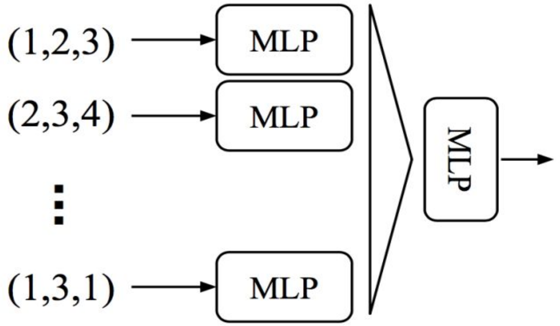
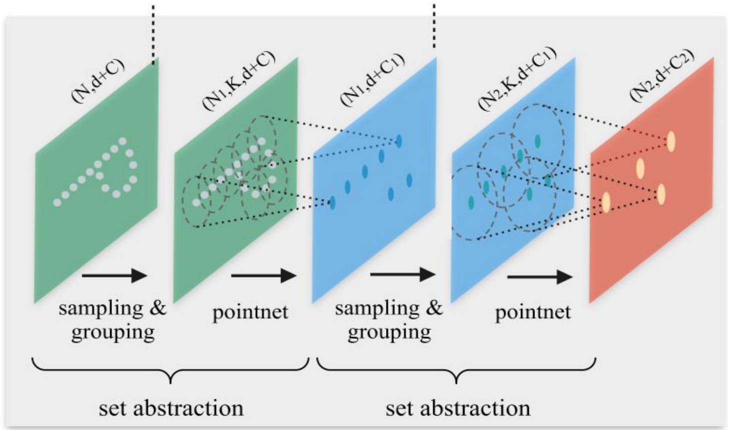
KPConv: Flexible and Deformable Convolution for Point Clouds

Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, Leonidas J. Guibas

Deep Neural Networks for Point Clouds



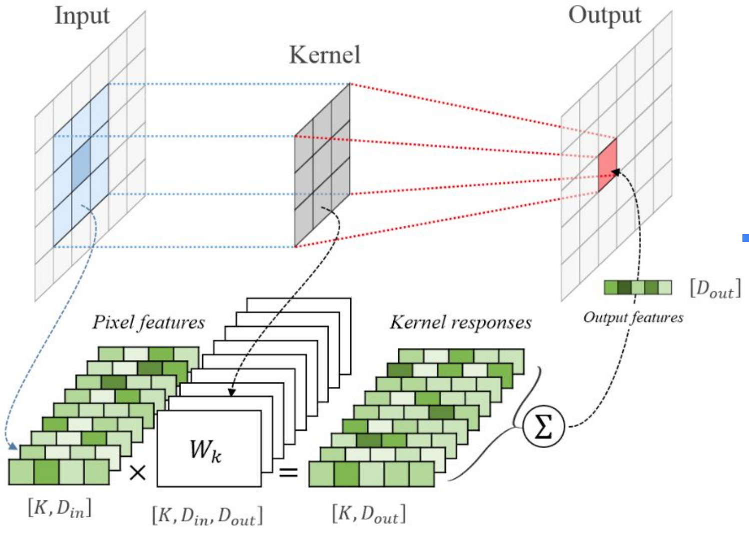


PointNet (vanilla), Qi et al. 2016
Shared MLPs + maxpool

PointNet++, Qi et al. 2017
Sampling + Grouping + PointNet

Pointwise MLPs → Local spatial relationship is overlooked.

Convolution



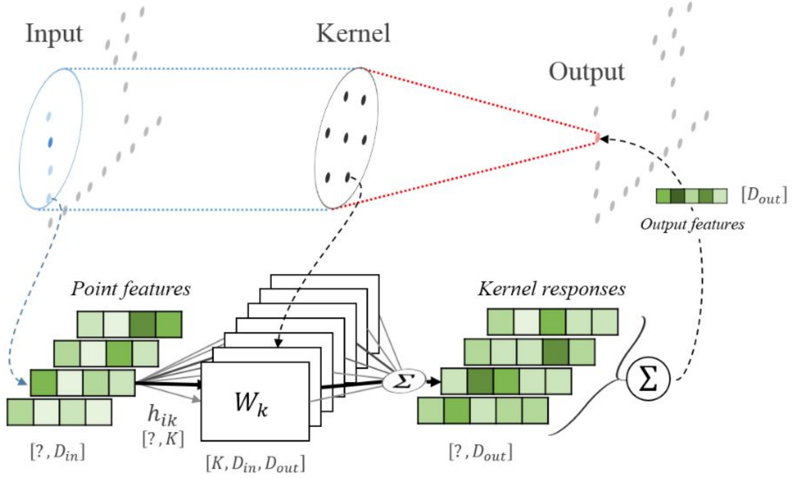


Image Convolution

Apply learnable spatial kernels locally.

Point Convolution

Kernel Point Convolution

- Continuous Spatial Conv

$$\underbrace{(\mathcal{F}}_{\text{Feature}} * \underbrace{g}_{\text{Kernel}})(x) = \int g(y-x)f(y)dy$$

- Empirical Conv on Point Clouds

$$(\mathcal{F} * g)(x) = \sum_{x_i \in \mathcal{N}_x} \boxed{g(x_i - x)} f_i$$

x_i: points from $P \in R^{N*3}$,
f_i: corresponding features from $F \in R^{N*D}$
N_x: $\{x_i \in P \mid \|x_i - x\| \leq r\}$

How to define kernel g on

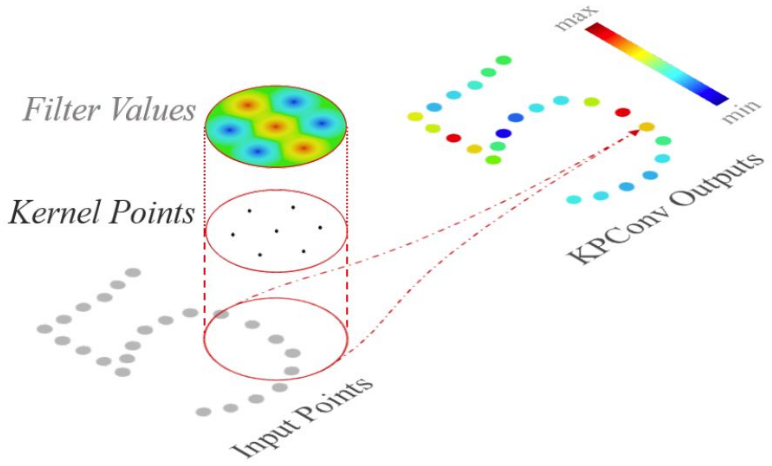
$$\mathcal{B}^3_r = \{y \in \mathbb{R}^3 \mid \|y\| \leqslant r\}$$

Define Kernel Function by Points

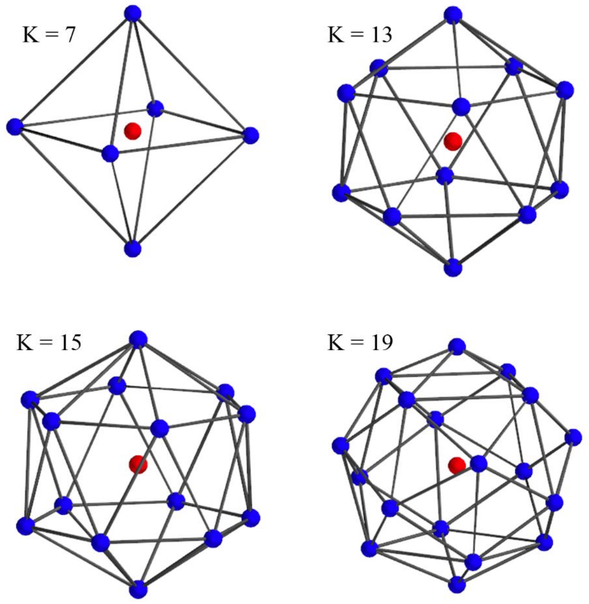
- Define kernel by a set of **kernel points** with weights.
- Kernel points
  $\{\widetilde{x}_k \mid k < K\} \subset \mathcal{B}_r^3$
- Associated learnable weights
  $\{W_k \mid k < K\} \subset \mathbb{R}^{D_{in} \times D_{out}}$
- Propagate to arbitrary position using correlation function h (closer, higher)

$$g(y_i) = \sum_{k<K} \boxed{h\left(y_i, \widetilde{x}_k\right)} W_k$$

$$h\left(y_i, \widetilde{x}_k\right) = \max\left(0, 1 - \frac{\|y_i - \widetilde{x}_k\|}{\sigma}\right)$$



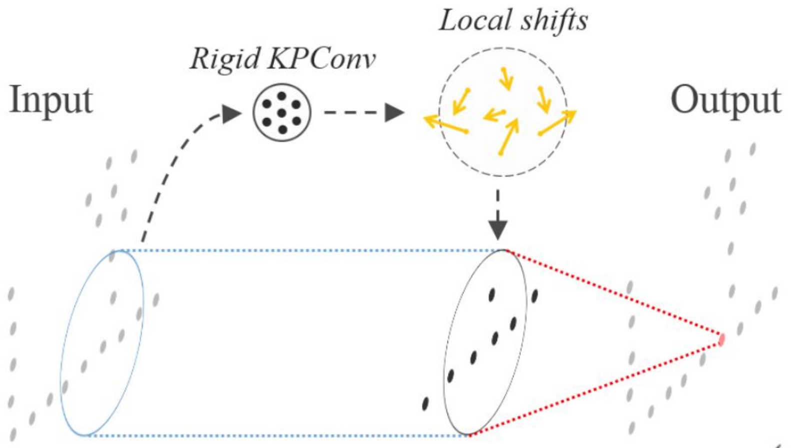
Rigid Kernel



For any K, an arrangement of points can be computed:

- Repulsive force between points → small overlap
- Attractive force to stay in the sphere → good coverage

Deformable Kernel

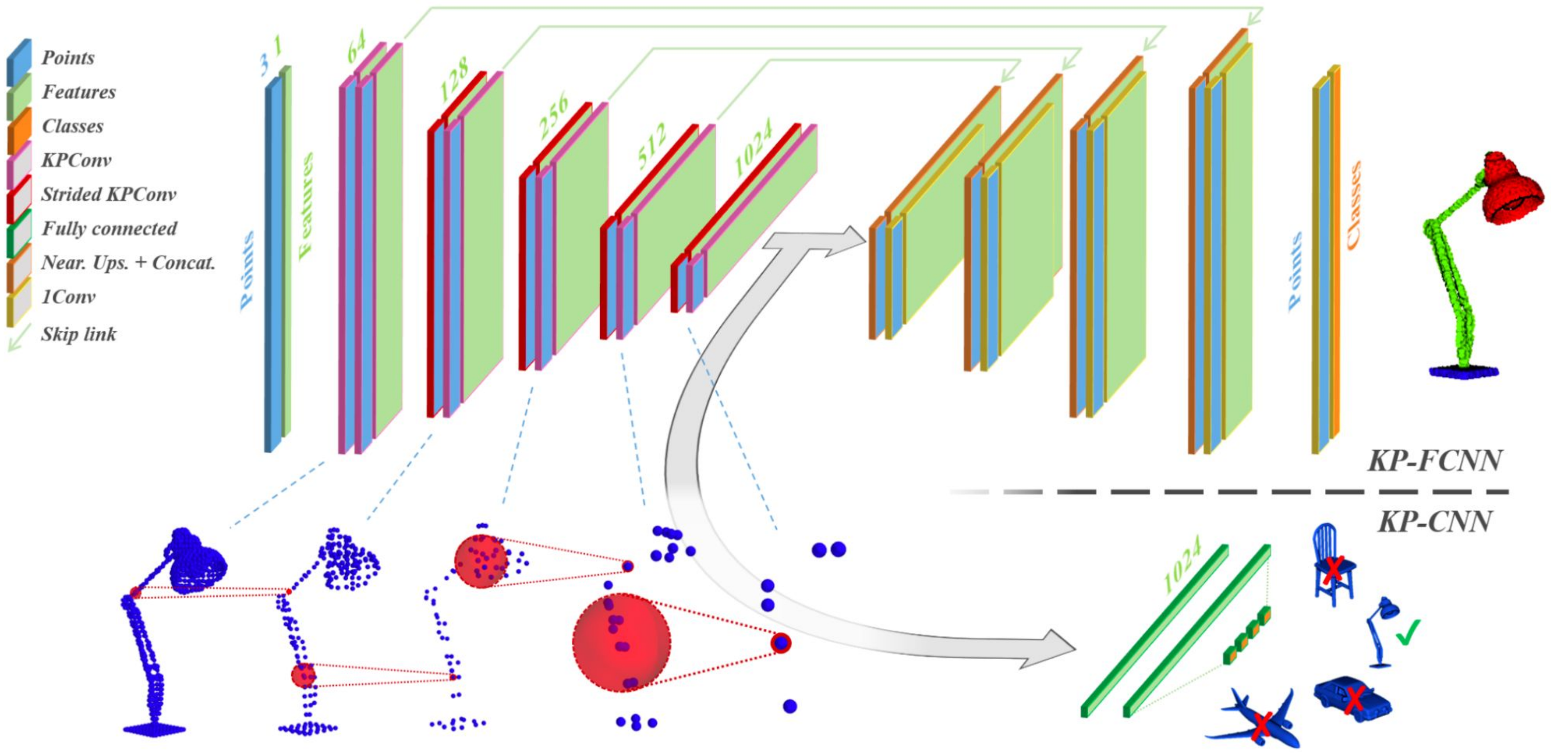


- Increase capacity & adapt to local geometry at x, by introducing a set of K shifts Δ(x) to deform kernel points.
- Δ(x): output of a rigid KPConv at x.
- Learned with point weights (needs regularization to avoid degeneration)

$$(\mathcal{F} * g)(x) = \sum_{x_i \in \mathcal{N}_x} g_{deform}(x - x_i, \Delta(x)) f_i$$

$$g_{deform}(y_i, \Delta(x)) = \sum_{k<K} h\left(y_i, \widetilde{x}_k + \Delta_k(x)\right) W_k$$

Network Architecture



Evaluation

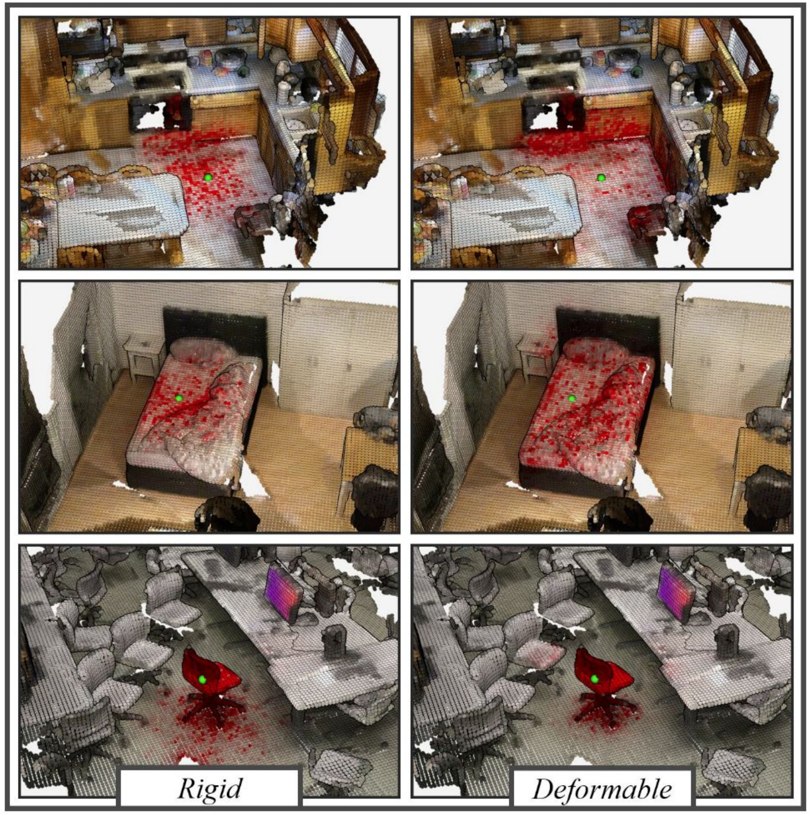
	ModelNet40	ShapeNetPart	
Methods	OA	mcIoU	mIoU
SPLATNet [34]	-	83.7	85.4
SGPN [42]	-	82.8	85.8
3DmFV-Net [9]	91.6	81.0	84.3
SynSpecCNN [48]	-	82.0	84.7
RSNet [15]	-	81.4	84.9
SpecGCN [40]	91.5	-	85.4
PointNet++ [27]	90.7	81.9	85.1
SO-Net [19]	90.9	81.0	84.9
PCNN by Ext [2]	92.3	81.8	85.1
SpiderCNN [45]	90.5	82.4	85.3
MCConv [13]	90.9	-	85.9
FlexConv [10]	90.2	84.7	85.0
PointCNN [20]	92.2	84.6	86.1
DGCNN [43]	92.2	85.0	84.7
SubSparseCNN [9]	-	83.3	86.0
KPConv *rigid*	**92.9**	85.0	86.2
KPConv *deform*	92.7	**85.1**	**86.4**

3D shape classification & segmentation

Methods	Scannet	Sem3D	S3DIS	PL3D
Pointnet [26]	-	-	41.1	-
Pointnet++ [27]	33.9	-	-	-
SnapNet [4]	-	59.1	-	-
SPLATNet [34]	39.3	-	-	-
SegCloud [37]	-	61.3	48.9	-
RF_MSSF [38]	-	62.7	49.8	56.3
Eff3DConv [50]	-	-	51.8	-
TangentConv [36]	43.8	-	52.6	-
MSDVN [30]	-	65.3	54.7	66.9
RSNet [15]	-	-	56.5	-
FCPN [28]	44.7	-	-	-
PointCNN [20]	45.8	-	57.3	-
PCNN [2]	49.8	-	-	-
SPGraph [17]	-	73.2	58.0	-
ParamConv [41]	-	-	58.3	-
SubSparseCNN [9]	**72.5**	-	-	-
KPConv *rigid*	68.6	**74.6**	65.4	72.3
KPConv *deform*	68.4	73.1	**67.1**	**75.9**

3D scene segmentation

Visualization of Effective Receptive Field (ERF)



- Rigid kernels have consistent ERFs.
- Deformable kernels **adapt to local geometry** - reach further on a flat surface, concentrate more on chairs instead of the ground, etc.

Summary

- Defines continuous spatial kernel by a point set: expressive & easy to learn.
- Further increases expressiveness by making kernel point positions learnable(deformable), which can adapt to local geometry.
- However, deformable versions are also harder to learn and may overfit on simpler tasks.

Point Convolutional Neural Networks by Extension Operators

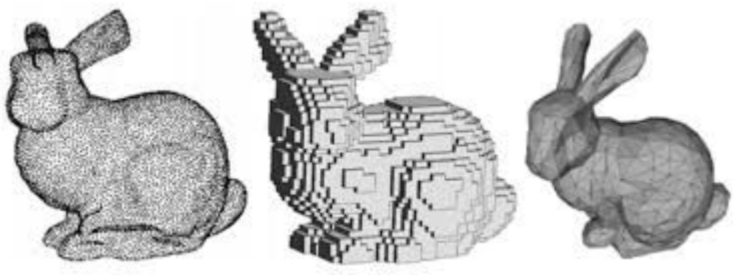
Matan Atzmon, Haggai Maron, Yaron Lipman

Main Problem

Pointcloud networks need to be:

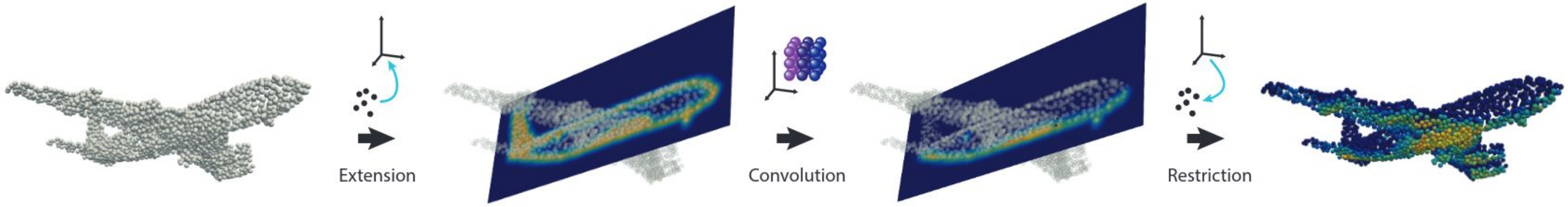
1. Invariant to order of points
2. Robust to sampling density and distribution of points
3. Translationally invariant.

Volumetric grid methods - as a direct extension of pixels - approximate underlying shape crudely



Their solution

Operate directly on pointcloud using a pair of operators: **extension** and **restriction**.

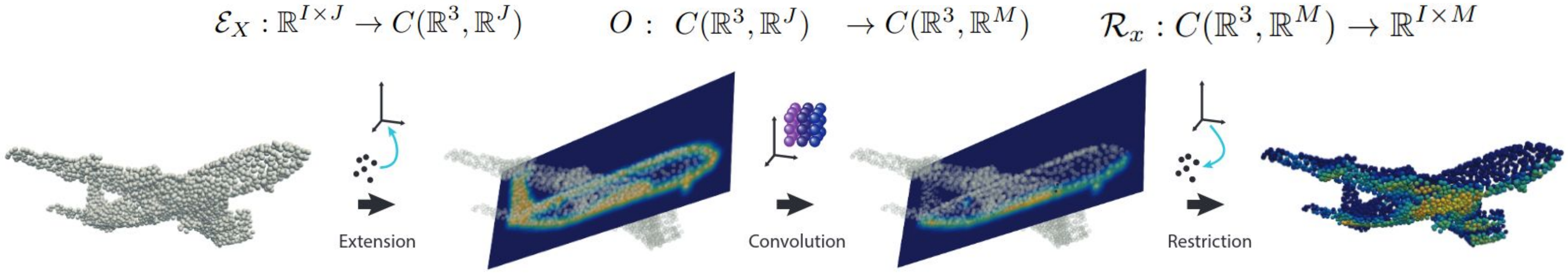


$$O_X = \mathcal{R}_X \circ O \circ \mathcal{E}_X.$$

Extension operator taken to be a radial basis function, which is invariant to point orders.

Restriction is sampling operator.

O is volumetric convolution, allows for translational invariance.


$\mathcal{E}_X : \mathbb{R}^{I\times J} \to C(\mathbb{R}^3, \mathbb{R}^J)$
$O : C(\mathbb{R}^3, \mathbb{R}^J) \to C(\mathbb{R}^3, \mathbb{R}^M)$
$\mathcal{R}_x : C(\mathbb{R}^3, \mathbb{R}^M) \to \mathbb{R}^{I\times M}$
Extension
Convolution
Restriction

Extension - letting the values "extend" from the surface

$$\mathcal{E}_X[f](x) = \sum_i f_{ij} \ell_i(x),$$

$$\ell_i \in C(\mathbb{R}^3, \mathbb{R}) \quad f \in \mathbb{R}^{I \times J}$$

$$\ell_i(x) = c\omega_i \Phi(|x - x_i|),$$

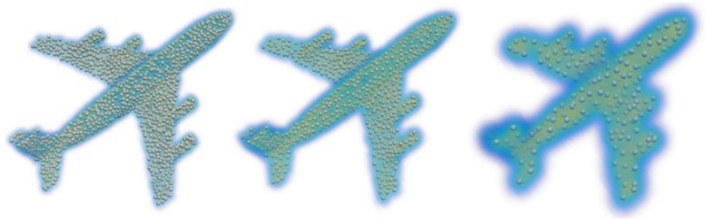


Figure 2: Applying the extension operator to the constant **1** function over three airplane point clouds in different sampling densities: 2048, 1024 and 256 points. Note how the extended functions resemble the airplane indicator function, and hence similar to each other.

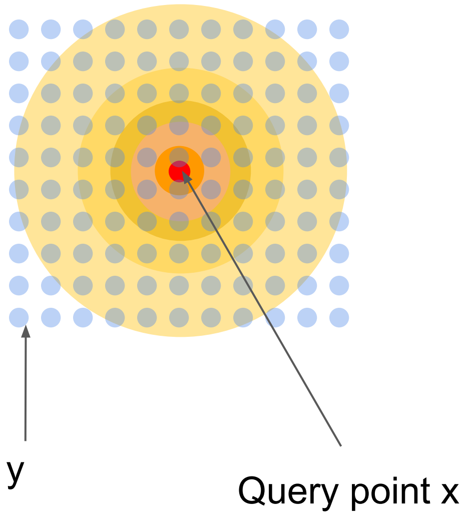
Kernel model - volumetric convolutions

We consider a continuous convolution operator $O : C(\mathbb{R}^3, \mathbb{R}^J) \to C(\mathbb{R}^3, \mathbb{R}^M)$ applied to vector valued function $\psi \in C(\mathbb{R}^3, \mathbb{R}^J)$,

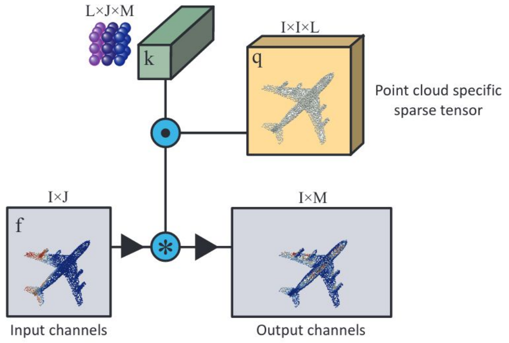
$$O[\psi](x) = \psi * \kappa\,(x) = \int_{\mathbb{R}^3} \sum_j \psi_j(y)\, \kappa_{jm}(x-y)\, dy, \quad (7)$$

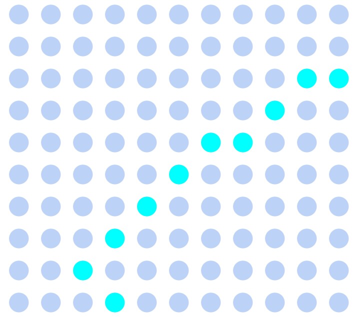
where $\kappa \in C(\mathbb{R}^3, \mathbb{R}^{J\times M})$ is the convolution kernel that is also represented in the same RBF basis:

$$\kappa_{jm}(z) = \sum_l k_{ljm} \Phi(|z - y_l|), \quad (8)$$



Restriction - sampling from this function over 3-space on the surface of the object.

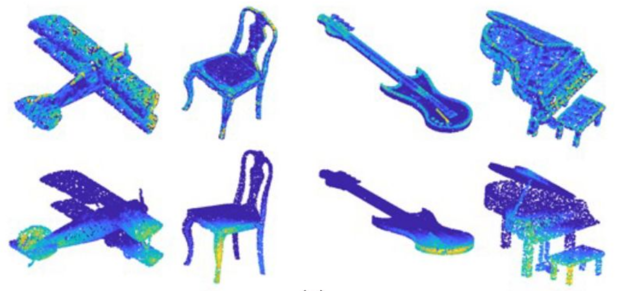




RBF choice - the Gaussian RBF

$$\Phi(|x - x_i|), \Phi_\sigma(r) = \exp\left(-\frac{r^2}{2\sigma^2}\right)$$

Strengths:
- Has desired approximation properties
- Has efficient close-formed solution



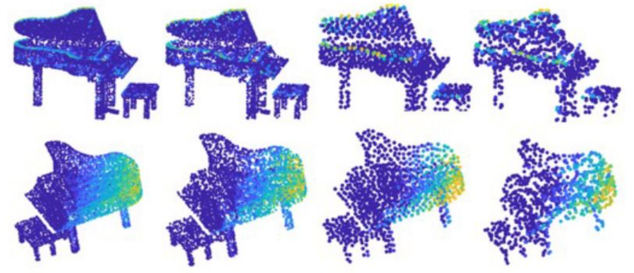


Figure 5: Our point cloud convolution is translation invariant and robust to sample size and density: (a) shows feature activations of two kernels (rows) learned by our network's first convolution layer on different shapes (columns). The features seems consistent across the different models; (b) shows another pair of kernels (rows) on a single model with varying sampling density (from left to right): $10K$ points, $5K$ points (random sampling), 1K points (farthest point sampling) and $1K$ (random sampling). Note that the convolution captures the same geometric properties on all models regardless of the sampling.

algorithm	# points	10 models	40 models
Point cloud methods			
pointnet [31]	1024	-	89.2
pointnet++ [33]	1024	-	90.7
deep sets [46]	1000	-	87.1
ECC [37]	1000	90.8	87.4
kd-network [21]	1024	93.3	90.6
kd-network [21]	32k	94.0	91.8
ours	1024	**94.9**	**92.3**
Additional input features			
FusionNet (uses mesh structure) [17]	-	93.1	90.8
VRN single(uses mesh structure) [6]	-	-	92.0
OctNet [35]	-	90.9	86.5
VRN ensemble(uses mesh structure) [6]	-	97.1	95.5
MVCNN (uses mesh structure)[32]	-	-	92.0
MVCNN-MultiRes(uses mesh structure)[32]	-	-	93.8
pointnet++ (uses normals) [33]	5K	-	91.9
OCNN (uses normals) [40]	-	-	90.6

Table 1: Shape classification results on the ModelNet40 and ModelNet10 datasets.

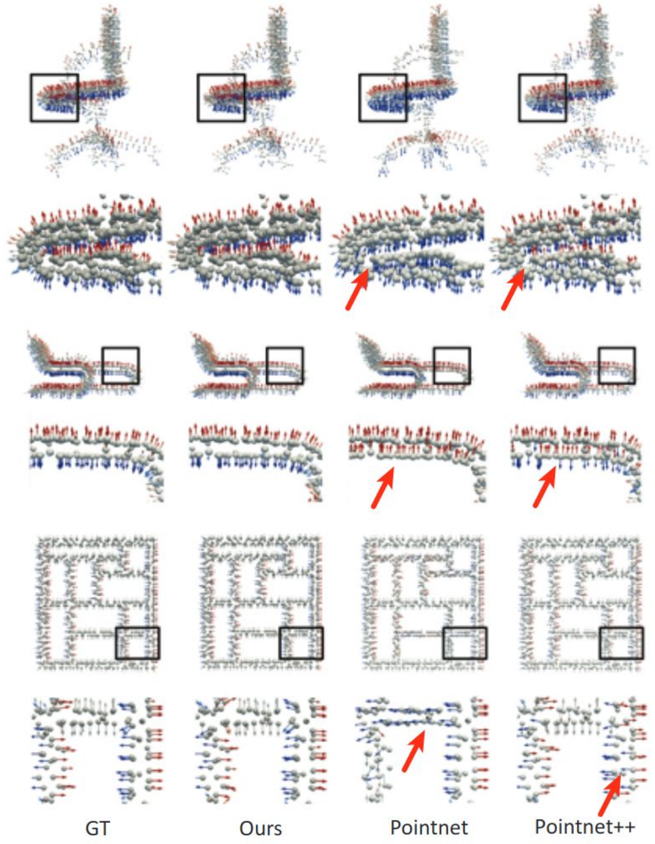


Figure 8: Normal estimation in ModelNet40. We show normal estimation of four models (rows) with blow-ups. Normals are colored by one of their coordinates for better visualization. Note that competing methods sometimes fail to recognize the outward normal direction (examples indicated by red arrows).

Data	algorithm	# points	error
ModelNet40	PointNet	1024	0.47
	PointNet++	1024	0.29
	ours	**1024**	**0.19**

Table 4: Normal estimation in ModelNet40.

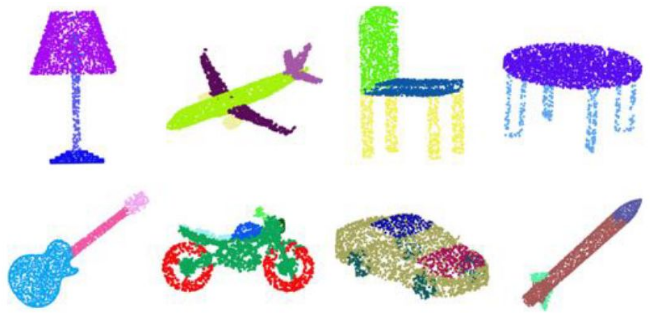


Figure 7: Results of PCNN on the part segmentation benchmark from [44]

Pros

Allows finer-grained convolutions than prior volumetric approaches.

Invariant to point cloud order

Robust to point sampling and density

Translational invariance

Cons

Compared to ImageCNN’s, extra computational burden.

An End-to-End Transformer Model for 3D Object Detection

Ishan Misra, Rohit Girdhar, and Armand Joulin

ICCV 2021

Motivating Question

Processing point cloud data with convolution operations is a well-studied problem

Transformer-based architectures have recently shown great success in NLP and images [1,2]

Can we use a transformer-based architectures for point cloud reasoning?

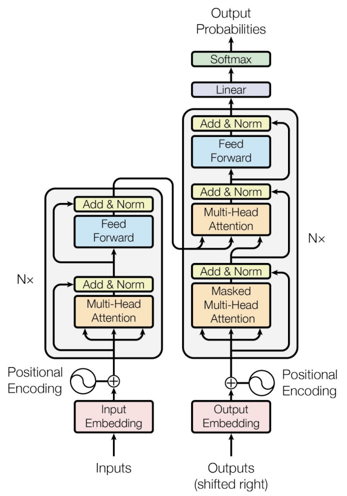
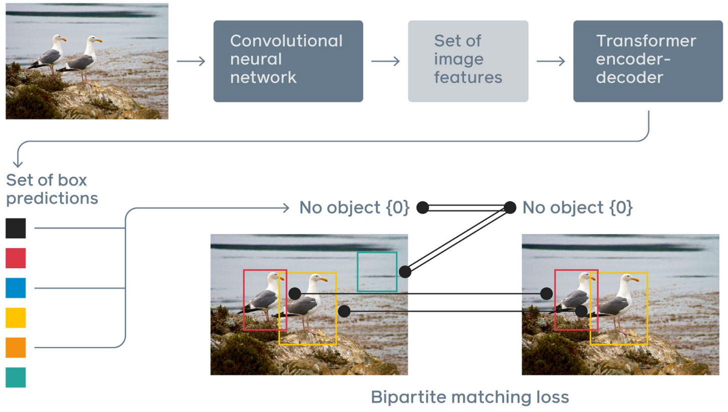


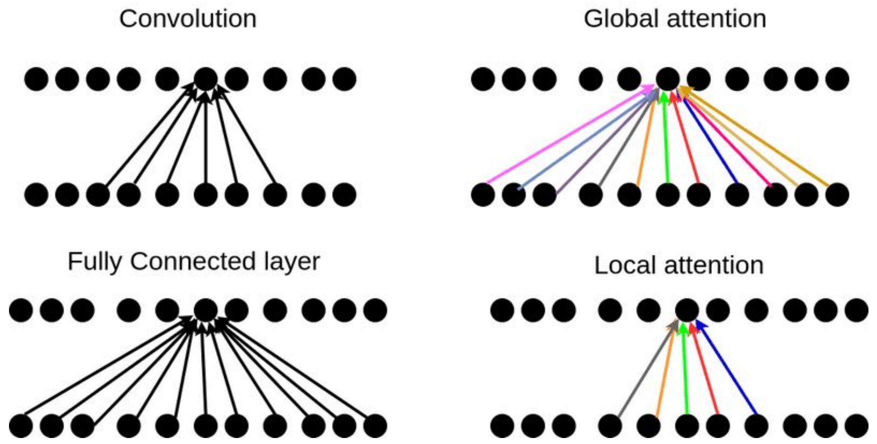
Figure 1: The Transformer - model architecture.



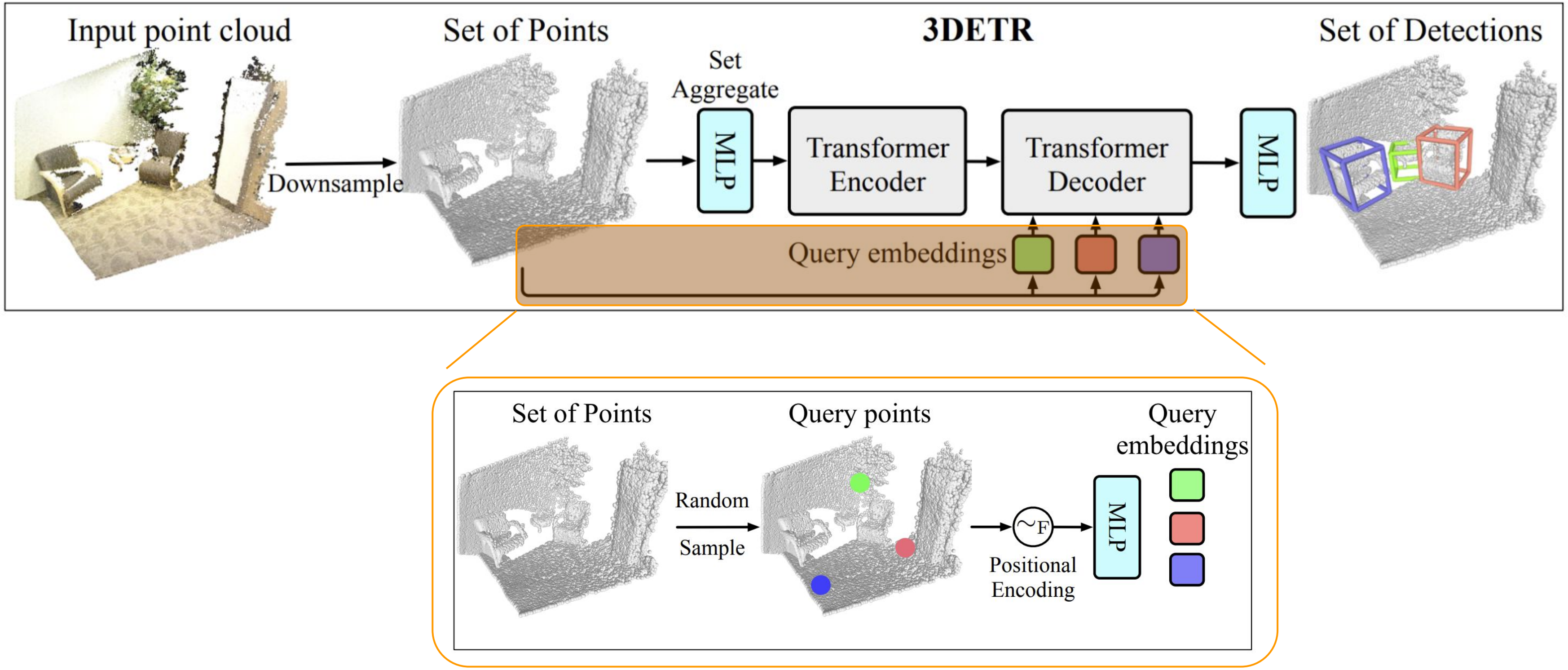
[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. arXiv preprint arXiv:1706.03762, 2017.
[2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-toend object detection with transformers. In European Conference on Computer Vision, pages 213–229. Springer, 2020

Why Transformers for 3D?

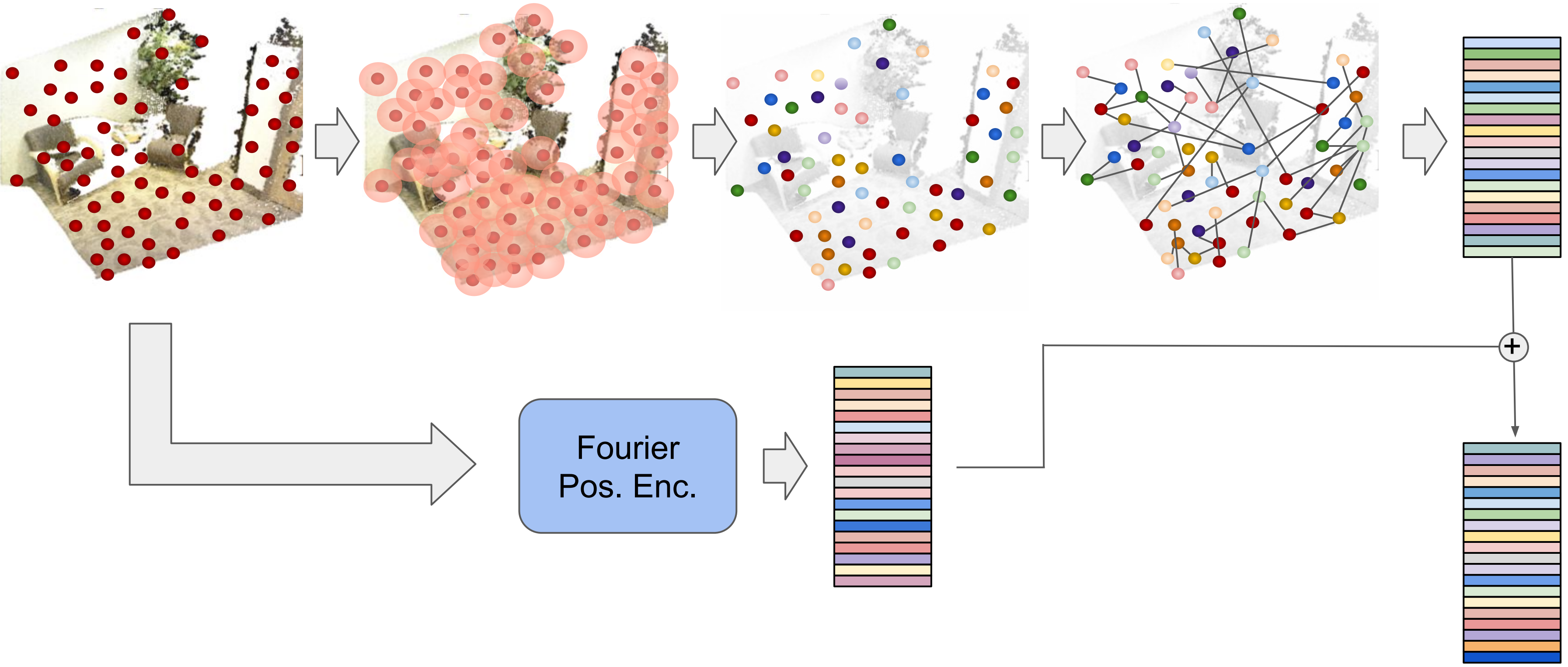
1. Transformers permit better processing of global-context
2. Transformers typically have greater expressive capabilities
3. **Transformers are permutation invariant**



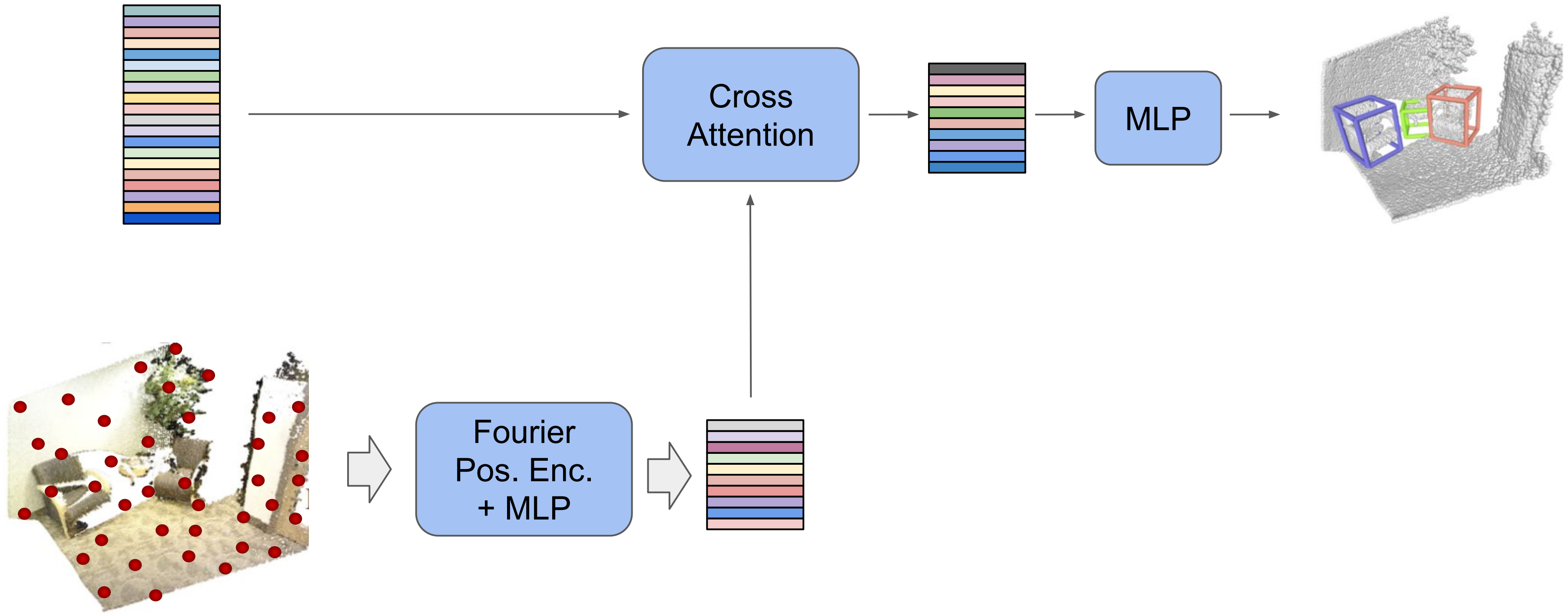
3DETR Overview



3DETR Encoding Scheme

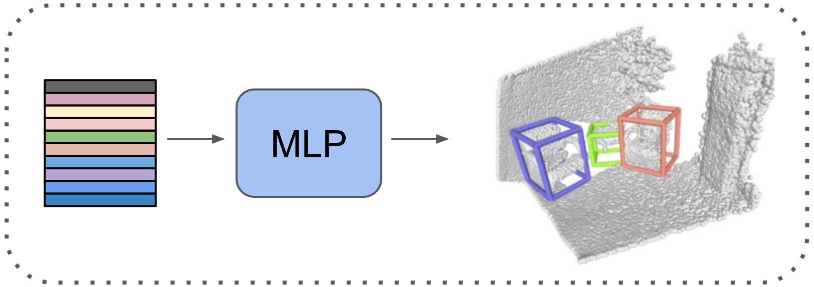


3DETR Decoding Scheme

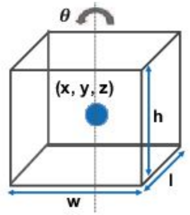


Fixed Set of Box Predictions

- Predict fixed set of bounding boxes based on randomly sampled queries



- Boxes parameterized by **center, size, orientation,** and **semantic class**
  - A background semantic class is included



- Predictions assigned to GT via **bipartite-matching**

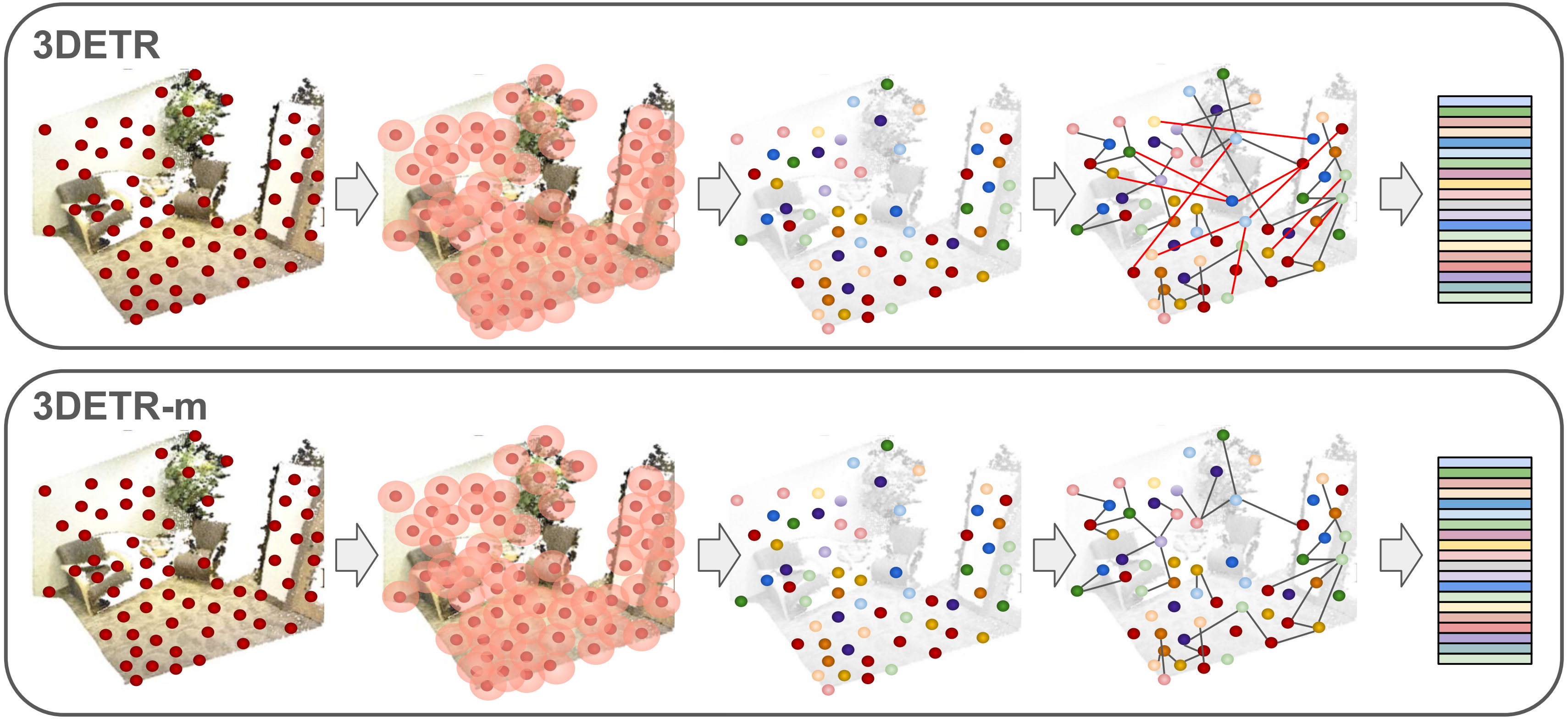
$$C_{\text{match}}(\hat{\mathbf{b}}, \mathbf{b}) = \underbrace{-\lambda_1 \, \text{GIoU}(\hat{\mathbf{b}}, \mathbf{b}) + \lambda_2 \|\hat{\mathbf{c}} - \mathbf{c}\|_1}_{\text{geometric}} \underbrace{- \lambda_3 \hat{\mathbf{S}}[s_{\text{gt}}] + \lambda_4 (1 - \hat{\mathbf{s}}[s_{\text{bg}}])}_{\text{semantic}}$$

Losses: straightforward

$$\mathcal{L}_{\text{3DETR}} = \underbrace{\lambda_c \|\hat{\mathbf{c}} - \mathbf{c}\|_1}_{\text{L1 center loss}} + \underbrace{\lambda_d \|\hat{\mathbf{d}} - \mathbf{d}\|_1}_{\text{L1 size loss}} + \underbrace{\lambda_{ar} \|\hat{\mathbf{a}}_r - \mathbf{a}_r\|_{\text{huber}}}_{\text{Huber yaw residual loss}} - \underbrace{\lambda_{ac} \mathbf{a}_c^\top \log \hat{\mathbf{a}}_c}_{\text{Cross-entropy yaw binning loss}} - \underbrace{\lambda_s \mathbf{s}_c^\top \log \hat{\mathbf{s}}_c}_{\text{Cross-entropy semantic class loss}}$$

3DETR-m

- Mask encoding **self-attention** based on **spatial locality**

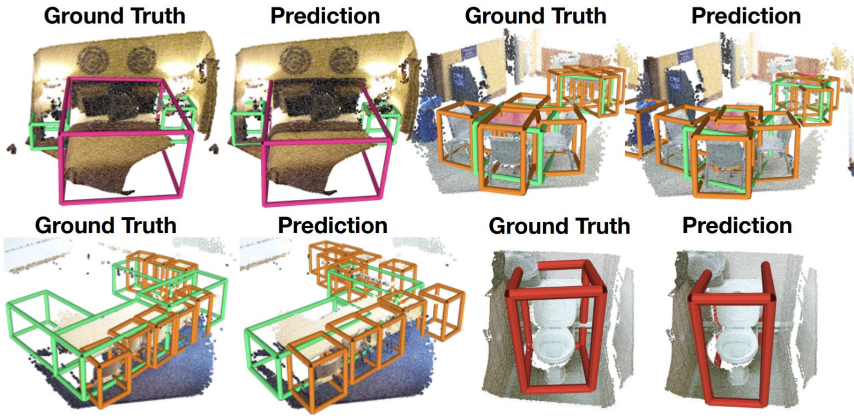


Results: Object Detection

Takeaways:

- 3DETR outperforms BoxNet
- 3DETR-m outperforms BoxNet and VoteNet
- Geometric improvements outlined in SOTA H3DNet can also be applied to 3DETR

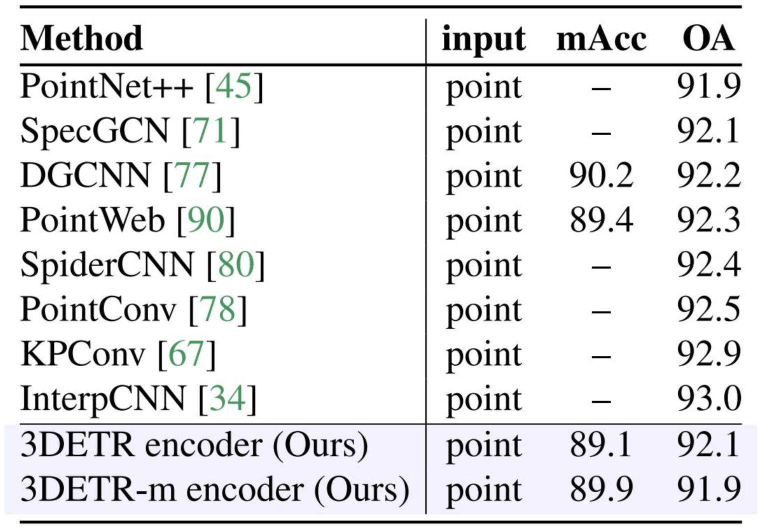
Method	ScanNetV2		SUN RGB-D	
	AP_{25}	AP_{50}	AP_{25}	AP_{50}
BoxNet† [42]	49.0	21.1	52.4	25.1
3DETR	62.7	37.5	58.0	30.3
VoteNet† [42]	60.4	37.5	58.3	33.4
3DETR-m	65.0	47.0	59.1	32.7
H3DNet [89]	67.2	48.1	60.1	39.0



Results: Shape Classification via 3DETR Encoder

Takeaway:

- The 3DETR **encoder performs competitively** to existing 3D feature-extraction methods

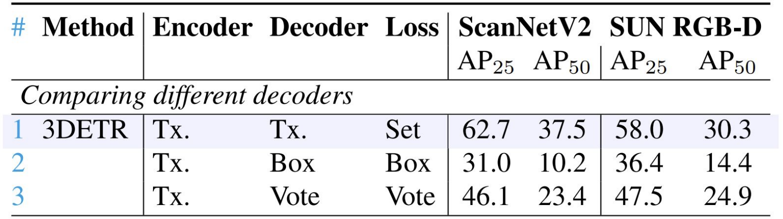


Method	input	mAcc	OA
PointNet++ [45]	point	–	91.9
SpecGCN [71]	point	–	92.1
DGCNN [77]	point	90.2	92.2
PointWeb [90]	point	89.4	92.3
SpiderCNN [80]	point	–	92.4
PointConv [78]	point	–	92.5
KPConv [67]	point	–	92.9
InterpCNN [34]	point	–	93.0
3DETR encoder (Ours)	point	89.1	92.1
3DETR-m encoder (Ours)	point	89.9	91.9

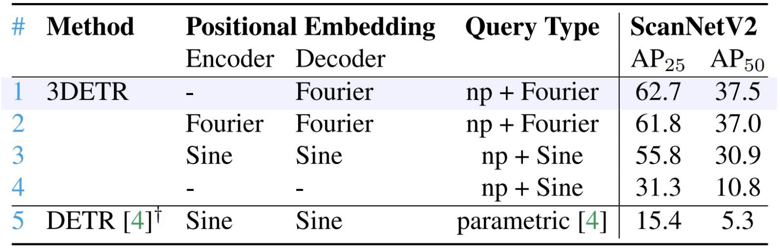
Results: Ablation Studies

Takeaways:

- BoxNet or VoteNet decoding heads greatly reduce performance
- Using a positional encoding in the encoder degrades performance
- Fourier positional encodings are better than Sine positional encodings
- Using learned decoding queries completely fails

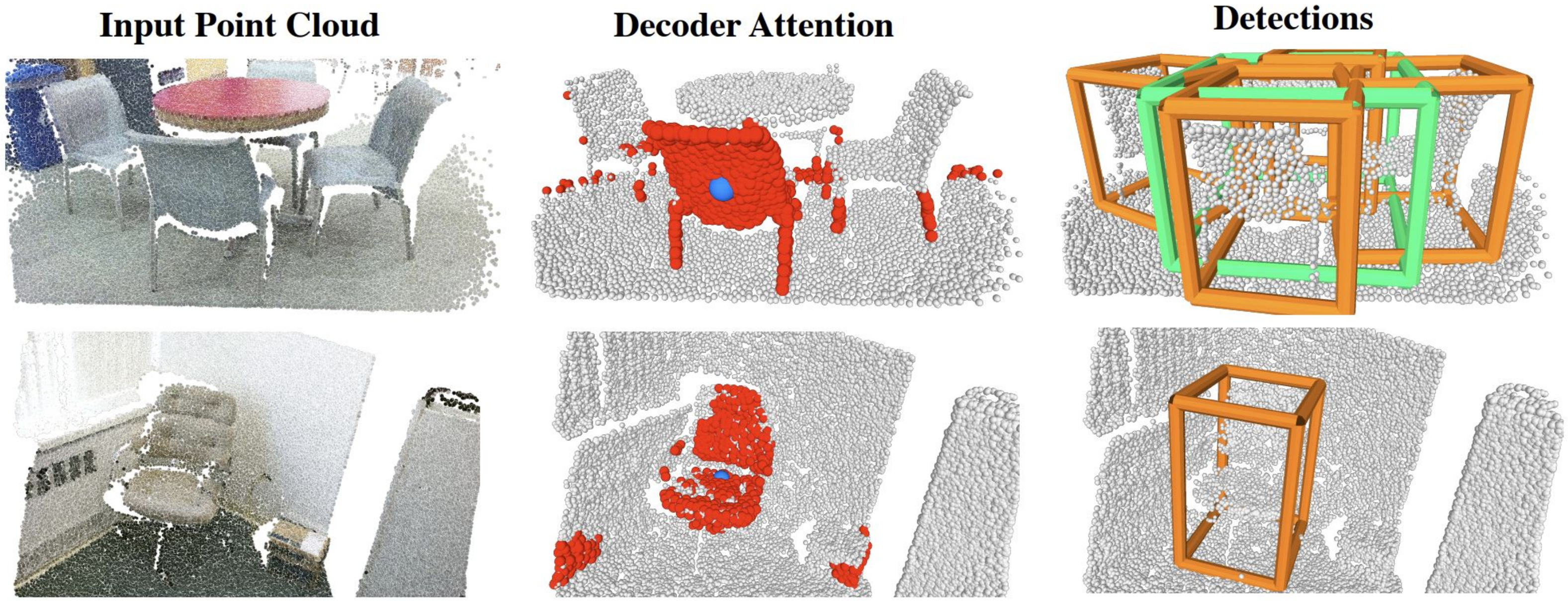


#	Method	Encoder	Decoder	Loss	ScanNetV2		SUN RGB-D	
					AP_{25}	AP_{50}	AP_{25}	AP_{50}
Comparing different decoders								
1	3DETR	Tx.	Tx.	Set	62.7	37.5	58.0	30.3
2		Tx.	Box	Box	31.0	10.2	36.4	14.4
3		Tx.	Vote	Vote	46.1	23.4	47.5	24.9



#	Method	Positional Embedding		Query Type	ScanNetV2	
		Encoder	Decoder		AP_{25}	AP_{50}
1	3DETR	-	Fourier	np + Fourier	62.7	37.5
2		Fourier	Fourier	np + Fourier	61.8	37.0
3		Sine	Sine	np + Sine	55.8	30.9
4		-	-	np + Sine	31.3	10.8
5	DETR [4]†	Sine	Sine	parametric [4]	15.4	5.3

Results: Example Decoder Attention



Paper Takeaways

Strengths:

- Great early paper in still-developing field of 3D transformers

Improvements / Next Steps:

- They show that adding a positional embedding to the encoder degrades performance...
- Their method cannot produce per-point features (necessary for segmentation, canonicalization, and more)

Implicit Autoencoder for Point Cloud Self-supervised Representation Learning

Siming Yan, Zhenpei Yang, Haoxiang Li, Li Guan, Hao Kang, Gang Hua, Qixing Huang

Their solution

In broad strokes: decoder outputs a CONTINUOUS representation shared among different point cloud samplings of the same model.

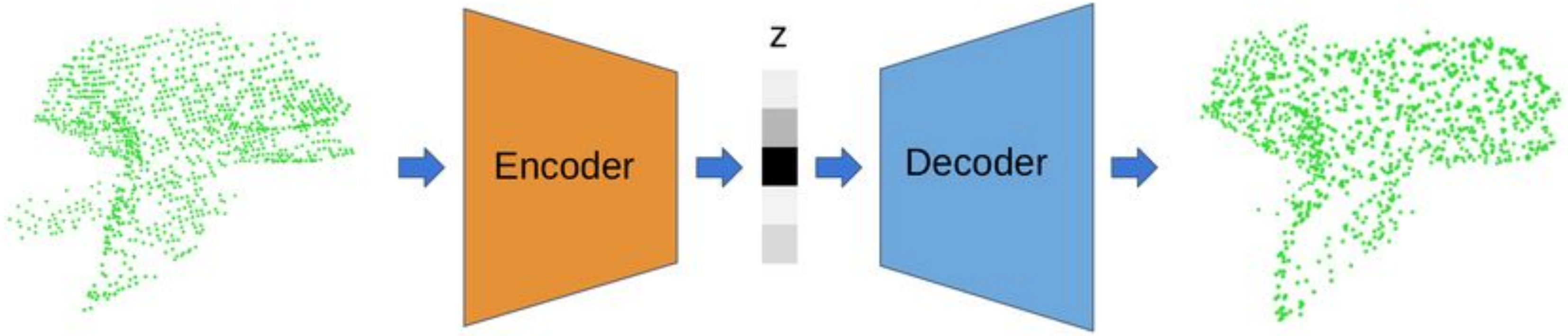
Two strengths:

- Discards sampling variations in the output of decoder
- Minimizing discrepancy between two implicit functions DOES NOT require computing correspondence (e.g. using chamfer distance). Faster.

Main problem

Autoencoders - traditionally input = output = point cloud

But point clouds have sampling variations, which does not capture information useful for 3D understanding. The claim: a point-based AE forces the encoder to remember useless variations.



Explicit AE

$$\min_{\Theta,\Phi} d_{\mathrm{exp}}((g_{\Phi} \circ f_{\Theta})(\mathcal{P}^{\mathrm{in}}), \mathcal{P}^{\mathrm{gt}}_{\mathrm{sub}})$$

Example of distance function can be the chamfer distance, which gives a distance between two point-clouds.

Implicit AE

$$\min_{\Theta,\Phi} d_{\mathrm{imp}}((g_{\Phi} \circ f_{\Theta})(x|\mathcal{P}^{\mathrm{in}}), g_0(x))$$

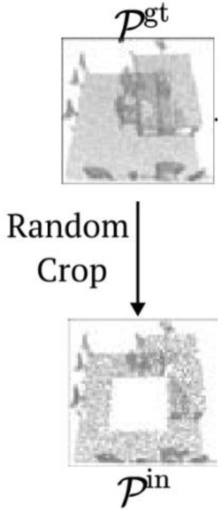
Groundtruth implicit function obtained as SDF, occupancy grid …etc. Choice of distance function is coupled with the type of implicit representation used.

Loss Function

$$\mathcal{L} = \frac{1}{N}\sum_{i=0}^{N} \|(g_{\Phi} \circ f_{\Theta})(x_i|\mathcal{P}^{\text{in}}) - g_0(x_i)\|$$

Random crops

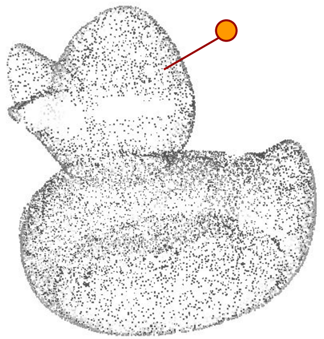
To capture high-level semantic features, random crop part of the input point cloud, then reconstruct missing parts. They show resulting point cloud can perform better on downstream tasks



How do you obtain ground truth implicits?

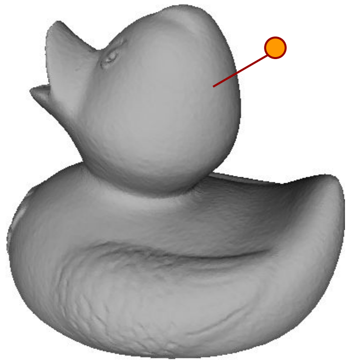
Real data:

- Compute closest distance between query point and groundtruth points
- Use unsigned distance function



Synthetic data:

- Signed distance function obtained from underlying water-tight meshes.



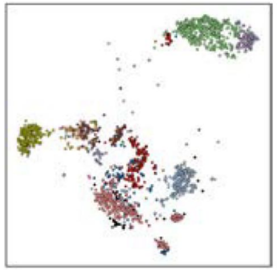
Shape classification

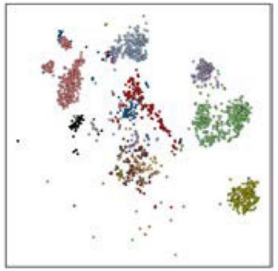
Method	ModelNet40
3D-GAN [49]	83.3%
Latent-GAN [1]	85.7%
SO-Net [21]	87.3%
MAP-VAE [16]	88.4%
Jigsaw* [42]	84.1%
FoldingNet* [54]	90.1%
Orientation* [36]	90.7%
STRL* [20]	90.9%
OcCo* [47]	89.7%
IAE(ours)	**92.1%**

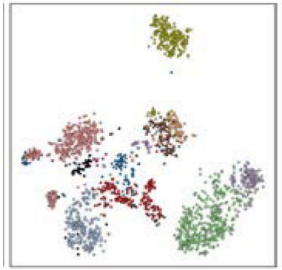
Table 1: **Linear evaluation for shape classification on ModelNet40.** Note that to make a fair comparison, different * methods use the same DGCNN encoder backbone.

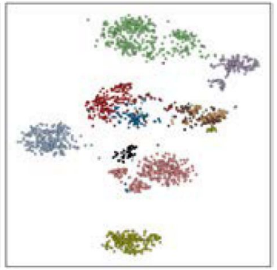
Category	Method	ModelNet40
Supervised	PointNet [38]	89.2%
	PointNet++ [39]	90.7%
	PointCNN [22]	92.2%
	KPConv [44]	92.9%
	DGCNN [48]	92.9%
	PointTransform [59]	93.7%
Self-Supervised	FoldingNet [54]	93.1%
	STRL [20]	93.1%
	OcCo [47]	93.0%
	IAE(ours)	**93.7%**

Table 2: **Shape classification fine-tuned results on ModelNet40.** Supervised learning methods train the model from scratch. Self-supervised methods use the pre-trained models as the initial weight for supervised fine-tuning. All the self-supervised methods shown here use the same DGCNN encoder backbone.


(a) Random


(b) FoldingNet


(c) OcCo


(d) IAE(ours)

Object Detection in 3D Scenes

Method	ScanNet		SUN RGB-D	
	AP_{50}	AP_{25}	AP_{50}	AP_{25}
VoteNet [37]	33.5	58.6	32.9	57.7
STRL [20]	38.4	59.5	35.0	58.2
RandomRooms [40]	36.2	61.3	35.4	59.2
PointContrast [53]	38.0	59.2	34.8	57.5
DepthContrast[2] [58]	39.1	**62.1**	35.4	**60.4**
IAE (Ours)	**39.8**	61.5	**36.0**	**60.4**

Table 3: **3D object detection results.** We fine-tuned our pre-trained model on ScanNetV2 and SUN-RGBD validation set using a popular detection framework, VoteNet [37]. We show mean of average precision(mAP) across all semantic classes with 3D IoU threshold 0.25 and 0.5. Our method outperforms prior work across most metrics.

Main takeaway: transferable! Unlike other methods for self-supervision in the past in this task setting.

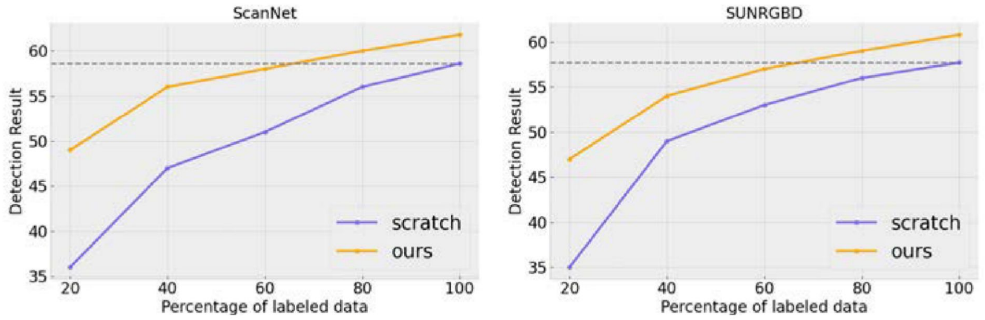
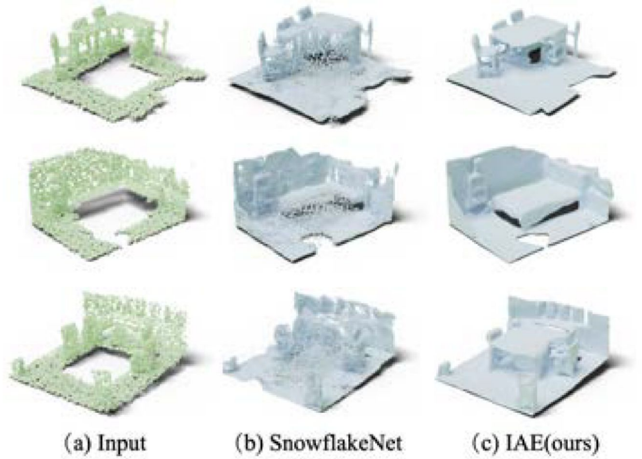


Fig. 4: **Label efficiency training.** We pre-train our model on ScanNet and then fine-tune on ScanNet and SUN RGB-D separately. During fine-tuning, different percentages of labeled data are used. Our pre-training model outperforms training from scratch and achieves nearly the same result with only 60% labeled data.

Decoder	Method	ModelNet40
Explicit	FoldingNet [54]	90.1%
	OcCo [47]	89.7%
	SnowflakeNet [52]	89.9%
Implicit	OccNet [26]	91.5%
	Conv-OccNet [35]	**92.1%**

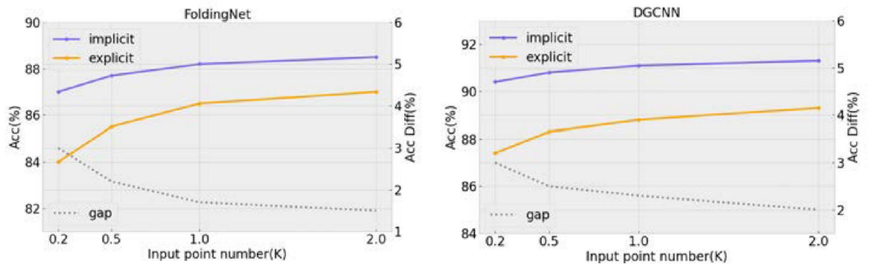
Decoder	Functions	ModelNet40
Explicit	Point Cloud	90.1%
Implicit	Occ Value	91.3%
	UDF	91.7%
	SDF	**92.1%**

Table 6: Left: **Ablation study on different decoder model.** On ModelNet40, we show linear evaluation results. Our implicit auto-encoder formulations can be improved upon explicit counterpart under various decoder models. Right: **Ablation study on implicit function.** For explicit representation, we use FoldingNet as the decoder. For implicit representation, we experimented with Occupancy Value(Occ Value), Unsigned Distance Function(UDF), and Signed Distance Function(SDF) and find consistent improvement over explicit representation.



Pros

More robust to point cloud resolutions



No need to compute difference between two sets, saves on compute time.

Allows for better self-supervision for a range of downstream tasks.

Cons

Need groundtruth implicit from point cloud, which is an additional preprocessing step.

Additionally, sampling has to be done across the VOLUME, instead of the SURFACE.