ResT: An Efficient Transformer for Visual Recognition

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 Appendix

We provide the related work and more experimental results to complete the experimental sections of the main paper.

A.1 Related Work

Convolutional Networks. As the cornerstone of deep learning computer vision, CNNs have been evolved for years and are becoming more accurate and faster. Among them, the ResNet series [8, 20, 22] are the most famous backbone networks because of their simple design and high performance. The base structure of ResNet is the residual bottleneck, which can be defined as a stack of one 1×1, one 3×3, and one 1×1 Convolution layer with residual learning. Recent works explore replacing the 3×3 Convolution layer with more complex modules [15, 10] or combining with attention modules [24, 19]. Similar to vision Transformers, CNNs can also capture long-range dependencies if correctly incorporated with self-attention such as Non-Local or MSA. These studies show that the advantage of CNN lies in parameter sharing and focuses on the aggregation of local information, while the advantage of self-attention lies in the global receptive field and focuses on the aggregation of global information. Intuitively speaking, global and local information aggregation are both useful for vision tasks. Effectively combining global information aggregation and local information aggregation may be the right direction for designing the best network architecture.

Vision Transformers. Transformer is a type of neural network that mainly relies on self-attention to draw global dependencies between input and output. Recently, Transformer-based models are explored to solve various computer vision tasks such as image processing [3], classification [5, 4, 21], and object detection [2, 25], etc. Here, we focus on investigating the classification vision Transformers. These Transformers usually view an image as a sequence of patches and perform classification with a Transformer encoder. The encoder consists of several Transformer blocks, each including an MSA and an FFN. Layer-norm (LN) is applied before each layer and residual connections are employed in both the self-attention and FFN module.

Among them, ViT [4] is the first fully Transformer classification model. In particular, ViT split each image into 14×14 or 16×16 with a fixed length, then several Transformer layers are adopted to model global relation among these tokens for input classification. DeiT [16] explores the data-efficient training and distillation of ViT. Tokens-to-Tokens (T2T-ViT) [21] point out that the simple tokenization of input images in ViT fails to model the important local structure (e.g., edges, lines) among neighboring pixels, leading to its low training sample efficiency. Transformer-in-Transformer (TNT) [6] split each image into a sequence of patches and each patch is reshaped to pixel sequence. After embedding, two Transformer layers are applied for representation learning where the outer Transformer layer models the global relationships among the patch embedding and the inner one extracts local structure information of pixel embedding. Pyramid Vision Transformer (PVT) [18] follows the ResNet paradigm to construct Transformer backbones, making it more suitable for downstream tasks. MViT [7] further apply pooling function to reduce computation costs.

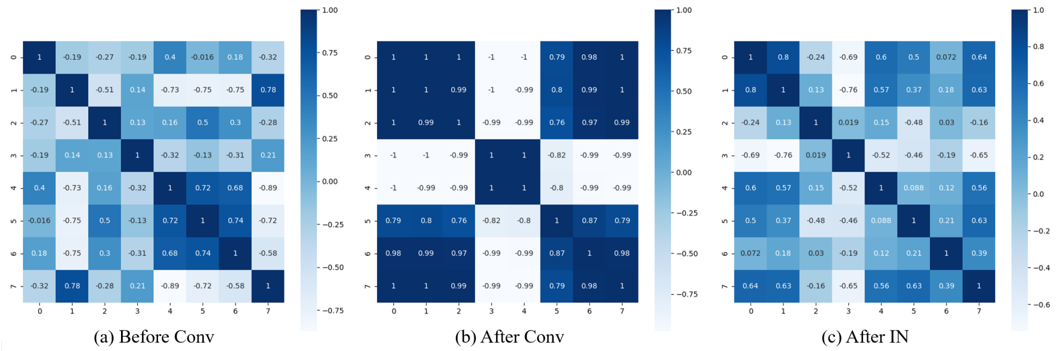
[*]Corresponding author.

Positional Encoding. Different from CNNs, which can implicitly encode spatial position information by zero-padding [9], the self-attention in Transformers has no ability to distinguish token order in different positions. Therefore, positional encoding is essential for the patch embeddings to retain positional information. There are mainly two types of positional encoding most commonly applied in vision Transformers, i.e., absolute positional encoding and relative positional encoding. The absolute positional encoding is used in ViT [4] and its extended methods, where a standard learnable 1D position embedding is added to the sequence of embedded patches. The relative method is used in BoTNet [15] and Swin Transformer [13], where the split 2D relative position embeddings are added. Generally speaking, the relative positional encodings are better suited for vision tasks, this can be attributed to attention not only taking into account the content information but also relative distances between features at different locations [14].

A.2 Visualization and Interpretation

Analysis on EMSA. In this part, we measure the diversity of EMSA. To simplify, we apply $\mathbf{A} \in \mathbb{R}^{k\times n\times n'}$ to denote the attention map, with k be the number of heads in EMSA. Assume $\mathbf{A}_i \in \mathbb{R}^{1\times m}$ is the i-th attention map (after reshape), where $m = nn'$. Then we can compute the cross-layer similarity to measure the diversity of different heads.

$$\mathrm{M}_{ij} = \frac{\mathbf{A}_i\mathbf{A}_j^{\mathrm{T}}}{\|\mathbf{A}_i\|\|\mathbf{A}_j\|} \tag{1}$$



(a) Before Conv (b) After Conv (c) After IN

Figure 1: Attention map visualization of the last blocks of stage 4 of the ResT-Lite.

To thoroughly measure the diversity, we calculate extract three types of $\mathbf{A}$: the attention map before Conv-1 (i.e., $\mathbf{QK}^{\mathrm{T}}/\sqrt{d_k}$), the one after Conv-1 (i.e., $\mathrm{Conv}(\mathbf{QK}^{\mathrm{T}}/\sqrt{d_k})$), and the one after Instance Normalization [17] (i.e., $\mathrm{IN}(\mathrm{Softmax}(\mathrm{Conv}(\mathbf{QK}^{\mathrm{T}}/\sqrt{d_k})))$). We randomly sample 1,000 images from the ImageNet-1k validation set and visualize the mean diversity in Figure 1.

As shown in Figure 1b, although the 1×1 convolution model the interactions among different heads, it impairs the ability of MSA to jointly attend to information from different representation subsets at different positions. After instance normalization (in Figure 1c), the diversity ability is restored.

Interpretabilty. In order to validate the effectiveness of ResT more intuitively, we sample 6 images from the ImageNet-1k validation split. We use Group-CAM [23] to visualize the heatmaps at the last convolutional layer of ResT-Lite. For comparison, we also draw the heatmaps of its counterpart ResNet-50. As shown in Figure 2, the proposed ResT-Lite can adaptively produce heatmaps according to the image content.

A.3 More Experiments

Comparisons of EMSA and MSA. In this part, we make a quantitative comparison of the performance and efficiency of the two modules on ResT-Lite (keeping other components intact). Experimental settings are the same as Ablation Study (Section **??**7), except for the batch size of MSA, which is 512 since each V100 GPU can only tackle 64 images at the same time. We report the results

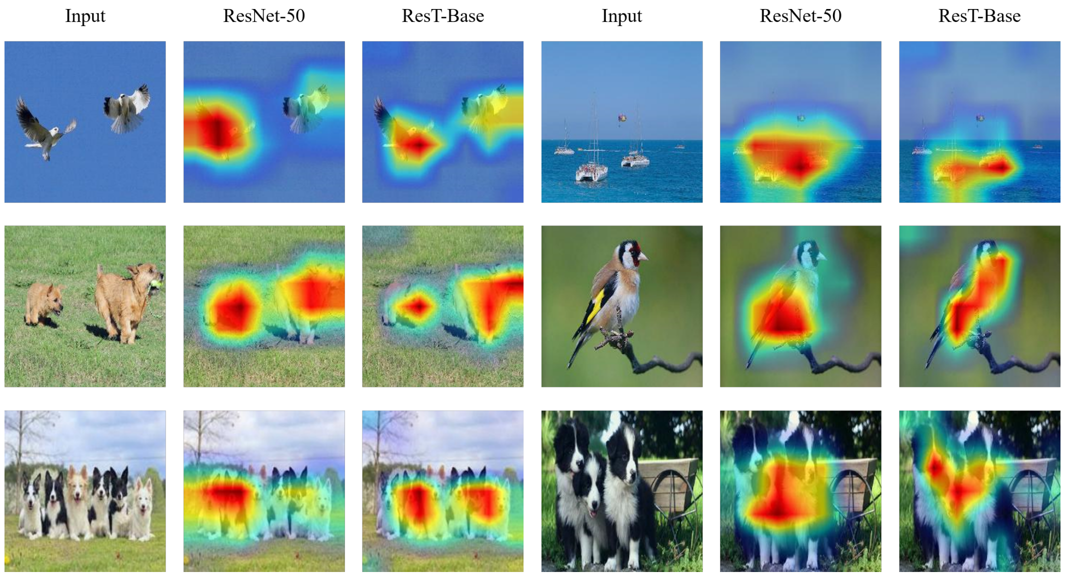


Figure 2: Sample visualization on ImageNet-1k val split generated by Group-CAM [23].

in Table 1. Both versions of ResT-Lite share almost the same parameters. However, the EMSA version achieves better Top-1 accuracy (+0.2%) with fewer computations(-0.2G). The actual inference throughput indicates EMSA (1246) is 2.4x faster than the original MSA(512). Therefore, EMSA can capably serve as an effective replacement for MSA.

Table 1: Comparison of MSA and EMSA.

Model	#Params (M)	FLOPs (G)	Throughput	Top-1 (%)	Top-5 (%)
MSA	10.48	1.6	512	72.68	90.46
EMSA	10.49	1.4	1246	72.88	90.62

Object Detection. In Section 3.2, we replace the backbone in RetinaNet [12] with ResT and add a layer normalization (LN [1]) for the output of each stage (before FPN [11]), just like Swin. Here, we further validate the effectiveness of LN. We follow the same setting as Section **??**. Results are reported in Table 2.

Table 2: Object Detection.

Backbones	Setting	AP50:95
ResT-Small	w/o LN	39.5
	w LN	40.3
ResT-Base	w/o LN	41.2
	w LN	42.0

From Table 2, we can see, LN is indeed matters in downstream tasks. An average +0.8 box AP improvement is achieved with LN for RetinaNet [12].

A.4 Discussions

Mathematical Definition of GL. Given x with size $\mathbb{R}^{n \times d_m}$, where n is spatial dimension and d_m is the channel dimension, GL first splits x into g non-overlapping groups,

$$x = Concat(x_1, \cdots, x_g) \tag{2}$$

where the size of x_i is $n \times \frac{d_m}{g}$.

All x_i's are then simultaneously transformed by g linear operations to produce g outputs

$$y_i = x_i W_i \tag{3}$$

where W_i is the linear operation weight.

y_i's are then concatenated to produce the final $n \times d_m$ output $y = Concat(y1, \cdots, y_g)$. In ResT, we set $g = d_m$, i.e., the channel dimension of the input.

Ablation Study Settings. Note that the settings between ablation study and the main results are different. Here, we give the explanations. We adopt the simplest data augmentation and hyper-parameters settings in ResNet [8] to thoroughly investigate the important components of ResT, i.e., eliminating the influence of strong data augmentation and training tricks. Under this setting, we demonstrate that Vision Transformers can still achieve better results without training tricks. Specifically, the Top-1 accuracy of ResT-Lite is 72.88, which outperforms the ResNet-18 (69.76) by +3.1 improvement. We believe the same setting as ResNet in the ablation study can eliminate the doubt of Vision Transformer to some extent, i.e., the improvements of Vision Transformer over CNN mainly come with strong data augmentation and training tricks. This can promote the ongoing and future research of Vision Transformer.

References

[1] Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *CoRR*, abs/1607.06450, 2016.

[2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part I*, volume 12346 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2020.

[3] Hanting Chen, Yunhe Wang, Tianyu Guo, Chang Xu, Yiping Deng, Zhenhua Liu, Siwei Ma, Chunjing Xu, Chao Xu, and Wen Gao. Pre-trained image processing transformer. *CoRR*, abs/2012.00364, 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, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[5] Peng Gao, Jiasen Lu, Hongsheng Li, Roozbeh Mottaghi, and Aniruddha Kembhavi. Container: Context aggregation network. *CoRR*, abs/2106.01401, 2021.

[6] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *CoRR*, abs/2103.00112, 2021.

[7] Karttikeya Mangalam Yanghao Li Zhicheng Yan Jitendra Malik Christoph Feichtenhofer Haoqi Fan, Bo Xiong. Multiscale vision transformers. *arXiv:2104.11227*, 2021.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016.

[9] Md. Amirul Islam, Sen Jia, and Neil D. B. Bruce. How much position information do convolutional neural networks encode? In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[10] Xiang Li, Wenhai Wang, Xiaolin Hu, and Jian Yang. Selective kernel networks. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 510–519. Computer Vision Foundation / IEEE, 2019.

[11] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 936–944, 2017.

[12] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2999–3007. IEEE Computer Society, 2017.

[13] 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. *CoRR*, abs/2103.14030, 2021.

[14] Niki Parmar, Prajit Ramachandran, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. Stand-alone self-attention in vision models. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 68–80, 2019.

[15] Aravind Srinivas, Tsung-Yi Lin, Niki Parmar, Jonathon Shlens, Pieter Abbeel, and Ashish Vaswani. Bottleneck transformers for visual recognition. *CoRR*, abs/2101.11605, 2021.

[16] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *CoRR*, abs/2012.12877, 2020.

[17] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *CoRR*, abs/1607.08022, 2016.

[18] 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. *arXiv preprint arXiv:2102.12122*, 2021.

[19] Xiaolong Wang, Ross B. Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. *CoRR*, abs/1711.07971, 2017.

[20] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 5987–5995. IEEE Computer Society, 2017.

[21] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Francis E. H. Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. *CoRR*, abs/2101.11986, 2021.

[22] Hang Zhang, Chongruo Wu, Zhongyue Zhang, Yi Zhu, Zhi Zhang, Haibin Lin, Yue Sun, Tong He, Jonas Mueller, R. Manmatha, Mu Li, and Alexander J. Smola. Resnest: Split-attention networks. *CoRR*, abs/2004.08955, 2020.

[23] Qing-Long Zhang, Lu Rao, and Yubin Yang. Group-cam: Group score-weighted visual explanations for deep convolutional networks. *CoRR*, abs/2103.13859, 2021.

[24] Qing-Long Zhang and Yu-Bin Yang. Sa-net: Shuffle attention for deep convolutional neural networks. *CoRR*, abs/2102.00240, 2021.

[25] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable DETR: deformable transformers for end-to-end object detection. *CoRR*, abs/2010.04159, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [N/A]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]