

# A LINEAR TIME AND SPACE LOCAL GEOMETRY ENCODER VIA VECTORIZED KERNEL MIXTURE (VECKM)

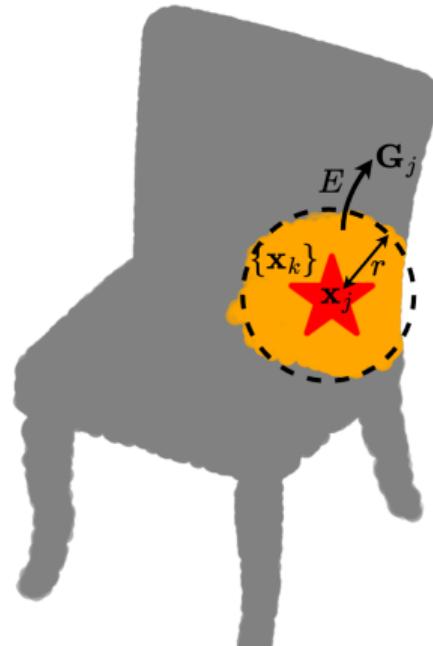
Dehao Yuan

Department of Computer Science  
University of Maryland, College Park

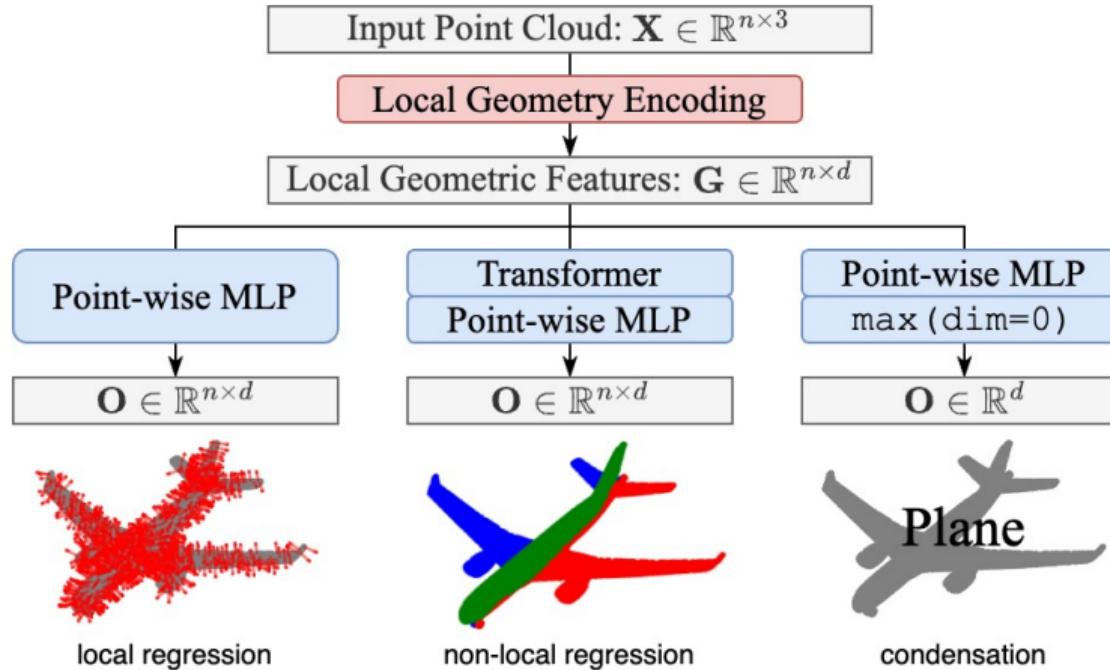


# LOCAL GEOMETRY ENCODING

- Input: point cloud  $\mathbf{X}_{n \times 3}$ .
- For each point  $\mathbf{x}_j$ , get the centered neighborhood  $\mathfrak{N}(\mathbf{x}_j) := \{\mathbf{x}_k - \mathbf{x}_j : \|\mathbf{x}_k - \mathbf{x}_j\| < r\}$ .
- Output:  $\mathbf{G}_{n \times d}$ , where  $\mathbf{G}_j = E(\mathfrak{N}(\mathbf{x}_j))$ .
- WANT: the encoder  $E$  that receives a local point cloud and produces a vector.
- Implicit Assumption: Translation Invariance.

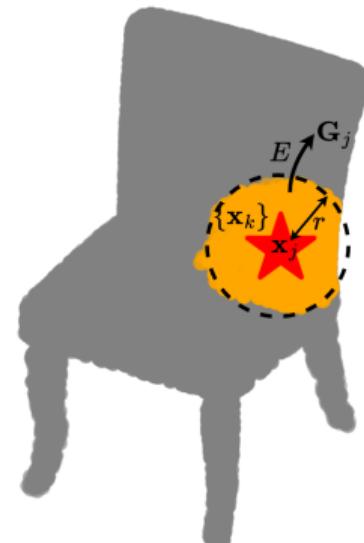
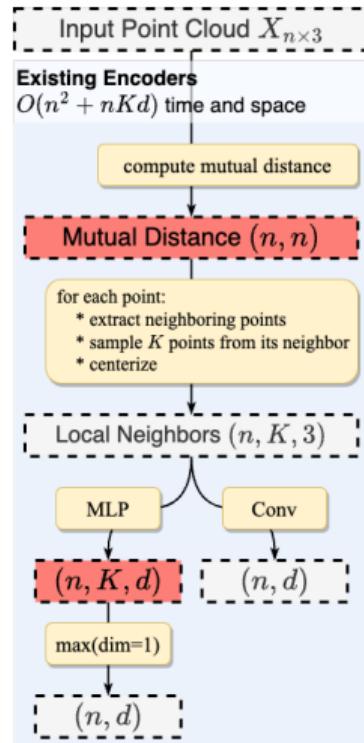


# AFTER LOCAL GEOMETRY ENCODING...



# EXISTING METHODS

- Group and downsample.
- Memory Bottleneck:  
mutual distance matrix.
- Computation Bottleneck:  
require  $nK$  MLPs.

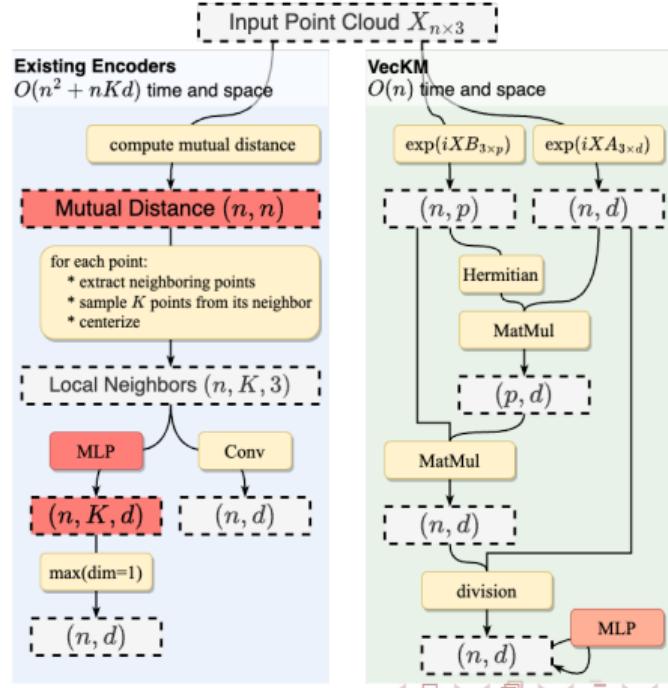


# OUR SOLUTION: VECKM

A local geometry encoding that is

- efficient to compute.
- descriptive;
- robust to noise;
- simple to implement.

## Computational Graphs: Others v.s. VecKM

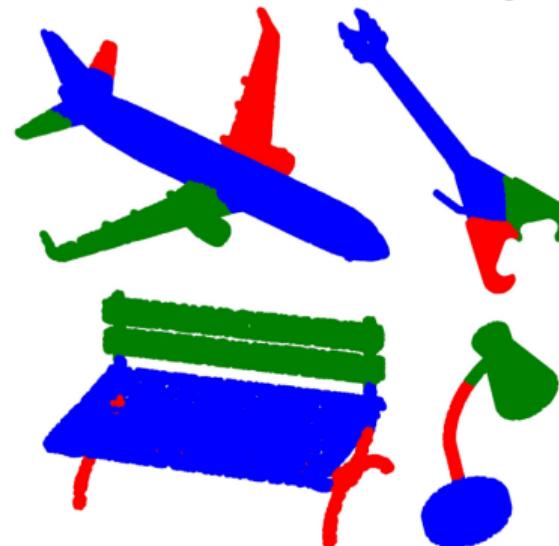


# OUR SOLUTION: VECKM

A local geometry encoding that is

- efficient to compute.
- **descriptive;**
- robust to noise;
- simple to implement.

Cluster of Raw VecKM Encodings

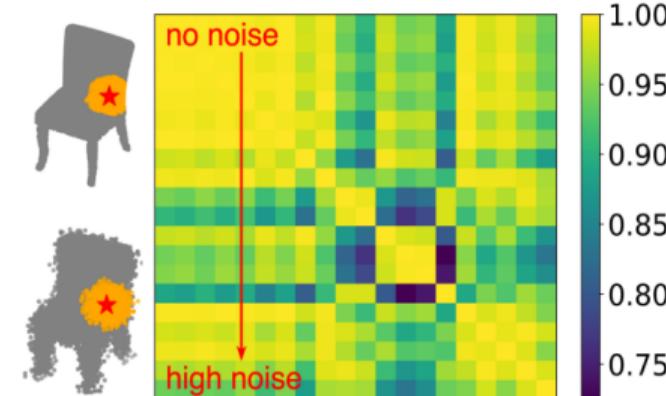


# OUR SOLUTION: VECKM

A local geometry encoding that is

- efficient to compute.
- descriptive;
- **robust to noise;**
- simple to implement.

## Similarity of VecKM Encodings under Difference Noise Levels



# OUR SOLUTION: VECKM

A local geometry encoding that is

- efficient to compute.
- descriptive;
- robust to noise;
- **simple to implement.**

Initialization:  $\mathbf{A}_{3 \times d} \sim \mathcal{N}(0, \alpha^2)$ ,  $\mathbf{B}_{3 \times p} \sim \mathcal{N}(0, \beta^2)$ .

Input:  $\mathbf{X}_{n \times 3}$ .

$$\mathcal{A}_{n \times d} = \exp(i\mathbf{X}_{n \times 3}\mathbf{A}_{3 \times d})$$

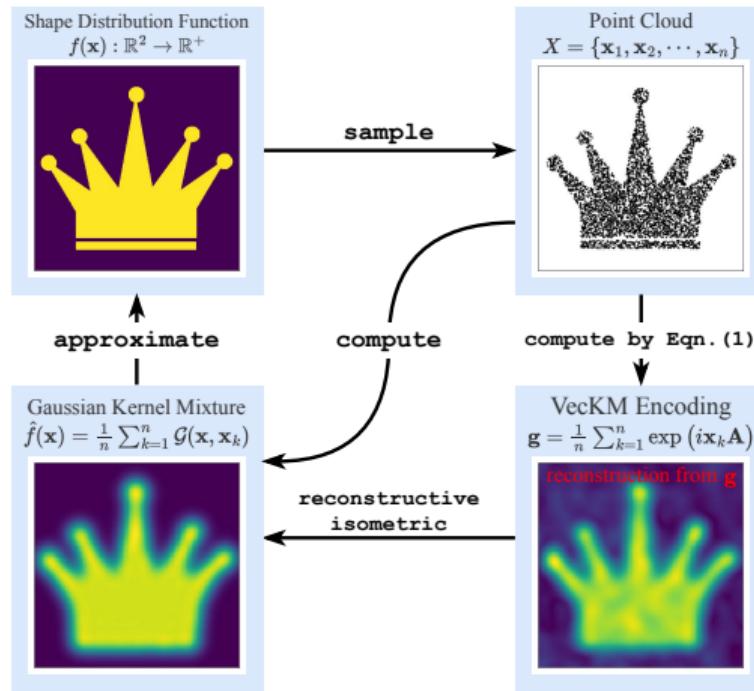
$$\mathcal{B}_{n \times p} = \exp(i\mathbf{X}_{n \times 3}\mathbf{B}_{3 \times p})$$

$$\mathbf{G}_{n \times d} = \text{normalize}((\mathcal{B} \times \mathcal{B}^H \times \mathcal{A}) ./ \mathcal{A})$$

Return:  $MLP(\mathbf{G}_{n \times d})$ .



## POINTWISE LOCAL GEOMETRY ENCODING – INTUITION



### Key Observation:

$$\mathbf{g} = \frac{1}{n} \sum_{k=1}^n \exp(i\mathbf{x}_k \mathbf{A})$$

“encodes” the density distribution of  $\{\mathbf{x}_k\}_{k=1}^n$ :



# CONNECTION WITH VECTOR FUNCTION ARCHITECTURE (VFA)

## THEOREM (VECTOR FUNCTION ARCHITECTURE (VFA))

*A function with the form  $f(x) = \sum_{k=1}^n \alpha_k \cdot K(x, x_k)$  can be represented by  $\sum_{k=1}^n \alpha_k \cdot \mathbf{z}(x_k)$  if  $K(x, y) \approx \langle \mathbf{z}(x), \mathbf{z}(y) \rangle$ .*

Here we set  $\mathbf{z}(x) := \exp(ix\mathbf{A})$ .



# POINTWISE LOCAL GEOMETRY ENCODING – DETAIL

## THEOREM (POINTWISE LOCAL GEOMETRY ENCODING)

Denote the neighbors of the point  $\mathbf{x}_0$  as  $\mathfrak{N}(\mathbf{x}_0) := \{\mathbf{x}_k - \mathbf{x}_0\}_{k=1}^n$ . The local geometry encoding of  $\mathbf{x}_0$  is computed as

$$\mathbf{g}_0 = \frac{1}{n} \sum_{k=1}^n \exp(i(\mathbf{x}_k - \mathbf{x}_0)\mathbf{A}_{3 \times d}) \quad (1)$$

where  $i$  is the imaginary unit and  $\mathbf{A} \in \mathbb{R}^{3 \times d}$  is a fixed random matrix where each element follows the normal distribution  $\mathcal{N}(0, \alpha^2)$ .

Detailed proof in the paper. We skip here.



# DENSE LOCAL GEOMETRY ENCODING

- Pointwise local geometry encoding:

$$\mathbf{g}_j = \frac{1}{n} \sum_{k=1}^n \exp(i(\mathbf{x}_k - \mathbf{x}_j)\mathbf{A}_{3 \times d}) \cdot \mathbb{I}(\|\mathbf{x}_k - \mathbf{x}_j\| < r)$$

- We can already compute the dense local geometry by grouping and computing one-by-one.
- However, no need!!! Because of **factorizability**:

$$\exp(i(\mathbf{x}_k - \mathbf{x}_j)\mathbf{A}_{3 \times d}) = \exp(i\mathbf{x}_k\mathbf{A}_{3 \times d}) ./ \exp(i\mathbf{x}_j\mathbf{A}_{3 \times d})$$

- By reusing computation, let  $\mathbf{J}$  be the (sparse) adjacency matrix:

$$\begin{aligned}\mathcal{A}_{n \times d} &= \exp(i\mathbf{X}_{n \times 3}\mathbf{A}_{3 \times d}) \\ \mathbf{G}_{n \times d} &= (\mathbf{J}_{n \times n}\mathcal{A}_{n \times d}). / \mathcal{A}_{n \times d}\end{aligned}$$



# DENSE LOCAL GEOMETRY ENCODING

- Pointwise local geometry encoding:

$$\mathbf{g}_j = \frac{1}{n} \sum_{k=1}^n \exp(i(\mathbf{x}_k - \mathbf{x}_j)\mathbf{A}_{3 \times d}) \cdot \mathbb{I}(\|\mathbf{x}_k - \mathbf{x}_j\| < r)$$

- We can already compute the dense local geometry by grouping and computing one-by-one.
- However, no need!!! Because of **factorizability**:

$$\exp(i(\mathbf{x}_k - \mathbf{x}_j)\mathbf{A}_{3 \times d}) = \exp(i\mathbf{x}_k\mathbf{A}_{3 \times d}) ./ \exp(i\mathbf{x}_j\mathbf{A}_{3 \times d})$$

- By reusing computation, let  $\mathbf{J}$  be the (sparse) adjacency matrix:

$$\begin{aligned}\mathcal{A}_{n \times d} &= \exp(i\mathbf{X}_{n \times 3}\mathbf{A}_{3 \times d}) \\ \mathbf{G}_{n \times d} &= (\mathbf{J}_{n \times n}\mathcal{A}_{n \times d}). / \mathcal{A}_{n \times d}\end{aligned}$$



# DENSE LOCAL GEOMETRY ENCODING – VERSION 1

$$\begin{aligned}\mathbf{J}[j, k] &= \mathbb{I}(\|\mathbf{x}_k - \mathbf{x}_j\| < r) \\ \mathcal{A}_{n \times d} &= \exp(i\mathbf{X}_{n \times 3}\mathbf{A}_{3 \times d}) \\ \mathbf{G}_{n \times d} &= (\mathbf{J}_{n \times n}\mathcal{A}_{n \times d})./\mathcal{A}_{n \times d}\end{aligned}$$



## DENSE LOCAL GEOMETRY ENCODING – LINEAR COMPLEXITY

- Recall Version 1:

$$\mathbf{J}[j, k] = \mathbb{I}(\|\mathbf{x}_k - \mathbf{x}_j\| < r)$$

$$\mathcal{A}_{n \times d} = \exp(i\mathbf{X}_{n \times 3}\mathbf{A}_{3 \times d})$$

$$\mathbf{G}_{n \times d} = (\mathbf{J}_{n \times n}\mathcal{A}_{n \times d}) ./ \mathcal{A}_{n \times d}$$

- Trick: Replace  $\mathbf{J}[j, k] = \exp(-\beta^2\|\mathbf{x}_k - \mathbf{x}_j\|^2)$ .
- Motivation:  $\mathbf{J} \approx [\mathcal{B}]_{n \times p}[\mathcal{B}^H]_{p \times n}$ .
- Then

$$\mathbf{G}_{n \times d} = ([\mathcal{B}]_{n \times p}[\mathcal{B}^H]_{p \times n}\mathcal{A}_{n \times d}) ./ \mathcal{A}_{n \times d} \quad (3)$$

Linear Time and Space!



# SUMMARY

Initialization:  $\mathbf{A}_{3 \times d} \sim \mathcal{N}(0, \alpha^2)$ ,  $\mathbf{B}_{3 \times p} \sim \mathcal{N}(0, \beta^2)$ .

Input:  $\mathbf{X}_{n \times 3}$ .

$$\mathcal{A}_{n \times d} = \exp(i\mathbf{X}_{n \times 3}\mathbf{A}_{3 \times d})$$

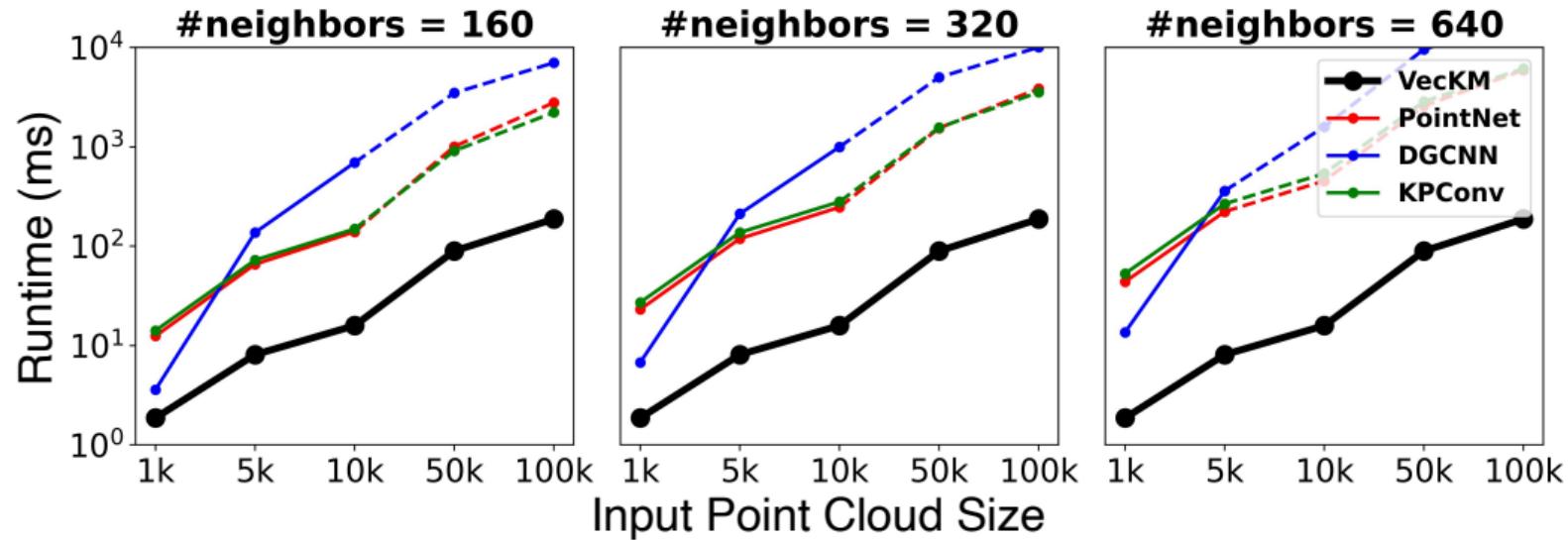
$$\mathcal{B}_{n \times p} = \exp(i\mathbf{X}_{n \times 3}\mathbf{B}_{3 \times p})$$

$$\mathbf{G}_{n \times d} = \text{normalize}((\mathcal{B} \times \mathcal{B}^H \times \mathcal{A}) ./ \mathcal{A})$$

Return:  $MLP(\mathbf{G}_{n \times d})$ .



# RUNTIME AND MEMORY COST



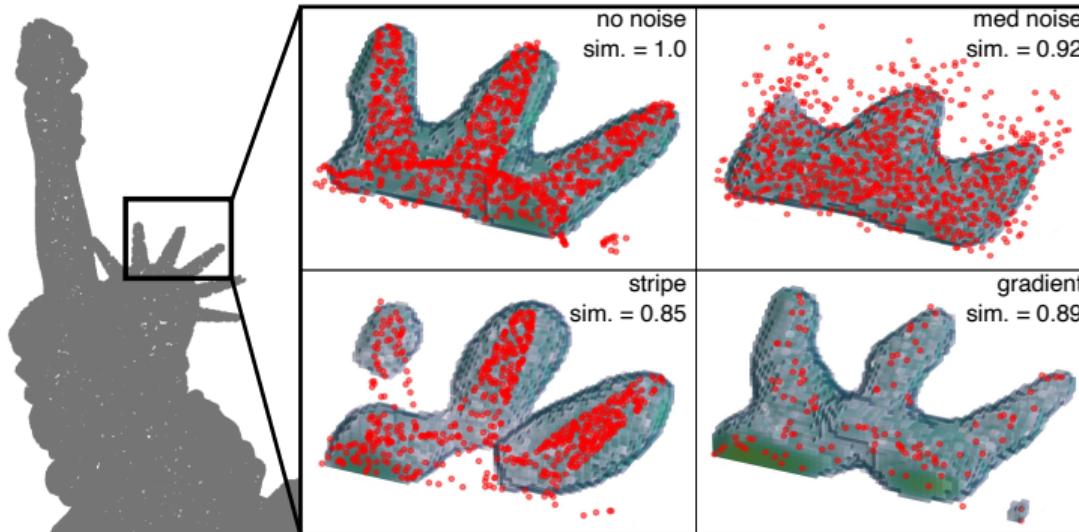
## NORMAL ESTIMATION ON PCPNET DATASET

**TABLE:** Normal estimation RMSE on the PCPNet dataset.

	Perturbations				Density Variation		Average
	None	Low	Med	High	Gradient	Stripe	
KPConv, #kp=16	22.68	23.09	25.21	29.05	34.40	25.61	26.67
KPConv, #kp=32	22.74	22.21	24.08	28.25	32.24	24.94	25.74
KPConv, #kp=64	22.09	22.12	23.90	28.45	28.60	24.05	24.86
DGCNN, #nbr=32	24.08	24.04	25.19	28.24	27.12	27.55	26.03
DGCNN, #nbr=64	23.21	25.34	25.66	26.01	28.86	28.20	26.21
DGCNN, #nbr=128	18.46	18.71	20.38	25.62	23.01	21.29	21.24
PointNet, #nbr=300	14.98	16.30	20.19	26.83	23.68	19.00	20.17
PointNet, #nbr=500	16.10	16.54	21.38	26.93	26.06	18.89	20.99
PointNet, #nbr=700	15.59	16.25	20.99	26.21	24.66	17.87	20.27
VecKM (Ours)	<b>13.59</b>	<b>13.99</b>	<b>18.04</b>	<b>22.21</b>	<b>18.98</b>	<b>17.20</b>	<b>17.34</b>



# EXPLANATION OF ROBUSTNESS



**FIGURE:** VecKM can reconstruct the local shape under corrupted inputs. The VecKM encodings remain highly similar under data corruptions.



## CLASSIFICATION ON MODELNET40

TABLE: Classification performance on the ModelNet40 dataset.

	Instance Accuracy	Avg. Class Accuracy	Inference Time (ms) (1 batch)	# parameters
PointMLP	93.1%	90.1%	325.85	13.2M
PointNet	90.8%	87.1%	3.04	1.61M
VecKM → PN	92.9%	89.7%	14.32	9.06M
Difference	↑ 2.1%	↑ 2.6%	not comparable	+7.61M
PointNet++	92.7%	89.4%	117.13	1.48M
VecKM ⇌ PN++	93.0%	89.7%	65.78	3.94M
Difference	↑ 0.3%	↑ 0.3%	78% faster	+2.46M
PCT	92.9%	89.8%	149.72	2.88M
VecKM ⇌ PCT	93.1%	90.6%	21.44	5.07M
Difference	↑ 0.2%	↑ 0.8%	5.98x faster	+2.19M



## PART SEGMENTATION ON SHAPENET DATASET

TABLE: Part segmentation performance on the ShapeNet dataset.

	Instance mIoU	Avg. Class mIoU	Inference Time (ms) (1 batch)	parameters
PointMLP	85.1%	82.1%	240.39	16.76M
PointNet	83.1%	77.6%	15.1	8.34M
VecKM → PN	84.9%	81.8%	40.8	1.29M
Difference	↑ 1.8%	↑ 4.2%	not comparable	+7.05M
PointNet++	85.0%	81.9%	130.8	1.41M
VecKM ⇄ PN++	85.3%	82.0%	65.9	1.50M
Difference	↑ 0.3%	↑ 0.1%	98% faster	+0.09M
PCT	85.7%	82.6%	145.2	1.63M
VecKM ⇄ PCT	85.8%	82.6%	46.6	1.71M
Difference	↑ 0.1%	0.0%	2.11x faster	+0.08M



## SEMANTIC SEGMENTATION ON S3DIS DATASET

**TABLE:** Semantic segmentation performance on the S3DIS dataset.

	Instance mIoU	Avg. Class mIoU	Overall Accuracy	Inference Time (ms) (per scene)	#parameters
PointNet++	64.05	71.52	87.92	96	0.968M
VecKM → PN++	67.48	73.53	89.33	391	1.11M
Difference	↑3.43	↑2.01	↑1.41	not comparable	+0.142M
Point Transformer	69.29	75.66	90.36	559	7.77M
VecKM ≈ PT	69.53	75.84	90.39	447	7.93M
Difference	↑0.24	↑0.18	↑0.03	20% faster	+0.16M

