NPNet: A Non-Parametric Network with Adaptive Gaussian–Fourier Positional Encoding for 3D Classification and Segmentation

Mohammad Saeid[1], Amir Salarpour[2], Pedram MohajerAnsari[2], Mert D. Pesé[2]

***Abstract*— We present NPNet, a fully non-parametric approach for 3D point-cloud classification and part segmentation. NPNet contains no learned weights; instead, it builds point features using deterministic operators such as farthest point sampling, k-nearest neighbors, and pooling. Our key idea is an *adaptive Gaussian–Fourier* positional encoding whose bandwidth and Gaussian–cosine mixing are chosen from the input geometry, helping the method remain stable across different scales and sampling densities. For segmentation, we additionally incorporate fixed-frequency Fourier features to provide global context alongside the adaptive encoding. Across ModelNet40/ModelNet-R, ScanObjectNN, and ShapeNetPart, NPNet achieves strong performance among non-parametric baselines, and it is particularly effective in few-shot settings on ModelNet40. NPNet also offers favorable memory use and inference time compared to prior non-parametric methods. Code is available at GitHub: m-saeid/NPNet.**

I. Introduction

Accurate and efficient 3D perception is central to intelligent vehicle systems. Point clouds captured by LiDAR and depth sensors support scene understanding, localization, and planning in autonomous driving [1]–[3]. Similar needs also arise in robotics [4], augmented reality [5], and digital-heritage mapping [6]. Beyond 3D perception, modern sensing pipelines in other application domains also face analogous calibration, prediction, and reliability challenges that motivate efficient and transferable models [7]–[9]. However, point clouds are irregularly sampled and often large, which makes it difficult to learn representations efficiently [10]. This difficulty is amplified by real-time, on-vehicle constraints where compute and memory budgets are limited [11], [12]. In this setting, non-parametric architectures are a practical alternative. They rely on deterministic geometric operators rather than trainable weights, and they can transfer across scenes and sensors without retraining.

Parametric point-cloud networks such as PointNet/++ [13], [14], PointConv [15], KPConv [16], DGCNN [17], PCT [18], and PointMLP [19] achieve accuracy. However, they rely on millions of learned weights and typically require expensive training, which makes rapid adaptation difficult and can be limiting in few-shot settings [20]–[22]. These limitations motivate non-parametric architectures that remove learned parameters while aiming to keep competitive performance.

[1]Mohammad Saeid is with Sirjan University of Technology, Sirjan, Iran m.saeid@stu.sirjantech.ac.ir

[2]Amir Salarpour, Pedram MohajerAnsari, and Mert D. Pesé are with Clemson University, Clemson, SC, USA {asalarp, pmohaje, mpese}@clemson.edu

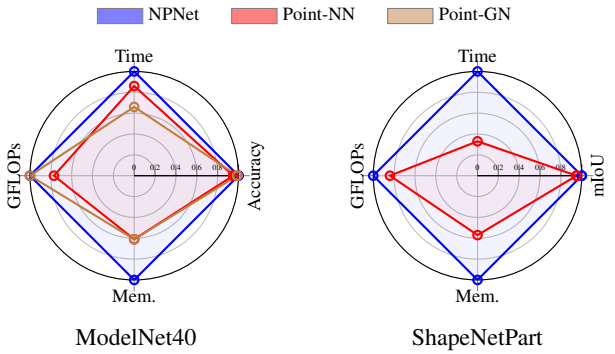


Fig. 1: Radar comparison on ModelNet40 and ShapeNetPart. Metrics are normalized to $[0, 1]$ per dataset (higher is better). GPU memory, GFLOPs, and inference time are inverted so larger values indicate better efficiency.

Recent non-parametric models such as Point-NN [23] and Point-GN [24] are promising. However, their positional encodings are usually fixed, which can make them sensitive to changes in point density, object scale, and sampling distribution. As a result, performance often degrades when moving across datasets or when switching between tasks.

We introduce **NPNet**, a fully non-parametric framework for 3D point-cloud analysis. The core idea is an *adaptive Gaussian–Fourier positional encoding* that chooses its bandwidth and Gaussian–cosine mixing from simple statistics of the input, which helps it stay stable across different scales and sampling densities. For segmentation, we also add fixed-frequency Fourier features to provide global context. This produces a hybrid encoding that combines local adaptivity with global structure without using any trainable weights. NPNet builds multi-scale features using only deterministic operators such as farthest point sampling, k-nearest-neighbor grouping, and pooling. For classification, we perform similarity-based inference, while a Fourier-augmented branch is used for segmentation. As shown in Fig. 1, NPNet improves both accuracy (or mIoU) and efficiency compared with prior zero-parameter methods on ModelNet40 [25] and ShapeNetPart [26].

Across ModelNet40, ModelNet-R [20], ScanObjectNN [27], and ShapeNetPart, NPNet achieves state-of-the-art performance among non-parametric methods and remains competitive with parametric networks. On ModelNet40 few-shot evaluation, it remains competitive, supporting the generalization benefits of the proposed adaptive, training-free encoding.

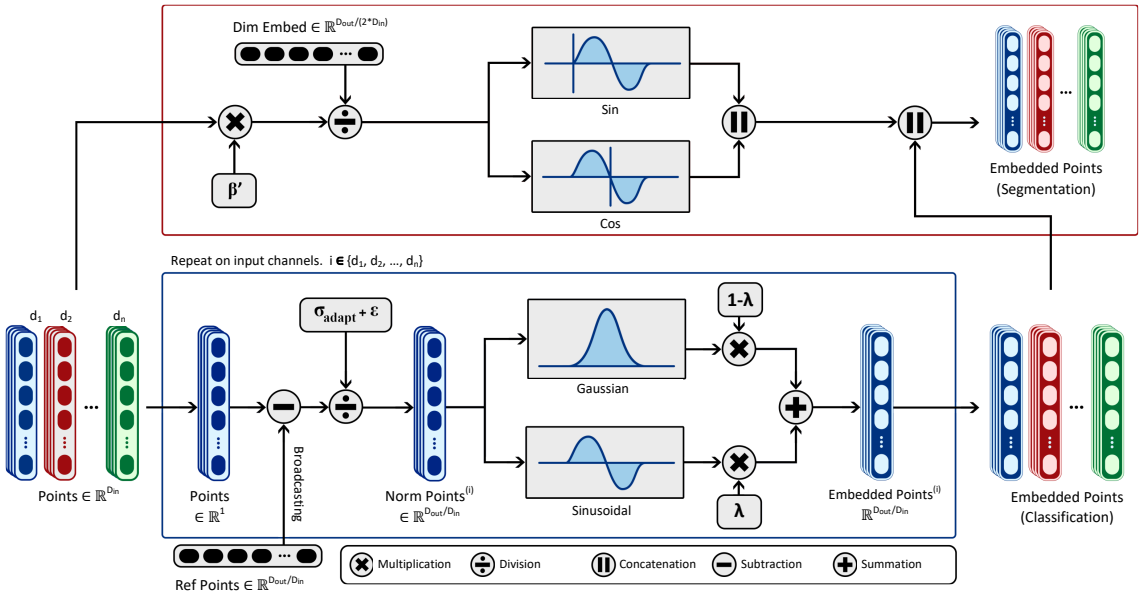


Fig. 2: Adaptive Gaussian–Fourier positional encoding. The encoding adapts bandwidth σ and mixing coefficient λ from input geometry; an additional fixed-frequency Fourier branch provides global context for segmentation.

Contributions.

- We propose NPNet, a fully non-parametric framework with a shared encoder for point-cloud classification and part segmentation.
- We introduce an adaptive Gaussian–Fourier positional encoding. It selects the bandwidth and the Gaussian–cosine mixing from input statistics, and it adds fixed-frequency Fourier features to provide global context for segmentation.
- We show that NPNet is competitive with, and often improves over, prior non-parametric baselines on standard benchmarks, including few-shot ModelNet40, while offering favorable memory use and inference time.

II. Related Work

Representations. Projection-based methods render point clouds into images or depth maps. This allows reuse of 2D CNNs, but it can introduce occlusion and discretization artifacts [28]–[30]. Voxel and octree approaches enable 3D convolutions, yet they often incur high memory cost because of sparsity [31]–[33]. Point-based models operate directly on 3D coordinates and capture geometric detail, but they typically rely on large learned networks [13], [14], [16], [17], [34].

Efficient and non-parametric models. Hand-crafted pipelines and fixed transforms can reduce the need for learning, but they are often brittle in practice [35]–[37]. More recent non-parametric networks such as Point-NN and Point-GN remove trainable weights by combining sampling, k-NN grouping, and fixed positional encodings [23], [24]. However, these encodings use static bandwidths, which can make performance sensitive to changes in point density, object scale, and sampling distribution.

Positional encoding. Positional encodings play an important role in 3D learning [38]. Fixed trigonometric features and static Gaussian kernels can improve discrimination, but they effectively assume a single global setting. In contrast, **NPNet** uses an *adaptive Gaussian–Fourier* encoding that selects the bandwidth and the Gaussian–cosine mixing from input statistics. For segmentation, we also add fixed-frequency Fourier terms to provide global context. This design keeps the simplicity and efficiency of non-parametric models while improving transfer across tasks and datasets.

III. Methodology

A. Overview of NPNet

NPNet is a fully non-parametric framework for 3D point-cloud classification and part segmentation. It uses no trainable weights; instead, it builds features with deterministic geometric operators—farthest point sampling (FPS), k-nearest neighbors (k-NN), pooling—and an *adaptive Gaussian–Fourier* positional encoding. For classification, a multi-stage encoder aggregates local neighborhoods into a single global descriptor and predicts labels by similarity matching to a memory bank of training shapes. For segmentation, an encoder–decoder produces pointwise descriptors and assigns part labels by matching to stored part prototypes. The whole pipeline is training-free: once the memory bank is built, inference reduces to encoding and nearest-prototype style matching.

B. Adaptive Gaussian–Fourier Positional Encoding

Given coordinates $X \in \mathbb{R}^{N\times 3}$, we build positional features with two components: (i) an *adaptive* channel that blends Gaussian RBF and cosine responses using simple input statistics, and (ii) a *Fourier* channel (segmentation only) that adds fixed multi-frequency context. Fig. 2 illustrates the encoding

itself and Fig. 3 shows where the encoding is injected within each stage.

a) Adaptive channel.: Let $\sigma_{\mathrm{g}} = \frac{1}{3}\sum_{i=1}^{3}\mathrm{Std}(X_{:,i})$ denote a global dispersion statistic. We set the bandwidth $\sigma_{\mathrm{a}} = \sigma_0(1+\sigma_{\mathrm{g}})$ and the blend $\lambda = \mathrm{sigmoid}((\sigma_{\mathrm{g}}-\tau)\kappa)$. For anchors $\{v_m\}_{m=1}^{M}$ (fixed reference locations) and a scalar coordinate x,

$$\phi_{\mathrm{RBF}}(x, v_m) = \exp\left(-\tfrac{1}{2}\left(\tfrac{x-v_m}{\sigma_{\mathrm{a}}+\epsilon}\right)^2\right),$$

$$\phi_{\cos}(x, v_m) = \cos\left(\tfrac{x-v_m}{\sigma_{\mathrm{a}}+\epsilon}\right),$$

and the adaptive response is

$$\phi_{\mathrm{adaptive}}(x) = \lambda\,\phi_{\mathrm{RBF}}(x) + (1-\lambda)\,\phi_{\cos}(x).$$

Concatenation over anchors and coordinate channels yields $H_{\mathrm{adaptive}} \in \mathbb{R}^{N\times d'}$.

b) Fourier channel (segmentation).: With frequencies $\omega_j = \alpha^{j/L}$ for $j{=}1\ldots L$ and scale β,

$$\phi_{\mathrm{Fourier}}(x) = [\sin(\beta x/\omega_1), \cos(\beta x/\omega_1), \ldots, \sin(\beta x/\omega_L), \cos(\beta x/\omega_L)].$$

c) Hybrid code and neighborhood modulation.: Classification uses only H_{adaptive}; segmentation concatenates

$$H_{\mathrm{pos}} = \left[\, H_{\mathrm{Fourier}} \,\|\, H_{\mathrm{adaptive}} \,\right].$$

Within each k-NN neighborhood $\mathcal{N}$ (using relative coordinates), features are modulated by

$$\tilde{H}_{\mathcal{N}} = (H_{\mathcal{N}} + H_{\mathrm{pos}}) \odot H_{\mathrm{pos}}, \tag{1}$$

where $\odot$ denotes element-wise multiplication.

C. NPNet Classification Architecture

A T-stage encoder builds a feature pyramid by repeating FPS to N_t centroids, k-NN grouping, adaptive modulation (Eq. 1), and mean+max pooling (Fig. 3). The full architecture is shown in Fig. 4. Let $F_j^{(t)}$ be the pooled feature of centroid j in stage t. We form the global descriptor by concatenating stage summaries:

$$F^{\mathrm{enc}} = \Big\|_{t=1}^{T}\left[\, \max_j F_j^{(t)} \,\|\, \mathrm{mean}_j F_j^{(t)} \,\right] \in \mathbb{R}^D,$$

with normalization/nonlinearities and *no learned layers*. This descriptor is used for similarity-based inference (Sec. III-E).

D. NPNet Segmentation Architecture

The segmentation encoder mirrors the classification backbone but uses the hybrid code H_{pos}. A non-parametric decoder propagates coarse features to finer resolutions via inverse-distance weighting (IDW). For point x_i at level t with neighbors in level $t{+}1$,

$$\hat{h}_i^{(t)} = \sum_{j\in\mathcal{N}(x_i, X^{(t+1)})} \frac{\|x_i - x_j^{(t+1)}\|_2^{-1}}{\sum_k \|x_i - x_k^{(t+1)}\|_2^{-1}}\, h_j^{(t+1)}. \tag{2}$$

The interpolated features $\hat{H}^{(t)}$ are concatenated with encoder features $H^{(t)}$ and propagated to the next finer level until reaching the input resolution, yielding per-point descriptors.

TABLE I: ModelNet40 shape classification. Accuracy (%), trainable parameters (M), and GFLOPs (G).

Method	Acc. (%)	Param (M)	GFLOPs (G)
Parametric Methods			
PointNet [13]	89.2	3.5	0.4
PointNet++ [14]	90.7	1.5	0.8
PointCNN [39]	92.2	0.6	-
PointConv [15]	92.5	18.6	-
OctFormer [40]	92.7	4	31.3
KPConv [16]	92.9	15.2	-
DGCNN [17]	92.9	1.8	2.7
PCT [18]	93.2	2.9	2.3
PointNext-S [41]	93.2	1.4	-
GBNet [42]	93.8	8.4	-
CurveNet [43]	93.8	2.1	0.3
PointMLP [19]	**94.1**	13.2	15.7
Non-parametric Methods			
Point-NN [23]	81.8	0.0	0.0
Point-GN [24]	85.3	0.0	0.0
NPNet (ours)	**85.45**	0.0	0.0

E. Non-Parametric Training and Inference

NPNet follows a simple routine: encode the training set once to build a memory bank, then predict by similarity—no gradient updates.

a) Classification.: For training shapes (X_i, y_i), compute normalized descriptors $f_i = \mathcal{E}_{\mathrm{cls}}(X_i)/\|\mathcal{E}_{\mathrm{cls}}(X_i)\|_2$ and store $F = [f_1, \ldots, f_{M_{\mathrm{tr}}}] \in \mathbb{R}^{D\times M_{\mathrm{tr}}}$ with one-hot labels $Y \in \{0,1\}^{M_{\mathrm{tr}}\times C}$. For a test feature f, similarities $s = f^\top F$ yield weights $w = \mathrm{softmax}(\gamma s)$ (temperature γ). The prediction is $\hat{z} = w^\top Y$ and $\hat{y} = \arg\max_c \hat{z}_c$.

b) Segmentation.: For each training shape, extract per-point features $F_i = \mathcal{E}_{\mathrm{seg}}(X_i)$ and form part prototypes $\mu_{i,p}$ by averaging features over points with label p. Store prototypes per shape category s as (F_s, Y_s). For a test point feature f_j (category s), compute $s_j = f_j^\top F_s$, $w_j = \mathrm{softmax}(\gamma s_j)$, $\hat{z}_j = w_j^\top Y_s$; assign $\hat{y}_j = \arg\max_p \hat{z}_{j,p}$.

IV. Experiments

We evaluate NPNet on standard benchmarks for shape classification, few-shot classification, and part segmentation. We summarize datasets and metrics, describe the training-free setup and hyperparameters, then report results and efficiency. Finally, we ablate the adaptive Gaussian–Fourier encoding and key design choices.

A. Datasets and Evaluation Metrics

ModelNet40: 40 CAD categories (9,843 train / 2,468 test). Metric: overall accuracy (Tab. I).
ModelNet-R: relabeled/cleaned ModelNet40 variant. Metric: overall accuracy (Tab. II).
ScanObjectNN: real-world RGB-D scans with clutter, occlusion, and noise. Metric: accuracy on OBJ-BG, OBJ-ONLY, and PB-T50-RS (Tab. III).
Few-shot ModelNet40: N-way/K-shot episodic classification. Metric: mean accuracy (Tab. IV).
ShapeNetPart: 16 categories with 50 part labels. Metric: instance mIoU (Tab. V).

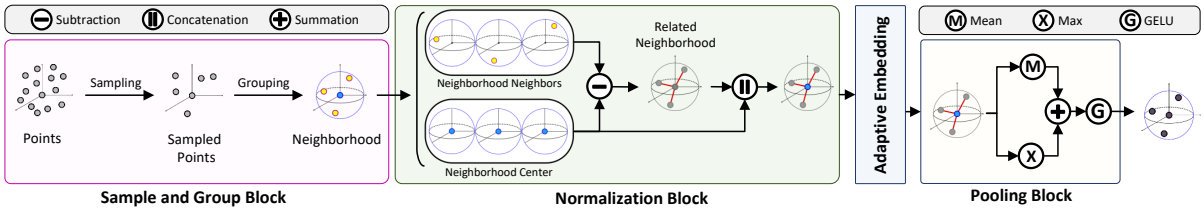


Fig. 3: Stage block used in NPNet. FPS selects centroids, k-NN groups local neighborhoods, positional encoding modulates features, and mean/max pooling produces a stage descriptor; concatenating stages forms a multi-scale representation.

TABLE II: ModelNet-R shape classification. Accuracy (%), trainable parameters (M), and GFLOPs (G) where available.

Method	Acc. (%)	Param (M)	GFLOPs (G)
Parametric Methods			
PointNet [13]	91.4	3.5	0.4
PointNet++ (SSG) [14]	94.0	1.5	0.8
PointNet++ (MSG) [14]	94.0	1.7	4.0
DGCNN [17]	94.0	1.8	2.7
CurveNet [43]	94.1	2.1	0.3
Point-SkipNet [20]	94.3	1.5	-
PointMLP [19]	**95.3**	13.2	15.7
Non-parametric Methods			
Point-NN [23]	84.75	**0.0**	**0.0**
NPNet (ours)	**85.65**	**0.0**	**0.0**

B. Implementation Details and NPNet Hyperparameters

NPNet is training-free: one forward pass over the training set builds the memory banks (shape descriptors for classification; part prototypes for segmentation). Unless stated otherwise, we sample N=1,024 points per shape and disable augmentation to isolate the effect of the adaptive encoding.

Batch sizes and hardware. ModelNet40 and few-shot use batch size 10, ScanObjectNN 16, and ShapeNetPart 1 (per-shape). All runs use a single NVIDIA GeForce RTX 3090 (24 GB, CUDA 11.5).

Baseline hyperparameters. ModelNet40/ModelNet-R/few-shot: d=35, k=110, stages= 4, N=1,024.
ScanObjectNN: d=27, k=120, stages= 4, N=1,024.
ShapeNetPart: d=144, k=70, stages= 2, N=1,024.
These settings follow Sec. IV-H and balance accuracy against runtime and memory.

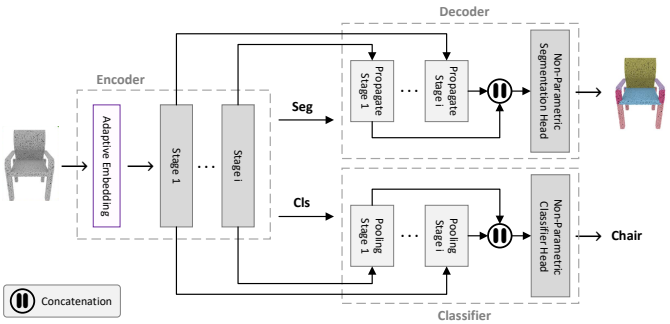


Fig. 4: NPNet architecture for classification and part segmentation. The model contains no trainable weights; inference is performed by similarity matching to stored shape descriptors (classification) or part prototypes (segmentation).

C. Classification Results

On **ModelNet40** and **ModelNet-R** (Tabs. I, II), NPNet achieves **85.45%** and **85.65%** accuracy with **0.0M** parameters, outperforming prior non-parametric methods. On **ScanObjectNN** (Tab. III), NPNet reaches **86.1%/86.1%/84.9%** on OBJ-BG/OBJ-ONLY/PB-T50-RS, leading non-parametric baselines on OBJ-BG and OBJ-ONLY while remaining competitive on PB-T50-RS. Recent parametric models score higher but rely on millions of trainable weights; NPNet offers a strong training-free alternative.

D. Few-Shot Classification

Few-shot evaluation matches NPNet's inference: each episode is solved by building descriptors and matching prototypes, with no fine-tuning. Averaged over 10 runs on ModelNet40, NPNet attains **92.0%/93.2%** (5-way 10-/20-shot) and **82.5%/87.6%** (10-way 10-/20-shot). Tab. IV shows NPNet is best among the listed methods across all settings.

E. Segmentation Results

Tab. V reports ShapeNetPart results. NPNet achieves **73.56%** instance mIoU using the hybrid Gaussian–Fourier encoding. Compared with Point-NN (70.4%), the gain suggests that adding fixed-frequency Fourier terms alongside the adaptive channel helps capture global structure and repeated patterns that are useful for part boundaries. We use the baseline hyperparameters from Sec. IV-B (d=144, k=70, stages= 2).

TABLE III: ScanObjectNN classification. Accuracy (%) on OBJ-BG, OBJ-ONLY, and PB-T50-RS, with trainable parameters (M).

Method	OBJ -BG	OBJ- ONLY	PB-T50 -RS	Param (M)
Parametric Methods				
PointNet [13]	73.3	79.2	68.2	3.5
PointNet++ [14]	82.3	84.3	77.9	1.5
DGCNN [17]	82.8	**86.2**	78.1	1.8
PointCNN [39]	**86.1**	85.5	78.5	-
GBNet [42]	-	-	80.5	8.4
PointMLP [19]	-	-	85.4	12.6
PointNeXt-S [41]	-	-	87.7	1.5
PointMetaBase-S [44]	-	-	**87.9**	0.6
Non-parametric Methods				
Point-NN [23]	71.1	74.9	64.9	0.0
Point-GN [24]	85.2	86.0	**86.4**	0.0
NPNet (ours)	**86.1**	**86.1**	84.9	0.0

TABLE IV: Few-shot classification on ModelNet40. Mean accuracy (%) over 10 runs for 5-way and 10-way tasks; baseline results are taken from the cited reference.

Method	5-way		10-way	
	10-shot	20-shot	10-shot	20-shot
Parametric Methods				
DGCNN [17]	31.6	40.8	19.9	16.9
FoldingNet [45]	33.4	35.8	18.6	15.4
PointNet++ [14]	38.5	42.4	23.0	18.8
PointNet [13]	52.0	57.8	46.6	35.2
3D-GAN [46]	55.8	65.8	40.3	48.4
PointCNN [39]	65.4	68.6	46.6	50.0
Non-parametric Methods				
Point-NN [23]	88.8	90.9	79.9	84.9
Point-GN [24]	90.7	90.9	81.6	86.4
NPNet (ours)	**92.0**	**93.2**	**82.5**	**87.6**

TABLE V: ShapeNetPart part segmentation. Instance mIoU (%), input points, trainable parameters (M), GFLOPs (G), and training time (when reported).

Method	Inputs	Inst. mIoU	Param (M)	GFLOPs (G)	Train Time
Parametric Methods					
PointNet [13]	2k	83.7	8.3	5.8	-
PointNet++ [14]	2k	85.1	1.8	1.1	26.5 h
DGCNN [17]	2k	85.2	1.4	4.9	-
APES(local) [47]	2k	85.6	2.0	18.37	-
APES(global) [47]	2k	85.8	2.0	15.56	-
PAConv [48]	2k	86.0	-	-	-
PointMLP [19]	2k	86.1	16.8	6.2	47.1 h
CurveNet [43]	2k	**86.6**	5.5	2.5	56.9 h
Non-parametric Methods					
Point-NN [23]	1k	70.4	0.0	0.0	0.0
NPNet (ours)	1k	**73.56**	0.0	0.0	0.0

TABLE VI: Efficiency of non-parametric methods. GFLOPs (G), peak GPU memory (MB), and inference time (ms/sample) on ModelNet40 and ShapeNetPart using 1,024 points; measured on an RTX 3090.

Method	Dataset	GFLOPs (G)	Mem. (MB)	Inference (ms/sample)	Num Points
Point-NN [23]	ModelNet40	0.0027	161.0	4.44	1024
Point-GN [24]	ModelNet40	**0.0021**	161.0	5.80	1024
NPNet (ours)	ModelNet40	**0.0021**	**99.1**	**3.86**	1024
Point-NN [23]	ShapeNetPart	0.0054	442.9	16.83	1024
NPNet (ours)	ShapeNetPart	**0.0045**	**256.4**	**5.63**	1024

F. Efficiency and Deployment Analysis

NPNet's cost profile is driven by neighborhood aggregation and positional encoding rather than learned layers: (i) **no training** (a single pass builds the banks), (ii) **inference** dominated by encoding, grouping, and lightweight similarity, and (iii) **memory/runtime** controlled mainly by d and k. On **ModelNet40** (Tab. VI), NPNet uses **0.0021 GFLOPs**, **99.1 MB** GPU memory, and **3.86 ms/sample**; on **ShapeNetPart** it uses **0.0045 GFLOPs**, **256.4 MB**, and **5.63 ms/sample**—faster and leaner than Point-NN/Point-GN.

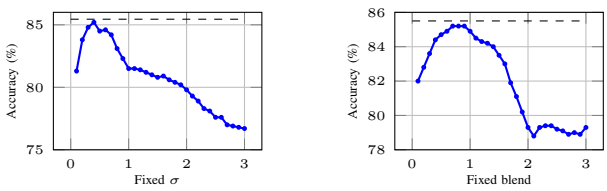
One-off bank construction and scaling. NPNet incurs a one-time preprocessing cost to build the memory banks: for classification, storing one descriptor per training shape; for segmentation, storing part prototypes per category. This cost is linear in the number of training shapes and is typically amortized over many test queries. The bank memory footprint scales with descriptor dimension, i.e., $\mathcal{O}(M_{\text{train}}D)$ for classification and $\mathcal{O}(M_{\text{proto}}D)$ for segmentation, where M_{proto} depends on the number of shapes and parts per category. When memory is constrained, descriptor compression (e.g., FP16 storage) or prototype subsampling/clustering can reduce the footprint with minimal impact. For very large banks, approximate nearest-neighbor search can further reduce query time while keeping the training-free pipeline unchanged.

G. Discussion: Complexity, Scope, and Reproducibility

Complexity and scaling. Let N be the number of input points, T the number of stages, k the neighborhood size, and d the embedding dimension. At stage t, FPS selects N_t centroids, and each centroid aggregates a k-NN neighborhood with modulation (Sec. III-B) and mean–max pooling. The per-stage work scales as $\mathcal{O}(N_t k d)$ and memory as $\mathcal{O}(N_t d + k d)$, so the encoder cost is $\sum_{t=1}^{T} \mathcal{O}(N_t k d)$. Segmentation adds IDW upsampling with the same order (Eq. 2). In practice, k and d are the main levers: larger d improves accuracy, larger k improves context but increases cost, and deeper hierarchies help classification more than segmentation.

Assumptions and limitations. (1) *Rotation equivariance:* the encoding is not rotation-equivariant; canonical alignment or test-time rotation averaging can help. (2) *Category for segmentation:* following ShapeNetPart protocol, the object category is known at test time; removing this assumption is future work. (3) *Memory-bank size:* the bank grows with the number of training shapes/prototypes; clustering or subsampling can cap memory with modest impact. (4) *k-NN backend:* very sparse or very large N may benefit from approximate neighbors; NPNet is compatible with GPU ANN backends.

Protocol and reproducibility. All experiments use N=1,024 points per shape unless stated. Some parametric baselines in Tab. V report 2k-point results; our efficiency numbers (Tab. VI) are measured at 1k points for a consistent runtime/memory comparison. We fix random seeds and evaluate on a single RTX 3090 (24 GB). Code and scripts to reproduce tables and figures are publicly available.



(a) Fixed-σ sweep (b) Fixed-blend sweep

Fig. 5: Effect of disabling adaptivity on ModelNet40. Accuracy when (a) σ is fixed and (b) the Gaussian–cosine mixing ratio is fixed in the positional encoding.

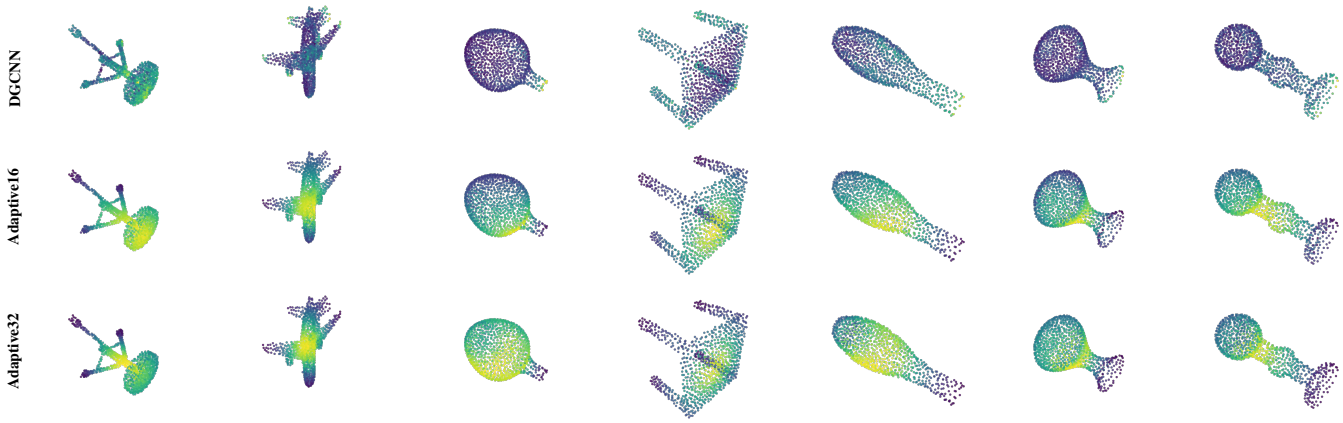


Fig. 6: Qualitative attention visualization. Top: DGCNN attention after the second EdgeConv block. Middle/Bottom: activation maps from the proposed AdaptiveEmbedding module with output dimension 16 and 32, respectively.

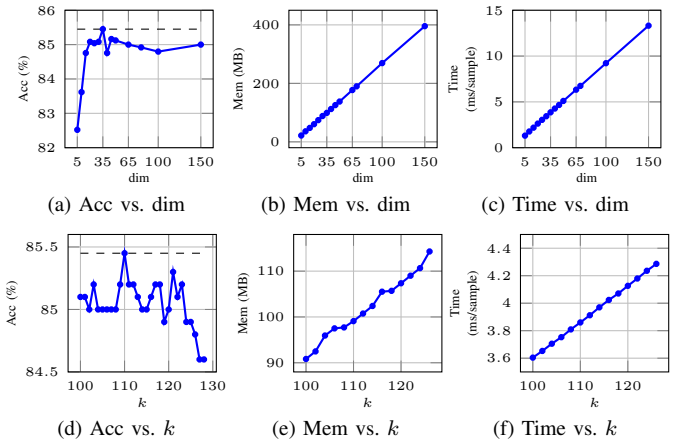
H. Ablation Studies

We ablate the key claim: *input-aware adaptivity* in the positional encoding is necessary for stable, training-free performance across datasets and tasks. We use ModelNet40 for classification and ShapeNetPart for segmentation. Each sweep varies one factor while holding others fixed: (i) adaptivity (Fig. 5); (ii) embedding dimension d (Figs. 7, 8); (iii) neighborhood size k (Figs. 7, 8); and (iv) stage depth (Fig. 9).

Adaptivity of bandwidth and blend. Fixing σ or the Gaussian/cosine blend consistently hurts performance (Figs. 5a, 5b). Accuracy peaks in a narrow window and drops quickly outside it, while the adaptive scheme maintains strong results without per-dataset tuning. This supports the use of per-input σ and λ.

Embedding dimension. Classification saturates around $d = 30$–40 (Fig. 7a); we use $d = 35$ to stay near the peak while keeping cost low. In contrast, segmentation benefits from larger embeddings and saturates later (Fig. 8a); we choose $d = 144$ as a balanced point before the marginal mIoU gains become small relative to the additional compute and memory. This task-dependent behavior is expected: classification mainly requires a global shape signature, whereas segmentation must preserve part-level cues and boundary detail, which benefit from higher-capacity embeddings. As d increases, GPU memory and inference time grow approximately linearly for both tasks (Figs. 7b–7c and 8b–8c), making d an effective knob for trading accuracy against efficiency.

Neighborhood size k. Accuracy peaks near $k = 110$ on ModelNet40 (Fig. 7d) and near $k = 70$ on ShapeNetPart (Fig. 8d). Larger neighborhoods provide more geometric context, which can help classification up to a point; however, overly large k can dilute local detail that is important for part boundaries and can interact unfavorably with downsampling in deeper hierarchies. Consistent with this intuition, ShapeNetPart prefers a moderate k, while ModelNet40 tolerates a larger neighborhood for stronger context aggregation. In all cases, increasing k raises GPU memory and inference time steadily because neighborhood grouping dominates the encoder cost (Figs. 7e–7f and 8e–8f), so k is a direct lever for reducing latency when needed.



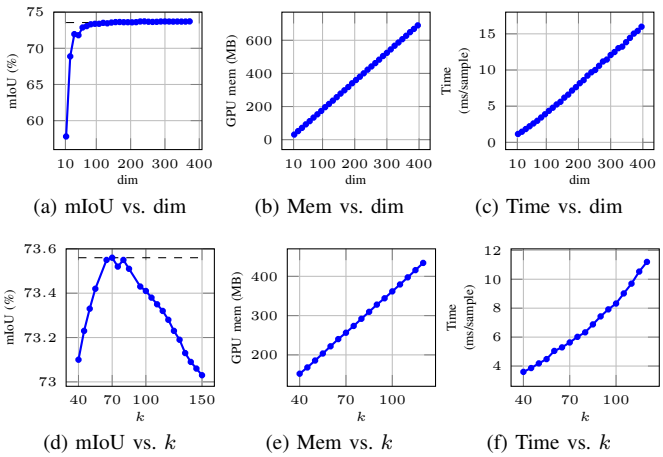
(a) Acc vs. dim (b) Mem vs. dim (c) Time vs. dim

(d) Acc vs. k (e) Mem vs. k (f) Time vs. k

Fig. 7: ModelNet40 ablation (classification). Top: varying embedding dimension d. Bottom: varying neighborhood size k. Columns show accuracy, GPU memory, and inference time.

Number of stages. Using more stages improves classification, with 4 stages performing best on ModelNet40 (Fig. 9). Deeper hierarchies increase receptive field and multi-scale aggregation, which benefits global recognition. For segmentation, using more than 2 stages reduces mIoU and increases cost (Fig. 9), likely because aggressive downsampling and broad aggregation blur fine-grained part boundaries. Accordingly, we use 4 stages for classification and 2 stages for segmentation as a simple, robust default.

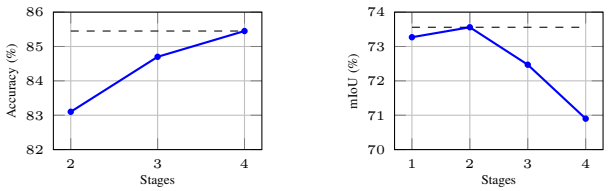
Resource–accuracy trade-offs and practical tuning. Across both tasks, the ablations reveal a trade-off: accuracy (or mIoU) improves with larger d and richer neighborhoods, while GPU memory and inference time grow nearly monotonically with d and k (Figs. 7–8). In practice, d is a coarse knob that sets representational capacity, whereas k controls the locality–context balance. For ModelNet40, performance saturates quickly with d and exhibits a clear peak with k, so a moderate embedding with a moderately large neighborhood is sufficient. In contrast,



(a) mIoU vs. dim (b) Mem vs. dim (c) Time vs. dim

(d) mIoU vs. k (e) Mem vs. k (f) Time vs. k

Fig. 8: ShapeNetPart ablation (segmentation). Top row: varying embedding dimension d. Bottom row: varying neighborhood size k. Columns show instance mIoU, GPU memory, and inference time.



(a) ModelNet40 accuracy (b) ShapeNetPart mIoU

Fig. 9: Effect of stage depth. (a) ModelNet40 classification accuracy vs. number of stages. (b) ShapeNetPart instance mIoU vs. number of stages.

ShapeNetPart benefits from larger embeddings to capture part-level structure, but overly large neighborhoods and deeper hierarchies can hurt because aggressive downsampling and broad aggregation blur fine boundaries. hu From a deployment perspective, these trends provide simple guidelines. If memory is the bottleneck, reducing d yields the most direct savings with limited accuracy loss near the saturation region. If inference time is the bottleneck, reducing k and/or stage depth offers the largest speedups because neighborhood aggregation dominates runtime. Finally, the adaptivity in σ and the Gaussian–cosine mixing reduces the need to retune these hyperparameters across datasets with different sampling densities (Fig. 5), allowing d, k, and stages to serve as the main task-dependent controls.

Qualitative comparison. Fig. 6 compares attention maps from the parametric DGCNN with those produced by the proposed AdaptiveEmbedding module. Despite being non-parametric, the adaptive embedding yields more coherent and part-aware responses, indicating that it captures meaningful geometric structure without learned weights.

I. Takeaways

NPNet demonstrates that well-designed non-parametric components—particularly the adaptive Gaussian–Fourier positional encoding—can substantially close the gap to parametric networks on both classification and segmentation. The method is especially appealing in few-shot and rapid-deployment scenarios because it eliminates the training step and adapts to the input geometry at inference time. Ablations provide clear guidance for selecting d, k, and stage depth to meet differing accuracy and efficiency requirements.

V. Conclusion

We presented **NPNet**, a fully non-parametric approach for 3D point-cloud classification and part segmentation. Its core is an *adaptive Gaussian–Fourier positional encoding* that adjusts to the input geometry, improving stability across scales and sampling densities without introducing trainable parameters. Across ModelNet40, ModelNet-R, ScanObjectNN, and ShapeNetPart, NPNet outperforms prior non-parametric baselines and remains competitive with parametric networks while eliminating training altogether. NPNet is also efficient in practice: memory and latency are primarily controlled by the embedding dimension and neighborhood size, which makes deployment tuning straightforward. Few-shot results further show that NPNet adapts well to limited supervision through prototype-based inference, with no fine-tuning.

Future work will extend NPNet to detection and scene-level tasks and investigate lightweight hybrid designs that preserve the training-free core while adding minimal learnable capacity when needed.

References

[1] Y. Li, L. Ma, Z. Zhong, F. Liu, M. A. Chapman, D. Cao, and J. Li, "Deep learning for lidar point clouds in autonomous driving: A review," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 8, pp. 3412–3432, 2020. 1

[2] Y. Cui, R. Chen, W. Chu, L. Chen, D. Tian, Y. Li, and D. Cao, "Deep learning for image and point cloud fusion in autonomous driving: A review," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 2, pp. 722–739, 2021. 1

[3] R. Abbasi, A. K. Bashir, H. J. Alyamani, F. Amin, J. Doh, and J. Chen, "Lidar point cloud compression, processing and learning for autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 1, pp. 962–979, 2022. 1

[4] H. Duan, P. Wang, Y. Huang, G. Xu, W. Wei, and X. Shen, "Robotics dexterous grasping: The methods based on point cloud and deep learning," *Frontiers in Neurorobotics*, vol. 15, p. 658280, 2021. 1

[5] B. Mahmood, S. Han, and D.-E. Lee, "Bim-based registration and localization of 3d point clouds of indoor scenes using geometric features for augmented reality," *Remote Sensing*, vol. 12, no. 14, p. 2302, 2020. 1

[6] L. Melas-Kyriazi, C. Rupprecht, and A. Vedaldi, "Pc2: Projection-conditioned point cloud diffusion for single-image 3d reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 12 923–12 932. 1

[7] H. Rajoli, P. Afshin, and F. Afghah, "Thermal image calibration and correction using unpaired cycle-consistent adversarial networks," in *2023 57th Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2023, pp. 1425–1429. 1

[8] M. Saberian, V. Samadi, and I. Popescu, "Probabilistic hierarchical interpolation and interpretable configuration for flood prediction," *Hydrology and Earth System Sciences Discussions*, vol. 2024, pp. 1–41, 2024. 1

[9] A. Kokhahi and D. Li, "Glhad: A group lasso-based hybrid attack detection and localization framework for multistage manufacturing systems," *arXiv preprint arXiv:2304.07673*, 2023. 1

[10] Y. Lin, Z. Zhang, H. Tang, H. Wang, and S. Han, "Pointacc: Efficient point cloud accelerator," in *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, 2021, pp. 449–461. 1

[11] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham, "Randla-net: Efficient semantic segmentation of large-scale point clouds," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 108–11 117. 1

[12] Y. Cao, Y. Wang, and H. Chen, "Real-time lidar point cloud compression and transmission for resource-constrained robots," *arXiv preprint arXiv:2502.06123*, 2025. 1

[13] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660. 1, 2, 3, 4, 5

[14] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in neural information processing systems*, vol. 30, 2017. 1, 2, 3, 4, 5

[15] W. Wu, Z. Qi, and L. Fuxin, "Pointconv: Deep convolutional networks on 3d point clouds," in *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, 2019, pp. 9621–9630. 1, 3

[16] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, "Kpconv: Flexible and deformable convolution for point clouds," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 6411–6420. 1, 2, 3

[17] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *ACM Transactions on Graphics (tog)*, vol. 38, no. 5, pp. 1–12, 2019. 1, 2, 3, 4, 5

[18] M.-H. Guo, J.-X. Cai, Z.-N. Liu, T.-J. Mu, R. R. Martin, and S.-M. Hu, "Pct: Point cloud transformer," *Computational visual media*, vol. 7, pp. 187–199, 2021. 1, 3

[19] X. Ma, C. Qin, H. You, H. Ran, and Y. Fu, "Rethinking network design and local geometry in point cloud: A simple residual mlp framework," *arXiv preprint arXiv:2202.07123*, 2022. 1, 3, 4, 5

[20] M. Saeid, A. Salarpour, and P. MohajerAnsari, "Enhancing 3d point cloud classification with modelnet-r and point-skipnet," *arXiv preprint arXiv:2509.05198*, 2025. 1, 4

[21] L. Gu, X. Yan, L. Nan, D. Zhu, H. Chen, W. Wang, and M. Wei, "Pointenet: A lightweight framework for effective and efficient point cloud analysis," *Computer Aided Geometric Design*, vol. 110, p. 102311, 2024. 1

[22] K. Sugiura, M. Yasuda, and H. Matsutani, "Pointode: Lightweight point cloud learning with neural ordinary differential equations on edge," *arXiv preprint arXiv:2506.00438*, 2025. 1

[23] R. Zhang, L. Wang, Z. Guo, Y. Wang, P. Gao, H. Li, and J. Shi, "Parameter is not all you need: Starting from non-parametric networks for 3d point cloud analysis," *arXiv preprint arXiv:2303.08134*, 2023. 1, 2, 3, 4, 5

[24] M. Mohammadi and A. Salarpour, "Point-gn: A non-parametric network using gaussian positional encoding for point cloud classification," *arXiv preprint arXiv:2412.03056*, 2024. 1, 2, 3, 4, 5

[25] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. 1

[26] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su *et al.*, "Shapenet: An information-rich 3d model repository," *arXiv preprint arXiv:1512.03012*, 2015. 1

[27] M. A. Uy, Q.-H. Pham, B.-S. Hua, T. Nguyen, and S.-K. Yeung, "Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 1588–1597. 1

[28] P. Ahn, J. Yang, E. Yi, C. Lee, and J. Kim, "Projection-based point convolution for efficient point cloud segmentation," *IEEE Access*, vol. 10, pp. 15 348–15 358, 2022. 2

[29] A. Jhaldiyal and N. Chaudhary, "Semantic segmentation of 3d lidar data using deep learning: a review of projection-based methods," *Applied Intelligence*, vol. 53, no. 6, pp. 6844–6855, 2023. 2

[30] M. Gopi and S. Krishnan, "A fast and efficient projection-based approach for surface reconstruction," in *Proceedings. XV Brazilian Symposium on Computer Graphics and Image Processing*. IEEE, 2002, pp. 179–186. 2

[31] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4490–4499. 2

[32] Y. Xu, X. Tong, and U. Stilla, "Voxel-based representation of 3d point clouds: Methods, applications, and its potential use in the construction industry," *Automation in Construction*, vol. 126, p. 103675, 2021. 2

[33] H. Aljumaily, D. F. Laefer, D. Cuadra, and M. Velasco, "Point cloud voxel classification of aerial urban lidar using voxel attributes and random forest approach," *International Journal of Applied Earth Observation and Geoinformation*, vol. 118, p. 103208, 2023. 2

[34] S. M. Peyghambarzadeh, F. Azizmalayeri, H. Khotanlou, and A. Salarpour, "Point-planenet: Plane kernel based convolutional neural network for point clouds analysis," *Digital Signal Processing*, vol. 98, p. 102633, 2020. 2

[35] M. Zhang, H. You, P. Kadam, S. Liu, and C.-C. J. Kuo, "Pointhop: An explainable machine learning method for point cloud classification," *IEEE Transactions on Multimedia*, vol. 22, no. 7, pp. 1744–1755, 2020. 2

[36] P. Kadam, M. Zhang, S. Liu, and C.-C. J. Kuo, "R-pointhop: A green, accurate, and unsupervised point cloud registration method," *IEEE Transactions on Image Processing*, vol. 31, pp. 2710–2725, 2022. 2

[37] M. Mohammadi, A. Salarpour, and P. MohajerAnsari, "Point-ln: A lightweight framework for efficient point cloud classification using non-parametric positional encoding," *arXiv preprint arXiv:2501.14238*, 2025. 2

[38] D. Lu, Q. Xie, M. Wei, K. Gao, L. Xu, and J. Li, "Transformers in 3d point clouds: A survey," *arXiv preprint arXiv:2205.07417*, 2022. 2

[39] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "Pointcnn: Convolution on x-transformed points," *Advances in neural information processing systems*, vol. 31, 2018. 3, 4, 5

[40] P.-S. Wang, "Octformer: Octree-based transformers for 3d point clouds," *ACM Transactions on Graphics (TOG)*, vol. 42, no. 4, pp. 1–11, 2023. 3

[41] G. Qian, Y. Li, H. Peng, J. Mai, H. Hammoud, M. Elhoseiny, and B. Ghanem, "Pointnext: Revisiting pointnet++ with improved training and scaling strategies," *Advances in neural information processing systems*, vol. 35, pp. 23 192–23 204, 2022. 3, 4

[42] S. Qiu, S. Anwar, and N. Barnes, "Geometric back-projection network for point cloud classification," *IEEE Transactions on Multimedia*, vol. 24, pp. 1943–1955, 2021. 3, 4

[43] T. Xiang, C. Zhang, Y. Song, J. Yu, and W. Cai, "Walk in the cloud: Learning curves for point clouds shape analysis," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 915–924. 3, 4, 5

[44] H. Lin, X. Zheng, L. Li, F. Chao, S. Wang, Y. Wang, Y. Tian, and R. Ji, "Meta architecture for point cloud analysis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 17 682–17 691. 4

[45] Y. Yang, C. Feng, Y. Shen, and D. Tian, "Foldingnet: Point cloud auto-encoder via deep grid deformation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 206–215. 5

[46] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum, "Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling," *Advances in neural information processing systems*, vol. 29, 2016. 5

[47] C. Wu, J. Zheng, J. Pfrommer, and J. Beyerer, "Attention-based point cloud edge sampling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 5333–5343. 5

[48] M. Xu, R. Ding, H. Zhao, and X. Qi, "Paconv: Position adaptive convolution with dynamic kernel assembling on point clouds," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 3173–3182. 5