VOLO: Vision Outlooker for Visual Recognition

1、Motivation

虽然ViT很nice，但是脱离了大规模数据集效果会不如CNN。这是因为ViT无法建模细粒度的上下文特征。因此作者提出了vision outlooker来解决这个问题。

2、Apporach

2.1 Outlooker

outlooker包含了一个用于编码空间信息的outlook attention层和用于融合channel information的MLP。对于一个输入$X \in \mathbb{R}^{H*W*C}$，用公式表示：

$$\widetilde{X} = OutlookAtt(LN(X)) + X$$
$$Z = MLP(LN(\widetilde{X})) + \widetilde{X}$$

①outlook attention

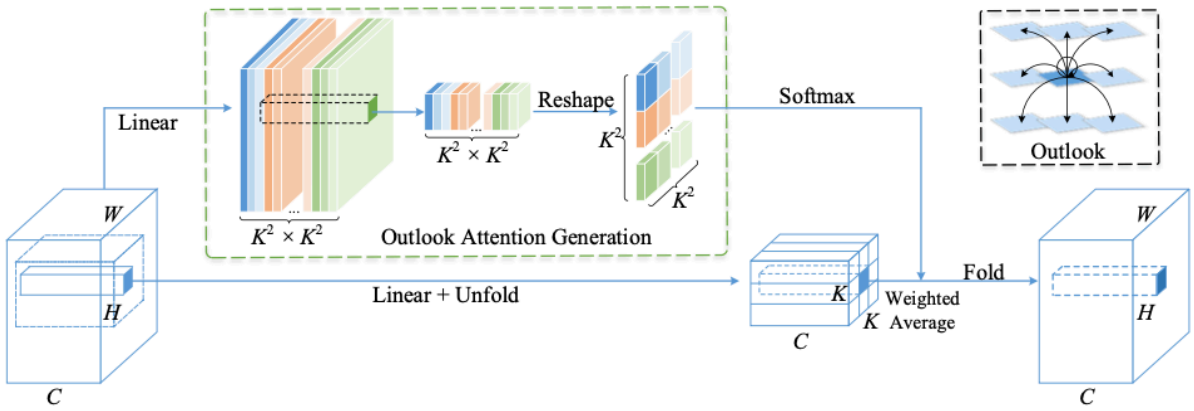

Figure 2. Illustration of outlook attention. The outlook attention matrix for a local window of size $K \times K$ can be simply generated from the center token with a linear layer followed by a reshape operation (highlighted by the green dash box). As the attention weights are generated from the center token within the window and act on the neighbor tokens and itself (as demonstrated in the black dash block), we name these operations as outlook attention.

该模块有两个基本的insights：

1. the feature at each spatial location is representative enough to generate attention weights for locally aggregating its neighboring features;
2. the dense and local spatial aggregation can encode fine-level information efficiently.

对于模块输入X空间上的每个点，outlook attention计算其与临近K*K的窗格中各个点之间的相似度。具体的计算方法为：

1. 对于输入$X \in H*W*C$，用$W_A \in \mathbb{R}^{C \times K^4}$和$W_V \in \mathbb{R}^{C \times C}$将输入分别映射成outlook weight：$A \in \mathbb{R}^{H*W*K^4}$和value representation：$V \in \mathbb{R}^{H*W*C}$。并用$V_{\triangle i,j} = \{V_{i+p-[\frac{K}{2}],j+q-[\frac{K}{2}]}\}$来表示点i，j对应的local window，其中0≤p,q≤K。
2. 对于一个点(i,j)，从A中取出对应的K^4维向量并reshape成$K^2 * K^2$的注意力矩阵$\hat{A}_{i,j}$，与$V_{\triangle i,j}$做矩阵乘法。$Y_{\triangle i,j} = MatMul(Softmax(\hat{A}_{\triangle i,j}), V_{\triangle i,j})$
3. 最后将local window中计算的Y值融合得到输出：$\widetilde{Y}_{i,j} = \sum_{0 \le m,n < K} Y_{\triangle i+m-\lfloor \frac{K}{2} \rfloor, j+n-\lfloor \frac{K}{2} \rfloor}$

具体如何实现需要看代码。不过这部分其实有一个疑问：outlooker模块中的attention weight生成是对一个点对应的向量做linear projection得到的，为什么可以表征周围部分的权值呢?

②multi-head outlook attention

这个基本上跟传统的multihead attention一致，可以直接看论文。

③discussion

作者描述了outlooker的三大优点：①计算了token之间的相似度；②使用slide window去建模细粒度的局部关系；③用reshape的简单的projection代替了self attention的KQV模式，减少的计算量。

$$\text{M-Adds}(\mathbf{SA}) \approx 4HWC^2 + 2(HW)^2C \quad (6)$$

$$\text{M-Adds}(\mathbf{LSA}) \approx 4HWC^2 + 2HWK^2C \quad (7)$$

$$\text{M-Adds}(\mathbf{OA}) \approx HWC(2C + NK^4) + HWK^2C. \quad (8)$$

Considering a normal case in which $C = 384$, $K = 3$, and $N = 6$, our outlook attention is more computationally efficient as $NK^4 < 2C$.

2.2 Network Architecture Variants

Table 2. Architecture information of different variants of VOLO. The resolution information is based on an input image of size 224×224. The number of parameters includes that of weights for both the network backbone and the classifier head. 'Layer' refers to either a Outlooker block or a Transformer block.

Specification	VOLO-D1	VOLO-D2	VOLO-D3	VOLO-D4	VOLO-D5
Patch Embedding	8×8	8×8	8×8	8×8	8×8
Stage 1 (28×28)	[head: 6, stride: 2 kernel: 3×3 mlp: 3, dim: 192] $\times 4$	[head: 8, stride: 2 kernel: 3×3 mlp: 3, dim: 256] $\times 6$	[head: 8, stride: 2 kernel: 3×3 mlp: 3, dim: 256] $\times 8$	[head: 12, stride: 2 kernel: 3×3 mlp: 3, dim: 384] $\times 8$	[head: 12, stride: 2 kernel: 3×3 mlp: 4, dim: 384] $\times 12$
Patch Embedding	2×2	2×2	2×2	2×2	2×2
Stage 2 (14×14)	[#heads: 12 mlp: 3, dim: 384] $\times 14$	[#heads: 16 mlp: 3, dim: 512] $\times 18$	[#heads: 16 mlp: 3, dim: 512] $\times 28$	[#heads: 16 mlp: 3, dim: 768] $\times 28$	[#heads: 16 mlp: 4, dim: 768] $\times 36$
Total Layers	18	24	36	36	48
Parameters	26.6M	58.7M	86.3M	193M	296M

这里的patch embedding用的是卷积。

3、Experiment

setup：

感觉数据增强很重要。同时有一些超参数的设计也很值得学习。

Table 3. Model settings. We use a linear learning rate scaling strategy $lr = \mathrm{LR}_{\mathrm{base}} \cdot \frac{\mathrm{batch\_size}}{1024}$. For all models, we set the batch size to 1024.

Specification	D1	D2	D3	D4	D5
MLP Ratio	3	3	3	3	4
Parameters	27M	59M	86M	193M	296M
Stoch. Dep. Rate	0.1	0.2	0.5	0.5	0.75
Crop Ratio	0.96	0.96	0.96	1.15	1.15
$\mathrm{LR}_{\mathrm{base}}$	1.6e-3	1e-3	1e-3	1e-3	8e-4
weight decay	5e-2	5e-2	5e-2	5e-2	5e-2

具体的实验结果不在此处阐述了，需要的话看一下paper吧。

遗留问题：

①上面提到的关于权值生成的问题

②为什么VOLO比swin transformer好?