# SOP$^2$: Transfer Learning with Scene-Oriented Prompt Pool on 3D Object Detection

Ching-Hung Cheng[1], Hsiu-Fu Wu[2], Bing-Chen Wu[1], Khanh-Phong Bui[1],Van-Tin Luu[1], Ching-Chun Huang[1*]
[1]National Yang Ming Chiao Tung University, Hsinchu, Taiwan
[2]Internet of Things Laboratory, Chunghwa Telecom Laboratories, Taoyuan, Taiwan
{hong3924.cs10, evan20010126.cs12, phongbk.ee13, tinery.ee12, chingchun}@nycu.edu.tw
q104769424@cht.com.tw

## Abstract

*With the rise of Large Language Models (LLMs) such as GPT-3, these models exhibit strong generalization capabilities. Through transfer learning techniques such as fine-tuning and prompt tuning, they can be adapted to various downstream tasks with minimal parameter adjustments. This approach is particularly common in the field of Natural Language Processing (NLP). This paper aims to explore the effectiveness of common prompt tuning methods in 3D object detection. We investigate whether a model trained on the large-scale Waymo dataset can serve as a foundation model and adapt to other scenarios within the 3D object detection field. This paper sequentially examines the impact of prompt tokens and prompt generators, and further proposes a Scene-Oriented Prompt Pool (**SOP**$^2$). We demonstrate the effectiveness of prompt pools in 3D object detection, with the goal of inspiring future researchers to delve deeper into the potential of prompts in the 3D field.*

## 1. Introduction

3D object detection is a crucial task in computer vision, enabling applications such as autonomous driving, robotics, augmented reality, and urban planning. Large-scale datasets like Waymo [10] and KITTI [3] have significantly advanced this field, yet models trained on one dataset often fail to generalize well across different domains due to domain gaps. These gaps arise from variations in sensor configurations, environmental conditions, and data collection methodologies, ultimately leading to degraded performance when models are deployed in new environments. Addressing this challenge is essential for developing robust and adaptable 3D object detection systems.

One common strategy to bridge the domain gap is Unsupervised Domain Adaptation (UDA), which aims to improve model generalization without requiring labeled target domain data. UDA enables models to learn shared feature representations between source and target domains, enhancing cross-domain performance. However, due to the complexity of designing effective training strategies and loss functions, UDA models often struggle to extract meaningful features from unlabeled target data, resulting in suboptimal adaptation. Fig. 1a illustrates how UDA improves generalization by leveraging target domain data.

Recent advancements in Natural Language Processing (NLP) have introduced Large Language Models (LLMs) like GPT-3 [2], which learn from massive datasets and demonstrate strong generalization capabilities. Instead of fine-tuning all parameters, researchers have developed Parameter-Efficient Fine-Tuning (PEFT) methods, such as Low-Rank Adaptation (LoRA) [4] and Prompt Tuning [6], to reduce computational costs while preserving adaptability. LoRA, as shown in Fig. 1b, approximates linear layers with low-rank matrices to minimize trainable parameters. Meanwhile, Prompt Tuning (Fig. 1c) introduces task-specific prompts into the model input, optimizing performance with minimal adjustments.

An emerging approach, Prompt Pools, enhances prompt tuning by storing multiple prompt tokens and dynamically selecting the most relevant ones during inference, as illustrated in Fig. 1d. Inspired by these advancements, this paper investigates the role of prompt tuning in 3D object detection, exploring whether a model trained on the Waymo dataset can serve as a foundation model for other domains. We systematically analyze prompt tokens and prompt generators, ultimately proposing the Scene-Oriented Prompt Pool (**SOP**$^2$) to improve domain adaptation. Our work demonstrates the effectiveness of prompt-based adaptation in 3D object detection, providing insights for future research on leveraging LLM-inspired techniques in 3D vision.

*Corresponding author

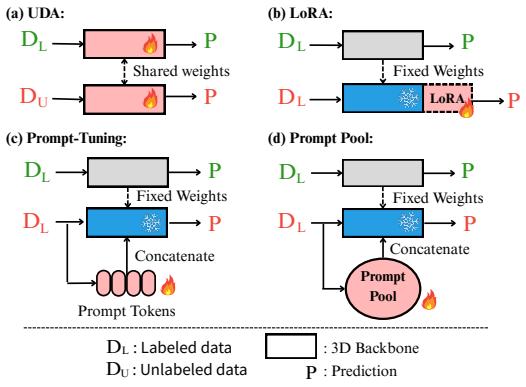



Figure 1: Comparison of various transfer learning methods: (a) Unsupervised Domain Adaptation, (b) Low-Rank Adaptation, (c) Prompt Tuning and (d) Prompt Pool. Green text represents the source domain, while red text represents the target domain.

## 2. Related Work

### 2.1. Parameter-Efficient Fine-Tuning

Recent research in fine-tuning has focused on Parameter-Efficient Fine-Tuning (PEFT) to adapt pretrained models more effectively, as traditional fine-tuning often incurs high training costs. Two prominent techniques, Prompt-Tuning and LoRA, have gained attention for their efficiency in adapting pretrained models to specific tasks. Prompt-Tuning fine-tunes models by introducing task-specific prompts, enabling adaptation with minimal supervision. It guides the model to focus on relevant information for the target task, improving performance while preserving pretrained benefits. On the other hand, LoRA addresses the computational challenges of fine-tuning large models by using low-rank approximations of weight matrices, reducing the parameter space and enabling more efficient fine-tuning with faster convergence, making it suitable for resource-constrained environments.

### 2.2. Prompt-Tuning in CV

While Prompt-Tuning traditionally involves providing task-specific prompts to guide the fine-tuning of pretrained language models, visual prompt tuning (VPT) [5], extends this idea to the realm of visual data. Instead of text prompts, VPT utilizes visual cues or soft prompts to guide the adaptation of pretrained vision models for specific tasks. This approach offers a promising avenue for leveraging pretrained representations in CV tasks while allowing for task-specific adaptation with minimal labeled data. Dynamic visual prompt tuning (DVPT) [9] further extends this approach by dynamically generating prompts based on the input data. Unlike VPT, where prompts remain fixed throughout the training process, prompts in DVPT adapt based on the characteristics of the current input, allowing for more flexible and context-aware model behavior.

### 2.3. Transformer for 3D Perception

Transformers [13] have gained significant traction in computer vision, particularly for 3D perception tasks. Votr [8], a Transformer-based model, excels at extracting 3D features by capturing global dependencies between voxels in point clouds. However, its dense architecture incurs high computational costs, especially when scaling to large 3D datasets. To mitigate this, Window Attention techniques [7] have emerged, offering a more efficient approach to 3D perception. Building on this, SST [1] introduces a locality-aware spatial attention mechanism that dramatically reduces computational overhead, while SWFormer [11] further refines this by incorporating a sliding window-based mechanism to strike a balance between accuracy and efficiency. More recently, DSVT [15] has taken sparse transformer models to new heights by dynamically generating sparse voxel representations, enabling efficient modeling of global dependencies while significantly lowering computational costs. This approach has pushed the boundaries of Transformer-based 3D perception, showcasing its vast potential.

## 3. Proposed Method

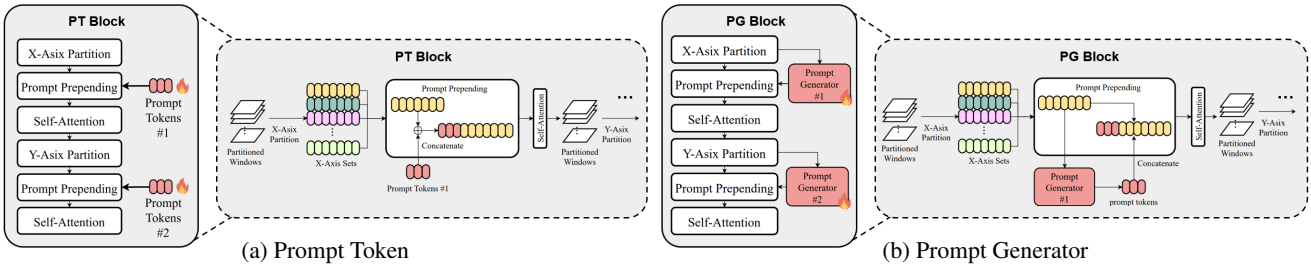
### 3.1. Prompt Friendly Framework

DSVT is well-suited for prompt-based 3D perception due to its efficient sparse data handling and dynamic attention mechanism, reducing computational overhead and improving scalability. By dynamically adjusting attention per prompt, it enhances contextual understanding across scene components. We adopt a pillar-based DSVT framework, where the input point cloud is processed through a voxel feature encoding (VFE) module [17], followed by max-pooling along the voxel dimension. The scene is partitioned into windows, with nonzero voxels evenly distributed along the X/Y axes to form non-overlapping sets. Each DSVT block applies two set partitions: one along the X-axis and one along the Y-axis. The $j$-th partition is defined as:

$$\boldsymbol{S}_j = \{S_i\}_{i=1}^{N},\ j = 1, ..., Total\ \#\ of\ Partitions, \quad (1)$$

where $S_i \in \mathbb{R}^{n_s \times C}$ represents a set of $n_s$ voxels with feature dimension $C$, and $N$ denotes the total number of sets, depending on voxel sparsity. The full partition is expressed as $\boldsymbol{S}_j \in \mathbb{R}^{N \times n_s \times C}$. To enable information exchange within each set, multi-head self-attention (MHSA) is applied.

We consider $N$ as the batch size, treating voxels as tokens. By leveraging multi-scale window partitioning and directional set partitioning, DSVT facilitates efficient token interaction, enhancing 3D scene understanding.




(a) Prompt Token (b) Prompt Generator

Figure 2: The architecture of (a) PT block: Adding prompt token to $\boldsymbol{S}_j$ and (b) PG block: Adding prompt generator to $\boldsymbol{S}_j$.

### 3.2. Scene Analysis

Unlike previous task-oriented prompt-tuning approaches such as [6, 14], which focus on adapting pre-trained models for downstream tasks, we address the shift from 2D image classification to 3D point cloud object detection. This shift has increased data volume, and existing pre-trained models are less powerful and generalizable than large LLMs. We propose a scene-oriented prompt approach, where prompt tokens focus on learning domain-specific information to capture the unique characteristics of the target domain. We further analyze the scene using t-SNE [12] on partitioned sets $\boldsymbol{S}_j$, observing distinct distribution patterns, as shown in Fig. 4a. This suggests the need for independent prompt tokens for each partition, rather than a single generic prompt, ensuring better alignment with the specific characteristics of each data subset and enhancing the model's ability to capture domain-specific features.

### 3.3. Prompt Token

In the VPT framework, a group of prompt tokens are concatenated to the embedding patches prior to the model input. During the fine-tuning process, only the prompt tokens are updated, while the backbone model is kept frozen. By adopting this approach, the prompt tokens are able to learn the information relevant to the downstream tasks, while the backbone model retains the original source domain knowledge. Taking inspiration from VPT, in Fig. 2a, for each set partition $\boldsymbol{S}_j$, we assign a collection of prompt tokens $PT_j$ as follows:

$$PT_j,\ j = 1, ..., Total\ \#\ of\ Partitions, \tag{2}$$

where $PT_j \in \mathbb{R}^{n_T \times C}$ is the prompt tokens corresponds to $\boldsymbol{S}_j$, $n_T$ is the number of prompt. Then we cooperate prompt and scene as follows:

$$\boldsymbol{S}_j^* = \{[PT_j; S_i]\}_{i=1}^N = \{S_i^*\}_{i=1}^N, \tag{3}$$

where ; denoted concatenation along the dimension of token numbers. we cooperate the prompt token $PT_j$ with each set $S_i$ in $j$-th set partition to obtain the prompted sets $S_i^*$, where $S_i^* \in \mathbb{R}^{(n_T+n_s)\times C}$, $n_T$ is the prompt number of $PT_j$. Then, we conduct the MHSA as mentioned in Section 3.1 on the prompted set partition $\boldsymbol{S}_j^*$

In Section 4.2, we found that simply using a prompt token has limited effectiveness.

### 3.4. Prompt Generator

Since providing a simple prompt for each set may not be sufficient for the model, we further draw inspiration from DVPT to enhance prediction performance. Instead of a simple prompt token, a prompt generator is employed to generate a prompt token of the same size as the set token, ensuring that each prompt can refer to the current input scene. As shown in Fig. 2b, we design a prompt generator $f^G : \mathbb{R}^{n_s \times C} \rightarrow \mathbb{R}^{n_G \times C}$ to dynamically produce prompt tokens corresponding to $\boldsymbol{S}_j$. We denote the $j$-th prompt generator as:

$$f_j^G,\ j = 1, ..., Total\ \#\ of\ Partitions \tag{4}$$

The sets $S_i$ in $j$-th set partition are fed into $f_j^G$ to get the set-wise dynamic prompts $P_i^G$, where $P_i^G \in \mathbb{R}^{n_G \times C}$, $n_G$ is the prompt number of $P_i^G$. Then we cooperate the set-wise prompts and scene as follows:

$$\boldsymbol{S}_j^* = \left\{\left[f_j^G(S_i); S_i\right]\right\}_{i=1}^N = \left\{\left[P_i^G; S_i\right]\right\}_{i=1}^N = \{S_i^*\}_{i=1}^N, \tag{5}$$

where the prompted sets $S_i^* \in \mathbb{R}^{(n_G+n_s)\times C}$. In Section 4.2, we found that even with the addition of a prompt generator to dynamically generate prompts, it may still be insufficient for 3D scenes.

### 3.5. Scene-Oriented Prompt Pool (SOP$^2$)

After the aforementioned attempts of prompt token and prompt generator, it was observed that using prompts generated based on the scene is more effective than general prompts. We further draw on the concept of the prompt pool from L2P [16] by storing prompt tokens in a prompt pool, allowing the current scene to select the most suitable prompt token for use. Building upon the previous attempts and the concept of L2P. As shown in Fig. 3, we conducted a prompt pool $PP_j$ with the corresponding $\boldsymbol{S}_j$, we denoted the prompt pool as follows:

$$PP_j = \{(k_m, P_m)\}_{m=1}^M,\ M = size\ of\ prompt\ pool, \tag{6}$$

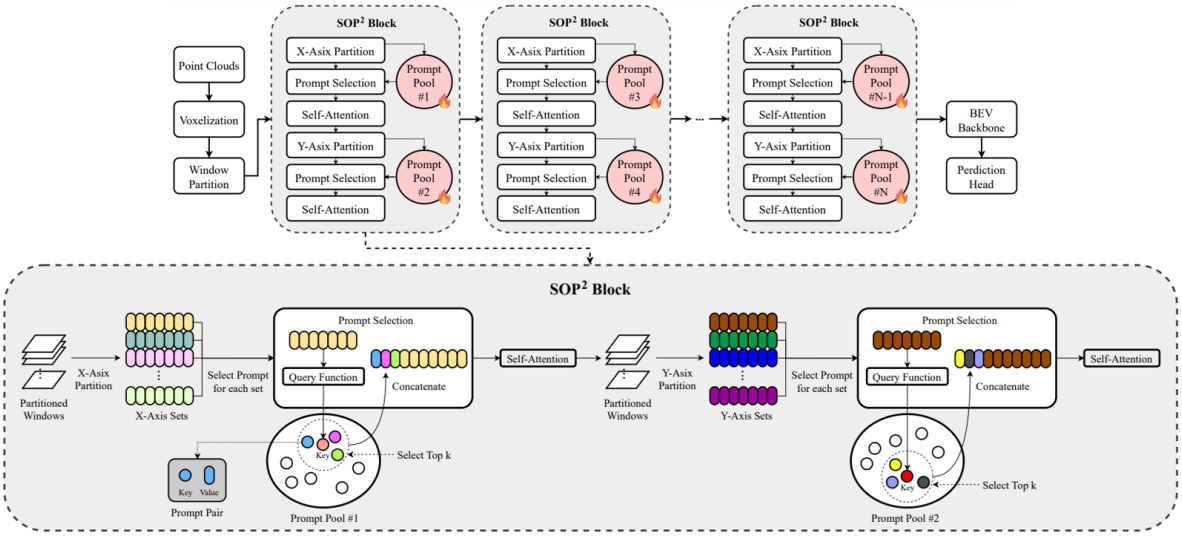



Figure 3: The overall architecture of our proposed SOP$^2$. The upper part shows the overall pipeline, while the lower part illustrates a single SOP$^2$ block. Each set partition is assigned a corresponding prompt pool, allowing the set to select suitable prompt tokens from the prompt pool.

where $k_m$ and $P_m$ are key-value pair, $k_m \in \mathbb{R}^C$ and $P_m \in \mathbb{R}^{n_P \times C}$, $n_P$ is the length of prompt. We aim to dynamically select the most suitable prompts from the prompt pool. and cooperate these prompts with $S_i$ which is set-wise. The selection function of prompt pool is as follows:

$$f^s(PP_j, f^q(S_i)) = P_i^P, \ where \ P_i^P \in \mathbb{R}^{K \times n_P \times C} \quad (7)$$

First, we use a query function $f^q : \mathbb{R}^{n_s \times C} \rightarrow \mathbb{R}^C$ to project the input set to the same dimension of key, the output of $f^q$ is the query key, then $f^s$ will score the similarity of all $k_m$ in prompt pool $PP_j$ and select the top-$K$ similar keys with the query key. Finally, collect the corresponding value of the top-$K$ key to construct the set-wise prompt $P_i^P$. Then we integrate the set-wise prompts with the scene to obtain the prompted set $S_i^* \in \mathbb{R}^{(K \times n_p + n_s) \times C}$ as follows:

$$\begin{aligned} \boldsymbol{S}_j^* &= \{[f^s(PP_j, f^g(S_i)); S_i]\}_{i=1}^N \\ &= \left\{\left[P_i^P; S_i\right]\right\}_{i=1}^N = \{S_i^*\}_{i=1}^N, \end{aligned} \quad (8)$$

# 4. Experiments Results

## 4.1. Experimental Setup

**Dataset Preparation.** The experimental section uses the DSVT pillar version with pre-trained weights from the Waymo dataset to evaluate prompt tuning on the KITTI dataset. The KITTI dataset includes 3,712 *train*, 3,769 *val*, and 7,518 *test* samples, commonly used for 3D object detection. In comparison, the Waymo Open Dataset is much larger, with 158,361 *train*, 40,077 *val*, and 40,832 *test* samples, collected across diverse locations and weather conditions, providing a more generalized dataset for model development.

**Evaluation Metrics.** We use the official KITTI evaluation metric to assess our method, based on mean Average Precision (mAP) with a rotated Intersection over Union (IoU) threshold. For cars, the threshold is 0.7, and for pedestrians and cyclists, it is 0.5. The mAP is calculated on the validation set with 40 recall positions, providing a comprehensive measure of object detection accuracy, including localization and classification.

**Implementation Details.** We utilize the same learning rate scheme as [17] and strictly follow the DSVT setup. The backbone consists of four DSVT blocks, each containing two set partitions (X and Y Axis). We adopt a grid size of (0.32m, 0.32m, 6m) and hybrid window sizes of (12, 12, 1) and (24, 24, 1). The maximum number of voxels assigned to each set, $n_s$, is set to 36. The voxel feature map is downsampled using standard max-pooling along the Z-Axis. All attention modules utilize 192 input channels. In Section 3.3, $n_T$, the number of prompt tokens $PT_j$, is set to 1. In Section 3.4, the prompt generator $f^G$ is a 4-layer MLP, and the number of prompts $n_G$ is set to 1. In Section 3.5, the query function $f^q$ is a max pooling function, and $f^s$ uses cosine similarity to score the query and prompt key.

## 4.2. Main Results

As shown in Table 1, we first train the DSVT model from scratch on the KITTI dataset as the baseline. We then fine-tune the model using Waymo pre-trained weights and explore two approaches: **Full Fine-Tuning** (updating all weights) and **Head Fine-Tuning** (updating only the Prediction Head to avoid overfitting). We also investigate **BitFit** (updating only the backbone's bias terms) and **LoRA**, a PEFT method using low-rank approximation to reduce

Table 1: Performance comparison on the KITTI *val* set. The results are evaluated by the mean Average Precision with 40 recall positions. Best results among all methods except Full Fine-tune and From Scratch are **bolded**.

| Method | Car | | | Pedestrian | | | Cyclist | | | Trainable |
|---|---|---|---|---|---|---|---|---|---|---|
| | Easy | Mod. | Hard | Easy | Mod. | Hard | Easy | Mod. | Hard | Params |
| From Scratch | 90.08 | 81.88 | 80.22 | 53.47 | 48.54 | 45.20 | 92.68 | 70.91 | 66.65 | 8.65M |
| Head Fine-tune | 88.51 | 78.87 | 77.45 | 57.81 | 51.69 | 47.57 | 82.57 | 63.57 | 60.47 | 0.45M |
| Full Fine-tune | 89.85 | 83.32 | 81.19 | 60.44 | 55.81 | 52.04 | 93.81 | 72.25 | 68.23 | 8.65M |
| BitFit [18] | 88.14 | 79.94 | 79.28 | 59.97 | 54.70 | 50.13 | 86.00 | 66.67 | 63.49 | 0.47M |
| LoRA [4] | 89.00 | 80.90 | 79.45 | 56.66 | 51.87 | 49.11 | 86.82 | 66.96 | 63.42 | 0.47M |
| Prompt Token [5] | 88.55 | 78.78 | 77.65 | 61.95 | 54.86 | 50.10 | 83.60 | 64.92 | 61.50 | 0.45M |
| Prompt Generator [9] | 89.03 | 80.44 | 79.13 | 65.99 | 59.16 | 54.71 | 83.71 | 66.44 | 63.10 | 1.64M |
| **SOP$^2$ (Ours)** | 90.72 | 82.07 | 80.34 | **67.24** | 61.12 | **56.43** | **86.69** | **68.24** | **64.56** | 0.82M |
| **SOP$^2$ (Ours)**+LoRA | **90.79** | **82.17** | **80.38** | 66.34 | **61.15** | 56.20 | 84.18 | 67.90 | 63.98 | 0.84M |

Table 2: Analysis of adding a prompt pool $PP_j$ to different corresponding set partitions $\boldsymbol{S}_j$.

| $Pa_1$ | $Pa_2$ | $Pa_3$ | $Pa_4$ | $P_5$ | $Pa_6$ | $Pa_7$ | $Pa_8$ | 3D mAP |
|---|---|---|---|---|---|---|---|---|
| ✓ | | | | | | | | 80.34 |
| | ✓ | | | | | | | 79.77 |
| | | ✓ | | | | | | 81.21 |
| | | | ✓ | | | | | 80.65 |
| | | | | ✓ | | | | 81.54 |
| | | | | | ✓ | | | 80.83 |
| | | | | | | ✓ | | 80.64 |
| | | | | | | | ✓ | 80.60 |
| ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | 82.07 |

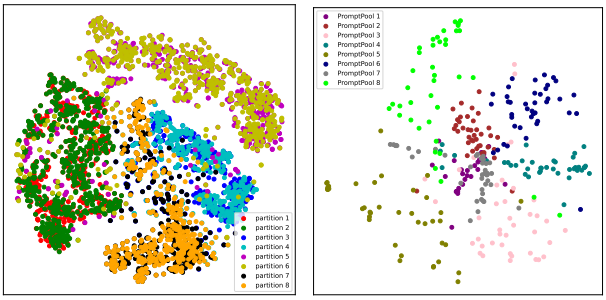
Table 3: Comparison of inference speed, memory usage, parameter count, and FLOPs.

| Method | Inference Time | Memory Peak | Params | FLOPs |
|---|---|---|---|---|
| DSVT | 88.74 ms | 2951.47 MB | 8.65 M | 1460.66 G |
| Ours | 96.64 ms | 2953.23 MB | 9.02 M | 1464.68 G |

trainable parameters in the transformer layers. Additionally, we test a **Prompt Token** method, where prompt tokens store domain-specific information for each set. Despite its simplicity, this method underperforms the From Scratch baseline. To improve upon this, we propose the **Prompt Generator** method (Section 3.4), which dynamically generates prompts based on the scene. This method outperforms the Prompt Token approach by 1.66%, 4.3%, and 1.52% for Cars, Pedestrians, and Cyclists, respectively, under moderate conditions.

We also introduce **SOP$^2$** (Section 3.5), inspired by the L2P prompt pool concept. SOP$^2$ uses prompt pools for each set partition, enabling the model to select the most suitable prompts for the current scene. SOP$^2$ outperforms the Prompt Generator by 1.63%, 1.96%, and 1.80% for the three moderate classes, and the Prompt Token method by 3.29%, 6.26%, and 3.32%. Compared to LoRA, SOP$^2$ shows a significant advantage of 1.17%, 9.25%, and 1.28%. Finally, when combining SOP$^2$ with LoRA, the results demonstrate a 0.1% improvement in Car class performance compared to SOP$^2$ alone, indicating effective synergy between SOP$^2$ and PEFT methods.

Regarding parameter count, we report the number of trainable parameters, excluding frozen weights, including the 0.45M parameters in the detection head. SOP$^2$ uses only 0.82M parameters (9% of Full Fine-Tuning), which is 0.35M more than LoRA but achieves over 1% higher accuracy, showing an excellent trade-off between parameter count and performance.




(a) Set Partitions $\boldsymbol{S}_j$ (b) Prompt Pools $PP_j$

Figure 4: For the visual representation of t-SNE. (a) Distribution of different set partitions $\boldsymbol{S}_j$, where different colors represent different partitions, and (b) Distribution of different prompt pools $PP_j$ corresponding to set partitions $\boldsymbol{S}_j$, where different colors represent different prompt pools.

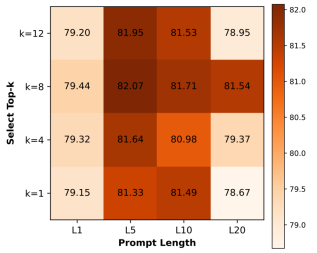



Figure 5: Comparison of 3D mAP with 40 recall positions for Prompt Length $n_P$ and Select top-$K$ when the prompt pool size M = 40.

### 4.3. Effectiveness of Prompt Pool

**Prompt Pool Position.** Section 3.2 discusses t-SNE analysis on set partitions $\boldsymbol{S}_j$, revealing distinct distributions, as shown in Figure 4a. To capture partition-specific information, we assign a prompt pool to each partition. Further t-SNE analysis of the prompt pool values in Figure 4b confirms that each pool encodes distinct information. Table 2 shows that odd-numbered partitions benefit more from the prompt pool, suggesting that X-axis partitions contribute more effectively than Y-axis partitions.

**Hyperparameters of Prompt Pool.** We first analyze the prompt pool size, as shown in Fig. 6a, with $M$ ranging from 20 to 50. The optimal performance is achieved at $M = 40$, suggesting that an appropriately sized pool is crucial for effective information accommodation. Next, we examine the relationship between prompt length $n_P$ and top-$K$ selection for $M = 40$, as illustrated in Fig. 5. The best performance occurs at $n_P = 5$ and $K = 8$, with performance degrading for larger or smaller values, potentially due to underfitting or overfitting influenced by the target domain size.

**Complexity**. Table 3 compares the baseline (DSVT) with

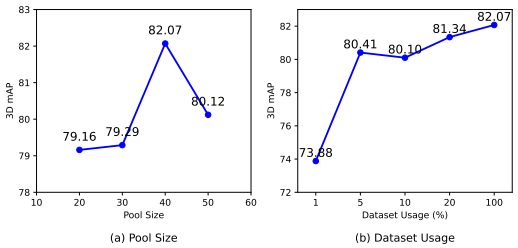



Figure 6: Comparison of 3D mAP with 40 recall positions: (a) Effect of different pool sizes M, and (b) Effect of the quantity of the target dataset (KITTI) used for training SOP$^2$.

our model, showing that adding the prompt pool causes only a slight increase in complexity. Inference time was measured on an Intel(R) Xeon(R) Silver 4210R CPU @ 2.40GHz and an NVIDIA RTX 3090 GPU.

**Target Domain Dataset Usage.** We analyzed the effect of the target domain data size (KITTI) on SOP$^2$ performance, as shown in Fig. 6b. Using 1%, 5%, 10%, 20%, and 100% of the KITTI training set (3712 samples), the results show that with only 1% of the data, the 3D mAP is around 73.6, indicating underfitting. As the data usage increases, performance improves, with a sharp increase to 80.41 at 5%. At 10%, the accuracy slightly drops to 80.10, but further increases to 81.34 at 20% and reaches 82.07 at 100%, demonstrating continuous performance enhancement.

## 5. Conclusion

This paper investigates the feasibility of prompts in 3D perception, extending prompt tuning from LLMs and 2D vision to the 3D domain. Unlike previous **Task-Oriented** prompt tokens, we focus on **Scene-Oriented** prompts that encode scene-specific information. We study Prompt Tokens and Generators, proposing **SOP$^2$**, which dynamically selects the best prompt from multiple pools for each scene. **SOP$^2$** outperforms common fine-tuning and PETF methods, paving the way for future 3D prompt research.

## References


[1] L. Fan, Z. Pang, T. Zhang, Y.-X. Wang, H. Zhao, F. Wang, N. Wang, and Z. Zhang. Embracing single stride 3d object detector with sparse transformer. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8448–8458, 2022.

[2] L. Floridi and M. Chiriatti. Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30:681–694, 2020.

[3] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012.

[4] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[5] M. Jia, L. Tang, B.-C. Chen, C. Cardie, S. Belongie, B. Hariharan, and S.-N. Lim. Visual prompt tuning. In *European Conference on Computer Vision*, pages 709–727. Springer, 2022.

[6] B. Lester, R. Al-Rfou, and N. Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

[7] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.

[8] J. Mao, Y. Xue, M. Niu, H. Bai, J. Feng, X. Liang, H. Xu, and C. Xu. Voxel transformer for 3d object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3164–3173, 2021.

[9] C. Ruan and H. Wang. Dynamic visual prompt tuning for parameter efficient transfer learning. In *Chinese Conference on Pattern Recognition and Computer Vision (PRCV)*, pages 293–303. Springer, 2023.

[10] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2446–2454, 2020.

[11] P. Sun, M. Tan, W. Wang, C. Liu, F. Xia, Z. Leng, and D. Anguelov. Swformer: Sparse window transformer for 3d object detection in point clouds. In *European Conference on Computer Vision*, pages 426–442. Springer, 2022.

[12] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[14] T. Vu, B. Lester, N. Constant, R. Al-Rfou, and D. Cer. Spot: Better frozen model adaptation through soft prompt transfer. *arXiv preprint arXiv:2110.07904*, 2021.

[15] H. Wang, C. Shi, S. Shi, M. Lei, S. Wang, D. He, B. Schiele, and L. Wang. Dsvt: Dynamic sparse voxel transformer with rotated sets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13520–13529, 2023.

[16] Z. Wang, Z. Zhang, C.-Y. Lee, H. Zhang, R. Sun, X. Ren, G. Su, V. Perot, J. Dy, and T. Pfister. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 139–149, 2022.

[17] T. Yin, X. Zhou, and P. Krahenbuhl. Center-based 3d object detection and tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11784–11793, 2021.

[18] E. B. Zaken, S. Ravfogel, and Y. Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*, 2021.