ResT V2: Simpler, Faster and Stronger

Qing-Long Zhang, Yu-Bin Yang
State Key Laboratory for Novel Software Technology
Nanjing University, Nanjing 21023, China
wofmanaf@smail.nju.edu.cn, yangyubin@nju.edu.cn

A Comparison with different MSA variants.

This section provides more results about the effectiveness of the upsampling operation in different MSA variants under 100 training epochs settings following Section 4.3. As shown in Table 1, MSA variants with the "downsample-upsample" branch can compete favorably with the standard MSA in terms of the Top-1 accuracy but with much higher inference throughput.

Table 1: **Results of different MSAs.** † means the implementation of adding an upsample operation for SRA[10] or EMSA[14], and $*$ means replacing SRA or EMSA with the standard MSA, leaving other components unchanged. Inference throughput (images / s) is measured on a V100 GPU, following [14].

Methods	Params	FLOPs	Throughput	Top-1 (%)
PVT-Tiny[10]	13.27M	1.94G	1144	70.46
PVT-MSA-Tiny*	11.36M	4.81G	617	72.38
PVT-EMSAv2-Tiny†	13.95M	1.95G	947	**72.53**
ResTv1-Lite[14]	10.50M	1.44G	1091	74.108
ResTv1-MSA-Lite*	10.48M	4.37G	625	**75.06**
ResTv1-EMSAv2-Lite†	10.66M	1.45G	926	75.04

B Results of Multi-head Interaction Module

In this section, we estimate the role of the Multi-head Interaction Module (short for MHIM). Training settings are the same as Section 4.3, i.e., under the 100 training epochs settings. As shown in Table 2, MHIM can significantly improve the Top-1 accuracy with the cost of decreased inference throughput when the channel dimension of each head d_k is smaller. When d_k is larger (e.g., 96), the accuracy improvement is limited. Therefore, we may choose whether to apply MHIM in EMSAv2 according to different scenarios.

Table 2: **Results of short for MHIM.** ResTv2-Lite is a shallow ResTv2 variant with blocks number={2, 2, 2, 2} and $C = 64$. Inference throughput (images / s) is measured on a V100 GPU, following [14].

Methods	head_dim	Params	FLOPs	Throughput	Top-1 (%)
ResTv2-Lite	64	10.66M	1.45G	945	74.54
+MHIM	64	10.66M	1.45G	926(-19)	75.04(+0.5)
ResTv2-Tiny	96	30.43 M	4.10G	826	80.33
+MHIM	96	30.44M	4.10G	792(-34)	80.47 (+0.14)

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

C Experimental Settings

C.1 ImageNet pre-training and finetuning settings

We list the settings for training and fine-tuning on ImageNet-1k in Table 3. The settings are used for our main results (Section 4.2). The fine-tuning starts from the best checkpoint weights obtained in pre-training.

Table 3: **ImageNet-1k training and fine-tuning settings.** Mutiple stochastic depth rates (e.g., 0.1/0.2/0.3/0.5) are for each model (e.g., ResTv2-T/S/B/L) respectively.

configure	pre-training	fine-tuning
input crop.	224^2	384^2
optimizer	AdamW	AdamW
base learning rate	1.5e-4	1.5e-5
weight decay	0.05	1e-8
optimizer momentum	$\beta_1 = 0.9, \beta_2 = 0.999$	$\beta_1 = 0.9, \beta_2 = 0.999$
batch size	2048	512
training epoch	300	30
learning rate schedule	cosine decay	cosine decay
warmup epochs	50	N/A
warmup schedule	linear	N/A
RandAugment [3]	(9, 0.5)	(9, 0.5)
label smoothing [9]	0.1	0.1
Mixup [13]	0.8	N/A
Cutmix [12]	1.0	N/A
stochastic depth [5]	0.1/0.2/0.3/0.5	0.1/0.2/0.3/0.5
gradient clip	1.0	1.0
EMA [7]	0.9999	N/A

C.2 Downstream Tasks.

For COCO and ADE20K experiments, we follow the training settings used in ResTv1 [14] and ConvNeXt [6]. We adopt Detectron2 [11] toolboxes for object detection, and MMSegmentation [2] toolboxes for semantic segmentation. We use the best model weights (instead of EMA weights) from ImageNet pre-training as network initialization.

Object Detection on COCO. We utilize multi-scale training, i.e., resizing the input such that the short side is between 480 and 800 while the longer side is at most 1333, AdamW optimizer with initial learning rate of 0.0001, weight decay of 0.05, and batch size of 16 (8 V100 GPUs with two images per GPU) , "$\times 1$" schedule (90K iterations with learning rate decayed by $\times 10$ at 60K and 80K steps) for ablation study, and "$\times 3$" schedule (270K iterations with learning rate decayed by $\times 10$ at 210K and 250K steps) for main results. We apply standard data augmentation, that is resize, random flip and normalize. Additionally, we adopt stochastic depth with ratio of 0.1/0.2/0.3 for ResTv2-T/S/B. All results are reported on the validation set.

Semantic segmentation on ADE20K. We employ the AdamW optimizer with an initial learning rate of 1.5e-4, a weight decay of 0.05, stage-wise learning rate decay with a ratio of 0.9, and a linear warmup of 1,500 iterations. Models are trained on 8 GPUs with two images per GPU for 160K iterations. Stochastic depth with ratio of 0.1/0.2/0.3 for ResTv2-T/S/B. All models are trained on the standard setting with an input of 512×512. We report validation mIoU results using multi-scale testing.

D Detailed Architectures

The detailed architecture specifications are shown in Table 4, where an input image size of 224×224 is assumed for all architectures. "Conv-K_C_S" means convolution layers with kernel size K, output channel C and stride S. "MLP_C" is the FFN layer with hidden channel $4C$ and output channel C. "Ev2_H_R" is the EMSAv2 operation with the number of heads H and reduction ratio R. "C" is 96 for ResTv2-T/S/B, and 128 for ResTv2-L."PA" [14] is short for pixel-wise attention.

Table 4: **Detailed architecture specifications.** FLOPs are calculated with image size (224, 224).

Module	Output	ResTv2-T	ResTv2-S	ResTv2-B	ResTv2-L
stem	56×56	Patch Embedding: Conv-3_C/2_2, Conv-3_C_2, Conv-1_C_1, PA			
stage1	56×56	$\begin{bmatrix} \text{Ev2\_1\_8} \\ \text{MLP\_96} \end{bmatrix} \times 1$	$\begin{bmatrix} \text{Ev2\_1\_8} \\ \text{MLP\_96} \end{bmatrix} \times 1$	$\begin{bmatrix} \text{Ev2\_1\_8} \\ \text{MLP\_96} \end{bmatrix} \times 1$	$\begin{bmatrix} \text{Ev2\_2\_8} \\ \text{MLP\_128} \end{bmatrix} \times 2$
stage2	28×28	Patch Embedding: Conv-3_2C_2, PA			
		$\begin{bmatrix} \text{Ev2\_2\_4} \\ \text{MLP\_192} \end{bmatrix} \times 2$	$\begin{bmatrix} \text{Ev2\_2\_4} \\ \text{MLP\_192} \end{bmatrix} \times 2$	$\begin{bmatrix} \text{Ev2\_2\_4} \\ \text{MLP\_192} \end{bmatrix} \times 3$	$\begin{bmatrix} \text{Ev2\_4\_4} \\ \text{MLP\_256} \end{bmatrix} \times 3$
stage3	14×14	Patch Embedding: Conv-3_4C_2, PA			
		$\begin{bmatrix} \text{Ev2\_4\_2} \\ \text{MLP\_384} \end{bmatrix} \times 6$	$\begin{bmatrix} \text{Ev2\_4\_2} \\ \text{MLP\_384} \end{bmatrix} \times 12$	$\begin{bmatrix} \text{Ev2\_4\_2} \\ \text{MLP\_384} \end{bmatrix} \times 16$	$\begin{bmatrix} \text{Ev2\_8\_2} \\ \text{MLP\_512} \end{bmatrix} \times 16$
stage4	7×7	Patch Embedding: Conv-3_8C_2, PA			
		$\begin{bmatrix} \text{Ev2\_8\_1} \\ \text{MLP\_768} \end{bmatrix} \times 2$	$\begin{bmatrix} \text{Ev2\_8\_1} \\ \text{MLP\_768} \end{bmatrix} \times 2$	$\begin{bmatrix} \text{Ev2\_8\_1} \\ \text{MLP\_768} \end{bmatrix} \times 3$	$\begin{bmatrix} \text{Ev2\_16\_1} \\ \text{MLP\_1024} \end{bmatrix} \times 2$
Classifier		Average Pooling, 1000D Fully-Connected Layer			
FLOPs		4.0G	5.8G	7.7G	13.5G

E Positional Embedding

In Section **??**, we explore three types of positional embedding in ResTv2. Here, we give the mathematical definition of them. Assume $\mathrm{x} \in \mathbb{R}^{hw \times d_m}$ be the input token sequence, where h, w and d_m are the input's height, width and channel dimensions, respectively.

APE. Let $\theta \in \mathbb{R}^{n \times c}$ be position parameters, then APE[4] can be represented as

$$\hat{x} = x + \theta \tag{1}$$

In APE, θ should be the same size as the input x, i.e., $n = h \cdot w$, whatever θ is sinusoidal or learnable.

RPE. RPE applied in this paper is the relative position variant introduced in [8], which is added to the attention map to represent the structure of inputs (i.e., 2D images). Let Q be the queries, K be the keys, k be the number of heads, and d_k be the head dimension. Then position P can be obtained by $P = P_h + P_w$, where $P_h \in \mathbb{R}^{k \times h' \times 1 \times d_k}$ and $P_w \in \mathbb{R}^{k \times 1 \times w' \times d_k}$ represent the positions along the height dimension and the width dimension, respectively. The RPE can then be defined as:

$$\mathrm{Attn}(Q, K) = \mathrm{Softmax}(\frac{Q(K+P)^{\mathrm{T}}}{\sqrt{d_k}}) \tag{2}$$

Therefore, RPE can capture both the "content-content" and "content-position" information. However, P is required to share the same resolution as K, i.e., $h' = h, w' = w$ in the standard MSA, and $h' = h/r, w' = w/r$ in EMSAv2, where r is the reduction factor.

PA. PA is one kind of spatial attention that utilizes a simple 2D depth-wise convolution as the transform function [14]. Thanks to the shared parameters and locality of convolution, PA can tackle arbitrary lengths of inputs and capture their absolute positions with the help of zero paddings.The mathematical formulation of PA can be defined as

$$\mathrm{PA}(x) = x \cdot \sigma(\mathrm{DWConv}(x)) \tag{3}$$

where $\sigma(\cdot)$ is the Sigmoid function adopted to scale the spatial attention weights.

PEG. PEG [1] adopt a 2D depth-wise convolution to obtain position parameters, i.e.,

$$\hat{x} = x + \mathrm{DWConv}(x) \tag{4}$$

Similar to PA, PEG can tackle arbitrary lengths of inputs without interpolating.

References

[1] Xiangxiang Chu, Bo Zhang, Zhi Tian, Xiaolin Wei, and Huaxia Xia. Conditional positional encodings for vision transformers. *arXiv preprint arXiv:2102.10882*, 2021.

[2] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation, 2020.

[3] Ekin Dogus Cubuk, Barret Zoph, Jon Shlens, and Quoc Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Advances in Neural Information Processing Systems, NeurIPS 2020*, 2020.

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021*. OpenReview.net, 2021.

[5] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. Deep networks with stochastic depth. In *14th European Conference on Computer Vision, ECCV 2016*, volume 9908, pages 646–661. Springer, 2016.

[6] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022*, 2022.

[7] Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855, 1992.

[8] Aravind Srinivas, Tsung-Yi Lin, Niki Parmar, Jonathon Shlens, Pieter Abbeel, and Ashish Vaswani. Bottleneck transformers for visual recognition. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021*, pages 16519–16529. Computer Vision Foundation / IEEE, 2021.

[9] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016*, pages 2818–2826. IEEE Computer Society, 2016.

[10] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021*, pages 548–558. IEEE, 2021.

[11] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019.

[12] Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Seong Joon Oh, Youngjoon Yoo, and Junsuk Choe. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019*, pages 6022–6031. IEEE, 2019.

[13] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *6th International Conference on Learning Representations, ICLR 2018*. OpenReview.net, 2018.

[14] Qinglong Zhang and Yu bin Yang. Rest: An efficient transformer for visual recognition. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems, NeurIPS 2021*, 2021.