深度学习课程论文阅读报告

PB20111689 蓝俊玮

阅读论文：

> **Swin Transformer: Hierarchical Vision Transformer using Shifted Windows** *Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, Baining Guo*; Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2021, pp. 10012-10022 [2103.14030] Swin Transformer: Hierarchical Vision Transformer using Shifted Windows (arxiv.org)

选择原因：我是从同学做 MSRA 项目时听说了 Swin Transformer，在了解到这项工作是来自 MSRA 的最佳论文后，因此才选择了 Swin Transformer 这篇论文作为我的课程阅读论文。

1. 论文摘要

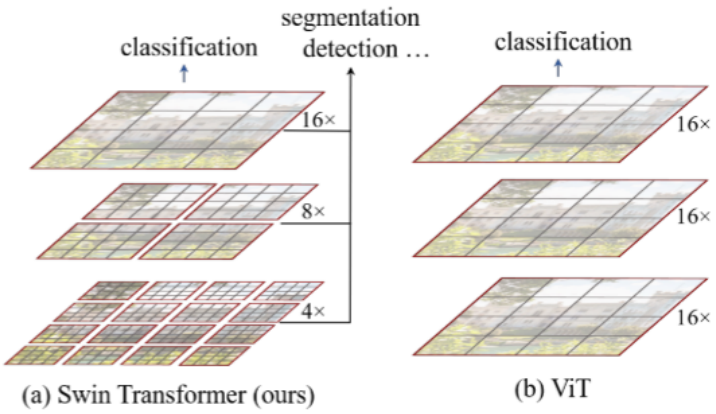
本文提出了一种新的视觉 Transformer，称为 Swin Transformer，它可以作为计算机视觉的通用主干网络。直接将 Transformer 从 NLP 任务上迁移应用到 CV 任务上的困难来源于两个领域之间的差异。例如视觉实体的尺度变化较大、图像中的像素相对于文本中的文字具有较高的分辨率。为了解决这些差异，本文提出了层级式 (Hierarchical) 的 Transformer，它的特征就是通过移动窗口 (Shifted Windows) 的的方式来计算的。移动窗口通过将自注意力的计算限制在非重叠的局部窗口，从而带来更高的效率，同时还允许跨窗口的联系。这种层级式的结构具有在不同尺度下建模的灵活性，并且具有与图像尺寸相关的线性的计算复杂度。Swin Transformer 的这些特性使其可以兼容广泛的视觉任务，包括图像分类和稠密预测任务，例如目标检测和语义分割。

2. 论文引言

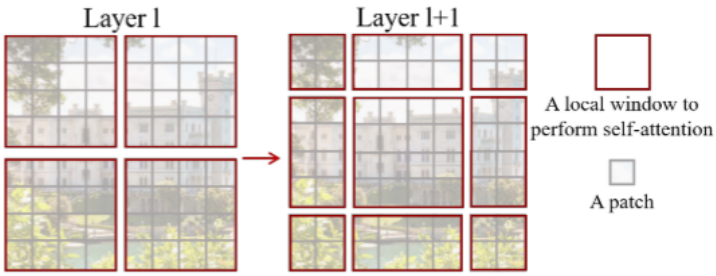
计算机视觉中的建模长期以来由卷积神经网络 (CNN) 主导。卷积神经网络结构通过更大的规模、更广泛的连接和更复杂的卷积形式变得越来越强大。随着卷积神经网络作为各种视觉任务的主干网络，这些结构的改善促进了性能的提高从而使整个领域广泛地受益。另一方面，自然语言处理 (NLP) 中网络结构的演化已经走向了一条不同的道路，现在盛行的结构是 Transformer。为了序列建模和转换任务而设计，Transformer 的特点是利用注意力来建模数据中的远程依赖关系。它在语言领域的巨大成功使研究人员开始研究它对计算机视觉的适应性，在图像分类和联合视觉语言建模任务中都表现出了很好的结果。

那么在本文之前，就已经有将 Transformer 应用到视觉领域的 Vision Transformer。Vision Tranformer 将图片分成了很多个 patch，每一个 patch 的大小是 16×16。那么假设图片的大小是 224×224，则序列长度就是 $224 \times 224 = 50176$，这个序列长度对 Transformer 来说是很难接受的，因为自注意力计算的成本非常高。那么经过 patch 处理之后，则么序列长度就变成了 16×16，这样对于 Transformer 是可以接受的。但是 Vision Transformer 将图片固定的划分成 16×16，虽然它通过这种全局的自注意力操作，可以达到全局的建模能力，但是这种固定的划分方式会影响模型对多尺寸特征的学习。对于视觉任务来说，多尺寸的特征是至关重要的。比如说对目标检测而言，运用最广的一个方法就是特征金字塔网络 (FPN)：当有一个分层式的卷积神经网络之后，每一个卷积层出来的特征的感受野是不一样的，能抓住物体不同尺寸的特征，从而能够很好的处理物体不同尺寸的问题；对于物体分割任务来说，最常见的就是 U-Net，U-Net 为了处理物体不同尺寸的问题，通过 skip-connection 可以把较浅的卷积层特征引过来，这样就可以把那些特征分辨率较高的图像细节给恢复出来，从而提高分割的准确性。但是在 Vision Transformer 中，其处理的特征都是单尺寸的，因此 Vision Transformer 并不适用于除了图像分类之外的其它视觉任务。

因此在本文中，作者希望拓展 Transformer 的适用性，使其可以作为作为计算机视觉的通用主干网络。与语言任务中的 Transformers 中作为处理的基本元素的单词 token 不同，视觉元素可以在尺度上有很大的变化。在现有的基于 Transformer 的模型中，token 都是固定尺度的，这一特性不适合这些视觉应用。另一个不同之处在于，图像中像素的分辨率要远高于文本段落中的文字。有许多视觉任务，如语义分割，需要在像素级进行稠密预测，这对于高分辨率图像上的 Transformer 来说是很难解决的，因为它的自注意力的计算复杂度是图像尺寸的二次方。为了克服这些问题，作者提出了一个通用的 Transformer 主干，称为 Swin Transformer，它构造了层级式的特征图，并且具有与图像尺寸呈线性的计算复杂度。



如图所示，Swin Transformer 从小尺寸的 patch 开始，在更深的 Transformer 层中逐步合并相邻的 patch，从而构建层级式的表征。通过这些层级式的特征图，Swin Transformer 模型可以方便地利用先进的稠密预测技术，如特征金字塔网络 (FPN) 或 U-Net。而线性的计算复杂度是通过在划分图像的非重叠窗口内局部地计算自注意力来实现的。由于每个窗口中的 patch 的数量是固定的，因此复杂度与图像尺寸成线性关系。



Swin Transformer 的一个关键设计要素是在连续的自注意层之间移动窗口分区，如图所示。在后续的实验中表明，移动窗口方法比之前的滑动窗口方法具有更低的延迟，同时在建模能力方面是相似的。同时移动窗口方法也被证明对 all-MLP 架构是有益的。

3. 模型流程

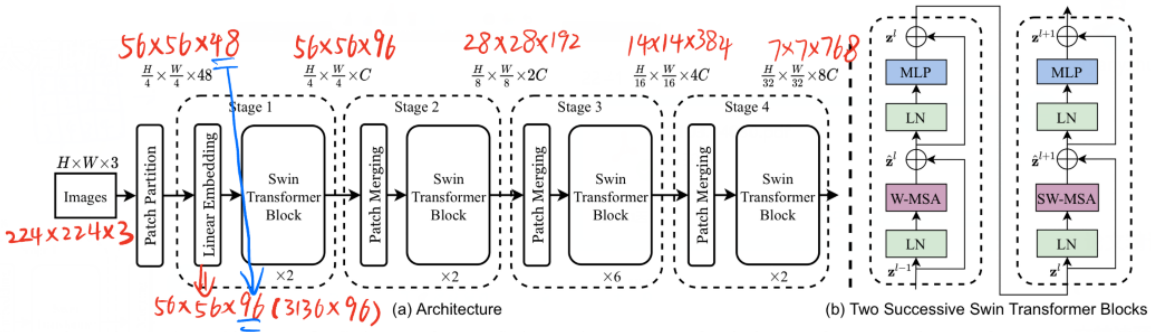
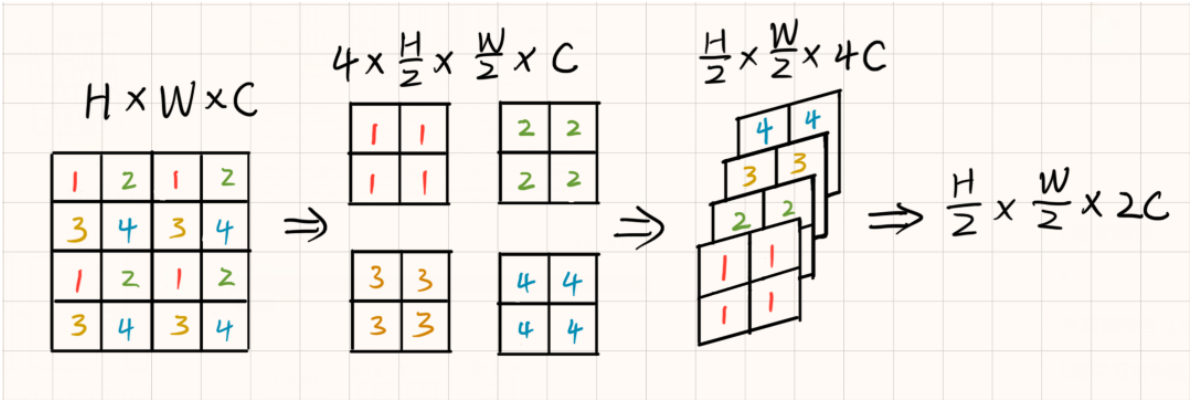


Figure 3. (a) The architecture of a Swin Transformer (Swin-T); (b) two successive Swin Transformer Blocks (notation presented with Eq. (3)). W-MSA and SW-MSA are multi-head self attention modules with regular and shifted windowing configurations, respectively.

假设现在有一张 ImageNet 标准尺寸的输入图片 $224 \times 224 \times 3$ ($H \times W \times 3$)。首先经过 Patch Partition 将其划分成多个 4×4 的 patch，所以经过 Patch Partition 之后图片的尺寸就是 $56 \times 56 \times 48$ ($\frac{H}{4} \times \frac{W}{4} \times 48$)，即经过 Patch Partition 后的图片大小为 56×56 个 patch，每个 patch 的向量维度为 $4 \times 4 \times 3 = 48$。接着经过 Linear Embedding 之后，这些 patch 的向量维度被投影到 C 维，在 Swin Transformer Tiny 中就是 96 维，那么此时的图像尺寸就是 $56 \times 56 \times 96$ ($\frac{H}{4} \times \frac{W}{4} \times C$)，然后被拉直之后就是 3136×96。则此时的序列长度就是 patch 的长度 3136。对于这个序列来说，对其做自注意力机制的计算复杂度非常高，因此 Swin Transformer 就提出了基于窗口的自注意力计算。Swin Transformer 的窗口设置为 7×7，则就只有 49 个 patch，所以处理的序列长度就只有 49，相较于 Vision Transformer 的序列长度 16×16 来说小了不少，这样就解决了计算复杂度高的复杂度。所以在 Swin Transformer Block 中就基于窗口进行自注意力计算。

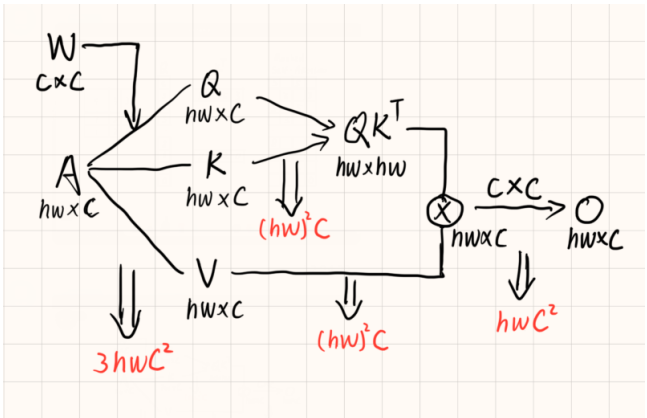
经过 Swin Transformer Block 之后，得到的尺寸维度还是和输入的尺寸维度一致。那么接下来如果还想要得到多尺寸的特征信息，那么就需要构建一个层级式的结构，也就是说需要像卷积神经网络一样，需要有一个池化的操作。那么在这篇论文中，作者就提出了 Patch Merging 的操作。随着网络的加深，通过 Patch Merging 可以减少 tokens 的数量 (即序列的长度)，产生层级化的表征。Patch Merging 将小的临近的 Patch 合并成大的 Patch，这样就可以起到下采样一个特征图的效果。这里就类似与卷积神经网络中增加感受野的操作，为了对其进行下采样 2 倍操作，所以在选择 Patch 的时候是每隔一个 Patch 选取一个。所以在图中相同序号的 Patch 就会被合并在一起。



因此经过采样操作，原来尺寸为 $56 \times 56 \times 96$ ($\frac{H}{4} \times \frac{W}{4} \times C$) 的张量就会变成 4 个尺寸为 $28 \times 28 \times 96$ ($4C$) 的张量。对这 4 个张量在 C 这个维度上进行拼接，就可以得到 $28 \times 28 \times 384$ 的张量。然后再使用 1×1 的卷积将其转化成 196 ($2C$) 个通道数。也就是说，Patch Merging 层将图片的空间大小减半，但是通道数乘 2。这样，在第 2 阶段中得到图片的尺寸大小就是 $28 \times 28 \times 192$ ($\frac{H}{8} \times \frac{W}{8} \times 2C$) 了。所以最后经过 4 个阶段之后，最后的图片尺寸为 $7 \times 7 \times 768$ ($\frac{H}{32} \times \frac{W}{32} \times 8C$)。

4. 模型原理

4.1 基于移动窗口的自注意力



为了高效的建模计算，作者提出了在非重叠的局部窗口中计算自注意力，用来取代全局自注意力。对于每个有 $h \times w$ 个 patch 的图片，其中给定输入的 A 矩阵，它的大小是 $hw \times C$ 的，首先将其变成 K，Q 和 V 三个向量，这部分经过一个 $C \times C$ 的权值矩阵的计算复杂度是 $3 \times hwC^2$ (因为对于两个大小分别为 $M \times N$ 和 $N \times P$ 的矩阵来说，它们的相乘计算复杂度为 $O(MNP)$)。然后将查询 Q 和键 K 进行相乘得到自注意力，其复杂度为 $(hw)^2C$，将自注意力与值 V 进行相乘的计算复杂度为 $(hw)^2C$，最后将计算的结果经过一个线性投影，其计算复杂度为 hwC^2。因此其计算复杂度的最终估值为

$$\Omega(\text{MSA}) = 3 \times hwC^2 + (hw)^2C + (hw)^2C + hwC^2 = 4hwC^2 + 2(hw)^2C$$

那么基于窗口的自注意力的计算复杂度就可以借用上面的计算结果。因为我们在每个不重叠的窗口计算的仍然是多头自注意力，只不过是它的计算高度和宽度不再是 $h \times w$ 了，而是窗口的大小 $M \times M$，因此在一个窗口内的计算复杂度为：

$$\Omega_{\text{one}}(\text{W-MSA}) = 4M^2C^2 + 2(M^2)^2C$$

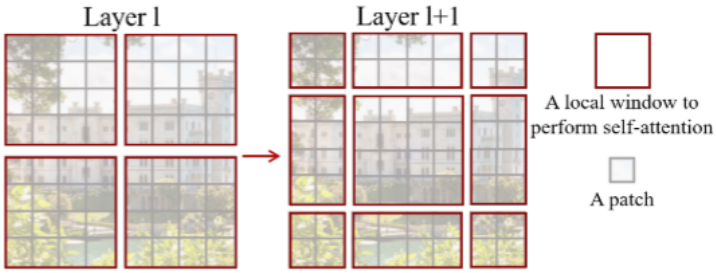
那么所有窗口的计算复杂度之和就是：

$$\Omega(\text{W-MSA}) = (\frac{h}{M} \times \frac{w}{M})\Omega_{\text{one}}(\text{W-MSA}) = 4hwC^2 + 2M^2hwC$$

所以，基于窗口的子注意力计算的复杂度是 $h \times w$ 的线性复杂度。而全局的自注意力计算是 $h \times w$ 的二次复杂度。全局自注意计算对于大型硬件系统来说通常是负担不起的，而基于窗口的自注意是具有良好的扩展性的。

4.2 在连续块中的移动窗口划分

基于窗口的自注意力模块 (W-MSA) 虽将计算复杂度从二次降为线性，但跨窗口之间交流与联系的匮乏将限制其建模表征能力。为引入跨窗口的联系且同时保持非重叠窗口的计算效率，作者提出一个移动窗口划分方法，该方法在连续 Swin Transformer Blocks 中的两种分区配置间交替进行。



如图所示，首个模块使用一个规则的窗口划分策略，从左上角像素开始，将 8×8 的特征图均匀划分成 2×2 个 4×4 大小的窗口 (此时窗口大小 $M = 4$)，然后下一个模块采用前一层的窗口移动配置，即将前面一层规则划分好的窗口循环移位 $(\lfloor \frac{M}{2} \rfloor, \lfloor \frac{M}{2} \rfloor)$ 个像素。

那么移动的窗口就桥接了前一层的窗口，提供了窗口之间的联系，因此可以增强建模能力。再配合上之前提出的 Patch Merging，合并到 Swin Transformer 最后几层的时候，每一个 patch 本身的感受野就很大了，就可以获取图片大部分的信息了，然后再加上移动窗口的操作，那么所谓的窗口内的局部注意力其实也就相当于是一个全局的自注意力操作了。这样不仅可以达到与全局自注意力操作相同的效果，还可以显著地降低自注意力的计算复杂度。

因此 Swin Transformer Blocks 通常是成对的。在进入 Swin Transformer Blocks 之后，输入会先做一次基于窗口的多头自注意力 (左边的窗口设置)，然后紧接着要再做一次基于移动窗口的多头自注意力 (右边的窗口设置)。移动窗口划分方法引入了上一层相邻无重叠窗口之间的联系，在图像分类、目标检测和语义分割方面是有效的。

这样，基于移动窗口策略，两个连续的 Swin Transformer Block 之间的计算过程如下：

$$
\begin{aligned}
\hat{\mathbf{z}}^{l} &= \text{W-MSA}\left(\text{LN}\left(\mathbf{z}^{l-1}\right)\right) + \mathbf{z}^{l-1}, \\
\mathbf{z}^{l} &= \mathbf{MLP}\left(\text{LN}\left(\hat{\mathbf{z}}^{l}\right)\right) + \hat{\mathbf{z}}^{l}, \\
\hat{\mathbf{z}}^{l+1} &= \text{SW-MSA}\left(\text{LN}\left(\mathbf{z}^{l}\right)\right) + \mathbf{z}^{l}, \\
\mathbf{z}^{l+1} &= \mathbf{MLP}\left(\mathbf{LN}\left(\hat{\mathbf{z}}^{l+1}\right)\right) + \hat{\mathbf{z}}^{l+1},
\end{aligned}
$$

4.3 移动窗口的高效批量计算

一个关于移动窗口划分的问题是，窗口数量会从 $\lceil \frac{h}{M} \rceil \times \lceil \frac{w}{M} \rceil$ 增长到 $(\lceil \frac{h}{M} \rceil + 1) \times (\lceil \frac{w}{M} \rceil + 1)$，同时有些窗口的尺寸将小于 $M \times M$。因此如果我们想要去做快速运算，就把这些 patch 压成一个 batch，直接去算自注意力，现在就做不到了。因为这些窗口的大小不一样了。一个朴素的解决方法是，将更小的窗口填充至 $M \times M$，且在基于移动窗口计算自注意力时使用掩码屏蔽掉填充值。这种方法在划分窗口数量很小时，例如 2×2 时，该朴素方法的计算量将从 2×2 增长到 3×3，这个计算增长量是相当大的。

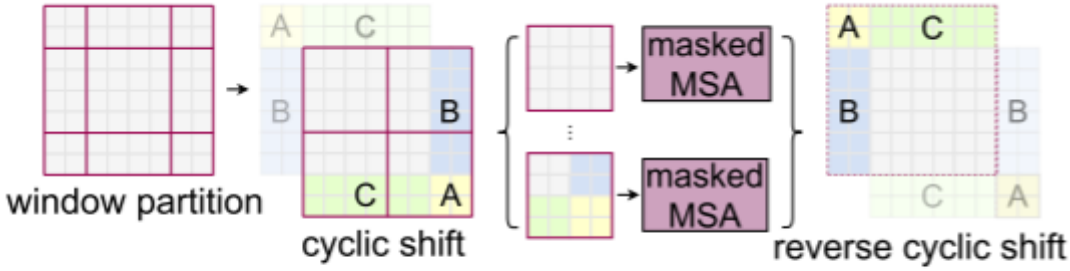


Figure 4. Illustration of an efficient batch computation approach for self-attention in shifted window partitioning.

因此作者提出了一个更有效的批量计算方法，即通过循环移动 patch 实现，这样就可以在移动窗口后仍然保持窗口的数量和窗口大小。如图所示，在这种移动窗口下，批量窗口可以由特征图中不相邻的子窗口组成。因为此时批量窗口中的元素可能是从不同窗口中得到的，所以它们之间按道理来说是不应该去计算这个自注意力的，因为它们之间没有太大的联系。此时再次使用掩码屏蔽机制，将子注意力计算限制在每个子窗口内。最后再通过逆循环移动方法将每个窗口的自注意力结果返回。

在论文中并没有详细介绍这部分 masked MSA 是如何实现的。通过阅读其它材料得知，masked MSA 机制先正常计算自注意力，然后再采用 mask 操作将不需要的自注意力置为 0，从而将自注意力计算限制再各个子窗口内。

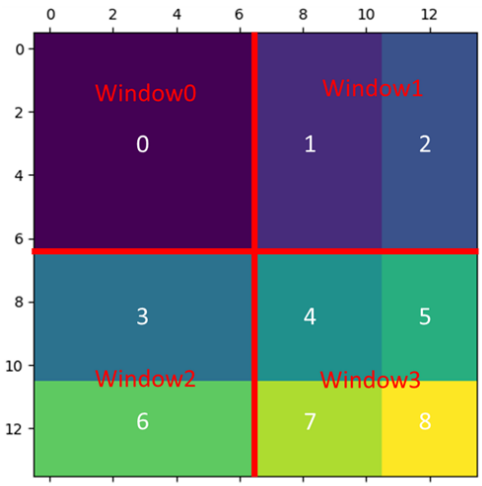
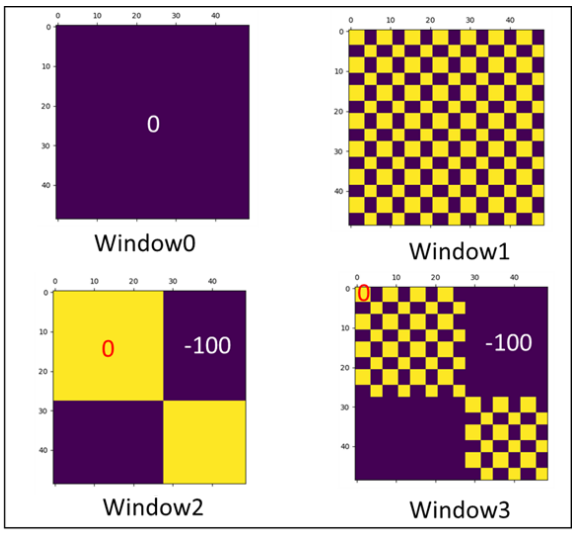
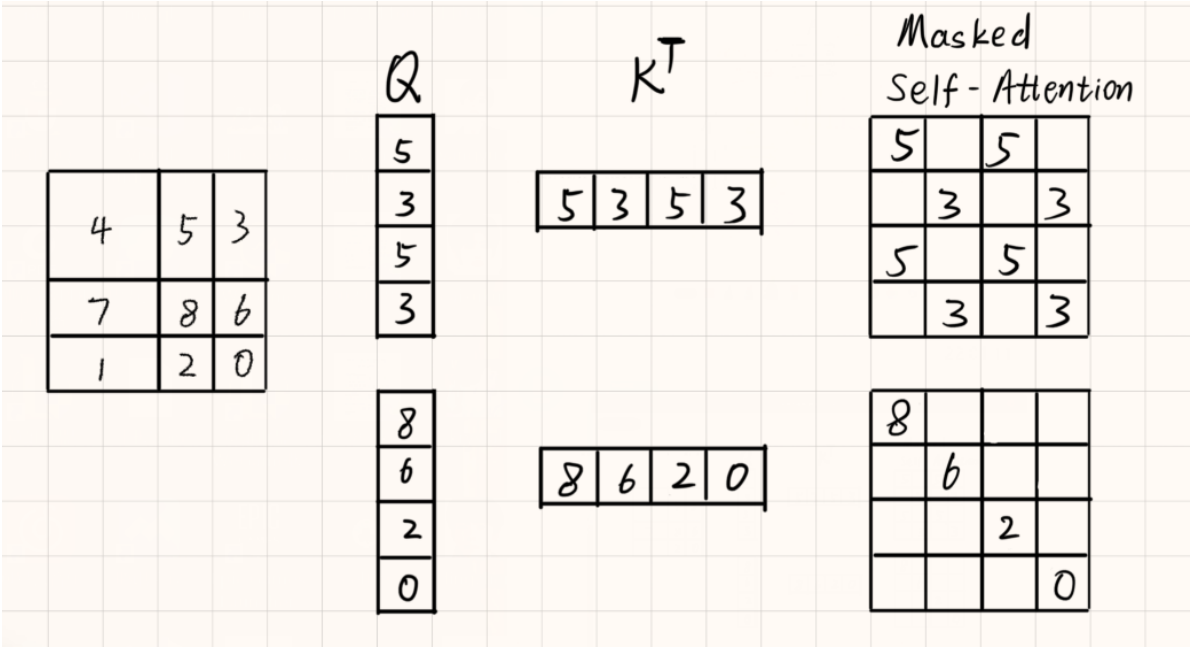


Image Mask
(14x14, window 7x7, shift 3)



Attn Mask

上述左边这个图就是已经经过循环位移后的图像特征了，里面有 4 个窗口，分成了 9 个区域。对于窗口 0 来说，它里面是不需要使用掩码的，而对于需要进行掩码位置的地方，将其掩码值设置成 -100。最后这个掩码矩阵会与图像进行相加，然后经过 $\mathrm{SoftMax}()$ 就可以将计算出的自注意力给置为 0。其详细的示例计算过程如下：对其做自注意力时，首先正常的计算出自注意力，然后根据是否需要计算自注意力来设置掩码 (图中为空的地方就是不需要的地方，设置掩码为 0)。



所以作者通过这种巧妙设计的掩码模板，从而实现了只需要一次前向过程，就能把所有需要的这个自注意力值给计算出来，而且只需要计算四个窗口，计算复杂度并没有因为移动窗口而增加。

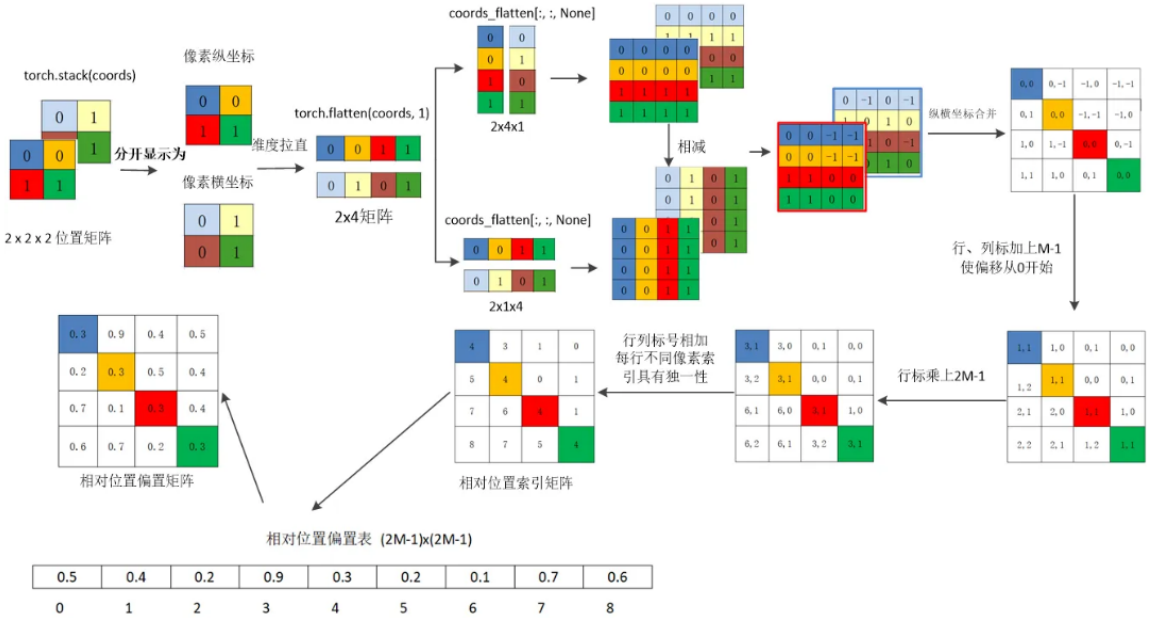
4.4 相对位置编码

在计算自注意力的时候，在计算相似度的过程中对每个头加入相对位置 $B \in \mathbb{R}^{(2M-1)\times(2M-1)}$：

$$\mathrm{Attetion}(Q,K,V)=\mathrm{SoftMax}(QK^T/\sqrt{d}+B)V$$

其中 $Q,K,V \in \mathbb{R}^{M^2\times d}$ 分别是 Query, Key 和 Value 矩阵。d 是 Query 和 Key 矩阵的维度，M^2 是窗口的大小 (即窗口内的 patchs 数量)。因为沿各轴的相对位置均处于 $[-M+1, M-1]$ 范围内，因此作者参数化了一个更小尺寸的偏置矩阵 $\hat{B} \in \mathbb{R}^{(2M-1)\times(2M-1)}$，且 B 中的值全都取自 $\hat{B}$。

由于这部分的工作并不是本文提出的，同时这一部分在论文中也没有详细介绍，所以通过查阅其它资料可以知道其大致的流程如下：



5. 实验结果

本文中作者对 ImageNet-1K 图像分类、COCO 对象检测和 ADE20K 语义分割进行了实验。下面就直接展示实验结果：

(a) Regular ImageNet-1K trained models					
method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
RegNetY-4G [48]	224^2	21M	4.0G	1156.7	80.0
RegNetY-8G [48]	224^2	39M	8.0G	591.6	81.7
RegNetY-16G [48]	224^2	84M	16.0G	334.7	82.9
EffNet-B3 [58]	300^2	12M	1.8G	732.1	81.6
EffNet-B4 [58]	380^2	19M	4.2G	349.4	82.9
EffNet-B5 [58]	456^2	30M	9.9G	169.1	83.6
EffNet-B6 [58]	528^2	43M	19.0G	96.9	84.0
EffNet-B7 [58]	600^2	66M	37.0G	55.1	84.3
ViT-B/16 [20]	384^2	86M	55.4G	85.9	77.9
ViT-L/16 [20]	384^2	307M	190.7G	27.3	76.5
DeiT-S [63]	224^2	22M	4.6G	940.4	79.8
DeiT-B [63]	224^2	86M	17.5G	292.3	81.8
DeiT-B [63]	384^2	86M	55.4G	85.9	83.1
Swin-T	224^2	29M	4.5G	755.2	81.3
Swin-S	224^2	50M	8.7G	436.9	83.0
Swin-B	224^2	88M	15.4G	278.1	83.5
Swin-B	384^2	88M	47.0G	84.7	84.5
(b) ImageNet-22K pre-trained models					
method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
R-101x3 [38]	384^2	388M	204.6G	-	84.4
R-152x4 [38]	480^2	937M	840.5G	-	85.4
ViT-B/16 [20]	384^2	86M	55.4G	85.9	84.0
ViT-L/16 [20]	384^2	307M	190.7G	27.3	85.2
Swin-B	224^2	88M	15.4G	278.1	85.2
Swin-B	384^2	88M	47.0G	84.7	86.4
Swin-L	384^2	197M	103.9G	42.1	87.3

Table 1. Comparison of different backbones on ImageNet-1K classification. Throughput is measured using the GitHub repository of [68] and a V100 GPU, following [63].

(a) Various frameworks								
Method	Backbone	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	#param.	FLOPs	FPS
Cascade Mask R-CNN	R-50	46.3	64.3	50.5	82M	739G	18.0
	Swin-T	**50.5**	**69.3**	**54.9**	86M	745G	15.3
ATSS	R-50	43.5	61.9	47.0	32M	205G	28.3
	Swin-T	**47.2**	**66.5**	**51.3**	36M	215G	22.3
RepPointsV2	R-50	46.5	64.6	50.3	42M	274G	13.6
	Swin-T	**50.0**	**68.5**	**54.2**	45M	283G	12.0
Sparse R-CNN	R-50	44.5	63.4	48.2	106M	166G	21.0
	Swin-T	**47.9**	**67.3**	**52.3**	110M	172G	18.4

(b) Various backbones w. Cascade Mask R-CNN									
	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}	param	FLOPs	FPS
DeiT-S†	48.0	67.2	51.7	41.4	64.2	44.3	80M	889G	10.4
R50	46.3	64.3	50.5	40.1	61.7	43.4	82M	739G	18.0
Swin-T	**50.5**	**69.3**	**54.9**	**43.7**	**66.6**	**47.1**	86M	745G	15.3
X101-32	48.1	66.5	52.4	41.6	63.9	45.2	101M	819G	12.8
Swin-S	**51.8**	**70.4**	**56.3**	**44.7**	**67.9**	**48.5**	107M	838G	12.0
X101-64	48.3	66.4	52.3	41.7	64.0	45.1	140M	972G	10.4
Swin-B	**51.9**	**70.9**	**56.5**	**45.0**	**68.4**	**48.7**	145M	982G	11.6

(c) System-level Comparison						
Method	mini-val		test-dev		#param.	FLOPs
	AP^{box}	AP^{mask}	AP^{box}	AP^{mask}		
RepPointsV2* [12]	-	-	52.1	-	-	-
GCNet* [7]	51.8	44.7	52.3	45.4	-	1041G
RelationNet++* [13]	-	-	52.7	-	-	-
SpineNet-190 [21]	52.6	-	52.8	-	164M	1885G
ResNeSt-200* [78]	52.5	-	53.3	47.1	-	-
EffcientDet-D7 [59]	54.4	-	55.1	-	77M	410G
DetectoRS* [46]	-	-	55.7	48.5	-	-
YOLOv4 P7* [4]	-	-	55.8	-	-	-
Copy-paste [26]	55.9	47.2	56.0	47.4	185M	1440G
X101-64 (HTC++)	52.3	46.0	-	-	155M	1033G
Swin-B (HTC++)	56.4	49.1	-	-	160M	1043G
Swin-L (HTC++)	57.1	49.5	57.7	50.2	284M	1470G
Swin-L (HTC++)*	**58.0**	**50.4**	**58.7**	**51.1**	284M	-

Table 2. Results on COCO object detection and instance segmentation. †denotes that additional decovolution layers are used to produce hierarchical feature maps. * indicates multi-scale testing.

ADE20K		val	test			
Method	Backbone	mIoU	score	#param.	FLOPs	FPS
DANet [23]	ResNet-101	45.2	-	69M	1119G	15.2
DLab.v3+ [11]	ResNet-101	44.1	-	63M	1021G	16.0
ACNet [24]	ResNet-101	45.9	38.5	-		
DNL [71]	ResNet-101	46.0	56.2	69M	1249G	14.8
OCRNet [73]	ResNet-101	45.3	56.0	56M	923G	19.3
UperNet [69]	ResNet-101	44.9	-	86M	1029G	20.1
OCRNet [73]	HRNet-w48	45.7	-	71M	664G	12.5
DLab.v3+ [11]	ResNeSt-101	46.9	55.1	66M	1051G	11.9
DLab.v3+ [11]	ResNeSt-200	48.4	-	88M	1381G	8.1
SETR [81]	T-Large‡	50.3	61.7	308M	-	-
UperNet	DeiT-S†	44.0	-	52M	1099G	16.2
UperNet	Swin-T	46.1	-	60M	945G	18.5
UperNet	Swin-S	49.3	-	81M	1038G	15.2
UperNet	Swin-B‡	51.6	-	121M	1841G	8.7
UperNet	Swin-L‡	**53.5**	**62.8**	234M	3230G	6.2

Table 3. Results of semantic segmentation on the ADE20K val and test set. † indicates additional deconvolution layers are used to produce hierarchical feature maps. ‡ indicates that the model is pre-trained on ImageNet-22K.

由于对其他模型以及对象检测和语义分割任务的不了解，因此就不对实验结果进行分析了。但从实验结果来看，Swin Transformer 的性能较以往的模型都提升不少。

6. 论文总结

本文介绍了 Swin Transformer，这是一个新的视觉 Transformer，它可以生成一个分层次的特征表示，并且具有相对于输入图像大小的线性计算复杂度。Swin Transformer 在 COCO 对象检测和 ADE20K 语义分割方面都取得了最先进的性能，大大超过了以往最好的方式。作者希望 Swin Transformer 在各种视觉问题上的强大表现可以推进视觉和语言处理的统一建模。作为 Swin Transformer 的关键元素，基于移动窗口的自注意力在视觉任务上展示出了有效且高效的结果，作者也期待着移动窗口在自然语言处理中的应用。

7. 阅读体会

在本论文中，作者希望拓展 Transformer 的适用性，使其可以作为作为计算机视觉的通用主干网络，因此提出了 Swin Transformer 这个架构。它构造了层级式的特征图，并且具有与图像尺寸呈线性的计算复杂度，因此相较于 Vision Transformer，Swin Transformer 能够在对象检测和物体分割任务上作为主干网络。Swin Transformer 借鉴了很多卷积神经网络的设计理念以及先验知识，是 Transformer 在局部自注意力策略上一次很好的尝试。

在卷积神经网络中，卷积操作由于其权值共享、局部性、平移不变性以及滑动窗口等特性，十分适合对图像的各种特征进行建模。但是由于卷积神经网络中的局部性特性，使得其对于远距离依赖的建模成本较高，因此通常需要通过堆叠更深层数的卷积层或者通过空洞卷积的方式来增大感受野。

那么 Swin Transformer 就借鉴了卷积神经网络中这些理念：

- 引入卷积神经网络的层次化构建方式，构建层级式 (Hierarchical) 的 Swin Transformer。
- 引入卷积神经网络的局部性思想，在不重叠的窗口内进行局部自注意力计算 (Local Windows Self-Attention)。
- 引入卷积神经网络的池化操作，提出了 Patch Merging 操作，获取多尺寸的特征。
- 引入卷积神经网络的滑动窗口思想，提出使用移动窗口 (Shifted Windows) 来提供不同窗口之间的联系，从而实现全局的建模能力。
- 引入卷积神经网络的平移不变性特性，移动窗口时采用循环移动的方式，并且采用了模板掩码。

基于这些理念，Swin Transformer 实现后在各类视觉任务上的表现效果都十分好，证实了其有作为通用的视觉骨干网络的潜力。那么相较于 Vision Transformer，Swin Transformer 基于许多卷积神经网络中的先验知识搭建了这样的网络架构，而 Vision Transformer 是直接采用了 Transformer 的原结构，其相比于卷积神经网络而言要少非常多的图像先验知识。在 Vision Transformer 中，所有关于图像块之间的距离信息、场景信息等，以及视觉中的先验知识，都需要从头开始学习。

但是，Swin Transformer 在多模态方面的任务可能表现的并不会很好。Swin Transformer 更多的还是借鉴了视觉任务中的先验知识，毕竟 Swin Transformer 的目标是作为计算机视觉的通用主干网络，所以在语言处理方面，其表现效果可能并不会如 Transformer 那么好。因此作者在论文结束时也期待着移动窗口在自然语言处理中的应用。

除此之外，我还对这种移动窗口的方式产生疑问。在论文中，只提到了一种 $\left(\lfloor \frac{M}{2} \rfloor, \lfloor \frac{M}{2} \rfloor\right)$ 移动的方式，那么是否可以使用多种移动窗口的方式，这样做是否可以更好的在窗口之间建立联系，从而更快捷地学习到远程的依赖关系，从而提高模型的全局建模能力。

另外，Swin Transformer 中局部自注意力和移动窗口等理念实际上还是借鉴于卷积神经网络的核心思想，所以 Swin Transformer 实际上和卷积神经网络的差别不大。因此我觉得需要考虑是否能够探索出一种新的图像处理技巧更好地适用于 Transformer 架构，从而不必像 Swin Transformer 一样提出层级式的结构，而是继续利用原有的 Transformer 架构，那么这样或许会更有利于多模态模型的发展。

总体来说，这篇论文是一个十分具有影响力的论文，里面具有十分多巧妙的设计让人眼前一亮，给我留下了深刻的印象。读完这篇论文，我感觉还是有非常多收获的。