ViT-Split: Unleashing the Power of Vision Foundation Models via Efficient Splitting Heads

Supplementary Material

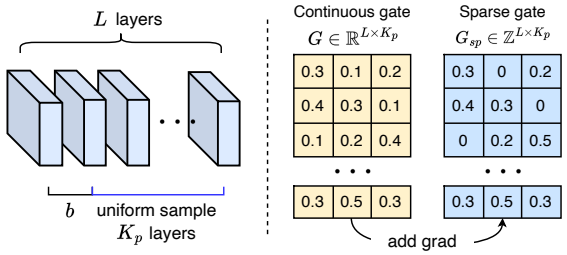


Figure 1. Illustration of our proposed layer selection methods: uniform sampling (left) and sparse gate (right). Uniform sampling selects K_p layers from L prior features, ranging from the b-th to L-th layer. The sparse gate, utilizing the STE technique (see Eq. (1)), aggregates multiple layer features and filters out irrelevant ones.

Method	Head	#Train Param	mIoU	Iters
ViT-Split-L (sparse gate)	Linear	164.1M	85.7	20k
ViT-Split-L (uniform)	Linear	164.1M	**85.8**	20k

Table 1. Comparison of two layer selection methods on semantic segmentation. The results are conducted on Cityscales val with 896*896 resolution image.

Method	Head	#Train Param	mIoU	Iters
ViT-Split-S (sparse gate)	Linear	10.2M	51.5	40k
ViT-Split-S (uniform)	Linear	10.2M	**51.6**	40k
ViT-Split-B (sparse gate)	Linear	40.5M	55.5	40k
ViT-Split-B (uniform)	Linear	40.5M	**55.7**	40k
ViT-Split-L (sparse gate)	Linear	88.6M	58.1	40k
ViT-Split-L (uniform)	Linear	88.6M	**58.2**	40k

Table 2. Comparison of two layer selection methods on semantic segmentation. The results are conducted on ADE20K val with 512*512 resolution image.

A. Training details

A.1. Hyper-parameter setting

We outline the settings for several key hyperparameters of ViT-Split in Tab. 3, including weight initialization, the number of tuning layers (K_t), and the number of selected prior features (K_p), *etc*. We conduct experiments across four tasks: semantic segmentation, monocular depth prediction, detection, and visual question answering (VQA).

The selection guideline of K_t, K_p and b. As shown in Tab. 3, these hyperparameters vary across tasks, with their importance ranked as $K_t > K_p > b$. As shown in Fig. 2, K_t is the most critical hyperparameter and is task-dependent. For dense prediction tasks (*e.g.*, segmentation or monocular depth estimation), tuning smaller layers (around $1/6$ to $1/4$) yields good performance. For detection tasks, since the pretrained task differs significantly from detection (see Fig. 4), tuning more layers is necessary for better results. K_p has a smaller impact on results compared to K_t, and $K_p = 4$ works well in most cases. Typically, we set $b = 2$ to sample prior features from both shallow and deep layers. However, for tasks like VQA, only the last-layer features are needed, as the LLM decoder benefits more from high-level features while low-level features may introduce noise.

A.2. Sizes of various heads

We provide the sizes of the various heads used in ViT-Split for different tasks in Tab. 5, including segmentation (seg.), detection (det.) and monocular depth estimation (mde).

A.3. Details of tuning the VLLM

LLaVA-1.5 employs a CLIP-based vision encoder for image encoding. We introduce a single-layer task head copied from CLIP's original final layer (*i.e.*, $K_t = 1$) and utilize only the last-layer feature of CLIP as the input to the prior head (*i.e.*, $K_p = 1$). We replace the original MLP projector in LLaVA-1.5 with with our ViT-Split for two-stage training. The training follows the same hyperparameter settings as the original LLaVA-1.5.

A.4. Architecture details of various used VFMs

We provide the architecture details of various VFMs used in the main content in Tab. 6.

B. Layer selection

B.1. Sparse gate

Another way is to learn the sparse gate $G_{sp} \in \mathbb{R}^{L \times K_p}$ from the dataset. This method eliminates the need for carefully tuning hyperparameters to select prior features. To remove noisy features, we enforce the sparsity in the gate by selecting top K_p scores, and normalizing the remained ones. However, directly optimizing G_{sp} is infeasible since the sparsity operation is non-differentiable. To address this issue, we employ the Straight-Through Estimator (STE) technique which allows for approximate gradient optimization. Specifically, let $G \in \mathbb{R}^{L \times K_p}$ be the learnable gate, which is continuous. From G, we obtain the sparse gates G_{sp} by selecting the top K_p elements in each column. We then apply

Method	Initialization	Tasks	Datasets	Image res.	K_t	K_p	b
ViT-Split-S	DINOv2	Semantic segmentation	ADE20K	(512, 512)	3	4	2
ViT-Split-B	DINOv2	Semantic segmentation	ADE20K	(512, 512)	3	4	2
ViT-Split-L	DINOv2	Semantic segmentation	ADE20K	(512, 512)	4	8	2
ViT-Split-L	DINOv2	Semantic segmentation	ADE20K	(896, 896)	4	8	2
ViT-Split-G	DINOv2	Semantic segmentation	ADE20K	(896, 896)	8	14	26
ViT-Split-L	DINOv2	Semantic segmentation	Cityscapes	(896, 896)	10	8	2
ViT-Split-L	DINOv2	Semantic segmentation	Pascal Context	(480, 480)	6	10	2
ViT-Split-S	DINOv2	Monocular depth estimation	NYU-V2	(416, 544)	3	4	2
ViT-Split-B	DINOv2	Monocular depth estimation	NYU-V2	(416, 544)	4	4	2
ViT-Split-L	DINOv2	Monocular depth estimation	NYU-V2	(416, 544)	3	4	6
ViT-Split-S	DINOv2	Detection, instance segmentation	COCO17	(1024, 1024)	11	4	2
ViT-Split-B	DINOv2	Detection, instance segmentation	COCO17	(1024, 1024)	11	6	2
ViT-Split-L	DINOv2	Detection, instance segmentation	COCO17	(1024, 1024)	23	8	6
LLaVA+ViT-Split-L	CLIP	Vision question answering	VQA benchmarks	(512, 512)	1	1	23

Table 3. The settings of the important hyper-parameters of ViT-Split on different tasks, including semantic segmentation, monocular depth estimation, detection and instance segmentation, and vision question answering (VQA).

Architecture	Head	#Train Param (↓)	AbsRel (↓)	RMSE (↓)	$\log_{10}$ (↓)	δ_1 (↑)	δ_2 (↑)	δ_3 (↑)
ResNet-101 [10]	DORN [9]	110M	0.115	0.509	0.051	0.828	0.965	0.992
ViT [7]	DPT [23]	–	0.110	0.357	0.045	0.904	0.988	0.998
EfficientNet-B5 [26]	AdaBins [2]	77M	0.103	0.364	0.044	0.903	0.984	0.997
ResNet-101 [10]	P3Depth [20]	–	0.104	0.356	0.043	0.898	0.981	0.996
Swin-L [15]	BinsFormer [13]	273M	0.094	0.329	0.04	0.923	0.989	0.997
Swin-L [15]	NeWCRFs [28]	270M	0.095	0.334	0.041	0.922	0.992	–
Swin-L [16]	PixelFormer [1]	–	0.090	0.322	0.039	0.929	0.991	0.998
Swin-L [15]	VA-DepthNet [14]	–	0.086	0.304	–	0.937	0.992	0.998
Swin-L [15]	iDisc [21]	209M	0.086	0.314	0.038	0.940	0.993	0.999
Swin-L [16]	IEBins [24]	273M	0.087	0.314	0.038	0.936	0.992	0.998
SwinV2-L [16]	AiT-P [18]	–	0.076	0.275	0.033	0.954	0.994	0.999
DINOV2-G‡ [19]	DPT [23]	–	0.0907	0.279	0.0371	0.9497	0.996	0.9994
ViT-Split-S‡	Linear	9.3M	0.0897	0.3358	0.039	0.9327	0.9908	0.9985
ViT-Split-B‡	Linear	37.0M	0.0853	0.3019	0.0365	0.9412	0.9947	0.9991
ViT-Split-L‡	Linear	65.5M	**0.078**	**0.2672**	**0.0327**	**0.9622**	**0.9967**	**0.9994**

Table 4. **Monocular depth estimation results on NYU-V2 with 416*544 resolution image.** "‡" represents the use of DINOv2. Other backbones are initialized with ImageNet-1K/22K weights

Architectures	Task	Small	Base	Large
Linear	segmentation	4.78M	19.00M	37.91M
Mask-RCNN	detection	17.54M	17.54M	17.54M
Linear	MDE	0.34M	1.28M	2.23M

Table 5. The size of different heads used for ViT-Split.

Architectures	Embed dim	Layers	Params (M)
DINOv2-S	384	12	22
DINOv2-B	768	12	87
DINOv2-L	1024	24	304
DINOv2-G	1536	40	1137
CLIP-L	1024	24	428

Table 6. The architecture details of used VFMs.

STE by optimizing the gradient of G:

$$G_{sp} = G_{sp} + G - G_{no\_grad}. \tag{1}$$

After obtaining the sparse gate $G_{sp} \in \mathbb{R}^{L \times K_p}$, we can get the selected prior features by multiplying with the prior feature map $\mathbf{f}_p \in \mathbb{R}^{h \times w \times L \times D}$ from the layer dimension.

B.2. Performance on segmentation task

We present a comparison of layer selection methods on segmentation benchmarks, including Cityscapes and ADE20K, in Tab. 1 and Tab. 2. For a fair comparison, we set the same K_t for both selection methods and use $K_p = 4$ for all sparse-gate-based experiments. Our results show that sparse gate selection achieves comparable performance to

uniform sampling on segmentation tasks without requiring manual hyper-parameter selection. It indicates that sparse gate selection is a promising and versatile approach for reducing the number of hyper-parameters.

C. Motivation of freezing the backbone

Freezing the backbone has three main motivations. ① **Improved training and inference speed.** Fig. 7 shows our ViT-Split achieves 2.4∼5×, and 2∼6× speedup over other VFM-Adapters on training and inference efficiency. Additionally, as detailed in Tab. 8, ViT-Split is 1.4∼3× faster than finetuning the entire backbone with a linear/UperNet head. ② **Enhanced performance with prior features.** We admit that the inference speed will decrease compared with finetuning DINOv2-linear due to the extra heads (around 30% on segmentation tasks). However, the performance can be further improved, which is also the main motivation of other VFM-adapters. Compared with these, ViT-Split achieves better training and inference efficiency. ③ **Task adaptivity.** ViT-Split requires storing only separate task-specific heads, rather than the entire model, making it more adaptive and memory-efficient for deployment across multiple tasks.

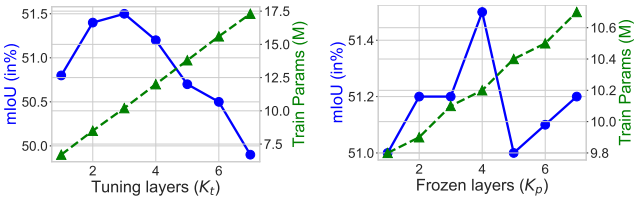
D. Explanation of the lower performance on detection task

We acknowledge that the performance difference between ViT-Split and ViT-CoMer on Mask R-CNN (Tab. 5) is relatively small. However, ViT-Split uses only 90%–95% of ViT-CoMer's trainable parameters, already demonstrating clear advantages in training efficiency while maintaining comparable accuracy. The primary reason ViT-Split does not significantly outperform other VFM-adapters lies in the relatively weak task alignment of the prior features from DINOv2 for object detection tasks. Unlike DETR-style models, which are pre-trained with strong detection-oriented objectives, self-supervised models like DINOv2 tend to provide less directly transferable features for detection. This necessitates using more layers in the task head (*i.e.*, larger K_t), effectively making ViT-Split rely more on fine-tuning, similar to other VFM-adapters. As self-supervised models begin to offer stronger detection-aware priors, we expect ViT-Split to better leverage them and close the gap with current SOTA DETR-style models.

E. More results

E.1. An apple-to-apple comparison with other VFM-adapters on segmentation

We provide an apple-to-apple comparison with the SOTA VFM-adapters in Tab. 7, *i.e.*, ViT-CoMer [27] and ViT-Adapter [6]. All models are trained for 40K iterations on ADE20K, using a UperNet head for the baselines and a linear head for ViT-Split. For VFM-adapters, we adopt a learning rate schedule similar to that used in detection tasks, incorporating layer-wise decay with carefully tuned rates for each baseline to ensure strong performance. Results show that with DINOv2 initialization, ViT-Split consistently outperforms other VFM-adapters across different model sizes. This highlights ViT-Split's ability to better leverage the strong prior knowledge from DINOv2 without altering the original feature representations, which often results in suboptimal performance in other adapters.



(a) Tuning layers K_t. (b) Frozen layers K_p

Figure 2. Parameter sensitivity analysis of K_t and K_p in ViT-Split. The experiments are conducted using ViT-Split-S on ADE20K.

Table 7. VFM-adapter comparison on ADE20K (40K iterations).

Method	DINOv2-S		DINOv2-B		DINOv2-L	
	mAP	#Param	mAP	#Param	mAP	#Param
ViT-Adapter	45.4	57.1M	49.9	134.5M	52.2	364.6M
ViT-CoMer	44.6	61.8M	48.4	145.6M	51.8	384.2M
ViT-Split	51.6	10.2M	55.7	40.5M	58.2	88.6M

E.2. Hyper-parameter sensitivity analysis

We provide the analysis of two important hyper-parameters K_t and K_p in our ViT-Split, which is given in Fig. 2.

Influence of K_t. As shown in Fig. 2a, the mIoU initially improves when tuning between one and three layers. This improvement is likely due to the task head previously underfitting the task. However, as more layers are tuned, overall performance begins to decline, suggesting that the task head starts to overfit. This experiment demonstrates that tuning additional layers does not necessarily guarantee better performance and can easily lead to overfitting. Therefore, we opt to tune three layers in this case.

Influence of K_p. As shown in Fig. 2b, the mIoU peaks when selecting four prior layer features. Selecting too few layers may result in missing critical information, while selecting too many can introduce noise. Additionally, we observe that increasing the number of selected layers does not increase more training parameters, highlighting the efficiency of the prior head. As a result, we choose four prior features in this case.

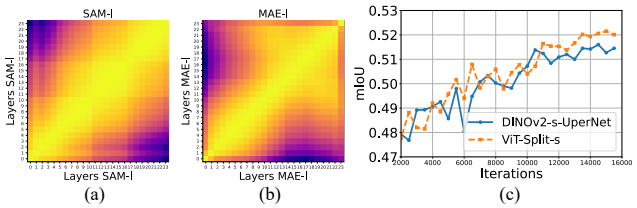


Figure 3. The CKA of SAM (a) and MAE (b). (c) Training comparison between ViT-Split-s and DINOv2-s-UperNet on ADE20K.

E.3. Visualization

E.3.1. CKA analysis of other VFMs

We also present the CKA results for MAE-L [11] and SAM-L [12] in Fig. 3. The feature representations in the early layers of these VFMs exhibit similar patterns, as do those in the later layers. Based on these findings as well as those in the main paper, we hypothesize that our observation–that the layers of several VFMs can be divided into two components–may hold true for self-supervised models pretrained on large-scale dataset (e.g., DINOv2 [19], MAE [11], EVA2 [8], *etc.*), as well as weakly supervised ones (say CLIP [22], SigLip [29], SAM [12], *etc.*).

E.3.2. CKA analysis of different DINOv2 sizes

We also provide the CKA visualizations of different DINOv2 sizes in Fig. 6. From these visualizations, we observe that features in the early layers are more similar across different DINOv2 sizes compared to those in the later layers. As earlier mentioned, the early layers serve as an encoder to capture low-level features, while the later layers act as a decoder to produce task-specific features.

E.3.3. More layer feature comparison

We present additional visualizations of DINOv2 layer features across different tasks (*i.e.*, DINOv2 pretraining, segmentation, and detection) in Fig. 4. These results demonstrate that earlier-layer features from various tasks consistently focus on detailed, low-level information. However, deeper-layer features diverge significantly between tasks [3–5]. Specifically, features from both the original DINOv2 pretraining and semantic segmentation emphasize semantic-level information of particular objects, whereas detection features tend to highlight object corners and boundaries.

E.3.4. Semantic segmentation and instance segmentation results

We present semantic segmentation and instance segmentation results based on our ViT-Split-L (DINOv2 pretrained) in Fig. 5. We utilize ADE20K and COCO2017 datasets for training these two tasks, respectively, and evaluate both on the ADE20K validation dataset.

It is worth noting that both results are obtained using the same frozen DINOv2-L backbone, meaning only the task-specific adapters and heads require training. Consequently, the overall computational cost and the number of parameters are significantly reduced compared to previous VFM-adapters, while achieving competitive or superior performance. These visualizations demonstrate the strong generalization capability of ViT-Split, highlighting its versatility, effectiveness, and efficiency across multiple downstream tasks.

E.4. Training efficiency comparison

Type	ViT-Split-linear	DINOv2-linear	DINOv2-UperNet
Small	9m25s	15m28s	31m21s
Base	17m41s	25m16s	40m23s
Large	32m49s	44m22s	1h19m25s

Table 8. Training time comparison on ADE20K (tuning 10k iterations on 4*A6000Ada). DINOv2-linear and DINOv2-UperNet are finetuned end to end.

To further illustrate the training efficiency compared with different heads on segmentation task, we provide the training time comparison in Tab. 8. For fair comparison, all of these baselines (except for ViT-Split) are finetuned using the DINOv2 backbone with two different heads (linear and UperNet) for 10k iterations on 4*A6000Ada.

From Tab. 8, we observe that our ViT-Split reduces the training time on average of DINOv2-linear by approximately 42% on average while maintaining the same linear head. This improvement in training efficiency is attributed to the task-head design, which *prevents gradients from propagating to the early layers of the backbone*. Compared to finetuning a VFM with a larger segmentation head (DINOv2-UperNet), our ViT-Split is 2.5 times faster across three sizes on average. This highlights the huge computation overhead introduced by a large segmentation head and demonstrates the efficiency of our ViT-Split.

E.5. Longer training time

We try to increase the training time to illustrate the upper bound of ViT-Split. We conduct an experiment in Fig. 3 to explore the performance upper bound with extended training (*i.e.*, 160K iterations). As shown in Fig. 3 (c), ViT-Split-s achieves 52.2%, improving from 51.5% at 40K iterations and surpassing DINOv2s-UperNet (51.6%) while maintaining faster training speeds. This demonstrates that ViT-Split can achieve better performance when training for longer time.

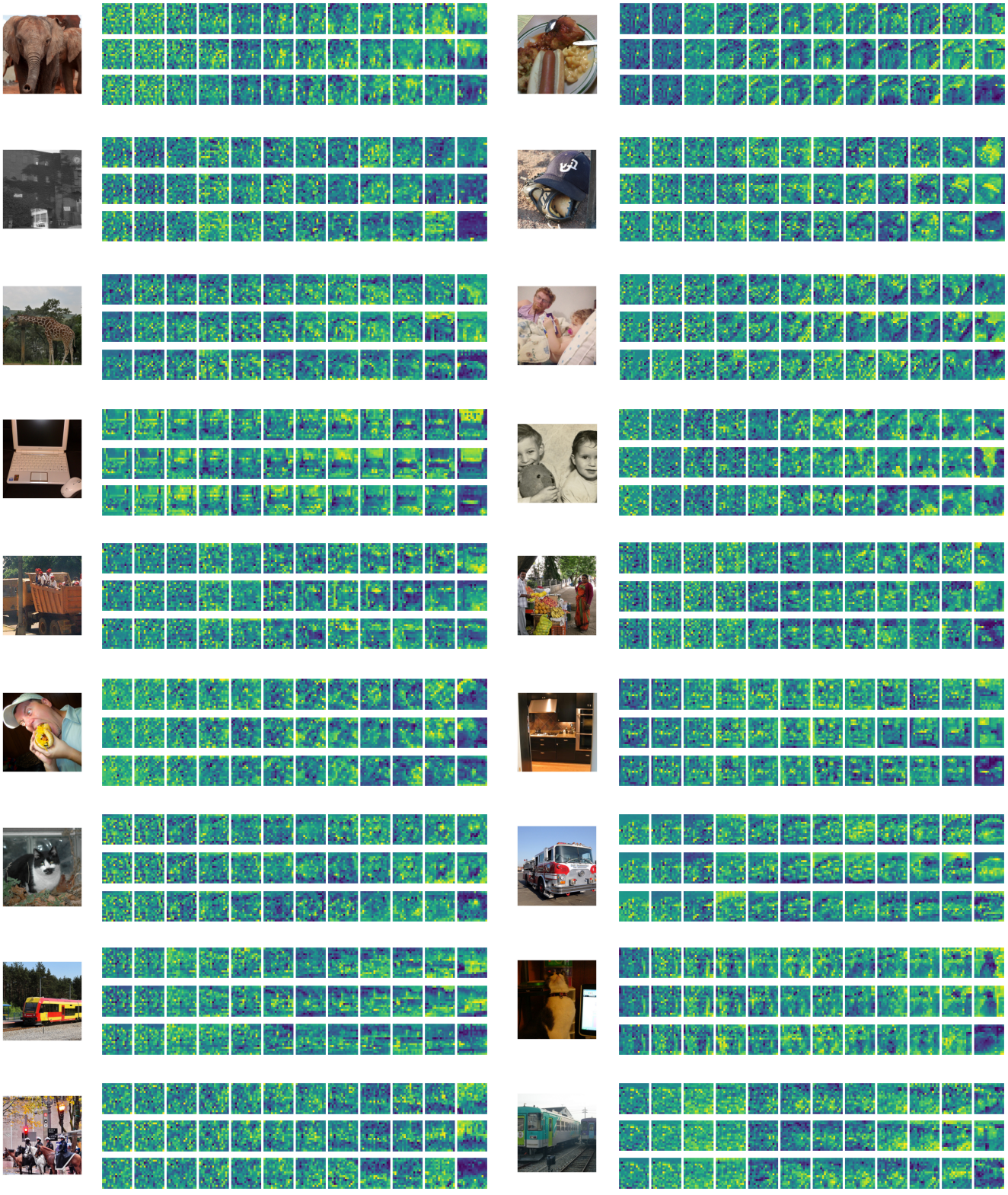
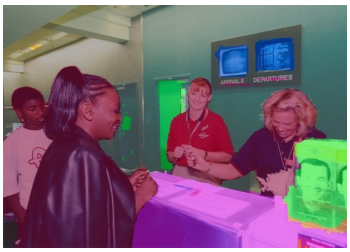
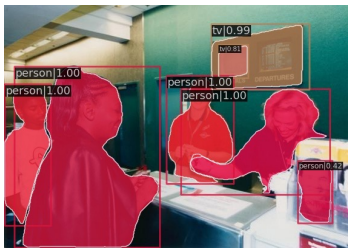
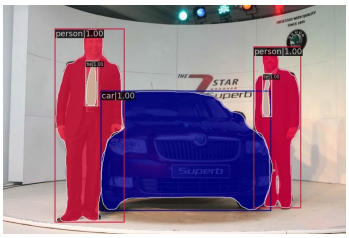


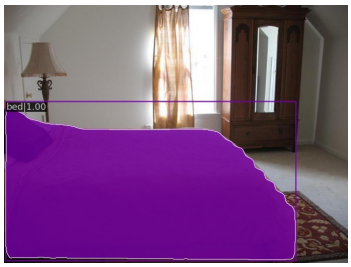
Figure 4. Further comparison of DINOv2-S layer features across original features, segmentation, and detection tasks. In each figure, the **first, second, and third rows correspond to original, segmentation, and detection features**, respectively. It can be observed that features from earlier layers exhibit similar patterns across different tasks, reflecting common low-level local features. However, features from deeper layers diverge significantly according to their specific downstream tasks.

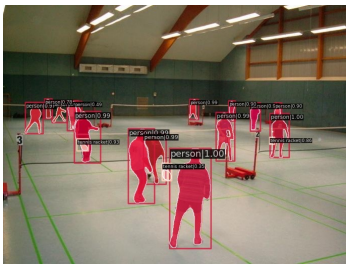



tv|0.99
person|1.00
person|1.00
person|1.00
person|1.00
person|0.42



person|1.00
book|0.73
dining table|0.55
book|0.70
book|0.75



person|1.00
person|1.00
car|1.00



bed|1.00



person|1.00



car|1.00

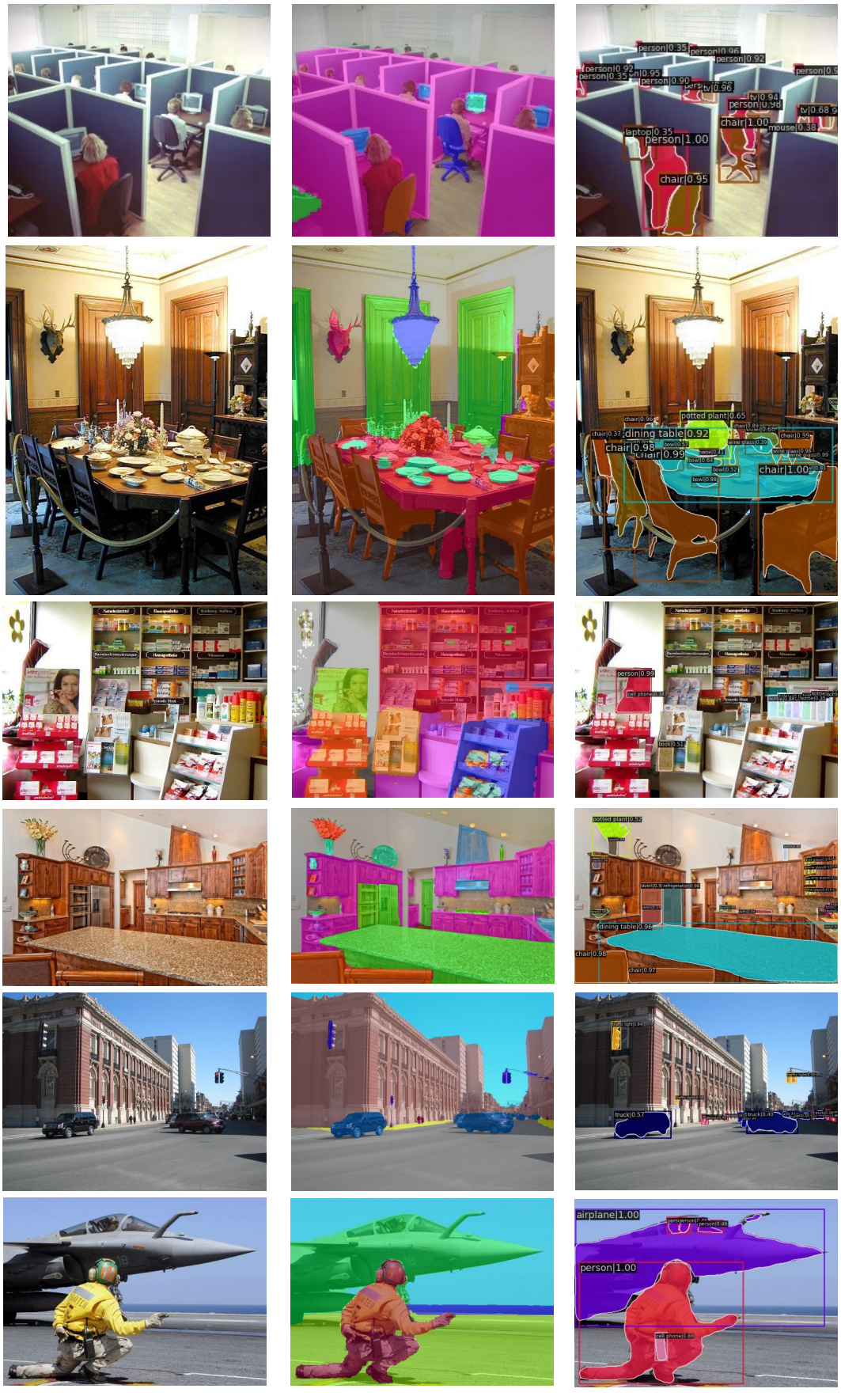

person 0.35
person 0.96
person 0.92
person 0.92
person 0.35
person 0.95
person 0.90
tv 0.96
tv 0.94
person 0.98
tv 0.68
chair 1.00
mouse 0.38
laptop 0.35
person 1.00
chair 0.95
potted plant 0.65
dining table 0.92
chair 0.98
chair 0.99
chair 1.00
person 0.99
truck 0.57
truck 0.40
airplane 1.00
person 1.00
cell phone 0.88

Figure 5. Semantic segmentation and instance segmentation results based on our ViT-Split-L (**left: original image, middle: semantic segmentation results, right: instance segmentation results**).

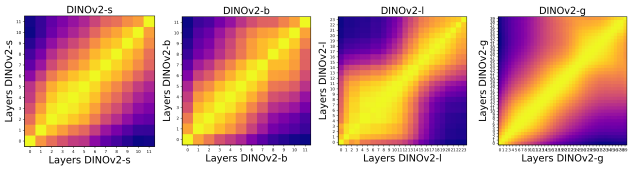


Figure 6. The CKA visualizations of different sizes of DINOv2.

Method	Head	#Train Param	mIoU (SS/MS)	Schedule
ViT-Adapter-B†	Mask2former	120M	64.0/64.4	40k
ViT-Adapter-L†	UperNet	451M	67.0/67.5	80k
ViT-Adapter-L†	Mask2former	568M	67.8/68.2	80k
ViT-Split-B‡	Linear	47.5M	66.4/66.8	20k
ViT-Split-L‡	Linear	115.8M	**68.1/68.6**	20k

Table 9. **Semantic segmentation results on the Pascal Context val with 480*480 resolution image.** "†" indicates the BEiT initialization and "‡" represents the use of DINOv2.

E.6. Monocular depth estimation

Settings. To further investigate the effectiveness of our ViT-Split, we also provide the results on monocular depth estimation (MDE) on NYU-V2 [25] benchmark in Tab. 4. Following [13], we utilize the AdamW optimizer with an initial learning rate of 3e-4 and a weight decay of 1e-2. We multiply 0.1 by the learning rate of the task head during training. Moreover, one cycle learning rate decay schedule is utilized for better performance. We train ViT-Split for 384K iterations with a total batch size of 16 on 4*A6000ada GPUs.

As shown in Tab. 4, our ViT-Split achieves competitive or even superior results compared to previous state-of-the-art methods, while using a minimal number of trainable parameters. Notably, ViT-Split employs only a single linear head rather than a specially designed head, highlighting the potential of our approach. Leveraging the prior knowledge embedded in vision foundation models (VFMs), we believe the size of the downstream task head (e.g., for depth prediction) can be further reduced to improve efficiency.

When compared to DINOv2-G with DPT [23], which uses the same DINOv2 initialization but a larger and more sophisticated head, our smaller ViT-Split-B version achieves similar performance with fewer parameters, demonstrating both the effectiveness and efficiency of our method. Furthermore, compared to traditional end-to-end fine-tuning approaches, ViT-Split achieves better performance by fully utilizing the prior knowledge inherent in VFMs. This also highlights the significant potential of large-scale self-supervised learning initialization over traditional supervised learning initialization.

E.7. Segmentation on Pascal Context

Settings. Apart from ADE20K and Cityscapes, we also provide the results on Pascal Context [17] in Tab. 9. We utilize the AdamW optimizer with an initial learning rate of 1e-4 and weight decay of 1e-2. We multiply by 0.1 to the task head during training. We train our model for 20K iterations, and the total batch size is set to 16.

As shown in Tab. 9, our method outperforms ViT-Adapter, achieving a 2% improvement for the base model and a 0.3% improvement for the large model, using just a simple linear head and training for only 20K iterations. The results demonstrate the strength of VFMs, with our method achieving both effectiveness and efficiency by fully utilizing the prior knowledge within the VFMs.

F. Limitations

Currently, we have demonstrated the effectiveness of ViT-Split only on a limited set of VFMs, such as DINOv2 and CLIP, leaving its performance on a broader range of models to be explored in future work.

References

[1] Ashutosh Agarwal and Chetan Arora. Attention attention everywhere: Monocular depth prediction with skip attention. In *WACV*, pages 5861–5870, 2023. 2

[2] Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. Adabins: Depth estimation using adaptive bins. In *CVPR*, pages 4009–4018, 2021. 2

[3] Ruoyu Chen, Hua Zhang, Siyuan Liang, Jingzhi Li, and Xiaochun Cao. Less is more: Fewer interpretable region via submodular subset selection. In *ICLR*, 2024. 4

[4] Ruoyu Chen, Siyuan Liang, Jingzhi Li, Shiming Liu, Maosen Li, Zhen Huang, Hua Zhang, and Xiaochun Cao. Interpreting object-level foundation models via visual precision search. In *CVPR*, pages 30042–30052, 2025.

[5] Ruoyu Chen, Siyuan Liang, Jingzhi Li, Shiming Liu, Li Liu, Hua Zhang, and Xiaochun Cao. Less is more: Efficient black-box attribution via minimal interpretable subset selection. *arXiv preprint arXiv:2504.00470*, 2025. 4

[6] Zhe Chen, Yuchen Duan, Wenhai Wang, Junjun He, Tong Lu, Jifeng Dai, and Yu Qiao. Vision transformer adapter for dense predictions. In *ICLR*, 2023. 3

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2020. 2

[8] Yuxin Fang, Wen Wang, Binhui Xie, Quan Sun, Ledell Wu, Xinggang Wang, Tiejun Huang, Xinlong Wang, and Yue Cao. Eva: Exploring the limits of masked visual representation learning at scale. In *CVPR*, pages 19358–19369, 2023. 4

[9] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. In *CVPR*, pages 2002–2011, 2018. 2

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 2

[11] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, pages 16000–16009, 2022. 4

[12] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *ICCV*, pages 4015–4026, 2023. 4

[13] Zhenyu Li, Xuyang Wang, Xianming Liu, and Junjun Jiang. Binsformer: Revisiting adaptive bins for monocular depth estimation. *IEEE TIP*, 2024. 2, 9

[14] Ce Liu, Suryansh Kumar, Shuhang Gu, Radu Timofte, and Luc Van Gool. Va-depthnet: A variational approach to single image depth prediction. In *ICLR*, 2023. 2

[15] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, pages 10012–10022, 2021. 2

[16] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, et al. Swin transformer v2: Scaling up capacity and resolution. In *CVPR*, pages 12009–12019, 2022. 2

[17] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The role of context for object detection and semantic segmentation in the wild. In *CVPR*, pages 891–898, 2014. 9

[18] Jia Ning, Chen Li, Zheng Zhang, Chunyu Wang, Zigang Geng, Qi Dai, Kun He, and Han Hu. All in tokens: Unifying output space of visual tasks via soft token. In *ICCV*, pages 19900–19910, 2023. 2

[19] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2023. 2, 4

[20] Vaishakh Patil, Christos Sakaridis, Alexander Liniger, and Luc Van Gool. P3depth: Monocular depth estimation with a piecewise planarity prior. In *CVPR*, pages 1610–1621, 2022. 2

[21] Luigi Piccinelli, Christos Sakaridis, and Fisher Yu. idisc: Internal discretization for monocular depth estimation. In *CVPR*, pages 21477–21487, 2023. 2

[22] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, pages 8748–8763, 2021. 4

[23] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *ICCV*, pages 12179–12188, 2021. 2, 9

[24] Shuwei Shao, Zhongcai Pei, Xingming Wu, Zhong Liu, Weihai Chen, and Zhengguo Li. Iebins: Iterative elastic bins for monocular depth estimation. In *NeurIPS*, 2024. 2

[25] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, pages 746–760, 2012. 9

[26] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, pages 6105–6114, 2019. 2

[27] Chunlong Xia, Xinliang Wang, Feng Lv, Xin Hao, and Yifeng Shi. Vit-comer: Vision transformer with convolutional multi-scale feature interaction for dense predictions. In *CVPR*, pages 5493–5502, 2024. 3

[28] Weihao Yuan, Xiaodong Gu, Zuozhuo Dai, Siyu Zhu, and Ping Tan. Neural window fully-connected crfs for monocular depth estimation. In *CVPR*, pages 3916–3925, 2022. 2

[29] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *ICCV*, pages 11975–11986, 2023. 4