Supplementary material for paper "ScalableViT: Rethinking the Context-oriented Generalization of Vision Transformer"

Anonymous Authors

Affiliation

A Additional Analyses for IWSA

As shown in Fig. 1, IWSA is composed of a window-based self-attention (WSA) and a local interactive module (LIM). WSA splits the global self-attention into many limited windows and yields a collection of discrete *value* matrices. LIM build connections between these *value* matrices through a fusion function $\mathcal{F}$. In practice, this function is replaced with a 3×3 depth-wise convolution. Additionally, WSA can be viewed as a 7×7 depth-wise convolution with an adaptive weight. Thus, $\mathcal{F}$ brings information exchange through a kind of interleaving effect (illustrated by yellow squares in Fig. 1). This parallel stagger makes IWSA realize a global receptive field in a single layer.

In Table 1, we compare the LIM and the LEM on the ADE20K [17] using Semantic FPN [6] framework. All settings are recorded in the Section C. ScalableViT-S with the LIM achieves +3.8 mIoU than the LEM under the same overhead because IWSA can model the long-range dependency in single layer. This result also proves that the global receptive field plays a more critical role on the downstream vision task. Moreover, the LIM can be expanded to other window-based self-attention with different window division styles.

Table 1: LIM vs. LEM on ADE20K using Semantic FPN. #Param. refers to total parameters of Semantic FPN based on ScalableViT-S backbone. FLOPs are measured at resolution 512×2048.

Model	#Param.	FLOPs	Top-1	mIoU(%)
ScalableViT-S w. LEM	30M	174G	83.0	41.1
ScalableViT-S w. LIM	30M	174G	**83.1**	**44.9**

B Comparing visualizations from other blocks

We visualize the feature maps after the 2nd, 4th, and 24th blocks in Figure 2. In the 2nd and 4th blocks, the WSA focuses on local regions, especially the ears and

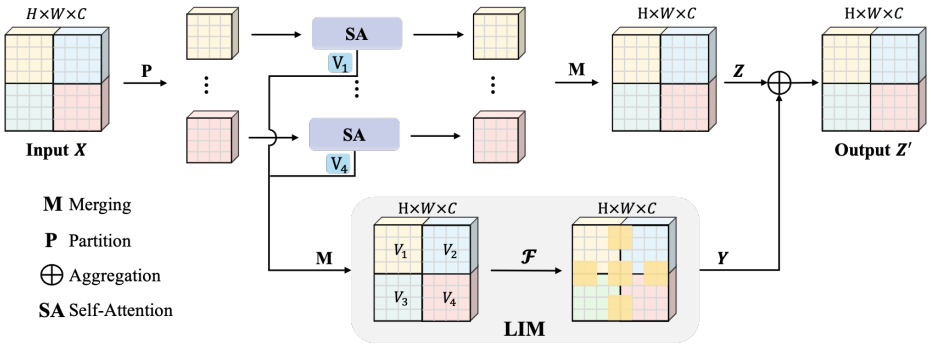

Fig. 1: Interactive Window-based Self-Attention (IWSA). Besides the proposed LIM, other parts compose the WSA. The LIM extracts a set of discrete *value* matrices $\{V_1, V_2, V_3, V_4\}$ from WSA and merges them via a fusion function $\mathcal{F}$. The output Y is added on Z for an output Z' with information interaction.

nose. In the latter 24th block, the WSA attends to contextual information but losses some semantic cues. Since feature aggregation from a large downsampling ratio (16) causes the foreground and background to be poorly separated. By contrast, the SSA can retain a trail of details although the feature map of the later block are not as continuous as the earlier ones.

C More Implementary Details

Classification. The classification settings mainly follow DeiT [12]. All variants are trained under a resolution of 224 × 224. During training from scratch, we employ the AdamW optimizer [9] with a weight decay of 0.05 and a momentum of 0.9 to train models for 300 epochs. The learning rate is set to 0.001 initially and varies with the cosine scheduler, where a 5-epochs linear warm-up is used to stabilize training. The global batchsize is set to 1024 on 8 V100 GPUs. Moreover, we apply data augmentations and regularizations, including random cropping, random horizontal flipping [10], mixup [15], CutMix [14], random erasing [16], label-smoothing [11], stochastic depth [5], and repeated augmentation [4]. For stochastic depth augmentation, we set the drop rate to 0.2, 0.5, and 0.5 for ScalableViT-S, ScalableViT-B, and ScalableViT-L, respectively. During testing on the validation set, the shorter side of an input image is first resized to 256, and a center crop of 224 × 224 is used to evaluate the classification accuracy.

Object Detection. We adopt RetinaNet [7] and Mask R-CNN [3] detection frameworks on COCO [8] that contains 118K training images and 5K validation images. Before training, we initialize the backbone with the weight pre-trained on ImageNet-1K, FPN with Xavier [2] scheme, and other new layers with Normal scheme ($std = 0.01$). All models utilize AdamW [9] optimizer, 500-iteration

Table 2: Settings of the initial learning rate and weight decay.

Model	#lr scheduler	learning rate	weight decay
Object Detection			
RetinaNet(1×)	Multi-step	1×10^{-4}	1×10^{-4}
RetinaNet(3×)	Multi-step	1×10^{-4}	5×10^{-2}
Mask R-CNN(1×)	Multi-step	2×10^{-4}	1×10^{-4}
Mask R-CNN(3×)	Multi-step	1×10^{-4}	5×10^{-2}
Semantic Segmentation			
Semantic FPN	Polynomial	1×10^{-4}	1×10^{-4}
UperNet	Polynomial	6×10^{-5}	1×10^{-2}

warm-up, 1× (12 epochs), and 3× (36 epochs) schedule with a global batch size of 16 on 8 GPUs. Settings of initial learning rate and weight decay are shown in Table 2. For 1× schedule, the short side of training images is resized to 800 pixels, and the long side is never more than 1333 pixels. The learning rate is declined at the $8th$ and $11th$ epoch with a decay rate of 0.1. For the 3× schedule, we adopt the multi-scale training, which randomly resizes the short side of the input images within the range of [480, 800] while keeping the longer side at most 1333. The learning rate is declined at the $27th$ and $33rd$ with a decay rate of 0.1. When testing, the image size is set as the same as the 1× schedule.

Semantic Segmentation. Semantic segmentation experiments are conducted on the challenging ADE20K [17], with 20K images for training and 2K images for validation. We use the typical Semantic FPN [6] and UperNet [13] as segmentation frameworks to evaluate our models. Following the common practice, we use the MMSegmentation [1] to implement all related experiments. We employ the AdamW [9] to optimize two models. The initial learning rate and weight decay are shown in Table 2. For the Semantic FPN, we train 80K iterations with a batch size 16 on 4 GPUs. The polynomial policy schedules the learning rate with a power of 0.9. For the UperNet, we train 160K iterations with a batch size 16 on 8 GPUs. The polynomial policy schedules the learning rate with a power of 1.0. During training, we first resize the short side of input images to 512 pixels, and the long side is never more than 2048 pixels, then randomly crop to 512×512. During testing, we resize input images the same as the training phase but without cropping. We also use the test time augmentation for UperNet, including multi-scale test ($[0.5, 0.75, 1.0, 1.25, 1.5, 1.75]\times$ resolution) and flip, for better results.

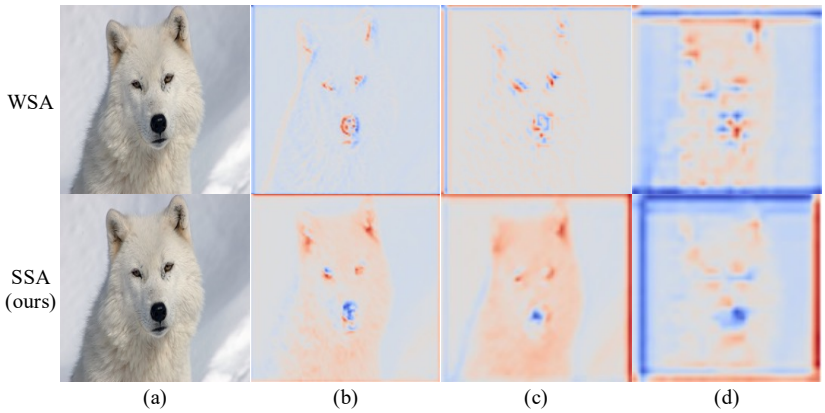

Fig. 2: Visualization for feature maps of other blocks. (b), (c), and (d) are output features of the 2nd, 4th, and 24th blocks, respectively.

References

1. Contributors, M.: MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation (2020)
2. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Teh, Y.W., Titterington, D.M. (eds.) Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. vol. 9, pp. 249–256 (2010)
3. He, K., Gkioxari, G., Dollár, P., Girshick, R.B.: Mask R-CNN. IEEE TPAMI **42**(2), 386–397 (2020)
4. Hoffer, E., Ben-Nun, T., Hubara, I., Giladi, N., Hoefler, T., Soudry, D.: Augment your batch: Improving generalization through instance repetition. In: CVPR. pp. 8126–8135 (2020)
5. Huang, G., Sun, Y., Liu, Z., Sedra, D., Weinberger, K.Q.: Deep networks with stochastic depth. In: ECCV. vol. 9908, pp. 646–661 (2016)
6. Kirillov, A., Girshick, R.B., He, K., Dollár, P.: Panoptic feature pyramid networks. In: CVPR. pp. 6399–6408 (2019)
7. Lin, T., Goyal, P., Girshick, R.B., He, K., Dollár, P.: Focal loss for dense object detection. IEEE TPAMI **42**(2), 318–327 (2020)
8. Lin, T., Maire, M., Belongie, S.J., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: common objects in context. In: ECCV. Lecture Notes in Computer Science, vol. 8693, pp. 740–755 (2014)
9. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: ICLR (2019)
10. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S.E., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: CVPR. pp. 1–9 (2015)
11. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: CVPR. pp. 2818–2826 (2016)
12. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. In: Proceedings of the 38th International Conference on Machine Learning, ICML. vol. 139, pp. 10347–10357 (2021)
13. Xiao, T., Liu, Y., Zhou, B., Jiang, Y., Sun, J.: Unified perceptual parsing for scene understanding. In: ECCV. Lecture Notes in Computer Science, vol. 11209, pp. 432–448 (2018)
14. Yun, S., Han, D., Chun, S., Oh, S.J., Yoo, Y., Choe, J.: Cutmix: Regularization strategy to train strong classifiers with localizable features. In: ICCV. pp. 6022–6031 (2019)
15. Zhang, H., Cissé, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. In: ICLR (2018)
16. Zhong, Z., Zheng, L., Kang, G., Li, S., Yang, Y.: Random erasing data augmentation. In: AAAI. pp. 13001–13008 (2020)
17. Zhou, B., Zhao, H., Puig, X., Xiao, T., Fidler, S., Barriuso, A., Torralba, A.: Semantic understanding of scenes through the ADE20K dataset. Int. J. Comput. Vis. **127**(3), 302–321 (2019)