UPP: Unified Point-Level Prompting for Robust Point Cloud Analysis

Supplementary Material

Training Detail

Downstream tasks in noisy and incomplete condition. In our experiments, we train the downstream classifiers under noisy conditions. For fair comparisons, identical hyper-parameters and training strategies are applied across fine-tuning and proposed methods, following the pioneering work Point-MAE [26], as shown in Table 4. For example, when fine-tuning on Noisy ModelNet40 [34], the training process spans 300 epochs, using a cosine learning rate scheduler [22] that starts at 0.0005, with a 10-epoch warm-up period. The AdamW optimizer [21] is employed to facilitate optimization. To evaluate performance, we utilize the overall accuracy metric, comparing the model's predictions on the clean test set.

All of our experiments across the four datasets adhere to the settings outlined in Table 4, with the exception of the ScanObjectNN dataset. For ScanObjectNN, we set the point number to 2048 and adopt 128 patches to better accommodate the characteristics of real-world scanned data, following previous works [26, 41].

Parameter-Efficient Fine-tuning Settings. As a parameter-efficient fine-tuning method, we merely train our inserted modules with pre-trained backbone weights frozen. Following the approach of DAPT [44], we load pre-trained weights into a Point-MAE [26] model for efficient fine-tuning, excluding residual components for consistency. Notably, ReCon [28] and Point-FEMAE [41] extend Point-MAE [26] with additional modules. We drop these parameters, thus leading to a slight saving of FLOPs. All experiments are implemented using PyTorch version 1.13.1 and conducted on a single GeForce RTX 4090 GPU.

Staged Optimization Strategy. While adapting to downstream tasks, we impose additional objective loss functions to regularize our point-level promoters, the Rectification Prompter and Completion Prompter. During training, we adopt a staged optimization strategy to avoid randomly initialized prompt points disrupting the training of downstream tasks. We add 50 epochs to optimize the point-level promoters, in which the former 20 epochs optimize both the Rectification Prompter and Completion Prompter, and the later 30 epochs optimize only the Completion Prompter. During the downstream adaptation process, we optionally enable the training of the two point-level promoters with the Shape-Aware Unit when the learning rate narrows to 0.0001. To simulate real-world noise and incompletion, we introduce additional outlier points and apply random cropping, ranging from 25% to 50%, to create labeled data pairs for supervision. During training, the backbone weights are frozen, and only the Rectification Prompter, Completion Prompter, and their associated Shape-Aware Unit modules are optimized.

Task	Classification	Segmentation
Optimizer	AdamW	AdamW
Learning rate	0.0005	0.0002
Weight decay	0.05	0.05
Scheduler	cosine	cosine
Training epochs	300	300
Warmup epochs	10	10
Batch size	32	32
Outliers number	24	24
Surface noise number	64	64
Shape missing rate	25%	25%
Points number	1024	2048
Patches number	64	128
Patch size	32	32

Table 4. Training details for downstream classification and segmentation tasks in noise condition.

Method	Param.(M)	Cls. mIoU(%)	Inst. mIoU(%)
Point-MAE[26]	27.06	83.3	85.6
+Point-PEFT[40]	5.62	80.5	83.1
+DAPT[44]	5.65	80.9	83.7
+UPP (Ours)	6.43	**82.2**	**84.4**
Point-FEMAE[41]	27.06	83.5	85.9
+Point-PEFT[40]	5.62	80.7	83.9
+DAPT[44]	5.65	81.3	84.1
+UPP (Ours)	6.43	**82.5**	**84.8**

Table 5. Point cloud part segmentation experiment results on ShapeNetPart [37] dataset under noisy and incomplete setting.

Additional Experiments

Segmentation Experiments on Noisy ShapeNetPart. ShapeNetPart [37] includes 16,881 samples across 16 categories for the object-level part segmentation task. It is challenging to accurately recognize class labels for each point within point cloud instances. Furthermore, we add additional simulated noise points and incompleteness, which are detailed in Table 4.

As shown in Table 5, our method outperforms other state-of-the-art PEFT approaches, such as Point-PEFT [32]

Method	Reference	Param.(M)	Acc.(%)
Point-FEMAE[26]	AAAI 24	27.4	94.0
Linear Probing	-	0.3	91.9
VPT[13]	ECCV 22	0.4	92.6
Adapter[5]	NeurIPS 22	0.9	92.4
LoRA[12]	ICLR 22	0.9	92.3
IDPT[40]	ICCV 23	1.7	93.4
Point-PEFT[32]	AAAI 24	0.7	94.0
DAPT[44]	CVPR 24	1.1	93.2
SA-Unit (Ours)	This Paper	**0.6**	**94.2**

Table 6. Comparison with other PEFT methods on clean ModelNet40 [34] dataset. Our method only utilizes the PEFT module, Shape-Aware Unit (SA-Unit). Classification accuracy without voting is reported. All methods adopt Point-FEMAE as the backbone.

and DAPT [44], on both pre-trained Point-MAE and Point-FEMAE backbones. This success verifies our method's superior robustness to low-quality data and validates its generalizability across diverse downstream tasks. However, we observe that it remains challenging to surpass the performance of full fine-tuning methods in fine-grained analysis tasks like part segmentation, which require substantial model capacity to memorize the training data distribution. Additionally, current PEFT methods, including ours, exhibit greater susceptibility to noise and incompleteness compared to full fine-tuning.

It is worth noting that the majority of trainable parameters in our framework originate from the large downstream task head, highlighting the efficiency of our approach in minimizing additional parameter overhead while maintaining competitive performance.

Comparison with Other PEFT Methods. As shown in Table 6, we present classification results on the clean ModelNet40 dataset and compare our Shape-Aware Unit with other PEFT approaches [5, 12, 13, 32, 40, 44]. Since other methods struggle to tackle low-quality point clouds, we ensure a fair comparison by applying no noise or incompletion settings. Despite these adjustments, our approach achieves the highest accuracy of **94.2%**, outperforming both the state-of-the-art PEFT method DAPT [44] and the full fine-tuning. This success is attributed to the effective interaction of the feature similarity-based self-attention mechanism and spatial distance-based Shape-Aware Attention, capturing critical shape information. These results highlight the adaptability and potential to serve as a general 3D PEFT method.

Impact of Different Prompting Order. The order of point-level prompting is a critical factor influencing the performance of our method. As shown in Table 7, we compare the impact of different prompting orders. Our results indicate that UPP achieves the highest performance of 92.95% when the Rectification Prompter is applied first. This suggests that reducing noise levels forms a solid foundation for accurate point cloud understanding, which is essential for both completion prompting and analysis. Intuitively, performing both completion and rectification concurrently could offer better computational parallelism. However, this approach yields only marginal performance improvements. This is because the Completion Prompter relies on the Rectification Prompter to rectify noisy points, enabling filtered features of the point cloud. Interestingly, performing completion before rectification results in improved performance than concurrent, as the Rectification Prompter helps to correct artifacts introduced by low-quality completion. Based on these empirical findings, we adopt the rectification first strategy in our method.

Concurrently	Complete First	Rectify First	Acc.(%)
✓	-	-	90.76
-	✓	-	91.18
-	-	✓	**92.95**

Table 7. Abaltion on point-level prompting order.

Rect. Prompter	Compl. Prompter	Shape-Aware Unit
0.148M	0.370M	0.028M

Table 8. Parameters of each component in our UPP.

Parameters Efficiency

Our UPP paradigm employs only 1.4 M trainable parameters and requires 6.1 G FLOPs, significantly reducing computational costs compared to the ensemble paradigm while achieving superior performance. This efficiency is attributed to our compact module design and the progressive extraction of point cloud features from shallow to deep layers.

The enhancement in parameter efficiency arises from the insight that, in the ensemble paradigm, the denoising, completion, and analysis models each include dedicated feature extraction modules designed for task-specific knowledge. By contrast, our approach leverages a unified pre-trained backbone for robust feature extraction. Lightweight Shape-Aware Unit modules are then employed to adaptively adjust feature distributions for specific tasks. This unified design substantially improves the efficiency of both the total parameter count and the trainable parameters, achieving a balance between performance and resource utilization, as detailed in Table 8.

Implementation Detail

Spatial Interpolation

We provide detailed formulations for the spatial interpolation operation $\mathcal{F}(\cdot)$ utilized in Equation 2 and Equation 11.

Given a set of center points with coordinates $\{c_i\} \in \mathbb{R}^{C\times 3}$, where $i = 1, \ldots, C$, and the corresponding features $\{f(c_i)\}$, the objective of the Propagation operation is to compute the features of a neighboring point $x \in \mathbb{R}^3$ using spatial interpolation. The resulting feature $f(x)$ is derived from x, the coordinates $\{c_i\}$, and the features $\{f(c_i)\}$, demonstrated as:

$$f(x) = \mathcal{F}(\{f(c_i)\}, \{c_i\}, x) \tag{17}$$

First, we compute the Euclidean distance from x to each center point c_i:

$$d(x, c_i) = \|x - c_i\|. \tag{18}$$

Next, we calculate the weight by taking the inverse of the spatial distance:

$$w(x, c_i) = \frac{1}{d(x, c_i)^p}, \tag{19}$$

where p is typically set to 2.

This results in a set of weights $w(x, c_i)$ for $i = 1, \ldots, C$. We then select only the top-K weights for interpolation:

$$\{w(x, c_j)\} = \text{Top-}K(\{w(x, c_i)\}), \tag{20}$$

where $j = 1, \ldots, K$, and K is typically set to 6. Subsequently, the interpolation of features is based on the weights, formulated as:

$$f(x) = \frac{\sum_{j=1}^{K} w(x, c_j) f(c_j)}{\sum_{j=1}^{K} w(x, c_j)}. \tag{21}$$

Finally, this procedure is repeated for each neighboring point to obtain their features for further utilization. The Propagation operation effectively transfers and aggregates features by leveraging spatial relationships, enabling robust and efficient feature refinement. This mechanism is particularly suited for point clouds, where irregular data distribution necessitates dynamic interpolation based on spatial distances.