Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Parameter-Efffcient Topology-Guided Cross-Scale Adapter for Point Cloud Learning

Rongqian Yang

rqyang@scut.edu.cn

School of Electronic and Information Engineering, South ChinaUniversity of Technology

Lingxiang Zheng

School of Electronic and Information Engineering, South ChinaUniversity of Technology

Article

Keywords:

Posted Date: February 25th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-8886209/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: There is **NO** Competing Interest.

Parameter-Efficient Topology-Guided Cross-Scale Adapter for Point Cloud Learning

Lingxiang Zheng[1] and Rongqian Yang[1*]

[1]School of Electronic and Information Engineering, South China University of Technology, Guangzhou, China.

*Corresponding author(s). E-mail(s): rqyang@scut.edu.cn;
Contributing authors: zlx@aimooe.com;

Abstract

Recently, large-scale pre-training has become a dominant paradigm for improving point cloud representations and enabling strong transfer to downstream three-dimensional (3D) tasks. However, adapting large pre-trained point-cloud transformers in practice still often relies on full fine-tuning, which is storage-intensive and computationally demanding when multiple tasks or domains must be supported. Moreover, for real scans, the main obstacle is not only the parameter budget but also the *topology shift* induced by density variation, occlusion, missing regions, and background clutter, which corrupts local neighborhoods and makes token-level adaptation unstable. To address these issues, we propose a novel parameter-efficient fine-tuning (PEFT) framework for point clouds, called **TGCS** (**T**opology-**G**uided **C**ross-**S**cale adapter). TGCS freezes the pre-trained backbone and introduces a lightweight, trainable tuning branch that performs topology-conditioned residual calibration across transformer blocks. The core idea is built on two observations: (1) under a frozen backbone, feature-space prompts and adapters may be misled by unreliable semantic tokens when neighborhood topology is distorted, and (2) topology corruption is inherently multi-scale, so effective tuning should couple explicit topology cues with cross-scale context. Concretely, TGCS combines Cross-Scale Token Mixing (CS-Mixing), Saliency-Aware Token Gating (SA-Gating), and a Topology-Guided Cross-Scale Adapter (TG-Adapter) that conditions residual updates on multi-scale topology descriptors computed from token anchors, including density and dispersion statistics as well as eigenvalue-derived local shape cues. Extensive experiments on ScanObjectNN, ModelNet40, and ShapeNetPart demonstrate that TGCS consistently improves the accuracy-efficiency trade-off across MAE-style and GPT-style backbones. Notably, with Point-MAE, TGCS tunes only **0.6**M parameters (**2.68%**) yet improves the hardest ScanObjectNN setting

PB_T50_RS from **85.18%** to **88.03%**. With the stronger PointGPT-L backbone, TGCS achieves **98.97%**, **97.42%**, and **95.00%** on OBJ_BG, OBJ_ONLY, and PB_T50_RS, respectively while tuning only **2.2**M parameters, establishing the state-of-the-art performance under an efficient fine-tuning regime. TGCS also yields stable gains in few-shot classification and preserves competitive part-segmentation mIoU with a compact tunable budget, validating topology-guided cross-scale conditioning as a practical solution for resource-efficient point cloud adaptation.

3D point clouds have become a fundamental modality for perceiving and understanding the physical world[1], supporting a wide spectrum of applications such as autonomous driving, robotic manipulation, embodied AI, AR/VR, and industrial inspection[1, 2]. Compared with images, point clouds are inherently irregular, sparse, and orderless, and their semantics are tightly coupled with local geometric structures induced by sampling, occlusion, clutter, and sensor noise[6, 7]. Recently, transformer-based pre-training paradigms have substantially improved point cloud representations and enabled strong transfer to downstream tasks[8–13]. By learning generalizable 3D priors from large-scale unlabeled data, pre-trained point cloud transformers reduce the reliance on expensive annotations and have achieved impressive performance across classification and segmentation benchmarks. However, transferring large pre-trained 3D models in a practical way remains challenging: fully fine-tuning the entire backbone for each downstream task is computationally expensive, storage-inefficient, and often unnecessary when only limited labeled data are available[14].

This has motivated growing interest in parameter-efficient fine-tuning (PEFT)[16], which adapts a frozen pre-trained backbone by updating only a small set of tunable parameters such as prompts, adapters, or low-rank updates[17–20]. In natural language processing (NLP) and two-dimensional (2D) vision, PEFT has become a standard recipe for efficient adaptation[21, 22], largely because it provides a favorable trade-off between accuracy, storage, and training efficiency[17–20, 23–26]. It also offers practical benefits for continual deployment that a single frozen backbone can be retained, and multiple small task-specific modules can be attached [17, 20], greatly reducing the cost of maintaining and distributing model variants [14, 16]. Yet, directly porting these PEFT designs to 3D point clouds is often suboptimal, especially on real scanned datasets[27–30]. In point cloud learning, the difficulty is not merely to reduce tunable parameters, but to decide what signals should guide a lightweight update when the backbone remains frozen. In practice, performance is strongly influenced by neighborhood-level geometric and topological patterns, including density variation, anisotropy, surface incompleteness, and local structural diversity. These patterns are far less structured than grid-aligned 2D images and tend to shift significantly across domains.

Recent works have made encouraging progress toward PEFT for point cloud transformers through dynamic prompting, hybrid prompt-adapter designs, and domain-aware tuning strategies. These methods demonstrate that a small number of trainable

parameters can indeed adapt pre-trained point cloud models, and they provide important insights into the design space of 3D PEFT. Nonetheless, a central limitation remains that most PEFT modules still operate primarily in the token and feature space, while the key discrepancies that hurt transfer originate from geometry and topology shifts in the input space. Real scans often contain missing regions, varying point density, and cluttered background, creating multiple modes within a semantic class[31, 32]. Under such shifts, a frozen pre-trained transformer can produce miscalibrated token representations, lightweight tuning that ignores explicit topology cues may become sensitive to sampling changes and yield unstable results, particularly under few-shot protocols where variance across folds is common[9, 27]. In other words, although existing PEFT approaches reduce trainable parameters, they may not fully address the geometry-rooted failure modes that dominate real-scan transfer.

A key challenge is that point clouds express structure through local neighborhoods, and these neighborhoods are inherently unstable under domain shift and sampling randomness[3, 4]. For the same semantic object, different sensors or acquisition conditions may lead to drastically different point distributions that local density can vary by orders of magnitude, partial observation and occlusion create missing regions, and background clutter introduces outlier neighborhoods[1, 31]. Such changes affect not only low-level geometry but also the tokenization process itself, which subsequently alters how a transformer aggregates information across layers[8, 9]. If a PEFT module treats all tokens uniformly or relies on feature-space updates alone, it may fail to adapt the representation to these topology-altered neighborhoods. This is particularly evident on challenging real-scan benchmarks where the hardest variants are designed to stress robustness against background and partiality. Therefore, recent work suggests that inducing and preserving relational structure during pre-training can be important for transfer[15].

Moreover, effective tuning for point clouds should be inherently cross-scale[10, 33–36]. Transformer backbones learn hierarchical representations: early layers emphasize local geometry, while deeper layers encode higher-level semantics[33, 34, 37]. When the downstream distribution differs from the pre-training distribution, the mapping between local structures and global semantics can change, leading to cross-scale misalignment under a frozen hierarchy. A lightweight tuning module that only perturbs a subset of layers or injects a single-scale signal may not be sufficient to reconcile this mismatch[17, 20]. Instead, it is desirable to inject multi-scale context and propagate geometry corrections consistently across layers so that the model can maintain stable local-to-global reasoning. At the same time, few-shot adaptation amplifies overfitting: with scarce labels, a PEFT module may latch onto spurious correlations, and performance can fluctuate sharply across different folds and sampling randomness. These considerations suggest that PEFT for point clouds should be explicitly guided by topology and couple cross-scale information flow, so that lightweight updates can recalibrate frozen features in a robust and stable manner rather than simply adding generic capacity.

Therefore, we propose a parameter-efficient **T**opology-**G**uided **C**ross-**S**cale adapter for point cloud learning (**TGCS**), a new PEFT framework that adapts large frozen pre-trained point cloud transformers with minimal tunable parameters by injecting

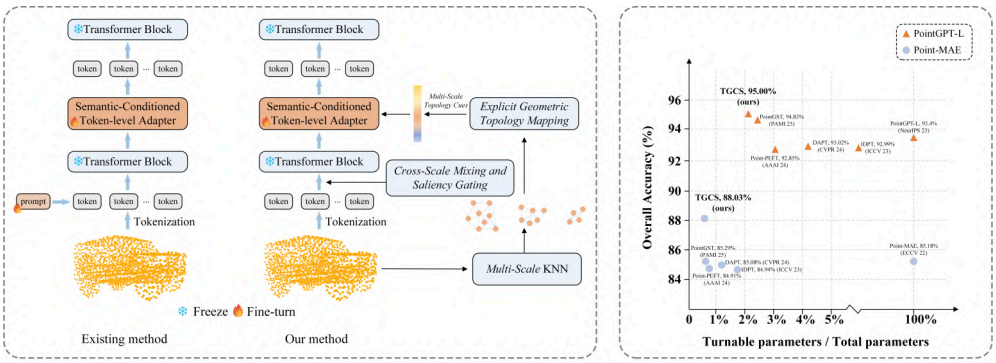
topology-conditioned, cross-scale residual calibration into the transformer flow. TGCS is designed around a simple but important premise that when the backbone is frozen, the tunable module should behave as a geometry-grounded calibrator that corrects topology-induced discrepancies, rather than a generic feature perturbation that risks overfitting under scarce supervision. Concretely, TGCS is built around three complementary components that jointly improve robustness and stability while keeping the parameter budget small.

First, TGCS introduces cross-scale token aggregation to enrich patch tokens with local-to-regional geometric cues before they enter the frozen transformer. Different from prior PEFT designs that mainly inject prompts and adapters in the token space and thus implicitly assume reliable local neighborhoods, TGCS explicitly strengthens tokens with geometry-grounded cross-scale context before they are processed by the frozen backbone (Fig. 1(a)). By summarizing neighborhood contexts at multiple spatial scales and injecting the aggregated context as a lightweight residual, TGCS provides an efficient way to enhance local structure modeling without modifying backbone weights. This cross-scale aggregation also reduces the sensitivity to sampling variation. Even when fine-scale neighborhoods fluctuate due to resampling or partial scans, the aggregated multi-scale context offers a more stable structural prior that is consistent with the underlying object geometry[10, 35, 36].

Second, TGCS employs token saliency gating to account for unreliable patches caused by occlusion, clutter, or sensor artifacts. Instead of assuming all patch tokens are equally informative, the saliency gate predicts soft token-wise reliability weights and interpolates between original and gated representations with a learnable scaling factor. This mechanism is particularly beneficial in few-shot regimes since it acts as a lightweight regularizer that suppresses noisy evidence and encourages model to focus on meaningful structures, improving generalization when labeled data are scarce.

Third and most importantly, TGCS proposes a topology-guided cross-scale adapter with cross-layer conditioning as the core tuning module. Unlike generic bottleneck adapters, TGCS explicitly extracts multi-scale topology descriptors from 3D neighborhoods, such as density-related statistics, local variation cues, and covariance-spectrum features capturing anisotropy and surface characteristics, and uses them as conditioning signals to modulate adapter updates. This design results in geometry-aware adaptation, where the tuning direction is guided by the local topological state instead of being determined purely by token features[38]. Furthermore, TGCS leverages cross-layer cached representations as additional conditions to stabilize adaptation across the transformer hierarchy. Intuitively, earlier layers often contain geometry-sensitive information, while later layers encode semantics. Conditioning on cross-layer cached representations provides additional context for the adapter and improves the stability of feature calibration under a frozen backbone.

TGCS is general and can be plugged into different pre-trained backbones with minimal engineering effort. Empirically, TGCS consistently achieves a strong parameter–performance trade-off across tasks and backbones. For example, a particularly notable result is on ScanObjectNN with a Point-MAE backbone that TGCS fine-tunes only 0.6M parameters (2.68%) yet achieves 88.03% accuracy on the hardest variant PB_T50_RS, demonstrating that topology-guided cross-scale calibration can effectively handle severe real-scan distribution shifts.



(a) The comparison between existing PEFT methods and our method.

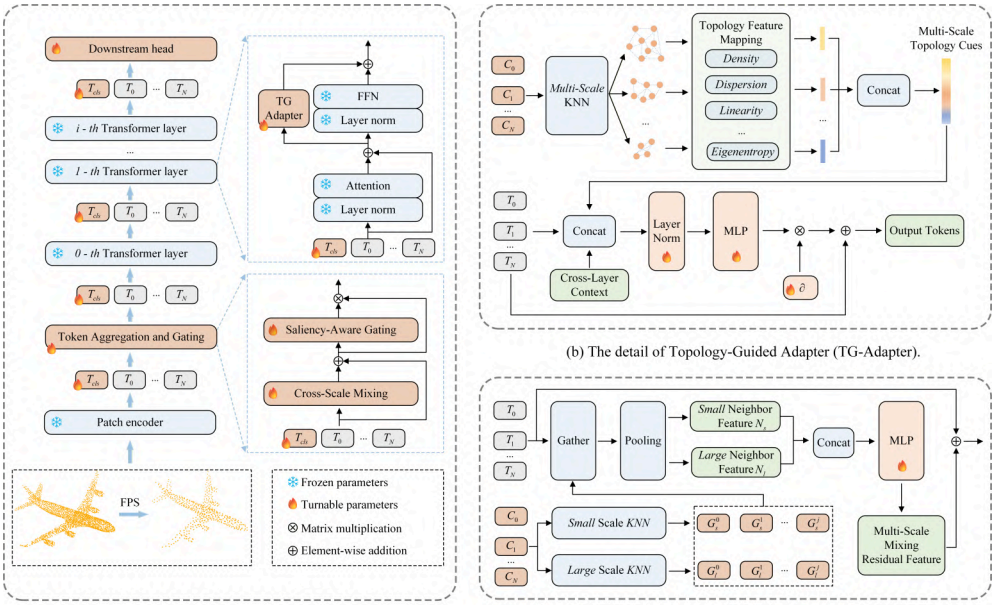
(b) Results on the hardest variant part of ScanObjectNN dataset.

Fig. 1: Comparison of PEFT paradigms for point clouds. (a) Prior PEFT mainly adapts in semantic-conditioned token and feature space, which can be unstable under real-scan *topology shifts*. TGCS performs *topology-guided* residual calibration with cross-scale context. (b) Accuracy-efficiency trade-off on ScanObjectNN. Overall accuracy (OA) (%) versus trainable parameters ratio (%), showing TGCS achieves higher accuracy with fewer trainable parameters.

In summary, our contributions are three-fold. We propose TGCS, a parameter-efficient framework for adapting frozen pre-trained point cloud transformers via topology-guided and cross-scale conditioning. We introduce a topology-guided cross-scale adapter with cross-layer conditioning, enabling stable residual calibration under domain shift while preserving pre-trained semantics. We also design two lightweight yet effective mechanisms consist of token cross-scale mixing and saliency gating to inject multi-scale context and reliability-aware tuning, improving robustness and reducing overfitting in low-data regimes. Extensive experiments on classification and segmentation benchmarks demonstrate that TGCS consistently improves the parameter-performance trade-off and yields strong, robust transfer across backbones and tasks.

Results

This section evaluates TGCS on 3D classification, few-shot learning, and part segmentation. Our analysis is structured around two complementary questions that (i) how different adaptation strategies behave under a fixed pre-trained backbone, and (ii) how TGCS compares with representative point-cloud PEFT baselines under matched parameter budgets. The central objective is to verify whether explicitly incorporating multi-scale topology cues leads to consistent and practical improvements, especially under topology-shifted real-scan conditions.



(a) The overall of the presented TGCS.

(b) The detail of Topology-Guided Adapter (TG-Adapter).

(c) The detail of Cross-Scale Token Mixing (CS-Mixing).

Fig. 2: The overview of the presented TGCS . (a) Overall framework. TGCS adapts a frozen pre-trained backbone with three lightweight trainable components consist of CS-Mixing, SA-Gating, and TG-Adapter. (b) Topology-Guided Adapter inside each block conditions residual updates on multi-scale topology descriptors and shallow cross-layer context. (c) Cross-Scale Token Mixing aggregates neighbors at two KNN scales on token anchors and injects a scaled residual feature.

3D Classification

Comparisons of different fine-tuning strategies. Table 1 compares different adaptation strategies on ScanObjectNN and ModelNet40. On real-scan data, topology shift is prominent and typically manifests as density variation, clutter, and partial observations. A clear pattern is that TGCS provides consistent improvements over lightweight tuning baselines while remaining competitive with full fine-tuning, despite updating only a small trainable branch.

On ScanObjectNN, TGCS improves the baseline of each backbone across all three settings. For the Point-MAE backbone[9], TGCS increases accuracy from 90.02% to 92.60% on `OBJ_BG` (**+2.58%**), from 88.29% to 92.08% on `OBJ_ONLY` (**+3.79%**), and from 85.18% to 88.03% on `PB_T50_RS` (**+2.85%**). The gains are particularly notable on `OBJ_ONLY` and `PB_T50_RS`, where occlusion and density variation substantially perturb local neighborhoods. On a stronger GPT-style backbone, *i.e.*, PointGPT-L[13], TGCS further pushes performance to 98.97%, 97.42%, and 95.00% on `OBJ_BG`, `OBJ_ONLY`, and `PB_T50_RS`, respectively, indicating that topology-guided adaptation benefits both MAE-style and GPT-style representations rather than being tied to a specific pre-training paradigm.

Pre-trained model	Fine-tuning strategy	Reference	Params. (M)	FLOPs (G)	ScanObjectNN			ModelNet40
					OBJ_BG	OBJ_ONLY	PB_T50_RS	OA (%)
Point-MAE [9] (ECCV 22)	Fully fine-tune	–	22.1 (100%)	4.76	90.02	88.29	85.18	93.2
	IDPT [27]	ICCV 23	1.7 (7.69%)	7.10	91.22(+1.20)	90.02(+1.73)	84.94(-0.24)	93.3(+0.1)
	Point-PEFT [29]	AAAI 24	0.7 (3.13%)	7.61	89.67(-0.35)	88.98(+0.69)	84.91(-0.27)	93.3(+0.1)
	DAPT [28]	CVPR 24	1.1 (4.97%)	4.96	90.88(+0.86)	90.19(+1.90)	85.08(-0.10)	93.5(+0.3)
	PointGST [38]	PAMI 25	0.6 (2.77%)	4.81	91.74(+1.72)	90.19(+1.90)	85.29(+0.11)	93.5(+0.3)
	TGCS (ours)	–	**0.6 (2.68%)**	**4.78**	**92.60**(+2.58)	**92.08**(+3.79)	**88.03**(+2.85)	**93.6**(+0.4)
ACT [11] (ICLR 23)	Fully fine-tune	–	22.1 (100%)	4.76	93.29	91.91	88.21	93.7
	IDPT [27]	ICCV 23	1.7 (7.69%)	7.10	93.12(-0.17)	92.26(+0.35)	87.65(-0.56)	94.0(+0.3)
	Point-PEFT [29]	AAAI 24	0.7 (3.13%)	7.61	90.36(-2.93)	90.02(-1.89)	85.74(-2.47)	93.4(-0.3)
	DAPT [28]	CVPR 24	1.1 (4.97%)	4.96	92.60(-0.69)	91.57(-0.34)	87.54(-0.67)	93.2(-0.5)
	PointGST [38]	PAMI 25	0.6 (2.77%)	4.81	93.46(+0.17)	**92.60**(+0.69)	88.27(+0.06)	**94.0**(+0.3)
	TGCS (ours)	–	**0.6 (2.68%)**	**4.78**	**93.80**(+0.51)	**92.60**(+0.69)	**88.48**(+0.27)	93.7(+0.0)
ReCon [12] (ICML 23)	Fully fine-tune	–	22.1 (100%)	4.76	94.32	92.77	90.01	92.5
	IDPT [27]	ICCV 23	1.7 (7.69%)	7.10	93.29(-1.03)	91.57(-1.20)	87.27(-2.74)	93.4(+0.9)
	Point-PEFT [29]	AAAI 24	0.7 (3.13%)	7.61	91.91(-2.41)	90.19(-2.58)	86.36(-3.65)	93.3(+0.8)
	DAPT [28]	CVPR 24	1.1 (4.97%)	4.96	94.32(-0.00)	92.43(-0.34)	89.38(-0.63)	93.5(+1.0)
	PointGST [38]	PAMI 25	0.6 (2.77%)	4.81	**94.49**(+0.17)	**92.94**(+0.17)	89.49(-0.52)	93.6(+1.1)
	TGCS (ours)	–	**0.6 (2.68%)**	**4.78**	**94.49**(+0.17)	**92.94**(+0.17)	**89.76**(-0.25)	**93.7**(+1.2)
PointGPT-L [13] (NeurIPS 23)	Fully fine-tune	–	360.5 (100%)	67.71	97.2	96.6	93.4	94.1
	IDPT [27]	ICCV 23	10.0 (2.77%)	75.19	98.11(+0.91)	96.04(-0.56)	92.99(-0.41)	93.4(-0.7)
	Point-PEFT [29]	AAAI 24	3.1 (0.86%)	73.62	96.39(-0.81)	94.66(-1.94)	92.85(-0.55)	93.5(-0.6)
	DAPT [28]	CVPR 24	4.2 (1.17%)	71.64	98.11(+0.91)	96.21(-0.39)	93.02(-0.38)	94.2(+0.1)
	PointGST [38]	PAMI 25	2.4 (0.67%)	67.95	**98.97**(+1.77)	**97.59**(+0.99)	94.83(+1.43)	94.8(+0.7)
	TGCS (ours)	–	**2.2 (0.61%)**	**67.83**	**98.97**(+1.77)	97.42(+0.82)	**95.00**(+1.60)	**95.1**(+1.0)

Table 1: Classification accuracy without voting on real-scanned ScanObjectNN and synthetic ModelNet40 point clouds. We report trainable parameters (M), FLOPs (G), and overall accuracy (%) on ScanObjectNN and ModelNet40.

Beyond Point-MAE and PointGPT-L, the same conclusion holds for other MAE-family pre-trained encoders. ReCon and ACT provide complementary pre-training priors. ReCon emphasizes reconstruction-oriented geometric consistency, while ACT strengthens token modeling under masked reconstruction objectives[11, 12]. On ACT, TGCS yields consistent gains across ScanObjectNN variants and achieves strong robustness among the compared lightweight methods. Specifically, TGCS reaches 93.80% on `OBJ_BG`, improving over the ACT baseline (93.29%) by **+0.51%**, and also surpassing IDPT (93.12%), DAPT (92.60%), and Point-PEFT (90.36%). On `OBJ_ONLY`, TGCS attains 92.60% (**+0.69%**), matching the strongest competitor (PointGST) and exceeding IDPT (92.26%) and DAPT (91.57%). On the hardest `PB_T50_RS`, TGCS improves to 88.48% (**+0.27%**), outperforming IDPT (87.65%) and DAPT (87.54%), and also remaining above the full fine-tuning reference (88.21%). These margins support the interpretation that topology-guided conditioning remains effective even when the backbone priors differ, and the advantage becomes more visible when neighborhood reliability is degraded by occlusion and density shift.

On ReCon, TGCS is consistently competitive and often the strongest among lightweight tuning baselines. It achieves 94.49% on `OBJ_BG` and 92.94% on `OBJ_ONLY`, improving upon the ReCon baseline (94.32%/92.77%) by **+0.17%** in both cases and matching PointGST, while surpassing IDPT (93.29%/91.57%) and Point-PEFT (91.91%/90.19%). On `PB_T50_RS`, TGCS attains 89.76%, which is only 0.25% below full fine-tuning (90.01%) while still improving over IDPT (87.27%), DAPT (89.38%),

Method	Reference	Params. (M)	ScanObjectNN		
			OBJ_BG	OBJ_ONLY	PB_T50_RS
Supervised Learning Only					
PointNet [57]	CVPR 17	3.5	73.3	79.2	68.0
PointNet++ [3]	NeurIPS 17	1.5	82.3	84.3	77.9
DGCNN [4]	TOG 19	1.8	82.8	86.2	78.1
MVTN [70]	ICCV 21	11.2	-	-	82.8
3D-GCN [71]	TPAMI 22	-	-	-	-
RepSurf-U [69]	CVPR 22	1.5	-	-	84.3
PointNeXt [58]	NeurIPS 22	1.4	-	-	87.7
PTv2 [33]	NeurIPS 22	12.8	-	-	-
PointMLP [68]	ICLR 22	13.2	-	-	85.4
PointMeta [61]	CVPR 23	-	-	-	87.9
ADS [72]	ICCV 23	-	-	-	87.5
X-3D [59]	CVPR 24	5.4	-	-	90.7
GPSFormer [60]	ECCV 24	2.4	-	-	**95.0**
Self-Supervised Representation Learning (Full fine-tuning)					
OcCo [73]	ICCV 21	22.1	84.85	85.54	78.79
Point-BERT [8]	CVPR 22	22.1	87.43	88.12	83.07
MaskPoint [67]	ECCV 22	22.1	89.70	89.30	84.60
Point-MAE [9]	ECCV 22	22.1	90.02	88.29	85.18
Point-M2AE [10]	NeurIPS 22	15.3	91.22	88.81	86.43
ACT [11]	ICLR 23	22.1	93.29	91.91	88.21
ReCon [12]	ICML 23	43.6	94.15	93.12	89.73
PointGPT-L [13]	NeurIPS 23	360.5	97.2	96.6	93.4
Point-FEMAE [74]	AAAI 24	27.4	95.18	93.29	90.22
P2P++ [49]	TPAMI 24	16.1	-	-	90.3
PointDif [75]	CVPR 24	-	93.29	91.91	87.61
PointMamba [76]	NeurIPS 24	12.3	94.32	92.60	89.31
MH-PH [77]	ECCV 24	-	97.4	96.8	93.8
Self-Supervised Representation Learning (Efficient fine-tuning)					
PointGPT-L [13] w/ TGCS (ours)	-	**2.2**	**98.97**	**97.42**	**95.00**

Table 2: Classification results without voting on three variants of ScanObjectNN dataset, and we report the number of trainable parameters (M) and overall accuracy (%).

and PointGST (89.49%). This suggests that under severe topology disruption, explicitly conditioning on multi-scale topology cues provides a more reliable adaptation signal than relying solely on intermediate token semantics, while preserving the parameter-efficiency of lightweight tuning.

On ModelNet40, where synthetic objects are cleaner and topology shift is milder, TGCS still delivers consistent improvements. For Point-MAE, TGCS raises accuracy from 93.2% to 93.6% (**+0.4%**), and for PointGPT-L, from 94.1% to 95.1% (**+1.0%**). A similar positive trend is also observed on ACT and ReCon: for ACT, TGCS reaches 93.7% (on par with full fine-tuning), and for ReCon, TGCS improves from 92.5% to

93.7% (**+1.2%**). Although the margins are naturally smaller than those on ScanObjectNN, these results indicate that topology-guided conditioning is not only beneficial for real-scan robustness, but can also regularize parameter-efficient adaptation and improve feature calibration without modifying pre-trained backbone weights.

Overall, TGCS shows consistent gains across diverse pre-training priors, with the largest benefits emerging under the hardest real-scan setting where topology shift is most pronounced.

Compared with the state-of-the-art methods. Table 2 positions TGCS in a broader landscape that spans three regimes: (i) supervised learning from scratch, (ii) self-supervised pretraining followed by full fine-tuning, and (iii) self-supervised pretraining with parameter-efficient adaptation. We highlight three observations that clarify where TGCS contributes and why its gains are practically relevant on ScanObjectNN.

(1) Pre-training dominates supervised-only training on real scans. Across all three ScanObjectNN variants, the supervised-only baselines generally trail behind models that leverage self-supervised representation learning, particularly on `PB_T50_RS` where topology shift is amplified by occlusion, missing regions, and density variation[3, 4, 57]. This trend indicates that large-scale pre-training supplies transferable geometric priors that are hard to recover from ScanObjectNN labels alone. Notably, even strong supervised-only approaches that report competitive performance on `PB_T50_RS` (e.g., PointNeXt, PointMeta, X-3D, and GPSFormer) typically rely on carefully designed architectures and still do not guarantee uniformly strong results across `OBJ_BG` and `OBJ_ONLY`[58–61]. In contrast, self-supervised pre-trained backbones (e.g., Point-MAE, ACT, Recon, and PointGPT-L) consistently reach a higher accuracy regime on all variants[9, 11–13], reinforcing that ScanObjectNN is primarily a robustness benchmark where pre-trained structural priors are pivotal.

(2) Parameter-efficient adaptation is competitive only when the tuning signal is well grounded. While full fine-tuning can exploit the entire parameter space of a pre-trained model, it is not always the most attractive option when deployment demands per-task efficiency. However, efficient tuning is not guaranteed to retain the robustness of the pre-trained model. On real scans, the adaptation module must cope with neighborhood corruption and topology drift. Table 2 suggests that the strongest performance is achieved when adaptation is driven by reliable structural information rather than relying exclusively on intermediate semantic tokens[38]. TGCS follows this principle by explicitly extracting multi-scale topology cues on token anchors and using them to guide residual updates, which directly targets the source of domain shift in ScanObjectNN. This is particularly important for `PB_T50_RS`, where purely feature-driven updates are more likely to be misled by incomplete or cluttered observations.

(3) TGCS achieves the state-of-the-art accuracy with the minimum trainable budget. With PointGPT-L as the backbone, TGCS reaches **98.97%**, **97.42%**, and **95.00%** on `OBJ_BG`, `OBJ_ONLY`, and `PB_T50_RS` while tuning only **2.2M** parameters. This result is notable from two complementary perspectives. First, it improves upon the full fine-tuning counterpart of PointGPT-L (97.2%, 96.6%, and 93.4%), indicating that the gains are not simply attributable to increasing optimization

Methods	Reference	5-way		10-way	
		10-shot	20-shot	10-shot	20-shot
Point-MAE [9] (baseline)	ECCV 22	96.3±2.5	97.8±1.8	92.6±4.1	95.0±3.0
+ IDPT [27]	ICCV 23	97.3±2.1	97.9±1.1	92.8±4.1	95.4±2.9
+ Point-PEFT [29]	AAAI 24	95.5±2.9	97.6±1.7	91.7±4.3	94.7±3.0
+ DAPT [28]	CVPR 24	96.8±1.8	98.0±1.0	93.0±3.5	95.5±3.2
+ PointGST [38]	PAMI 25	98.0±1.8	98.3±0.9	93.7±4.0	95.7±2.4
+ TGCS (ours)	-	**98.1±2.1**	**98.5±1.0**	**93.9±2.9**	**96.1±1.3**
ACT [11] (baseline)	ICLR 23	96.8±2.3	98.0±1.4	93.3±4.0	95.6±2.8
+ IDPT [27]	ICCV 23	96.7±2.6	98.2±0.9	92.4±4.5	95.5±3.0
+ Point-PEFT [29]	AAAI 24	95.4±2.7	97.7±1.7	91.8±5.1	94.9±3.0
+ DAPT [28]	CVPR 24	94.4±3.9	97.3±2.4	90.6±6.1	95.1±3.7
+ PointGST [38]	PAMI 25	97.2±1.9	98.6±1.5	92.8±4.0	95.5±3.1
+ TGCS (ours)	-	**97.5±2.0**	**98.7±1.3**	**93.7±4.0**	**95.5±3.0**
ReCon [12] (baseline)	ICML 23	97.3±1.9	98.9±1.2	93.3±3.9	95.8±3.0
+ IDPT [27]	ICCV 23	96.8±2.2	98.6±0.8	92.7±3.8	95.9±3.2
+ Point-PEFT [29]	AAAI 24	95.4±2.6	97.7±1.4	91.5±5.0	95.2±3.4
+ DAPT [28]	CVPR 24	95.6±3.4	97.3±2.0	91.9±4.9	94.5±3.2
+ PointGST [38]	PAMI 25	96.9±2.2	98.7±0.9	92.9±3.8	95.8±2.8
+ TGCS (ours)	-	**97.1±2.2**	**98.8±1.0**	**93.2±3.8**	**95.9±2.9**
PointGPT-L [13] (baseline)	NeurIPS 23	98.0±1.9	99.0±1.0	94.1±3.3	96.1±2.8
+ IDPT [27]	ICCV 23	96.8±2.0	98.3±1.4	95.3±2.5	96.5±2.5
+ Point-PEFT [29]	AAAI 24	95.7±2.5	98.0±1.8	94.8±2.7	96.3±2.4
+ DAPT [28]	CVPR 24	97.2±3.4	98.4±1.3	95.9±2.7	97.3±2.4
+ PointGST [38]	PAMI 25	97.4±2.0	99.6±0.5	96.3±3.6	97.6±2.9
+ TGCS (ours)	-	**99.0±1.1**	**99.7±0.5**	**97.7±1.9**	**98.3±2.3**

Table 3: Few-shot classification on ModelNet40. Overall accuracy (%) ± the standard deviation (%) without voting is reported.

freedom, but to *better-conditioned* adaptation under topology shift. Second, TGCS attains the best reported performance on `PB_T50_RS` (95.00%), matching the strongest supervised-only entry in the table (GPSFormer at 95.0%) while simultaneously achieving near-saturated accuracy on the other two variants. These comparisons suggest that TGCS offers a favorable accuracy-efficiency trade-off. It preserves the deployment advantages of PEFT while reaching or exceeding the accuracy of much heavier fine-tuning baselines, especially in the regime where topology disruption is most severe.

Few-shot Learning

Table 3 reports few-shot classification on ModelNet40 under 5-way/10-way and 10-shot/20-shot settings[56]. Across all backbones, TGCS consistently ranks among the strongest PEFT variants, and the advantage is more evident when the task is harder

(e.g., 10-way) or the backbone is stronger (PointGPT-L). Compared with IDPT, Point-PEFT, DAPT, and PointGST, TGCS either improves the mean accuracy or matches the best result with comparable variance.

On Point-MAE, TGCS improves the baseline from 96.3% to 98.1% in 5-way 10-shot and from 92.6% to 93.9% in 10-way 10-shot, while also increasing 5-way 20-shot (97.8% $\rightarrow$ 98.5%) and 10-way 20-shot (95.0% $\rightarrow$ 96.1%). TGCS is comparable to PointGST and can be marginally better on Point-MAE (e.g., 98.1% vs. 98.0% and 96.1% vs. 95.7%). Meanwhile, Point-PEFT shows non-negligible regressions in several settings, indicating that few-shot PEFT depends critically on the conditioning signal. Similar trends hold for ACT and ReCon that TGCS remains competitive in easier regimes and is more reliable in harder regimes, without introducing noticeable instability.

Remarkably, TGCS exhibits a clearer advantage on PointGPT-L, reaching $99.0 \pm 1.1\%$ and $97.7 \pm 1.9\%$ for 5-way/10-way 10-shot, and $99.7 \pm 0.5\%$ and $98.3 \pm 2.3\%$ for 20-shot. Relative to PointGST, TGCS improves the mean by +1.6% in 5-way 10-shot (99.0% vs. 97.4%) and by +1.4% in 10-way 10-shot (97.7% vs. 96.3%), while also reducing the standard deviation in these harder settings, suggesting more stable adaptation under limited supervision.

Few-shot PEFT is prone to overfitting because the adaptation module must generalize from scarce labels while the backbone remains fixed[18–20]. TGCS alleviates this issue by conditioning residual updates on explicit multi-scale topology cues, providing a geometry-grounded signal that complements semantic tokens and is less sensitive to episode-specific noise. The consistent improvements, especially under 10-way settings and on PointGPT-L, support the effectiveness of topology-guided conditioning for low-data adaptation.

Part Segmentation

Table 4 reports ShapeNetPart part segmentation with class/instance mIoU and trainable parameters. Across MAE-style backbones, full fine-tuning reaches strong accuracy (e.g., Point-MAE: 84.19%/86.1%, ACT: 84.66%/86.1%, ReCon: 84.52%/86.1%) but requires 27.06M trainable parameters. TGCS reduces the trainable budget to 5.56M while keeping competitive segmentation quality. Notably, TGCS achieves 84.15% class mIoU and 85.9% instance mIoU on ACT, and matches the best instance mIoU (85.9%) across MAE-family settings, indicating that topology-guided residual updates preserve dense geometric sensitivity with a compact tuning branch.

For the GPT-style backbone, PointGPT-L full tuning attains 84.80%/86.6% with 339.39M parameters, whereas TGCS uses 31.70M and maintains 84.01%/85.9%. Although a gap to full tuning remains, TGCS offers a favorable accuracy-efficiency trade-off for large backbones. Overall, segmentation is stricter than classification because it depends on fine-grained local cues[3, 62]. The stable mIoU under TGCS suggests that explicit multi-scale topology conditioning helps retain boundary- and part-level discrimination under limited tunable capacity.

Methods	Reference	Params. (M)	Cls. mIoU (%)	Inst. mIoU (%)
Point-MAE [9] (baseline)	ECCV 22	27.06	84.19	86.1
+ IDPT [27]	ICCV 23	5.69	83.79	85.7
+ Point-PEFT [29]	AAAI 24	5.62	83.20	85.2
+ DAPT [28]	CVPR 24	5.65	**84.01**	85.7
+ PointGST [38]	PAMI 25	5.59	83.81	85.8
+ TGCS (ours)	–	**5.56**	83.87	**85.9**
ACT [11] (baseline)	ICLR 23	27.06	84.66	86.1
+ IDPT [27]	ICCV 23	5.69	83.82	**85.9**
+ Point-PEFT [29]	AAAI 24	5.62	81.25	84.4
+ DAPT [28]	CVPR 24	5.65	83.44	85.5
+ PointGST [38]	PAMI 25	5.59	84.04	85.8
+ TGCS (ours)	–	**5.56**	**84.15**	**85.9**
ReCon [12] (baseline)	ICML 23	27.06	84.52	86.1
+ IDPT [27]	ICCV 23	5.69	83.66	85.7
+ Point-PEFT [29]	AAAI 24	5.62	83.10	85.1
+ DAPT [28]	CVPR 24	5.65	83.87	85.7
+ PointGST [38]	PAMI 25	5.59	83.98	**85.8**
+ TGCS (ours)	–	**5.56**	**84.07**	85.7
PointGPT-L [13] (baseline)	NeurIPS 23	339.39	84.80	86.6
+ IDPT [27]	ICCV 23	33.17	82.91	85.1
+ Point-PEFT [29]	AAAI 24	31.94	82.34	84.9
+ DAPT [28]	CVPR 24	33.85	83.03	85.2
+ PointGST [38]	PAMI 25	31.85	83.91	85.8
+ TGCS (ours)	–	**31.70**	**84.01**	**85.9**

Table 4: Part segmentation on the ShapeNetPart. The mIoU for all classes (Cls.) and for all instances (Inst.) are reported without voting. Params. represents the trainable parameters.

Compared with PEFT Methods from Other Domains

PEFT has become a broadly applicable adaptation recipe across domains[66]. In this section we focus on representative PEFT paradigms from NLP and 2D vision for comparison. Table 5 compares TGCS with representative PEFT methods originally designed for NLP and 2D vision, as well as recent 3D-specific PEFT approaches, on the hardest ScanObjectNN setting (`PB_T50_RS`)[17, 19, 20, 23–26]. This variant exhibits the strongest topology disruption, making it a stress test for whether a PEFT module can adapt the representation without relying on extensive parameter updates.

The first observation is that directly transferring general-purpose PEFT mechanisms yields limited gains for point clouds. With Point-MAE as the backbone, linear probing already drops to 75.99%, indicating that the target-domain gap can not be bridged by a shallow classifier alone. Classical NLP-oriented tuning methods including Adapter (83.93%), Prefix tuning (77.72%), BitFit (82.62%), and LoRA (81.74%),

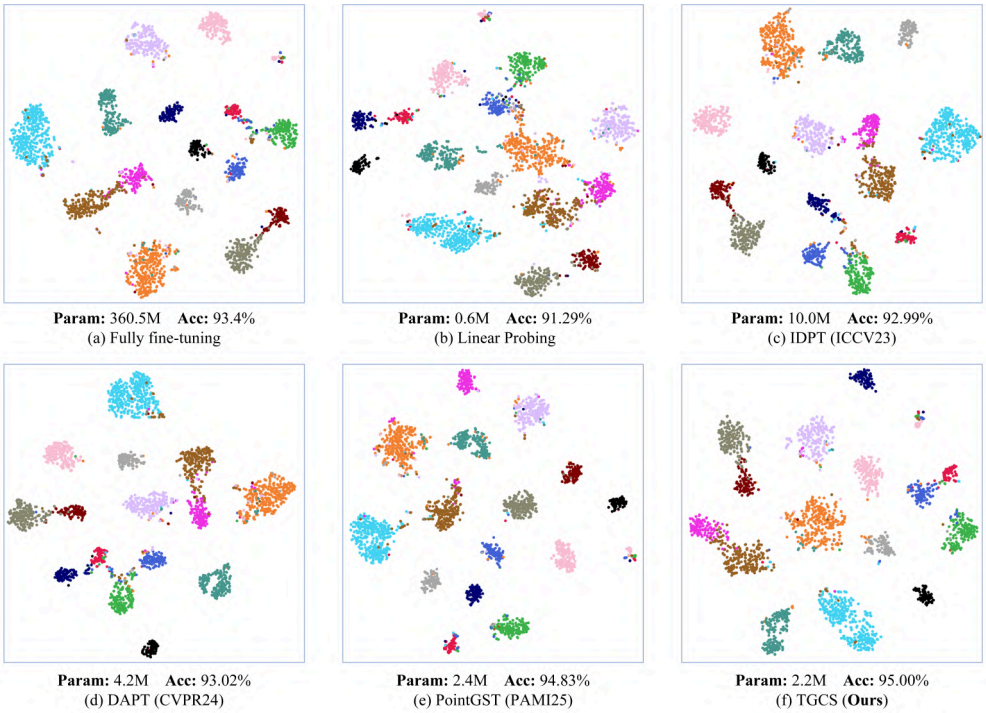


Fig. 3: The t-SNE visualization on ScanObjectNN `PB_T50_RS` dataset, along with trainable parameters and overall accuracy under the pre-trained PointGPT-L with different tuning strategies.

remain below the full fine-tuning baseline (85.18%). More recent language-domain variants such as DEPT (79.70%) and FourierFT (78.57%) show similar behavior. Likewise, 2D vision PEFT approaches (e.g., VPT-Deep 81.09%, Side-Adapter 83.45%, SSF 82.58%, FacT 78.76%, Bi-AdaptFormer 83.66%, SCT 80.40%) improve over linear probing but still fall short of recovering the backbone's fine-tuning capability[24–26, 63–65]. This gap suggests that, for point-cloud transfer under severe scan artifacts, simply injecting small trainable branches in token space is insufficient when the conditioning signal is dominated by intermediate semantic activations that may be unstable under topology shift.

In contrast, 3D-specific PEFT methods are substantially more competitive. IDPT (84.94%) and DAPT (85.08%) narrow the gap to full fine-tuning by tailoring the tuning mechanism to point-cloud transformers, while PointGST (85.29%) further improves the result with a compact parameter budget[27, 28, 38]. TGCS achieves 88.03% with only 0.6M trainable parameters, outperforming all cross-domain PEFT baselines by a clear margin and also surpassing the strongest 3D PEFT competitor in this table (PointGST) by +2.74%. Notably, this improvement is obtained with the same parameter scale as PointGST, indicating that the gain is not driven by larger tunable capacity but by a more informative adaptation signal.

Method	Reference	Design for	Params. (M)	PB_T50_RS
Point-MAE [9] (baseline)	ECCV 22	-	22.1	85.18
Linear probing	-	-	0.3	75.99
+ Adapter [17]	ICML 19	NLP	0.9	83.93
+ Prefix tuning [19]	ACL 21	NLP	0.7	77.72
+ BitFit [23]	ACL 21	NLP	0.3	82.62
+ LoRA [20]	ICLR 22	NLP	0.9	81.74
+ DEPT [78]	ICLR 24	NLP	0.3	79.70
+ FourierFT [79]	ICML 24	NLP	0.3	78.57
+ VPT-Deep [24]	ECCV 22	2D	0.4	81.09
+ Side-Adapter [25]	NeurIPS 22	2D	0.9	83.45
+ SSF [26]	NeurIPS 22	2D	0.4	82.58
+ FacT [63]	AAAI 23	2D	0.5	78.76
+ Bi-AdaptFormer [64]	ICCV 23	2D	0.4	83.66
+ SCT [65]	IJCV 24	2D	0.3	80.40
+ IDPT [27]	ICCV 23	3D	1.7	84.94
+ DAPT [28]	CVPR 24	3D	1.1	85.08
+ PointGST [38]	PAMI 25	3D	0.6	85.29
+ TGCS (ours)	-	3D	**0.6**	**88.03**

Table 5: Comparisons of Parameter-Efficient Fine-Tuning (PEFT) methods from NLP and 2D vision on the hardest variant of ScanObjectNN. Overall accuracy (%) without voting is reported. Params. is the trainable parameters.

The results highlight the central advantage of TGCS that it conditions parameter-efficient residual updates on explicit multi-scale topology cues computed from token anchors. Under `PB_T50_RS`, neighborhood relations are precisely what becomes unreliable, and purely semantic token-driven updates can be misled by distorted local context. By extracting density and dispersion statistics together with eigenvalue-derived local shape cues across multiple neighborhood scales, TGCS provides a geometry-grounded signal that remains meaningful when sampling becomes non-uniform and parts are missing.

Overall, Table 5 demonstrates that adapting PEFT methods across domains is non-trivial for point clouds, especially in the most challenging real-scan regime. TGCS offers a practical solution by aligning the PEFT conditioning with the dominant form of domain shift in 3D scans, achieving strong robustness gains under a strict parameter budget.

t-SNE Visualization

Fig. 3 shows the t-SNE projections of test-set features on ScanObjectNN `PB_T50_RS` using PointGPT-L under different tuning strategies. Linear probing yields substantial inter-class overlap, indicating limited separability when the backbone is frozen. Spatial token-space PEFT baselines, *i.e.*, IDPT and DAPT, reduce overlap compared with linear probing but still exhibit noticeable cross-cluster mixing. In comparison, structure-aware PEFT methods including PointGST and TGCS produce more separable manifolds with fewer inter-class ambiguities. Notably, TGCS achieves the highest accuracy (95.00%) with a smaller trainable budget (2.2M) than competing PEFT baselines, and its embedding exhibits clearer category separation, consistent with the quantitative gains under severe topology shift.

Topology-Aware Token Response Visualization

Fig. 4 compares token-response heatmaps of full fine-tuning and TGCS on ScanObjectNN test samples, where the boxed regions highlight category-relevant geometric parts for qualitative comparison. The heatmaps are produced by aggregating multi-layer attention signals into token-level scores and projecting them back onto the input points for visualization. Across the shown instances, TGCS yields responses that are more concentrated on category-relevant geometric parts highlighted by the boxes. In representative cases, the activation patterns are spatially consistent with class-discriminative geometry. For Bag, higher responses tend to concentrate around the opening structures, whereas for Bin and Box the model assigns stronger weights to large planar faces together with their boundary and corner transitions. For Chair, the emphasized regions shift to structural junctions, particularly around the seat–back connection and supporting elements. For Sink and Toilet, the responses are more localized to contour-rich parts and attachment-related areas, indicating sensitivity to functional geometry rather than diffuse surface regions. In comparison, full fine-tuning exhibits more diffuse responses on these samples. This indicates that TGCS encourages tokens to attend to structurally informative regions and maintains a more consistent alignment between token responses and object topology.

Ablation Study

We conduct ablation studies on the challenging ScanObjectNN `PB_T50_RS` setting and report overall accuracy (OA). Unless otherwise stated, we keep the backbone, training schedule, and evaluation protocol identical across variants, so that the performance differences can be attributed to the ablated design factors.

The Effect of Each Component

Table 6 summarizes the contribution of each component in TGCS. As references, linear probing achieves 75.99% OA, while full fine-tuning reaches 85.18%. Introducing **CS-Mixing** alone improves OA to 82.51%, indicating that multi-scale token aggregation provides a stronger representation than a purely frozen backbone under linear probing, yet it still falls short of full fine-tuning. The **SA-Gating** mechanism by

CS-Mixing	SA-Gating	TG-Adapter	PB_T50_RS
Full fine-tuning			85.18
Linear Probing			75.99
✓	-	-	82.51
-	✓	-	78.83
✓	✓	-	84.18
-	-	✓	86.85
✓	✓	✓	**88.03**

Table 6: The effect of each component in TGCS. We enable CS-Mixing, SA-Gating, and TG-Adapter individually and jointly. Overall accuracy (%) on ScanObjectNN `PB_T50_RS` is reported without voting.

itself yields 78.83%, suggesting that gating without complementary structural guidance offers limited benefit and may be insufficient to reshape class-separable features. Notably, enabling only the **TG-Adapter** achieves 86.85%, surpassing full fine-tuning by 1.67%, which indicates that topology-guided adaptation can effectively exploit geometric cues while remaining parameter efficient.

We further examine component interactions. Combining MS-Aggregation and Gating increases OA to 84.18%, which is higher than either component alone, implying that aggregation provides useful multi-scale signals that gating can selectively propagate. However, this combination still underperforms full fine-tuning, highlighting that spatial mixing alone is not enough. Finally, integrating all three components yields the best OA of **88.03%**. Compared to the TG-Adapter-only variant (86.85%), the additional gain suggests that MS-Aggregation supplies richer cross-scale context, while Gating helps regulate the injected signals to avoid over-smoothing or noisy updates. Overall, Table 6 supports that the three components are complementary that TG-Adapter provides the main performance lift, and MS-Aggregation plus Gating further refine the adaptation process.

The Effect of Inserted Layers

We then study how the insertion depth impacts performance, as shown in Table 7. Injecting TGCS only into shallow layers (1→6) achieves 87.47% OA, while inserting only into deep layers (7→12) yields 87.23%. Both settings perform competitively, indicating that TGCS can be effective at different abstraction levels. The slight advantage of shallow insertion suggests that early-stage tokens, which preserve stronger geometric locality, may benefit more from topology-guided cross-scale refinement. In contrast, deeper layers encode more semantic and task-specific patterns, where topology guidance still helps but with slightly reduced marginal gains.

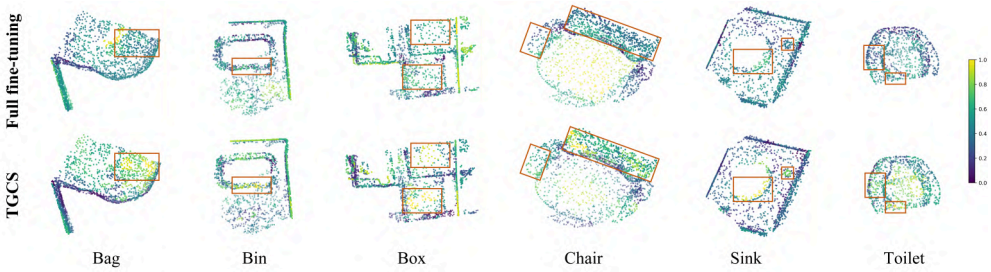


Fig. 4: Token-response visualization on ScanObjectNN test samples. Responses are obtained by aggregating multi-layer attention into token scores and projecting them back to points.

Layers	PB_T50_RS
1→6	87.47
7→12	87.23
1→12	**88.03**

Table 7: The effect of insertion depth. TGCS is inserted into different layer ranges and overall accuracy (%) is reported on ScanObjectNN `PB_T50_RS` without voting.

Importantly, inserting TGCS into all layers (1→12) obtains the best OA of **88.03%**. This result implies that the benefits are not localized to a particular stage. Instead, TGCS contributes cumulatively across the network hierarchy. A plausible explanation is that early layers improve geometric alignment and structural consistency of token interactions, which provides a better foundation for later layers to form more discriminative semantics. Therefore, we adopt the all-layer insertion as the default design for TGCS.

The Effect of Inserted Module

Next, we evaluate where to place TGCS within a transformer block (Table 8). Inserting TGCS into the attention branch achieves 86.92%, while inserting it into the FFN branch yields the best OA of **88.03%**. Applying TGCS to the entire transformer layer gives 87.68%. These results suggest that the FFN placement is the most effective adaptation point in our setting.

Position	PB_T50_RS
Attention	86.92
FFN	**88.03**
Entire transformer layer	87.68

Table 8: The effect of insertion position within a transformer block. We place TGCS in the attention branch, FFN branch, or the entire layer and report overall accuracy (%) on ScanObjectNN `PB_T50_RS` without voting.

Topology scale set	Param. (M)	PB_T50_RS
$(8, 16, 24, 48)$	2.07	94.59
$(4, 8, 16, 24, 48)$	2.08	94.80
$(2, 4, 8, 12, 16, 24, 32, 48)$	2.09	**95.00**
$(2, 4, 8, 12, 16, 24, 32, 48, 64)$	2.10	94.48

Table 9: The effect of topology scale set $\mathcal{K}$. We vary the multi-scale KNN neighborhoods used for topology descriptors and report overall accuracy (%) and trainable parameters (M) on ScanObjectNN `PB_T50_RS` without voting.

One interpretation is that the attention branch already performs token interaction and routing, and directly modifying it may interfere with the backbone's established attention patterns under limited trainable parameters. In contrast, the FFN branch is responsible for per-token transformation and feature refinement. Injecting topology-guided cross-scale information at this stage can adjust token representations without overly perturbing global routing. The slightly lower performance of adapting the entire block may stem from distributing limited capacity across multiple submodules, leading to less targeted updates. Consequently, we insert TGCS into the FFN by default.

The Effect of Topology Scale

Finally, we investigate the effect of topology scale configurations on PointGPT-L (Table 9). Using a coarse set of scales $(8, 16, 24, 48)$ achieves 94.59% OA. Adding a finer scale to obtain $(4, 8, 16, 24, 48)$ improves performance to 94.80%, indicating that incorporating additional local granularity can enhance discriminability. The configuration

$(2, 4, 8, 12, 16, 24, 32, 48)$ yields the best OA of **95.00%** with 2.09M parameters, suggesting that a moderately dense multi-scale design provides the most effective balance between capturing fine local structures and maintaining stable global context.

However, further extending the scale set to include 64 decreases OA to 94.48%, despite a slightly larger parameter count. This observation implies that overly dense scales may introduce redundancy or amplify noisy interactions, which can weaken generalization on challenging real-world scans. In practice, the best-performing scale set emphasizes fine-to-mid ranges while avoiding excessively large receptive fields. Based on these results, we adopt $(2, 4, 8, 12, 16, 24, 32, 48)$ as the default topology scale configuration for the pre-trained backbone PointGPT-L.

Discussion

Our results show that explicitly injecting multi-scale topology cues is an effective way to address topology-induced domain shift in real point-cloud scans while keeping the backbone frozen. Across both MAE-style and GPT-style backbones, TGCS consistently improves accuracy with a small trainable budget, and in several settings it matches or exceeds full fine-tuning. For example, with PointGPT-L, TGCS reaches 98.97%, 97.42%, and 95.00% on the three ScanObjectNN settings while training only 2.2M parameters, demonstrating that geometry-grounded residual conditioning can deliver strong adaptation without updating the entire model.

These gains also clarify when topology guidance is most valuable. The improvements remain stable in low-data regimes and dense prediction. TGCS surpasses full fine-tuning in episodic few-shot evaluation (e.g., +0.25% on 5-way 10-shot and +1.11% on 10-way 10-shot) and improves part segmentation from 82.69 to 86.69 mIoU, slightly above full fine-tuning. Ablations further suggest that the topology-guided adapter provides the core capability, while cross-scale mixing and saliency-aware gating refine where and how topology cues modulate feature updates, enabling all components yields the best and most consistent performance. This pattern supports the interpretation that topology signals act as a reliable, geometry-aligned adaptation cue when labels are scarce or when transfer requires preserving local geometric relations.

Several limitations point to promising future directions. TGCS currently relies on KNN-based topology construction and handcrafted statistics, which can be sensitive to neighborhood hyperparameters and introduce preprocessing overhead at large token counts. A natural next step is to learn topology descriptors end-to-end and to replace exact KNN with efficient approximations or dynamic neighborhoods that adapt to scale and density. Beyond classification and part segmentation, evaluating TGCS on detection, scene understanding, and open-vocabulary 3D tasks would clarify how topology-guided conditioning interacts with more complex objectives and long-range context. More broadly, the deployment-friendly design retaining a single frozen backbone and attaching lightweight task modules suggests a practical path toward scalable 3D foundation model customization under stricter compute and maintenance constraints.

Method

Overview: Topology-Guided Cross-Scale PEFT

As depicted in Fig. 2(a), we propose **Topology-Guided Cross-Scale (TGCS)** adapter as a plug-and-play parameter-efficient adaptation framework for point-cloud backbones. Given a pre-trained backbone f_θ, TGCS keeps backbone parameters θ frozen and introduces a lightweight set of trainable parameters ϕ. TGCS is designed to be *architecture-agnostic* that inserted into different backbones at the *same positions* with an identical interface, while allowing backbone-specific configurations.

Motivation

We build TGCS upon two observations. First, many real-scan domain shifts manifest primarily as *topology shifts*, which directly alter neighborhood relations in euclidean space and thus change the structural context before semantic features stabilize. Second, topology shifts are inherently *multi-scale*. A small neighborhood may be overly sensitive to sparsity and noise, whereas a large neighborhood may oversmooth local geometry or mix distinct structures. Therefore, a robust PEFT strategy should incorporate explicit topology cues and cross-scale context aggregation and mixing, while maintaining a strict parameter budget.

High-level pipeline

TGCS operates on a token sequence $\mathbf{X} \in \mathbb{R}^{B\times T\times d}$ and associated token anchors $\mathbf{P} \in \mathbb{R}^{B\times T\times 3}$, where T is the number of tokens. Patch tokens may plus an optional `[CLS]` token. TGCS consists of three components: a *CS-Mixing* applied before the backbone to inject early multi-scale context, a *SA-Gating* to allocate adaptation capacity selectively, and a *TG-Adapter* inserted into each backbone block to produce topology-guided residual updates. An overview of the full pipeline and where each component is inserted is shown in Fig. 2(a), with the block-level adapter design in Fig. 2(b) and the two-scale token mixing in Fig. 2(c).

Tokenization and Geometric Anchors

Most modern point-cloud backbones first convert an input point set into a sequence of patch tokens. Given a point cloud $\mathcal{P} = \{\mathbf{p}_n\}_{n=1}^{N}$, we sample G patch centers $\{\mathbf{c}_g\}_{g=1}^{G}$ and gather a local neighborhood around each center to form patches. A patch encoder maps each patch to a token $\mathbf{x}_g \in \mathbb{R}^d$, yielding

$$\mathbf{X}_{\text{patch}} = [\mathbf{x}_1; \ldots; \mathbf{x}_G] \in \mathbb{R}^{G\times d}. \tag{1}$$

$$\mathbf{C} = [\mathbf{c}_1; \ldots; \mathbf{c}_G] \in \mathbb{R}^{G\times 3}. \tag{2}$$

We treat $\mathbf{C}$ as *geometric anchors*, which support topology construction and scale-aware aggregation independent of the backbone's internal design. If a backbone uses a `[CLS]` token, we prepend it and attach a dummy anchor to keep the interface

consistent:

$$\begin{aligned} \mathbf{X} &= \mathrm{Concat}(\mathbf{x}_{\mathrm{cls}},\ \mathbf{X}_{\mathrm{patch}}) \in \mathbb{R}^{T\times d}, \\ \mathbf{P} &= \mathrm{Concat}(\mathbf{0},\ \mathbf{C}) \in \mathbb{R}^{T\times 3}, \qquad T = G+1. \end{aligned} \tag{3}$$

Multi-Scale Topology Descriptor Construction

TGCS derives topology cues from relative geometry among token anchors. For each token $i \in \{1,\dots,T\}$, we construct neighborhoods at multiple scales $\mathcal{K} = \{k_1,\dots,k_S\}$, and compute local statistics within each neighborhood.

Multi-scale KNN neighborhoods and relative offsets

For each scale $k_s \in \mathcal{K}$, we build a KNN neighborhood $\mathcal{N}_{k_s}(i)$ in euclidean space using token anchors $\mathbf{P}$. Let $\mathbf{p}_i \in \mathbb{R}^3$ denote the anchor of token i. We define relative offsets and distances:

$$\Delta\mathbf{p}_{ij} = \mathbf{p}_j - \mathbf{p}_i, \qquad d_{ij} = \|\Delta\mathbf{p}_{ij}\|_2, \qquad j \in \mathcal{N}_{k_s}(i). \tag{4}$$

They encode local directionality because each vector points from the anchor to a neighbor, capturing the local arrangement. Meanwhile, the magnitudes $\|\Delta\mathbf{p}_{ij}\|_2$ of these offsets implicitly reflect sampling density. In sparse regions they tend to be larger, while in dense regions they shrink, making simple summaries such as the mean or its inverse a stable proxy for local point density. In particular, its second-order statistics, *e.g.,* $\sum_j \Delta\mathbf{p}_{ij}\Delta\mathbf{p}_{ij}^\top$, reveal anisotropy and principal directions, distinguishing line-like, planar, or isotropic neighborhoods.

Density and dispersion statistics

We adopt two lightweight scalar statistics per token and per scale.

Density proxy

We use the inverse of the mean neighbor distance:

$$\rho_i^{(s)} = \frac{1}{\frac{1}{k_s}\sum_{j\in\mathcal{N}_{k_s}(i)} d_{ij} + \epsilon}, \tag{5}$$

where ϵ is a small constant for numerical stability. Intuitively, $\rho_i^{(s)}$ increases in denser regions and decreases under sparsity or missing points.

Dispersion proxy

We measure dispersion by averaging coordinate-wise variance of the offsets:

$$\nu_i^{(s)} = \frac{1}{3}\sum_{c\in\{x,y,z\}} \mathrm{Var}_{j\in\mathcal{N}_{k_s}(i)}\left(\Delta p_{ij}^{(c)}\right). \tag{6}$$

This statistic captures how widely neighbors spread around token i and complements density.

Optional distance-weighted variants

To reduce sensitivity to far neighbors, TGCS can optionally apply distance-based weights. We define normalized Gaussian weights:

$$\tilde{w}_{ij}^{(s)} = \exp\left(-\frac{d_{ij}^2}{2\tau^2}\right), \qquad w_{ij}^{(s)} = \frac{\tilde{w}_{ij}^{(s)}}{\sum_{u\in\mathcal{N}_{k_s}(i)} \tilde{w}_{iu}^{(s)} + \epsilon}, \tag{7}$$

where $\tau > 0$ is a learnable bandwidth. Then the density and dispersion can be computed as weighted statistics:

$$\begin{aligned} \rho_i^{(s)} &= \frac{1}{\sum_{j\in\mathcal{N}_{k_s}(i)} w_{ij}^{(s)} d_{ij} + \epsilon}, \\ \boldsymbol{\mu}_i^{(s)} &= \sum_{j\in\mathcal{N}_{k_s}(i)} w_{ij}^{(s)} \Delta\mathbf{p}_{ij}, \\ \nu_i^{(s)} &= \frac{1}{3} \sum_{c\in\{x,y,z\}} \sum_{j\in\mathcal{N}_{k_s}(i)} w_{ij}^{(s)} \left(\Delta p_{ij}^{(c)} - \mu_i^{(s,c)}\right)^2. \end{aligned} \tag{8}$$

Second-order geometry via eigenvalue-derived cues

Beyond first-order statistics, TGCS uses a compact second-order moment to capture local shape tendencies. For each token and scale, we compute a covariance-like matrix:

$$\mathbf{C}_i^{(s)} = \frac{1}{k_s - 1} \sum_{j\in\mathcal{N}_{k_s}(i)} \Delta\mathbf{p}_{ij} \Delta\mathbf{p}_{ij}^\top \in \mathbb{R}^{3\times 3}. \tag{9}$$

Let $\lambda_1^{(s)} \geq \lambda_2^{(s)} \geq \lambda_3^{(s)} \geq 0$ be the eigenvalues of $\mathbf{C}_i^{(s)}$. Geometrically, $\lambda_r^{(s)}$ measures the variance of local offsets along the r-th principal direction. Hence the relative magnitudes of $(\lambda_1, \lambda_2, \lambda_3)$ indicate whether the neighborhood is elongated, spread on a plane, or isotropic. These eigenvalue-derived features have proven particularly effective in characterizing local geometric structures under varying sampling densities and partial observations. Most cues below normalize by $\lambda_1^{(s)}$ or by $\sum_r \lambda_r^{(s)}$ to remove the absolute scale of the neighborhood. This makes the descriptors *dimensionless* and more robust to density and radius changes, focusing on *shape ratios* rather than raw magnitudes. Using the same denominator also makes different cues directly comparable and bounded. We compute six dimensionless geometry cues commonly used to describe local 3D structure:

$$\begin{aligned} \mathrm{L}_i^{(s)} &= \frac{\lambda_1^{(s)} - \lambda_2^{(s)}}{\lambda_1^{(s)} + \epsilon}, \qquad \mathrm{P}_i^{(s)} = \frac{\lambda_2^{(s)} - \lambda_3^{(s)}}{\lambda_1^{(s)} + \epsilon}, \\ \mathrm{S}_i^{(s)} &= \frac{\lambda_3^{(s)}}{\lambda_1^{(s)} + \epsilon}, \qquad \mathrm{A}_i^{(s)} = \frac{\lambda_1^{(s)} - \lambda_3^{(s)}}{\lambda_1^{(s)} + \epsilon}. \end{aligned} \tag{10}$$

Here *Linearity* $\mathrm{L}_i^{(s)}$ is high when $\lambda_1 \gg \lambda_2$, indicating a line-like neighborhood. *Planarity* $\mathrm{P}_i^{(s)}$ is high when $\lambda_1 \approx \lambda_2 \gg \lambda_3$, indicating a plane-like neighborhood. *Sphericity* $\mathrm{P}_i^{(s)}$ increases as $\lambda_1 \approx \lambda_2 \approx \lambda_3$, since λ_3/λ_1 approaches 1. Then *Anisotropy* $\mathrm{A}_i^{(s)}$ measures overall deviation from isotropy by comparing the strongest and weakest variances.

$$
\begin{aligned}
\mathrm{O}_i^{(s)} &= \left(\lambda_1^{(s)}\lambda_2^{(s)}\lambda_3^{(s)}\right)^{\frac{1}{3}}, \\
p_r^{(s)} &= \frac{\lambda_r^{(s)}}{\lambda_1^{(s)} + \lambda_2^{(s)} + \lambda_3^{(s)} + \epsilon}, \qquad r \in \{1,2,3\}, \\
\mathrm{E}_i^{(s)} &= -\sum_{r=1}^{3} p_r^{(s)} \log\left(p_r^{(s)} + \epsilon\right).
\end{aligned}
\tag{11}
$$

Omnivariance $\mathrm{O}_i^{(s)}$ is the geometric mean of eigenvalues, reflecting the overall volumetric spread of the neighborhood. In contrast to ratio-based cues, it partially preserves the magnitude of local spread. Finally, *Eigenentropy* $\mathrm{E}_i^{(s)}$ measures how evenly variance is distributed across principal directions, which is small when one direction dominates (peaked p_r), and large when variance is more uniform (nearly equal p_r), thus providing a stable isotropy indicator. Overall, these cues summarize how variance distributes across principal directions, providing a stronger structural signal than raw distances alone.

Descriptor assembly across scales

For each token i and each scale k_s, we assemble a per-scale descriptor

$$
\begin{aligned}
\mathbf{q}_i^{(s)} &= [\mathrm{L}_i^{(s)},\ \mathrm{P}_i^{(s)},\ \mathrm{S}_i^{(s)},\ \mathrm{A}_i^{(s)},\ \mathrm{O}_i^{(s)},\ \mathrm{E}_i^{(s)}], \\
\mathbf{t}_i^{(s)} &= \left[\rho_i^{(s)},\ \nu_i^{(s)},\ \mathbf{q}_i^{(s)}\right],
\end{aligned}
\tag{12}
$$

and concatenate all scales to obtain the final topology descriptor:

$$
\mathbf{t}_i = \mathrm{Concat}\left(\mathbf{t}_i^{(1)}, \ldots, \mathbf{t}_i^{(S)}\right) \in \mathbb{R}^{d_{\mathrm{topo}}}. \tag{13}
$$

This multi-scale construction yields a topology cue that is both expressive and lightweight, and its configuration, *e.g.*, choice of $\mathcal{K}$ and whether to use weighted variants, can be adapted to different backbones.

Cross-Scale Token Mixing (CS-Mixing)

As shown in Fig. 2(c), TGCS injects cross-scale context prior to the frozen backbone. This early mixing mitigates topology shifts by providing multi-scale neighborhood summaries to patch tokens before deeper processing. Given patch tokens $\mathbf{X}_{\mathrm{patch}}$ and anchors $\mathbf{C}$, we use two scales k_s and k_l to compute mean aggregated neighbors:

$$
\mathbf{a}_s(g) = \mathrm{mean}_{j \in \mathcal{N}_{k_s}(g)} \mathbf{x}_j, \qquad \mathbf{a}_l(g) = \mathrm{mean}_{j \in \mathcal{N}_{k_l}(g)} \mathbf{x}_j. \tag{14}
$$

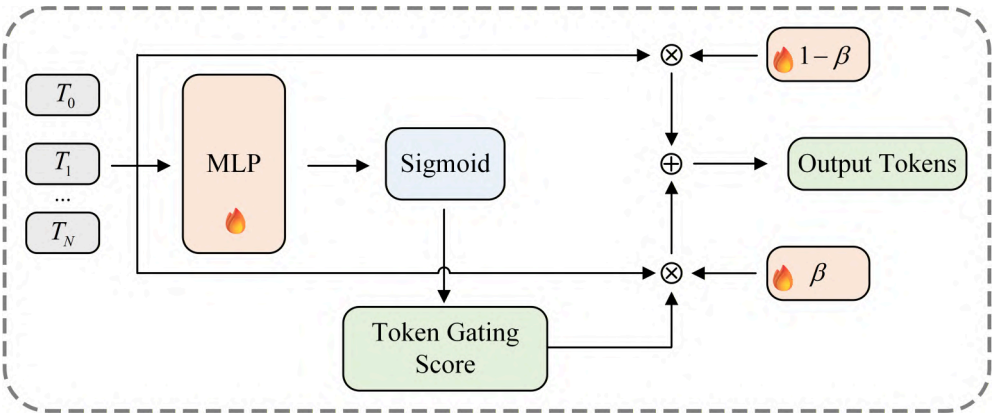


Fig. 5: Saliency-Aware Token Gating (SA-Gating). An MLP predicts token reliability and blends gated and original tokens.

The mixer aggregates token features over two neighborhood scales defined on the token anchors in euclidean space. We use k_s and k_l to denote the numbers of nearest neighbors at a *small* and a *large* scale, respectively, with $k_s < k_l$. Given a fixed KNN rule, the corresponding neighborhoods $\mathcal{N}_{k_s}(g)$ and $\mathcal{N}_{k_l}(g)$ induce two effective spatial radii,

$$\begin{aligned} r_s(g) &= \max_{j \in \mathcal{N}_{k_s}(g)} \|\mathbf{c}_j - \mathbf{c}_g\|_2, \\ r_l(g) &= \max_{j \in \mathcal{N}_{k_l}(g)} \|\mathbf{c}_j - \mathbf{c}_g\|_2, \end{aligned} \tag{15}$$

so k_s captures fine local geometry within a smaller receptive field, while k_l expands the receptive field to include broader context. Importantly, although k_s and k_l are discrete neighbor counts, the induced radii (r_s, r_l) adapt automatically to local sampling density. This density-adaptive behavior makes two-scale mixing robust to non-uniform sampling, as it provides both a local view and a larger-context view without requiring a fixed metric radius[3]. A lightweight function $h_{\text{mix}}(\cdot)$ fuses these contexts into a residual update:

$$\begin{aligned} \Delta \mathbf{x}_g^{\text{mix}} &= h_{\text{mix}}\Big(\text{Concat}(\mathbf{a}_s(g), \mathbf{a}_l(g))\Big), \\ \mathbf{x}_g &\leftarrow \mathbf{x}_g + \lambda_{\text{mix}} \cdot \Delta \mathbf{x}_g^{\text{mix}}, \end{aligned} \tag{16}$$

where λ_{mix} is a learnable scalar. This module is parameter-efficient because it introduces only a small MLP[80] and avoids modifying any backbone weights.

Saliency-Aware Token Gating (SA-Gating)

TGCS further introduces a soft saliency gate to allocate adaptation capacity, as illustrated in Fig. 5. Given token features $\mathbf{X} = [\mathbf{x}_1; \ldots; \mathbf{x}_T]$, we predict a token-wise gate $\gamma_i \in (0, 1)$:

$$s_i = h_{\text{gate}}(\mathbf{x}_i), \qquad \gamma_i = \sigma\left(\frac{s_i}{\tau_{\text{gate}}}\right), \tag{17}$$

where h_{gate} is a lightweight MLP, $\sigma(\cdot)$ is sigmoid, and τ_{gate} is a temperature. We then blend gated tokens with original tokens using a learnable scale λ_{gate}:

$$\mathbf{x}_i \leftarrow (1 - \lambda_{\text{gate}})\mathbf{x}_i + \lambda_{\text{gate}}(\gamma_i \mathbf{x}_i). \tag{18}$$

When a `[CLS]` token is present, it is kept unchanged to preserve global pooling semantics.

Topology-Guided Cross-Scale Adapter (TG-Adapter)

As shown in Fig. 2(b), the TG-Adapter is the parameter-efficient core of TGCS, inserted after the self-attention module in each transformer block. It fuses pre-computed multi-scale topological descriptors (Sec. 1) across scales to generate lightweight, topology-conditioned residual signals that calibrate token representations. This design achieves three essentials: (i) structure-aware adaptation guided by geometric topology, (ii) cross-scale stability via hierarchical cue propagation, and (iii) extreme parameter efficiency. The protocol generalizes seamlessly across backbone architectures while preserving frozen pre-trained knowledge.

Fixed insertion location

Let $\mathbf{X}^\ell$ denote token features at layer ℓ. A standard transformer attention update is

$$\mathbf{U}^\ell = \mathbf{X}^\ell + \mathrm{Attn}\big(\mathrm{LN}(\mathbf{X}^\ell)\big)\,. \tag{19}$$

TGCS inserts an adapter between attention and FFN:

$$\mathbf{U}^\ell \leftarrow \mathbf{U}^\ell + \alpha \cdot \Delta_\phi^\ell\Big(\mathbf{U}^\ell,\ \overline{\mathbf{X}}^{\ell-2},\ \mathbf{T}\Big)\,, \tag{20}$$

and then applies the frozen FFN:

$$\mathbf{X}^{\ell+1} = \mathbf{U}^\ell + \mathrm{FFN}\big(\mathrm{LN}(\mathbf{U}^\ell)\big)\,. \tag{21}$$

Here $\mathbf{X}^\ell \in \mathbb{R}^{N\times d}$ denotes the N token representations with feature dimension d at transformer layer ℓ. $\mathrm{Attn}(\cdot)$ and $\mathrm{FFN}(\cdot)$ represent the frozen self-attention and feed-forward sublayers, and $\mathrm{LN}(\cdot)$ is layer normalization. $\mathbf{U}^\ell$ is the intermediate token state after the attention update. The adapter function $\Delta_\phi^\ell(\cdot)$ is a lightweight learnable residual module with parameters ϕ, conditioned on the current tokens $\mathbf{U}^\ell$, cross-layer cached features $\overline{\mathbf{X}}^{\ell-2}$ from an earlier layer, and the multi-scale topology descriptor matrix $\mathbf{T}$.

Cross-layer context

Cross-scale information can be captured in two complementary ways that *explicitly* by aggregating cross-scale neighborhoods in the input geometry, and *implicitly* by network depth, since successive attention and MLP transformations progressively enlarge

the effective receptive field. TGCS exploits the latter by introducing a *shallow cross-layer context* as an additional conditioning signal. Specifically, we reuse a feature snapshot from an earlier layer as a proxy of a coarser-scale representation:

$$\overline{\mathbf{X}}^{\ell-2} = \mathrm{stopgrad}\big(\mathbf{X}^{\ell-2}\big)\,. \tag{22}$$

From a representation viewpoint, $\mathbf{X}^{\ell}$ and $\mathbf{X}^{\ell-2}$ encode information at different effective scales. The former retains more fine-grained, layer-local details, while the latter aggregates broader context through preceding blocks. Conditioning the adapter on $\overline{\mathbf{X}}^{\ell-2}$ therefore provides a stable coarse reference that helps disambiguate topology-induced feature perturbations, especially under density shift and partial observations. Moreover, applying stopgrad$(\cdot)$ aligns with the PEFT assumption by preventing gradients from propagating into the frozen backbone. Equivalently, the cached context acts as a fixed, pre-trained prior that guides the learned residual while avoiding degenerate solutions that would implicitly re-tune backbone representations. For early layers where such cross-layer context is not yet available, we simply omit this branch, which yields a natural warm-up behavior without altering the backbone computation.

Topology-conditioned residual prediction

The adapter predicts a residual using a lightweight network $h^{\ell}_{\mathrm{adp}}(\cdot)$ that conditions on current tokens $\mathbf{U}^{\ell}$, cached tokens $\overline{\mathbf{X}}^{\ell-2}$, and topology cues $\mathbf{T}$:

$$\begin{aligned} \mathbf{F}^{\ell} &= \mathrm{Concat}\Big(\mathbf{U}^{\ell},\ \overline{\mathbf{X}}^{\ell-2},\ \mathbf{T}\Big)\,, \\ \Delta^{\ell}_{\phi} &= h^{\ell}_{\mathrm{adp}}\big(\mathrm{LN}(\mathbf{F}^{\ell})\big)\,, \qquad \mathbf{U}^{\ell} \leftarrow \mathbf{U}^{\ell} + \alpha \cdot \Delta^{\ell}_{\phi}. \end{aligned} \tag{23}$$

In practice, h^{ℓ}_{adp} is implemented as a small MLP with dropout, leading to a negligible number of trainable parameters compared to the backbone. When the backbone is frozen, the model can not substantially reconfigure how neighborhoods are encoded. Topology descriptors provide an explicit signal of local structure and density that remains meaningful even when semantic features are imperfect under domain shift. Conditioning the adapter on $\mathbf{T}$ enables *structure-aware* residual updates, which is more targeted than generic token-space perturbations. Topology descriptors alone reflect local geometry at selected scales while cached features provide a complementary view of broader context. Their combination allows TGCS to correct representations in a manner that is both locally grounded and semantically informed.

Backbone-Agnostic Configuration and Plug-and-Play Protocol

TGCS is designed as a plug-and-play PEFT layer for token-based point-cloud backbones. Let $\mathbf{X}^{\ell} \in \mathbb{R}^{T\times d}$ be the token features at depth ℓ with anchors $\mathbf{P} \in \mathbb{R}^{T\times 3}$. TGCS keeps a *fixed insertion protocol* across backbones, which performs pre-encoder modulation on input tokens, and injects a topology-conditioned residual *at the same location* inside every block after token interactions are formed and before the next refinement stage.

Unified insertion form

For a backbone block $\mathcal{B}_\theta^\ell(\cdot)$, TGCS augments it by a dimension-preserving residual:

$$\mathbf{X}^{\ell+1} = \mathcal{B}_\theta^\ell(\mathbf{X}^\ell) + \alpha^\ell \, \Delta_\phi^\ell\big(\mathbf{X}^\ell,\ \mathbf{T}(\mathbf{P})\big)\,, \tag{24}$$

where θ is frozen, Δ_ϕ^ℓ is a lightweight adapter conditioned on the multi-scale topology cue $\mathbf{T}(\mathbf{P})$, and α^ℓ is a learnable scalar controlling adaptation strength. This form is architecture-agnostic because it only requires the backbone to expose token features and anchors, then outputs a residual with the same dimension d.

Backbone-dependent configuration

Across architectures, TGCS keeps Eq. (24) unchanged and only adjusts a small configuration tuple:

$$\Gamma = \Big(d,\ d_{\text{hid}},\ \mathcal{K},\ (k_s, k_l),\ \mathcal{L}_{\text{ins}}\Big), \tag{25}$$

where d is the token dimension, d_{hid} is the adapter hidden width, $\mathcal{K}$ specifies topology scales, (k_s, k_l) specifies mixing scales, and $\mathcal{L}_{\text{ins}}$ denotes the set of layers where TGCS is inserted. With fixed insertion and minimal configuration, TGCS can be attached to diverse backbones without modifying their internal weights or layer definitions.

Training Objective and Complexity

Given a downstream dataset $\mathcal{D} = \{(\mathcal{P}_n, y_n)\}_{n=1}^M$ and task loss $\mathcal{L}$, TGCS optimizes only ϕ:

$$\phi^\star = \arg\min_\phi \ \frac{1}{M} \sum_{n=1}^{M} \mathcal{L}\big(f_{\theta,\phi}(\mathcal{P}_n),\ y_n\big), \qquad \theta \text{ frozen.} \tag{26}$$

The main overhead comes from KNN neighborhood construction on G anchors and computing lightweight statistics. With sparse neighborhoods, the additional cost is approximately $\mathcal{O}(G \cdot \sum_s k_s)$ for topology cues plus small MLPs, which is modest compared to backbone.

Declarations

- Funding: This work was supported in part by Guangzhou Science and Technology Bureau Research and Development Project under Grant SL2022B01J00068, in part by Guangdong Natural Science Foundation-General Project under Grant 2023A1515012546, in part by the Key Project of the National Key Research Program under Grant SQ2024YFF0500187.

- Conflict of interest/Competing interests: The authors declare no competing interests.
- Ethics approval: (Not applicable)
- Consent to participate: (Not applicable)
- Consent for publication: (Not applicable)
- Data availability: The datasets used in this study are publicly available. ScanObjectNN is available at https://hkust-vgd.github.io/scanobjectnn/. ModelNet40 is

available at https://modelnet.cs.princeton.edu/. ShapeNetPart is available at https://shapenet.cs.stanford.edu/media/shapenetcore_partanno_v0.zip.

- Code availability: The code will be released at https://github.com/ZhengLingxiang/TGCS.
- Author contribution: Lingxiang Zheng designed the study and wrote the manuscript. Rongqian Yang provided supervision and revision.

References

[1] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu and M. Bennamoun, "Deep learning for 3d point clouds: a survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 12, pp. 4338-4364, 2021.

[2] 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 *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 11105-11114, 2020.

[3] C. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: deep hierarchical feature learning on point sets in a metric space," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, pp. 5105–5114, 2017.

[4] Y. Wang, Y. Sun, Z. Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon, "Dynamic graph cnn for learning on point clouds," *ACM Trans. Graph.*, vol. 38, no. 5, pp. 1-12, 2019.

[5] H. Zhao, L. Jiang, J. Jia, P. Torr and V. Koltun, "Point transformer," in *Proc. IEEE Int. Conf. Comput. Vis.*, pp. 16239-16248, 2021.

[6] Y. Li, R. Bu, M. Sun, and B. Chen, "PointCNN: convolution on x-transformed points," in *Proc. Neural Inf. Process. Syst.*, 2018.

[7] W. Wu, Z. Qi, and L. Fuxin, "PointConv: deep convolutional networks on 3d point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 9621-9630, 2019.

[8] 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 *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 19313-19322, 2022.

[9] Y. Pang, W. Wang, F.E.H. Tay, W. Liu, Y. Tian and L. Yuan, "Masked autoencoders for point cloud self-supervised learning," in *Proc. Eur. Conf. Comput. Vis.*, 2022.

[10] R. Zhang, Z. Guo, P. Gao, R. Fang, B. Zhao, D. Wang, and H. Li, "Point-M2AE: multi-scale masked autoencoders for hierarchical point cloud pre-training," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, pp. 27061-27074, 2022.

[11] R. Dong, Z. Qi, L. Zhang, J. Zhang, J. Sun, Z. Ge, L. Yi, and K. Ma, "Autoencoders as cross-modal teachers: can pretrained 2d image transformers help 3d representation learning?" in *Int. Conf. Learn. Represent.*, 2023.

[12] Z. Qi, R. Dong, G. Fan, Z. Ge, X. Zhang, K. Ma, and L. Yi, "Contrastive 3d representation learning guided by generative pretraining," in *Proc. Int. Conf. Mach. Learn.*, pp. 28223-28243, 2023.

[13] G. Chen, M. Wang, Y. Yi, K. Yu, L. Yuan, and Y. Yue, "PointGPT: auto-regressively generative pre-training from point clouds," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 36, pp. 29667-29679, 2023.

[14] N. Ding, Y. Qin, G. Yang, F. Wei, Z. Yang, Y. Su, and M. Sun, "Parameter-efficient fine-tuning of large-scale pre-trained language models," *Nat. Mach. Intell.*, vol. 5, no. 3, pp. 220-235, 2023.

[15] M. B. McDermott, B. Yap, P. Szolovits, and M. Zitnik, "Structure-inducing pre-training," *Nat. Mach. Intell.*, vol. 5, no. 6, pp. 612-621, 2023.

[16] Z. Han, C. Gao, J. Liu, J. Zhang, and S. Q., Zhang, "Parameter-efficient fine-tuning for large models: a comprehensive survey," *arXiv*, 2024.

[17] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, and S. Gelly, "Parameter-efficient transfer learning for nlp," in *Proc. Int. Conf. Mach. Learn.*, pp. 2790-2799, 2019.

[18] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," in *Proc. Conf. Empir. Methods Nat. Lang. Process.*, pp. 3045-3059, 2021.

[19] X. Li and P. Liang, "Prefix-tuning: optimizing continuous prompts for generation," in *Proc. Assoc. Comput. Linguist.*, pp. 4582-4597, 2021.

[20] E. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, and W. Chen, "LoRA: low-rank adaptation of large language models," in *Int. Conf. Learn. Represent.*, vol. 1, no. 2, pp. 3, 2022.

[21] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguist.*, pp. 4171-4186, 2019.

[22] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, and et al., "Language models are few-shot learners," in *Proc. Adv. Neural Inf. Process. Syst.*, pp. 1877-1901, 2020.

[23] E. Zaken, Y. Goldberg, and S. Ravfogel, "BitFit: simple parameter-efficient fine-tuning for transformer-based masked language-models," in *Proc. Assoc. Comput.*

Linguist., vol. 2, pp. 1-9, 2022.

[24] M. Jia, L. Tang, B. Chen, C. Cardie, S. Belongie, B. Hariharan, and S. Lim, "Visual prompt tuning," in *Proc. Eur. Conf. Comput. Vis.*, pp. 709-727, 2022.

[25] M. Xu, Z. Zhang, F. Wei, H. Hu, and X. Bai, "Side adapter network for open-vocabulary semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 2945-2954, 2023.

[26] D. Lian, D. Zhou, J. Feng, and X. Wang, "Scaling and shifting your features: a new baseline for efficient model tuning," in *Proc. Adv. Neural Inf. Process. Syst.*, pp. 109-123, 2022.

[27] Y. Zha, J. Wang, T. Dai, B. Chen, Z. Wang, and S.-T. Xia, "Instance-aware dynamic prompt tuning for pre-trained point cloud models," in *Proc. IEEE Int. Conf. Comput. Vis.*, pp. 14161–14170, 2023.

[28] 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 *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 14707–14717, 2024.

[29] Y. Tang, R. Zhang, Z. Guo, X. Ma, B. Zhao, Z. Wang, D. Wang, and X. Li, "Point-peft: parameter-efficient fine-tuning for 3d pre-trained models," in *Proc. AAAI Conf. Artif. Intell.*, vol. 38, no. 6, pp. 5171–5179, 2024.

[30] S. Wang, X. Liu, L. Kong, J. Xu, C. Hu, G. Fang, W. Li, J. Zhu, and X. Wang, "PointLoRA: low-rank adaptation with token selection for point cloud learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 6605-6615, 2025.

[31] M. A. Uy, Q. Pham, B. Hua, T. Nguyen and S. Yeung, "Revisiting point cloud classification: a new Benchmark dataset and classification model on real-world data," in *Proc. IEEE Int. Conf. Comput. Vis.*, pp. 1588-1597, 2019.

[32] X. Yan, C. Zheng, Z. Li, S. Wang and S. Cui, "PointASNL: robust point clouds processing using nonlocal neural networks with adaptive sampling," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 5588-5597, 2020.

[33] X. Wu, Y. Lao, L. Jiang, X. Liu, and H. Zhao, "Point transformer v2: grouped vector attention and partition-based pooling," in *Proc. Adv. Neural Inf. Process. Syst.*, pp. 33330–33342, 2022.

[34] X. Wu, L. Jiang, P. Wang, Z. Liu, X. Liu, Y. Qiao, W. Ouyang, T. He, and H. Zhao, "Point transformer v3: simpler, faster, stronger," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 4840-4851, 2024.

[35] Y. Wu, J. Liu, M. Gong, Z. Liu, Q. Miao and W. Ma, "MPCT: multiscale point cloud transformer with a residual network," in *IEEE Trans. Multimedia*, vol. 26, pp. 3505-3516, 2024.

[36] D. Li, S. Zheng, Z. Chen, X. Li, L. Wang and J. Du, "Hierarchical local global transformer for point clouds analysis," in *Int J Appl Earth Obs Geoinf.*, vol. 129, pp. 103813, 2024.

[37] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, N. Houlsby, "An image is worth 16x16 words: transformers for image recognition at scale," in *Int. Conf. Learn. Represent.*, 2021.

[38] D. Liang, T. Feng, X. Zhou, Y. Zhang, Z. Zou and X. Bai, "Parameter-efficient fine-tuning in spectral domain for point cloud learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 47, no. 12, pp. 10949-10966, 2025.

[39] S. Xie, J. Gu, D. Guo, CR. Qi, L. Guibas, and O. Litany, "Pointcontrast: unsupervised pre-training for 3d point cloud understanding," in *Proc. Eur. Conf. Comput. Vis.*, pp. 574-591, 2020.

[40] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proc. Int. Conf. Mach. Learn.*, pp. 1597-1607, 2020.

[41] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 9729-9738, 2020.

[42] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, "Emerging properties in self-supervised vision transformers," in *Proc. IEEE Int. Conf. Comput. Vis.*, pp. 9650-9660, 2021.

[43] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, et al., "Bootstrap your own latent: a new approach to self-supervised learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 21271-21284, 2020.

[44] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 16000-16009, 2022.

[45] V. Hondru, F. Croitoru, S. Minaee, R. Ionescu, "Masked image modeling: a survey," *Int. J. Comput. Vision*, vol. 133, no. 10, pp. 7154–7200, 2025.

[46] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," in *Proc. Int. Conf. Mach. Learn.*, vol. 139, pp. 8748-8763, 2021.

[47] R. Zhang, Z. Guo, W. Zhang, K. Li, X. Miao, B. Cui, Y. Qiao, P. Gao, and H. Li, "PointCLIP: point cloud understanding by clip," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 8552-8562, 2022.

[48] Z. Wang, X. Yu, Y. Rao, J. Zhou, and J. Lu., "P2P: tuning pre-trained image models for point cloud analysis with point-to-pixel prompting," in *Proc. Adv. Neural Inf. Process. Syst.*, pp. 14388–14402, 2022.

[49] Z. Wang, Y. Rao, X. Yu, J. Zhou, and J. Lu, "Point-to-pixel prompting for point cloud analysis with pre-trained image models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 6, pp. 4381–4397, 2024.

[50] H. Sun, Y. Wang, W. Chen, H. Deng, and D. Li, "Parameter-efficient prompt learning for 3d point cloud understanding," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2024.

[51] Q. He, J. Zhang, J. Peng, H. He, X. Li, Y. Wang, and C. Wang, "PointRWKV: efficient RWKV-Like model for hierarchical point cloud learning," in *Proc. AAAI Conf. Artif. Intell.*, pp. 3410-3418, 2024.

[52] M. Guo, J. Cai, Z. Liu, T. Mu, R. Martin, and S. Hu, "PCT: point cloud transformer," *Comput. Vis. Media*, vol. 7, no. 2, pp. 187-199, 2021.

[53] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Comput.*, vol. 15, no. 6, pp. 1373-1396, 2003.

[54] L. T. Hai, T. D. Le, Z. Ding, Q. Tian, and T.-S. Hy, "Topology-guided knowledge distillation for efficient point cloud processing," *arXiv*, 2025.

[55] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, pp. 6000–6010, 2017.

[56] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, J. Xiao, "3D shapeNets: a deep representation for volumetric shapes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 1912-1920, 2015.

[57] CR. Qi, H. Su, K. Mo, and LJ. Guibas, "PointNet: deep learning on point sets for 3d classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 652-660, 2017.

[58] G. Qian, Y. Li, H. Peng, J. Mai, H. Hammoud, M. Elhoseiny, and B. Ghanem, "PointNeXt: revisiting pointnet++ with improved training and scaling strategies," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, pp. 23192-23204, 2022.

[59] S. Sun, Y. Rao, J. Lu, and H. Yan, "X-3d: explicit 3d structure modeling for point cloud recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit*, pp. 5074–5083, 2024.

[60] C. Wang, M. Wu, S.-K. Lam, X. Ning, S. Yu, R. Wang, W. Li, and T. Srikanthan, "Gpsformer: a global perception and local structure fitting-based transformer for point cloud understanding," in *Proc. Eur. Conf. Comput. Vis.*, pp. 75–92, 2024.

[61] H. Lin, X. Zheng, L. Li, F. Chao, S. Wang, Y. Wang, Y. Tian, and R. Ji, "Meta architecture for point cloud analysis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit*, pp. 17682–17691, 2023.

[62] H. Thomas, CR. Qi, JE. Deschaud, B. Marcotegui, F. Goulette, and LJ. Guibas, "KPConv: flexible and deformable convolution for point clouds," in *Proc. IEEE Int. Conf. Comput. Vis.*, pp. 6411-6420, 2019.

[63] S. Jie and Z.-H. Deng, "Fact: factor-tuning for lightweight adaptation on vision transformer," in *Proc. AAAI Conf. Artif. Intell.*, vol. 37, no. 1, pp. 1060–1068, 2023.

[64] S. Jie, H. Wang, and Z.-H. Deng, "Revisiting the parameter efficiency of adapters from the perspective of precision redundancy," in *Proc. IEEE Int. Conf. Comput. Vis.*, pp. 17217–17226, 2023.

[65] H. Zhao, P. Wang, Y. Zhao, H. Luo, F. Wang, and M. Z. Shou, "Sct: a simple baseline for parameter-efficient fine-tuning via salient channels," *Int. J. Comput. Vis*, vol. 132, no. 3, pp. 731–749, 2024.

[66] F. He, R. Fei, J.E. Krull, and et al., "Harnessing the power of single-cell large language models with parameter-efficient fine-tuning using scPEFT," *Nat. Mach. Intell.*, vol. 8, pp. 118-133, 2026.

[67] H. Liu, M. Cai, and Y. J. Lee, "Masked discrimination for self-supervised learning on point clouds," in *Proc. Eur. Conf. Comput. Vis.*, pp. 657–675, 2022.

[68] 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," in *Proc. Int. Conf. Learn. Represent.*, 2022.

[69] H. Ran, J. Liu, and C. Wang, "Surface representation for point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 18942–18952, 2022.

[70] A. Hamdi, S. Giancola, and B. Ghanem, "Mvtn: multi-view transformation network for 3d shape recognition," in *Proc. IEEE Int. Conf. Comput. Vis.*, pp. 1-11, 2021.

[71] Z.-H. Lin, S.-Y. Huang, and Y.-C. F. Wang, "Learning of 3d graph convolution networks for point cloud analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 8, pp. 4212-4224, 2021.

[72] C.-Y. Hong, Y.-Y. Chou, and T.-L. Liu, "Attention discriminant sampling for point clouds," in *Proc. IEEE Int. Conf. Comput. Vis.*, pp. 14429–14440, 2023.

[73] H. Wang, Q. Liu, X. Yue, J. Lasenby, and M. J. Kusner, "Unsupervised point cloud pre-training via occlusion completion," in *Proc. IEEE Int. Conf. Comput. Vis.*, pp. 9782–9792, 2021.

[74] Y. Zha, H. Ji, J. Li, R. Li, T. Dai, B. Chen, Z. Wang, and S.- T. Xia, "Towards compact 3d representations via point feature enhancement masked autoencoders," in *Proc. AAAI Conf. Artif. Intell.*, vol. 38, no. 7, pp. 6962–6970, 2024.

[75] X. Zheng, X. Huang, G. Mei, Y. Hou, Z. Lyu, B. Dai, W. Ouyang, and Y. Gong, "Point cloud pre-training with diffusion models," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 22935–22945, 2024.

[76] D. Liang, X. Zhou, W. Xu, X. Zhu, Z. Zou, X. Ye, X. Tan, and X. Bai, "Pointmamba: a simple state space model for point cloud analysis," in *Proc. Adv. Neural Inf. Process. Syst.*, 2024.

[77] T. Feng, W. Wang, R. Quan, and Y. Yang, "Shape2scene: 3d scene representation learning through pre-training on shape data," in *Proc. Eur. Conf. Comput. Vis.*, pp. 73–91, 2024.

[78] Z. Shi and A. Lipani, "Dept: decomposed prompt tuning for parameter-efficient fine-tuning," in *Proc. Int. Conf. Learn. Represent.*, 2024.

[79] Z. Gao, Q. Wang, A. Chen, Z. Liu, B. Wu, L. Chen, and J. Li, "Parameter-efficient fine-tuning with discrete fourier transform," in *Proc. Int. Conf. Mach. Learn.*, 2024.

[80] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.