Task-Oriented Token Pruning for Efficient Object Detection and Segmentation

Hao Liang[1], Meina Kan[1], Shiguang Shan[1,2], and Xilin Chen[1]

Abstract— **Robots rely heavily on visual perception to understand and interact with complex environments. To support this capability, modern perception models have become increasingly large and powerful, resulting in high computational costs that hinder their real-time performance in robotic applications. Existing acceleration techniques, such as model pruning and token pruning, focus on reducing architectural or parameter redundancy but still process all object categories, regardless of task requirements. However, in real-world robotic scenarios, different tasks typically require only a subset of object categories. For instance, a service robot may focus on kitchenware while cooking, but shift to furniture and obstacles while cleaning. This task-dependent variation creates opportunities to reduce computational cost by selectively processing relevant information. Existing methods are not designed to exploit this potential for task-specific efficiency. To address this limitation, we propose TaskTP, a task-oriented token pruning method that dynamically adjusts token pruning based on the target category set. A dynamic gating network is introduced between successive Transformer blocks, which evaluates the relevance of each token to the given task. TaskTP allows for more aggressive pruning when fewer categories are required, optimizing computation without sacrificing performance. After a task-agnostic training phase, it can be flexibly configured at deployment time to support any category subset without retraining, making it both efficient and versatile. TaskTP improves the performance of Mask R-CNN from 31.4 fps to 38.5 fps on the COCO dataset. Furthermore, on the ScanNet dataset, where an object search task was defined to simulate real-world robotic applications, processing time was reduced from 3197 ms to 2437 ms, demonstrating significant efficiency gains.**

I. Introduction

In robotics, visual perception is critical for enabling robots to understand and interact with environments. Core tasks such as object detection and segmentation allow robots to identify objects, understand spatial layouts, and make decisions in scenarios like navigation, manipulation, and human-robot interaction. As robots become more autonomous and operate in increasingly complex environments, the demand for versatile and accurate perception models continues to grow. To meet these needs, state-of-the-art models have grown substantially in both size and complexity, enhancing their capacity to handle extensive object categories. For example, the widely used Faster R-CNN [1] model with ResNet-50 [2] backbone was proposed in 2016, contains 44.2 million parameters and can detect 80 categories. In contrast, the DINO network with an Intern-H [3] backbone scales to 2.18 billion parameters to recognize 1203 categories, representing a 49-fold increase in parameters and a 15-fold increase in categories.

However, robotic platforms face challenges that make traditional large-scale perception models impractical, including limited computational power, restricted battery capacity, and the need for real-time decision-making. Moreover, the increasing resolution of sensors produces more detailed images, which, although beneficial, place further strain on the processing costs of the system.

To address the computational challenges, token pruning has emerged as an effective acceleration strategy [4], [5], [6]. By identifying and discarding less informative tokens during inference, these methods reduce computation while maintaining accuracy. Despite their success, these acceleration methods typically process all object categories present in the dataset, regardless of the specific requirements of the task at hand. As the number of categories increases, the model must retain more tokens and parameters to support recognition across all possibilities, thereby limiting the achievable efficiency improvements.

However, in real-world robotic applications, the set of relevant categories often varies depending on the task. For instance, a service robot preparing a meal may need to focus on kitchen utensils and appliances, while the same robot performing cleaning duties might instead attend to furniture, floors, and obstacles. This task-specific variation in category relevance is both common and meaningful—it introduces a natural opportunity to prune tokens unrelated to the current task, significantly reducing unnecessary computation without compromising task success.

Unfortunately, existing token pruning methods are not designed to take advantage of this flexibility. They assume a fixed category set and lack mechanisms to dynamically adapt pruning behavior at deployment. This becomes especially problematic for general-purpose robots, which must handle diverse and dynamically changing tasks where relevant categories vary significantly.

To address this limitation, we propose TaskTP, a task-oriented token pruning method that supports dynamic configuration for any target category set at deployment time. Unlike existing approaches that must be re-optimized for each task, TaskTP is trained once in a task-agnostic manner and can be flexibly configured to prioritize different category

This work was partially supported by the National Natural Science Foundation of China (Nos. 62495082 and 62461160331)

[1]Authors are with the State Key Laboratory of AI Safety, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 100190, China, and also with the University of Chinese Academy of Sciences, Beijing 100049, China e-mail: lianghao21s@ict.ac.cn; kanmeina@ict.ac.cn; sgshan@ict.ac.cn; xlchen@ict.ac.cn

[2]Shiguang Shan is also with Peng Cheng National Laboratory, Shenzhen, 518055, China

subsets required by diverse task demands within a single robotic system. This adaptability is enabled by a lightweight gating network inserted between Transformer blocks, which leverages externally provided task category information to assess token relevance and generate binary pruning masks. By pruning irrelevant tokens while retaining those critical for the current task, TaskTP achieves efficient, task-specific acceleration within a single unified model. This offers a scalable solution for real-world scenarios where task requirements vary widely.

In summary, our contributions are as follows:

- The proposal of a novel task-oriented token pruning method, TaskTP, that selectively prunes tokens associated with irrelevant categories to dynamically improve visual perception efficiency based on the specific task requirements, all through a single, task-agnostic training process that can be applied to any task.
- A dynamic gating network is introduced to evaluate the relevance of each token to a given task and generate a mask that determines which tokens to preserve or prune, allowing for efficient task-specific pruning without the need for task-specific retraining.
- The proposed method significantly accelerates object detection and segmentation models, such as boosting Mask R-CNN from 31.4 fps to 38.5 fps on COCO and reducing processing time from 3197 ms to 2437 ms for the object search task on ScanNet. When all categories are required, TaskTP reverts to standard token pruning, making it an effective and adaptable solution.

II. Related Work

Transformers [7], [8], [9], [10] have achieved remarkable success in various visual tasks. However, their growing size and computational demands have prompted research into acceleration strategies, including model pruning, distillation, and token pruning. Existing approaches include model pruning, distillation, and token pruning.

Model Pruning Techniques. Model pruning reduces network size by removing less important parameters, typically based on weight magnitude [11], [12], weight sensitivity [13], [14], or their influence on model output [15], [16]. The process is repeated iteratively, resulting in a more efficient model with reduced computational demands.

Distillation Techniques. Knowledge distillation (KD) transfers knowledge from a larger model (teacher) to a smaller one (student) to accelerate inference [17]. KD methods include logit-based approaches [18], [19], which align output distributions, and feature-based methods [20], [21], [22], which match intermediate feature representations.

Token Pruning Techniques. Token pruning methods aim to accelerate Vision Transformers by reducing the number of tokens at each layer. In classification tasks, tokens are pruned based on intermediate predictions [23], [24], [4] or attention weights [25], [26]. The pruned tokens can either be discarded [25], [23] or fused into a single token [23], [27].

In dense tasks such as detection and segmentation, token pruning requires careful consideration to preserve spatial granularity. Token halting approaches [28], [29], [30], [5] skip further processing for confident tokens in early layers, while still using them in the final output. DoViT [29] gradually stops partial easy tokens from self-attention calculation and keeps the hard tokens forwarding. In addition to pruning the easy tokens, DToP [30] also retains some of the tokens with the highest centrality to maintain the representative context information. SViT [5] offers insights such as dynamic pruning rates, reactivating pruned tokens, and using a simple MLP for pruning.

Token merging methods reduce the number of tokens by clustering similar or redundant tokens. ToME [6] proposed a lightweight matching algorithm to find redundant pairs of tokens fast. ALGM [31] merges tokens in two stages: locally in the first layer and globally in the middle, which enhances throughput. ATC [32] adopts bottom-up hierarchical clustering without extra learnable parameters. ELViT [33] clusters tokens to reduce resolution and applies a reconstruction layer to restore high resolution features.

Most existing token pruning methods assume a fixed set of categories and are not designed to adapt to varying task demands without retraining. In contrast, our method enables dynamic token pruning for any subset of categories without retraining, allowing efficient task-specific acceleration.

III. Methodology

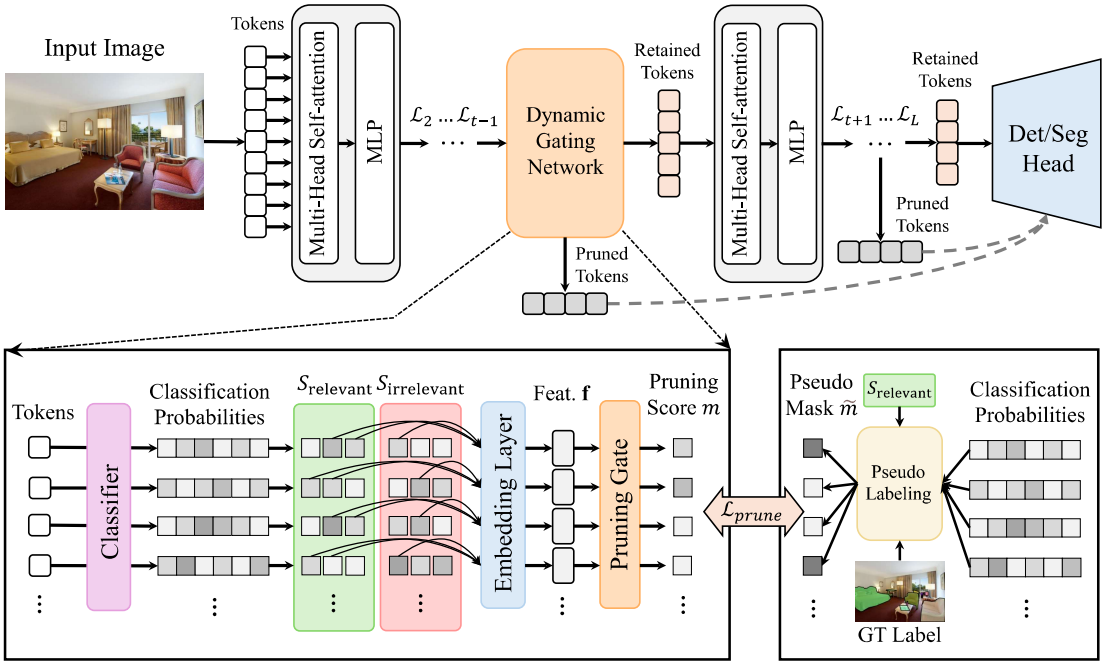
In this section, we introduce our task-oriented token pruning method TaskTP, which enhances the computational efficiency of Vision Transformers by dynamically pruning tokens based on a given task. The method consists of two key components: a dynamic gating network and a pseudo-labeling strategy for token-task relevance. The dynamic gating network determines which tokens to prune for a given task, while the pseudo-labeling strategy generates supervisory signals for token-task relevance to guide the optimization of the gating network. Fig. 1 illustrates the workflow of our method. The following subsections describe these components in detail.

A. Dynamic Gating Network

The dynamic gating network is inserted between successive Transformer blocks to dynamically prune tokens based on their relevance to the current task. Specifically, the network considers both the token's features from previous blocks and the task-specific information to assign a pruning score for each token, which indicates its relevance to the task at hand. Tokens with low scores are pruned as they do not significantly contribute to task objectives. Specifically, each token x_i is first encoded to produce a representation $\mathbf{f}_i$ that better captures its relevance to the task. Then, a gating network M predicts a pruning score m_i for each token.

First, each token x_i is encoded to produce a more informative representation $\mathbf{f}_i$. This is achieved by applying a feature constructor F, which processes the token x_i along with the task-relevant category set S_{relevant} as follows:

$$\mathbf{f}_i = F(x_i, S_{\text{relevant}}). \tag{1}$$



(a) Dynamic Gating Network

(b) Pseudo-Labeling Strategy for Token-Task Relevance

Fig. 1: Overview of our Task-Oriented Token Pruning method TaskTP. The top diagram illustrates how the dynamic gating network works within the ViT architecture. Positioned between two successive Transformer blocks, the dynamic gating network selectively prunes task-irrelevant tokens while allowing task-relevant tokens to bypass pruning. The two subfigures below detail the design of the dynamic gating network and the pseudo-labeling strategy for generating pruning labels.

S_{relevant} represents the categories relevant to the current task T, which is typically a subset of all training categories $S = \{1, \dots, C\}$, i.e. $S_{\text{relevant}} \subseteq S$. In practical applications, S_{relevant} can be obtained simply by a large language model or predefined rules from the task T at hand.

The probabilities predicted by a classifier can provide a direct indication of a token's relevance to specific categories. These probabilities can also be used to determine whether a token belongs to a relevant or irrelevant category, helping to decide whether it can be pruned. We therefore incorporate a classification network G to predict the probability distribution for each token x_i, denoted as p_i:

$$p_i = [p_i^1, p_i^2, \dots, p_i^C] = G(x_i), \tag{2}$$

where C is the total number of categories. This probability distribution provides an initial assessment of each token's relevance to the categories. To robustly represent the relevance of a token in both relevant and irrelevant categories, we select the top k categories with the highest classification probabilities among the task-relevant category set S_{relevant}, denoted as $r_1 \dots r_k$, and the top k categories from the irrelevant category set $S_{\text{irrelevant}}$, denoted as $s_1 \dots s_k$. By focusing on these top k categories, we mitigate the impact of varying set sizes in S_{relevant} and $S_{\text{irrelevant}}$, while also eliminating noise from low-probability categories. These probabilities of selected categories are then combined to construct a probability representation:

$$\mathbf{f}_i^{\text{prob}} = [p_i^{r_1}, \dots, p_i^{r_k}, p_i^{s_1}, \dots, p_i^{s_k}]. \tag{3}$$

To address the potential disparity in the probability distributions across different categories produced by the classification network G, we introduce a learnable embedding e_j for each category j. These embeddings serve as biases to mitigate disparity and balance the classification probabilities. Along with the top k relevant categories $r_1, \dots, r_k$ and the top k irrelevant categories $s_1, \dots, s_k$, a corresponding bias representation is obtained as:

$$\mathbf{f}_i^{\text{bias}} = [e_{r_1}; \dots; e_{r_k}; e_{s_1}; \dots; e_{s_k}]. \tag{4}$$

By concatenating the probability representation and the bias representation, a task-oriented feature representation for token x_i is finally obtained:

$$\mathbf{f}_i = F(x_i, S_{\text{relevant}}) = [\mathbf{f}_i^{\text{prob}}, \mathbf{f}_i^{\text{bias}}], \tag{5}$$

Second, based on the task-oriented feature representation $\mathbf{f}_i$, the pruning score m_i for each token x_i is generated by a light-weight gating network M:

$$m_i = M(\mathbf{f}_i). \tag{6}$$

Here, M is a simple multilayer perceptron (MLP) with two layers. The pruning score m_i lies in the range $[0, 1]$, and the tokens with $m_i < 0.5$ will be pruned.

During training, the Straight-Through Estimator (STE) is used to obtain a binary pruning mask $\tilde{m}_i$. This ensures gradient flow through the binary decision, where $\tilde{m}_i = 1$ indicates that the token x_i should be retained and $\tilde{m}_i = 0$ indicates that the token x_i should be pruned. After obtaining the

pruning mask $\tilde{m}$, the following Transformer block processes the tokens according to this mask. For easy implementation, we set the corresponding columns in the attention matrix to zero, ensuring that pruned tokens do not contribute to the self-attention computations. The output for each token is then calculated as:

$$x' = \tilde{m} \cdot \text{Block}(x, \tilde{m}) + (1 - \tilde{m}) \cdot x, \tag{7}$$

where $x = [x_1, \ldots, x_n]$ represents the concatenation of all input tokens, and $\tilde{m} = [\tilde{m}_1, \ldots, \tilde{m}_n]$ is the corresponding pruning mask for each token.

During inference, following previous work [5], only the retained tokens are passed through the current block, and the output is scattered back into the original feature map to preserve the spatial structure. Since pruned tokens do not enter the computationally expensive self-attention module and feed-forward network, the dynamic gating network significantly reduces the inference time.

B. Pseudo-Labeling Strategy for Token-Task Relevance

In order to optimize the dynamic gating network, it is necessary to access the ground truth of m_i. However, this information is not available in existing datasets. Therefore, we devise a pseudo-labeling strategy to generate a pseudo ground truth $\hat{m}_i$ for token-task relevance m_i. Two key criteria are introduced to determine which tokens should be pruned: (1) irrelevant tokens can be pruned if they are confidently classified into irrelevant categories, and (2) relevant tokens can be early pruned if they are already sufficiently perceived, i.e., classified correctly with high confidence.

First, if a token is confidently identified as belonging to an irrelevant category, it is marked for pruning. To determine this, we use the same classification network G to obtain the predicted category $\hat{y}_i$, and compare it with the true category y_i. If both $\hat{y}_i$ and y_i belong to irrelevant categories—i.e., the token's true label is in the set of irrelevant categories ($y_i \in S_{\text{irrelevant}}$) and is also predicted as a non-target category ($\hat{y}_i \in S_{\text{irrelevant}}$), —the token is marked for pruning. In this case, the token can be pruned regardless whether y_i equals $\hat{y}_i$, since the different irrelevant categories do not need to be distinguished for the current task.

Moreover, if a token belongs to a relevant category and has already been sufficiently perceived, further processing is unnecessary, and thus the token can be pruned. Specifically, if the predicted category matches its true label ($y_i = \hat{y}_i$), the token is marked for pruning.

Finally, in all other cases, the token is retained, as it either cannot be confidently classified as irrelevant or has not been sufficiently perceived.

To summarize, the pseudo-label $\hat{m}_i$ can be calculated as:

$$\hat{m}_i = \begin{cases} 0, & \text{if } \{y_i, \hat{y}_i\} \subseteq S_{\text{irrelevant}} \\ 0, & \text{if } \hat{y}_i = y_i \text{ and } y_i \in S_{\text{relevant}} \\ 1, & \text{otherwise} \end{cases} \tag{8}$$

Using (8), the pseudo-label mask $\hat{m}$ identifies which tokens can be pruned and serves as the ground truth to optimize the predicted mask m from the dynamic gating network. To optimize the pruning process, we employ a binary cross-entropy (BCE) loss between the predicted pruning mask m and the pseudo-label mask $\hat{m}$:

$$\mathcal{L}_{\text{prune}} = -\sum_i \text{BCE}(m_i, \hat{m}_i), \tag{9}$$

C. Objective Function

Our objective function is designed to jointly optimize task performance, token classification, and token pruning by guiding the dynamic gating network with task-relevant pseudo-labels. The objective function comprises three main components: the task loss $\mathcal{L}_{\text{task}}$, which drives the performance of the detection or segmentation task; the classification loss $\mathcal{L}_{\text{class}}$, which is a cross entropy loss used to train the classification network G for relevance assessment; and the pruning loss $\mathcal{L}_{\text{prune}}$, which aligns the dynamic gating network's output with the generated pseudo-labels.

The final objective function combines the task loss, classification loss, and pruning loss:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{task}} + \mathcal{L}_{\text{prune}} + \mathcal{L}_{\text{class}}. \tag{10}$$

This combined loss ensures that the model optimizes for both task performance and efficient computation by retaining only the most relevant tokens.

IV. Experiments

We evaluate our method TaskTP on two benchmarks: COCO [34] and ScanNet [35], comparing it with several state-of-the-art token pruning and merging techniques. The comparison includes SViT [5], EViT [26], EvoViT [36], DyViT [4], ToME [6], and ATC [32]. These methods either remove or merge visual tokens to accelerate inference, while ours supports dynamic, task-specific pruning.

A. Experimental Setup

COCO Dataset. The COCO dataset is a widely used benchmark for object detection and instance segmentation. We use the 2017 split of the COCO dataset, containing 118k training and 5k validation images across 80 categories. All methods are evaluated on object detection and instance segmentation tasks. For TaskTP, we additionally test scenarios with 1, 10, and 40 relevant categories. For each subset size, we randomly sample five category sets and report averaged results. Metrics include FPS (frames per second) and mAP (mean Average Precision) for both bounding box and mask.

ScanNet Dataset. ScanNet contains multiple RGB-D video sequences from indoor scenes with instance-level semantic annotations. Unlike the static nature of COCO images, ScanNet consists of video sequences with temporal dependencies between frames, allowing the evaluation of methods in continuous scenarios. To simulate a service robot application, we design an object search task across 20 randomly selected queries. The robot follows the ScanNet trajectory, capturing an image every 3 seconds. At each step, the segmentation method (ours or baseline) is used to identify the target object. The search stops when the object is

Method	#Categories	DeiT-Tiny Backbone			DeiT-Small Backbone			DeiT-Base Backbone		
		mAP(b)↑	mAP(m)↑	FPS↑	mAP(b)↑	mAP(m)↑	FPS↑	mAP(b)↑	mAP(m)↑	FPS↑
Baseline	80	45.8	40.9	31.4	48.5	42.8	21	48.9	43.3	11.6
SViT	80	45.5	40.7	33.5	48.2	42.5	25.5	-	-	-
EViT	80	44.5	39.8	34.2	47.1	41.6	24.8	-	-	-
EvoViT	80	44.8	39.9	33.2	47.2	41.6	25.1	-	-	-
DyViT	80	44.1	39.3	34.4	47.2	41.6	25.4	-	-	-
ToME	80	43.7	39.3	32.6	46.4	41.4	23.8	46.6	41.1	13.4
ATC	80	45.3	40.5	3.9	47.8	42.3	3.7	48.5	42.8	3.3
TaskTP (ours)	80	45.0	40.3	33.1	47.8	42.2	25.5	48.6	43.0	17.2
TaskTP* (ours)	40	-0.5	-0.8	34.9	-0.8	-0.7	26.1	-0.3	-0.3	18.2
TaskTP* (ours)	10	-0.1	-0.4	36.1	-0.5	-0.1	26.9	-0.6	-0.5	19.1
TaskTP* (ours)	1	-0.8	-0.2	**38.5**	-0.7	-0.7	**28.9**	-0.3	-0.0	**19.8**

TABLE I: Comparison with SOTA methods on the COCO dataset. mAP(b) and mAP(m) represent box mAP in detection and mask mAP in instance segmentation, respectively. #categories represents the size of the relevant category set. '*' represents that only a subset of all categories need to be detected/segmented, in which case the listed mAP(b) and mAP(m) represent the mAP difference between our method and the baseline, rather than the absolute values. *This adjustment is necessary because mAP is calculated only for the relevant categories, and the baseline's mAP on different subsets varies, making absolute mAP comparisons less appropriate. Instead, relative mAP values provide a clearer comparison.* The bold values in the FPS column represent the maximum values.

successfully located. We measure total runtime and number of steps, reflecting real-world efficiency.

Implementation Details. All methods are implemented using a ViT-Adapter architecture [37] with pre-trained DeiT-tiny, DeiT-small and DeiT-base [38] as the backbone within the Mask R-CNN framework [39]. The experiments are conducted on NVIDIA 4090 GPUs with 24 GB of memory. The PyTorch 2.3.1 framework with CUDA 11.8 is used for model training and inference. For our TaskTP, the feature vector $\mathbf{f}_i$ for each token is constructed by selecting the top-2 category probabilities among the relevant and irrelevant categories. The dynamic pruning network is inserted starting from the 4th Transformer layer to the 12th layer. In particular, this addition of the dynamic pruning network introduces approximately only 0.3% additional parameters, minimally impacting the size and speed of the original model. Our TaskTP is finetuned for 6 epochs, the same as SViT, with related category sets randomly constructed.

B. Results on COCO Dataset

The comparative results on the COCO dataset are presented in table I. As shown, existing methods are restricted to detecting all 80 categories, while TaskTP can flexibly focus on any subset of categories, including the full set of 80 categories. In the worst-case scenario, TaskTP is tasked with detecting all 80 categories. Even under this condition, TaskTP achieves performance comparable to existing methods. For example, using DeiT-T as the backbone, TaskTP achieves 45.0% and 40.3%, respectively, with only slight differences of 0.5% and 0.4% compared to the best-performing existing method SViT. With respect to speed, TaskTP reaches 33.1 fps, which is also close to the 33.5 fps of SViT. This demonstrates that TaskTP can degrade to a traditional token pruning method when detecting all categories, while still achieving comparable performance.

When focused on a subset of categories relevant to specific tasks, TaskTP achieves a significant speedup, ranging between 34.9 and 38.5 fps. The speedup becomes increasingly pronounced when focusing on fewer categories. Specifically, when focusing on just one category, TaskTP improves the speed from 31.4 to 38.5 fps. It highlights the practical advantage of TaskTP in task-specific scenarios, where only a subset of categories needs to be detected, enabling a more efficient processing pipeline. For each subset size, we randomly selected five category subsets and averaged the results. Despite the FPS gains when focusing on smaller category subsets, TaskTP maintains competitive accuracy. As shown in Table I, mAPs for object detection and instance segmentation remain high across all subsets. For example, when focusing on a single category, mAP drops only slightly by 0.8% (box mAP) and 0.2% (mask mAP), demonstrating that TaskTP effectively prioritizes relevant tokens with minimal impact on performance.

To evaluate the impact of the slight mAP decrease on detection quality, we calculated precision and recall for both the baseline and TaskTP with 0.5 threshold. The precision and recall differences are negligible. Specifically, precision decreased by 0.4% (from 67.2% to 66.8%), while recall increased by 0.2% (from 73.1% to 73.3%). These results confirm that the efficiency gains from pruning irrelevant tokens come with minimal loss in performance, making TaskTP effective for real-time applications.

C. Results on ScanNet Dataset

In addition to the COCO dataset, we also evaluate all methods in a simplified object search task using the ScanNet dataset, which is similar to robotic applications. As shown in Table II, TaskTP significantly reduces the number of runs of the perception module required to complete the task from 102 to 92, because it identifies the target object earlier. Furthermore, TaskTP reduces the total perception time to 2437 ms, a substantial improvement over the baseline, which takes 3197 ms. This speedup is achieved through cumulative pruning across successive frames. In navigation and search tasks,

Method	DeiT-T Backbone		DeiT-S Backbone	
	#Runs↓	Time↓	#Runs↓	Time↓
Baseline	102	3197	105	4659
SViT	103	2972	101	3784
EViT	102	3035	103	3798
EvoViT	102	3092	103	3761
DyViT	103	2817	101	3637
ToME	105	3084	104	4136
ATC	104	18781	105	19963
TaskTP (ours)	**92**	**2437**	**94**	**3365**

TABLE II: Comparison of number of runs and execution time (count in ms) on ScanNet for DeiT-Tiny and DeiT-Small backbones. The bold values represent the minimum values.

most frames do not contain the target object, allowing for early pruning of most tokens. This demonstrates the practical advantage of TaskTP for real-world robotic applications.

D. Ablation Study

We conducted a series of ablation studies to investigate the contribution of each component in TaskTP, including the number of top categories k used in token representation, the category bias embedding $\mathbf{f}^{\text{bias}}$, the two-stage pseudo-labeling strategy, and the pruning threshold.

	Values of k					
	1	2	3	4	5	10
mAP(box)	44.9	**45.0**	**45.0**	**45.0**	44.8	44.9
mAP(mask)	**40.3**	**40.3**	**40.3**	**40.3**	40.1	40.2
FPS-80	32.4	**33.1**	32.8	32.6	32.4	31.8
FPS-1	38.2	**38.5**	38.4	38.3	38.3	38.1

TABLE III: Comparison of different values of k on COCO dataset with DeiT-T backbone. FPS-80 and FPS-1 represent the FPS when the relevant category set size is 80 and 1, respectively. The bold values represent the maximum values.

a) Effect of Top-k Category Features: We first analyze the effect of varying k in the construction of $\mathbf{f}_i$ using the top k relevant and irrelevant categories in (3) and (4). As shown in Table III, mAP remains stable across different k values, while FPS is more sensitive to this setting. Increasing k from 1 to 2 improves FPS-80 from 32.4 to 33.1 and FPS-1 from 38.2 to 38.5, suggesting that using two category cues helps better distinguish irrelevant tokens. However, increasing k further (e.g., $k = 5$) leads to a drop in speed, likely due to higher feature dimensionality and limited gain in relevance modeling. This suggests that $k = 2$ achieves the best trade-off between efficiency and discriminative power.

	mAP(box)	mAP(mask)	FPS-80	FPS-1
w/o $\mathbf{f}^{\text{bias}}$	45.3	40.5	32.4	37.6
w $\mathbf{f}^{\text{bias}}$	45.0	40.3	**33.1**	**38.5**

TABLE IV: Impact of bias representation $\mathbf{f}^{\text{bias}}$ on performance for the COCO dataset (DeiT-Tiny backbone). FPS-80 and FPS-1 represent the FPS when the size of S_{relevant} is 80 and 1, respectively.

b) Effect of Category Bias Embedding: To assess the importance of the bias embedding $\mathbf{f}^{\text{bias}}$, we remove this component and observe the changes in performance. As shown in Table IV, removing $\mathbf{f}^{\text{bias}}$ leads to a reduction in FPS-80 from 33.1 to 32.4 and in FPS-1 from 38.5 to 37.6, while mAP remains nearly unchanged. This confirms that the bias representation helps guide the model in identifying irrelevant tokens, enhancing pruning efficiency.

Rel	Early	mAP(b)	mAP(m)	FPS-80	FPS-1
		45.8	40.9	31.4	31.4
✓		44.8	40.2	33.0	38.1
✓	✓	45.0	40.3	**33.1**	**38.5**

TABLE V: Impact of the two criteria in the pseudo-labeling strategy on performance for the COCO dataset (DeiT-Tiny backbone). FPS-80 and FPS-1 represent the FPS when the size of the relevant category set is 80 and 1, respectively. "Rel" and "Early" represents the first and second criterion in eq. (8) respectively.

c) Effect of Pseudo-Labeling Strategy: TaskTP employs a two-stage pseudo-labeling strategy: (1) pruning tokens of irrelevant categories, and (2) pruning tokens that have already been adequately perceived. As shown in Table V, applying the first criterion improves FPS-1 by 6.9 and FPS-80 by 1.6. The second criterion provides additional gains of 0.4 FPS (FPS-1) and 0.1 FPS (FPS-80). These results demonstrate that pruning irrelevant-category tokens is the primary contributor to speedup, while early pruning of sufficiently perceived tokens provides further incremental gains.

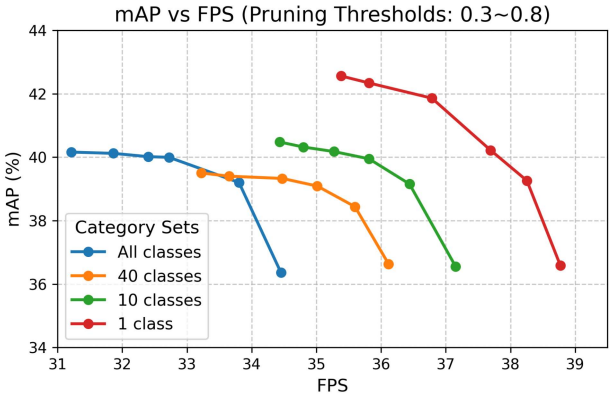


Fig. 2: Impact of pruning threshold on mAP and FPS across different category sets (1 class, 10 classes, 40 classes, and All classes). Higher thresholds generally lead to faster inference but reduced accuracy, with more significant trade-offs when detecting fewer categories.

d) Effect of Pruning Threshold: We also study the impact of varying the pruning threshold, which determines whether a token is retained. As illustrated in Fig. 2, increasing the threshold leads to faster inference but gradually reduces mAP. This trend holds consistently across different category set sizes. Notably, under moderate thresholds (e.g.,

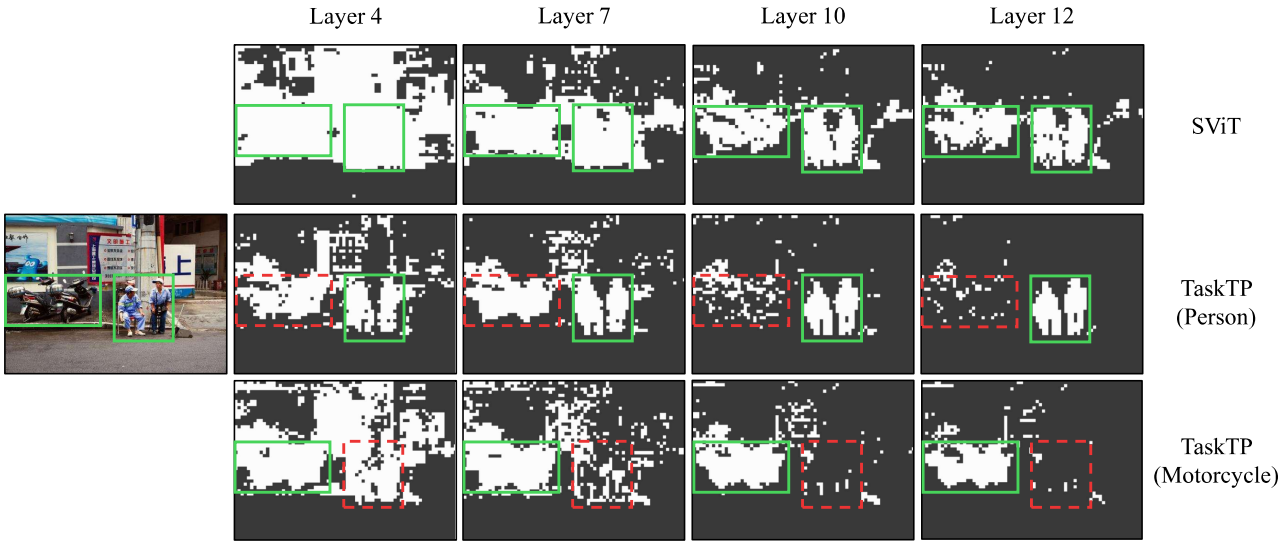


Fig. 3: Comparison of token pruning with TaskTP and the generic SViT [5] across different Transformer layers (4, 7, 10, 12) using DeiT-Tiny. The image on the left is the input image. The first row on the right shows the token pruning mask for SViT, which applies a fixed pruning strategy across all categories. The second and third rows show the token pruning masks for TaskTP, configured to detect only "person" and "motorcycle", respectively. White tokens represent those preserved, while black tokens are pruned. The green solid boxes enclose the instances of relevant categories, while the red dashed boxes highlight irrelevant categories.

below 0.6), TaskTP retains high detection accuracy while achieving significant speedups. This confirms the robustness and adaptability of TaskTP, allowing users to configure the pruning aggressiveness according to application needs and acceptable accuracy trade-offs.

E. Visualization of Token Pruning

We present visualizations to demonstrate how TaskTP prunes tokens across different layers of ViT, comparing it with the generic pruning strategy of SViT. Specifically, we show which tokens are retained and pruned at layers 4, 7, 10, and 12 for different relevant category sets. As shown in Fig. 3, pruned tokens are shown in black, and retained tokens are shown in white.

The first row shows SViT's fixed pruning strategy, which retains tokens for all objects, regardless of relevance. In contrast, TaskTP can be configured after training to adapt its pruning based on the specific categories relevant to the task at hand, as shown in the second and third rows. For example, when "person" is the target, TaskTP prunes tokens related to "motorcycle", and vice versa. This flexibility allows TaskTP to dynamically adjust its pruning, focusing computational resources on the relevant categories and improving efficiency, while still meeting the task's detection requirements.

V. Conclusion and Future Work

In this work, we propose a novel task-oriented token pruning method TaskTP for object detection and instance segmentation tasks. TaskTP reduces computational overhead while maintaining high accuracy. Extensive experiments on both COCO and ScanNet datasets demonstrate significant speedups, particularly when fewer categories are required. As the number of relevant categories decreases, pruning becomes more effective. This method is particularly suited for real-world applications, such as robotic perception, where efficiency is critical. For future work, we aim to extend our method to more diverse tasks, such as those involving natural language descriptions or long-term tasks where the relevant category set changes dynamically throughout the task. These tasks are challenging because extracting the relevant categories from such descriptions is difficult. Exploring how to adapt our method in these dynamic scenarios is a meaningful direction for further research.

References are important to the reader; therefore, each citation must be complete and correct. If at all possible, references should be commonly available publications.

References

[1] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2016.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[3] W. Wang, J. Dai, Z. Chen, Z. Huang, Z. Li, X. Zhu, X. Hu, T. Lu, L. Lu, H. Li, X. Wang, and Y. Qiao, "Internimage: Exploring large-scale vision foundation models with deformable convolutions," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2023, pp. 14 408–14 419.

[4] Y. Rao, W. Zhao, B. Liu, J. Lu, J. Zhou, and C.-J. Hsieh, "Dynamicvit: Efficient vision transformers with dynamic token sparsification," in *Advances in Neural Information Processing Systems (NeurIPS)*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 13 937–13 949.

[5] Y. Liu, M. Gehrig, N. Messikommer, M. Cannici, and D. Scaramuzza, "Revisiting token pruning for object detection and instance segmentation," in *Proceedings of the IEEE/CVF Winter Conference*

on Applications of Computer Vision (WACV), January 2024, pp. 2658–2668.

[6] D. Bolya, C.-Y. Fu, X. Dai, P. Zhang, C. Feichtenhofer, and J. Hoffman, "Token merging: Your ViT but faster," in *International Conference on Learning Representations (ICLR)*, 2023.

[7] F. Li, H. Zhang, H. Xu, S. Liu, L. Zhang, L. M. Ni, and H.-Y. Shum, "Mask dino: Towards a unified transformer-based framework for object detection and segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 3041–3050.

[8] X. Zhai, A. Kolesnikov, N. Houlsby, and L. Beyer, "Scaling vision transformers," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, 2022, pp. 12 104–12 113.

[9] J. Jain, J. Li, M. T. Chiu, A. Hassani, N. Orlov, and H. Shi, "Oneformer: One transformer to rule universal image segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 2989–2998.

[10] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, *et al.*, "Segment anything," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023, pp. 4015–4026.

[11] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *International Conference on Learning Representations (ICLR)*. OpenReview.net, 2019.

[12] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," in *International Conference on Learning Representations (ICLR)*, Y. Bengio and Y. LeCun, Eds., 2016.

[13] N. Lee, T. Ajanthan, and P. H. S. Torr, "Snip: single-shot network pruning based on connection sensitivity," in *International Conference on Learning Representations (ICLR)*. OpenReview.net, 2019.

[14] V. Sanh, T. Wolf, and A. Rush, "Movement pruning: Adaptive sparsity by fine-tuning," in *Advances in Neural Information Processing Systems (NeurIPS)*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 20 378–20 389.

[15] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz, "Importance estimation for neural network pruning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[16] N. Yang, Y. Jang, H. Lee, S. Jung, and K. Jung, "Task-specific compression for multi-task language models using attribution-based pruning," in *Findings of the Association for Computational Linguistics (ACL)*, 2022.

[17] G. E. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *ArXiv*, vol. abs/1503.02531, 2015.

[18] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," in *International Conference on Learning Representations (ICLR)*, Y. Bengio and Y. LeCun, Eds., 2015.

[19] J. Yang, B. Martínez, A. Bulat, and G. Tzimiropoulos, "Knowledge distillation via softmax regression representation learning," in *International Conference on Learning Representations (ICLR)*, 2021.

[20] W. Park, D. Kim, Y. Lu, and M. Cho, "Relational knowledge distillation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[21] P. Chen, S. Liu, H. Zhao, and J. Jia, "Distilling knowledge via knowledge review," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 5006–5015.

[22] Z. Yang, Z. Li, M. Shao, D. Shi, Z. Yuan, and C. Yuan, "Masked generative distillation," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Berlin, Heidelberg: Springer-Verlag, 2022, p. 53–69.

[23] Z. Kong, P. Dong, X. Ma, X. Meng, W. Niu, M. Sun, X. Shen, G. Yuan, B. Ren, H. Tang, M. Qin, and Y. Wang, "Spvit: Enabling faster vision transformers via latency-aware soft token pruning," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Berlin, Heidelberg: Springer-Verlag, 2022, p. 620–640.

[24] L. Meng, H. Li, B.-C. Chen, S. Lan, Z. Wu, Y.-G. Jiang, and S.-N. Lim, "Adavit: Adaptive vision transformers for efficient image recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 12 299–12 308.

[25] M. Fayyaz, S. A. Koohpayegani, F. R. Jafari, S. Sengupta, H. R. V. Joze, E. Sommerlade, H. Pirsiavash, and J. Gall, "Adaptive token sampling for efficient vision transformers," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Berlin, Heidelberg: Springer-Verlag, 2022, p. 396–414.

[26] Y. Liang, C. Ge, Z. Tong, Y. Song, J. Wang, and P. Xie, "Evit: Expediting vision transformers via token reorganizations," in *International Conference on Learning Representations (ICLR)*, 2022.

[27] H. Wang, B. Dedhia, and N. K. Jha, "Zero-tprune: Zero-shot token pruning through leveraging of the attention graph in pre-trained transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, pp. 16 070–16 079.

[28] E. Courdier, P. T. Sivaprasad, and F. Fleuret, "Paumer: Patch pausing transformer for semantic segmentation," in *British Machine Vision Conference (BMVC)*, 2022.

[29] Y. Liu, Q. Zhou, J. Wang, Z. Wang, F. Wang, J. Wang, and W. Zhang, "Dynamic token-pass transformers for semantic segmentation," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, January 2024, pp. 1827–1836.

[30] Q. Tang, B. Zhang, J. Liu, F. Liu, and Y. Liu, "Dynamic token pruning in plain vision transformers for semantic segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2023, pp. 777–786.

[31] N. Norouzi, S. Orlova, D. de Geus, and G. Dubbelman, "Algm: Adaptive local-then-global token merging for efficient semantic segmentation with plain vision transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2024, pp. 15 773–15 782.

[32] J. B. Haurum, S. Escalera, G. W. Taylor, and T. B. Moeslund, "Agglomerative token clustering," in *Proceedings of the European Conference on Computer Vision (ECCV)*, A. Leonardis, E. Ricci, S. Roth, O. Russakovsky, T. Sattler, and G. Varol, Eds. Cham: Springer Nature Switzerland, 2025, pp. 200–218.

[33] Y. Yuan, W. Liang, H. Ding, Z. Liang, C. Zhang, and H. Hu, "Expediting large-scale vision transformer for dense prediction without fine-tuning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 1, pp. 250–266, 2024.

[34] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2014, pp. 740–755.

[35] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "Scannet: Richly-annotated 3d reconstructions of indoor scenes," in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2017, pp. 5828–5839.

[36] Y. Xu, Z. Zhang, M. Zhang, K. Sheng, K. Li, W. Dong, L. Zhang, C. Xu, and X. Sun, "Evo-vit: Slow-fast token evolution for dynamic vision transformer," in *Proceedings of the AAAI Conference on Artificial Intelligence*, Jun. 2022, pp. 2964–2972.

[37] Z. Chen, Y. Duan, W. Wang, J. He, T. Lu, J. Dai, and Y. Qiao, "Vision transformer adapter for dense predictions," in *International Conference on Learning Representations (ICLR)*, 2023.

[38] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jegou, "Training data-efficient image transformers distillation through attention," in *Proceedings of the 38th International Conference on Machine Learning (ICML)*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 10 347–10 357.

[39] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision (CVPR)*, 2017, pp. 2961–2969.