Emerging Directions in Deep Learning

Learning approaches on point clouds

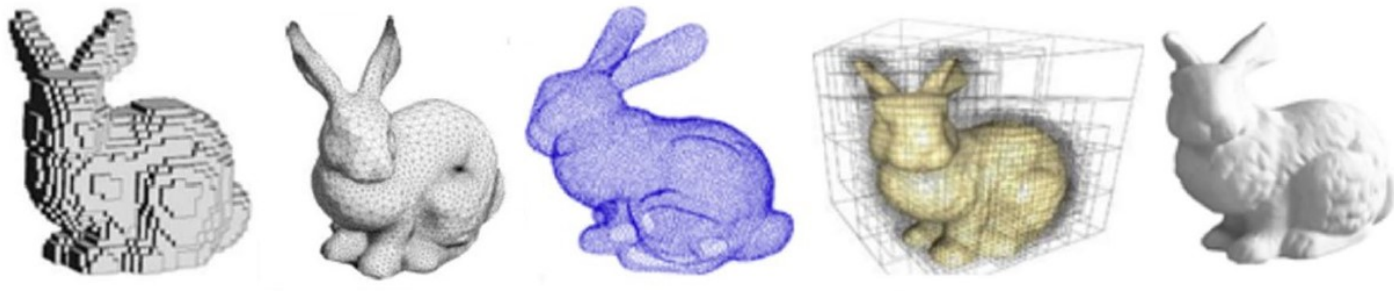
Mihai-Sorin Stupariu, 2025-2026

Data types, tasks: synthesis

- (Non)-Euclidean approaches for 3D data. Depend on the data representation:

Data types, tasks: synthesis

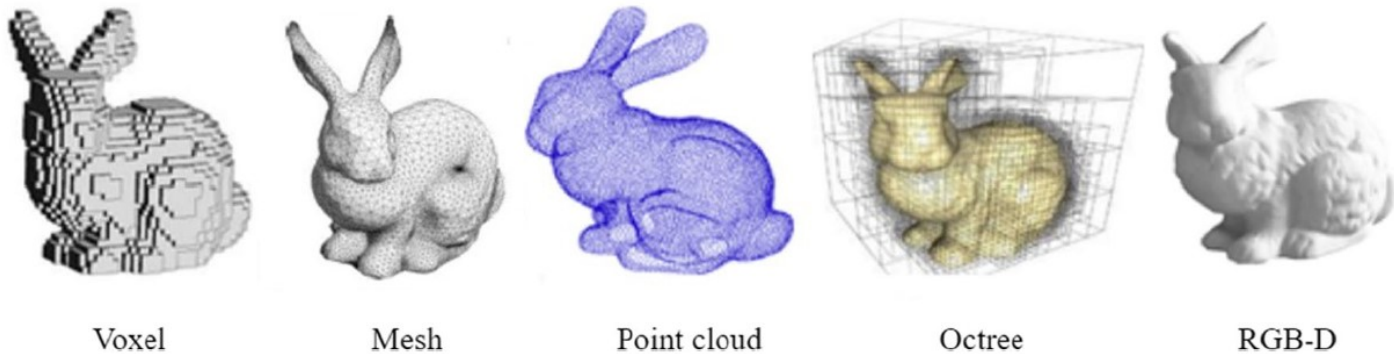
- (Non)-Euclidean approaches for 3D data. Depend on the data representation:
  - multi-view representations
  - voxel-based methods
  - triangle mesh representations
  - point cloud-based approaches
  - RGB-D



Source: [Sarker et al., 2024]

Data types, tasks: synthesis

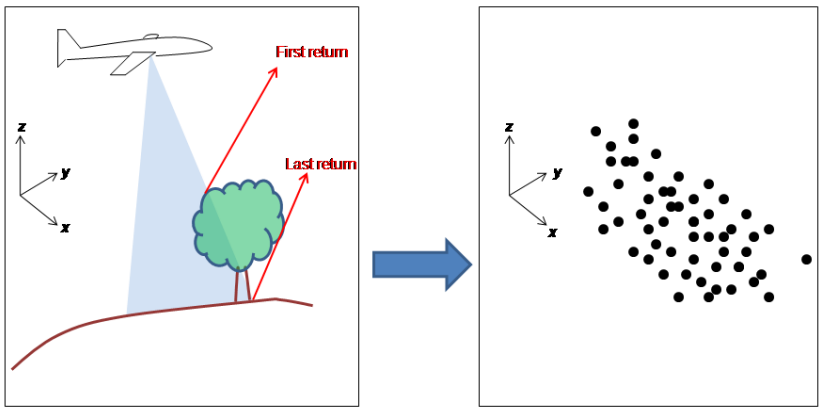
- (Non)-Euclidean approaches for 3D data. Depend on the data representation:
  - multi-view representations
  - voxel-based methods
  - triangle mesh representations
  - point cloud-based approaches
  - RGB-D



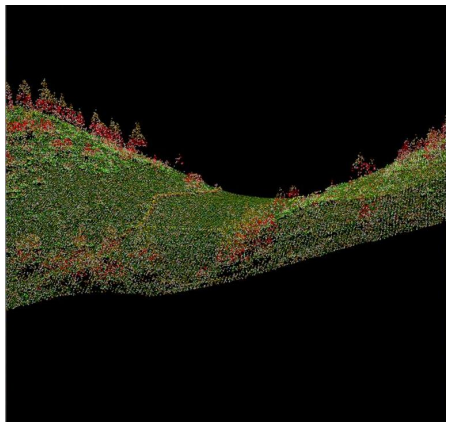
Source: [Sarker et al., 2024]

- The tasks are essentially related to tasks similar to those in Computer Vision.

This lecture is (mainly) about point cloud data. An example



Airborne laser scanning provides a cloud of points situated in the 3D-space (LiDAR data).
Such data is nowadays more and more accessible.



Such data sets contain a lot of information useful in practical problems.

Point clouds

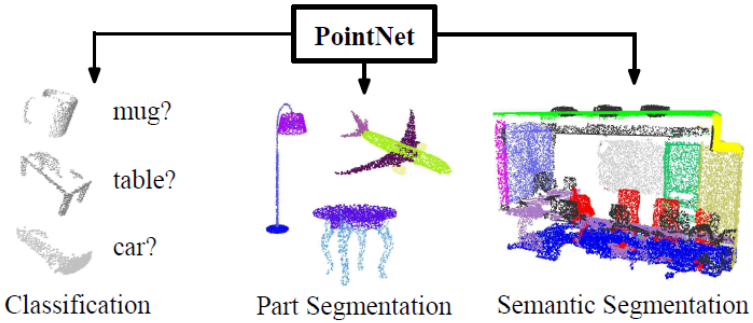


Figure 1. **Applications of PointNet.** We propose a novel deep net architecture that consumes raw point cloud (set of points) without voxelization or rendering. It is a unified architecture that learns both global and local point features, providing a simple, efficient and effective approach for a number of 3D recognition tasks.

Keypoints:

- tasks
- benchmark data sets
- taxonomy of methods
- examples

Point clouds

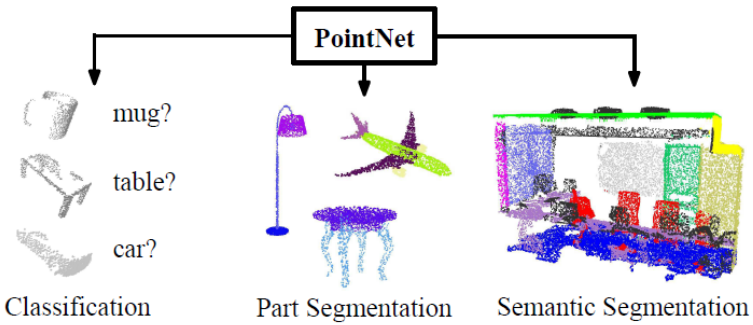


Figure 1. **Applications of PointNet.** We propose a novel deep net architecture that consumes raw point cloud (set of points) without voxelization or rendering. It is a unified architecture that learns both global and local point features, providing a simple, efficient and effective approach for a number of 3D recognition tasks.

Keypoints:

- tasks
- benchmark data sets
- taxonomy of methods
- examples

Presentation is based on:

Sarker et al, 2024, A comprehensive overview of deep learning techniques for 3D point cloud classification and semantic segmentation; see also

Zhang et al., 2023, Deep Learning-based 3D Point Cloud Classification: A Systematic Survey and Outlook

Tasks

- High-level tasks

Tasks

- High-level tasks
  - **classification**
  - **segmentation**

Tasks

- High-level tasks
  - **classification**
  - **segmentation**
  - object detection
  - object tracking
  - registration
  - point cloud video understanding

Tasks

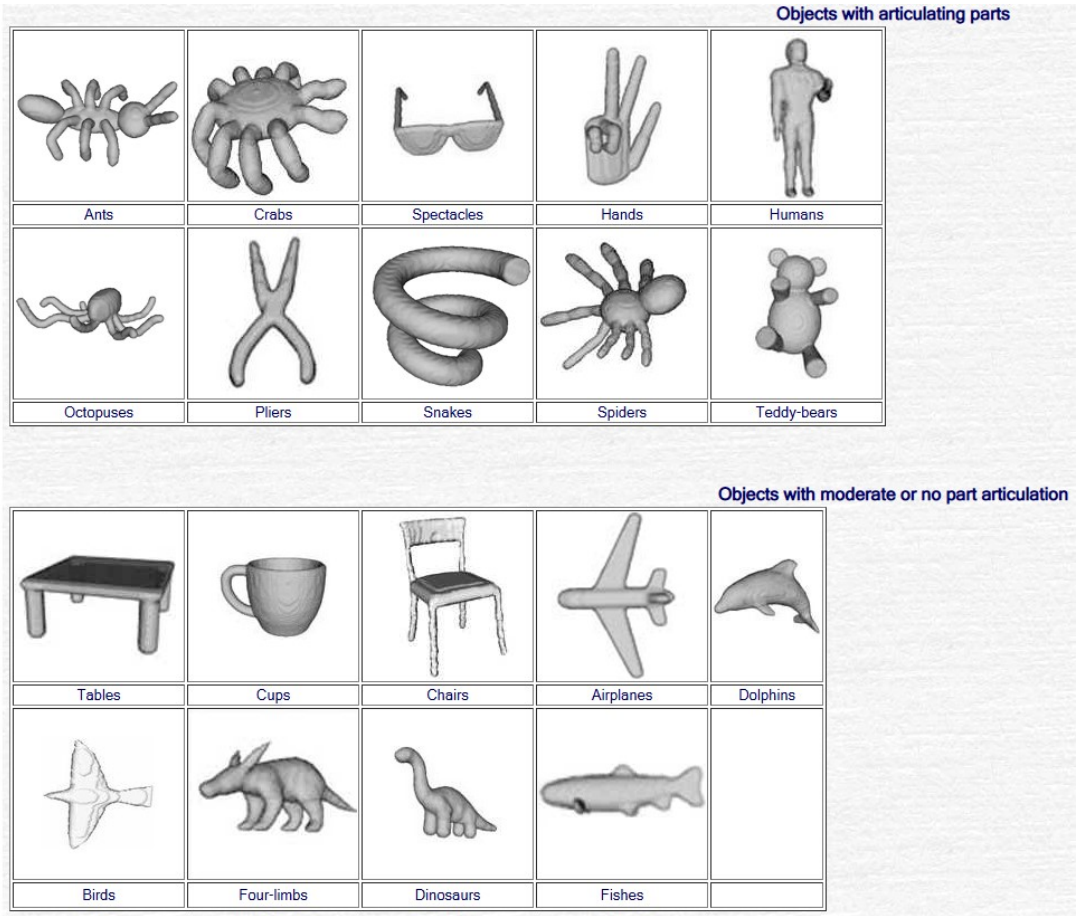
- High-level tasks
  - **classification**
  - **segmentation**
  - object detection
  - object tracking
  - registration
  - point cloud video understanding
- Low level tasks
  - downsampling
  - upsampling
  - denoising
  - completion

Classification: reference data sets

Dataset	Year	Type	Format
McGill 3D Shape Benchmark	2005	Synthetic	Polygonal mesh/Voxels
ModelNet10	2015	Synthetic	Polygonal mesh
ModelNet40	2015	Synthetic	Polygonal mesh
ModelNet40-C	2015 / 2022	Synthetic	Point cloud
ShapeNet	2015	Synthetic	Polygonal mesh
Object scans	2016	RW, indoor	RGB-D
ScanNet	2017	RW, indoor	RGB-D
ScanObjectNN	2019	RW, indoor	Point cloud
Sydney Urban Objects	2013	RW, outdoor	Point cloud
Paris-rueMadame	2014	RW, outdoor	Point cloud
Paris-Lille-3D	2017	RW, outdoor	Point cloud

Classification: reference data sets

The McGill 3D Shape Benchmark ($\sim$ 2005)



Source: McGill 3D Shape Benchmark

Classification: reference data sets

ModelNet (2015)

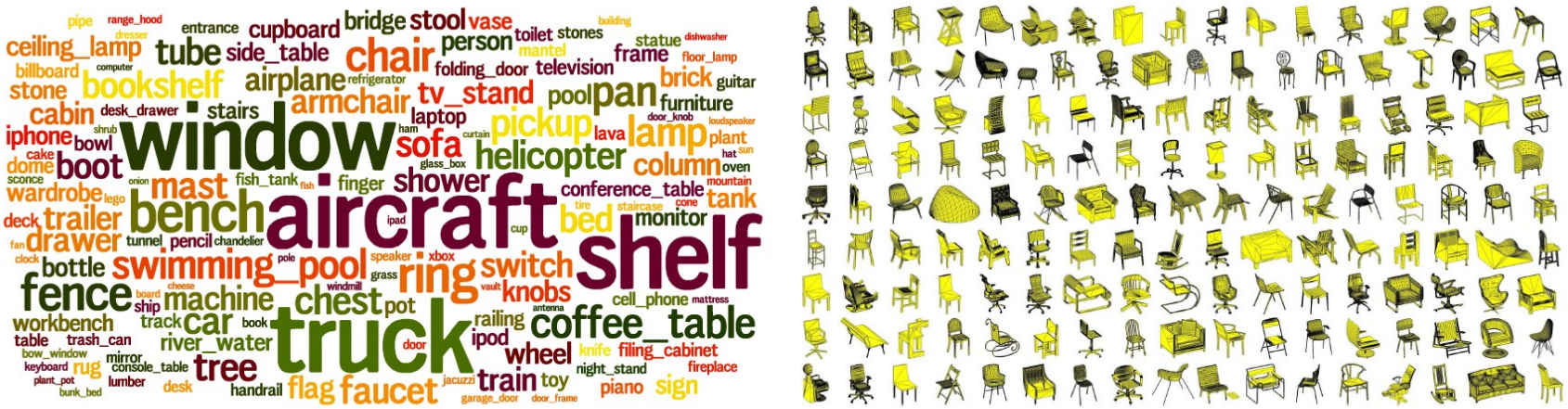
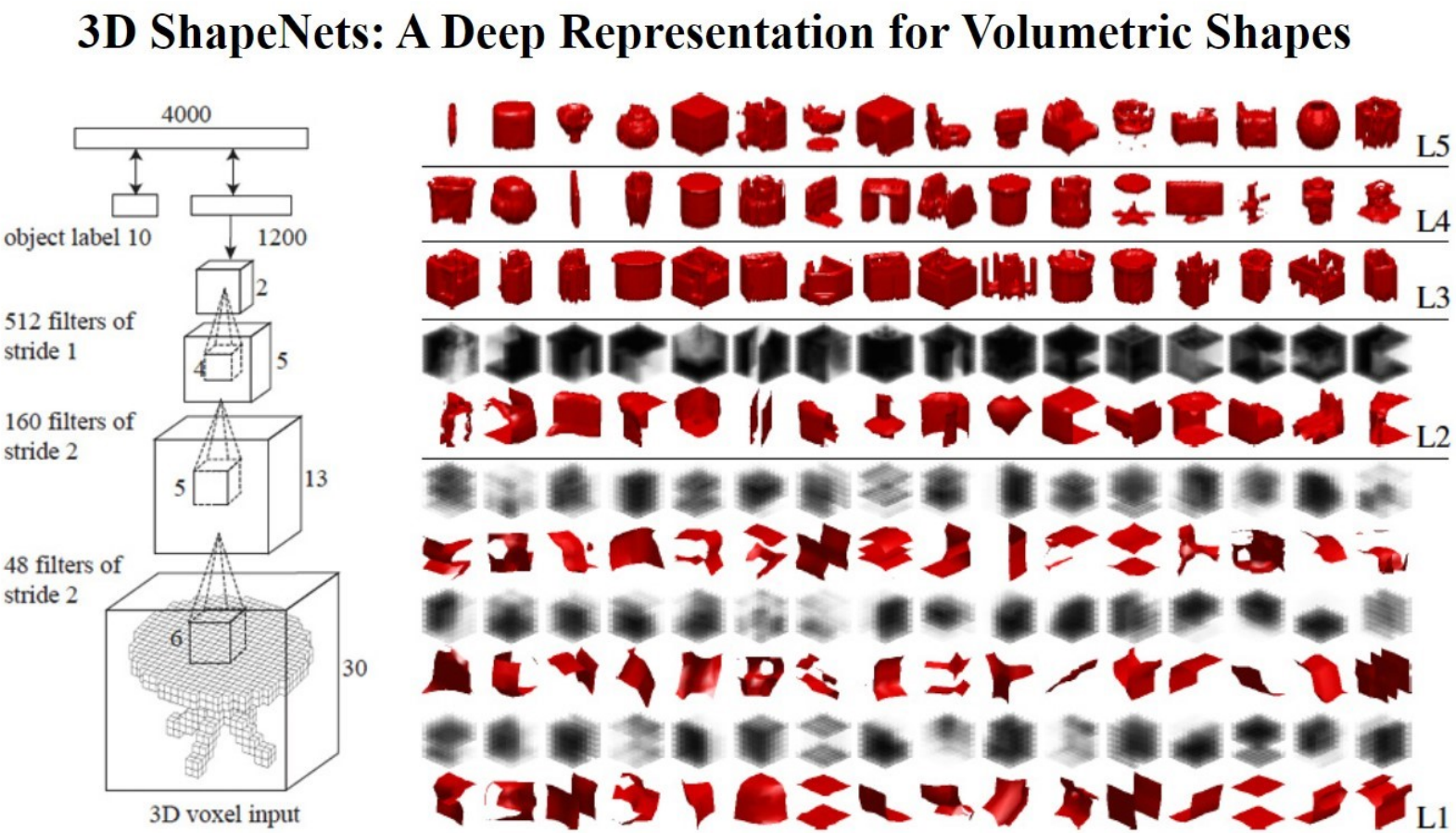


Figure 5: **ModelNet Dataset.** Left: word cloud visualization of the ModelNet dataset based on the number of 3D models in each category. Larger font size indicates more instances in the category. Right: Examples of 3D chair models.

Source: [Wu et al., 2015]

Classification: reference data sets

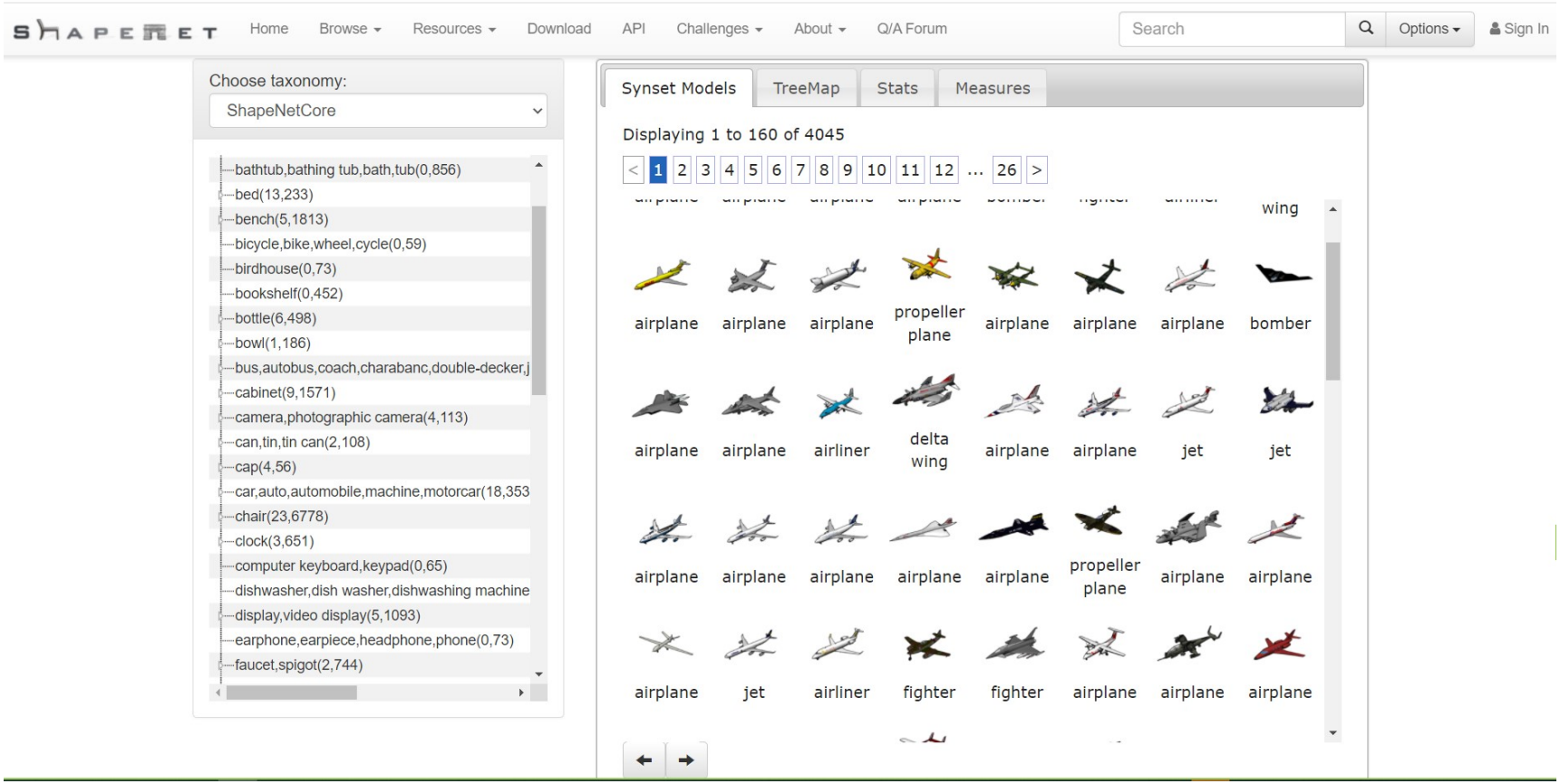
ModelNet (2015)



Source: ModelNet / 3D ShapeNets

Classification: reference data sets

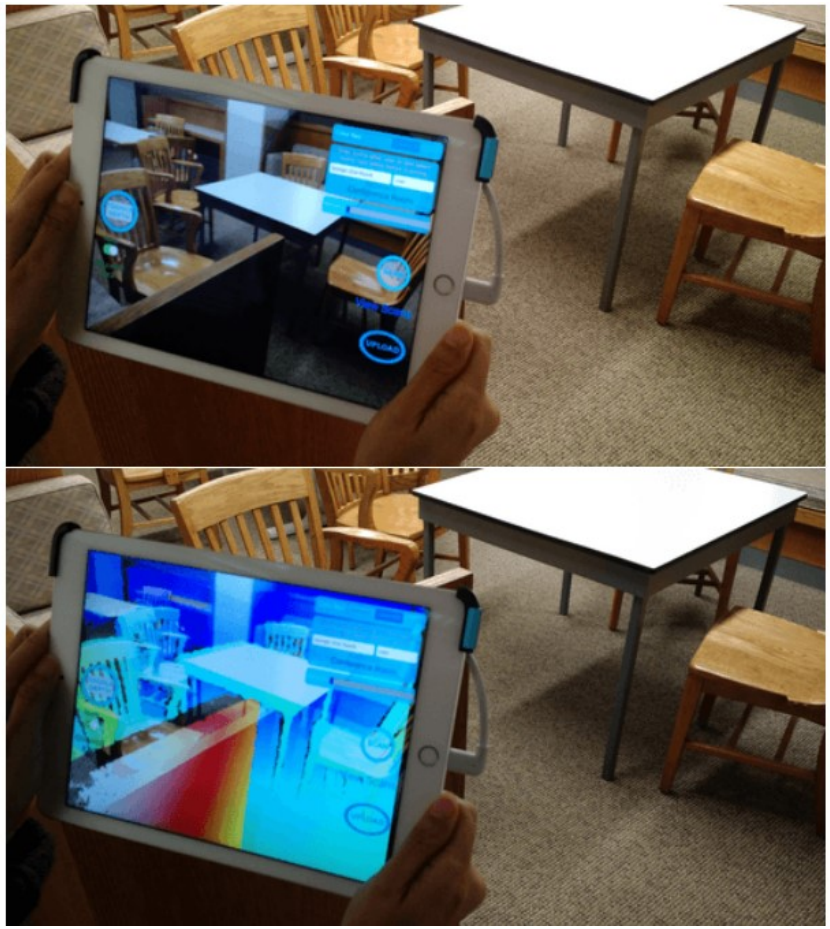
ShapeNet (2015)



Source: The ShapeNet repository

Classification: reference data sets

ScanNet (2017)



Source: Dai et al., 2017

Classification: reference data sets

ScanObjectNN (2019)

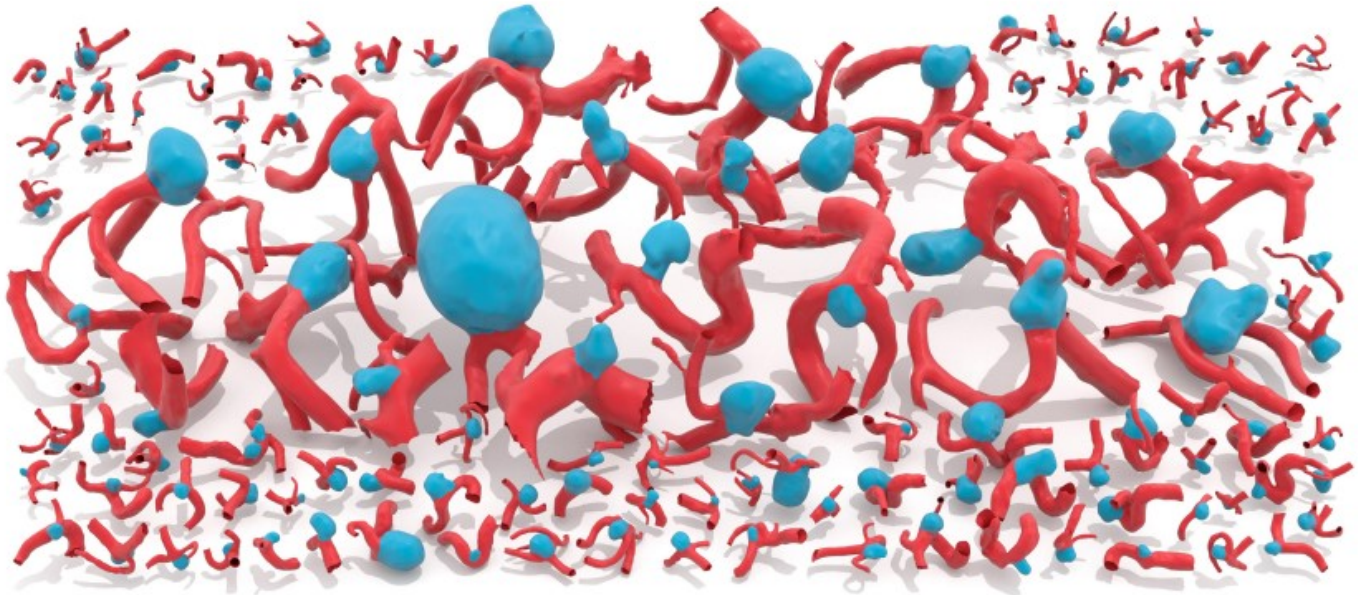


Sample objects from our ScanObjectNN dataset. The dataset contains ~15,000 objects that are categorized into 15 categories with 2902 unique object instances. The raw objects are represented by a list of points with global and local coordinates, normals, colors attributes and semantic labels. We also provide part annotations, which to the best of our knowledge is the first on real-world data.

Source: https://hkust-vgd.github.io/scanobjectnn/

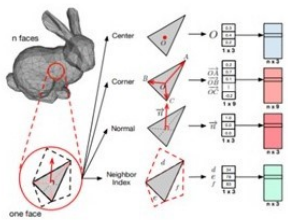
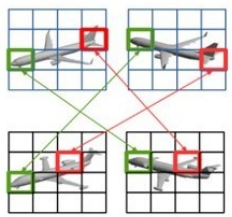
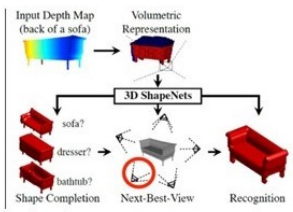
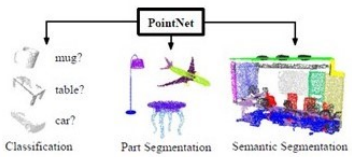
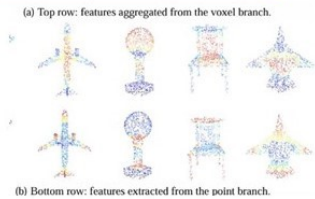
Classification: reference data sets

IntrA: 3D Intracranial Aneurysm Dataset for Deep Learning (2020)



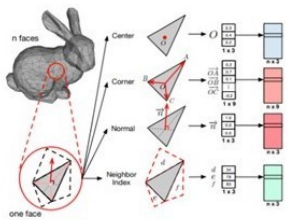
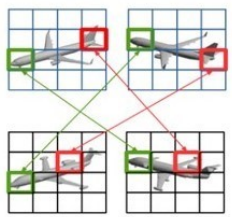
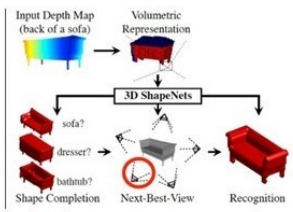
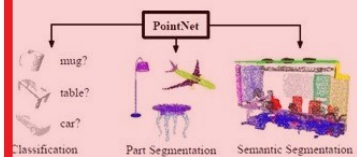
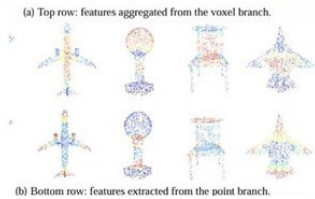
Source: Yang et al., 2020

Classification: taxonomy of DL methods

Mesh based methods	Projection based methods	Volumetric based methods	Raw point based methods	Hybrid methods
				
Example: MeshNet, 2018	Example: MHBN, 2018	Example: 3DShapeNets,2015	Example: PointNet, 2015	Example: PVT, 2021

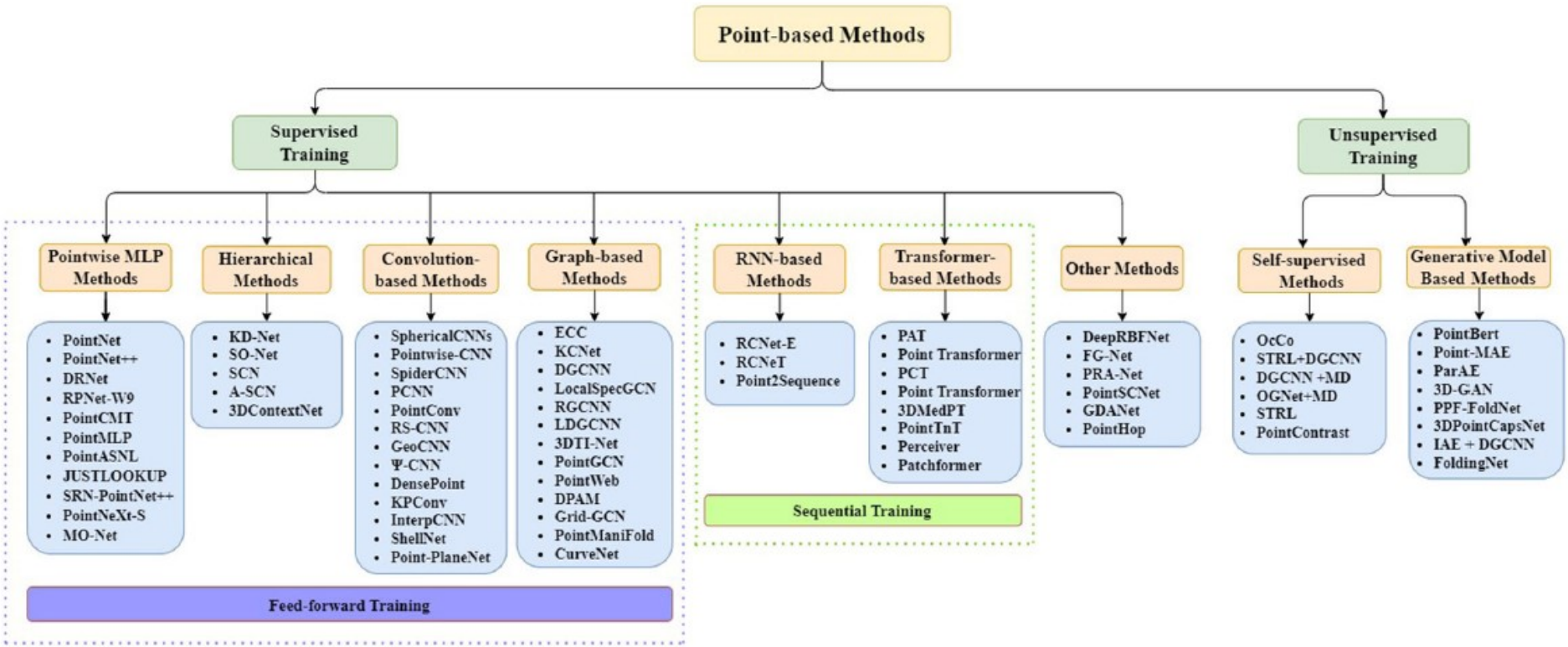
Adapted from [Sarker et al., 2024]

Classification: taxonomy of DL methods

Mesh based methods	Projection based methods	Volumetric based methods	Raw point based methods	Hybrid methods
				
Example: MeshNet, 2018	Example: MHBN, 2018	Example: 3DShapeNets,2015	Example: PointNet, 2015	Example: PVT, 2021

Adapted from [Sarker et al., 2024]

Classification: taxonomy of point-based DL methods



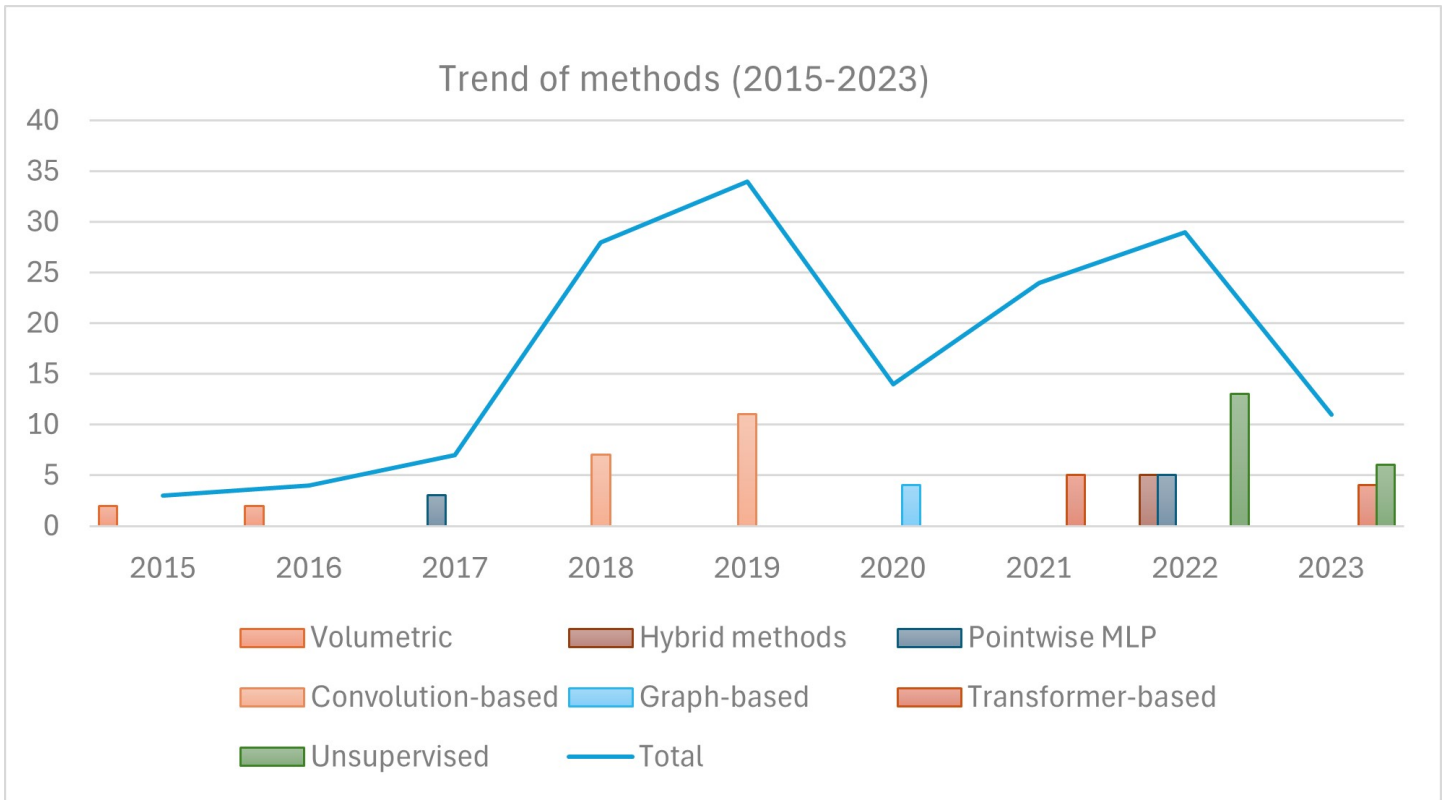
Source: [Sarker et al., 2024]

Classification: leaderboard (synthesis)

Model name	Year	ModelNet40 OA	ModelNet40 mAcc	ModelNet10 OA	ModelNet10 mAcc	ScanObjectNN OA	mAcc
Point-View GCN (hybrid method)	2021	**95.40%**	-	-	-	85.50%	-
VRN Ensemble (volumetric)	2016	-	**95.54%**	-	**97.14%**	-	-
CNN+LSTM+Voting (projection-based)	2019	91.05%	-	**95.29%**	-	-	-
PointCMT (hybrid method)	2022	94.40%	91.20%	-	-	**86.40%**	**84.40%**
PointGPT-L (8kP) (unsupervised)	2023	**94.90%**	-	-	-	-	-
FG-Net (other methods)	2020	93.80%	**93.10%**	-	-	-	
Grid-GCN (graph-based)	2020	93.10%	91.30%	**97.50%**	**97.40%**	-	-
PointGPT-L (1kP) (unsupervised)	2023	94.70%	-	-	-	**93.40%**	-
P2P (8kP) (unsupervised)	2023	94.00%	91.60%	-	-	89.30%	**88.50%**

Statistics cf. [Sarker et al., 2024]

Classification: trends of the number of papers and of the top 1 methods



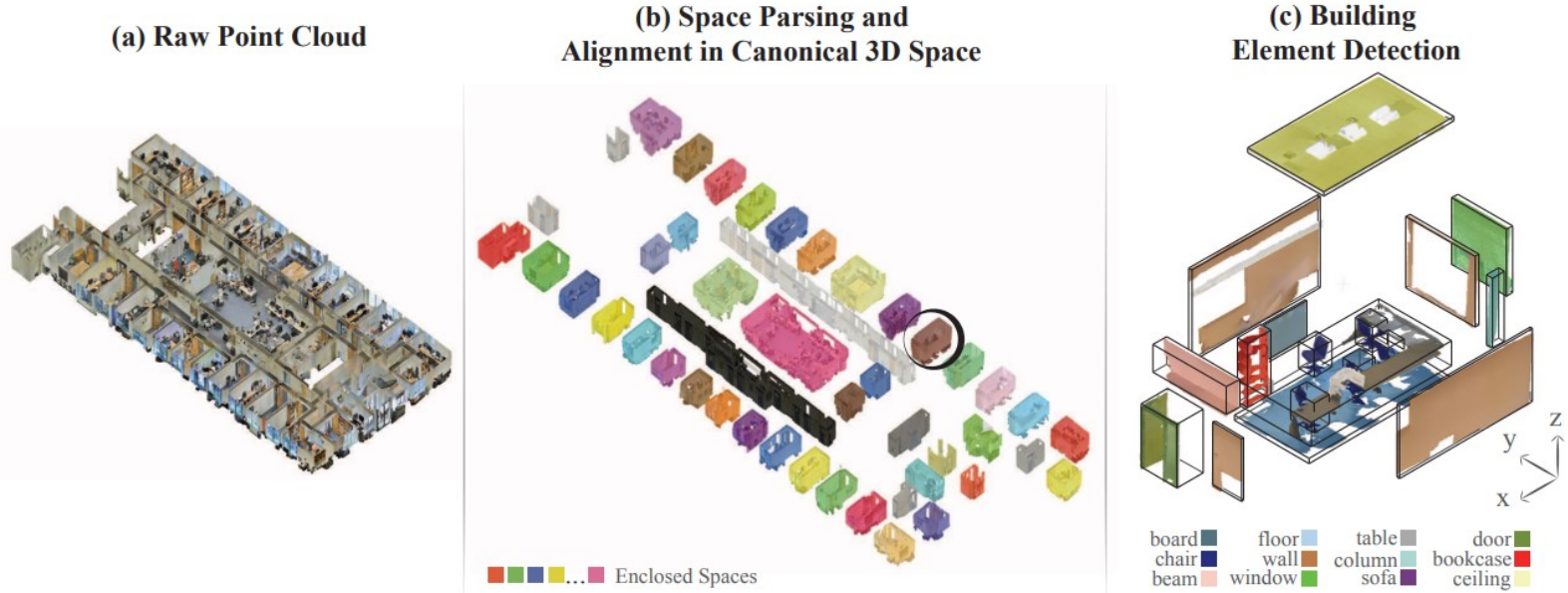
Statistics cf. [Sarker et al., 2024]

Segmentation: reference data sets

Dataset	Year	Type	Format
SynLIDAR	2022	Synthetic	Point cloud
STPLS3D-SyntheticV1	2022	Synthetic	Point cloud/UAV
ScanNet	2017	RW, indoor	RGB-D
S3DIS	2017	RW, indoor	Point cloud
Oakland	2009	RW, outdoor	Point cloud
IQmulus	2013	RW, outdoor	Point cloud
Paris-rueMadame	2014	RW, outdoor	Point cloud
Semantic3D	2017	RW, outdoor	Point cloud
SemanticKITTI	2019	RW, outdoor	Point cloud
Paris-Lille-3D	2017	RW, outdoor	Point cloud
RELLIs-3D	2021	RW, outdoor	Point cloud
WADS	2021	RW, outdoor	Point cloud
STPLS3D-Real1	2022	RW, outdoor	Point cloud/UAV
FOR-instance	2023	RW, outdoor	Point cloud

Segmentation: reference data sets

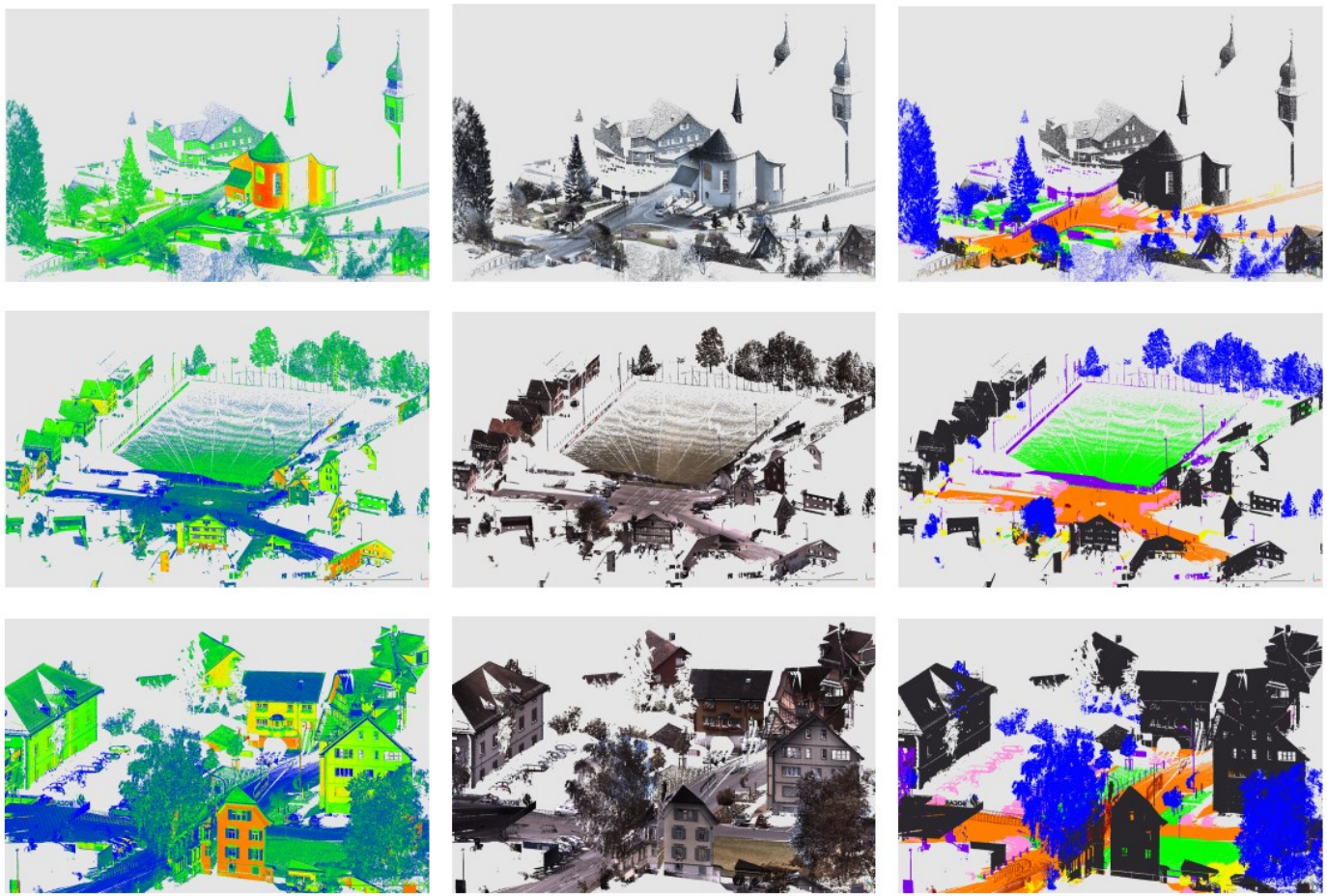
S3DIS (2016)



Source: Armeni et al., 2016

Segmentation: reference data sets

Semantic 3D (2017)



Source: Hackel et al., 2017

Segmentation: reference data sets

Semantic 3D (2017)



Source: Hackel et al., 2017

Segmentation: reference data sets

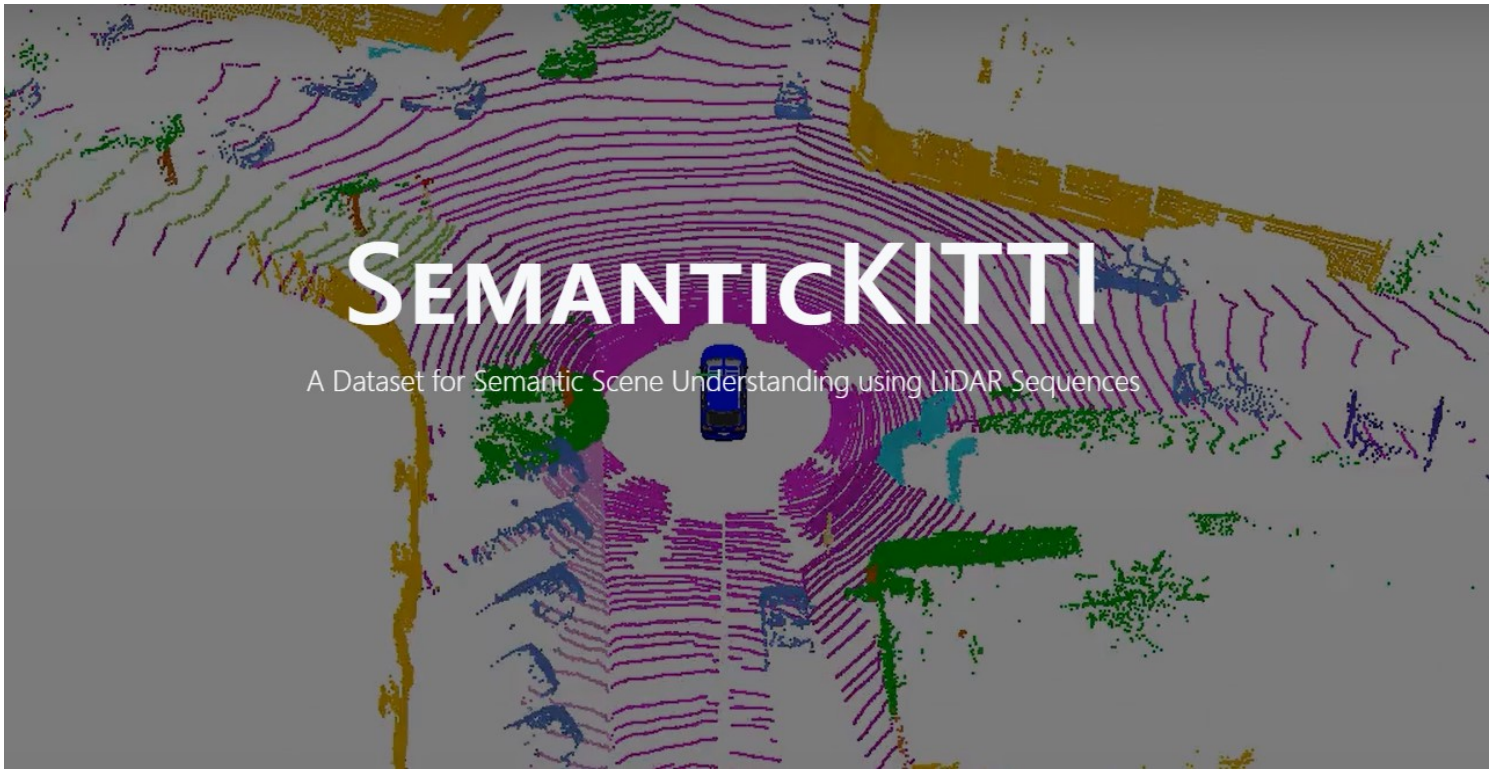
Paris-Lille-3D (2018)



Source: https://npm3d.fr/paris-lille-3d

Segmentation: reference data sets

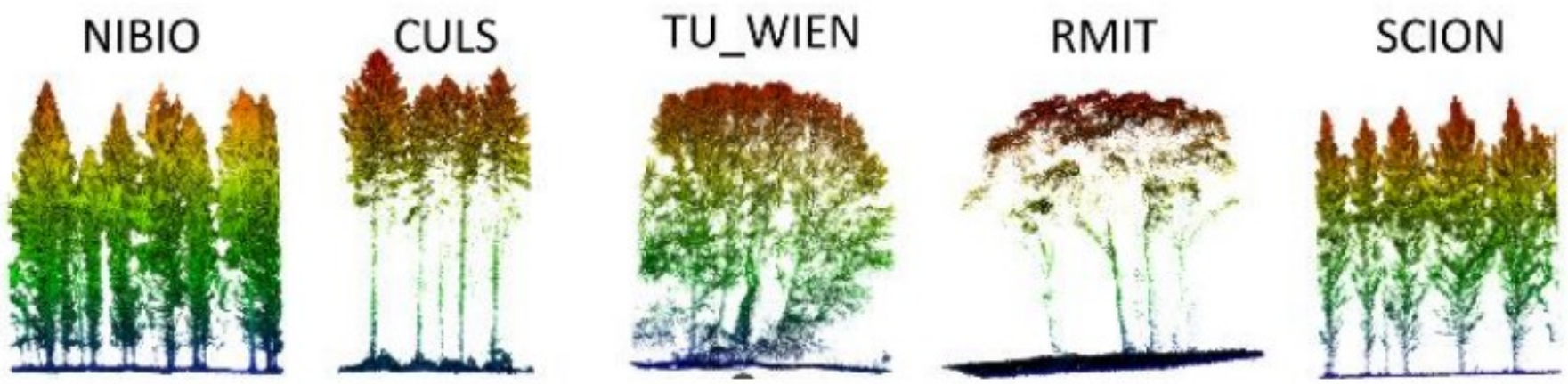
Semantic KITTI (2019)



Source: https://semantic-kitti.org/

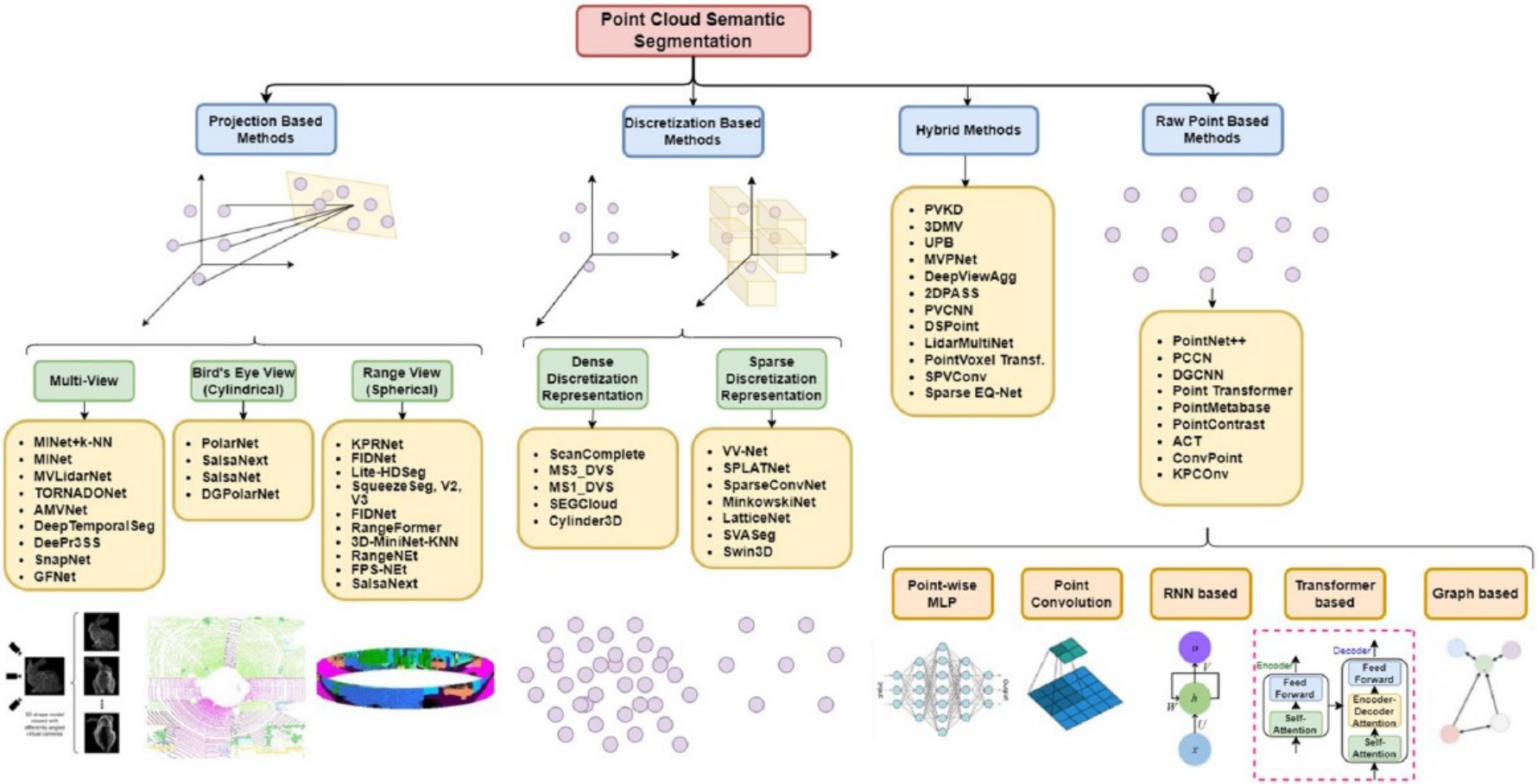
Segmentation: reference data sets

FOR-instance (2023)



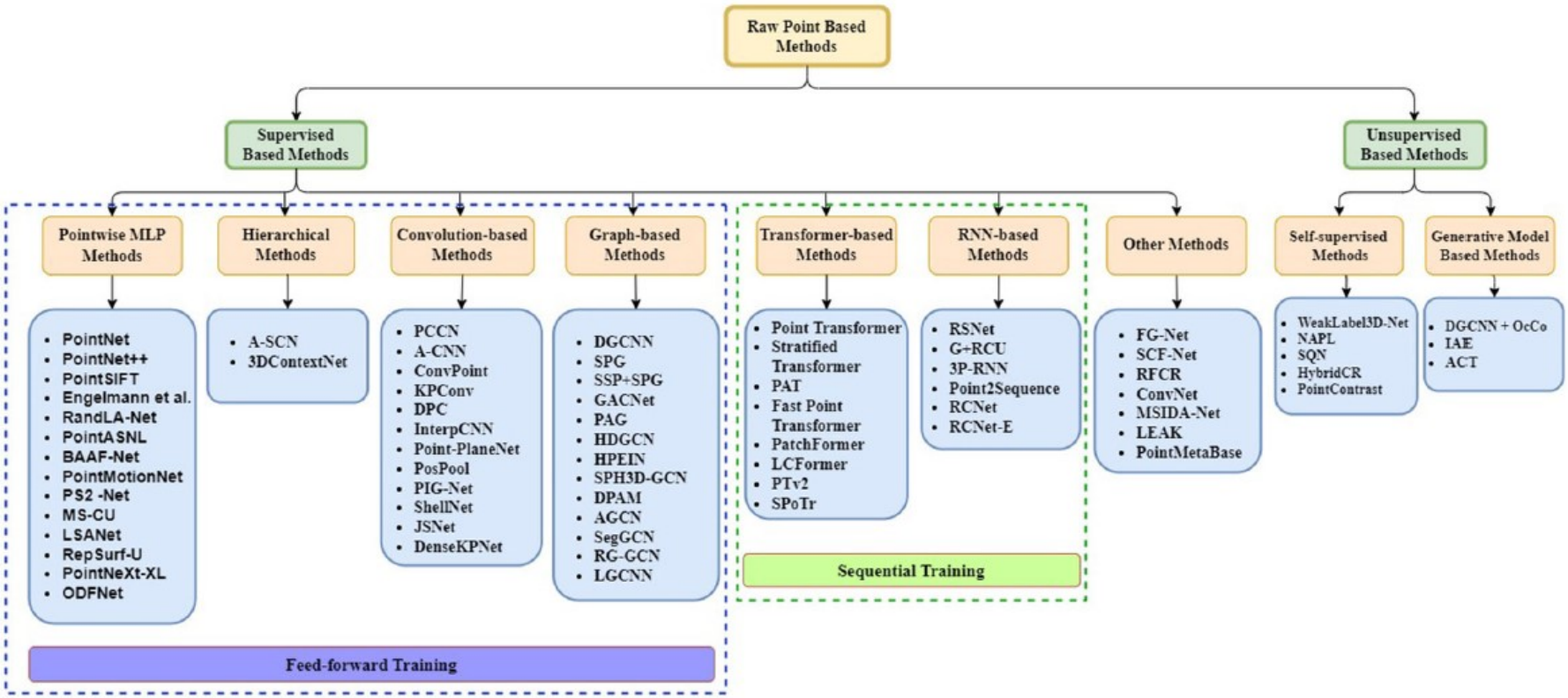
Source: Puliti et al., 2023

Segmentation: taxonomy of DL methods



Source: [Sarker et al., 2024]

Segmentation: taxonomy of point-based DL methods



Source: [Sarker et al., 2024]

Segmentation: leaderboard (synthesis - I)

Model name	Year	S3DIS Area5 OA	S3DIS Area5 mIoU	S3DIS 6-fold OA	S3DIS 6-fold mIoU	ScanNet OA	ScanNet mIoU	Sem. 3D OA	Sem. 3D mIoU	Sem. KITTI mIoU
PVCNN (hybrid)	2019	**86.87%**	57.63%	-	-	-	-	-	-	39.00%
Swin3D-L (discretiz.)	2023	-	**74.50%**	-	**79.80%**	-	**77.90%**	-	-	-
Pt-Vox Trsf. (hybrid)	2022	-	68.20%	**88.30%**	69.20%	-	-	-	-	64.90%
PMF (hybrid)	2021	-	-	-	-	**76.90%**	-	-	-	63.90%
DeePr3SS (proj.)	2017	-	-	-	-	-	-	**88.90%**	58.50%	-
MS3_DVS (discretiz.)	2018	57.93%	46.32%	-	-	-	-	88.40%	**65.30%**	-
RangeFormer (proj.)	2023	-	-	-	-	-	-	-	-	**73.30%**

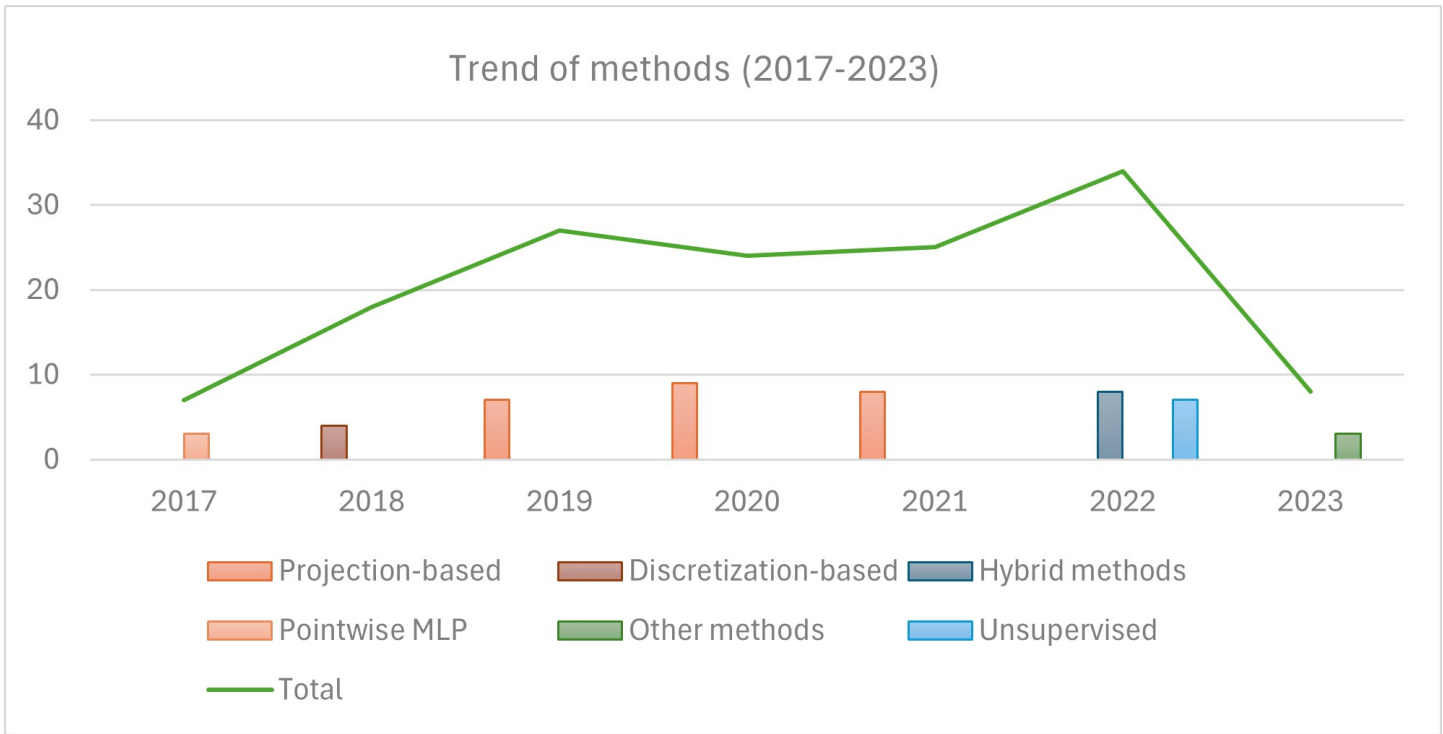
Statistics cf. [Sarker et al., 2024]

Segmentation: leaderboard (synthesis - II)

Model name	Year	S3DIS Area5 OA	S3DIS Area5 mIoU	S3DIS 6-fold OA	S3DIS 6-fold mIoU	ScanNet OA	ScanNet mIoU	Sem. 3D OA	Sem. 3D mIoU	Sem. KITTI mIoU
Strat.Trsf. (transf.)	2022	**91.50%**	72.00%	-	-	-	73.70%	-	-	-
Strat.Trsf. +PAGWN (transf.)	2022	91.40%	**72.20%**	**91.70%**	**77.60%**	-	-	-	-	-
PS^2-Net (pt.-wise)	2020	84.60%	52.95%	88.22%	66.60%	**87.21%**	44.90%	-	-	-
Bound.L (unsuperv.)	2022	90.10%	69.30%	90.20%	73.50%	-	**75.80%**	-	-	-
LGS-Net (other)	2022	-	-	88.90%	71.70%	-	-	**95.30%**	**79.00%**	-
Pt.MotionNet (pt.-wise)	2022	-	-	-	-	-	-	-	-	**81.80%**

Statistics cf. [Sarker et al., 2024]

Segmentation: trends of the number of papers and of the top 1 methods



Statistics cf. [Sarker et al., 2024]

PointNet - overview

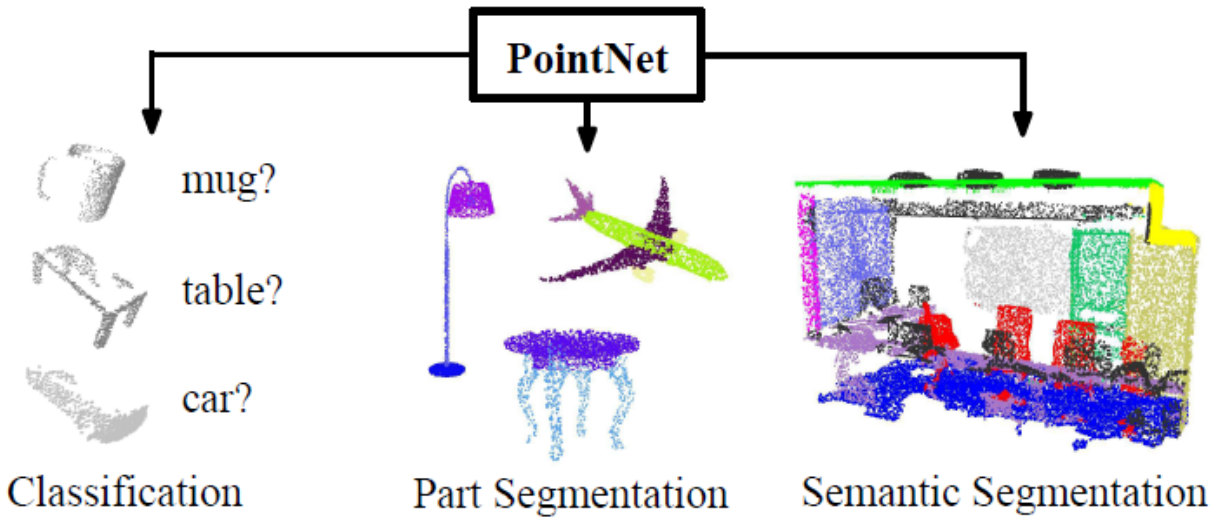


Figure 1. **Applications of PointNet.** We propose a novel deep net architecture that consumes raw point cloud (set of points) without voxelization or rendering. It is a unified architecture that learns both global and local point features, providing a simple, efficient and effective approach for a number of 3D recognition tasks.

Source: [Qi et al., 2017; PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation]

PointNet - architecture

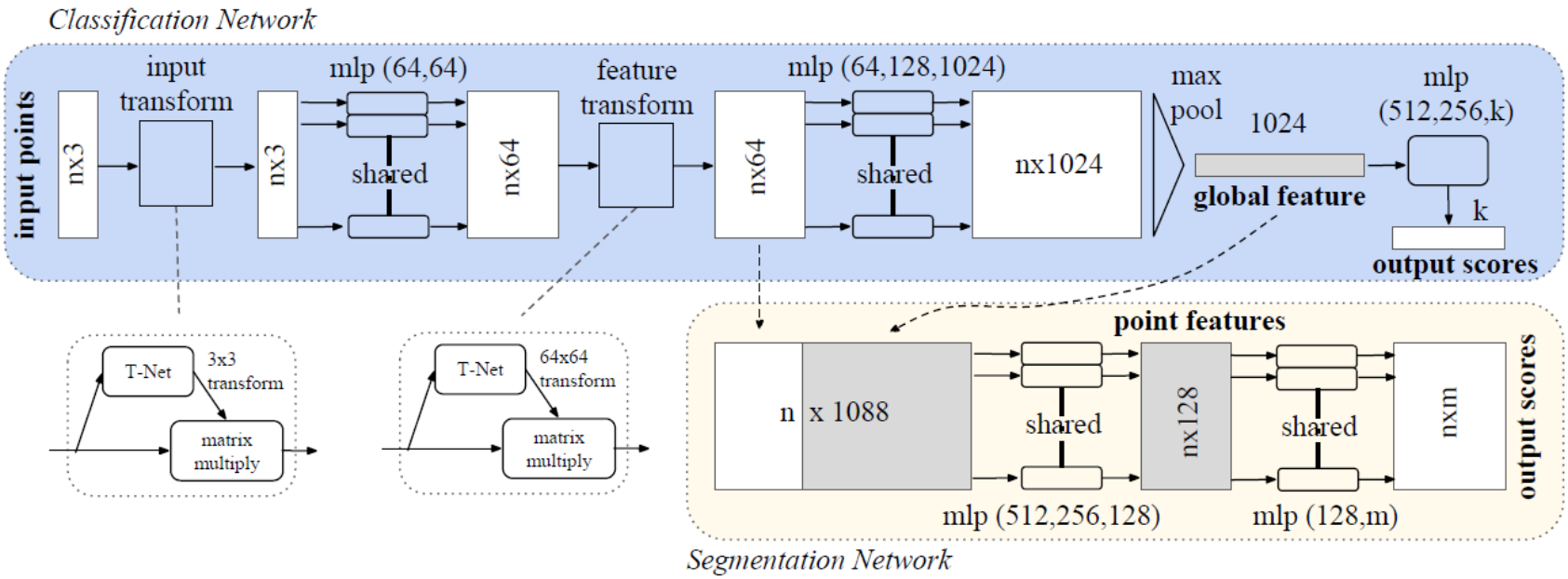


Figure 2. **PointNet Architecture.** The classification network takes n points as input, applies input and feature transformations, and then aggregates point features by max pooling. The output is classification scores for k classes. The segmentation network is an extension to the classification net.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

PointNet - keypoints (1)

- Extract relevant features of the point cloud.

PointNet - keypoints (1)

- Extract relevant features of the point cloud.
- Properties of the input point cloud reflected in the PointNet architecture

PointNet - keypoints (1)

- Extract relevant features of the point cloud.
- Properties of the input point cloud reflected in the PointNet architecture
  - *Unordered.* Possible solutions: (i) sort input, (ii) use RNN, (iii) use a symmetry function. The last approach was used. Approximate a general function defined on a point set by applying a symmetric function on transformed elements in the set

$$f(\{x_1, \ldots, x_n\}) \approx g(h(x_1), \ldots, h(x_n)), \ f : 2^{\mathbb{R}^N} \to \mathbb{R}, \ h : \mathbb{R}^N \to \mathbb{R}^K, \ g \text{ sym.} \tag{1}$$

Empirically: h is approximated by a MLP-network, g is the composition between a single variable function and max pooling.

PointNet - keypoints (1)

- Extract relevant features of the point cloud.
- Properties of the input point cloud reflected in the PointNet architecture
  - *Unordered.* Possible solutions: (i) sort input, (ii) use RNN, (iii) use a symmetry function. The last approach was used. Approximate a general function defined on a point set by applying a symmetric function on transformed elements in the set

$$f(\{x_1, \ldots, x_n\}) \approx g(h(x_1), \ldots, h(x_n)), \ f : 2^{\mathbb{R}^N} \to \mathbb{R}, \ h : \mathbb{R}^N \to \mathbb{R}^K, \ g \text{ sym.} \tag{1}$$

    Empirically: h is approximated by a MLP-network, g is the composition between a single variable function and max pooling.
  - *Invariance under isometries.* Use mini-network (T-Net) for predicting an affine transformation matrix; similarly a feature transformation matrix in the feature space ("close" to an orthogonal matrix).

PointNet - keypoints (1)

- Extract relevant features of the point cloud.
- Properties of the input point cloud reflected in the PointNet architecture
  - *Unordered.* Possible solutions: (i) sort input, (ii) use RNN, (iii) use a symmetry function. The last approach was used. Approximate a general function defined on a point set by applying a symmetric function on transformed elements in the set

$$f(\{x_1, \ldots, x_n\}) \approx g(h(x_1), \ldots, h(x_n)), \ f : 2^{\mathbb{R}^N} \to \mathbb{R}, \ h : \mathbb{R}^N \to \mathbb{R}^K, \ g \text{ sym.} \tag{1}$$

    Empirically: h is approximated by a MLP-network, g is the composition between a single variable function and max pooling.
  - *Invariance under isometries.* Use mini-network (T-Net) for predicting an affine transformation matrix; similarly a feature transformation matrix in the feature space ("close" to an orthogonal matrix).
  - *Interaction among points.* Use per-point features in the segmentation network.

PointNet - keypoints (2)

- **Universal approximation theorem.** Let $\mathcal{X} = \{S | S \subset [0,1]^m, |S| = n\}$. Suppose $f : \mathcal{X} \to \mathbb{R}$ is a continuous set function w.r.t Hausdorff distance $d_H(\cdot, \cdot)$. Then for any $\varepsilon > 0$ there exists a continuous function $h : \mathbb{R}^N \to \mathbb{R}^K$ and a symmetric function $g(x_1, \ldots, x_n) = \gamma \circ \max$, such that for any $S \in \mathcal{X}$,

$$\left| f(S) - \gamma \left(\max_{x_i \in S} \{h(x_i)\} \right) \right| < \varepsilon,$$

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

PointNet - keypoints (2)

- **Universal approximation theorem.** Let $\mathcal{X} = \{S | S \subset [0,1]^m, |S| = n\}$. Suppose $f : \mathcal{X} \to \mathbb{R}$ is a continuous set function w.r.t Hausdorff distance $d_H(\cdot, \cdot)$. Then for any $\varepsilon > 0$ there exists a continuous function $h : \mathbb{R}^N \to \mathbb{R}^K$ and a symmetric function $g(x_1, \ldots, x_n) = \gamma \circ \max$, such that for any $S \in \mathcal{X}$,

$$\left| f(S) - \gamma \left(\max_{x_i \in S} \{h(x_i)\} \right) \right| < \varepsilon,$$

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

- The theorem says that f can be arbitrarily approximated by the network given enough neurons at the max pooling layer, i.e., K in (1) is sufficiently large.

PointNet - keypoints (3)

Intuitively, the network learns to summarize a shape by a sparse set of key points. Let S be a shape. Then $f(S)$ is in fact totally determined by a finite subset $\mathcal{C}_S \subset S$ of less or equal to K elements. The set $\mathcal{C}_S$ is called the **critical point set of** S and K is called **the bottleneck dimension of** f. Moreover, $f(S)$ is unchanged up to the input corruption if all points in $\mathcal{C}_S$ are preserved; it is also unchanged with extra noise points up to an **upper bound shape** $\mathcal{N}_S$. Combined with the continuity of h, this explains the robustness of the model w.r.t point perturbation, corruption and extra noise points. The robustness is gained in analogy to the sparsity principle in machine learning models.

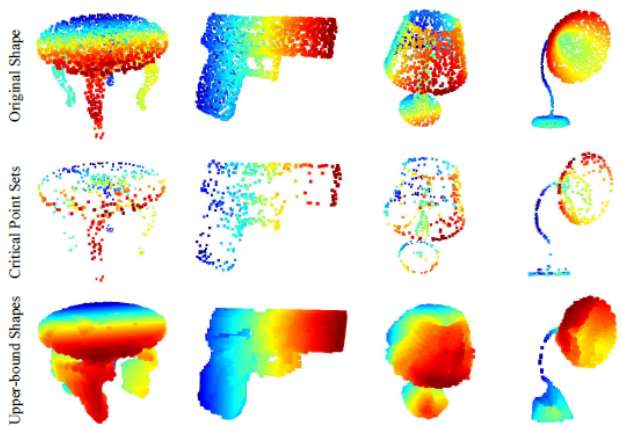


Figure 7. **Critical points and upper bound shape.** While critical points jointly determine the global shape feature for a given shape, any point cloud that falls between the critical points set and the upper bound shape gives exactly the same feature. We color-code all figures to show the depth information.

PointNet++

PointNet++ is an improvement of PointNet, aiming to better capture local structures. This is achieved by introducing a hierarchical feature learning framework.

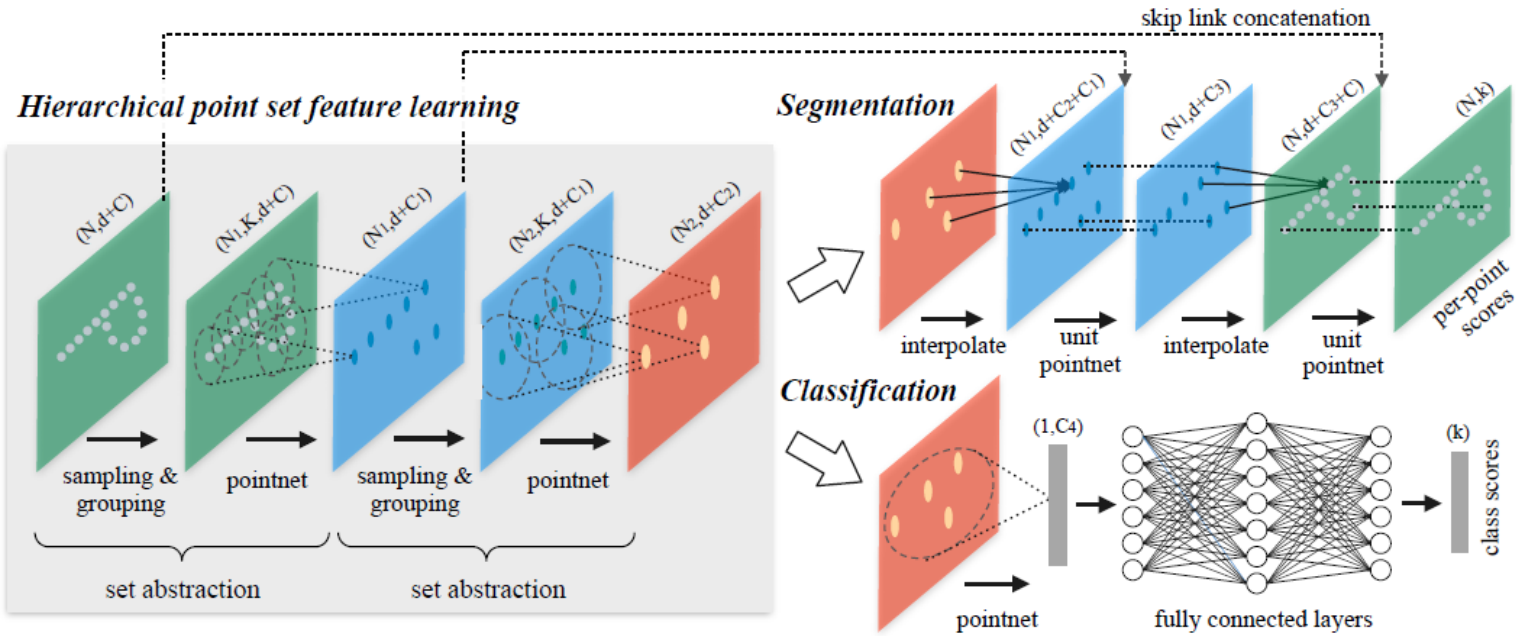
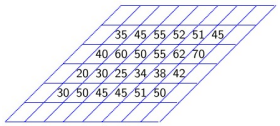


Figure 2: Illustration of our hierarchical feature learning architecture and its application for set segmentation and classification using points in 2D Euclidean space as an example. Single scale point grouping is visualized here. For details on density adaptive grouping, see Fig. 3

Source: [Qi et al., 2017, PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space]

DL in the 2D-framework





The sliding window mechanism relies on a grid regular structure.

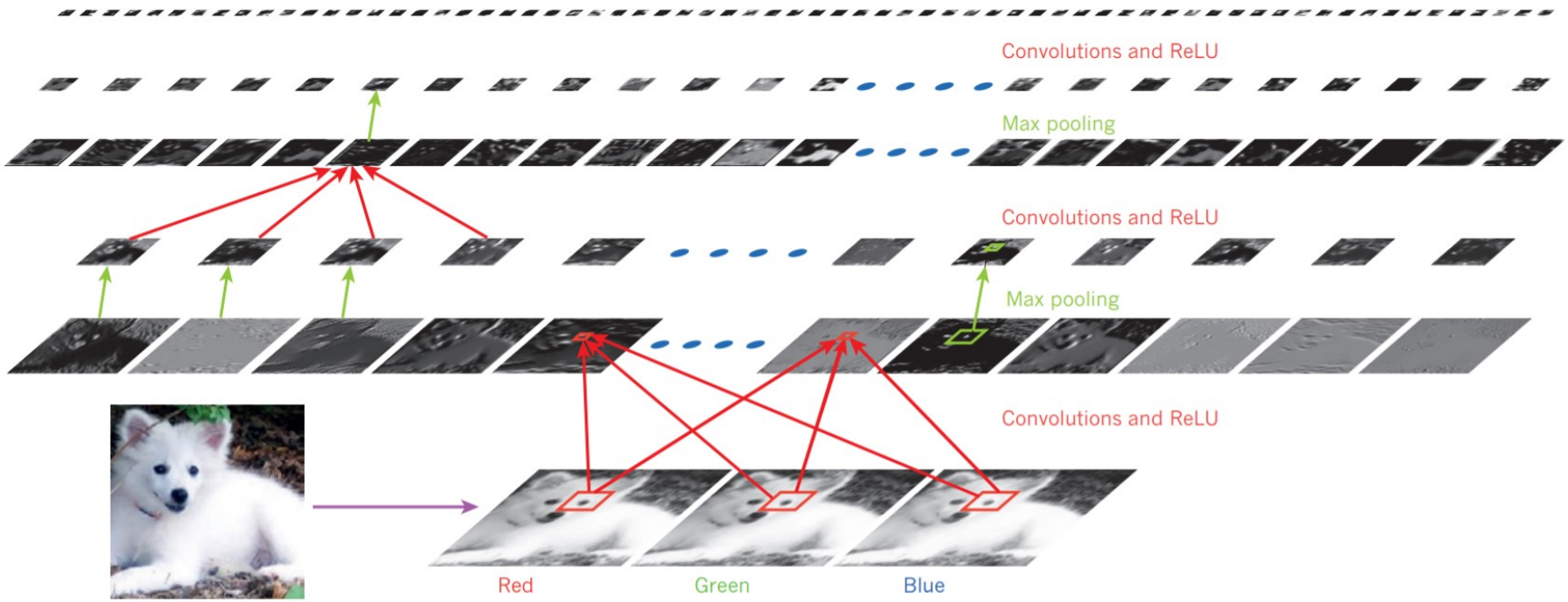


Figure 2 | Inside a convolutional network. The outputs (not the filters) of each layer (horizontally) of a typical convolutional network architecture applied to the image of a Samoyed dog (bottom left; and RGB (red, green, blue) inputs, bottom right). Each rectangular image is a feature map corresponding to the output for one of the learned features, detected at each of the image positions. Information flows bottom up, with lower-level features acting as oriented edge detectors, and a score is computed for each image class in output. ReLU, rectified linear unit.

Source: [LeCun et al., 2015]

Pointwise CNN

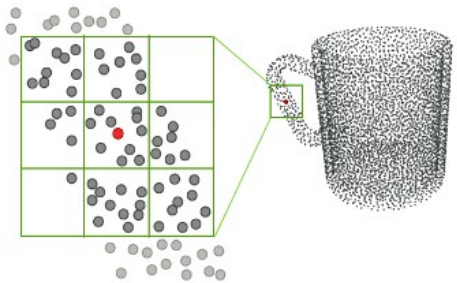


Figure 1: Pointwise convolution. We define a new convolution operator for point cloud input. For each point, nearest neighbors are queried on the fly and binned into kernel cells before convolving with kernel weights. By stacking pointwise convolution operators together, we can build fully convolutional neural networks for scene segmentation and object recognition for point clouds.

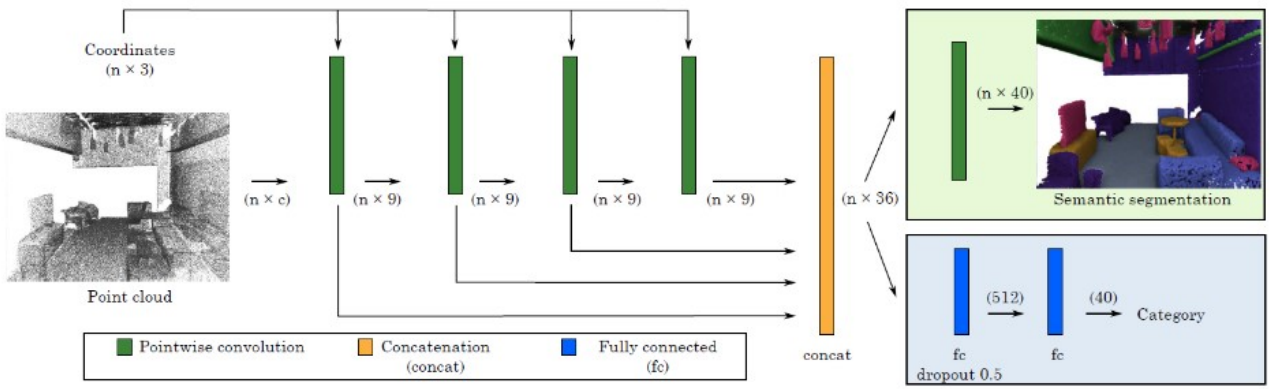


Figure 2: Pointwise convolutional neural network. The input point cloud is fed into each convolution operator, and all outputs are concatenated before being fed to a a final convolution layer for dense semantic segmentation, or to fully connected layers for object recognition. In this figure, we assume point cloud with n points and c attributes (colors, normals, coordinates, etc.). We use 9 output channels for each convolution operator before concatenation. Source code is available at our homepage [13].

Source: [Hua et al., 2018; Pointwise Convolutional Neural Networks]

Point CNN

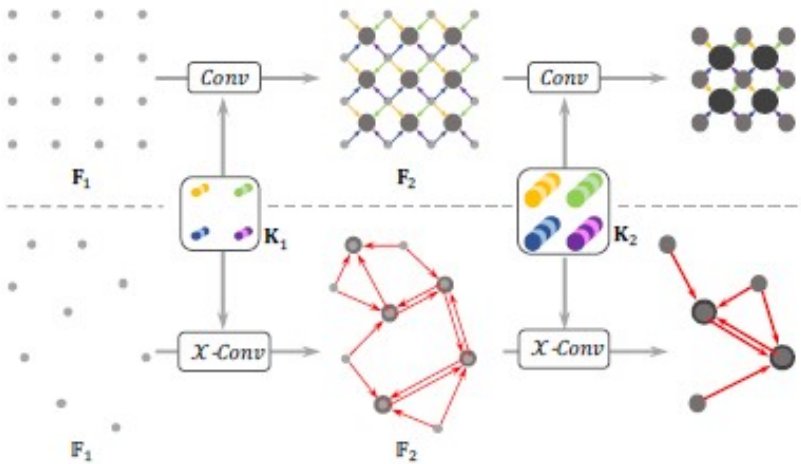


Figure 2: Hierarchical convolution on regular grids (upper) and point clouds (lower). In regular grids, convolutions are recursively applied on local grid patches, which often reduces the grid resolution ($4 \times 4 \rightarrow 3 \times 3 \rightarrow 2 \times 2$), while increasing the channel number (visualized by dot thickness). Similarly, in point clouds, $\mathcal{X}$-Conv is recursively applied to "project", or "aggregate", information from neighborhoods into fewer representative points ($9 \rightarrow 5 \rightarrow 2$), but each with richer information.

Source: [Li et al., 2018; PointCNN: Convolution On $\mathcal{X}$ -Transformed Points]

KPConv

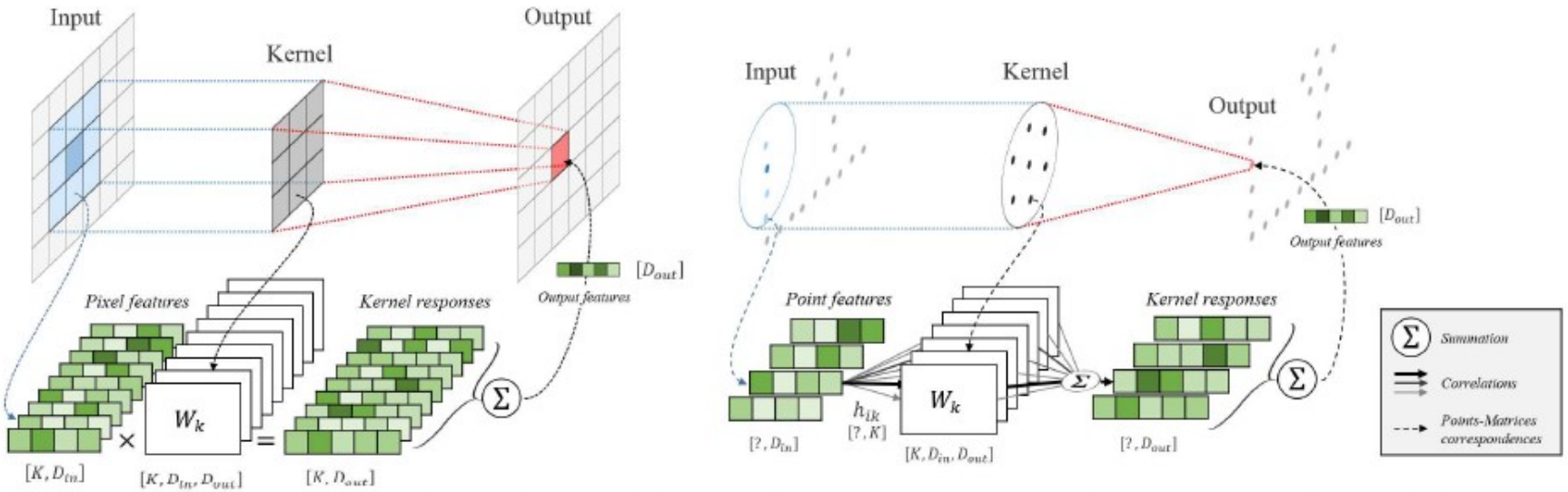


Figure 2. Comparison between an image convolution (left) and a KPConv (right) on 2D points for a simpler illustration. In the image, each pixel feature vector is multiplied by a weight matrix $(W_k)_{k<K}$ assigned by the alignment of the kernel with the image. In KPConv, input points are not aligned with kernel points, and their number can vary. Therefore, each point feature f_i is multiplied by all the kernel weight matrices, with a correlation coefficient h_{ik} depending on its relative position to kernel points.

Source: [Thomas et al., 2019; KPConv: Flexible and Deformable Convolution for Point Clouds]

Delta conv - preparations (I)

- The discrete Laplace operator is defined by the matrix

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$





Source: https://docs.opencv.org/3.4/d5/db5/tutorial_laplace_operator.html

Delta conv - preparations (I)

- The discrete Laplace operator is defined by the matrix

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$





Source: https://docs.opencv.org/3.4/d5/db5/tutorial_laplace_operator.html

- This is the "discrete" version of the Laplace operator, defined in the "continuous" Euclidean set-up by

$$\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}.$$

Delta conv - preparations (II)

The Laplace operator is an useful tool for handling a wide range of data / data types. Here an example: triangle mesh smoothing in computer graphics.

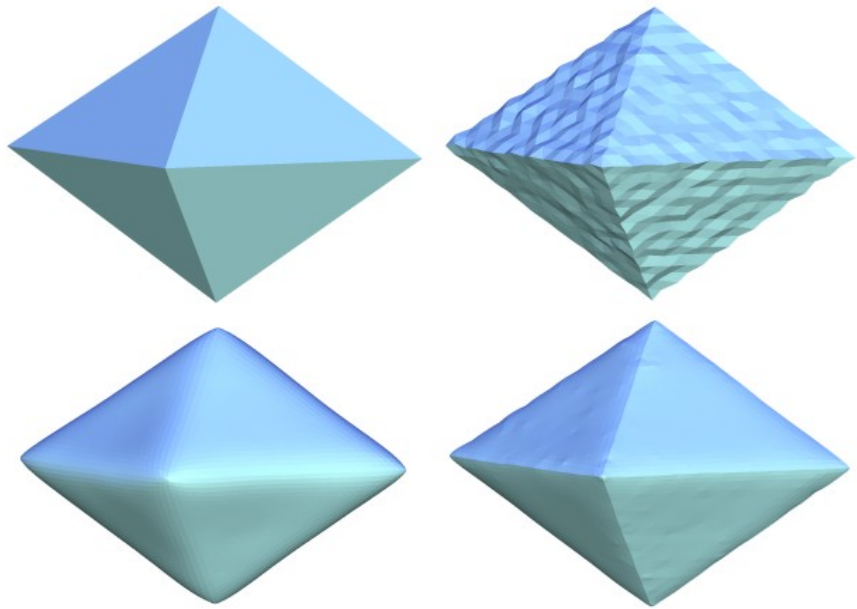


Figure 8: Smoothing a pyramid. Top row: original DOUBLEPYRAMID and the noisy version. Bottom row: Using $\mathbf{W}_p = \mathbf{W}_{linear}$ alone smoothes out sharp features, while additionally reducing the weights on Laplacian constraints of feature vertices recovers most of the original shape.

Source: Nealen et al., 2006; Laplacian Mesh Optimization

Delta conv - preparations (III)

- Let (M, g) be a compact manifold (e.g. the sphere S^n). On M one can consider *scalar fields* and *vector fields*. The metric g yields a volume form, allowing the definition of the integral of scalar fields over M: $\int_M f(x)dx$.
- One defines the Hilbert spaces of scalar and vector fields, denoted by $L^2(M), L^2(TM)$ with the inner products

$$\langle f, g\rangle_{L^2(M)} = \int_M f(x)g(x)dx, \quad \langle V, W\rangle_{L^2(TM)} = \int_M \langle V(x), W(x)\rangle_{T_xM} dx$$

- One defines the **intrinsic gradient** $\nabla : L^2(M) \to L^2(TM)$; it verifies the relation

$$df(x) = \langle \nabla f(x), \cdot \rangle_{T_xM},$$

  where $df : TM \to \mathbb{R}$ is the differential of f and it acts on tangent vector fields
- The **divergence** $\mathrm{div} : L^2(TM) \to L^2(M)$ is the formal adjoint of the gradient
- The **Laplace-Beltrami operator** $\Delta : L^2(M) \to L^2(M)$ is defined as

$$\Delta : L^2(M) \to L^2(M), \qquad \Delta f = -\mathrm{div}(\nabla f)$$

- The Laplace operator is self-adjoint (symmetric) and positive definite.

Delta conv

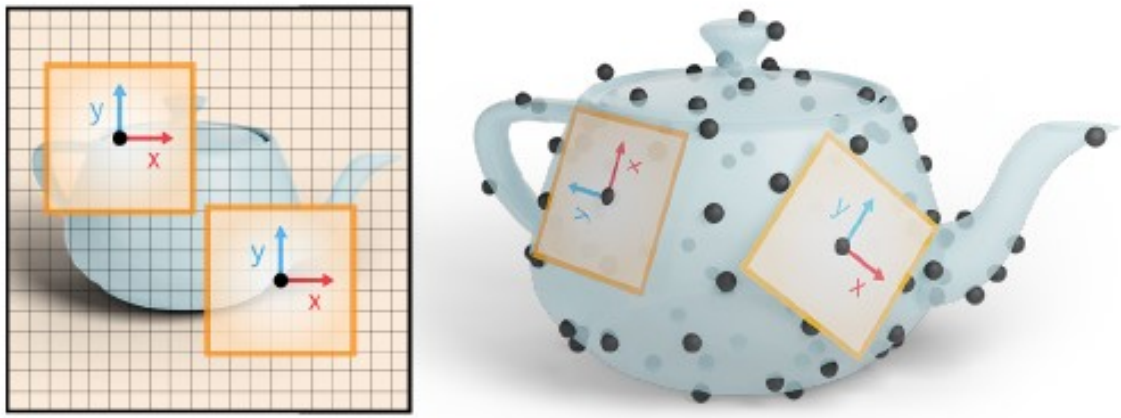


Fig. 1. Images have a global coordinate system (left). Point clouds do not (right), complicating the design of anisotropic convolutions.

Source: [Wiersma et al., 2021; DeltaConv: Anisotropic Operators for Geometric Deep Learning on Point Clouds]

EdgeConv

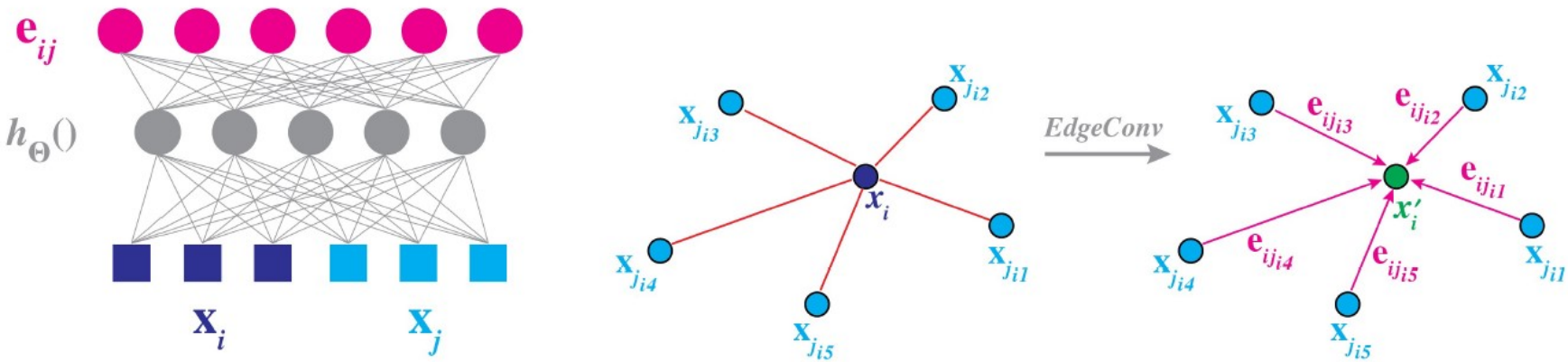


Fig. 2. **Left**: Computing an edge feature, $\mathbf{e}_{ij}$ (top), from a point pair, $\mathbf{x}_i$ and $\mathbf{x}_j$ (bottom). In this example, $h_\Theta()$ is instantiated using a fully connected layer, and the learnable parameters are its associated weights. **Right**: The EdgeConv operation. The output of EdgeConv is calculated by aggregating the edge features associated with all the edges emanating from each connected vertex.

Source: [Wang et al., 2021; Dynamic Graph CNN for Learning on Point Clouds]

Grid-GCN

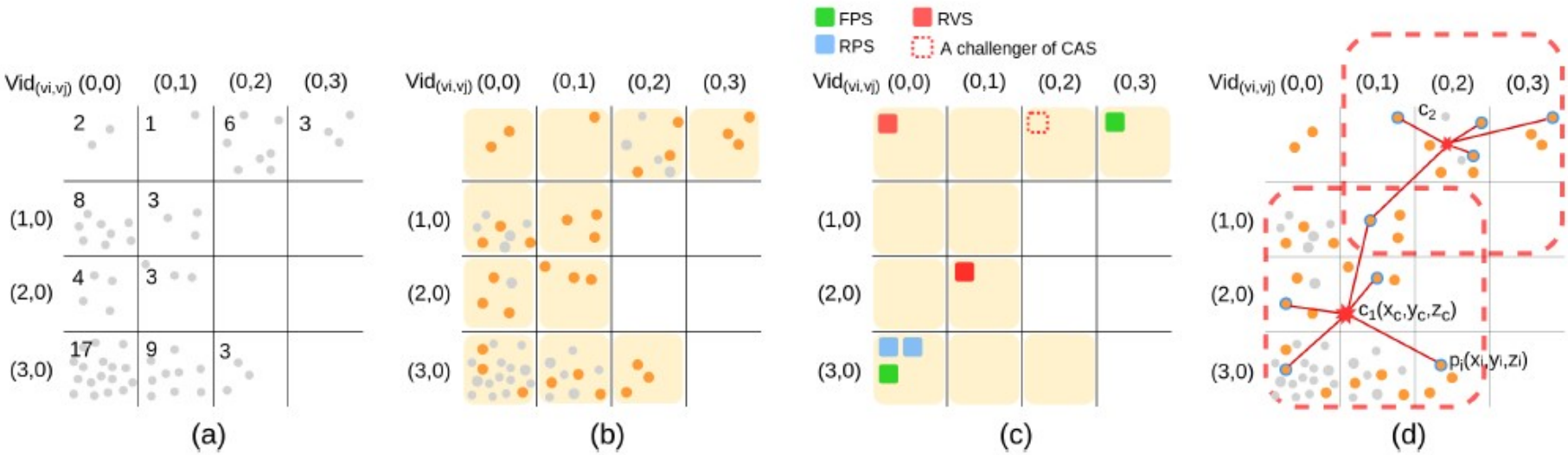


Figure 2: Illustration of Coverage-Aware Grid Query (CAGQ). Assume we want to sample $M = 2$ point groups and query $K = 5$ node points for each group. **(a)** The input is N points (grey). The voxel id and number of points is listed for each occupied voxel. **(b)** We build voxel-point index and store up to $n_v = 3$ points (yellow) in each voxel. **(c)** Comparison of different sampling methods: FPS and RPS prefer the two centers inside the marked voxels. Our RVS could randomly pick any two occupied voxels (e.g. (2,0) and (0,0)) as center voxels. If our CAS is used, voxel (0,2) will replace (0,0). **(d)** Context points of center voxel (2,1) are the yellow points in its neighborhood (we use 3×3 as an example). CAGQ queries 5 points (yellow points with blue ring) from these context points, then calculate the locations of the group centers.

Source: [Xu et al., 2020; Grid-GCN for Fast and Scalable Point Cloud Learning]

KD-Net

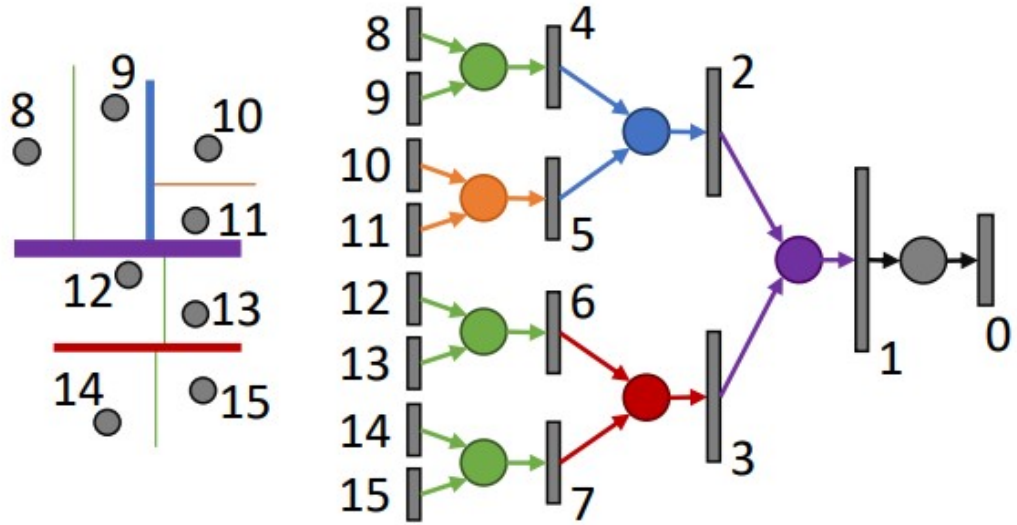


Figure 1. A kd-tree built on the point cloud of eight points (left), and the associated Kd-network built for classification (right). We number nodes in the kd-tree from the root to leaves. The arrows indicate information flow during forward pass (inference). The leftmost bars correspond to leaf (point) representations. The rightmost bar corresponds to inferred class posteriors $\mathbf{v}_0$. Circles correspond to affine transformations with learnable parameters. Colors of the circles indicate parameter sharing, as splits of the same type (same orientation, same tree level – three "green" splits in this example) share the transformation parameters.

Source: [Klokov & Lempitsky, 2017; Escape from Cells: Deep Kd-Networks for the Recognition of 3D Point Cloud Models]

Octnet

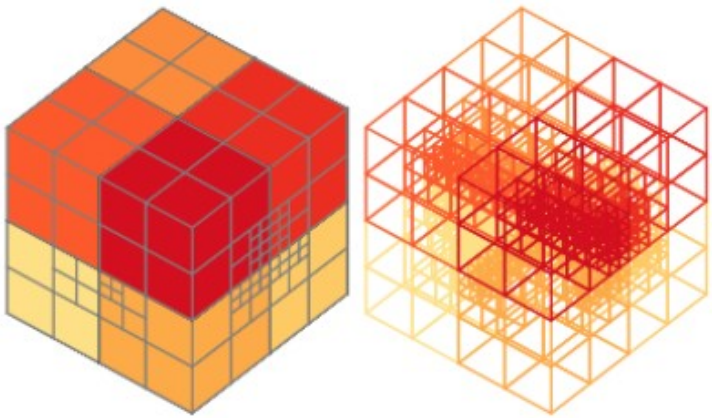


Figure 2: **Hybrid Grid-Octree Data Structure.** This example illustrates a hybrid grid-octree consisting of 8 shallow octrees indicated by different colors. Using 2 shallow octrees in each dimension with a maximum depth of 3 leads to a total resolution of 16^3 voxels.

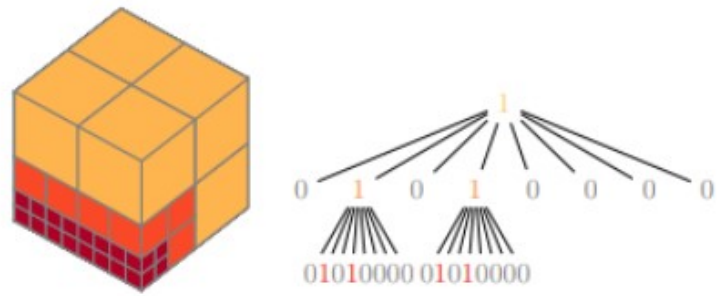


Figure 3: **Bit Representation.** Shallow octrees can be efficiently encoded using bit-strings. Here, the bit-string 1 01010000 00000000 01010000 00000000 01010000 0... defines the octree in (a). The corresponding tree is shown in (b). The color of the voxels corresponds to the split level.

Source: [Riegler et al., 2017; OctNet: Learning Deep 3D Representations at High Resolutions]

Octree guided CNN

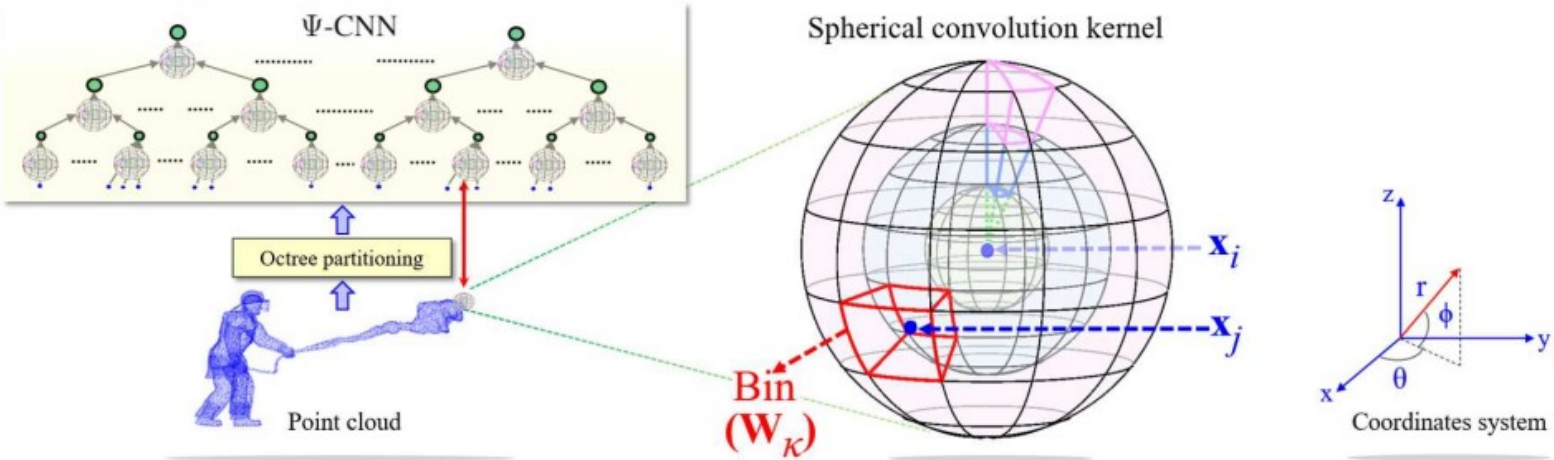


Figure 1. The proposed octree guided CNN, i.e. Ψ-CNN directly processes raw point clouds using octree partitioning information. The representation is hierarchically coarsened at each network layer (three layers depicted) by applying spherical convolutional kernels. A spherical kernel systematically splits the space around a point $\mathbf{x}_i$ into multiple volumetric bins. For the jth neighboring point $\mathbf{x}_j$, a kernel first determines its relevant bin and uses the weight matrix $\mathbf{W}_\kappa$ defined for that bin to compute the activation value. The proposed spherical kernel preserves translation-invariance and asymmetry properties of standard 2D convolutional kernel in 3D point cloud domain.

Source: [Lei et al., 2019; Octree Guided CNN With Spherical Kernels for 3D Point Clouds]

Point transformer

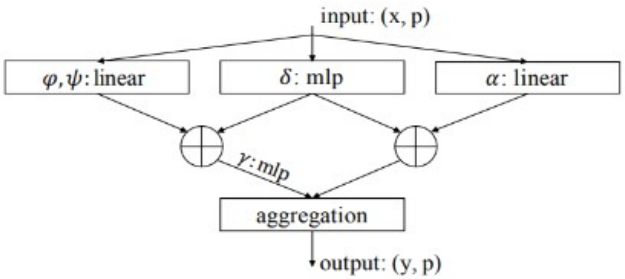


Figure 2. Point transformer layer.

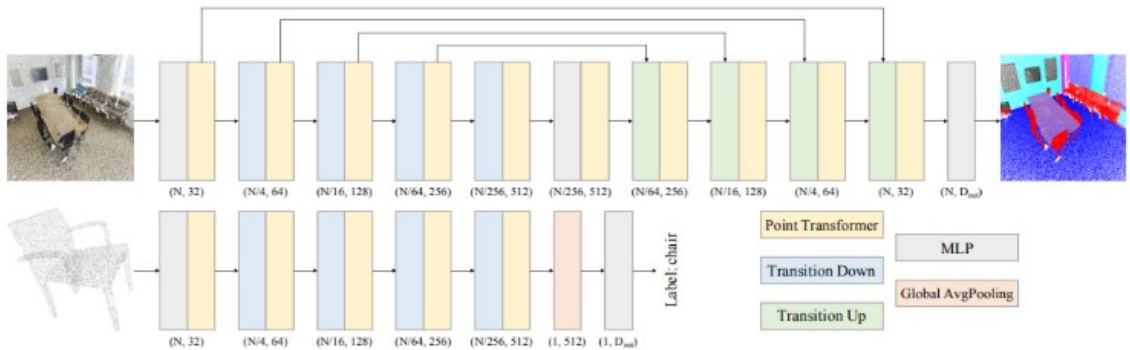


Figure 3. Point transformer networks for semantic segmentation (top) and classification (bottom).

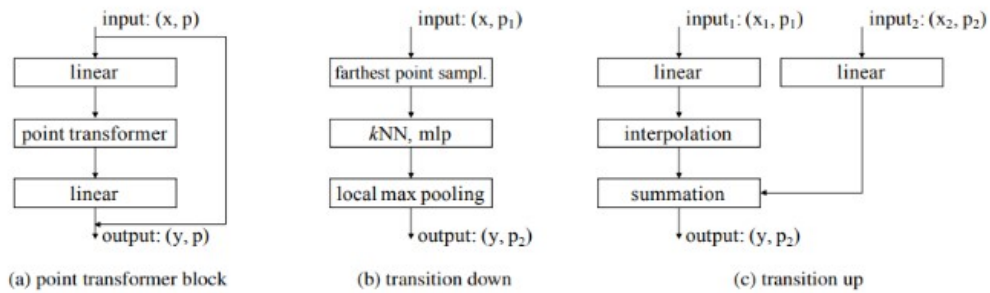


Figure 4. Detailed structure design for each module.

Source: [Zhao et al., 2021; Point transformer]

Point transformer V3

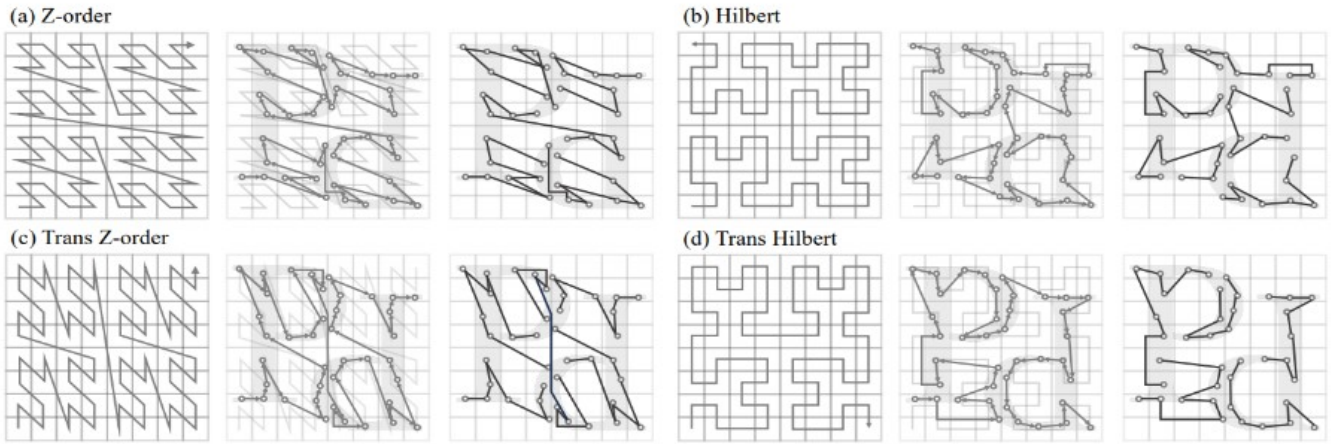


Figure 3. **Point cloud serialization.** We show the four patterns of serialization with a triplet visualization. For each triplet, we show the space-filling curve for serialization (left), point cloud serialization var sorting order within the space-filling curve (middle), and grouped patches of the serialized point cloud for local attention (right). Shifting across the four serialization patterns allows the attention mechanism to capture various spatial relationships and contexts, leading to an improvement in model accuracy and generalization capacity.

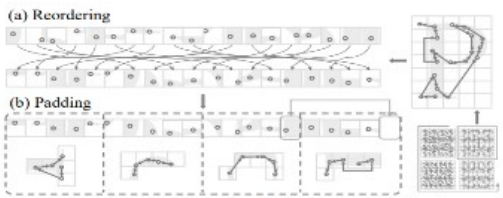


Figure 4. **Patch grouping.** (a) Reordering point cloud according to order derived from a specific serialization pattern. (b) Padding point cloud sequence by borrowing points from neighboring patches to ensure it is divisible by the designated patch size.

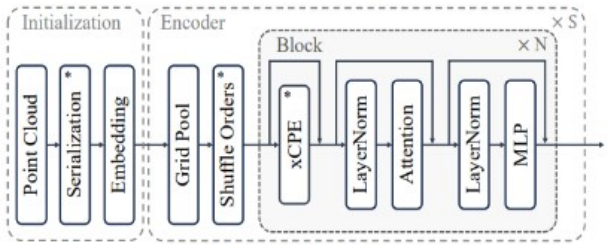


Figure 6. **Overall architecture.**

Source: [Wu et al., 2024; Point Transformer V3: Simpler, Faster, Stronger]

Stratified transformer

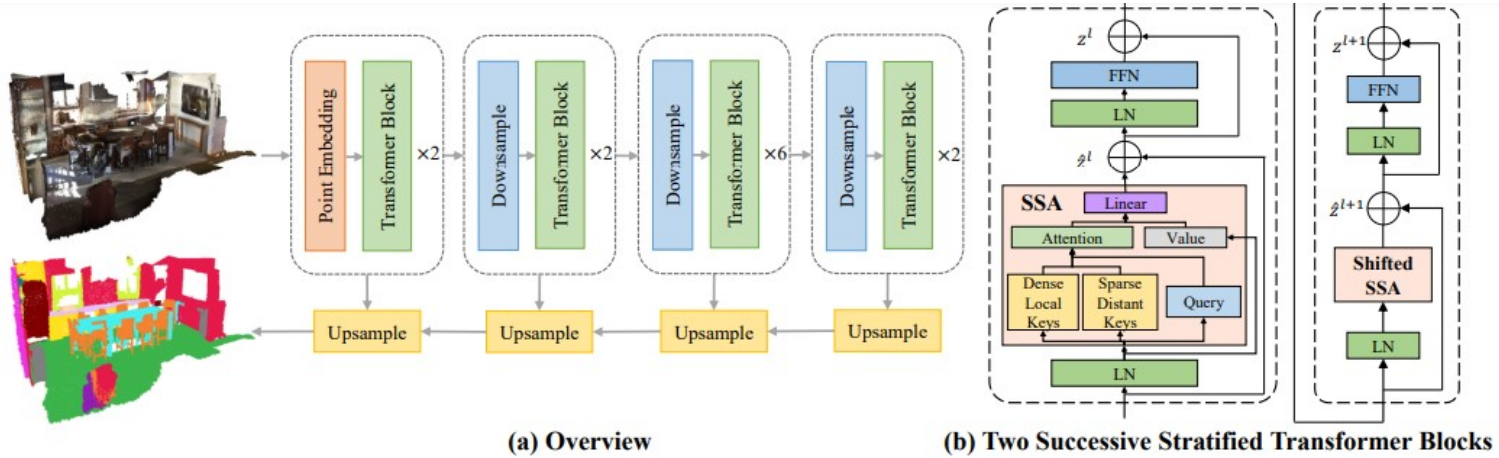


Figure 2. (a) Framework Overview. (b) Structure of Stratified Transformer Block. Hierarchical structure is employed to obtain multi-level features. Input point clouds firstly go through the Point Embedding module to aggregate local structure information. After several downsample layers and transformer blocks, the features are upsampled for segmentation. SSA: Stratified Self-attention. Shifted SSA: SSA with shifted window. Best viewed in color.

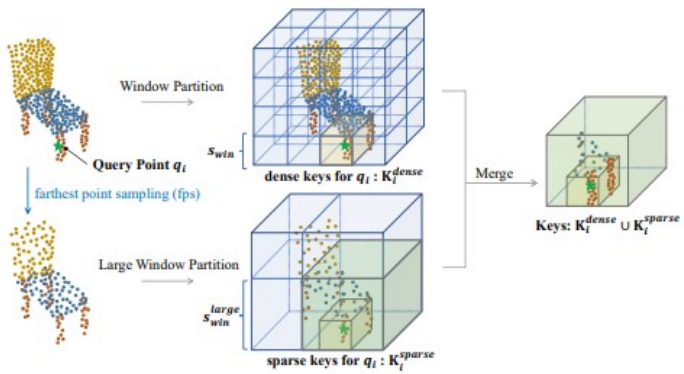


Figure 3. Illustration of the stratified strategy for keys sampling. The green star denotes the given query point.

Source: [Lai et al., 2022; Stratified Transformer for 3D Point Cloud Segmentation]

Stratified transformer + PAGWN

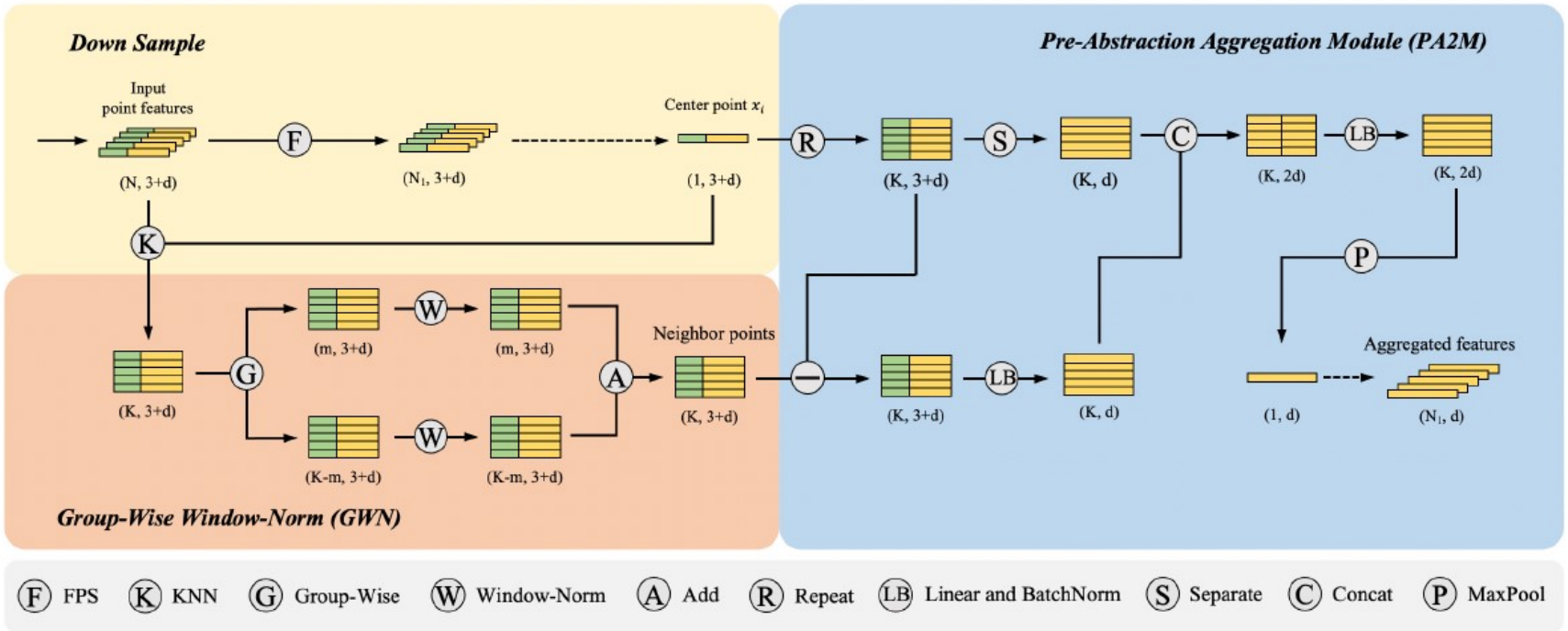


Figure 9. Illustration of our pre-abstraction group-wise window-normalization module. GWN module normalizes the volume of the local region and provides a variety of information. PAGWN module pre-aggregates the features of location encoding and neighborhood points to balance local and global features.

Source: [Wang et al., 2022; Window Normalization: Enhancing Point Cloud Understanding by Unifying Inconsistent Point Densities]

PS2-Net

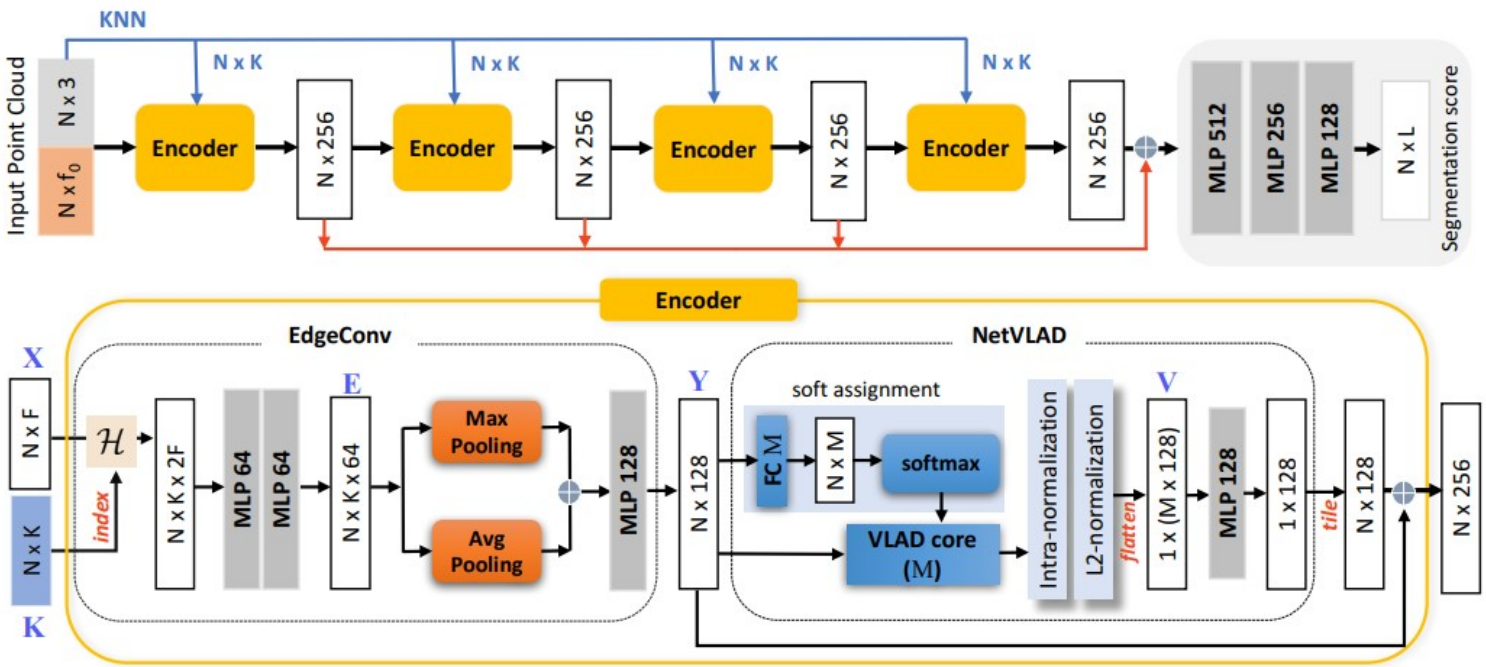


Figure 2. The network architecture of our PS^2-Net.

Source: [Zhao et al., 2019; PS^2-Net: A Locally and Globally Aware Network for Point-Based Semantic Segmentation]

PointMotionNet

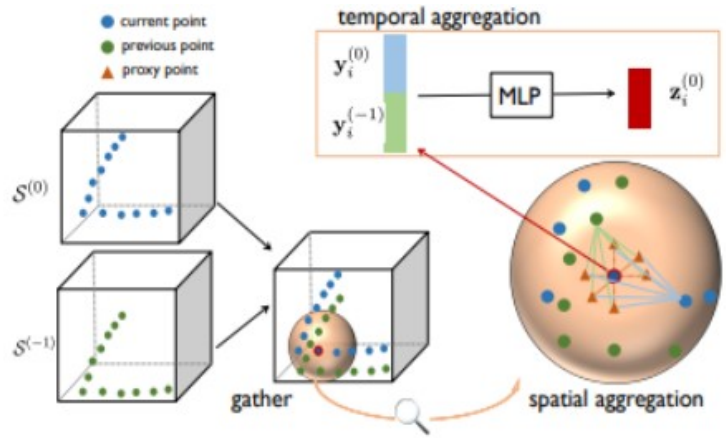


Figure 2. **4D point spatiotemporal convolution operation (Point-STC).** To extract spatiotemporal features, we gather all points across time to the same spatial coordinate system. For each query point (with red circle), we find its neighbors (in the orange ball) in each frame. With the set of proxy points (in red triangle) of the query point, we extract spatial features in each individual frame, which follows from (1a). We next concatenate the spatial features across all time stamps and use a MLP to obtain the final spatiotemporal features, which follows from (1b).

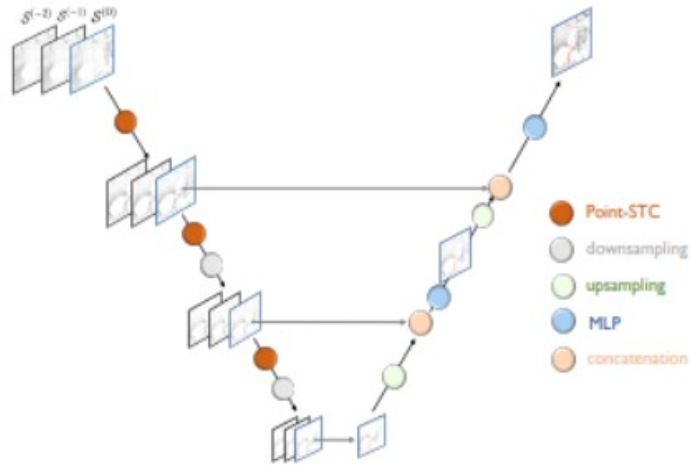


Figure 3. **PointMotionNet is a point-based spatiotemporal pyramid network.** Point-STC layers enable spatiotemporal feature learning. Downsampling and upsampling layers enable multiscale feature learning. With the skip connections, the aggregation layers fuse features from both encoder and decoder.

Source: [Wang et al., 2022; PointMotionNet: Point-Wise Motion Learning for Large-Scale LiDAR Point Clouds Sequences]

FG-Net

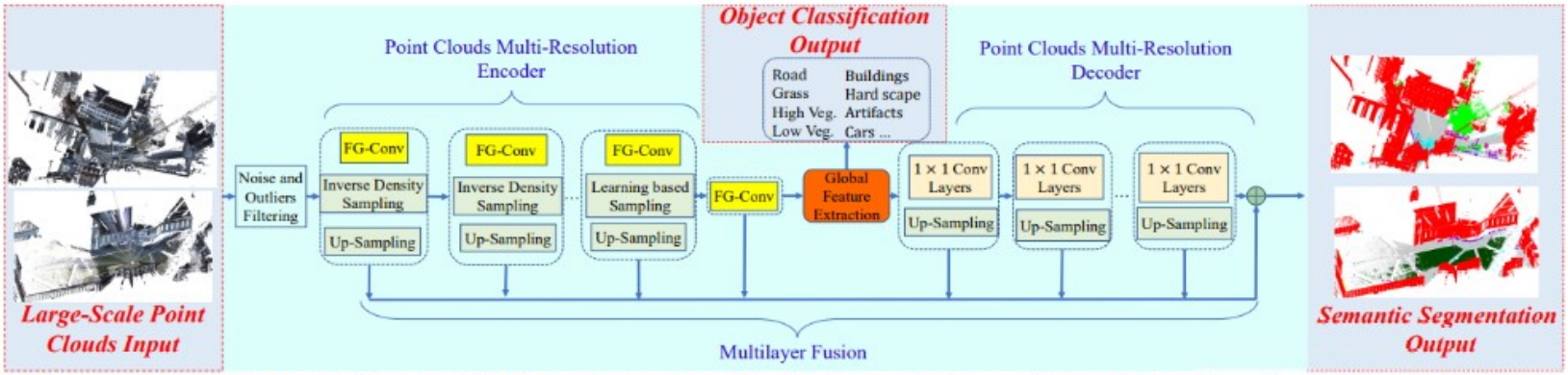


Fig. 2. Overall system framework of proposed ***FG-Net***, and the core module ***FG-Conv*** can be integrated into ***FG-Net*** with multi-resolution residual learning. 1 × 1 Conv stands for 1 × 1 convolutions. The network operations are done from top to bottom and from left to right.

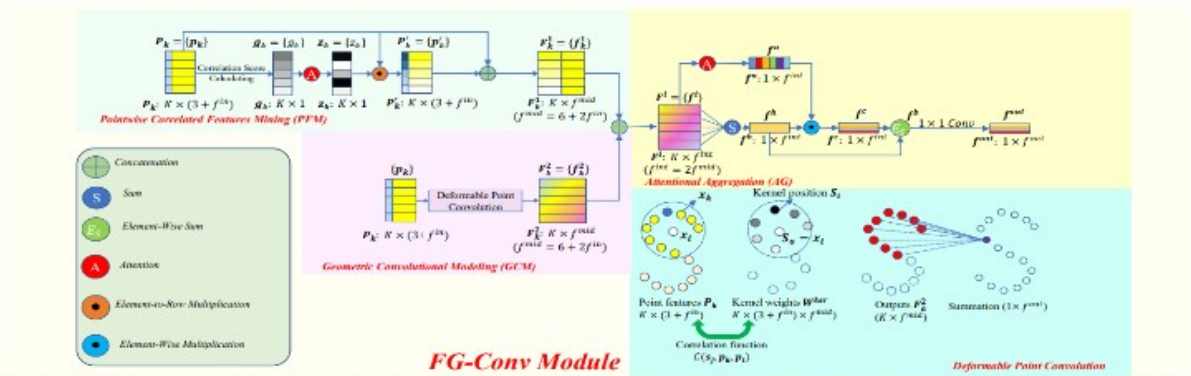


Fig. 3. Detailed illustration of our proposed novel pointwise correlated feature and geometric convolutional modelling module (***FG-Conv Module***), the deformable convolution operation is illustrated at the right bottom corner, with the query point denoted as p_i and the *k-th* neighbour point denoted as p_k, the output vector of point convolution is the dot product of point features and kernel weights. The feature vector is summed in attentional aggregation.

Source: [Liu et al., 2021; FG-Net: Fast Large-Scale LiDAR Point Clouds Understanding Network Leveraging Correlated Feature Mining and Geometric-Aware Modelling]

LGS-Net

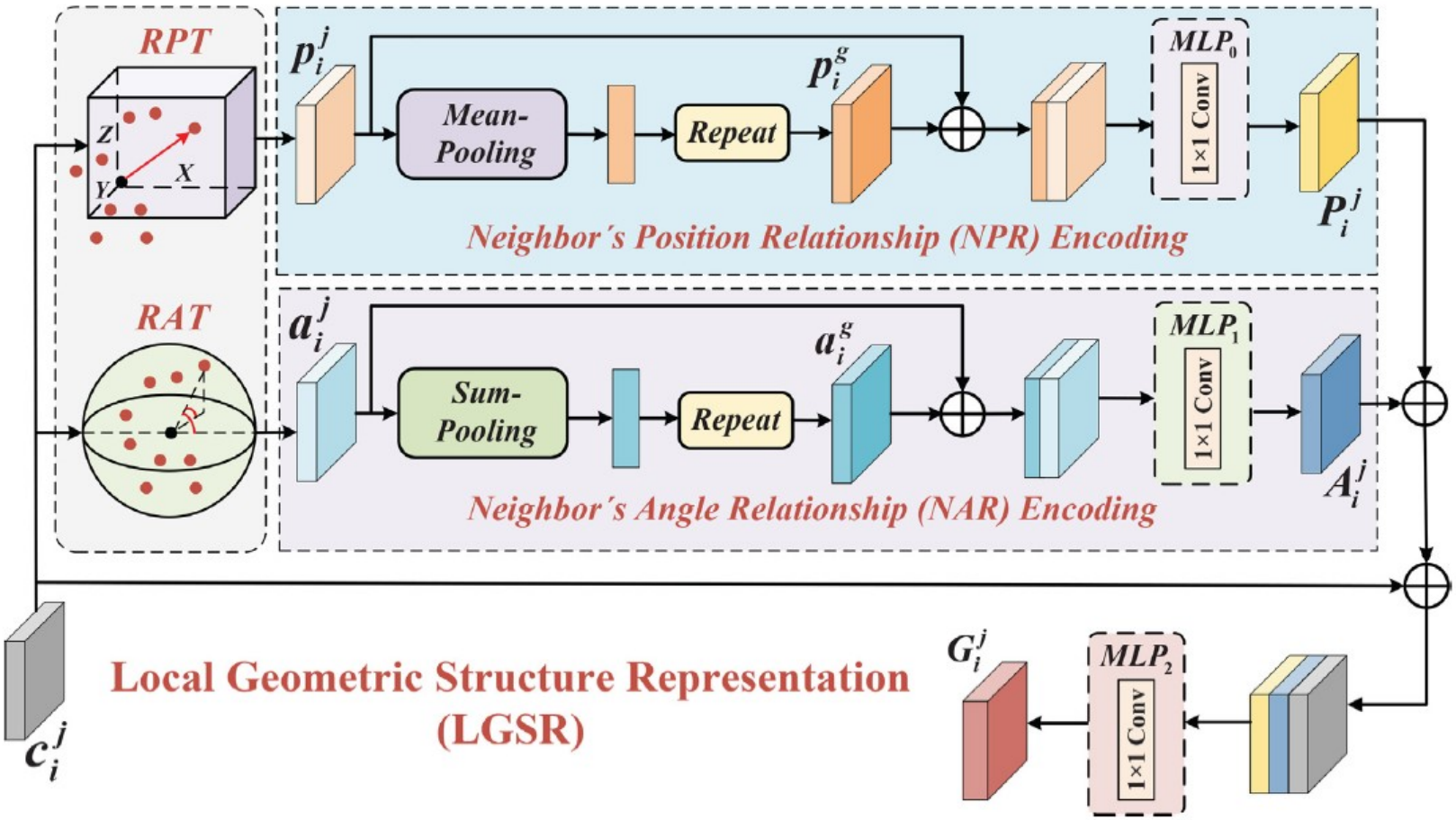


Fig. 4. The proposed Local Geometric Structure Representation (LGSR) block. Firstly, The Relative Position Transform(RPT) and Relative Angle Transform(RAT) transform raw input coordinate features c_i^j into relative position features p_i^j and relative angle features a_i^j. Then, the NPR and NAR Module learn neighbor's position features P_i^j and neighbor's angle features A_i^j based on relative and global relationships by using shared MLP and neighbor's mean and sum function respectively. The reason for using mean and sum function on position and angle features are analyzed in Set.3.1.1 and Set.4.4. Finally, the local geometric representation G_i^j is learned by concatenating local position features, local angel features, and raw input coordinate features.

Source: [Shao et al., 2022; Mining local geometric structure for large-scale 3D point clouds semantic segmentation]

P2P

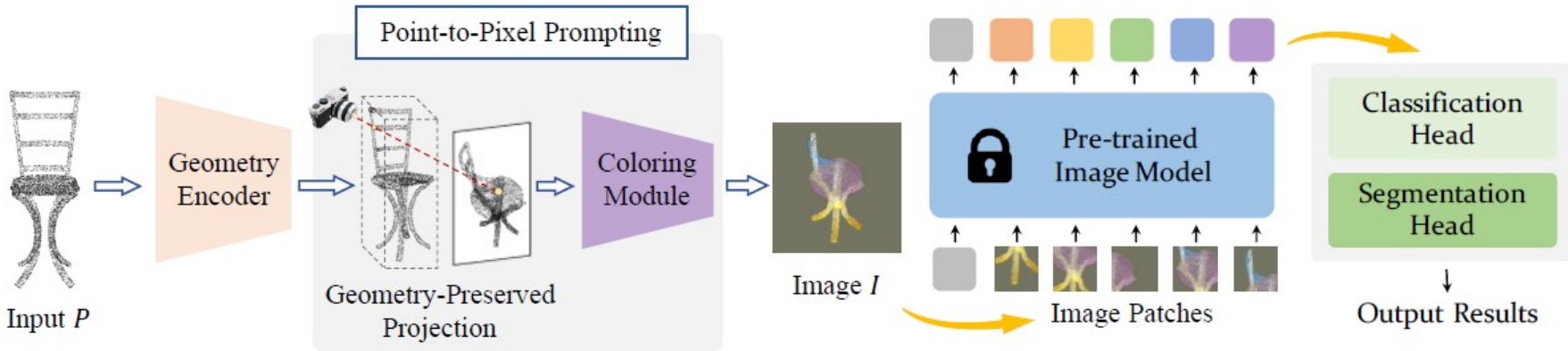


Figure 2: **The pipeline of our proposed P2P framework.** Taking a point cloud P as the input, we first encode the geometry information for each point. Then we sample a projection view and rearrange the point-wise features into an image-style layout to obtain the pixel-wise features with *Geometry-preserved Projection*. The colorless projection will be enriched to produce a colorful image I with the color information via a learnable *Coloring Module*. Our P2P framework can be easily transferred to several downstream tasks with a task-specific head with the help of the transferable visual knowledge from the pre-trained image model. We take the classical Vision Transformer [15] as our pre-trained image model for illustration in this pipeline.

Source: [Wang et al., 2022; P2P: Tuning Pre-trained Image Models for Point Cloud Analysis with Point-to-Pixel Prompting]

Point2vec

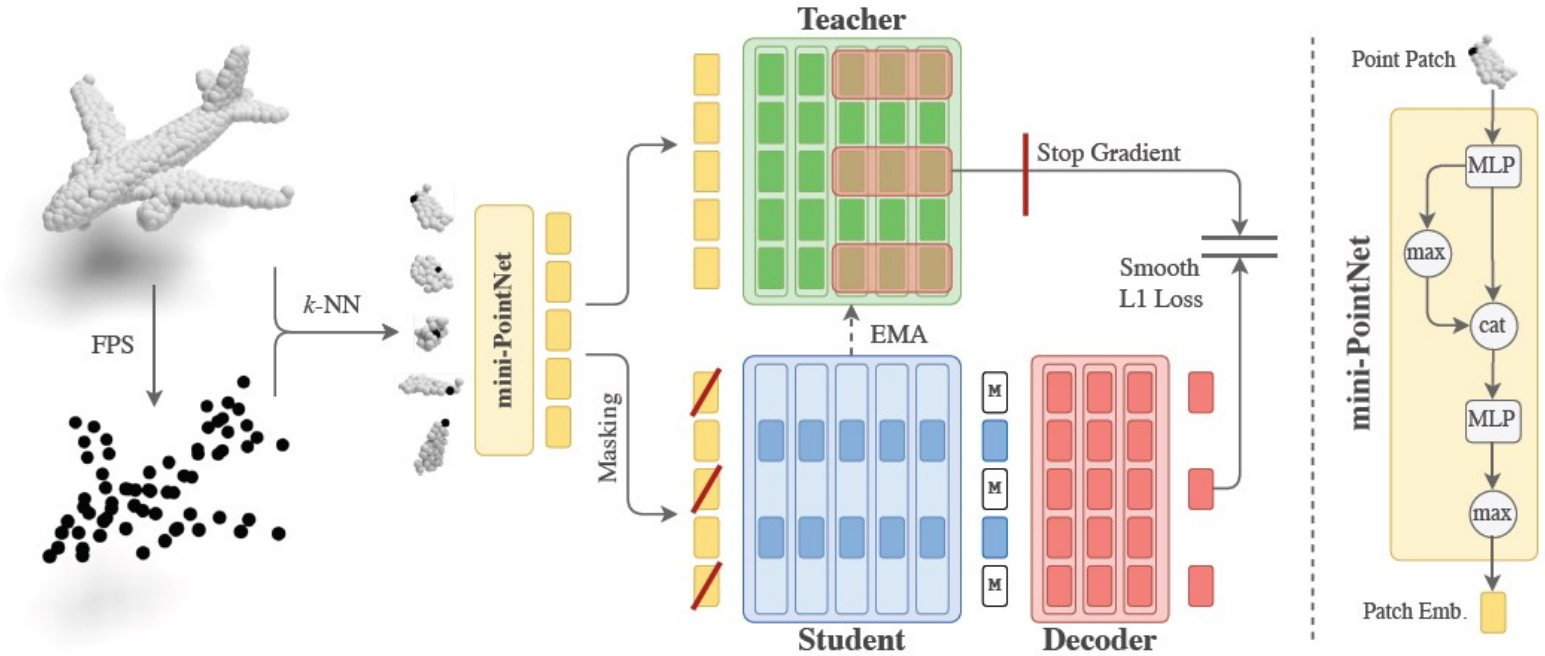


Fig. 2: Point2Vec pre-training. Our model divides the input point cloud into point patches using farthest point sampling (FPS) and k-NN aggregation. We obtain patch embeddings by applying a mini-PointNet to each point patch (*right*). The teacher Transformer encoder infers a contextualized representation for all patch embeddings which, after normalization and averaging over the last K Transformer layers, serve as training targets. The student's input is a masked view on the input data, *i.e.* we randomly mask out a ratio of patch embeddings and only pass the remaining embeddings into the student Transformer encoder. After applying a shallow decoder on the outputs of the student, padded with learned mask embeddings M, we train the student and decoder to predict the latent teacher representation of the patch embeddings.

PointGPT

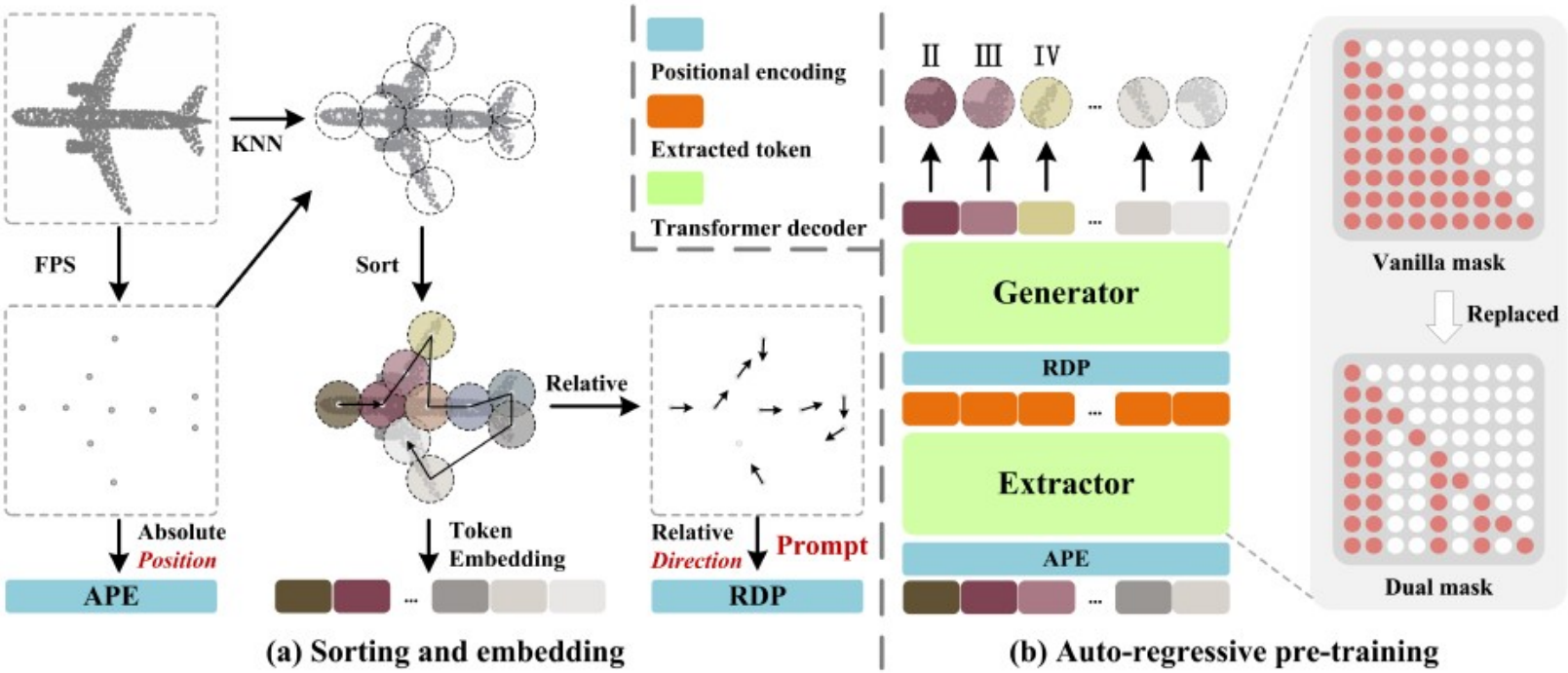


Figure 2: Overall architecture of our PointGPT. (a) The input point cloud is divided into multiple point patches, which are then sorted and arranged in an ordered sequence. (b) An extractor-generator based transformer decoder is employed along with a dual masking strategy for the auto-regressively prediction of the point patches. In this example, the additional mask of the dual masking strategy is applied to the same group of random tokens for better illustration purposes.

Source: [Chen et al., 2024; PointGPT: Auto-regressively Generative Pre-training from Point Clouds]