Supplementary Material: A Simple Single-Scale Vision Transformer for Object Detection and Instance Segmentation

Wuyang Chen[1⋆], Xianzhi Du[2], Fan Yang[2]
Lucas Beyer[2], Xiaohua Zhai[2], Tsung-Yi Lin[2], Huizhong Chen[2], Jing Li[2],
Xiaodan Song[2], Zhangyang Wang[1], and Denny Zhou[2]

[1] University of Texas at Austin, Austin TX 78712, USA
{wuyang.chen,atlaswang}@utexas.edu
[2] Google
{xianzhi,fyangf,lbeyer,xzhai,tsungyi,
huizhongc,jingli,xiaodansong,dennyzhou}@google.com

1 Pascal VOC semantic segmentation

Settings Semantic segmentation experiments are conducted on Pascal VOC 2012, which contains 20 foreground classes and 1 background. For training, we use an augmented version of the dataset [9] with extra annotations of 10582 images (trainaug). The default training setup uses scale jittering of [0.5, 2.0] and random horizontal image flipping.

1.1 Scaling rule of UViTs

To also achieve the best performance-efficiency trade-off on semantic segmentation task, we further systematically study the model scaling of UViTs on depths and widths on the Pascal VOC dataset. We show our results[3] in Figure 1. For all models (circle markers), we first train them on ImageNet-1k, then directly fine-tune them on Pascal VOC. We fix the input size as 512×512.

- Depth (number of attention blocks): we study different UViT models of depths selected from $\{12, 18, 24, 32\}$.
- Width (i.e. hidden size, or output dimension of attention blocks): we will tune the width to further control different model sizes and computation costs to make different scaling rules fairly comparable.

In summary, based on our compound scaling rule, we find the UViT of depth 32 performs the best on Pascal VOC.

⋆ Work done during the first author's research internship with Google.

[3] This scaling rule is studied before we study the attention window strategy in Section 1.3. Thus for all models in Figure 1 we adopt the window scale as $\frac{1}{2}$, for fair comparisons.

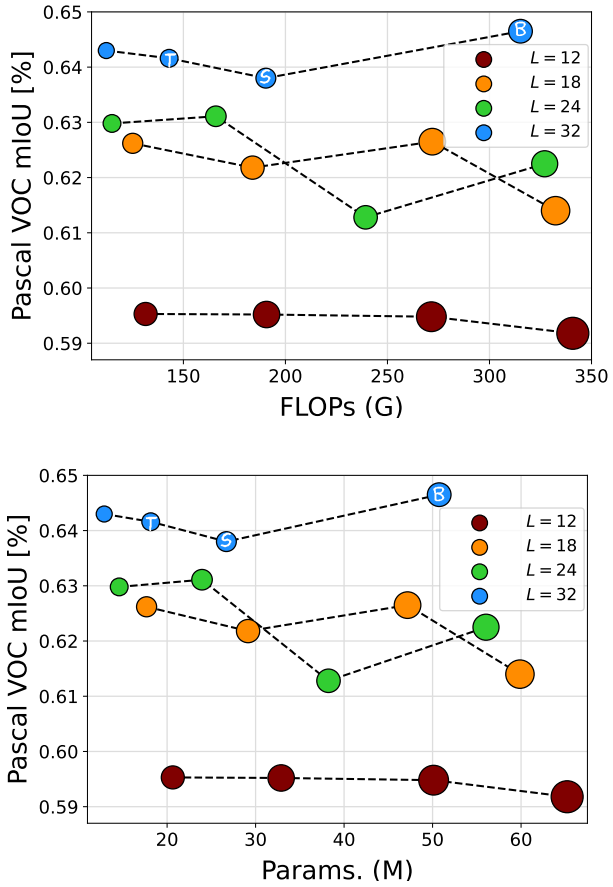

Fig. 1: **Model** scaling rule for UViT on Pascal VOC semantic segmentation (ImageNet pretrained, before COCO pretraining). **32 attention blocks** (blue) perform better than shallower UViTs. Different sizes of markers represent the hidden sizes (widths).

1.2 Architectures

We propose three variants of our UViT variants. The architecture configurations of our model variants are listed in Table 1, and are also annotated in Figure 1 ("T", "S", "B" in white). The number of heads is fixed as six, and the expansion ratio of each FFN (feed-forward network) layer is fixed as four in all experiments. We also scale up our UViT into a huge version following the design in [7,15], and denote it as "UViT-H".

1.3 Attention windows on Pascal VOC

In this section, we further study the attention window strategy on the Pascal VOC dataset, using our UViT-B. As shown in Table 2, progressive attention

Table 1: Architecture variants of our UViT for Pascal VOC semantic segmentation.

Name	Depth	Hidden Size	Params. (M)
UViT-T	32	192	18.1
UViT-S	32	240	26.7
UViT-B	32	342	50.7
UViT-H	32	1280	529.9

windows again achieve the best performance. Global attentions in deep layers are vital, and a smaller window in early attentions can improve efficiency. In conclusion, we set the window scale of our UViT as "$[2^{-1}] \times 28 \rightarrow [1^{-1}] \times 4$" for its reduced computation cost.

Table 2: Local attention windows in early layers can improve model efficiency, and global attention windows in deep layers are vital to the final performance on Pascal VOC. Model: UViT-B.

[window_scale] × #layers	GFLOPs	AP_{val}
$[1] \times 32$	596.7	81.1
$[2^{-1}] \times 32$	315.2	80.6
$[2^{-1}] \times 28 \rightarrow [1^{-1}] \times 4$	350.4	81.2

1.4 Final performance on Pascal VOC

Following the same procedure in [3,13,8,4,5,1,12,16,10,14], we employ the COCO dataset [11] to pretrain our model. From Table 3 we can see that our UViT is highly compact in terms of the number of parameters, and achieve competitive mIoU with comparable FLOPs. In addition, it is worth noting that since we only leverage a single-scale feature map, we do not adopt advanced segmentation decoders like ASPP [4] but just use plain convolutional layers for predictions, which is to our disadvantage. Again, without adopting any design conventions from CNNs, the results of our simple UViT architectures suggest that, a simple single-scale transformer backbone can fulfill the dense prediction tasks.

Table 3: Segmentation results on Pascal VOC 2012. Our UViT leverages a plain convolutional segmentation head, without any test-time augmentation.

Backbone	Resolution	GFLOPs	Params. (M)	mIoU
WASPnet-CRF [2]	-	-	47.5	80.4
DeepLabv3+ (ResNet-101) [6]	512×512	298	58.6	79.4
UViT-T (ours)	512×512	163	18.1	79.0
UViT-S (ours)	512×512	215	26.7	79.9
UViT-B (ours)	512×512	350	50.7	81.2
UViT-H (ours)	640×640	3846	529.9	88.1

2 Model architectures studied in Figure 2

We show details of all architectures studied in Figure 2 (main body) in Table 4 below. As mentioned in Section 3.1 (main body), we study all combinations of the above three techniques (spatial downsampling "SD", multi-scale features "MF", doubled channels "2×"), i.e. eight settings in total, and show the results in Figure 2 (main body). Each dot in Figure 2 (main body) indicates an individually designed and trained model. To make all comparisons fair, we carefully design all models such that they are all of around 72 million parameters. We control their FLOPs by changing the depths or attention windows allocated to different stages.

3 Model architectures in Figure 4 and Figure 5

We show all architectures studied in our compound scaling rule in Figure 4 and Figure 5 (main body). All models are of 2^{-1}-scale attention windows for fair comparisons.

Table 4: Model architectures in Figure 2 (main body), all studied under a 640×640 input size on MS-COCO. "SD": spatial downsampling. "MF": multi-scale features. "2×": doubled channels. Without any of these three techniques (first section in this table), the whole network has a constant feature resolution and hidden size; all other seven settings below will split the network into three stages, since they require either a progressive feature downsampling or multi-scale features from each stage. Input scale is relative to the 2D shape of the input image $H \times W$ (e.g. 8^{-1} indicates the 2D shape of the UViT's sequence feature is $\frac{1}{8}H \times \frac{1}{8}W$). The window scale is relative to the 2D shape of sequence feature's $h \times w$ (e.g. 8^{-1} indicates the 2D shape of the attention window is $\frac{1}{8}h \times \frac{1}{8}w$). Numbers with underscores in the column "Output Scale" indicate feature maps that will be fed into the FPN detection head (i.e., the last output of backbone if no "MF" is applied, or features from all three stages if "MF" is applied).

SD	MF	2×	Input Scale	#Layers	Window Scale	Hidden Size	Output Scale	Params. (M)	FLOPs (G)	mAP
			8^{-1}	18	16^{-1}	384	$\underline{8^{-1}}$	72.1	534.1	44.5
					8^{-1}				540.9	48.2
					4^{-1}				567.9	50.1
					2^{-1}				676.2	50.7
					1				1109.1	50.8

SD	MF	2×	Stage 1					Stage 2					Stage 3					Params. (M)	FLOPs (G)	mAP
			Input Scale	#Layers	Window Scale	Hidden Size	Output Scale	Input Scale	#Layers	Window Scale	Hidden Size	Output Scale	Input Scale	#Layers	Window Scale	Hidden Size	Output Scale			
✓			8^{-1}	6	1	384	8^{-1}	16^{-1}	6	1	384	16^{-1}	32^{-1}	6	1	384	$\underline{32^{-1}}$	72.1	607.1	41.0
✓				8					5					5					688.28	42.0
✓				10					4					4					769.47	42.6
✓				12					3					3					850.68	43.0
✓				14					2					2					931.88	43.4
	✓		8^{-1}	6	16^{-1} 8^{-1} 4^{-1} 2^{-1} 1	384	$\underline{8^{-1}}$	8^{-1}	6	16^{-1} 8^{-1} 4^{-1} 2^{-1} 1	384	$\underline{16^{-1}}$	8^{-1}	6	16^{-1} 8^{-1} 4^{-1} 2^{-1} 1	384	$\underline{32^{-1}}$	72.1	534.3	44.3
	✓																		541.03	47.6
	✓																		568.09	49.4
	✓																		676.33	50.3
	✓																		1109.3	50.2
		✓	8^{-1}	6	16^{-1} 8^{-1} 4^{-1} 2^{-1} 1	152	8^{-1}	8^{-1}	6	16^{-1} 8^{-1} 4^{-1} 2^{-1} 1	304	8^{-1}	8^{-1}	6	16^{-1} 8^{-1} 4^{-1} 2^{-1} 1	608	$\underline{8^{-1}}$	73.8	558.4	43.4
		✓																	561.5	44.4
		✓																	587.7	46.3
		✓																	692.2	46.6
		✓																	1110.2	48.3
✓	✓		8^{-1}	2	1	384	8^{-1}	16^{-1}	8	1	384	16^{-1}	32^{-1}	8	1	384	$\underline{32^{-1}}$	72.1	459.7	45.8
✓	✓			4					7					7					540.9	47.5
✓	✓			6					6					6					622.1	48.5
✓	✓			8					5					5					703.3	48.0
✓	✓			10					4					4					784.5	48.6
✓	✓			12					3					3					865.7	50.2
✓	✓			15					2					1					989.5	50.4
✓		✓	8^{-1}	16	1	128 160 192 224 256	8^{-1}	16^{-1}	1	1	256 320 384 448 512	16^{-1}	32^{-1}	9 5 3 2 1	1	512 640 768 896 1024	$\underline{32^{-1}}$	70.2 69.3 69.3 71.4 69.2	529.1 581.7 637.4 696.6 756.5	37.6 38.9 40.2 41.7 42.5
✓		✓																		
✓		✓																		
✓		✓																		
✓		✓																		
	✓	✓	8^{-1}	6	16^{-1} 8^{-1} 4^{-1} 2^{-1}	152	$\underline{8^{-1}}$	8^{-1}	6	16^{-1} 8^{-1} 4^{-1} 2^{-1}	304	$\underline{16^{-1}}$	8^{-1}	6	16^{-1} 8^{-1} 4^{-1} 2^{-1}	608	$\underline{32^{-1}}$	73.8	566.3	45.7
	✓	✓																	569.5	46.4
	✓	✓																	595.6	48.1
	✓	✓																	700.1	49.0
✓	✓	✓	8^{-1}	16	1	128 160 192 224 256	8^{-1}	16^{-1}	1	1	256 320 384 448 512	16^{-1}	32^{-1}	9 5 3 2 1	1	512 640 768 896 1024	$\underline{32^{-1}}$	73.3 72.4 72.4 74.5 72.4	552.1 604.9 660.7 719.9 779.9	44.3 45.5 47.6 48.8 49.4
✓	✓	✓																		
✓	✓	✓																		
✓	✓	✓																		
✓	✓	✓		28		224					448			1		896		72.1	992.3	49.5

Table 5: Model architectures in Figure 4 and Figure 5 (MS-COCO) (in main body). Configurations (depth, width) of UViT-T/S/B are annotated.

Input Size	Depth	Width	Params. (M)	FLOPs (G)	mAP
640×640	18	384	72.1	676.2	50.4
		432	80.9	748.3	50.5
		462	86.9	796.6	50.7
		492	93.3	847.4	50.4
		564	110.2	979.4	50.1
768×768	18	288	58.2	725.9	51.1
		306	60.7	761.1	51.5
		330	64.3	810.0	51.3
		384	73.1	928.5	51.5
		432	82.1	1043.5	51.6
		462	88.2	1120.1	51.3
896×896	18	186	47.4	710.2	51
		222 (UViT-T)	51.0	801.4	51.3
		246	53.8	866.1	51.7
		288 (UViT-S)	59.2	986.8	51.7
		330	65.4	1117.1	52.1
		384 (UViT-B)	74.4	1298.7	52.3
1024×1024	18	120	42.6	710.3	47.9
		132	43.5	750.1	48.9
		144	44.4	791.0	49.3
		162	45.8	854.3	50.4
		198	49.3	987.6	51.4
		246	54.7	1179.7	51.7
		288	60.3	1361.2	52.0
896×896	12	276	52.1	748.4	50.8
		300	54.4	796.2	50.8
		324	56.9	846.2	51.0
		360	60.9	925.0	51.5
		390	64.5	994.2	51.5
896×896	24	156	46.5	739.0	50.6
		180	49.2	813.8	50.8
		192	50.6	852.7	51.3
		258	60.1	1085.4	51.8
		294	66.3	1225.7	51.6
896×896	32	120	44.6	732.5	50
		132	45.9	777.4	50.4
		144	47.3	823.8	51.2
		180	52.3	971.1	51.5
		240	62.8	1244.4	52.0
896×896	40	96	43.2	723.2	48.5
		102	43.8	749.3	49.1
		114	45.2	802.9	50.1
		126	46.8	858.2	50.7
		150	50.3	974.0	51.2
		156	51.2	1004.0	51.2

References

1. Amirul Islam, M., Rochan, M., Bruce, N.D., Wang, Y.: Gated feedback refinement network for dense image labeling. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3751–3759 (2017)
2. Artacho, B., Savakis, A.: Waterfall atrous spatial pooling architecture for efficient semantic segmentation. Sensors **19**(24), 5361 (2019)
3. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Semantic image segmentation with deep convolutional nets and fully connected crfs. arXiv preprint arXiv:1412.7062 (2014)
4. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE transactions on pattern analysis and machine intelligence **40**(4), 834–848 (2017)
5. Chen, L.C., Papandreou, G., Schroff, F., Adam, H.: Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587 (2017)
6. Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with atrous separable convolution for semantic image segmentation. In: Proceedings of the European conference on computer vision (ECCV). pp. 801–818 (2018)
7. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)
8. Ghiasi, G., Fowlkes, C.C.: Laplacian pyramid reconstruction and refinement for semantic segmentation. In: European conference on computer vision. pp. 519–534. Springer (2016)
9. Hariharan, B., Arbeláez, P., Bourdev, L., Maji, S., Malik, J.: Semantic contours from inverse detectors. In: 2011 International Conference on Computer Vision. pp. 991–998. IEEE (2011)
10. Lin, G., Milan, A., Shen, C., Reid, I.: Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1925–1934 (2017)
11. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European conference on computer vision. pp. 740–755. Springer (2014)
12. Peng, C., Zhang, X., Yu, G., Luo, G., Sun, J.: Large kernel matters–improve semantic segmentation by global convolutional network. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4353–4361 (2017)
13. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention. pp. 234–241. Springer (2015)
14. Wang, P., Chen, P., Yuan, Y., Liu, D., Huang, Z., Hou, X., Cottrell, G.: Understanding convolution for semantic segmentation. In: 2018 IEEE winter conference on applications of computer vision (WACV). pp. 1451–1460. IEEE (2018)
15. Zhai, X., Kolesnikov, A., Houlsby, N., Beyer, L.: Scaling vision transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12104–12113 (2022)
16. Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2881–2890 (2017)