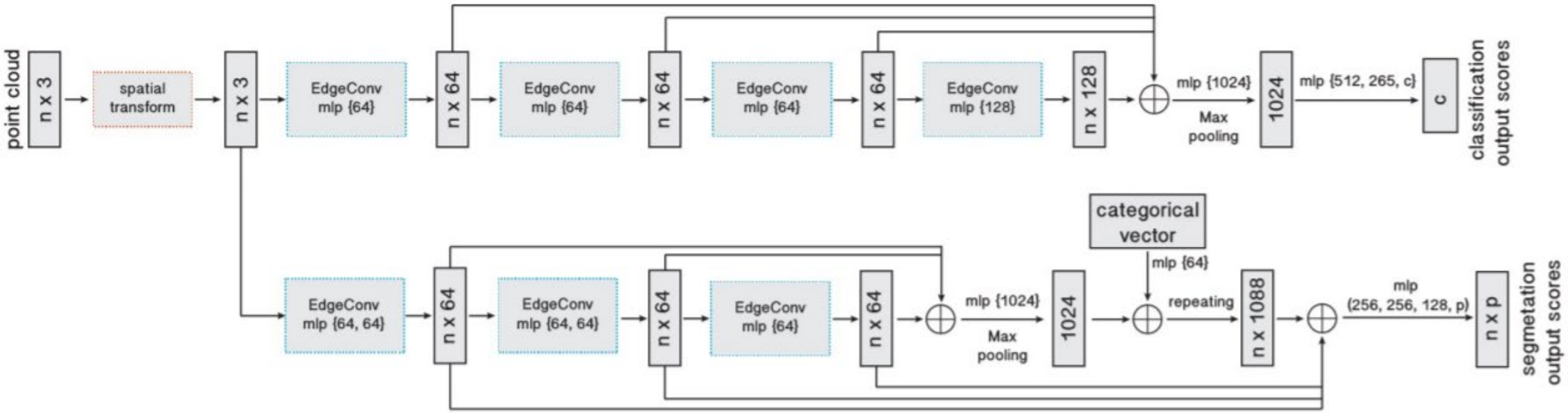


Exemplar Point Cloud Learning Architectures: DGCNN

香港城市大學
City University of Hong Kong

[2019-TOG] Dynamic Graph CNN for Learning on Point Clouds

- Motivation: Previous methods like PointNet++ define static graph by coordinate-driven KNN search, while the DGCNN method dynamically constructs local graph based on learned features.
- Proposed a powerful EdgeConv operator for dynamic point feature extraction:



Exemplar Point Cloud Learning Architectures: DGCNN

香港城市大學 City University of Hong Kong

[2019-TOG] Dynamic Graph CNN for Learning on Point Clouds

- **EdgeConv**:
  - Compute Euclidean distances in **feature space** to search K-nearest-neighbors.
  - Concatenate neighbors' coordinates and features with centroids.

$$h_\Theta(\mathbf{x}_i, \mathbf{x}_j) = \bar{h}_\Theta(\mathbf{x}_i, \mathbf{x}_j - \mathbf{x}_i). \tag{7}$$

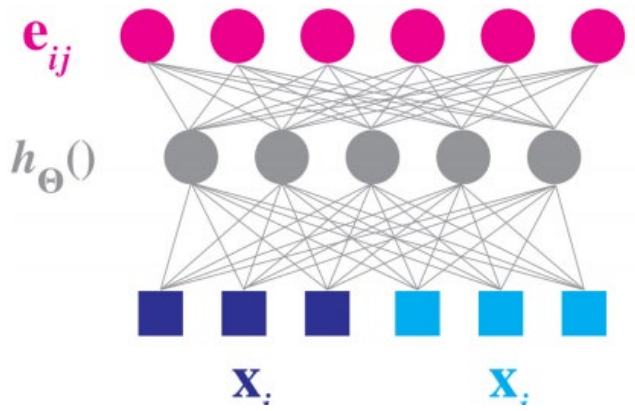
This explicitly combines global shape structure, captured by the coordinates of the patch centers $\mathbf{x}_i$, with local neighborhood information, captured by $\mathbf{x}_j - \mathbf{x}_i$. In particular, we can define our operator by notating

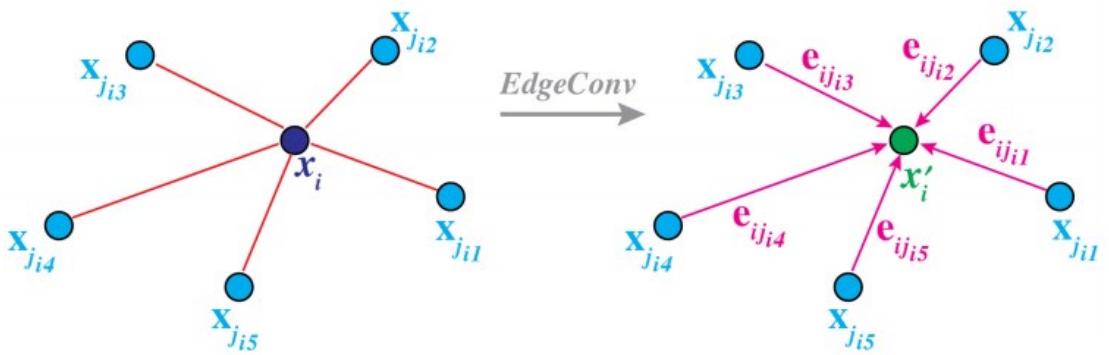
$$e'_{ijm} = \mathrm{ReLU}(\boldsymbol{\theta}_m \cdot (\mathbf{x}_j - \mathbf{x}_i) + \boldsymbol{\phi}_m \cdot \mathbf{x}_i), \tag{8}$$

which can be implemented as a shared MLP, and taking

$$x'_{im} = \max_{j:(i,j)\in\mathcal{E}} e'_{ijm}, \tag{9}$$

where $\Theta = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_M, \boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_M)$.





Exemplar Point Cloud Learning Architectures: DGCNN

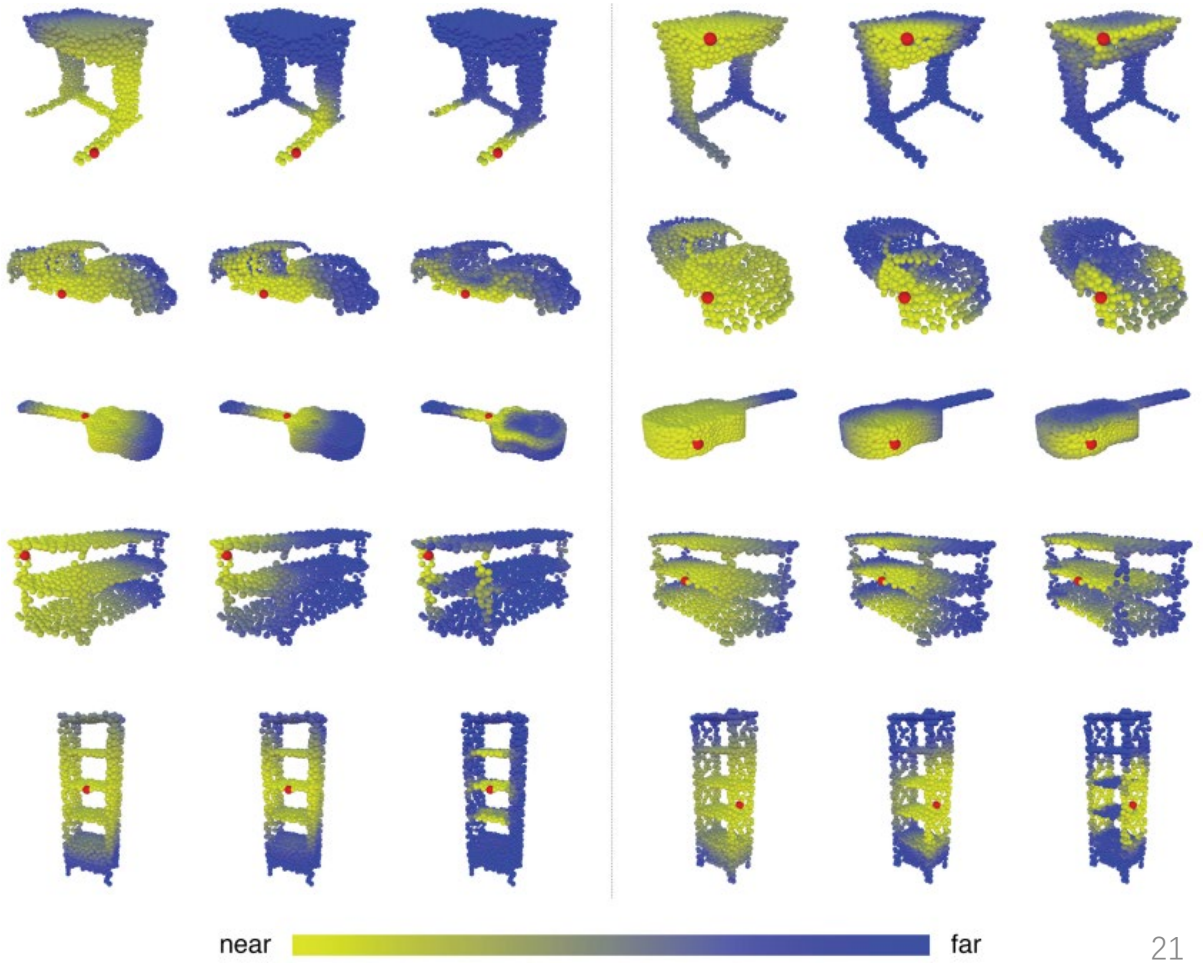
[2019-TOG] Dynamic Graph CNN for Learning on Point Clouds

- **Illustrations of how EdgeConv helps to learn semantic similarity**:

Left: Euclidean distance in the input point space

Middle: Distance after the point cloud transform stage

Right: Distance in the feature space of the last layer



Exemplar Point Cloud Learning Architectures: DGCNN

[2019-TOG] Dynamic Graph CNN for Learning on Point Clouds

- **Performance**:

Table 2. Classification Results on ModelNet40

	Mean Class Accuracy	Overall Accuracy
3DShapeNets (Wu et al. 2015)	77.3	84.7
VoxNet (Maturana and Scherer 2015)	83.0	85.9
Subvolume (Qi et al. 2016)	86.0	89.2
VRN (single view) (Brock et al. 2016)	88.98	—
VRN (multiple views) (Brock et al. 2016)	91.33	—
ECC (Simonovsky and Komodakis 2017)	83.2	87.4
PointNet (Qi et al. 2017b)	86.0	89.2
PointNet++ (Qi et al. 2017c)	—	90.7
Kd-net (Klokov and Lempitsky 2017)	—	90.6
PointCNN (Li et al. 2018a)	88.1	92.2
PCNN (Atzmon et al. 2018)	—	92.3
Ours (baseline)	88.9	91.7
Ours	**90.2**	**92.9**
Ours (2048 points)	**90.7**	**93.5**

Table 3. Complexity, Forward Time, and Accuracy of Different Models

	Model size(MB)	Time(ms)	Accuracy(%)
PointNet (Baseline) (Qi et al. 2017b)	9.4	6.8	87.1
PointNet (Qi et al. 2017b)	40	16.6	89.2
PointNet++ (Qi et al. 2017c)	12	163.2	90.7
PCNN (Atzmon et al. 2018)	94	117.0	92.3
Ours (Baseline)	11	19.7	91.7
Ours	21	27.2	92.9

Table 5. Results of Our Model with Different Numbers of Nearest Neighbors

Number of nearest neighbors (k)	Mean Class Accuracy(%)	Overall Accuracy(%)
5	88.0	90.5
10	88.9	91.4
20	90.2	92.9
40	89.4	92.4

Exemplar Point Cloud Learning Architectures

- Different from image analysis, in which everyone is using standard 2D CNNs, point cloud convolution still remains an open problem and is a very hot topic in the research community.

- **Key Issues**:
- How to define neighborhood?
- How to extract point-wise feature?
- How to aggregate local information?
- How to design the whole framework?
- How to incorporate geometric properties, such as normals, into the overall learning framework effectively?

Methods		Input	#params (M)	ModelNet40 (OA)	ModelNet40 (mAcc)	ModelNet10 (OA)	ModelNet10 (mAcc)
Pointwise MLP Networks	PointNet [5]	Coordinates	3.48	89.2%	86.2%	-	-
	PointNet++ [27]	Coordinates	1.48	90.7%	-	-	-
	MO-Net [30]	Coordinates	3.1	89.3%	86.1%	-	-
	Deep Sets [26]	Coordinates	-	87.1%	-	-	-
	PAT [29]	Coordinates	-	91.7%	-	-	-
	PointWeb [31]	Coordinates	-	92.3%	89.4%	-	-
	SRN-PointNet++ [32]	Coordinates	-	91.5%	-	-	-
	JUSTLOOKUP [33]	Coordinates	-	89.5%	86.4%	92.9%	92.1%
Convolution-based Networks	Pointwise-CNN [49]	Coordinates	-	86.1%	81.4%	-	-
	PointConv [38]	Coordinates+Normals	-	92.5%	-	-	-
	MC Convolution [39]	Coordinates	-	90.9%	-	-	-
	SpiderCNN [40]	Coordinates+Normals	-	92.4%	-	-	-
	PointCNN [52]	Coordinates	0.45	92.2%	88.1%	-	-
	Flex-Convolution [48]	Coordinates	-	90.2%	-	-	-
	PCNN [41]	Coordinates	1.4	92.3%	-	94.9%	-
	Boulch [36]	Coordinates	-	91.6%	88.1%	-	-
	RS-CNN [35]	Coordinates	-	92.6%	-	-	-
	Spherical CNNs [43]	Coordinates	0.5	88.9%	-	-	-
	GeoCNN [51]	Coordinates	-	93.4%	91.1%	-	-
	Ψ-CNN [50]	Coordinates	-	92.0%	88.7%	94.6%	94.4%
	A-CNN [55]	Coordinates	-	92.6%	90.3%	95.5%	95.3%
	SFCNN [57]	Coordinates	-	91.4%	-	-	-
	SFCNN [57]	Coordinates+Normals	-	92.3%	-	-	-
	DensePoint [37]	Coordinates	0.53	93.2%	-	96.6%	-
	KPConv rigid [42]	Coordinates	-	92.9%	-	-	-
	KPConv deform [42]	Coordinates	-	92.7%	-	-	-
	InterpCNN [53]	Coordinates	12.8	93.0%	-	-	-
	ConvPoint [47]	Coordinates	-	91.8%	88.5%	-	-
Graph-based Networks	ECC [58]	Coordinates	-	87.4%	83.2%	90.8%	90.0%
	KCNet [66]	Coordinates	0.9	91.0%	-	94.4%	-
	DGCNN [60]	Coordinates	1.84	92.2%	90.2%	-	-
	LocalSpecGCN [75]	Coordinates+Normals	-	92.1%	-	-	-
	RGCNN [72]	Coordinates+Normals	2.24	90.5%	87.3%	-	-
	LDGCNN [61]	Coordinates	-	92.9%	90.3%	-	-
	3DTI-Net [77]	Coordinates	2.6	91.7%	-	-	-
	PointGCN [76]	Coordinates	-	89.5%	86.1%	91.9%	91.6%
	ClusterNet [68]	Coordinates	-	87.1%	-	-	-
	Hassani et al. [64]	Coordinates	-	89.1%	-	-	-
	DPAM [65]	Coordinates	-	91.9%	89.9%	94.6%	94.3%
Data Indexing-based Networks	KD-Net [78]	Coordinates	2.0	91.8%	88.5%	94.0%	93.5%
	SO-Net [80]	Coordinates	-	90.9%	87.3%	94.1%	93.9%
	SCN [81]	Coordinates	-	90.0%	87.6%	-	-
	A-SCN [81]	Coordinates	-	89.8%	87.4%	-	-
	3DContextNet [79]	Coordinates	-	90.2%	-	-	-
	3DContextNet [79]	Coordinates+Normals	-	91.1%	-	-	-
Other Networks	3DmFV-Net [82]	Coordinates	4.6	91.6%	-	95.2%	-
	PVNet [83]	Coordinates+Views	-	93.2%	-	-	-
	PVRNet [84]	Coordinates+Views	-	93.6%	-	-	-
	3DPointCapsNet [85]	Coordinates	-	89.3%	-	-	-
	DeepRBFNet [86]	Coordinates	3.2	90.2%	87.8%	-	-
	DeepRBFNet [86]	Coordinates+Normals	3.2	92.1%	88.8%	-	-
	Point2Sequences [87]	Coordinates	-	92.6%	90.4%	95.3%	95.1%
	RCNet [88]	Coordinates	-	91.6%	-	94.7%	-
	RCNet-E [88]	Coordinates	-	92.3%	-	95.6%	-

Deep Learning-based 3D Point Cloud Processing

- We have discussed several popular paradigms on point cloud feature learning as well as their applications in point cloud classification and segmentation tasks.

- More application scenarios in point cloud research community:
  - Up-sampling
  - Down-sampling
  - Shape Reconstruction
  - Completion

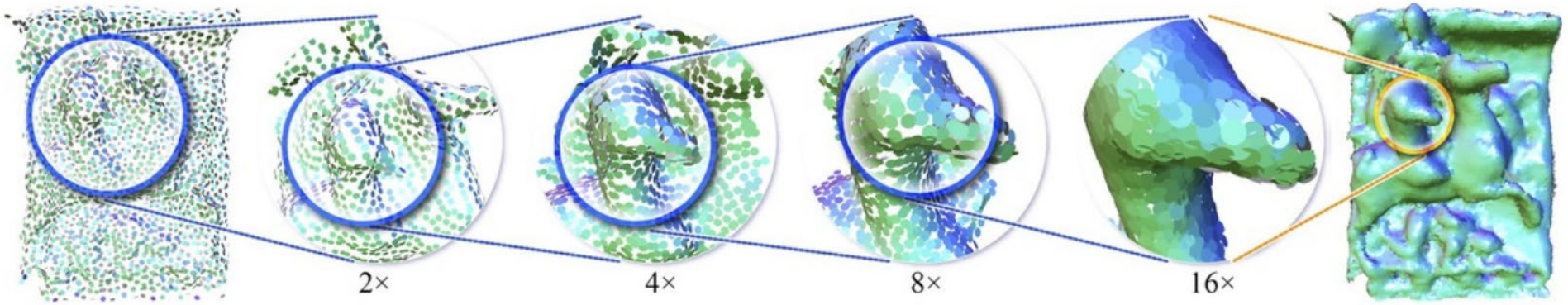
Deep Point Cloud Processing: Up-sampling

香港城市大學
City University of Hong Kong

Overview

- Given a sparse N×3 point cloud and the corresponding upsampling ratio R, we aim to generate a dense M×3 point cloud, where M = N × R.

- **Requirements**:
  - The generated points must locate on the underlying surface indicated by the original shape.
  - The generated points should be distributed uniformly.
  - It is a challenging generative task, which requires detailed geometry information extracted from sparse point cloud.

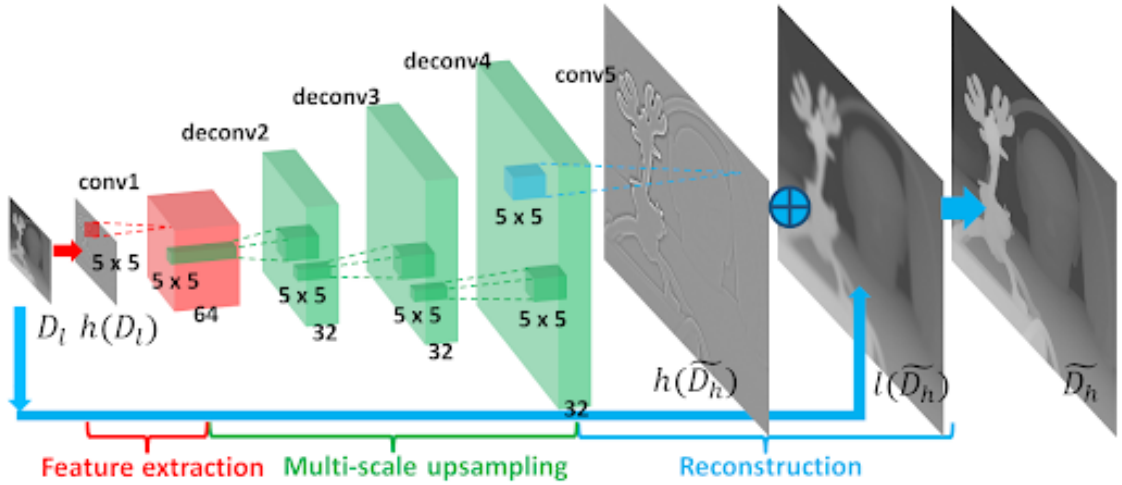


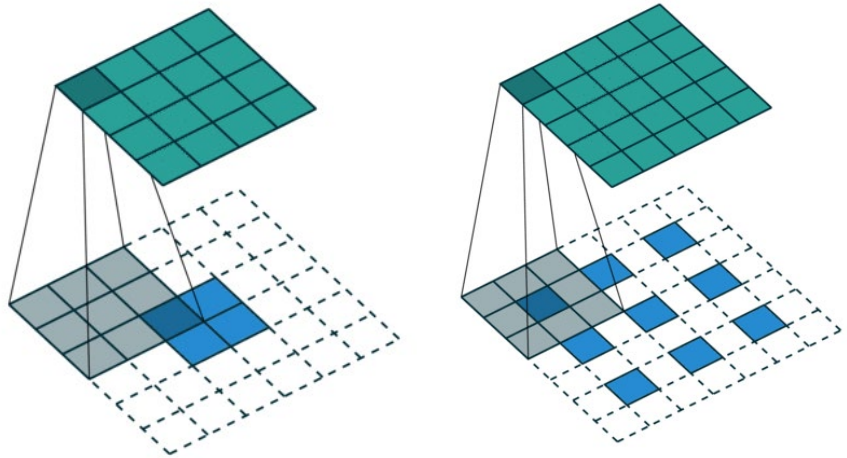
Deep Point Cloud Processing: Up-sampling

Point Cloud Upsampling *vs.* Image Super-resolution

- **Image super-resolution**
  - Given a low-resolution (LR) H×W×3 image with size and the corresponding upsampling ratio R, we sim to generate a high-resolution (HR) image with the size [H×R, W×R, 3].
  - The key component is the "deconvolution" operator for enlarging the feature resolution.

- **Point cloud upsampling**
  - How to expand sparse features to regress denser coordinates? (How to adapt the deconvolution operator to point cloud processing?)

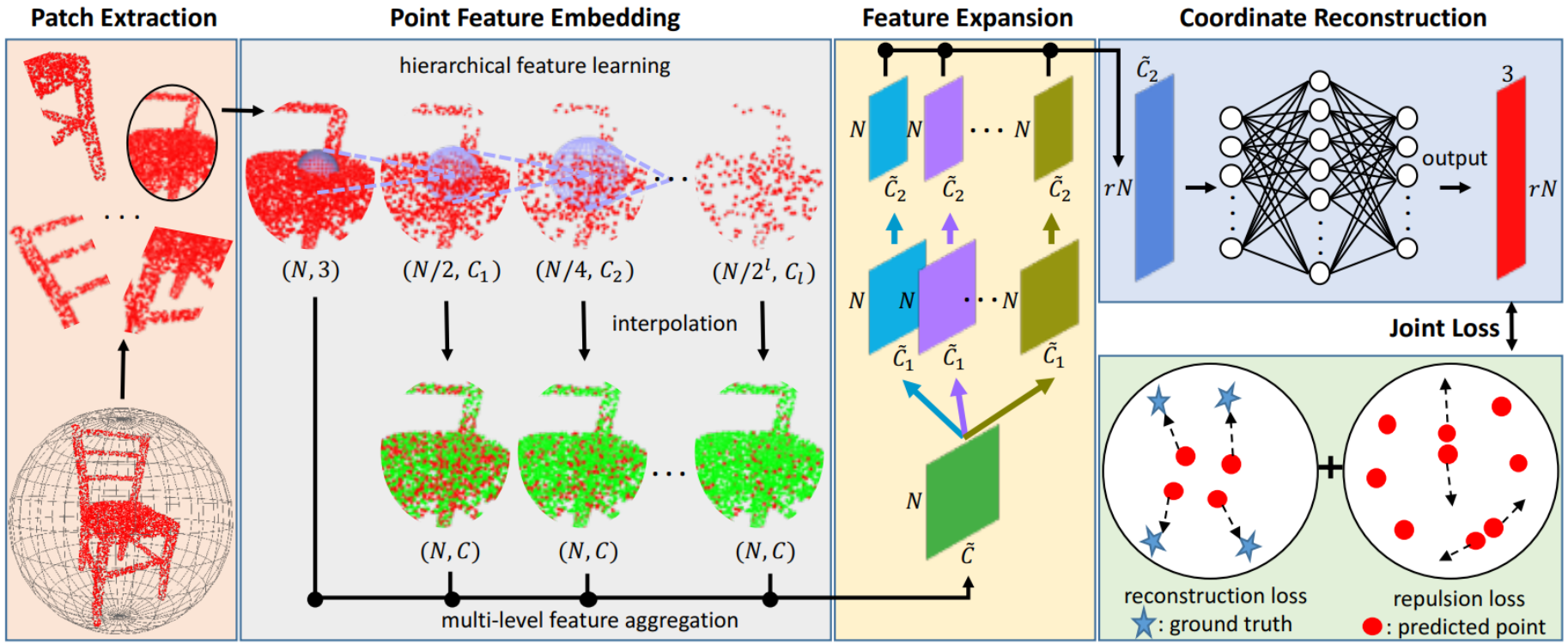




Deep Point Cloud Processing: Up-sampling

[2018-CVPR] PU-Net: Point Cloud Upsampling Network

- Patch-wise processing, extract patches by FPS.
- Use PointNet++ to extract features.
- Adopt R network to achieve R× up-sampling.

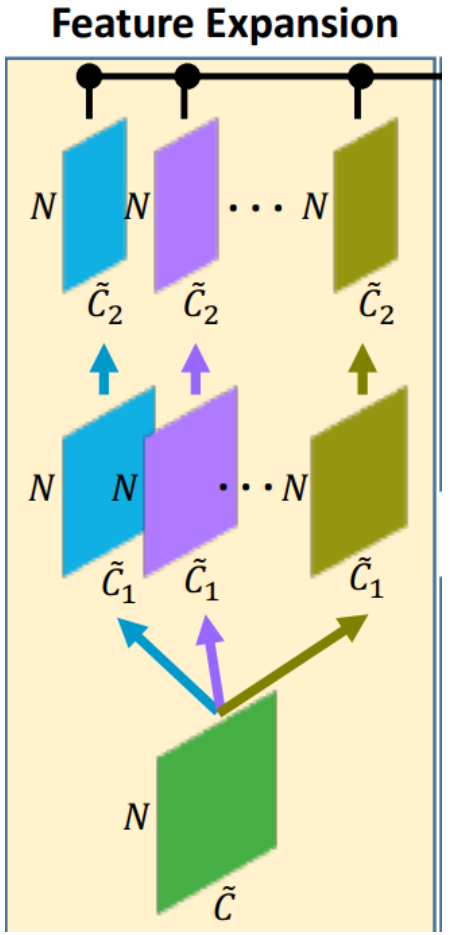


Deep Point Cloud Processing: Up-sampling

[2018-CVPR] PU-Net: Point Cloud Upsampling Network

- **Feature Expansion**
  - Copy the sparse N×C feature for R times.
  - Use R separate MLPs for each copy, therefore expanding features from [N, C] to R×[N, C2].
  - The expanded features are reshaped to [R×N, C2] to further regress the spatial coordinates.

- **Drawbacks**
  - Need to use R separate MLPs for each copy, thus less efficient and scalable.
  - The generated points tend to be clustered, thus need an additional repulsion loss to separate points:



Deep Point Cloud Processing: Up-sampling

[2018-CVPR] PU-Net: Point Cloud Upsampling Network

- Given the ground truth (GT) dense point cloud, we jointly minimize the Earth Mover Distance (EMD) and the repulsion loss:

$$L_{rec} = d_{EMD}(S_p, S_{gt}) = \min_{\phi: S_p \to S_{gt}} \sum_{x_i \in S_p} \|x_i - \phi(x_i)\|_2$$

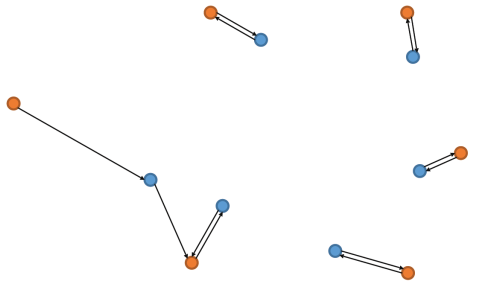
$$L_{rep} = \sum_{i=0}^{\hat{N}} \sum_{i' \in K(i)} \eta(\|x_{i'} - x_i\|) w(\|x_{i'} - x_i\|)$$

$$\eta(r) = -r \qquad w(r) = e^{-r^2/h^2}$$

Deep Point Cloud Processing: Up-sampling

- **Loss Function**
  - Chamfer Distance (CD):

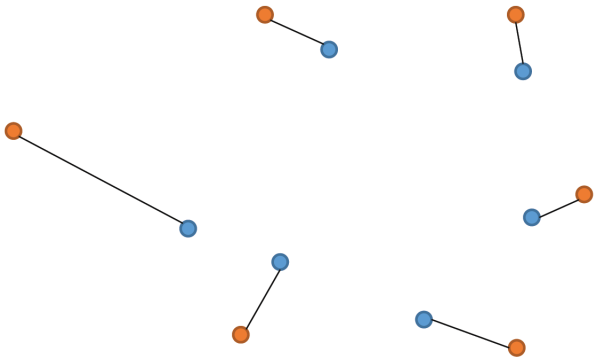
$$d_{CD}(S_1, S_2) = \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2^2 + \sum_{y \in S_2} \min_{x \in S_1} \|x - y\|_2^2$$



  - Earth Mover's Distance (EMD):

$$d_{EMD}(S_1, S_2) = \min_{\phi: S_1 \to S_2} \sum_{x \in S_1} \|x - \phi(x)\|_2$$

where $\phi : S_1 \to S_2$ is a bijection.



Deep Point Cloud Processing: Up-sampling

[2019-CVPR] Patch-based Progressive 3D Point Set Upsampling

- **Performance:**
  - 16× upsampling from sparse 5000 points to dense 80000 points.
  - Surface reconstruction via Poisson Surface Reconstruction algorithm.

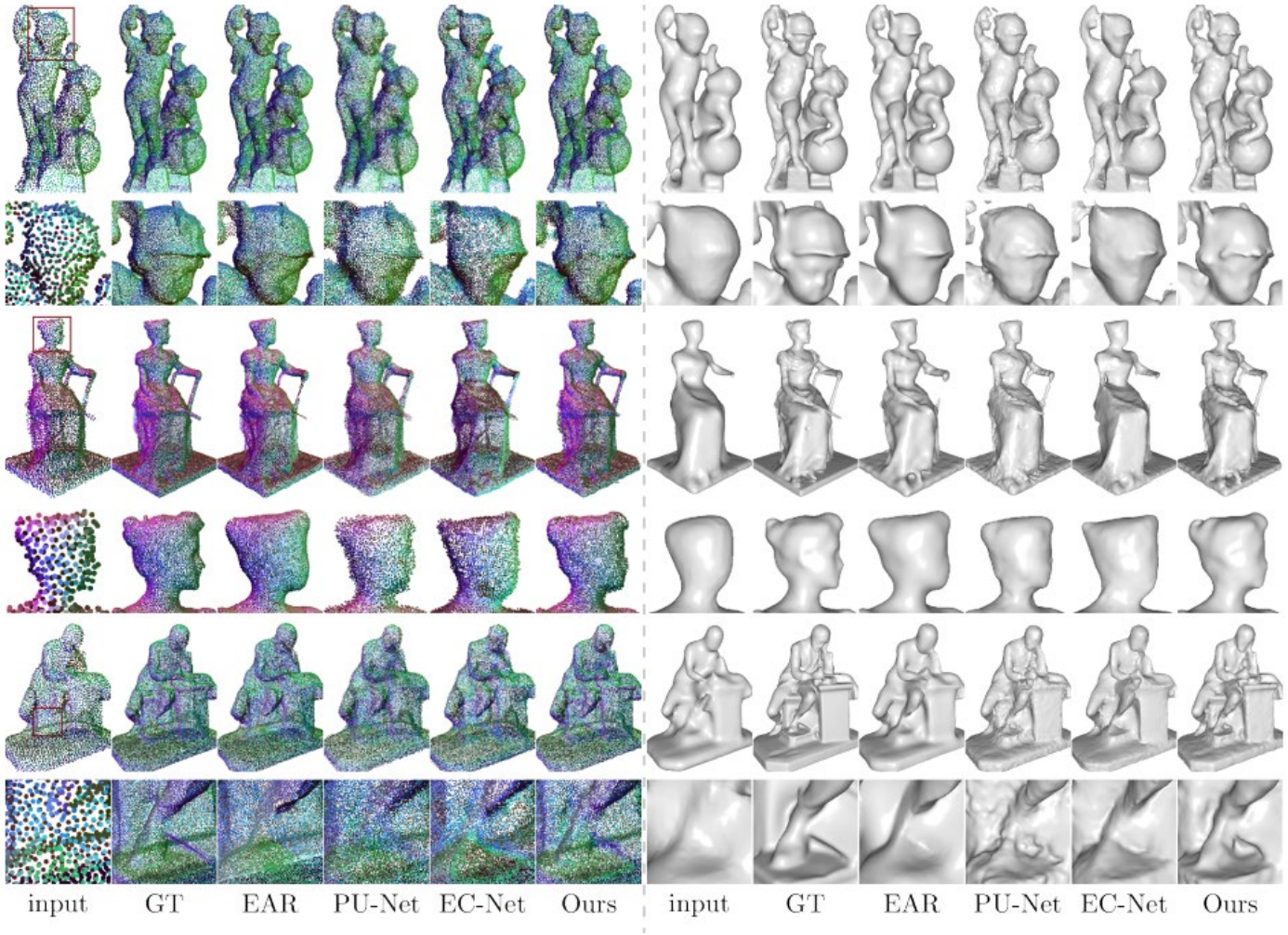


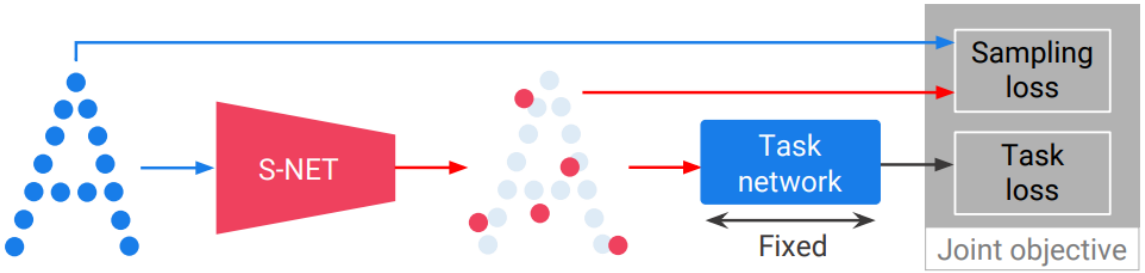
Figure 12: 16× upsampling results from 5000 input points (left) and reconstructed mesh (right).

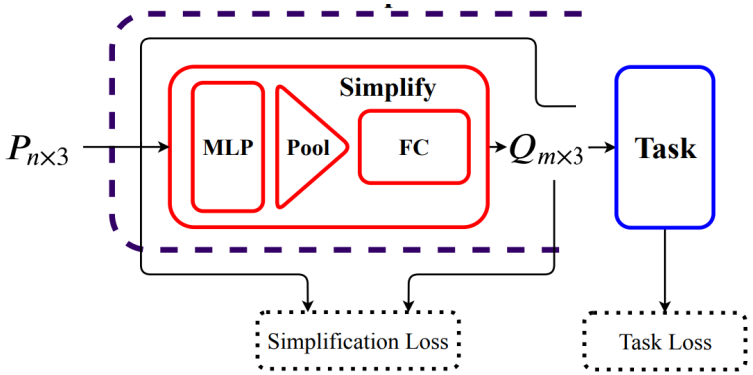
Deep Point Cloud Processing: Down-sampling

[2019-CVPR] Learning to Sample

- Generate sampled points from global features.
- With fixed deep learning task models, the whole framework can be learned end-to-end.
- Minimize joint loss: task loss & Chamfer distance between sampled point cloud G and input point cloud P.

$$L^{S-NET}(G,P) = L_{task}(G) + \alpha L_s(G,P)$$



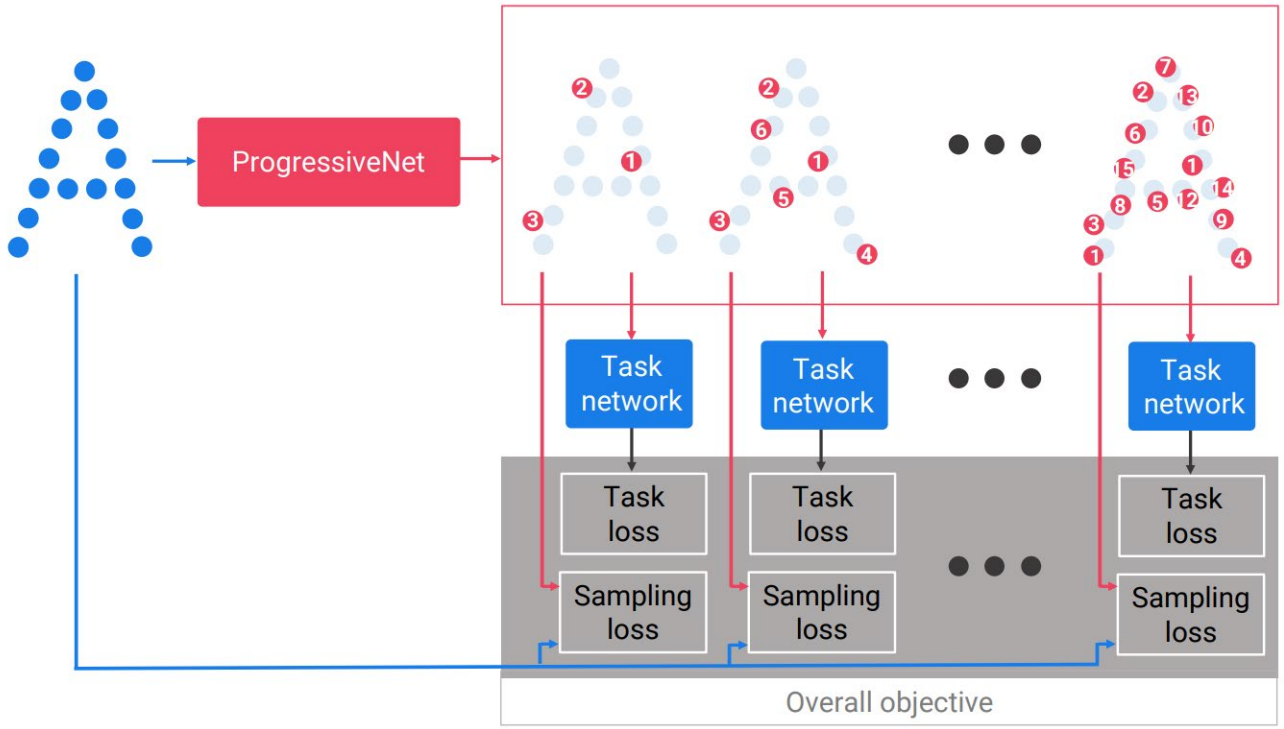


Deep Point Cloud Processing: Down-sampling

[2019-CVPR] Learning to Sample

- **Progressively Down-sampling**:
  - Pick top K points generated from S-Net.
  - Learn the importance ordering for generated sampling points.
  - Impose supervision at each stage.

$$L^{ProgressiveNet}(G, P) = \sum_{c \in C_s} L^{S-NET}(G_c, P))$$



Deep Point Cloud Processing: Down-sampling

[2019-CVPR] Learning to Sample

- **Performance:**
  - Compare S-Net with random sampling & FPS, non-learnable algorithms that are solely based on raw spatial coordinates and fixed selection rules.

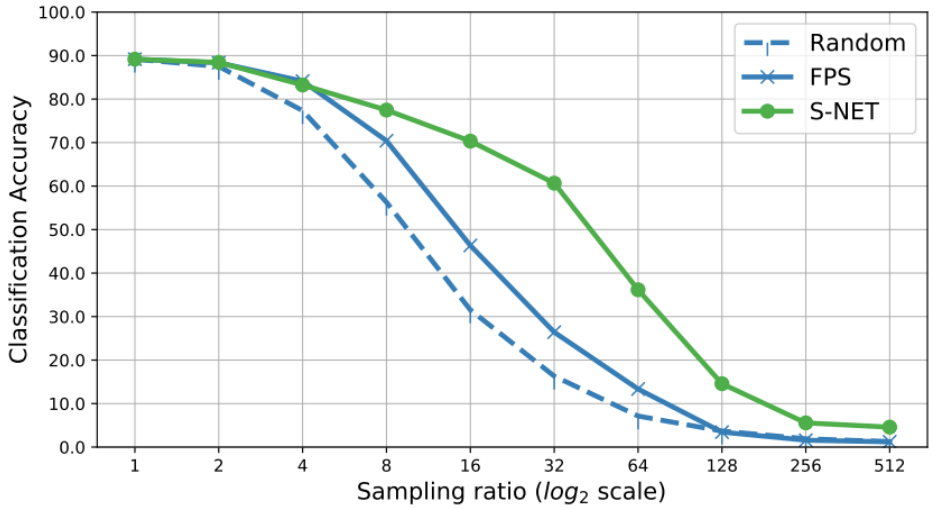


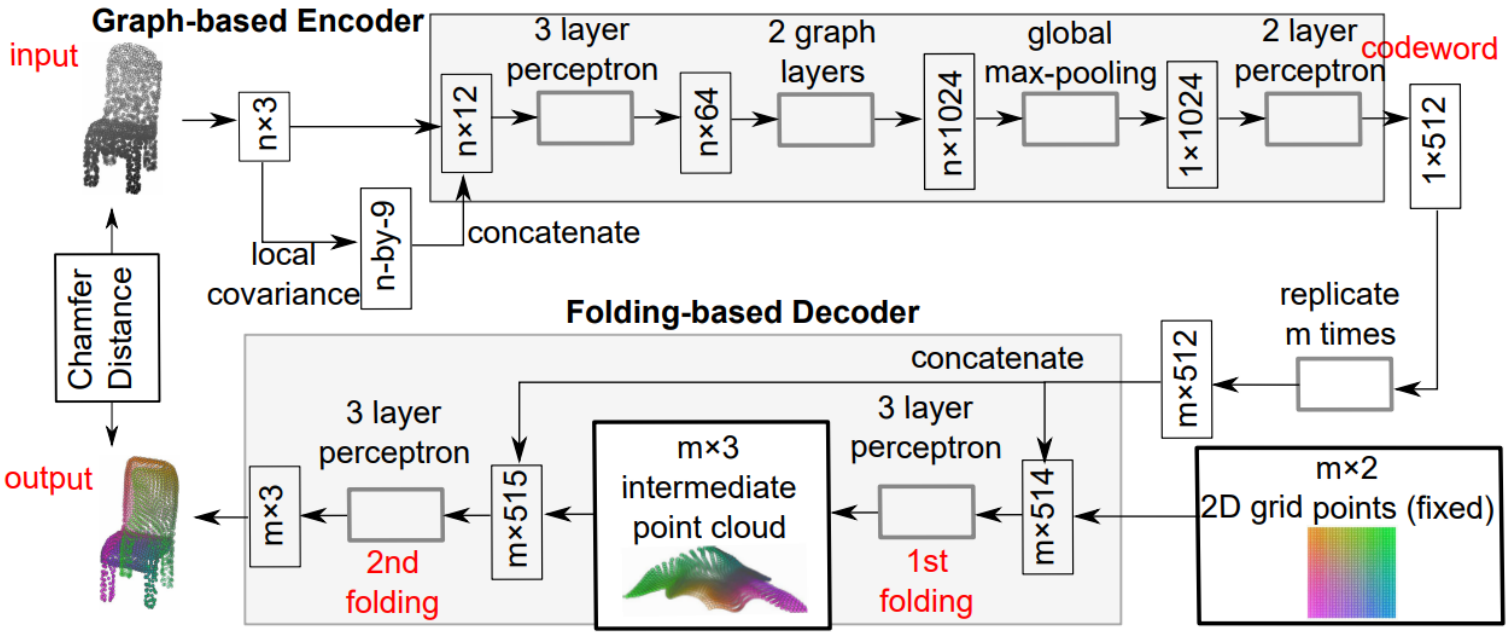
Figure 3. **S-NET for classification.** PointNet was trained on complete point clouds (1024 points) and evaluated on sampled point clouds of the test set using different sampling methods: random, FPS, and S-NET. The accuracy using S-NET is evidently higher.

Deep Point Cloud Processing: Shape Reconstruction

[2018-CVPR] FoldingNet: Point Cloud Auto-Encoder via Deep Grid Deformation

- Investigate an interesting folding operation, which deforms a 2D unit square grid to a complex 3D shape through shared multi-layer perceptrons.

- Folding-based auto-encoder turns to be more effective in unsupervised point cloud feature learning, i.e., representing the most important features of an input point cloud shape with a fixed-length codeword.



Deep Point Cloud Processing: Shape Reconstruction

[2018-CVPR] FoldingNet: Point Cloud Auto-Encoder via Deep Grid Deformation

- Observe how the initial 2D grid gradually transforms into the target 3D shape during the training stage.

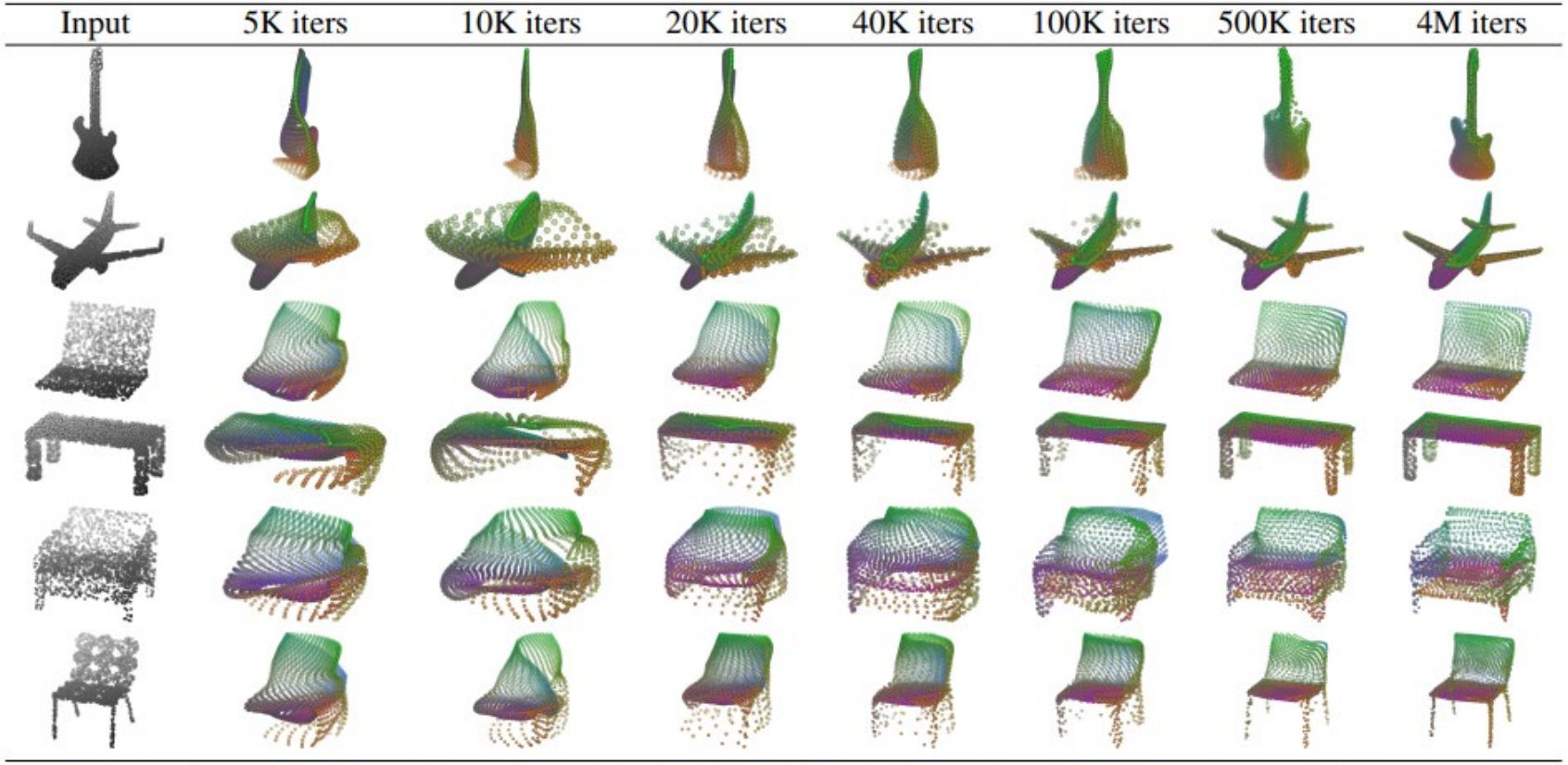


Table 2. Illustration of the training process. Random 2D manifolds gradually transform into the surfaces of point clouds.

Deep Point Cloud Processing: Shape Reconstruction

[2018-CVPR] FoldingNet: Point Cloud Auto-Encoder via Deep Grid Deformation

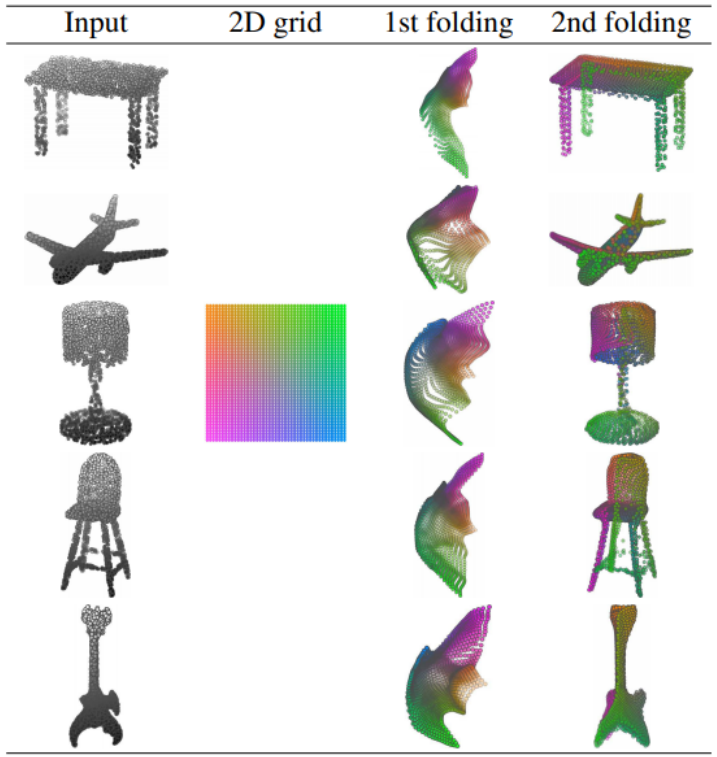


Table 1. **Illustration of the two-step-folding decoding**. Column one contains the original point cloud samples from the ShapeNet dataset [57]. Column two illustrates the 2D grid points to be folded during decoding. Column three contains the output after one folding operation. Column four contains the output after two folding operations. This output is also the reconstructed point cloud. We use a color gradient to illustrate the correspondence between the 2D grid in column two and the reconstructed point clouds after folding operations in the last two columns. Best viewed in color.

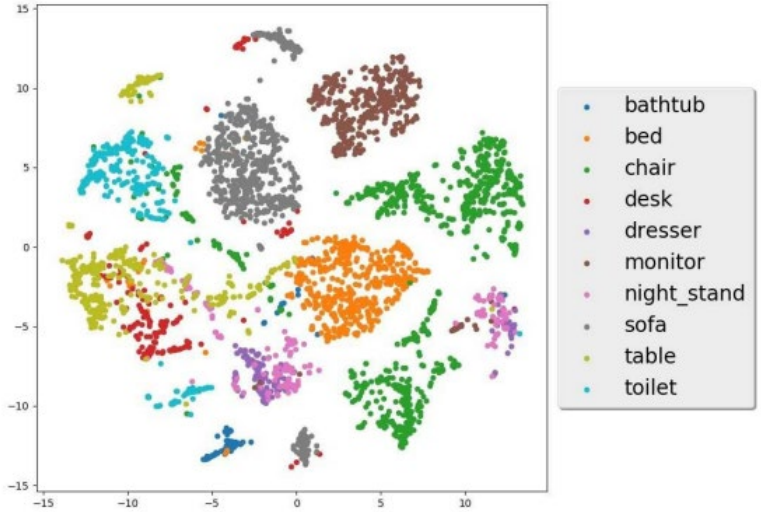


Figure 2. The T-SNE clustering visualization of the codewords obtained from FoldingNet auto-encoder.

Method	MN40	MN10
SPH [26]	68.2%	79.8%
LFD [8]	75.5%	79.9%
T-L Network [19]	74.4%	-
VConv-DAE [45]	75.5%	80.5%
3D-GAN [56]	83.3%	91.0%
Latent-GAN [1]	85.7%	**95.3%**
FoldingNet (ours)	**88.4%**	94.4%

Table 5. The comparison on classification accuracy between FoldingNet and other unsupervised methods. All the methods train a linear SVM on the high-dimensional representations obtained from unsupervised training.

Deep Point Cloud Processing: Shape Reconstruction

[2018-CVPR] AtlasNet: A Papier-Mache Approach to Learning 3D Surface Generation

- Different from FoldingNet that deforms a single 2D grid into 3D shape, AtlasNet uses multiple 2D grids to deform into the target shape.

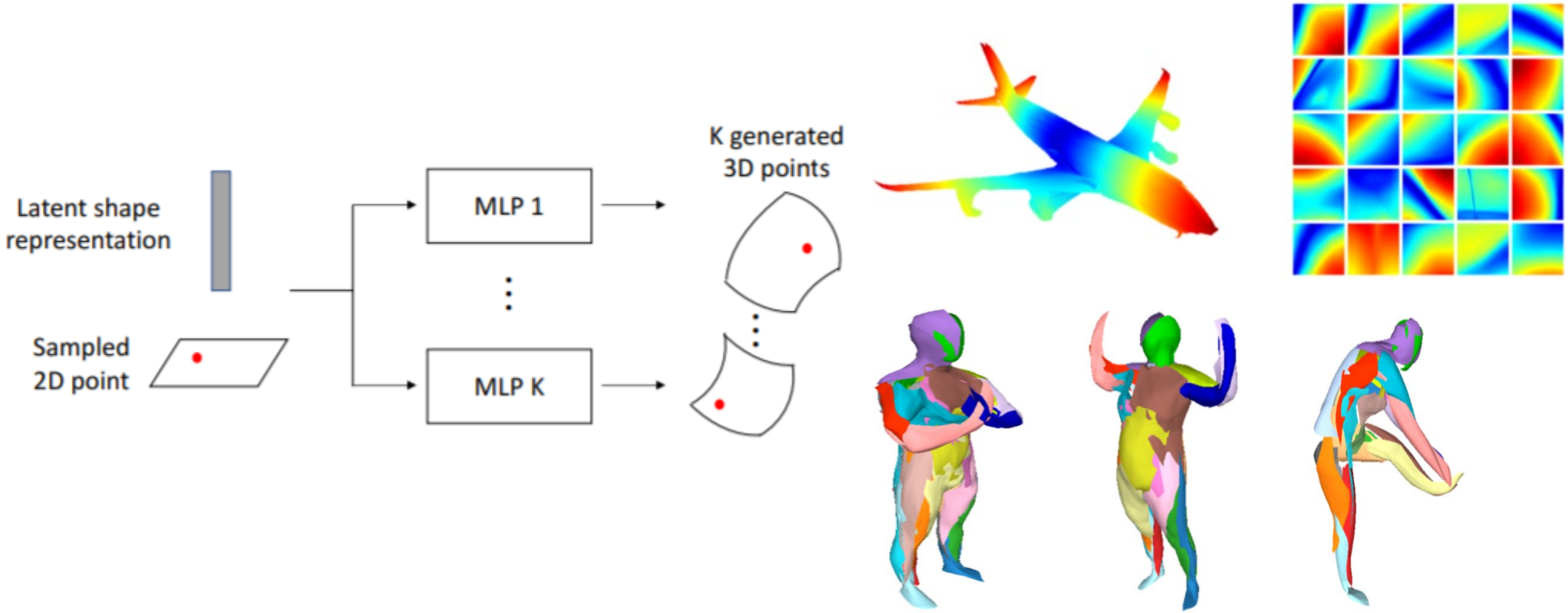


Figure 14. **Deformable shapes.** Our method learned on 250 shapes from the FAUST dataset to reconstructs a human in different poses. Each color represent one of the 25 parametrizations.

Deep Point Cloud Processing: Completion

[Background] and [Challenge]

- Point cloud completion aims to repair the missing region of a point cloud to obtain a complete and high-quality point cloud, which is a kind of "upsampling" under special point distribution.
- Partial known points and points to be inferred are merely related in semantic level and completely different in spatial arrangements. From less information to more information.

[2018-3DV] PCN: Point Completion Network

Deep Point Cloud Processing: Completion

[2020-AAAI] MSN: Morphing and Sampling Network

- Point cloud completion is a kind of upsampling under special point distribution.
- Using morphing-based decoder as the core module of the completion.
- Design Minimum Density Sampling to equalize the distribution of merged points.

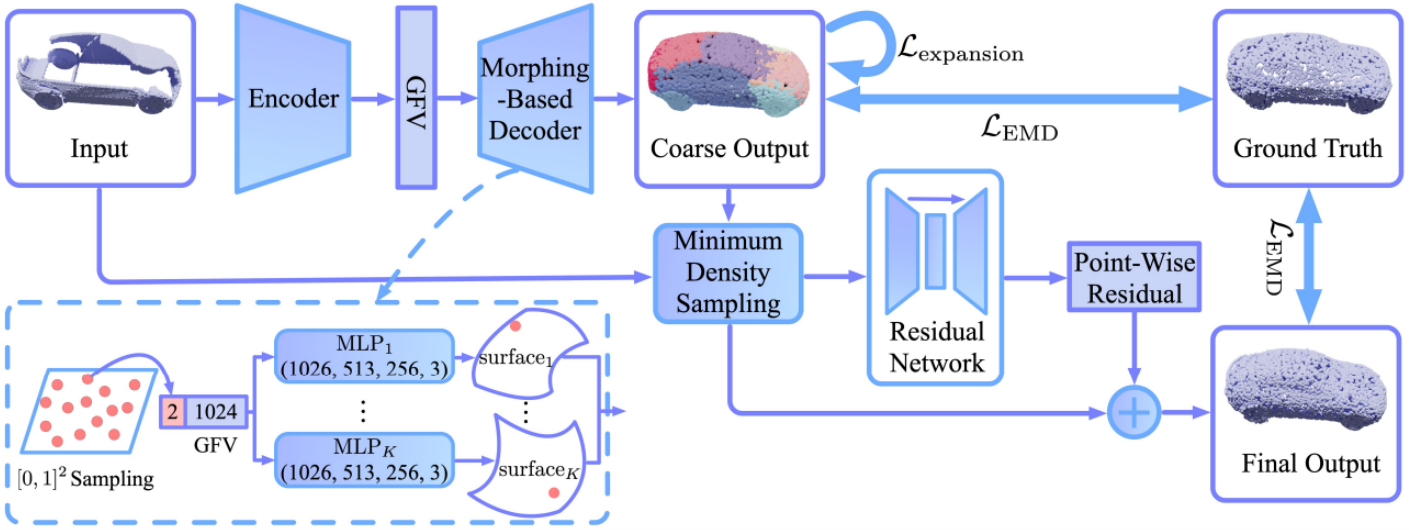


Figure 2: Architecture of our approach. "GFV" denotes the generalized feature vector, $\mathcal{L}_{\text{expansion}}$ and $\mathcal{L}_{\text{EMD}}$ denotes the expansion penalty and Earth Mover's Distance respectively. "$[0, 1]^2$ Sampling" denotes sampling 2D points on a unit square. The morphing-based decoder morphs the unit squares into a collection of surface elements, which are assembled into the coarse output. The minimum density sampling outputs an evenly distributed subset point cloud.

Deep Point Cloud Processing: Completion

[2020-AAAI] MSN: Morphing and Sampling Network

- Motivation: On the basis of the input and the coarse output, the original input fragment is expected to be restored faithfully, thus the coarse output and input fragment will be concatenated (to leverage the original information).

- Dilemma: this will introduce heterogeneous distribution (solved by MDN resampling).

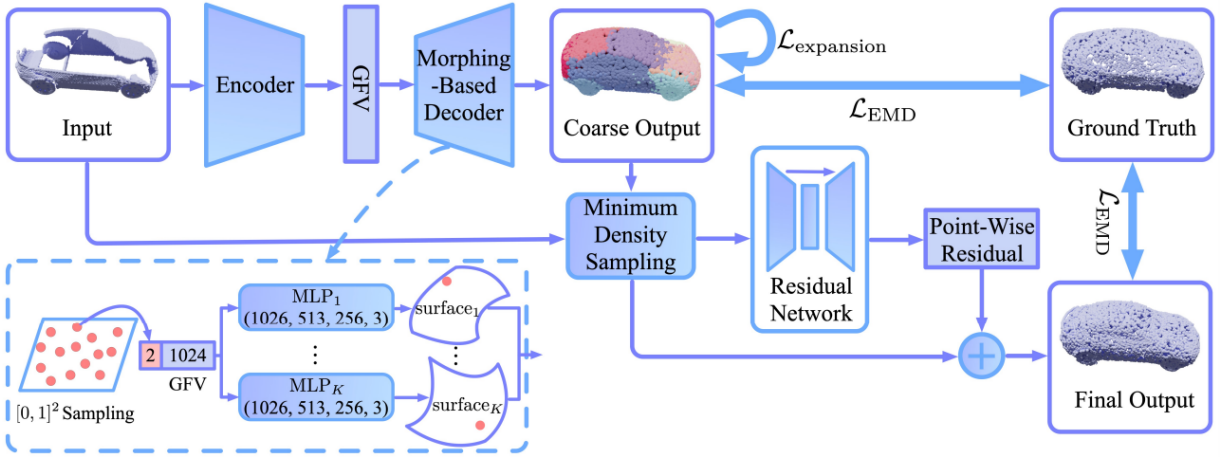
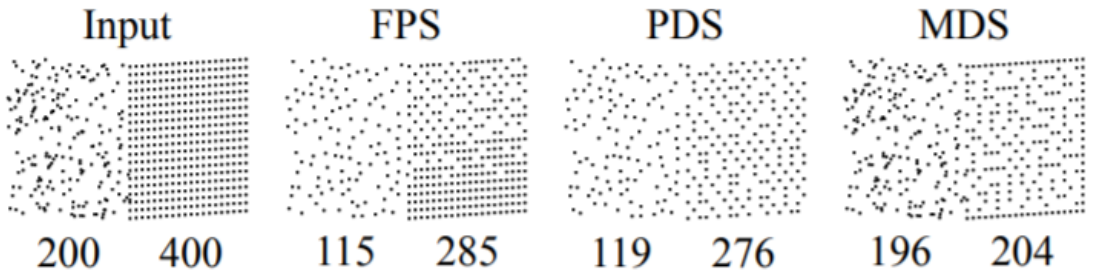


Figure 2: Architecture of our approach. "GFV" denotes the generalized feature vector, $\mathcal{L}_{\text{expansion}}$ and $\mathcal{L}_{\text{EMD}}$ denotes the expansion penalty and Earth Mover's Distance respectively. "$[0, 1]^2$ Sampling" denotes sampling 2D points on a unit square. The morphing-based decoder morphs the unit squares into a collection of surface elements, which are assembled into the coarse output. The minimum density sampling outputs an evenly distributed subset point cloud.

$$p_i = \underset{x \notin \mathrm{P}_{i-1}}{\operatorname{argmin}} \sum_{p_j \in \mathrm{P}_{i-1}} \exp(-\|x - p_j\|^2/(2\sigma^2))$$



香港城市大學
City University of Hong Kong

The End