Nuages de Points et Modélisation 3D

6 - Machine learning III
From convolution to transformers

Overview

Machine learning courses

- Surface reconstruction
- Descriptors and machine learning
- Image based processing
- Geometric deep learning
- Convolutional and Transformer based architectures

Today

- Tasks and corresponding architectures

ML course 4

Evaluation

QCM on the course

- No document
- Mainly course questions

I - Convolutions on points

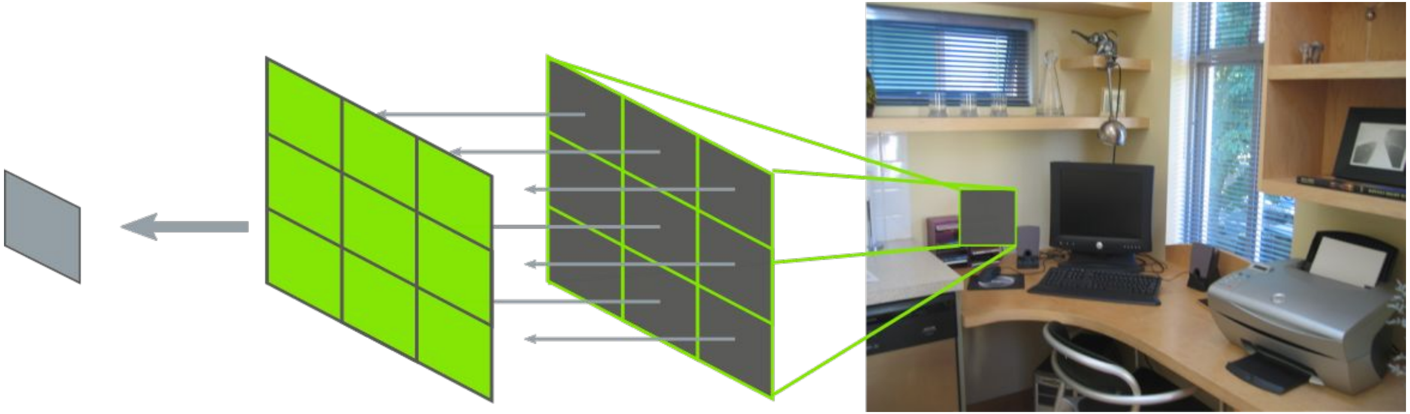
I - Convolutions on points

A - Convolution formulation

Convolution on images

Convolution for image processing

$$\mathbf{h}[n] = \sum_{f \in \{1,\ldots,C\}} \; \sum_{m \in \{-M/2,\ldots,M/2\}^d} \mathbf{K}_f[m]\, \mathbf{f}_f[n+m]$$

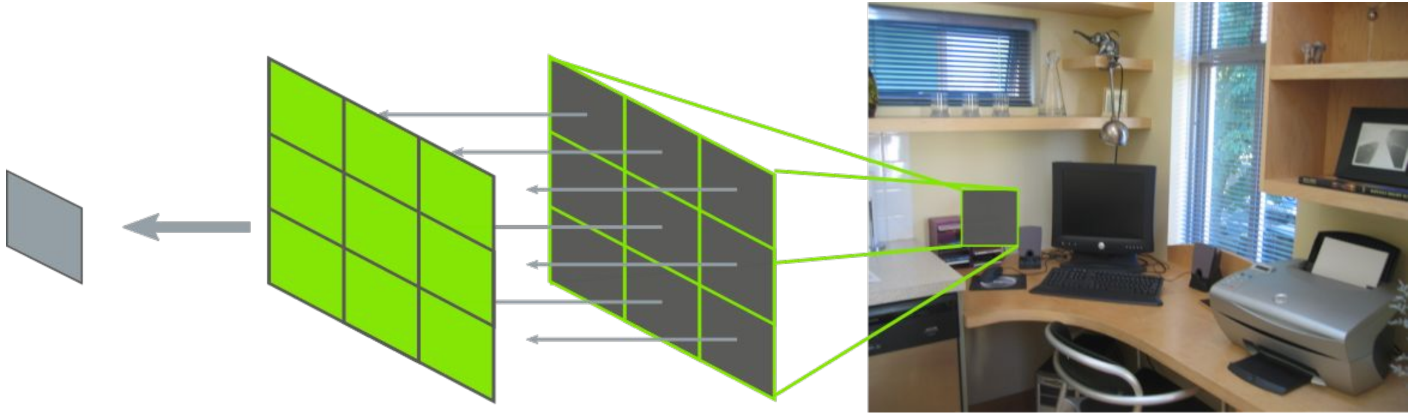


Convolution on images

Convolution on images

Convolution for image processing

$$\mathbf{h}[n] = \sum_{f \in \{1,\ldots,C\}} \mathbf{K}_f^\top \, \mathbf{f}_f(n)$$

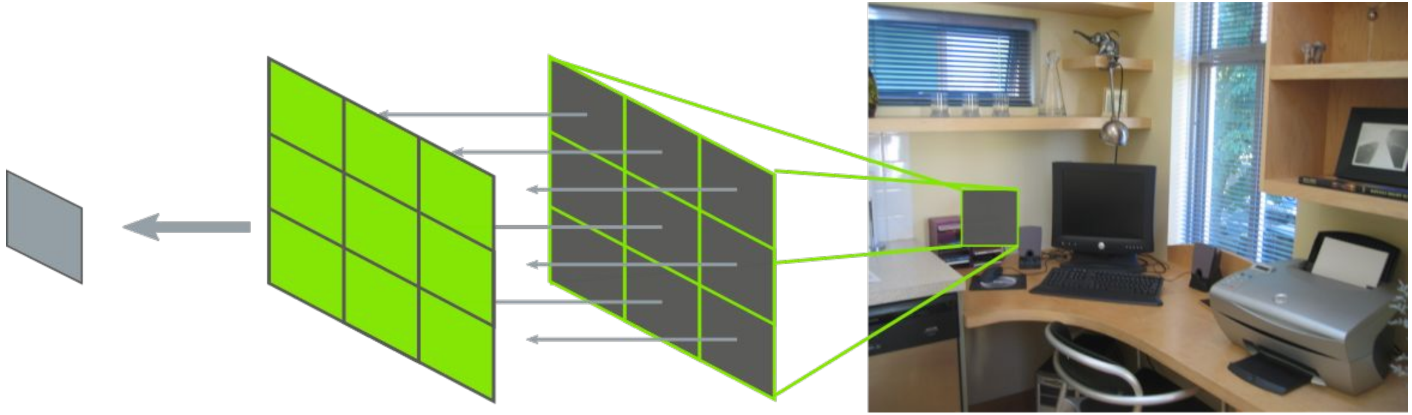


Convolution on images

Convolution on images

Convolution for image processing

$$\mathbf{h}[n] = \sum_{f\in\{1,\dots,C\}} \underbrace{\mathbf{K}_f^\top}_{\text{Kernel space}} \underbrace{\mathbf{f}_f(n)}_{\text{Feature space}}$$



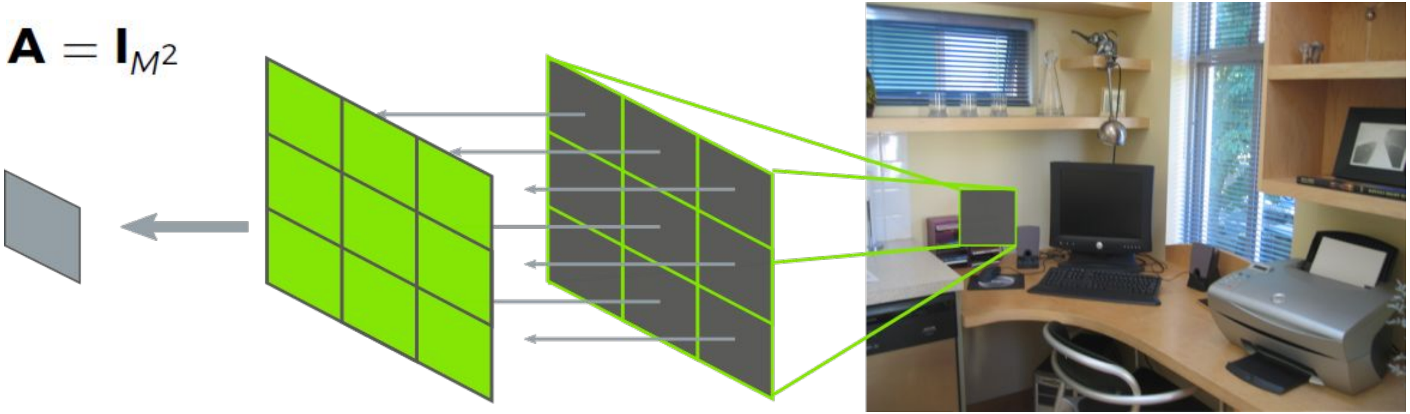
Convolution on images

Convolution for image processing

$$\mathbf{h}[n] = \sum_{f \in \{1,\dots,C\}} \underbrace{\mathbf{K}_f^\top}_{\text{Kernel space}} \mathbf{A} \underbrace{\mathbf{f}_f(n)}_{\text{Feature space}}$$

With **A** the alignment matrix

Image processing: $\mathbf{A} = \mathbf{I}_{M^2}$



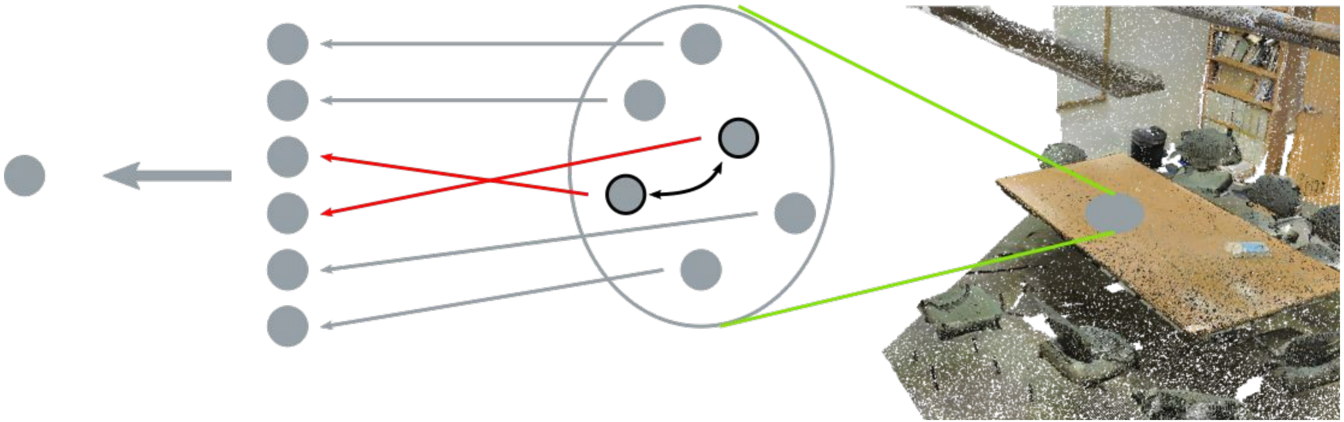
Convolution for points

Convolution for points

Apply the same formula on a small set of points:

$$\mathbf{h}[n] = \sum_{f \in \{1,\ldots,C\}} \underbrace{\mathbf{K}_f^\top}_{\text{Kernel space}} \mathbf{A} \underbrace{\mathbf{f}_f(n)}_{\text{Feature space}}$$

Problem: **A** is not permutation invariant



Convolution on points

Convolution on points

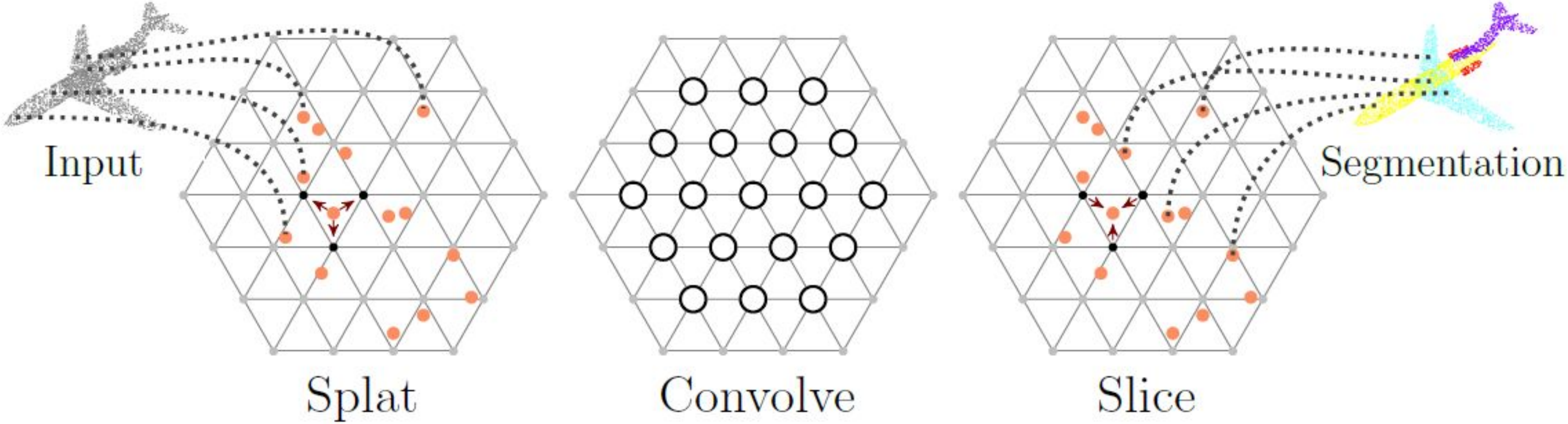
A must be estimated from the neighborhood N of n:

$$\mathbf{h}[n] = \sum_{f \in \{1,\dots,C\}} \underbrace{\mathbf{K}_f^\top}_{\text{Kernel space}} \mathbf{A}(\mathcal{N}) \underbrace{\mathbf{f}_f(n)}_{\text{Feature space}}$$

SplatNet

SplatNet

Estimation of A: Interpolation of the features on a regular grid (barycentric



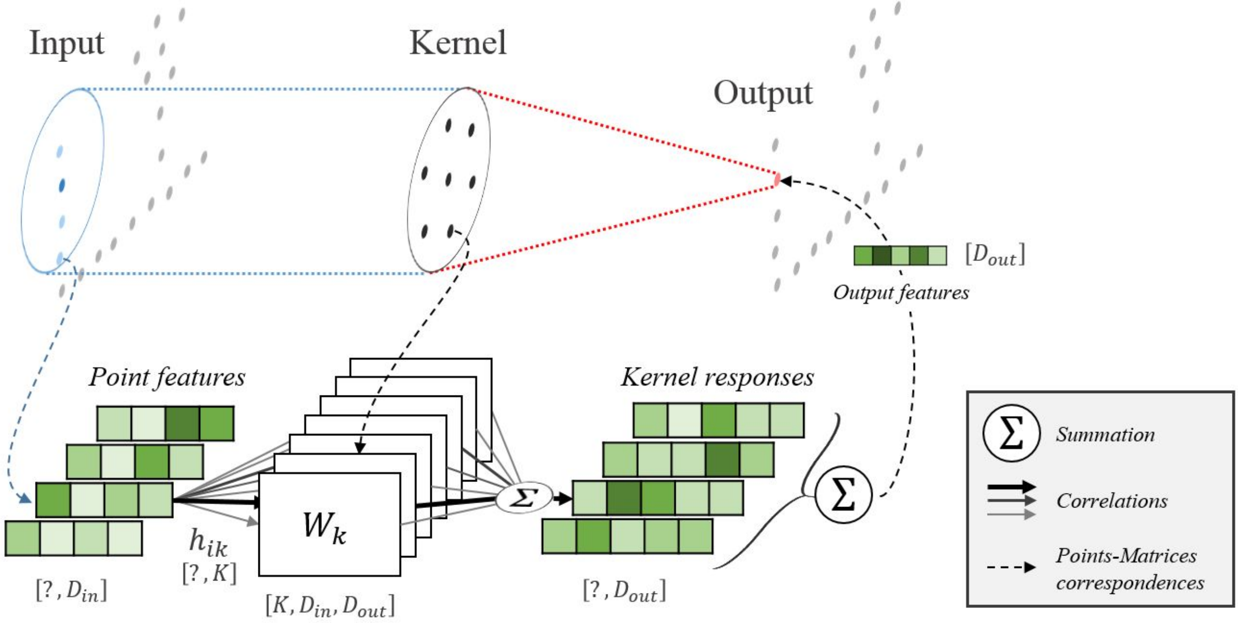
Hang Su et al. “SPLATNet: Sparse Lattice Networks for Point Cloud Processing”. In: arXiv preprint arXiv:1802.08275 (2018)

KPConv

KPConv

$$a_{i,j} = a(y_i, \hat{x}_j) = \max\left(0, 1 - \frac{\|y_i - \hat{x}_j\|}{\sigma}\right)$$

Estimation of A: Create kernel locations in space, weighted interpolation to all kernel
location based on distance.



Hugues Thomas et al. “Kpconv: Flexible and deformable convolution for point clouds”. In: ICCV 2029

ConvPoint

ConvPoint

Estimation of A: Create kernel locations in space, weighted interpolation learned with MLP.

$$a_{i,j} = a(y_i, \hat{x}_j) = \mathrm{MLP}(y_i - \hat{x}_j)$$

Optimization of both MLP weights and kernel point positions.

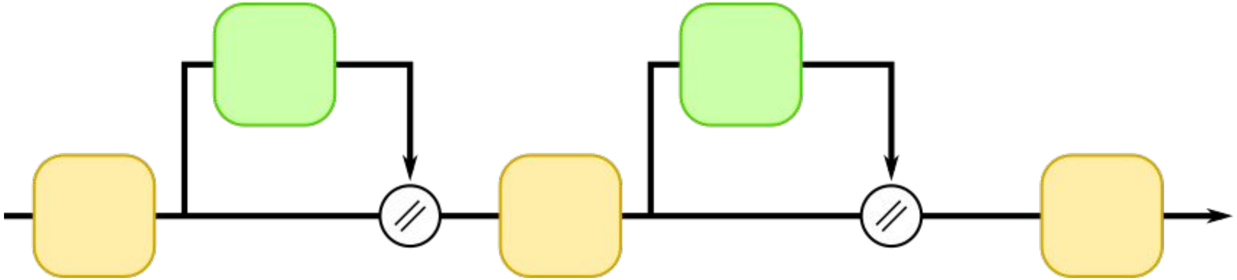
“Generalizing discrete convolutions for unstructured point clouds”. In: Computer and Graphics 2020

FKAConv

FKAConv

Estimation of A: Direct estimation of A using a mini-PointNet.

$$a_{i,j} = a_i(\hat{x}_j) = \mathrm{MLP}_i(\hat{x}_j, \{\hat{x}_k\}_k) \approx \mathrm{PointNet}(\{\hat{x}_k\}_k)$$



Neighborhood search

Convolution is a local operation.

- K-nearest neighbors search
- Ball search

K-nearest neighbors search

Let q be the support point (center of the neighborhood):

$$\text{argtop-K}_{\mathbf{p} \in P}\{-||\mathbf{p} - \mathbf{q}||\}$$

Pros:

- All neighborhoods have the same cardinal
- Relatively fast

Cons:

- Neighborhoods scales vary

Ball search

Ball search

Let q be the support point (center of the neighborhood):

$$\{\mathbf{p} \in P, s.t. ||\mathbf{p} - \mathbf{q}|| < r\}$$

with r the ball radius.

Pros:

- All neighborhoods have the same scale

Cons:

- Neighborhoods cardinals (number of points) vary
- Usually slower than K-nn

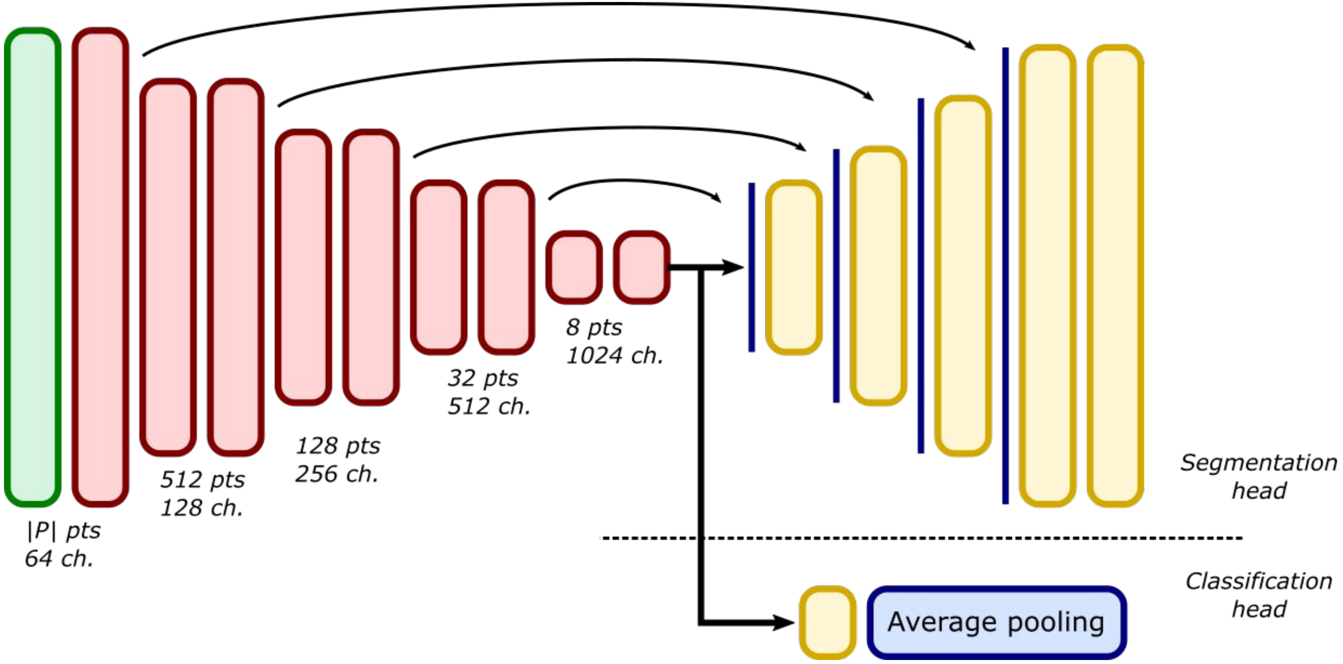
I - Convolutions on points

B - Sampling

Progressive dimension reduction

Progressive dimension reduction

What is the equivalent of stride for convolution on points ?

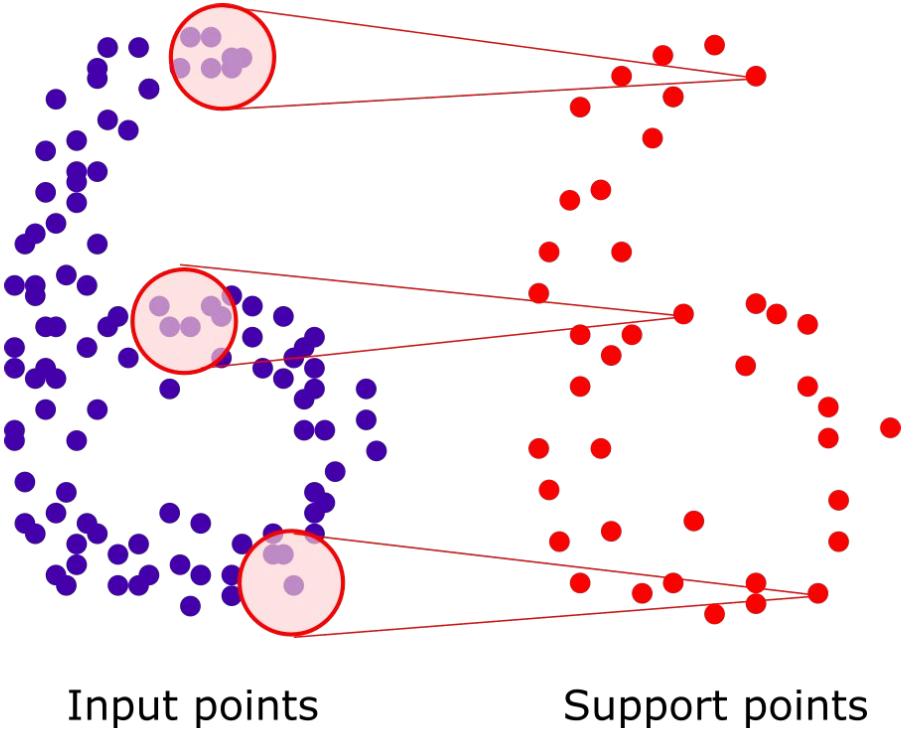


Support point sampling

Support point sampling

Q (Support points), points used as neighborhood centers for the convolution operation.

Usually Q is a subset of P



Random sampling

Random sampling

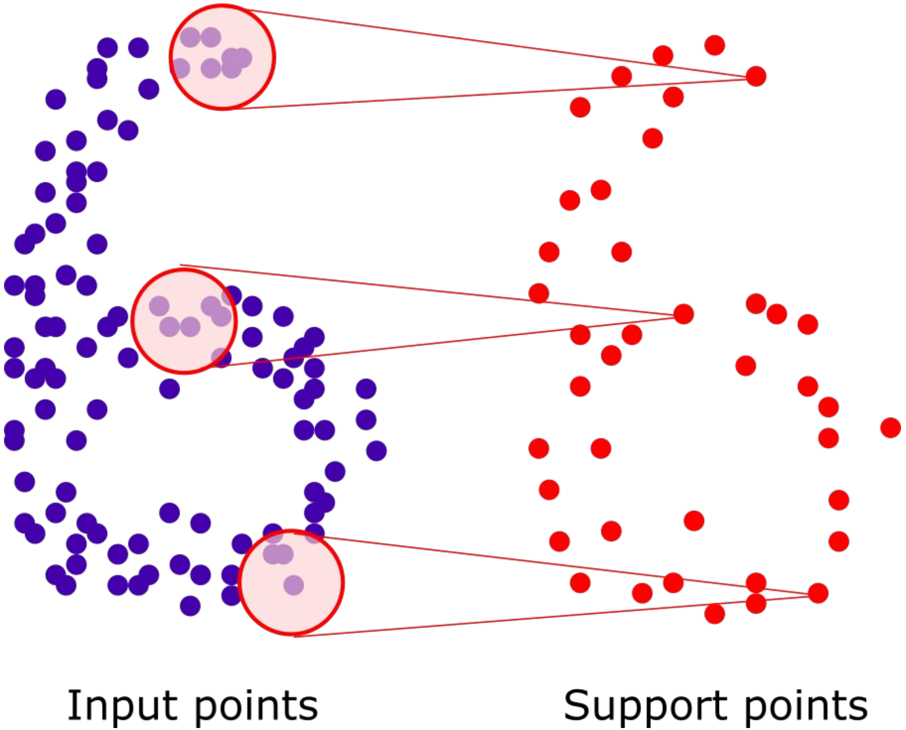
Uniform selection of the input points.

Pros:

- simple and fast.

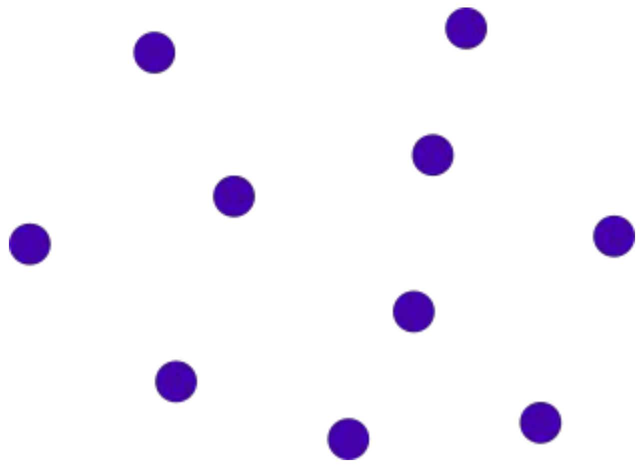
Cons:

- loss of geometric information on area with low density or extreme points.



Furthest point sampling

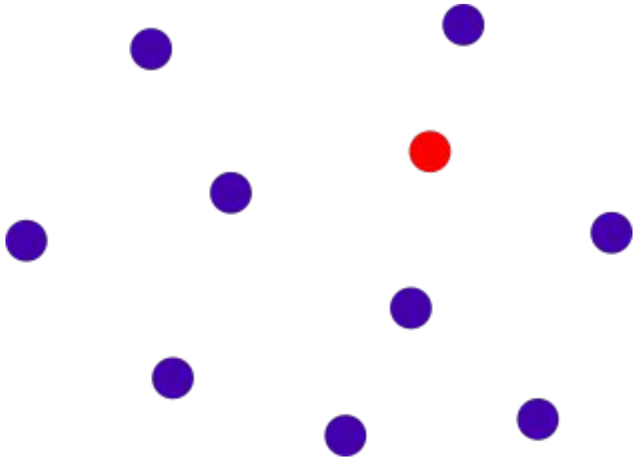
Introduced in PointNet++: iteratively select the further point from the previously selected.



Furthest point sampling

Furthest point sampling

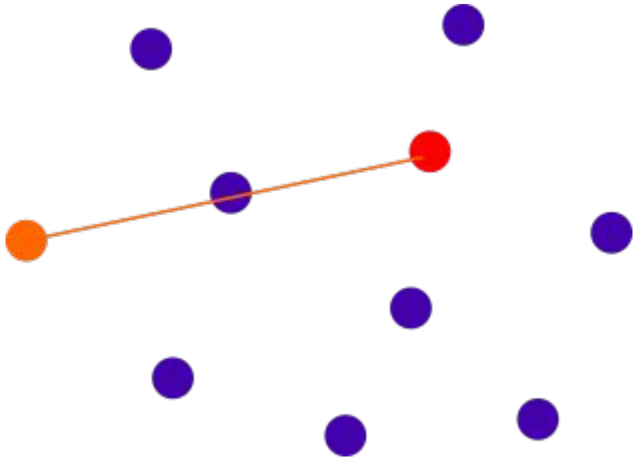
Introduced in PointNet++: iteratively select the further point from the previously selected.



Furthest point sampling

Furthest point sampling

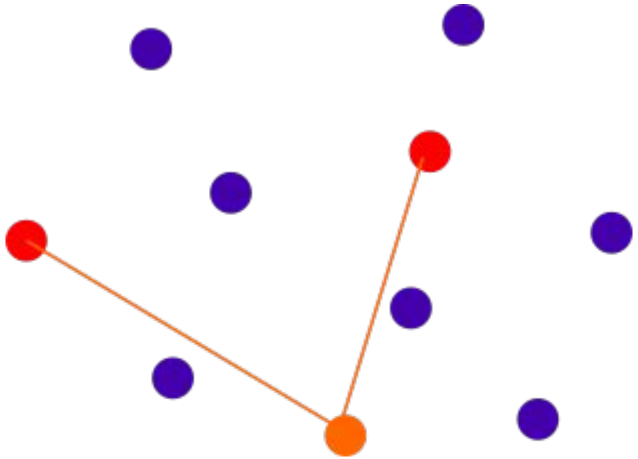
Introduced in PointNet++: iteratively select the further point from the previously selected.



Furthest point sampling

Furthest point sampling

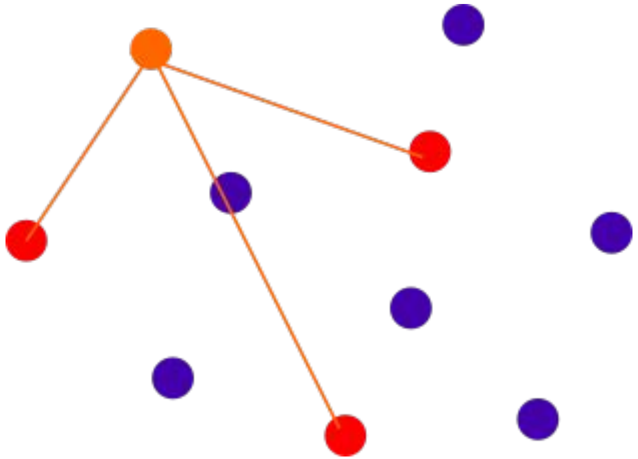
Introduced in PointNet++: iteratively select the further point from the previously selected.



Furthest point sampling

Furthest point sampling

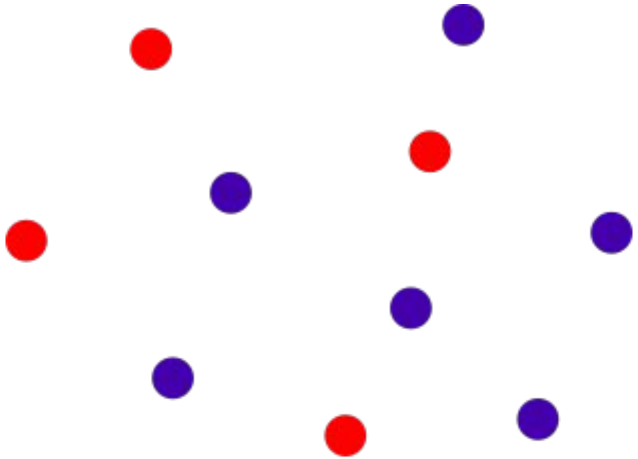
Introduced in PointNet++: iteratively select the further point from the previously selected.



Furthest point sampling

Furthest point sampling

Introduced in PointNet++: iteratively select the further point from the previously selected.



Voxel-grid sampling

Voxel-grid sampling

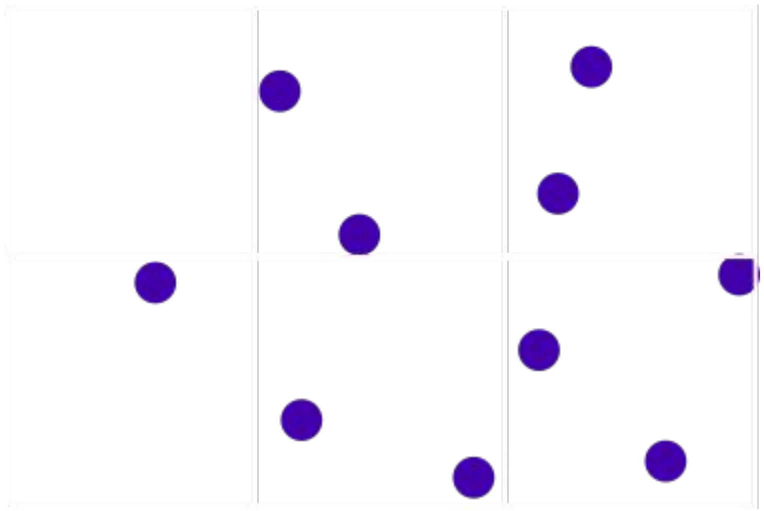
Apply a voxel pooling: select a point in each voxel.

Pros:

- fast

Cons:

- voxel size is extra parameter, may lead to variable number of points



Voxel-grid sampling

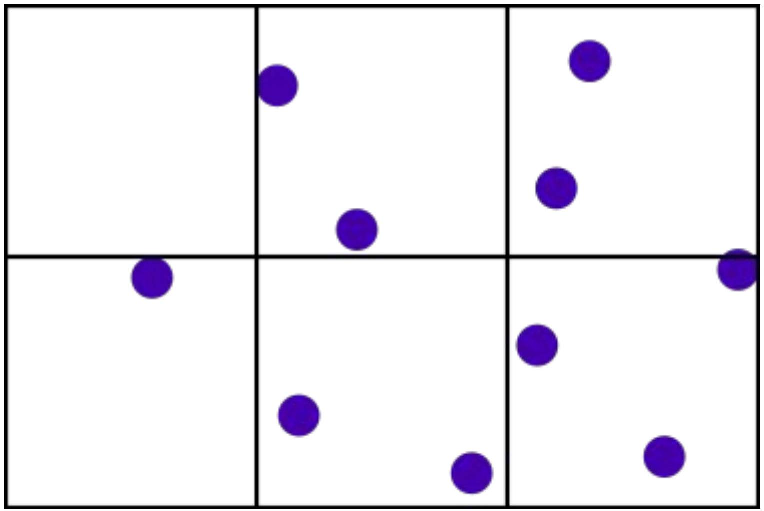
Apply a voxel pooling: select a point in each voxel.

Pros:

- fast

Cons:

- voxel size is extra parameter, may lead to variable number of points



Voxel-grid sampling

Voxel-grid sampling

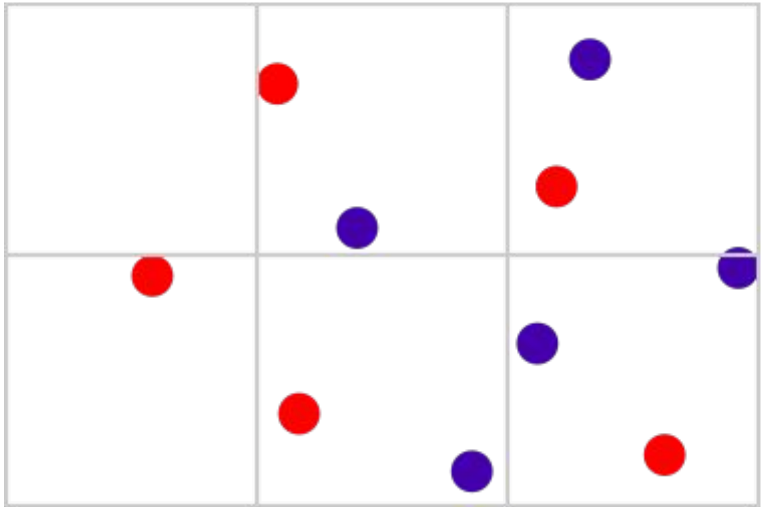
Apply a voxel pooling: select a point in each voxel.

Pros:

- fast

Cons:

- voxel size is extra parameter, may lead to variable number of points



Voxel-grid sampling

Voxel-grid sampling

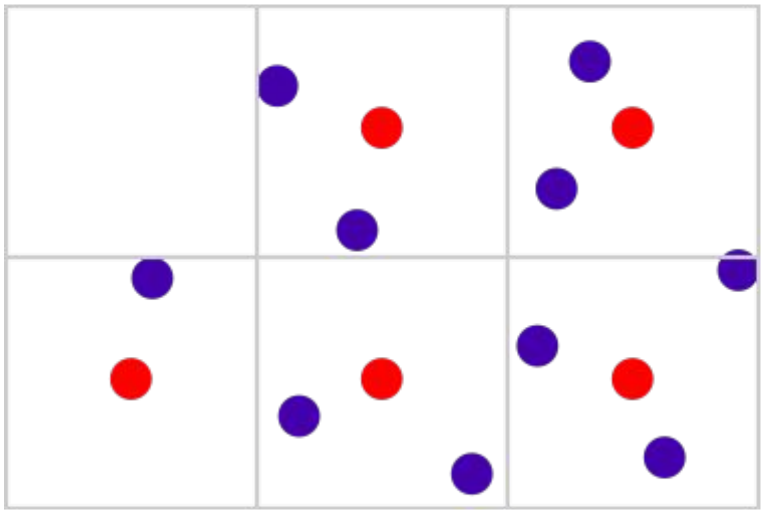
Apply a voxel pooling: select a point in each voxel.

Pros:

- fast

Cons:

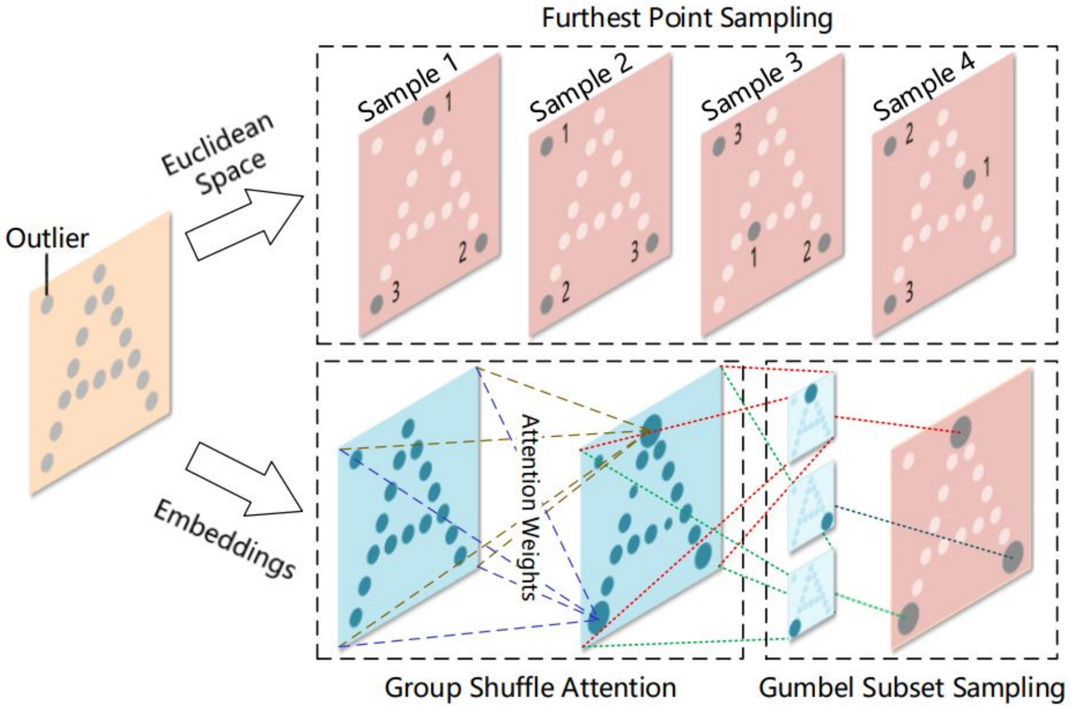
- voxel size is extra parameter, may lead to variable number of points



Attention pooling

Attention pooling

Learned attention on the points for outlier robustness.



Jiancheng Yang et al. “Modeling point clouds with self-attention and gumbel subset sampling”. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019, pp. 3323–3332

II - Voxels

3D grid convolution

II. Voxels

valeo.ai

3D convolution for an grid patch centered on n:

$$\mathbf{h}[n] = \sum_{f \in \{1,\ldots,C\}} \sum_{m \in \{-M/2,\ldots,M/2\}^3} \mathbf{K}_f[m]\, \mathbf{f}_f[n+m]$$

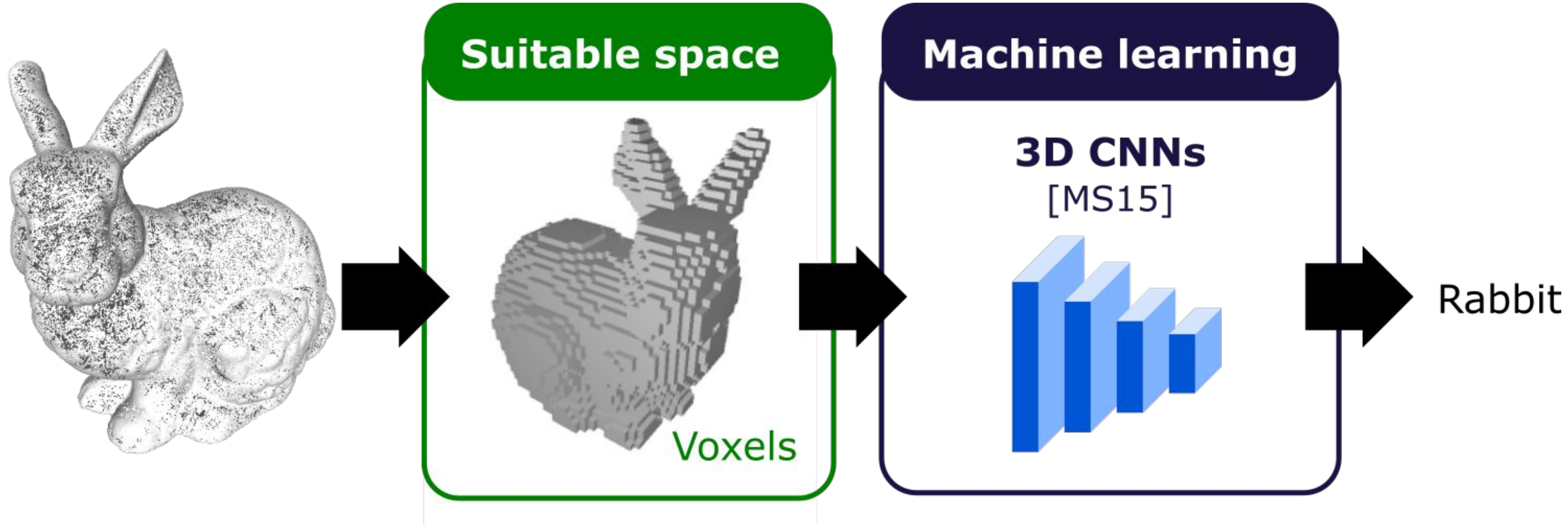
f: input features

K: convolution kernel

How to represent the scene as a 3D grid?

3D projections (voxels)

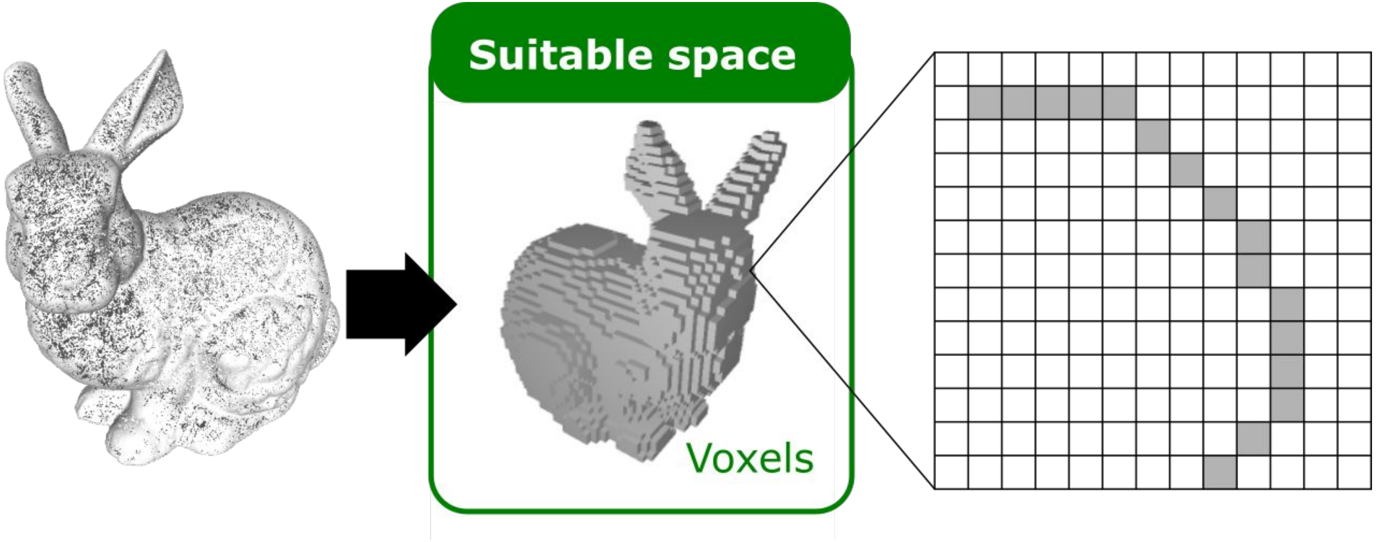
II. Voxels



Daniel Maturana and Sebastian Scherer. “Voxnet: A 3D convolutional neural network for real-time object recognition”. In: Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on. IEEE. 2015, pp. 922–928

Memory

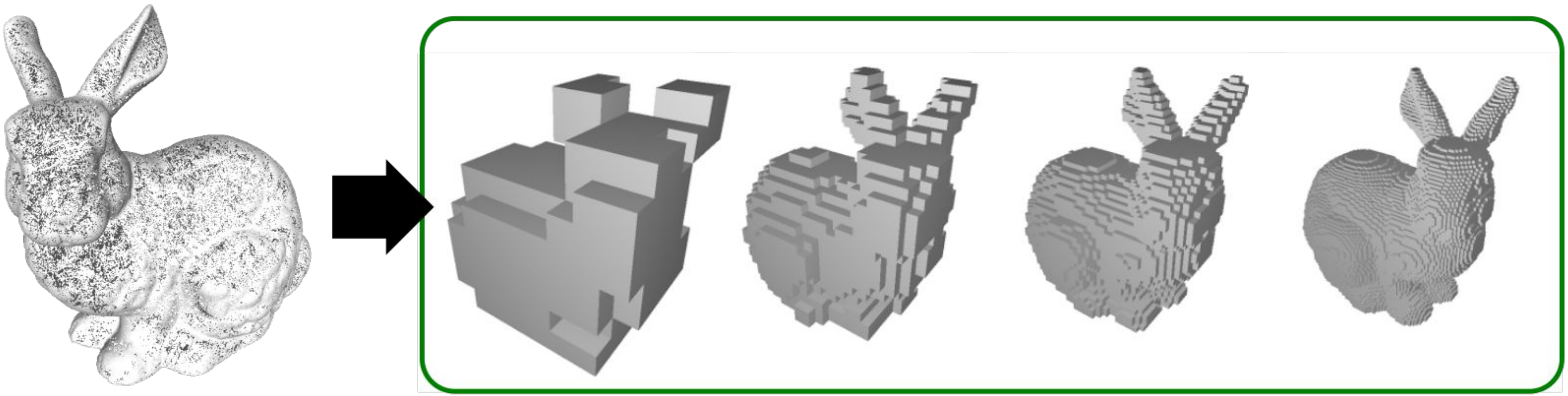
II. Voxels



Point clouds sampled on surfaces are very sparse.
We mostly encode empty voxels!

Memory vs representation power

II. Voxels



Memory efficience vs information loss

Are voxels doomed?

II. Voxels

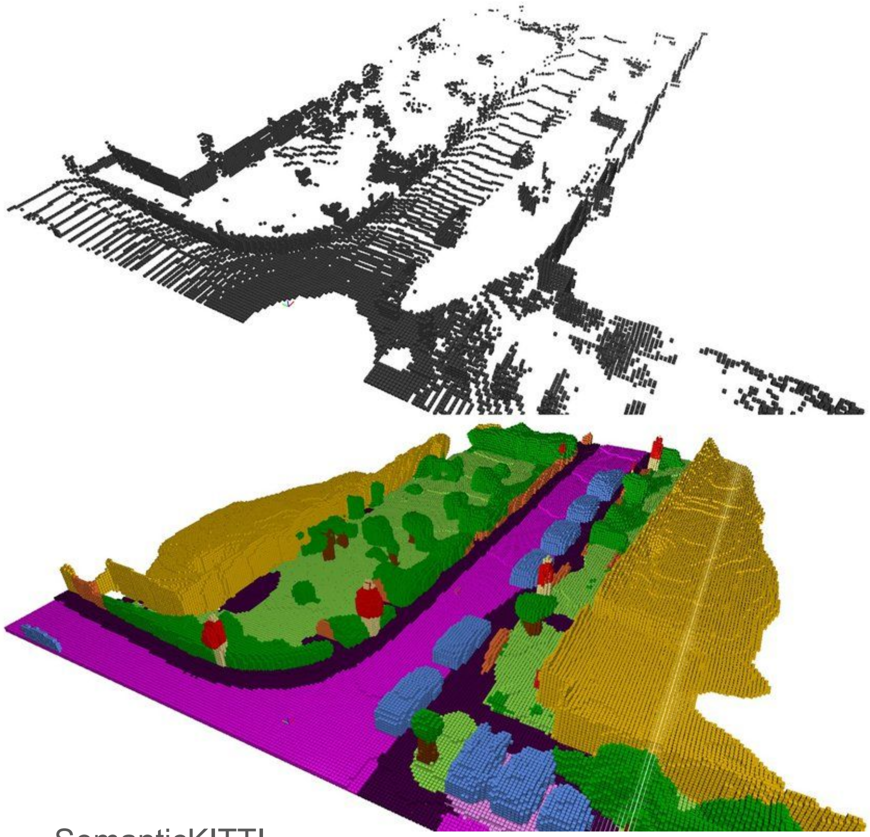
Voxels are OK for small scenes:

Shapes:
→ 32x32x32 = 32768 voxels

Scenes:

→ [100m,100m, 10m], vox 0.05:
800M voxels

While for a lidar point cloud only ~150k voxels are filled (0.02%)

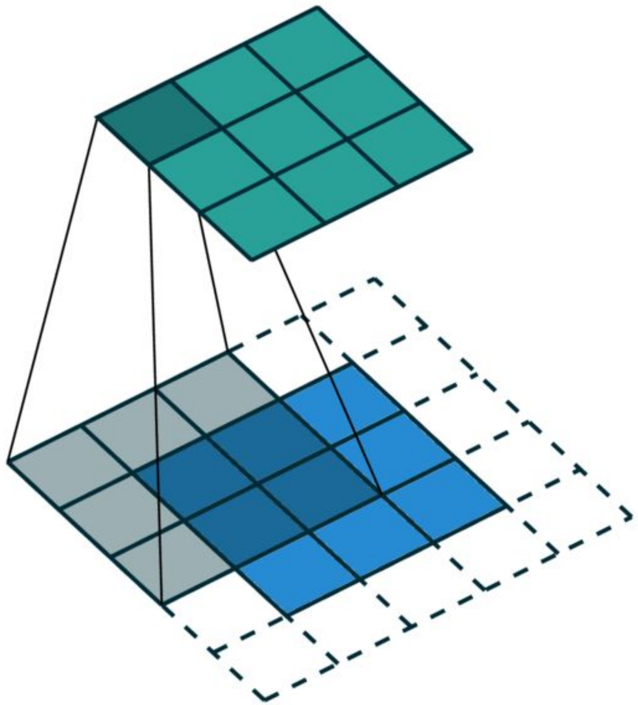


SemanticKITTI

Idea?

II. Voxels

Look at the functioning of the convolution for dense input



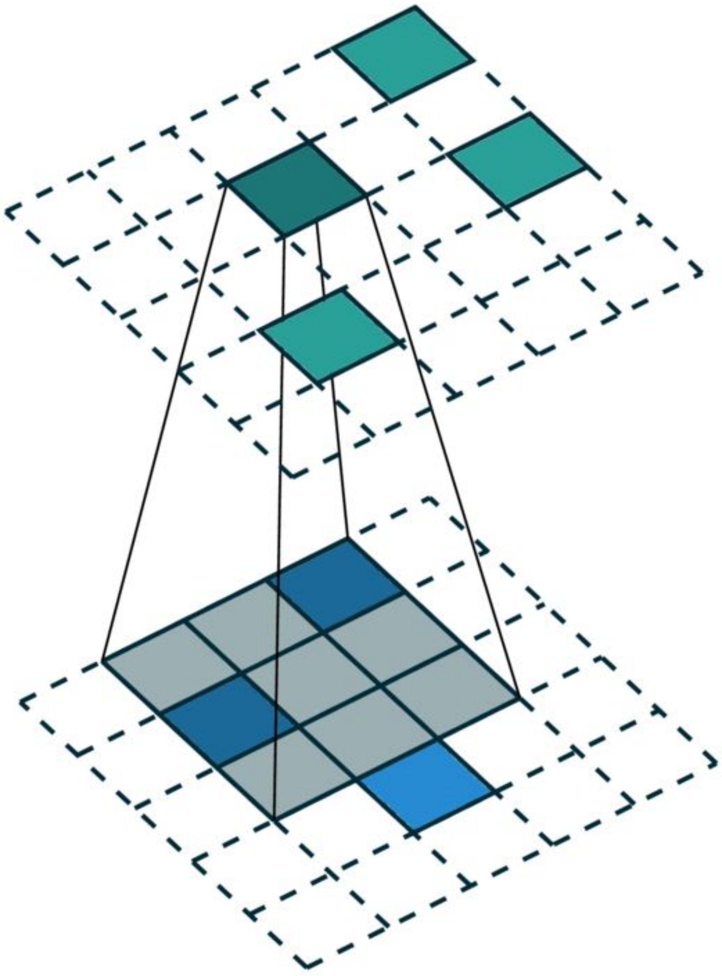
Idea?

II. Voxels

Look at the functioning of the convolution for dense input

Mimic the behavior only at point location

→ sparse convolution



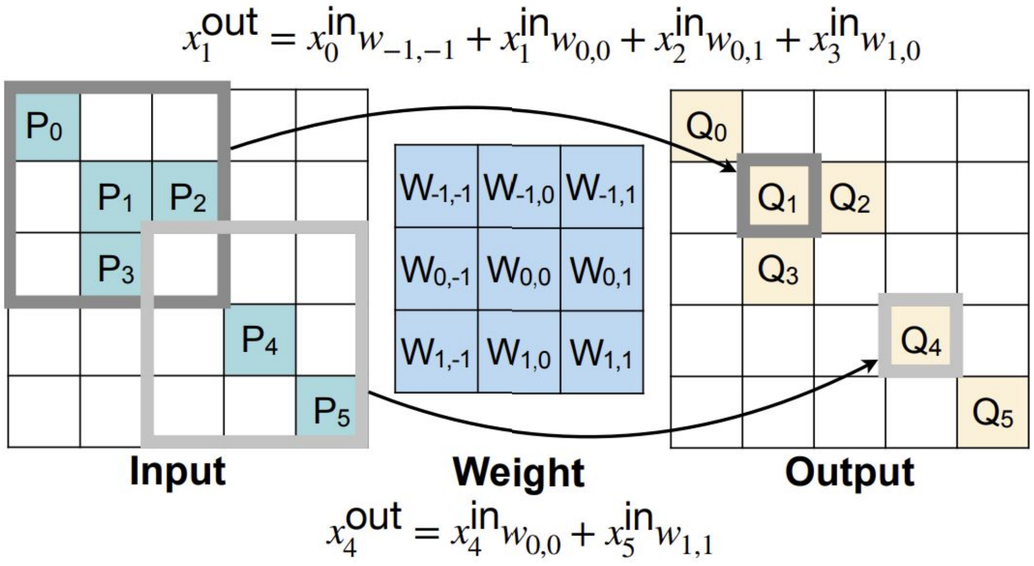
Sparse convolutions

II. Voxels

Look at the functioning of the convolution for dense input

Mimic the behavior only at point location

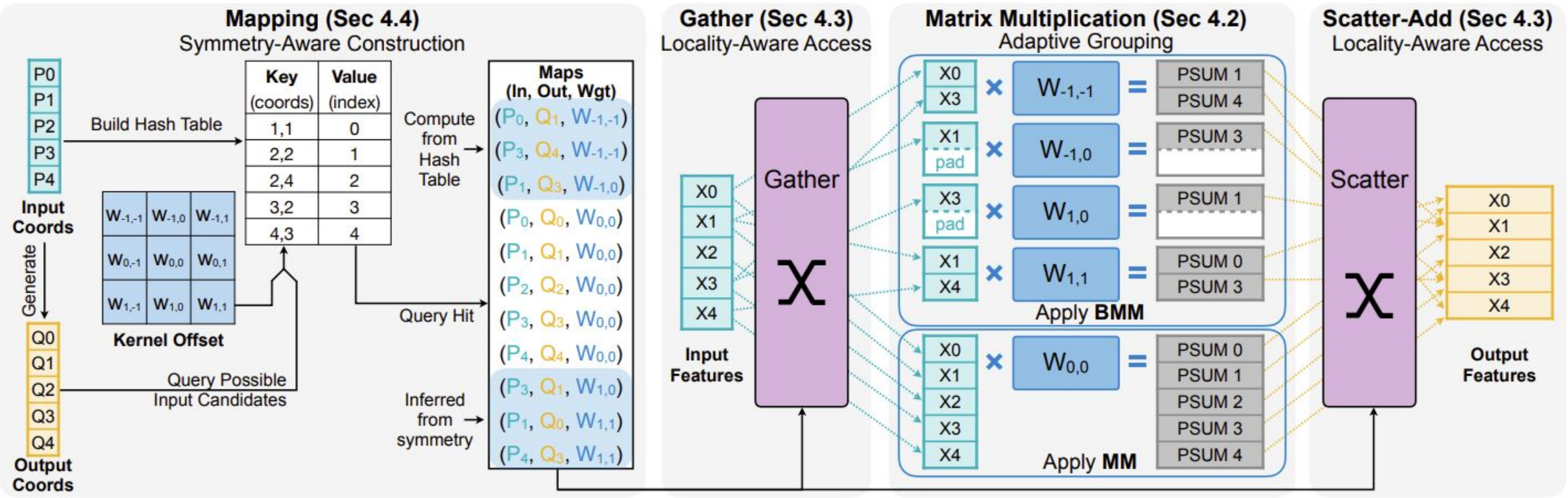
→ sparse convolution



Tang, Haotian, et al. "Torchsparse: Efficient point cloud inference engine." Proceedings of Machine Learning and Systems 4 (2022): 302-315.

Sparse convolutions

II. Voxels



Tang, Haotian, et al. "Torchsparse: Efficient point cloud inference engine." Proceedings of Machine Learning and Systems 4 (2022): 302-315.

Alternative: sparse convolutions

II. Voxels

Use sparse convolution for memory saving: do not code the empty cells.

- Minkowski engine (NVidia)
- SparseConvNet (Facebook)
- Torchsparse
- Spconv

Drawback: slower than dense convolution, extensive use of CPUs.

Only available for NVidia hardware

III - Mixers and transformers

III - Mixers and transformers

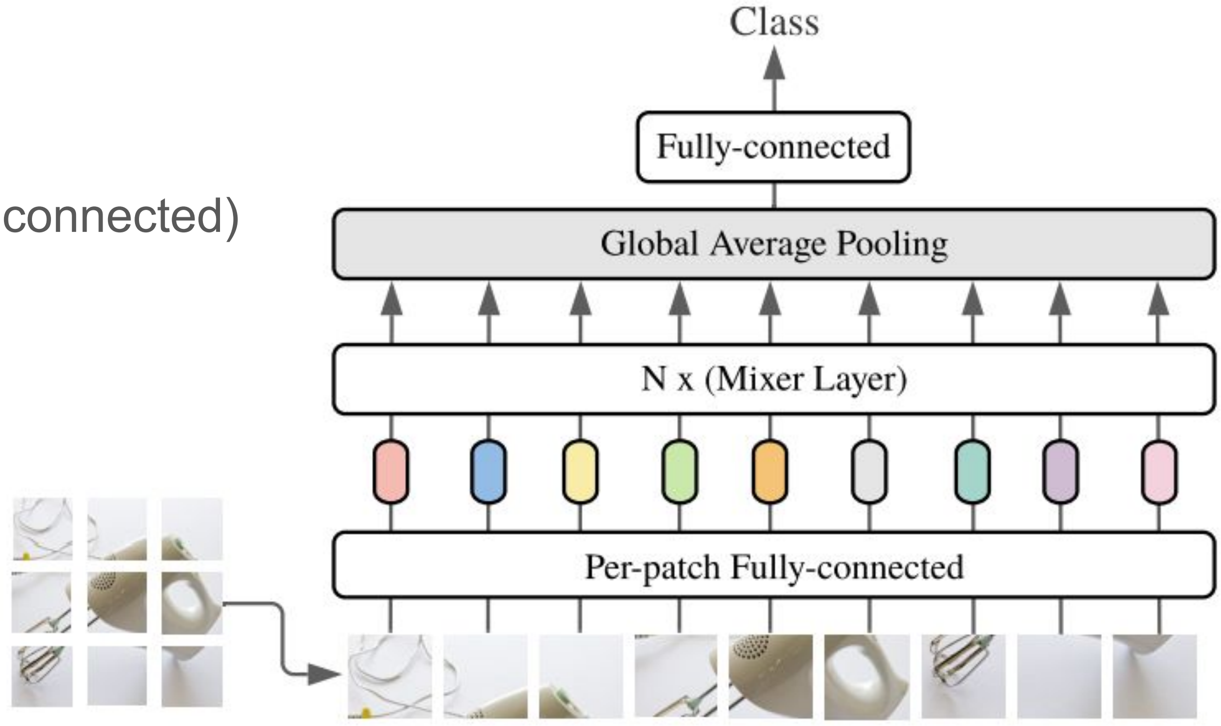
A - Mixers

MLP-Mixer

III-A Mixers

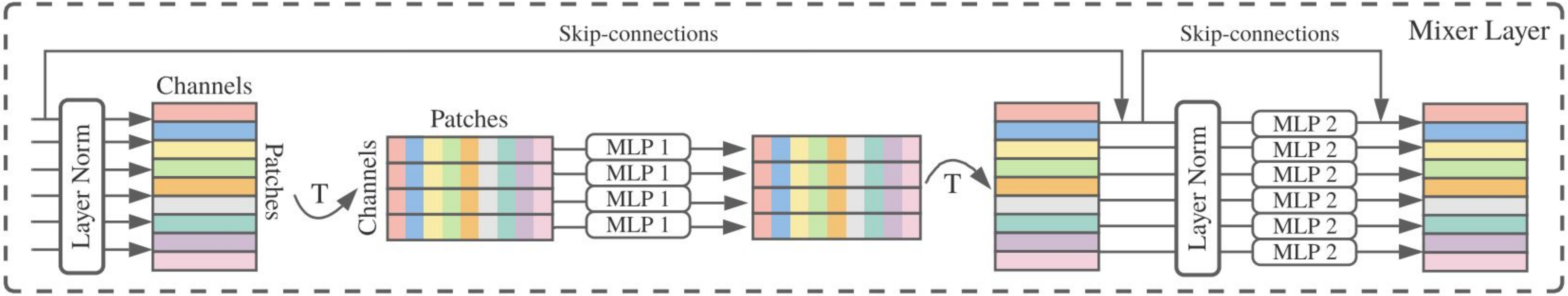
Image backbone

- Patchification
- Patch encoding (fully connected)
- N x Mixer Layer
- Global pooling
- Classification head



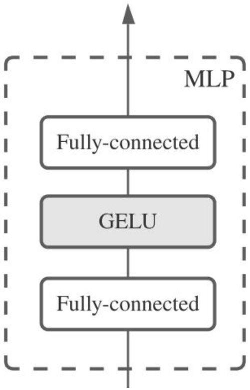
MLP-Mixer

III-A Mixers



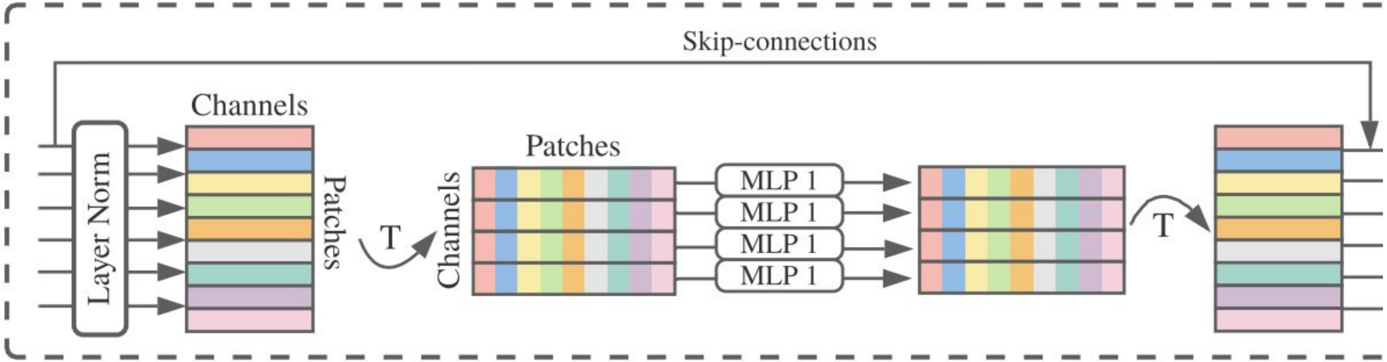
Two sub-blocks:

- **Spatial Mixing:** mixes the patch per channel
- **Spectral Mixing:** mixes the channels per patch



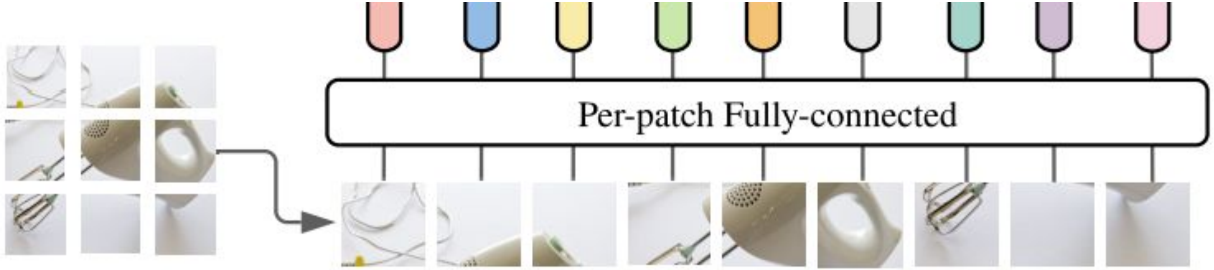
MLP-Mixer

III-A Mixers



Why does it work?

Patches are always in the same order



Incompatible with point clouds

PointMixer

III-A Mixers

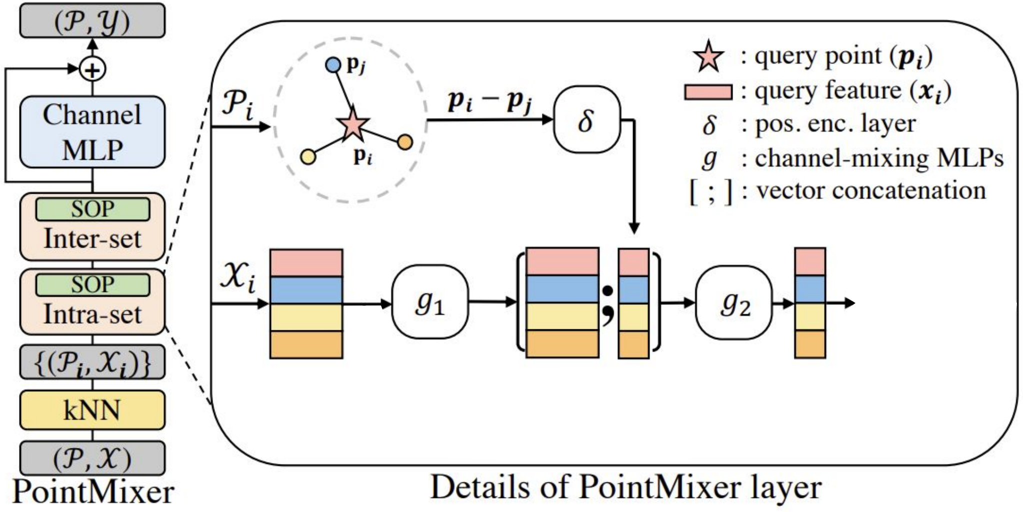
Neighborhood: $\mathcal{X}_i = \{\mathbf{x}_j\}$

1. Predict a score vector

   $\mathbf{s} = [s_1, \ldots, s_K] \quad \mathbf{s} \in \mathbb{R}^K$

   With

$$s_j = g_2\Big(\big[g_1(\mathbf{x}_j); \delta(\mathbf{p}_i - \mathbf{p}_j)\big]\Big)$$



Choe, Jaesung, et al. "Pointmixer: Mlp-mixer for point cloud understanding." *European conference on computer vision*. Cham: Springer Nature Switzerland, 2022.

PointMixer

III-A Mixers

Neighborhood: $\mathcal{X}_i = \{\mathbf{x}_j\}$

1. Predict a score vector

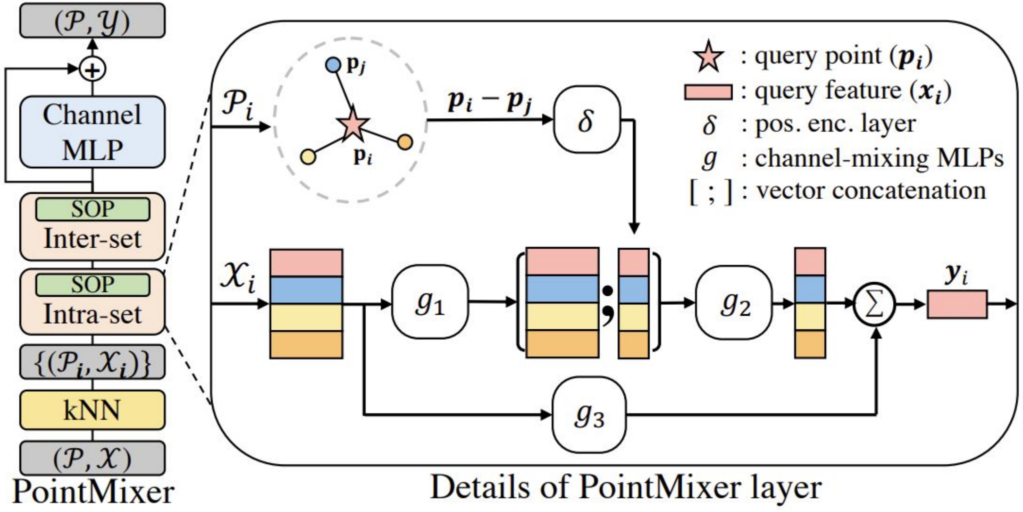
   $\mathbf{s} = [s_1, \ldots, s_K]$ $\mathbf{s} \in \mathbb{R}^K$

   With

$$s_j = g_2\Big(\big[g_1(\mathbf{x}_j); \delta(\mathbf{p}_i - \mathbf{p}_j)\big]\Big)$$

2. Use the scores to weight the features

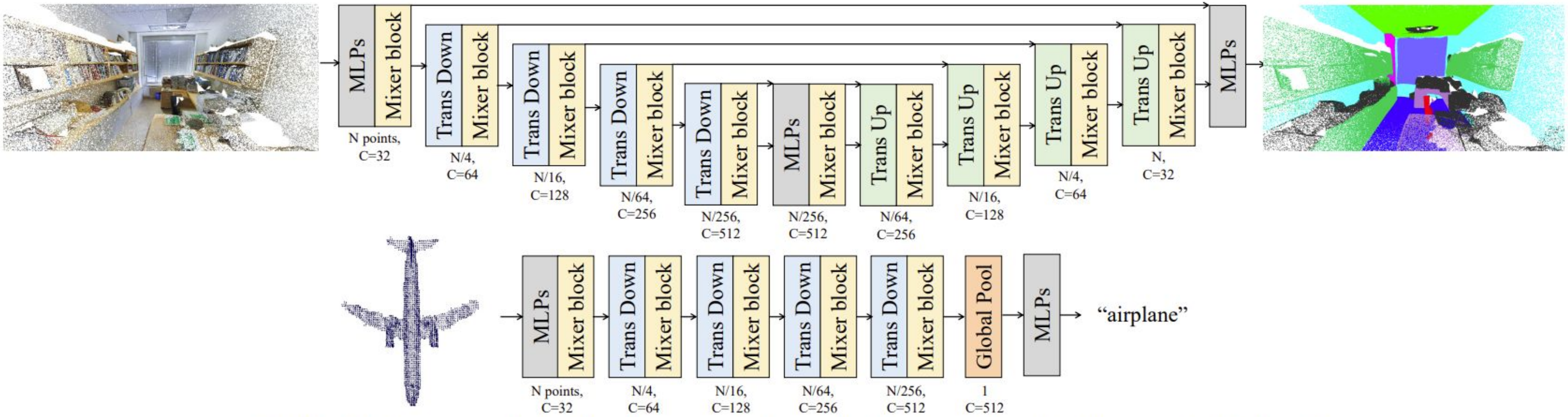
$$\mathbf{y}_i = \sum_{j \in \mathcal{M}_i} \mathrm{softmax}(s_j) \odot g_3(\mathbf{x}_j),$$



Choe, Jaesung, et al. "Pointmixer: Mlp-mixer for point cloud understanding." *European conference on computer vision*. Cham: Springer Nature Switzerland, 2022.

PointMixer

III-A Mixers



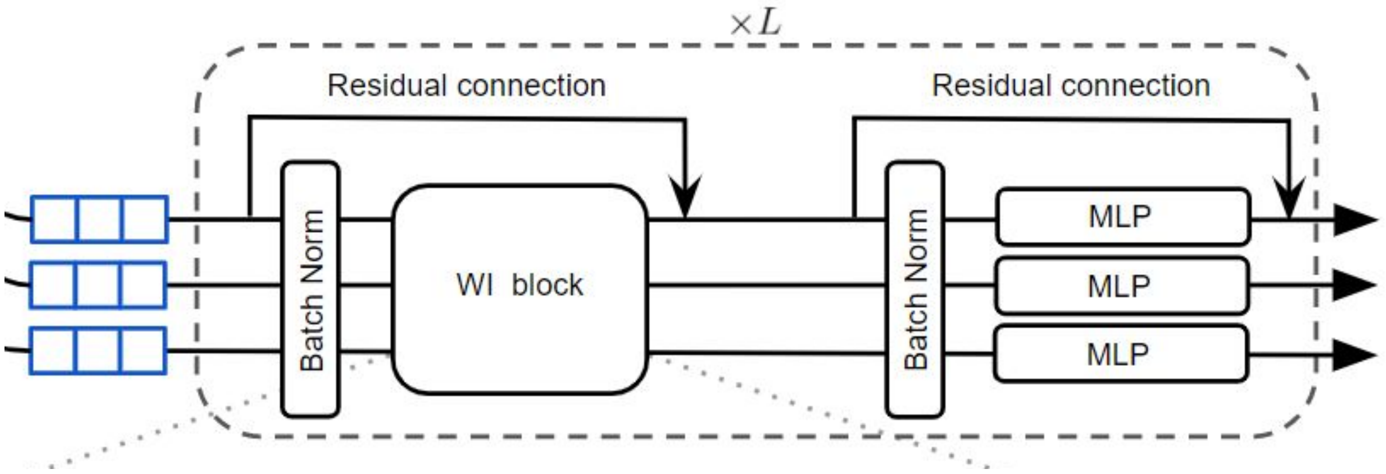
(a) PointMixer network for the dense prediction tasks (top) and the classification task (down).

Architecture: U-Net (closer to convolutional architectures than MLP-Mixers)

Choe, Jaesung, et al. "Pointmixer: Mlp-mixer for point cloud understanding." *European conference on computer vision*. Cham: Springer Nature Switzerland, 2022.

WaffleIron

III-A Mixers



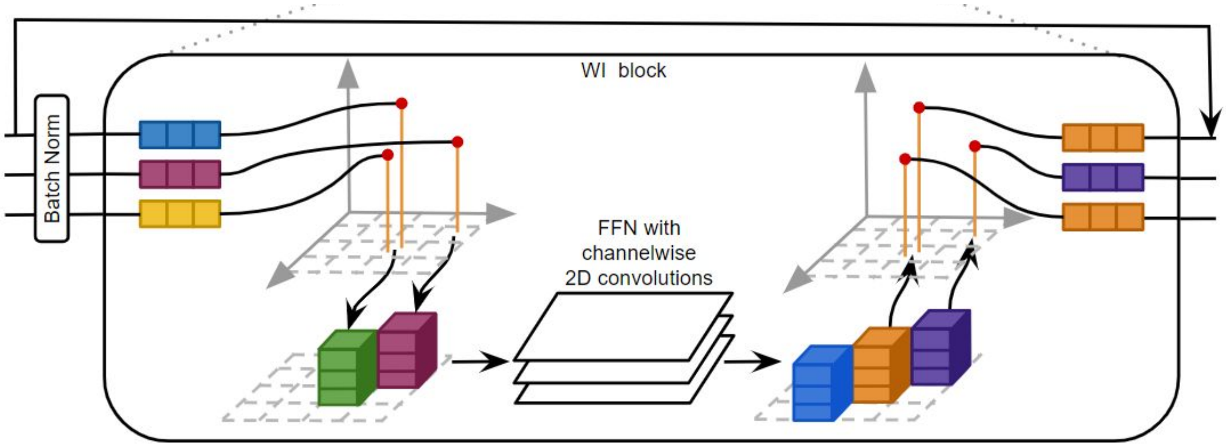
Architecture similar to MLP-Mixer:

- Spatial mixing (WI block)
- Channel mixing (MLP)

Gilles Puy et al., “Using a Waffle Iron for Automotive Point Cloud Semantic Segmentation” ICCV 2023

WaffleIron

III-A Mixers



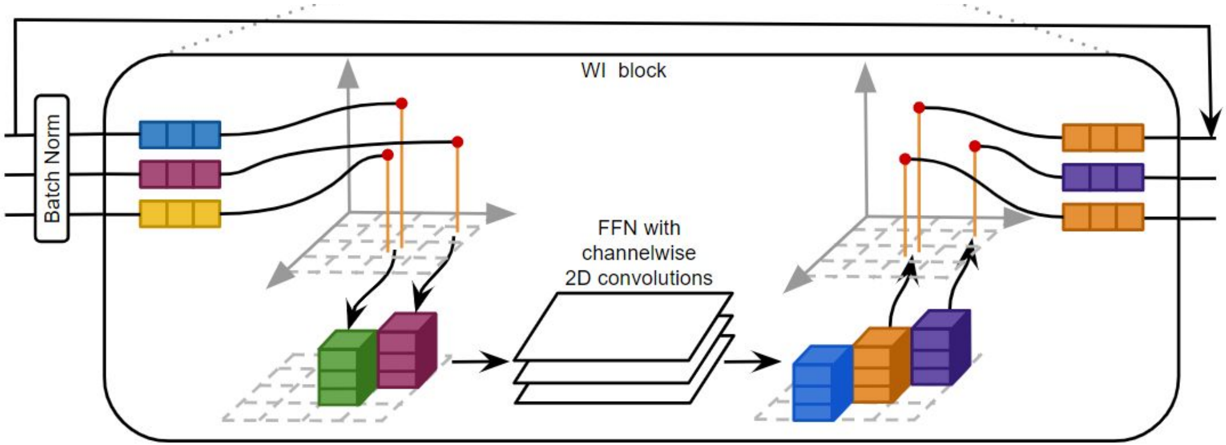
Spatial mixing:

- Project on a plane → makes it order invariant
- Apply convolutions
- Un-project to planes

Gilles Puy et al., “Using a Waffle Iron for Automotive Point Cloud Semantic Segmentation” ICCV 2023

WaffleIron

III-A Mixers



Advantage:

Do not rely on SparseConv → can be used on any hardware / any deep learning framework

Gilles Puy et al., “Using a Waffle Iron for Automotive Point Cloud Semantic Segmentation” ICCV 2023

III - Mixers and transformers

B - Transformers

Transformers

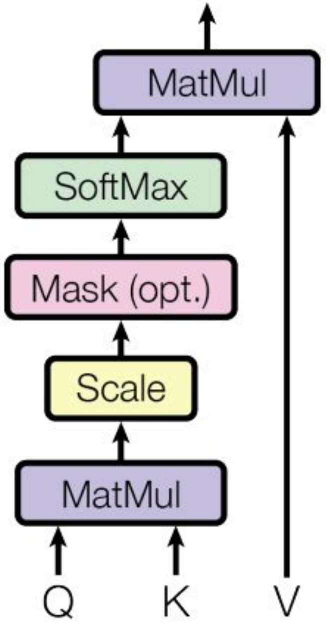
III-B Transformers

Attention as defined for transformers:

- Base block of all recent architectures (LLMs, VLM, ViTs…)
- Order invariant by design

→ Suitable for point clouds

Scaled Dot-Product Attention



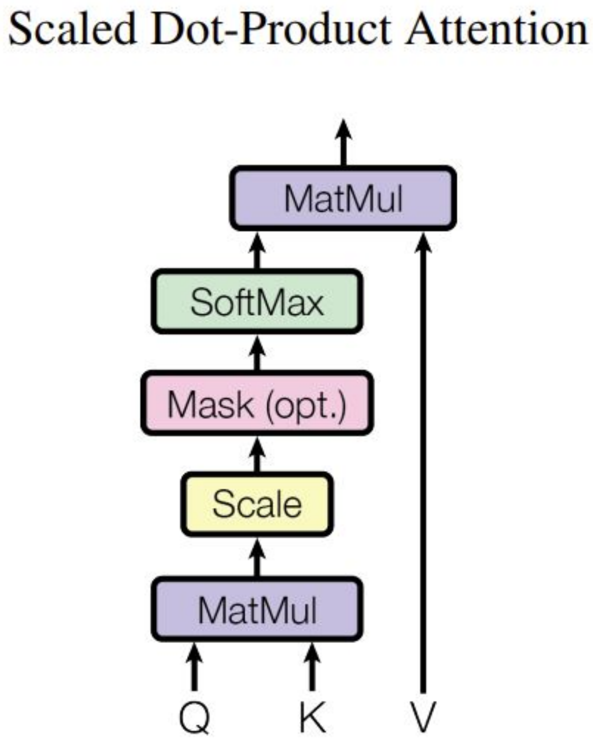
Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).

Transformers

III-B Transformers

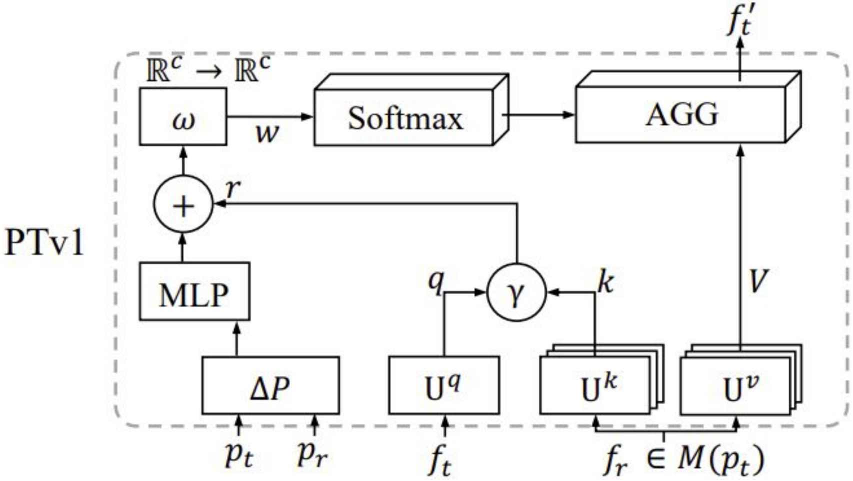
Difficulties

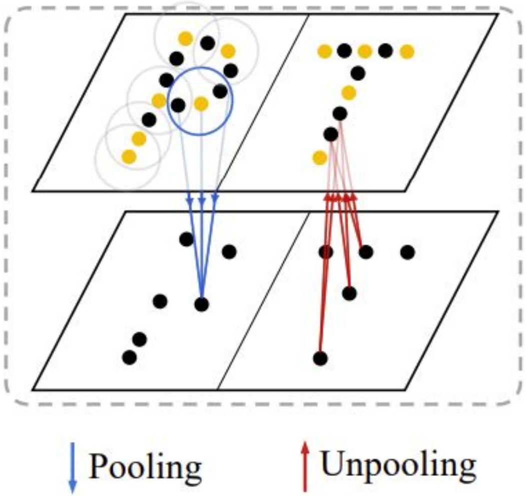
- Attention scales quadratically in memory (naive implementation)
  → Efficient attention, linear depending on the number or queries / keys / values
- Point clouds are large
  → attention matrix resolution may be under the float precision



Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).

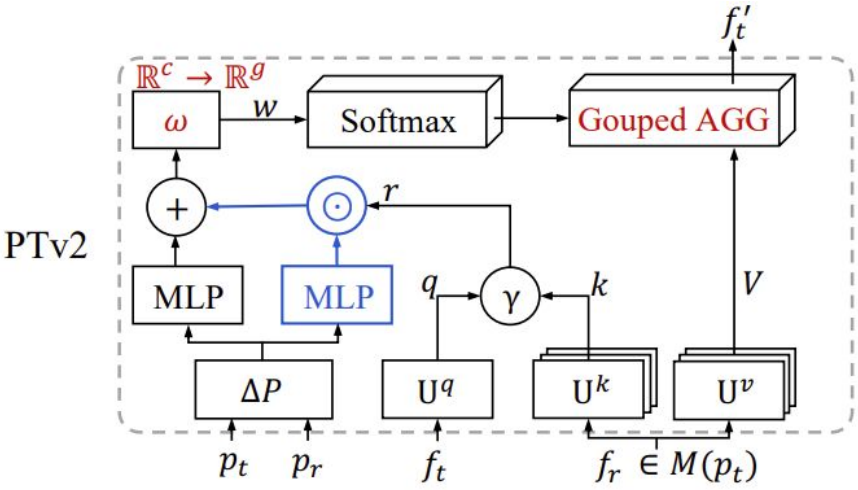
PointTransformer v1

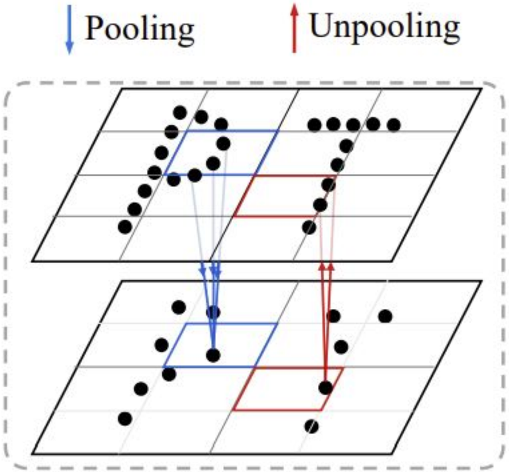




Zhao, Hengshuang, et al. "Point transformer." Proceedings of the IEEE/CVF international conference on computer vision. 2021.

PointTransformer v2





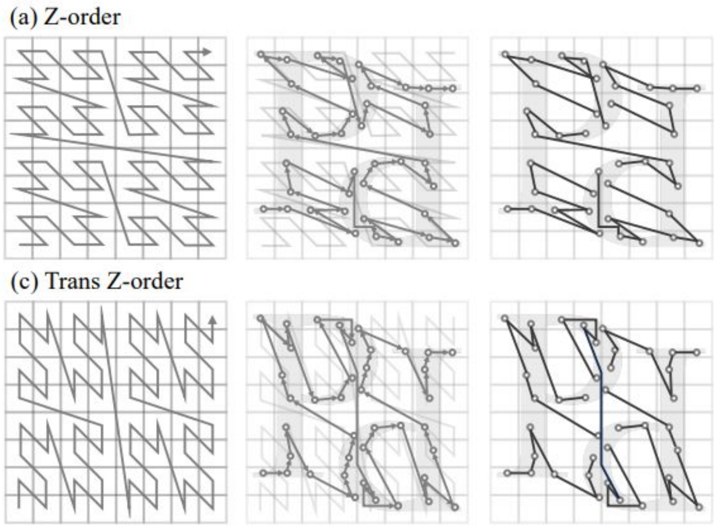
Wu, Xiaoyang, et al. "Point transformer v2: Grouped vector attention and partition-based pooling." Advances in Neural Information Processing Systems 35 (2022): 33330-33342.

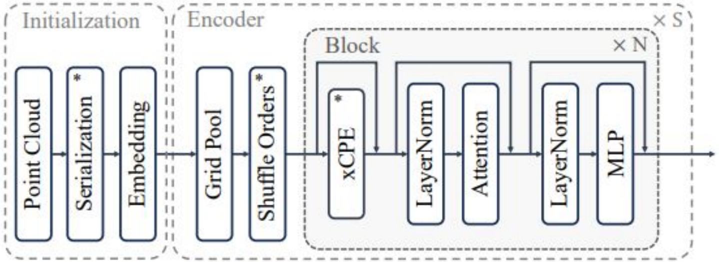
PointTransformer v3

U-Net architectures

Neighborhood defined by space filling curves

Attention on multiple scales





Wu, Xiaoyang, et al. "Point transformer v3: Simpler faster stronger." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2024.

Conclusion

Conclusion

Efficient architectures

- MinkUNet (for everything)
- PTv3 (flexible, sometimes hard to train)
- WaffeIron (outdoor lidar)

Practical sessions

- WaffleIron for part segmentation