SLNet: A Super-Lightweight Geometry-Adaptive Network for 3D Point Cloud Recognition

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

***Abstract*— We present SLNet, a lightweight backbone for 3D point cloud recognition designed to achieve strong performance without the computational cost of many recent attention, graph, and deep MLP based models. The model is built on two simple ideas: NAPE (Nonparametric Adaptive Point Embedding), which captures spatial structure using a combination of Gaussian RBF and cosine bases with input adaptive bandwidth and blending, and GMU (Geometric Modulation Unit), a per channel affine modulator that adds only $2D$ learnable parameters. These components are used within a four stage hierarchical encoder with FPS+kNN grouping, nonparametric normalization, and shared residual MLPs. In experiments, SLNet shows that a very small model can still remain highly competitive across several 3D recognition tasks. On ModelNet40, SLNet-S with 0.14M parameters and 0.31 GFLOPs achieves 93.64% overall accuracy, outperforming PointMLP-elite with $5\times$ fewer parameters, while SLNet-M with 0.55M parameters and 1.22 GFLOPs reaches 93.92%, exceeding PointMLP with $24\times$ fewer parameters. On ScanObjectNN, SLNet-M achieves 84.25% overall accuracy within 1.2 percentage points of PointMLP while using $28\times$ fewer parameters. For large scale scene segmentation, SLNet-T extends the backbone with local Point Transformer attention and reaches 58.2% mIoU on S3DIS Area 5 with only 2.5M parameters, more than $17\times$ fewer than Point Transformer V3. We also introduce NetScore$^+$, which extends NetScore by incorporating latency and peak memory so that efficiency can be evaluated in a more deployment oriented way. Across multiple benchmarks and hardware settings, SLNet delivers a strong overall balance between accuracy and efficiency. Code is available at: https://github.com/m-saeid/SLNet.**

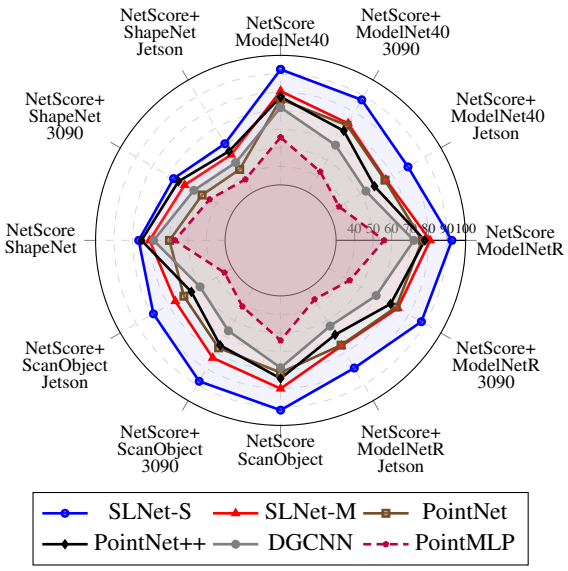

Fig. 1: NetScore and NetScore$^+$ across four datasets and two hardware platforms (RTX 3090 and Jetson Orin Nano), spanning 12 metric–dataset–hardware axes. SLNet-S and SLNet-M consistently occupy the outermost positions, establishing Pareto-optimal deployability across all evaluated configurations.

I. Introduction

Real-time 3D perception is important in applications such as autonomous driving [1], robotics [2], and augmented reality [3]. In many of these settings, especially on edge devices, models must operate under strict limits on latency, memory, and power. Point clouds remain a widely used 3D representation because they preserve fine geometric detail without quantization artifacts [4]. However, many strong point cloud backbones remain too expensive for resource-constrained deployment, often requiring more than 0.7M parameters and 1 GFLOP even at 1k input points.

Existing methods fall into three broad groups, each with a different efficiency bottleneck. Shared-MLP hierarchies, such as PointNet/++ [5], [6] and PointMLP [7], are effective but tend to grow in parameter count and latency with model capacity. Graph and kernel based methods, including DGCNN [8] and KPConv [9], rely on repeated neighborhood construction and local aggregation, which can be costly on edge hardware. Transformer based models, such as Point Transformer [10] and Point-BERT [11], often achieve strong accuracy but increase memory use and inference cost because of attention and large learned embeddings [12]. At the other extreme, ultra-compact non-parametric models such as NPNet [13], Point-GN [14], and Point-NN [15] are efficient but usually trail supervised baselines on harder benchmarks.

We present ***SLNet***, a lightweight backbone designed to improve the accuracy-efficiency trade-off in point cloud recognition. It is built around two components: **NAPE** (Non-parametric Adaptive Point Embedding), which encodes raw XYZ coordinates through an input-adaptive combination of Gaussian radial basis function (RBF) and cosine bases without learned parameters, and **GMU** (Geometric Modulation Unit), a simple per-channel affine recalibration module with only $2D$ learnable parameters. These components are combined with four hierarchical FPS+kNN stages, parameter-

[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

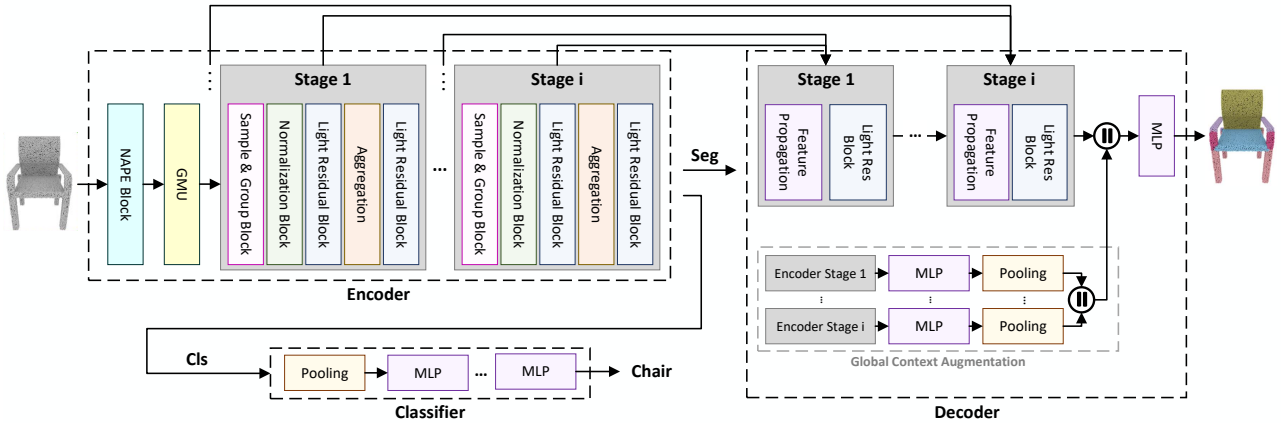

Fig. 2: ***SLNet*** architecture: a NAPE+GMU front end, a four-stage hierarchical encoder with FPS, parameter-free normalization, and lightweight residual refinement, and task-specific heads for classification and segmentation.

free normalization, and shared residual MLPs. For large-scale indoor scene segmentation, ***SLNet-T*** keeps the same overall hierarchy, replaces the NAPE front end with a learned linear projection, and adds local Point Transformer attention at all four encoder stages.

Figure 1 summarizes NetScore and NetScore$^+$ across 12 metric, dataset, and hardware combinations. In this comparison, SLNet-S shows the strongest overall deployability profile, while SLNet-M also remains consistently competitive across settings. In the few-shot setting, ***SLNet-M*** reaches 95.0% accuracy on 5-way 20-shot classification and 94.5% on 10-way 20-shot classification without large-scale pretraining.

We also introduce NetScore$^+$, a deployment-oriented metric that combines accuracy, parameter count, FLOPs, latency, and memory. Across 12 architectures, it shows strong correlation with measured throughput (Spearman $\rho > 0.9$). We evaluate it on ModelNet40, ModelNet-R, ScanObjectNN, ShapeNetPart, and ModelNet40 few-shot, and also report S3DIS results for ***SLNet***-T.

Contributions:

- We introduce **NAPE** and **GMU**, a lightweight combination of nonparametric geometric encoding and ultra-low-cost channel modulation.
- We present ***SLNet*** in three variants, **S**, **M**, and **T**, and show that it achieves a strong accuracy-efficiency trade-off across classification, few-shot learning, part segmentation, and scene segmentation.
- We introduce **NetScore$^+$**, a deployability metric that incorporates latency and memory in addition to standard efficiency measures.

II. Related Work

Point cloud backbones span a broad range of design choices, from highly expressive architectures to models developed specifically for efficient deployment.

Shared MLP frameworks. PointNet [5] introduced the use of shared MLPs and symmetric pooling for point cloud processing, while PointNet++ [6] extended this idea with hierarchical grouping to better capture local structure at multiple scales. More recent residual MLP based models, such as PointMLP [7], substantially improve recognition accuracy, but this often comes with increased parameter count and inference cost. Even relatively compact supervised baselines in this family typically exceed 0.7M parameters and 0.9 GFLOPs at 1k input points [16], [17].

Graph and edge convolution networks. Graph based models explicitly model local relationships between points and often provide strong geometric reasoning. DGCNN [8], for example, recomputes kNN graphs at each layer and applies edge convolution to capture local interactions. While effective, this repeated neighborhood construction can become expensive as the number of points or network depth increases [18], [19], making deployment on edge devices more difficult.

Kernel point convolutions. Methods such as PointConv [20] and KPConv [9] extend convolution to irregular point sets through continuous or kernel based operators. These approaches are well suited for modeling local geometry, but their kernel parameterization and density compensation mechanisms can add noticeable runtime and memory overhead [21], [22].

Transformer and token mixing models. Transformer based point cloud models, including PCT [23], Point Transformer [10], and masked pretraining approaches such as Point BERT [11] and Point MAE [24], have achieved strong performance on several benchmarks. However, attention operations and large learned embeddings often increase latency and memory consumption [12], which can limit their practicality in resource constrained settings.

Analytic and minimal parameter pipelines. A separate line of work focuses on reducing or even removing learnable parameters. Methods such as NPNet [13], Point-NN [15], Point-LN [25], Point-GN [14], and PointHop++ [26] demonstrate that point cloud recognition can be performed with very small or nearly parameter free pipelines. These methods are highly efficient, but they generally lag behind stronger supervised baselines on more challenging benchmarks [27].

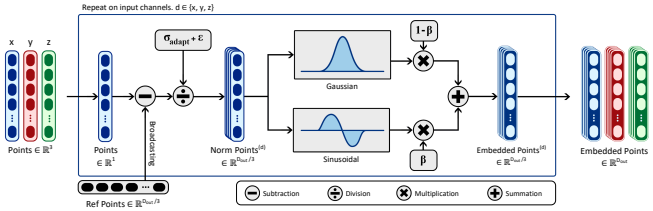

Fig. 3: NAPE: raw 3D coordinates mapped via fused Gaussian RBF and cosine bases with data-driven adaptive bandwidth, yielding a parameter-free geometry encoding.

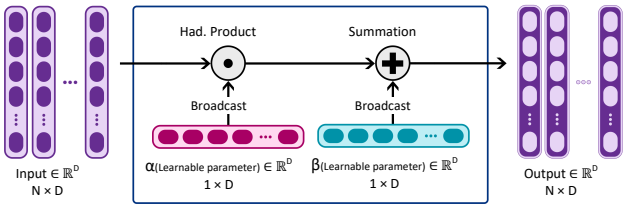

Fig. 4: GMU: per-channel affine recalibration of the NAPE output with only $2D$ learnable scalars.

III. Methodology

A. Overview

SLNet is a four-stage hierarchical backbone for 3D point cloud understanding. For object-level tasks (***SLNet-S***/M), the pipeline applies NAPE—a parameter-free geometric encoder—followed by GMU, FPS-based downsampling, parameter-free normalization, and shared residual MLPs. For scene-level segmentation (***SLNet-T***), NAPE is replaced with a learned linear projection, and the MLP stages are replaced with local Point Transformer attention. All three variants share the same hierarchical pipeline based on FPS, kNN grouping, local aggregation, and feature propagation; their key differences are summarized in Table I, and the overall architecture is shown in Figure 2.

TABLE I: SLNet variant comparison.

Variant	Params	GFLOPs	Front-end	Local op.
SLNet-S	0.14M	0.31	NAPE+GMU	Shared MLP
SLNet-M	0.55M	1.22	NAPE+GMU	Shared MLP
SLNet-T	2.5M	6.5	Lin. proj. (6ch)	Point Trans

B. Nonparametric Adaptive Point Embedding (NAPE)

NAPE is a fully parameter-free block that maps raw XYZ coordinates to a D-dimensional feature ($D=16$ for ***SLNet-S***; $D=32$ for ***SLNet-M***) through three components.

Global dispersion. Object scale is estimated as $\sigma_{\text{global}} = \frac{1}{3}(\sigma_x+\sigma_y+\sigma_z)$, the mean per-axis standard deviation across points—a moment-based statistic that is translation-invariant and less sensitive to point-wise noise.

Adaptive bandwidth. The kernel width is scaled by object size: $\sigma_{\text{adapt}} = \sigma_0(1+\sigma_{\text{global}})$, with base bandwidth $\sigma_0=0.4$ fixed across all experiments.

Basis blending. For each coordinate axis, Gaussian RBF and cosine bases are evaluated on a uniform grid $\mathbf{g}$ of $\lceil D/3 \rceil$ interior points in $(-1,1)$ (endpoints excluded):

$$\text{rbf} = \exp\!\left(-\frac{(x_i-\mathbf{g})^2}{2\,\sigma_{\text{adapt}}^2}\right), \quad \cos = \cos\!\left(\frac{x_i-\mathbf{g}}{\sigma_{\text{adapt}}}\right),$$

and blended via a sigmoid gate $\beta = \text{sigmoid}(\gamma(\sigma_{\text{global}} - b))$ ($\gamma=10$, $b=0.1$):

$$\mathbf{e}_i = \beta \cdot \text{rbf} + (1-\beta)\cdot \cos.$$

In practice, the Gaussian basis is more localized, while the cosine basis provides smoother responses over larger spatial ranges, with β shifting automatically between them based on cloud scale. Per-axis embeddings are concatenated across the three axes ($3\lceil D/3 \rceil$ total features), and a fixed index selection yields the final D-dimensional output. All hyperparameters are fixed across all datasets and model sizes.

C. Geometric Modulation Unit (GMU)

GMU applies a per-channel affine transformation after NAPE:

$$\mathbf{Y}_{b,d,n} = \alpha_d\, \mathbf{X}_{b,d,n} + \beta_d,$$

where $\alpha_d, \beta_d \in \mathbb{R}$ are learnable scalars ($2D$ parameters total: 32 for ***SLNet-S***, 64 for ***SLNet-M***). GMU is used where it is empirically beneficial and is omitted for ScanObjectNN based on ablation results.

D. Hierarchical Encoder (*SLNet-S/M*)

Sampling and grouping. Each of four stages uses FPS to select centroids and kNN to form local neighborhoods ($K=32$ for ModelNet40 and ShapeNet; $K=24$ for ScanObjectNN). Point counts halve per stage; channel dimensions expand by factors $[2,2,2,1]$.

Parameter-free normalization. Relative features are computed without learnable parameters:

$$\mathbf{H}^1_{ij} = [\mathbf{f}_{ij} \| \mathbf{x}_{ij}] - [\mathbf{f}_i \| \mathbf{x}_i],$$

providing a relational representation of local geometric context.

Light Residual Block (LRB). Features are refined via a shared residual MLP:

$$\mathbf{Y}_{ij} = \text{ReLU}\big(\mathbf{H}^3_{ij} + W_2\big(\text{ReLU}(\text{BN}(W_1\mathbf{H}^3_{ij}))\big)\big)\,,$$

with channel width ratio $r=0.25$ (with channel width ratio fixed to $r=0.25$ based on ablation results). Stage depths are $[1,1,2,1]$ for ModelNet40 and ShapeNet, and $[1,1,3,1]$ for ScanObjectNN. Neighbor features are then max-pooled per centroid before the next stage.

E. Task-Specific Heads

Classification (*SLNet-S/M*). Global adaptive max-pooling yields a fixed-length descriptor, fed to a two-layer MLP classifier.

Part segmentation (*SLNet-S/M*). A U-Net decoder restores resolution via four FP blocks with inverse-distance-weighted ($k=3$) interpolation and skip connections. Multi-scale GMP fusion concatenates pooled features from all five encoder

TABLE II: **ModelNet40 classification (supervised).** Metrics: overall accuracy (OA), mean class accuracy (mAcc), parameters (Param, M), FLOPs (G), memory (MB), latency (ms), and composite NetScore / NetScore$^+$. Our runs are 3-run averages.

Method	Paper results		Our runs		Param	FLOPs	Memory (MB)		Inference Time (ms) 3090		Inference Time (ms) Jetson		NetScore	NetScore$^+$	
	OA	mAcc	OA	mAcc	(M)	(G)	3090	Jetson	1024	2048	1024	2048		3090	Jetson
PointMLP [7]	**94.10**	**91.30**	**93.66**	**90.99**	13.24	15.67	86.68	105.80	4.18	4.23	64.61	65.64	55.69	42.87	36.48
APES (local) [28]	93.50	–	93.30	–	4.49	7.38	82.69	91.82	3.48	6.78	54.90	93.52	63.59	49.84	43.92
APES (global) [28]	93.80	–	93.23	–	4.49	5.49	82.69	91.82	2.50	4.89	37.28	65.97	64.86	51.83	45.95
PointNet++ (msg) [6]	–	–	92.51	89.87	1.75	4.00	99.87	108.99	4.56	5.92	209.96	218.54	70.21	56.35	48.32
DGCNN [8]	92.90	90.20	92.82	89.09	1.81	2.69	78.09	87.22	1.56	3.90	25.71	59.87	71.84	59.42	53.25
PointNet++ (ssg) [6]	90.70	–	92.31	89.65	1.48	0.86	25.93	35.15	2.26	2.73	171.04	181.25	77.59	68.34	58.57
CurveNet [29]	93.80	–	93.38	–	2.14	**0.33**	21.33	30.45	3.04	8.17	108.97	403.57	80.34	69.14	59.89
PointNet [5]	89.20	86.20	90.04	86.52	3.47	0.45	22.08	31.20	**0.19**	**0.35**	**3.85**	**5.40**	76.27	71.86	65.13
PointMLP (elite) [7]	93.60	90.90	93.28	**90.99**	0.72	0.91	**18.56**	**28.56**	1.18	1.22	25.28	25.99	80.64	**73.87**	**66.29**
SLNet-S	93.64	89.46	93.64	89.46	**0.14**	**0.31**	**11.49**	**21.30**	**0.74**	**0.76**	**17.31**	**18.04**	**92.42**	**87.71**	**79.50**
SLNet-M	**93.92**	**91.10**	**93.92**	**91.10**	**0.55**	1.22	23.97	33.00	1.39	1.41	29.48	30.26	**80.66**	73.01	65.67

TABLE III: **ModelNet-R classification (supervised).** Param, FLOPs, Mem, and Time match Table II and are omitted here.

Method	OA	mOAcc	NetScore	NetScore$^+$ 3090	NetScore$^+$ Jetson
PointMLP [7]	**95.33**	**94.30**	56.00	43.18	36.79
PointNet++ (msg) [6]	94.06	91.80	70.50	56.64	48.61
DGCNN [8]	94.03	92.64	72.06	59.65	53.47
PointNet++ (ssg) [6]	94.02	92.40	77.91	68.66	58.89
CurveNet [29]	94.12	92.65	80.48	69.27	60.03
PointNet [5]	91.39	88.79	76.53	72.12	65.39
SLNet-S	**94.53**	92.21	**92.59**	**87.87**	**79.66**
SLNet-M	94.81	**93.76**	**80.83**	**73.18**	**65.83**

levels with a class-label embedding and the final decoder output before per-point classification.

Semantic segmentation (*SLNet-T*). ***SLNet-T*** replaces NAPE with a learned linear projection ($6 \to 64$ channels, LayerNorm, ReLU) to accommodate XYZ+RGB inputs, followed by four encoder stages with channel dimensions $[64, 128, 256, 512]$ and local Point Transformer v1 attention [10]: subtraction-based weights $\mathbf{w} = \mathrm{MLP}(\mathbf{q} - \mathbf{k} + \mathbf{pe})$ (softmax over $k = [16, 16, 32, 64]$ neighbors) with a shared relative position encoding branch. GroupNorm enables stable training at batch size 1 on dense scene inputs. The decoder is a lightweight FP stack (dims $[256, 128, 64, 64]$, no GMP fusion), trained with weighted cross-entropy (inverse-square-root class weights, label smoothing $\epsilon = 0.1$) to address S3DIS class imbalance.

IV. Experiments

We evaluate ***SLNet*** on six benchmarks spanning object classification, part segmentation, few-shot learning, and large-scale semantic segmentation.

A. Setup

Object tasks (*SLNet-S*/M). Experiments were conducted on consumer GPUs (NVIDIA GTX 1080–RTX 3090) under Ubuntu 22.04 with CUDA 11.8. Inputs: 1024 points for classification/few-shot, 2048 points with normals for ShapeNetPart. ModelNet40/R and ScanObjectNN: 300 epochs, SGD (lr 0.1/0.01), cosine decay, EMA. ShapeNetPart: 350 epochs, AdamW (lr 0.003). FLOPs were computed with FVCore.

TABLE IV: Hardware and software used for efficiency benchmarking.

Device	Batch	Workers	Software stack	Memory
RTX 3090	32	6	Ubuntu 22.04, CUDA 11.8	25.3 GB
Jetson Orin Nano	4	4	JetPack 6.2.2, CUDA 12.6, Python 3.10	8 GB

Metrics include trainable parameters, GFLOPs, peak GPU memory, and average inference time per sample. Measurements follow the supplied routines for counting parameters, computing GFLOPs, measuring peak memory, and timing inference with warmup. Input sizes match experimental settings (Table IV).

Scene segmentation (*SLNet-T*). Trained on an RTX 5090 with AdamW ($\eta = 9 \times 10^{-4}$, $\lambda = 2 \times 10^{-3}$), cosine schedule with 15-epoch warm-up, mixed precision, and EMA ($\rho = 0.999$). Scene crops: $N = 16{,}384$ points, radius 2.5 m, 75 crops/room; 6D input (XYZ+RGB). Validation every 5 epochs via sliding-window inference (stride 0.75 m).

Edge profiling. All variants were benchmarked on a Jetson Orin Nano with 1024 input points and batch size 4; the hardware and software settings are summarized in Table IV.

B. Metrics

Classification: overall accuracy (OA), mean class accuracy (mAcc). Part segmentation: instance- and class-average IoU (ins-IoU, cls-IoU). Scene segmentation: mIoU, mAcc, OA. Efficiency: parameters p (M), FLOPs m (G), peak memory r (MB), latency t (ms).

NetScore and NetScore$^+$. To quantify accuracy–efficiency trade-offs as a single deployability metric, we adopt NetScore [30] and define:

$$\text{NetScore, NetScore}^+ = 20 \log_{10} \frac{a^2}{\sqrt{pm}\ \sqrt[4]{tr}^{\,\delta}}, \quad \delta = \begin{cases} 0 & \text{for NetScore,} \\ 1 & \text{for NetScore}^+ \end{cases} \tag{1}$$

where a is accuracy in percent (not fractional), p is number of parameters, m FLOPs, t inference time, and r memory footprint. Higher is better. Only methods with reproducible hardware metrics are reported.

TABLE V: **ScanObjectNN classification (supervised).** Metrics: OA, mAcc, Param (M), FLOPs (G), Mem (MB), Time (ms), and NetScore/NetScore$^+$.

Method	Paper results		Param	FLOPs	Memory (MB)		Inference Time (ms) 3090		Inference Time (ms) Jetson		NetScore	NetScore$^+$	
	OA	mAcc	(M)	(G)	3090	Jetson	1024	2048	1024	2048		3090	Jetson
PointNet++ (msg) [6]	–	–	1.74	4.00	99.84	108.97	4.49	5.83	211.28	213.84	–	–	–
CurveNet [29]	–	–	2.13	**0.33**	21.28	30.40	3.02	8.06	115.16	405.10	–	–	–
APES (global) [28]	–	–	4.49	5.49	82.67	91.79	2.50	4.88	37.31	66.15	–	–	–
APES (local) [28]	–	–	4.49	7.38	82.67	91.79	3.48	6.79	55.11	93.65	–	–	–
PointMLP [7]	**85.40**	**83.90**	13.24	15.67	86.68	99.80	4.12	4.19	64.98	67.51	54.09	41.29	34.95
DGCNN [8]	78.10	73.60	1.80	2.69	78.07	87.20	1.55	3.90	25.72	59.87	68.85	56.44	50.27
PointNet++ (ssg) [6]	77.90	75.40	1.47	0.86	25.91	35.03	2.23	2.73	175.69	180.70	74.66	65.42	55.66
PointNet [5]	68.20	63.40	3.47	0.45	22.05	31.18	**0.19**	**0.35**	**4.66**	**5.31**	71.45	67.04	60.35
PointMLP (elite) [7]	83.80	81.80	0.72	0.91	18.56	27.68	1.15	1.20	25.37	26.23	78.78	72.04	64.48
SLNet-S	83.45	81.45	**0.12**	**0.26**	**8.75**	**17.87**	**0.64**	**0.66**	**16.36**	**16.79**	**91.76**	**87.96**	**79.38**
SLNet-M	**84.25**	**82.86**	**0.48**	1.02	**18.28**	**27.37**	1.22	1.24	27.30	27.57	**80.12**	**73.34**	**65.73**

TABLE VI: **ShapeNetPart part segmentation (supervised).** Metrics: instance IoU (ins-IoU), class-average IoU (cls-IoU), Param (M), FLOPs (G), Mem (MB), Time (ms), and NetScore/NetScore$^+$.

Method	Paper results		Param	FLOPs (G)		Memory (MB) 3090		Memory (MB) Jetson		Inference Time (ms) 3090		Inference Time (ms) Jetson		NetScore	NetScore$^+$	
	ins-IoU	cls-IoU	(M)	1024	2048	1024	2048	1024	2048	1024	2048	1024	2048		3090	Jetson
PointNet++ (msg) [6]	–	–	1.74	4.76	4.82	99.83	103.84	107.95	111.96	5.04	6.64	211.25	210.46	–	–	–
APES (global) [28]	85.80	83.70	1.98	–	15.57	–	140.38	–	148.51	–	7.47	–	101.55	62.45	47.35	41.56
APES (local) [28]	85.60	83.10	1.98	–	18.37	–	140.38	–	148.51	–	9.38	–	128.43	61.69	46.09	40.29
CurveNet [29]	**86.60**	–	5.53	–	2.56	–	97.90	–	106.03	–	10.32	–	280.34	65.98	50.96	43.62
DGCNN [8]	85.20	82.30	1.46	2.34	4.96	77.38	149.14	86.50	158.27	**1.94**	5.29	**29.56**	64.81	68.62	54.13	48.56
PointMLP [7]	**86.10**	**84.60**	16.76	6.05	6.26	113.46	113.73	121.65	121.23	3.15	3.28	57.61	54.76	57.19	44.33	38.08
PointNet [5]	83.70	80.40	8.34	2.90	5.79	97.26	157.72	83.93	125.39	**0.49**	**1.02**	**8.60**	**11.54**	60.07	49.03	44.27
PointNet++ (ssg) [6]	85.10	81.90	**1.41**	**1.05**	**1.13**	**26.43**	58.19	**34.55**	**42.33**	2.67	3.35	174.82	190.38	**75.19**	**63.74**	**55.66**
SLNet-S	85.21	83.99	**1.24**	**0.83**	**0.91**	**21.99**	**38.12**	**30.09**	**46.24**	2.17	**2.49**	38.69	**40.88**	**76.70**	**66.81**	**60.31**
SLNet-M	85.53	**84.45**	1.90	2.25	2.33	40.15	**41.33**	48.27	49.46	3.39	3.72	58.59	60.66	70.81	59.88	53.43

C. Classification

ModelNet40. As shown in Table II, ***SLNet-S*** (0.14M, 0.31 G) achieves 93.64% OA, surpassing PointMLP-elite (93.28%) with 5× fewer parameters and attaining the highest NetScore (92.42) and NetScore$^+$ (87.71) among all evaluated methods. ***SLNet-M*** (0.55M, 1.22 G) reaches 93.92%, exceeding PointMLP in our runs (93.66%) at 24× fewer parameters and 13× fewer FLOPs, and matching it in mAcc (91.1%).

ModelNet-R. ModelNet-R is a refined version of the widely used ModelNet40 classification benchmark, with corrected labels, removal of near-2D and ambiguous samples, and improved class distinctions to yield a more reliable evaluation dataset [31]. Table III shows consistent gains on this re-annotated benchmark: ***SLNet-S*** reaches 94.53% and ***SLNet-M*** 94.81%, with NetScore and NetScore$^+$ rankings preserved.

ScanObjectNN (PB-T50-RS). This split includes background clutter, partial occlusion, random rotations, and point sparsity. ***SLNet-M*** achieves 84.25% OA with 28× fewer parameters and 15× fewer FLOPs than PointMLP (85.40%), while remaining within 1.15 percentage points in accuracy. ***SLNet-S*** reaches 83.45% OA and achieves stronger NetScore and NetScore$^+$ than PointMLP-elite (Table V).

D. Part Segmentation on ShapeNetPart

SLNet-S reaches 85.21% ins-IoU, and posts the highest NetScore$^+$ (66.81 / 60.31) of all evaluated methods (Table VI). ***SLNet-M*** attains 85.53% ins-IoU at only 3.72 ms per cloud (2048 points), 2.8× faster than CurveNet at just 1.1 pp lower IoU. Both variants demonstrate that the U-Net decoder with multi-scale GMP fusion preserves fine part boundaries while sustaining single-digit millisecond inference.

E. Semantic Segmentation on S3DIS

Table VII shows that ***SLNet-T*** achieves 58.2% mIoU, 85.3% OA, and 65.4% mAcc on S3DIS Area 5 with only 2.5M parameters and 6.5 GFLOPs. Although its absolute mIoU is lower than heavier transformer baselines such as PT (70.4%) and ConDAF (73.5%), ***SLNet-T*** attains a higher NetScore (58.5 vs. 57.5 for PT and 54.8 for FPT), indicating a stronger accuracy-efficiency trade-off under a much smaller model budget. This result reflects the design goal of ***SLNet-T***: prioritizing deployment efficiency under a ≤2.5M parameter budget, which is approximately 3× smaller than PT, 5× smaller than ConDAF, and 17× smaller than PT V3. Within this budget, replacing MLP stages with local attention improves mIoU by 14.4 percentage points over the MLP-only baseline (§IV-G), highlighting the importance of local relational modeling for scene-level understanding.

F. Few-Shot Classification

Table VIII reports performance on ModelNet40 under 5-way and 10-way episodic protocols. ***SLNet-M*** achieves 95.0% (5-way 20-shot) and **94.0%** (10-way 20-shot); ***SLNet-S*** reaches **93.5%** in the same 10-way 20-shot setting. Notably, in the 10-way 20-shot configuration both variants—despite being parametric—surpass all non-parametric

TABLE VII: **S3DIS Area 5 semantic segmentation**. SLNet-T (2.5M params) vs. transformer-based models. Despite lower mIoU, it achieves the highest NetScore (58.5), indicating the best accuracy per parameter.

Method	OA	mAcc	mIoU	Param (M)	FLOPs (G)	NetScore
PT	90.8	76.5	70.4	**7.8**	**5.6**	**57.5**
ST	**91.5**	78.1	72.0	–	–	–
PT V2	91.1	77.9	71.6	12.8	–	–
ConDAF	**91.6**	**78.9**	**73.5**	12.8	–	–
PT V3	–	–	**73.1**	42.6	–	–
FPT	**91.5**	**78.5**	72.2	10.9	8.3	54.8
SLNet-T	85.3	65.4	58.2	**2.5**	**6.5**	**58.5**

Results of prior methods are primarily taken from [32] and supplemented with values reported in the respective original papers when necessary.

TABLE VIII: **Few-shot classification (%) on ModelNet40.** 5-way/10-way, 10-shot and 20-shot. Baselines from Sharma et al. [33].

Method	5-way		10-way	
	10-shot	20-shot	10-shot	20-shot
Parametric Methods				
FoldingNet [34]	33.4	35.8	18.6	15.4
DGCNN [8]	31.6	40.8	19.9	16.9
PointNet [5]	52.0	57.8	46.6	35.2
PointNet++ [6]	38.5	42.4	23.0	18.8
3D-GAN [35]	55.8	65.8	40.3	48.4
PointCNN [36]	65.4	68.6	46.6	50.0
Non-Parametric Methods				
Point-NN [15]	88.8	90.9	79.9	84.9
Point-GN [37]	**90.7**	90.9	**81.6**	86.4
NPNet [13]	**92.0**	**93.2**	**82.5**	87.6
Our Method (Parametric)				
SLNet-S	84.0	89.0	75.5	**93.5**
SLNet-M	89.0	**95.0**	80.0	**94.0**

methods, including NPNet [13] (87.6%), improving by about 6 percentage points (SLNet-M: +6.4 pp; SLNet-S: +5.9 pp).

G. Ablation Study

Embedding and GMU placement (*SLNet-S*/M). Table IX compares embedding choices and GMU insertion points on ModelNet40. NAPE alone (without GMU) outperforms learned MLP, pure Gaussian, and pure cosine embeddings by about 0.9–1.1 percentage points, suggesting that adaptive basis blending provides a more effective geometric encoding than any single basis alone. Adding GMU with scale-then-shift ($\alpha\beta$) immediately after the embedding recovers a further 0.5 pp for both ***SLNet-S*** ($93.13 \rightarrow 93.64\%$) and ***SLNet-M*** ($93.44 \rightarrow 93.92\%$), while placing GMU after sampling & grouping or using the alternative $\beta\alpha$ order consistently underperforms; double GMU also fails to improve over single post-embedding placement.

Neighborhood size. Table X sweeps $K \in \{16, 24, 32, 64\}$. $K = 32$ is optimal for ModelNet40 (clean, uniform sampling), while $K = 24$ is optimal for ScanObjectNN (real-world clutter), consistent with the intuition that over-large neighborhoods on noisy data introduce irrelevant context. Performance degrades noticeably at $K = 64$ on both datasets (-0.75 pp on ***SLNet-M***/ModelNet40), so the default was tuned per dataset accordingly.

TABLE IX: Effect of embedding choice and GMU placement on ModelNet40 OA (%). NAPE with post-embedding $\alpha\beta$ modulation is optimal across both model sizes.

Embedding	GMU after Embedding	GMU after Sampling&Grouping	SLNet-S	SLNet-M
NAPE	—	—	93.13	93.44
NAPE	$\alpha\beta$	—	**93.64**	**93.92**
NAPE	—	$\alpha\beta$	93.27	93.52
NAPE	$\beta\alpha$	—	93.19	93.47
NAPE	—	$\beta\alpha$	92.63	92.88
NAPE	$\alpha\beta$	$\alpha\beta$	93.40	93.68
NAPE	$\beta\alpha$	$\beta\alpha$	93.23	93.51
mlp	$\alpha\beta$	—	92.71	92.93
mlp	—	$\alpha\beta$	92.49	92.59
mlp	$\alpha\beta$	$\alpha\beta$	92.60	92.78
gaussian	$\alpha\beta$	—	92.57	92.82
gaussian	—	$\alpha\beta$	92.31	92.74
gaussian	$\alpha\beta$	$\alpha\beta$	92.40	92.69
cosine	$\alpha\beta$	—	92.55	92.77
cosine	—	$\alpha\beta$	92.34	92.65
cosine	$\alpha\beta$	$\alpha\beta$	92.65	92.81

LRB channel width. Table XI evaluates the residual bottleneck ratio $r \in \{\div 8, \div 4, \div 2, 1\}$. $r = \div 4$ is the clear optimum on both datasets: $\div 8$ loses 1.6 pp on ***SLNet-S*** while saving only 0.03M parameters, and expanding to $\div 2$ or $r = 1$ adds parameters with no accuracy gain. This suggests that the $\div 4$ bottleneck provides a useful regularization effect without becoming a limiting capacity constraint.

Sampling strategy. FPS consistently outperforms random sampling (Table XII) by 0.3 pp on ModelNet40 and 0.6 pp on ScanObjectNN, the latter gap reflecting FPS's advantage in maintaining spatial coverage under occlusion and clutter.

SLNet-T encoder, loss, and training design. Table XIII reports 10 key runs across three groups.

Encoder architecture (§A). Replacing the MLP encoder with full local Point Transformer attention yields a +9.6 pp mIoU gain ($43.68 \rightarrow 53.24\%$), with the hybrid variant (attention only in the two coarsest stages) providing a midpoint at 48.79%. This large gap confirms that dense attention over local neighborhoods is essential for fine-grained scene understanding—shared MLPs lack the relational capacity to disambiguate adjacent categories such as *wall*, *column*, and *beam* in cluttered indoor scenes.

Loss function (§B). Switching from plain cross-entropy (B1, 55.47%) to inverse-square-root weighted CE (B2, 58.16%) yields +2.7 pp mIoU and +3.8 pp mAcc, directly addressing S3DIS's severe class imbalance. Focal loss ($\gamma = 2$, B3, 55.84%) is a closer competitor but still underperforms

TABLE X: Sensitivity to neighbourhood size K on ModelNet40 and ScanObjectNN. Optimal K differs across datasets due to point density and clutter.

Dataset	Model	16	24	32	64
Model Net40	SLNet-S	93.12	93.32	**93.64**	93.08
	SLNet-M	93.34	93.48	**93.92**	93.17
Scan Object	SLNet-S	81.99	**83.45**	83.38	82.13
	SLNet-M	82.86	**84.25**	83.58	81.83

TABLE XI: LRB bottleneck ratio ablation on ModelNet40 and ScanObjectNN (OA %, params M). ÷4 is the Pareto-optimal point.

Dataset	Model		÷8	÷4	÷2	1
ModelNet40	SLNet-S	Acc	92.07	**93.64**	93.51	93.49
		Param	0.11	0.14	0.20	0.31
	SLNet-M	Acc	92.34	**93.92**	93.79	93.65
		Param	0.44	0.55	0.77	1.20
ScanObjecNN	SLNet-S	Acc	81.23	**83.45**	83.28	83.00
		Param	0.09	0.12	0.19	0.31
	SLNet-M	Acc	82.72	**84.25**	84.11	83.82
		Param	0.35	0.48	0.73	1.23

WCE, suggesting that the static frequency-based reweighting is more stable than adaptive down-weighting for this distribution.

Training design (§C). EMA is the most impactful regulariser (-0.9 pp when removed; C1), while label smoothing ($\varepsilon = 0.1$) provides a marginal 0.1 pp benefit (C2). Halving neighbourhood size to $k = [8, 8, 16, 32]$ cuts FLOPs by 44% but costs 3.1 pp mIoU (C3), a poor trade-off at full model scale. Halving channel width (C4) reduces parameters by 75% (2.46 → 0.62 M) and FLOPs by 75% (6.49 → 1.64 G) at a cost of 5.5 pp mIoU, an attractive operating point for extremely constrained deployments.

Twelve additional post-optimisation experiments (§D–G: loss refinements, rare-class targeting, extended training, and LR/EMA schedules) consistently confirmed B2 as the robust global optimum. The closest challenger, G2 (EMA $\rho = 0.9999$, 57.54%), fell 0.62 pp short; combining OneCycle LR with slow EMA (G3) caused training divergence due to the mismatch between the high-LR exploration phase and the slow-moving EMA checkpoint.

a) Qualitative.: Gradient-based saliency maps from the NAPE block show more localized responses on semantic parts (e.g., chair legs and lamp shades), whereas DGCNN features appear more diffuse (Figure 5). This qualitative behavior is consistent with the quantitative trends observed in our experiments. This behavior is consistent across channel widths (16 and 32) and the maps remain robust to small input perturbations, indicating stable geometric reasoning at early layers.

V. Conclusion

We presented ***SLNet***, a lightweight hierarchical backbone for 3D point cloud understanding built on two simple ideas: parameter-free geometric encoding and minimal learned

TABLE XII: FPS vs. random sampling on ModelNet40 and ScanObjectNN (OA %). Larger gains on real-world data.

Dataset	Model	Random Sampling	FPS
ModelNet40	SLNet-S	93.31	**93.64**
	SLNet-M	93.68	**93.92**
ScanObjectNN	SLNet-S	82.83	**83.45**
	SLNet-M	83.86	**84.25**

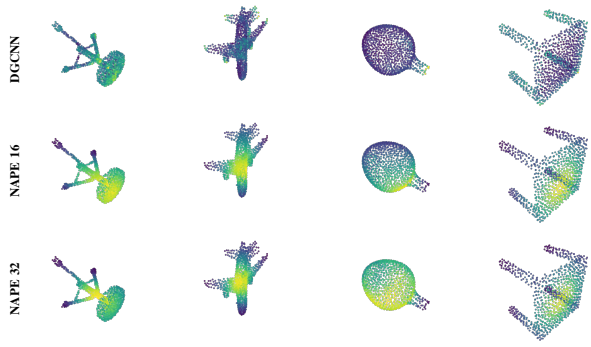

Fig. 5: Qualitative comparison of saliency on ModelNet40 samples: top row—DGCNN stage-2 EdgeConv gradient saliency; middle and bottom—NAPE gradient saliency (channel widths 16 and 32). NAPE yields sharper, semantically focused responses despite operating at the input layer.

modulation. **NAPE** captures raw XYZ geometry through an input-adaptive combination of Gaussian RBF and cosine bases without introducing learnable parameters, while **GMU** provides a lightweight per-channel affine recalibration with only $2D$ learnable scalars. Across object classification, few-shot learning, part segmentation, and scene segmentation, these design choices lead to a strong balance between recognition performance and deployment efficiency.

For object-level tasks, ***SLNet-S*** and ***SLNet-M*** achieve competitive accuracy with substantially fewer parameters and lower compute cost than many stronger supervised baselines. In few-shot classification, both variants also outperform non-parametric baselines in the 10-way 20-shot setting without pretraining. For large-scale indoor scene

TABLE XIII: SLNet-T ablation on S3DIS Area 5. Three groups: encoder architecture (§A), loss function (§B), and training-design components (§C). Each group's fixed settings and abbreviations follow note [a]. **Bold**: best mIoU per group. ⋆: global best (B2). †: proposed encoder baseline.

Configuration				*Accuracy (%)*			*Cost*	
ID	Description	Enc	Loss	mIoU	OA	mAcc	Param (M)	FLOPs (G)
§A *Encoder Architecture* (loss: F+L)								
A1	MLP aggregation	M	F+L	43.68	77.88	51.84	1.16	2.74
A2	Hybrid attn (st. 3–4)	H	F+L	48.79	80.10	56.02	2.39	5.60
A3†	Full attention	A	F+L	**53.24**	84.08	60.03	2.46	6.49
§B *Loss Function* (enc: A; base: A3†, 53.24 %)								
B1	Plain CE, $\varepsilon = 0$	A	CE	55.47	85.19	61.63	2.46	6.49
B2⋆	Weighted CE ($1/\sqrt{f_c}$)	A	WCE	**58.16**	85.30	65.40	2.46	6.49
B3	Focal, $\gamma = 2$	A	Foc	55.84	85.28	61.93	2.46	6.49
§C *Training Design* (enc: A, loss: F+L; base: A3†, 53.24 %)								
C1	No EMA	A	F+L	52.30	83.53	58.78	2.46	6.49
C2	No label smoothing	A	F+L	**53.30**	83.36	60.25	2.46	6.49
C3	$k = [8, 8, 16, 32]$	A	F+L	50.17	82.37	57.58	2.46	3.66
C4	Half-channel width[b]	A	F+L	47.73	81.08	54.57	**0.62**	**1.64**

[a] **Common settings:** $N = 16{,}384$ pts/crop; crop radius $r = 2.5$ m; $k = [16, 16, 32, 64]$; encoder dims $[64, 128, 256, 512]$ (C4: $[32, 64, 128, 256]$); decoder dims $[256, 128, 64, 64]$ (C4: $[128, 64, 32, 32]$); AdamW $\eta = 9\times10^{-4}$, $\lambda = 2\times10^{-3}$; cosine LR, EMA $\rho = 0.999$; label smoothing $\varepsilon = 0.1$ (C2: 0); batch 48/16; seed 42. **Enc:** M = MLP, H = Hybrid, A = Attn. **Loss:** CE; WCE = weighted CE; Foc = focal; F+L = focal+Lovász. [b] C4 bold cost entries indicate best within §C on those metrics.

segmentation, ***SLNet-T*** extends the same design philosophy with local Point Transformer attention and reaches a favorable efficiency-accuracy trade-off under a much smaller model budget than prior transformer-based alternatives. We further introduced **NetScore**$^{+}$, which incorporates latency and memory alongside standard efficiency measures to better reflect deployment-oriented performance. Overall, our results suggest that compact point cloud models can remain highly competitive when geometric encoding, feature modulation, and local aggregation are designed explicitly for efficiency.

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.

[2] 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.

[3] 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.

[4] E. Dumic and L. A. da Silva Cruz, "Three-dimensional point cloud applications, datasets, and compression methodologies for remote sensing: A meta-survey," *Sensors*, vol. 25, no. 6, p. 1660, 2025.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun, "Point transformer," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 16 259–16 268.

[11] X. Yu, L. Tang, Y. Rao, T. Huang, J. Zhou, and J. Lu, "Point-bert: Pre-training 3d point cloud transformers with masked point modeling," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 19 313–19 322.

[12] X. Wu, L. Jiang, P.-S. Wang, Z. Liu, X. Liu, Y. Qiao, W. Ouyang, T. He, and H. Zhao, "Point transformer v3: Simpler faster stronger," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2024, pp. 4840–4851.

[13] M. Saeid, A. Salarpour, P. MohajerAnsari, and M. D. Pesé, "Npnet: A non-parametric network with adaptive gaussian-fourier positional encoding for 3d classification and segmentation," *arXiv preprint arXiv:2602.00542*, 2026.

[14] 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.

[15] 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.

[16] X. Zhou, D. Liang, W. Xu, X. Zhu, Y. Xu, Z. Zou, and X. Bai, "Dynamic adapter meets prompt tuning: Parameter-efficient transfer learning for point cloud analysis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 14 707–14 717.

[17] H. Sun, Y. Wang, W. Chen, H. Deng, and D. Li, "Parameter-efficient prompt learning for 3d point cloud understanding," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 9478–9486.

[18] H. Wang, W.-L. Zhao, X. Zeng, and J. Yang, "Fast k-nn graph construction by gpu based nn-descent," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 1929–1938.

[19] J. Hou, S. Liu, Y. Zhang, and H. Qin, "Graph construction with flexible nodes for traffic demand prediction," *arXiv preprint arXiv:2403.00276*, 2024.

[20] 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.

[21] Z. Su, P. S. Tan, J. Chow, J. Wu, Y. Cheong, and Y.-H. Wang, "Dv-convnet: Fully convolutional deep learning on point clouds with dynamic voxelization and 3d group convolution," *arXiv preprint arXiv:2009.02918*, 2020.

[22] M. Atzmon, H. Maron, and Y. Lipman, "Point convolutional neural networks by extension operators," *arXiv preprint arXiv:1803.10091*, 2018.

[23] 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.

[24] Y. Pang, W. Wang, F. E. Tay, W. Liu, Y. Tian, and L. Yuan, "Masked autoencoders for point cloud self-supervised learning," in *European conference on computer vision*. Springer, 2022, pp. 604–621.

[25] 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.

[26] M. Zhang, Y. Wang, P. Kadam, S. Liu, and C.-C. J. Kuo, "Pointhop++: A lightweight learning model on point sets for 3d classification," in *2020 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2020, pp. 3319–3323.

[27] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, "Deep learning for 3d point clouds: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 12, pp. 4338–4364, 2020.

[28] 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.

[29] 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.

[30] A. Wong, "Netscore: Towards universal metrics for large-scale performance analysis of deep neural networks for practical usage," *arXiv preprint arXiv:1806.05512*, 2018.

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

[32] Y. He, H. Yu, M. Feng, T. Chen, Z. Li, A. Ulhaq, S. Anwar, and A. S. Mian, "Pointdiffuse: a dual-conditional diffusion model for enhanced point cloud semantic segmentation," *arXiv preprint arXiv:2503.06094*, 2025.

[33] C. Sharma and M. Kaul, "Self-supervised few-shot learning on point clouds," *Advances in Neural Information Processing Systems*, vol. 33, pp. 7212–7221, 2020.

[34] 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.

[35] 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.

[36] 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.

[37] 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.