Transferable and Principled Efficiency for Open-Vocabulary Segmentation

Supplementary Material

6. Experimental Details

6.1. Fine-tuning Settings After Transfer

This section provides a detailed introduction to the experimental settings for all three transfer experiments. The experimental settings used for Han et al. and Deeplabv3 are identical. We trained the models on the COCO panoptic dataset for 50,000 iterations using four 4090 Ti GPUs. The training was performed with a batch size of 28, utilizing the SGD optimizer with an initial learning rate of 0.00015. The learning rate was reduced by a factor of 0.1 at 40,000 iterations and 45,000 iterations. For FC-CLIP, we conducted training for 368,750 iterations using eight 4090 Ti GPUs. The training was performed with a batch size of 16, utilizing the AdamW optimizer with an initial learning rate of 0.0001. The learning rate was reduced by a factor of 0.1 at 327,778 iterations and 355,092 iterations.

6.2. Training FLOPs Calculation

To calculate the training FLOPs of our model backbone, we consider both the forward and backward processes. When calculating the backward FLOPs, we need to compute gradients for both model parameters and the hidden features. Hence, the FLOPs involved in the backward process will be doubled compared with the FLOPs of the forward process[1]. Suppose the inference FLOPs of the model is C. We provide a comprehensive calculation approach below where our subnetwork has a sparsity of 10% and 50% of the layers are frozen during fine-tuning:

1. **Standard Fine-tuning:** forward FLOPs $1\times C$, backward FLOPs $2\times C$.
2. **Model with layer-wise fine-tuning:** Frozen layers: forward FLOPs $1\times C$, backward FLOPs $1\times C$. Active layers: forward FLOPs $1\times C$, backward FLOPs $2\times C$.
3. **Model with subnetwork:** forward FLOPs $0.1\times C$, backward FLOPs $1.1\times C$ (1 for hidden features, 0.1 for sparse weights).
4. **Model with layer-wise fine-tuning and subnetwork:** Frozen layers: forward FLOPs $0.1\times C$, backward FLOPs $1\times C$. Active layers: forward FLOPs $0.1\times C$, backward FLOPs $1.1\times C$ (1 for hidden features, 0.1 for sparse weights).

To calculate the FLOPs of specific layers, we utilize the "$get\_model\_complexity\_info$" function provided by the $ptFLOPs$ library[2].

[1] https://epochai.org/blog/backward-forward-FLOP-ratio

[2] https://pypi.org/project/ptflops/

6.3. Visualizations of Heavy-tail Behaviors in Efficient Fine-tuning

Our Fig. 4 provides a comprehensive overview of the layer-wise fine-tuning implementation process. Fig. 6 showcases the α value for each layer in our backbone before fine-tuning (after pruning, before adopting sparse masks). Layers of α below the red horizontal line (median value of α across layers) will be frozen during fine-tuning. Additionally, Fig. 7 illustrates the changes in the median of α values throughout the entire fine-tuning process under our training method. It is evident that the α value of most layers remains relatively stable, leading us to not dynamically calculate α or adjust our fine-tuning layers in the main experiment.

7. Futher Experiments

7.1. Ablation Study on Different Knowledge Distillation Losses

In the method presented in Sec. 3.1, a loss function is introduced to align the text with the vision feature space. This loss, referred to as text-guided knowledge distillation (TGKD), facilitates the distillation of knowledge from textual information into visual embeddings. The process of text-guided knowledge distillation can be formulated as follows:

$$\mathcal{L}_{TGKD} = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N}\Big\|(\|\mathcal{V}_i - \mathcal{R}(I, M_j)\| - \|\mathcal{T}(Y_i) - \mathcal{T}(Y_j)\|)\Big\|, \tag{3}$$

where $\mathcal{T}$ denotes the CLIP text encoder, $\mathcal{R}$ denotes the CLIP image encoder, $\mathcal{V}_i$ denotes the visual embeddings and Y_i is the category name of i-th ground truth region. I represents the input image, while M represents the mask corresponding to the respective category. Correspondingly, a vision-guided knowledge distillation approach has also been proposed, which can be formulated as follows:

$$\mathcal{L}_{VGKD} = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N}\Big\|(\|\mathcal{V}_i - \mathcal{R}(I, M_j)\| - \|\mathcal{V}_i - \mathcal{V}_j\|)\Big\|, \tag{4}$$

Tab. 5 provides a comprehensive comparison of the results obtained from pruning using two distinct distillation losses. The analysis demonstrates that the utilization of text-guided distillation loss significantly enhances the OVS performance of the model.

7.2. Ablation Study on Pruning Strategies

We further study two pruning strategies. The first strategy is to naively apply IMP to prune the model using all training

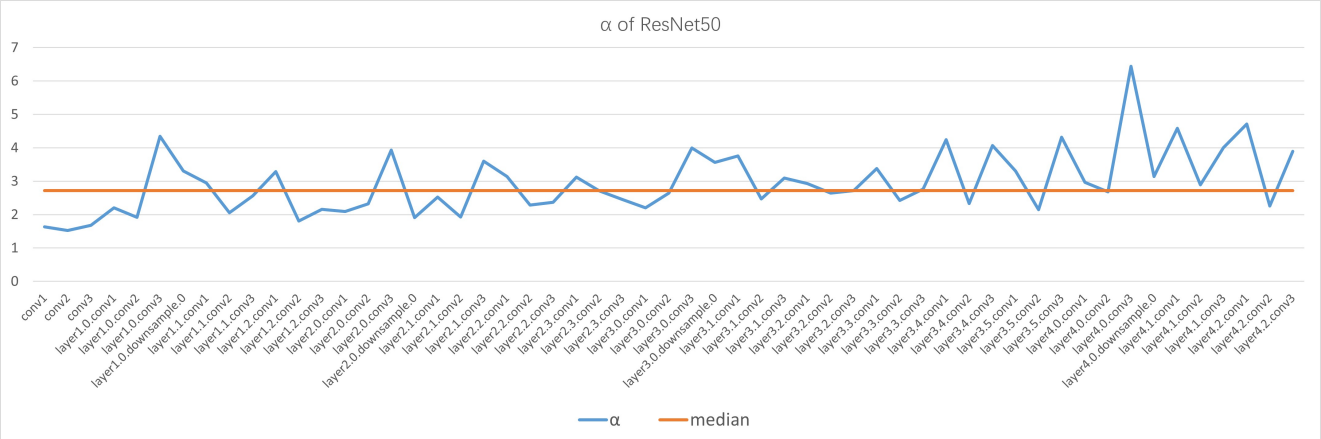

Figure 6. Layer-wise α values in model backbone before fine-tuning used in Sec. 4.3. Specifically, the blue curve represents the calculated α value for each layer of Resnet, while the red line represents their median. During the fine-tuning process, we freeze the layers that have values smaller than the median.

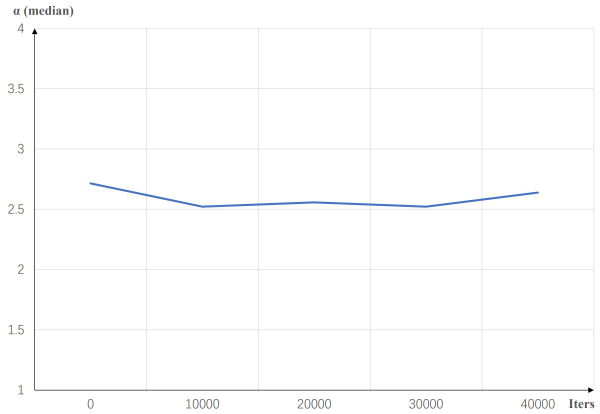

Figure 7. The median value of layer-wise α during the fine-tuning process. We can see α remains relatively stable without significant changes.

Table 5. Comparison of different distillation loss: $\mathcal{L}_{TGKD}$ vs. $\mathcal{L}_{VGKD}$.

Distillation Loss	COCO	PC-59	ADE20k-150
Text-guided ($\mathcal{L}_{TGKD}$)	42.5	35.1	15.8
Vision-guided ($\mathcal{L}_{VGKD}$)	40.0	32.6	14.2

losses (distillation loss plus mask loss and classification loss). The second strategy, which is proposed in our Sec. 3.2, first obtains subnetworks based on the semantic-agnostic distillation loss, and subsequently fine-tunes the model for a specific number of iterations.

Tab. 8 provides experimental details. The results demonstrate that the adopted pruning with fine-tuning method not only improves OVS performance but also requires fewer training iterations. Additionally, the subnetworks obtained

Table 6. Ablation study of different configurations for pruning with both distillation and segmentation loss. Pruning ratio (p) and the number of training iterations (t) used in Algorithm 1 were studied.

p	t	COCO	Cityscapes	ADE20K-150	PAS-20	PC-59
0.1	5000	40.6	29.1	15.0	60.1	33.0
0.1	2500	36.9	28.9	13.8	54.8	32.0
0.2	5000	32.1	27.8	11.6	50.4	27.8

Table 7. Ablation study of different configurations for pruning with only distillation loss followed by segmentation fine-tuning (our strategy in Sec. 3). Pruning ratio (p) and the number of training iterations (t) used in Algorithm 1 were studied.

p	t	COCO	Cityscapes	ADE20K-150	PAS-20	PC-59
0.1	5000	41.8	31.8	15.4	63.2	35.1
0.1	1000	41.9	32.7	15.0	63.9	35.1
0.1	100	42.4	31.0	14.3	63.2	34.8
0.3	1000	40.4	28.9	13.7	61.4	33.6
0.5	1000	39.8	30.3	13.6	60.7	33.3

without semantic supervision exhibit the ability to be further fine-tuned in the original model and can also be effectively transferred to other tasks, yielding superior experimental results (as shown in Tab. 1).

Ablation Study on Pruning Configurations. We also study the best configuration of t and p of two pruning strategies discussed above. Ablation studies on the first pruning strategy (distillation + segmentation loss) are shown in Tab. 6. It is important to highlight that reducing the value of t or increasing the value of p has a substantial impact on the performance, leading to a noticeable drop.

Additionally, for our pruning strategy (pruning with distillation only, followed by segmentation fine-tuning), different

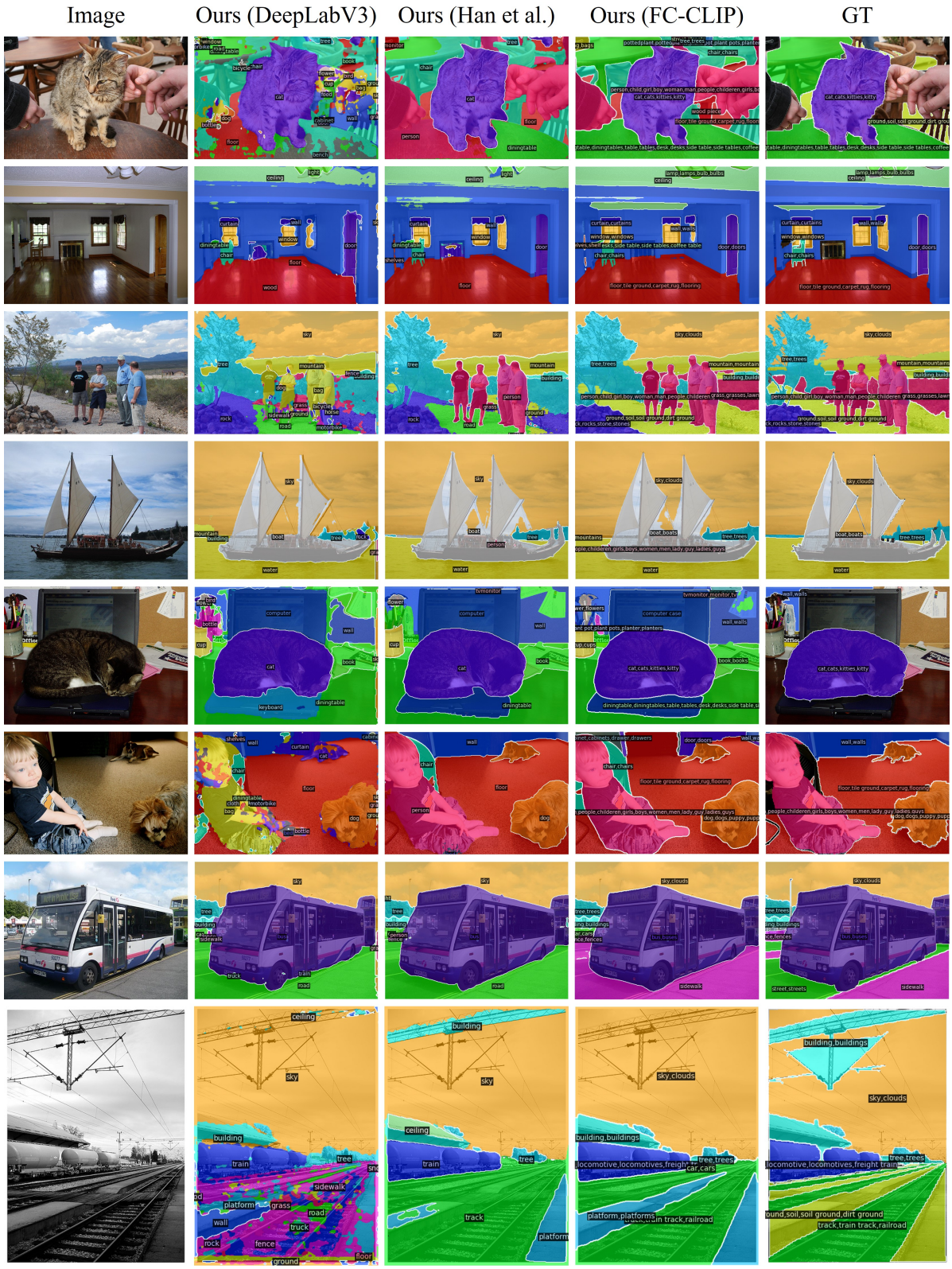

Figure 8. More visualizations of examples on the PC-59. "GT": ground truth.

Table 8. We compare the segmentation performance (mIoU) of two strategies for finding sparse models: 1) pruning with both distillation and segmentation loss; 2) our pruning (with only distillation loss) followed by segmentation fine-tuning (Sec. 3). The "Training Iters" parameter represents the total number of training iterations required by each of the two methods.

Pruning Method	COCO	PC-59	ADE20k-150	Training Iters.
Pruning (distillation + segmentation loss)	40.6	33.0	15.0	105000
Pruning (distillation only) + Segmentation Fine-tuning	42.5	35.1	15.8	95000

Table 9. CKA similarity between CLIP image encoder and DeeplabV3 backbones.

	DeeplabV3	Sparse DeeplabV3 (Ours)
CKA (vs. CLIP)	0.361	0.512

Table 10. Ablation study of freezing ratio

Ratio	COCO	Cityscapes	ADE-150	ADE-847	PAS-20	PC-59	PC-459
0.25	47.5	34.5	17.3	2.9	72.5	39.6	7.6
0.75	46.5	34.8	17.4	2.8	72.6	38.9	7.5
0.5	47.2	34.0	17.3	2.9	74.0	39.9	7.7

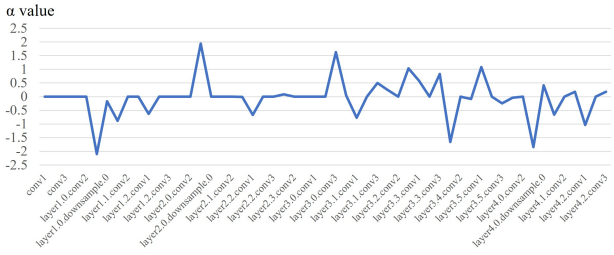

Figure 9. α changes before and after finetuning.

configurations are studied in Tab. 7. Initially, a pruning rate (p) of 0.1 was used, while different values of t were employed. The performance on the COCO dataset was found to be very similar across different t values. However, when t was reduced to a very small value, a significant drop in performance on the OVS datasets was observed. This observation further supports the claim that the subnetwork discovery method proposed in Sec. 3.2 is advantageous for the OVS task. If t is kept constant and the pruning rate is increased, it will result in IMP finding subnetwork with lower quality. It is worth noting that the final sparsity of the subnetworks discovered by different pruning rates will vary, making direct comparisons between them relatively unfair. However, this disparity is inevitable given the different pruning rates employed.

7.3. Ablation Study Implementation details

In this section, we conduct several ablations to justify the design choices of our proposed methods.

Compare with other compression methods In Tab. 1, we mainly compare our method with random pruning, since it can also be transferred to different models without incurring any additional costs. In Tab. 8, we compare our method with the established pruning technique IMP. Our approach outperforms IMP in terms of both performance and training time. Moreover, our subnetwork exhibits transferability, allowing for faster training across different segment architectures.

OVS capability analysis Our core method leverages the benefits of our transferable subnetwork to improve OVS performance and enable enhanced open-set knowledge distillation from CLIP. As shown in Tab. 9, by adopting our subnetwork, DeeplabV3 can achieve more similar features with CLIP than the baseline, measured by Centered Kernel Alignment (CKA) similarity averaged over layers [9].

Analysis of freezing layers As shown in Fig. 9, compared with early layers, α of deeper layers undergo more changes during fine-tuning, gradually becoming more "well-trained" as their αs decrease [34].

Based on the implicit self-regularization in deep networks [34], weight matrices with $\alpha < 2$ are generally considered "over-trained" and more prone to overfitting. Therefore, in our supplementary Figure 6, we observe certain layers with $\alpha < 2$. Freezing these layers during fine-tuning provides benefits, as it helps prevent overfitting. We also provide different ratios of freezing layers in Tab. 10. Users can adjust this ratio flexibly according to their own needs.

7.4. More Qualitative Results

Building upon the findings in Sec. 4.6, we present additional qualitative results in this section, along with a comparison to the ground truth. Fig. 8 illustrates a specific case where our model demonstrates robust OVS performance by accurately labeling some parts that are not labeled in the ground truth. This exemplifies the effectiveness of our model in accurately predicting labels even in challenging scenarios where ground truth annotations may be incomplete.