摘要

本文介绍了一种称为**Swin Transformer**的新视觉Transformer，它可以作为 CV 的**通用主干。**将 Transformer 从语言适应到视觉方面的挑战来自**两个域之间的差异，例如视觉实体的规模以及相比于文本单词的高分辨率图像像素的巨大差异。**为解决这些差异，**Swin Transformer**提出了一种**层次化 (hierarchical) Transformer**，其表示是用 **移位窗口 (Shifted Windows)** 计算的。移位窗口方案通过将***自注意力计算限制在不重叠的局部窗口的同时，还允许跨窗口连接来提高效率。***这种分层架构具有在各种尺度上建模的灵活性，并且 相对于图像大小具有线性计算复杂度。Swin Transformer 的这些特性使其与广泛的视觉任务兼容，包括图像分类（ImageNet-1K 的 87.3 top-1 Acc）和 密集预测任务，例如 目标检测（COCO test dev 的 58.7 box AP 和 51.1 mask AP）和语义分割（ADE20K val 的 53.5 mIoU）。它的性能在 COCO 上以 +2.7 box AP 和 +2.6 mask AP 以及在 ADE20K 上 +3.2 mIoU 的大幅度超越了 SOTA 技术，证明了基于 Transformer 的模型作为视觉主干的潜力。分层设计和移位窗口方法也证明了其对全 MLP 架构是有益的。

1.介绍

CV 建模主要依赖于 CNN。AlexNet在图像分类挑战中的出色表现开启了CNN架构的进化之路，其规模不断扩大，连接更加广泛，卷积形式更为复杂。CNN作为各种视觉任务的核心网络，架构的进步提升了性能，推动了整个领域的发展。然而，NLP领域的网络架构发展则另辟蹊径，**Transformer**成为主流。**Transformer**专为序列建模和转换任务而设计，其注意力机制能够有效处理数据中的长程依赖关系，因而声名远扬。它在语言领域的巨大成功，引发了研究人员对其在计算机视觉领域应用的研究，近期在某些任务中已展现出良好的效果，尤其在***图像分类和联合视觉-语言建模***方面。

将**Transformer**迁移至CV领域是面临显著挑战的，这因为语言领域的高性能与视觉领域具有显著差异，这种差异主要有**两方面**：
1.其中一种差异体现在尺度上。与语言**Transformer**中作为处理基本单位的**word token**不同，视觉元素的尺度可能存在较大差异，这在目标检测等任务中是一个值得关注的问题。在现有的基于**Transformer**的模型中，token的尺度是**固定**的，这种特性并**不适用于这些视觉应用。**
2.另一个差异是，**图像中的像素分辨率远高于文本段落中的文字。**还有许多视觉任务，例如语义分割，需要在像素级别进行密集预测，这对于高分辨率图像上的**Transformer**来说处理难度较大，因为***其自注意力的计算复杂度与图像大小的二次方相关。***

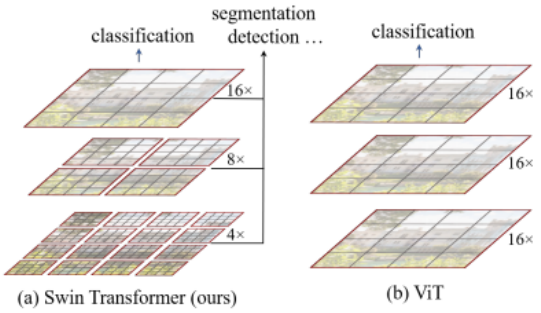

Figure 1. (a) The proposed Swin Transformer builds hierarchical feature maps by merging image patches (shown in gray) in deeper layers and has linear computation complexity to input image size due to computation of self-attention only within each local window (shown in red). It can thus serve as a general-purpose backbone for both image classification and dense recognition tasks. (b) In contrast, previous vision Transformers [20] produce feature maps of a single low resolution and have quadratic computation complexity to input image size due to computation of self-attention globally.

图1

为了解决这些问题，**Swin Transformer**构建了层次化特征图，***其计算复杂度与图像大小呈线性关系。***如图1所示，**Swin Transformer 从较小尺寸的 patch（灰色轮廓）开始，在更深入的Transformer层中逐渐合并相邻的 patch，从而构建出一个层次化表示。**通过这些层次化特征图，**Swin Transformer**模型可以轻松地应用先进技术进行密集预测，例如特征金字塔网络（FPN）或 U-Net。***线性计算复杂度是通过在图像分区的非重叠窗口内局部计算自注意力来实现的（红色轮廓）（而不是在整张图像的所有 patch 上进行）。每个窗口中的 patch 数量是固定的，因此复杂度与图像大小成线性关系。***这些优势使**Swin Transformer**成为各种视觉任务的通用主干，与之前基于 **Transformer**的架构形成对比，后者***生成单一分辨率的特征图且具有二次复杂度。***

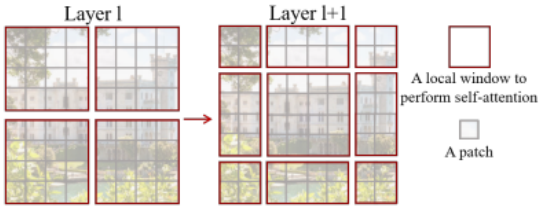

Figure 2. An illustration of the *shifted window* approach for computing self-attention in the proposed Swin Transformer architecture. In layer l (left), a regular window partitioning scheme is adopted, and self-attention is computed within each window. In the next layer $l+1$ (right), the window partitioning is shifted, resulting in new windows. The self-attention computation in the new windows crosses the boundaries of the previous windows in layer l, providing connections among them.

图2

Swin Transformer的一个关键设计要素是它在***连续自注意力层之间的窗口分区的移位（shift)***，如图 2 所示。**移位窗口连接了前一层的窗口，提供了两者之间的联系，显著增强了建模能力。**这种策略对于实际应用中的延迟也很有效：一个局部窗口内的所有查询 patch 共享相同的key集合，这有利于硬件中的内存访问。相比之下，早期基于滑动窗口的自注意力方法，由于不同的查询像素具有不同的key集合，在通用硬件上受到低延迟的影响。我们的实验表明，所提出的移位窗口方法的延迟比滑动窗口方法要低得多，而建模能力相似（见表 5/6）.

Swin Transformer在图像分类、目标检测和语义分割等识别任务上表现出色。它在三个任务上的延迟相似的情况下，明显优于**ViT/DeiT 和 ResNe(X)t**等优秀CV模型,它将促进视觉和文本信号的联合建模，并可以更深入地共享来自两个领域的建模知识。

2.整体架构

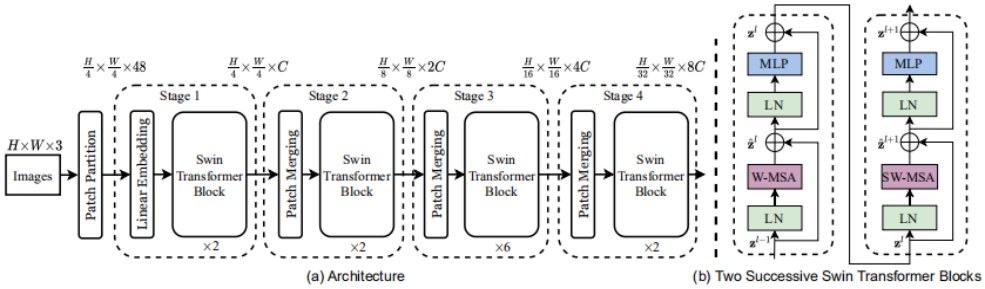

Figure 3. (a) The architecture of a Swin Transformer (Swin-T); (b) two successive Swin Transformer Blocks (notation presented with Eq. (3)). W-MSA and SW-MSA are multi-head self attention modules with regular and shifted windowing configurations, respectively.

图3

上图展示了**Swin Transformer**的基本架构（tiny版**Swin Transformer**）

- **Patch Partition模块**
  1.首先通过**Patch Partition模块**将输入的维度为$H \times W \times 3$的GRB图像拆分为**非重叠且尺寸相同的patch**,拆分后整体维度变为$N \times (p^2 \times 3)$，其中：$N=\frac{HW}{p^2}$,代表拆分出来多少个Patch(**Transformer**的有效输入序列长度)；$p^2 \times 3$代表每个Patch的维度，每个patch会被延展后作为一个patch token输入**Swin Transformer**中
  2.具体到上图，p被选为4，故将初始维度为$H \times W \times 3$输入**Patch Partition模块**后，该图像会被拆分为$N=\frac{HW}{4^2}$个形状为$4^2 \times 3 = 48$的Patch,将每个Patch展平为48维的token向量（类似于ViT的Flattened Patches），整体是一个展平且维度为$N \times 48$的$2D$ patch张量（矩阵）
- **线性嵌入模块（Linear Embedding）**是一个针对于各Patch token的全连接层FC，将经由**Patch Partition模块**所得维度为$N \times 48$的张量投影到任意维度C,得到维度为$N \times C$的(Linear Embedding)
- **Swin Transformer blocks**:Linear Embedding被输入若干具有改进自注意力的**Swin Transformer blocks**。首个**Swin Transformer block**保持输入输出tokens恒为N不变，且与**线性嵌入模块（Linear Embedding）**共同被指定为**Stage 1**.
- **层次化表示 (Hierarchical Representation)**：为了产生一个层次化表示，随着网络的加深，Patch tokens数量会被逐渐减小，这一操作通过各Stage的**Patch合并模块(patch meraging)**完成，它在每个Stage前下采样缩小分辨率并减半通道数。
  1.首个Patch合并模块拼接了每组2×2相邻Patch，所以Patch tokens数N变为原来的$\frac{1}{4}$,即为$N=\frac{H}{8} \times \frac{W}{8}$.
  2.进而，Patch token的维度扩大4倍，即$4C$。
  3.然后，对$4C$维度的拼接Patch tokens使用了一个线性层，将输出维度将为$2C$
  4.最后，使用**Swin Transformer blocks**进行特征转换，保持其分辨率（token数N）为$\frac{H}{8} \times \frac{W}{8}$不变。
  5.首个**Patch合并模块(patch meraging)**与后面的**Swin Transformer blocks**构成**Stage 2**.
  6.重复2次**Stage 2**,分别指定为**Stage 3**，**Stage 4**，输出分辨率（token数N）分别为$\frac{H}{16} \times \frac{W}{16}$，$\frac{H}{32} \times \frac{W}{32}$.
  7.每个**Stage i**都会改变Patch tokens张量的维度，从而形成一种层次化的表示。

下图分别为：
图4：**Patch合并模块(patch meraging)下采样过程**
图5：**Swin Transformer数据流维度变化**

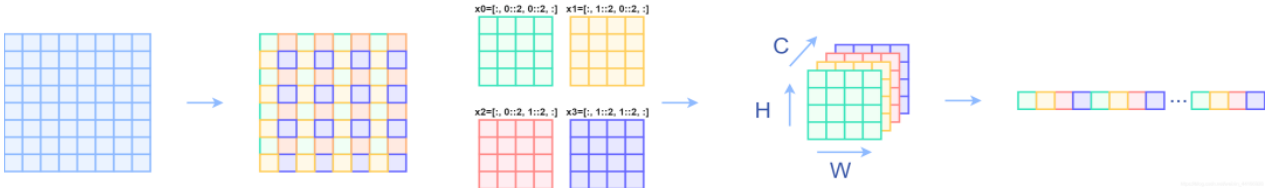

图4

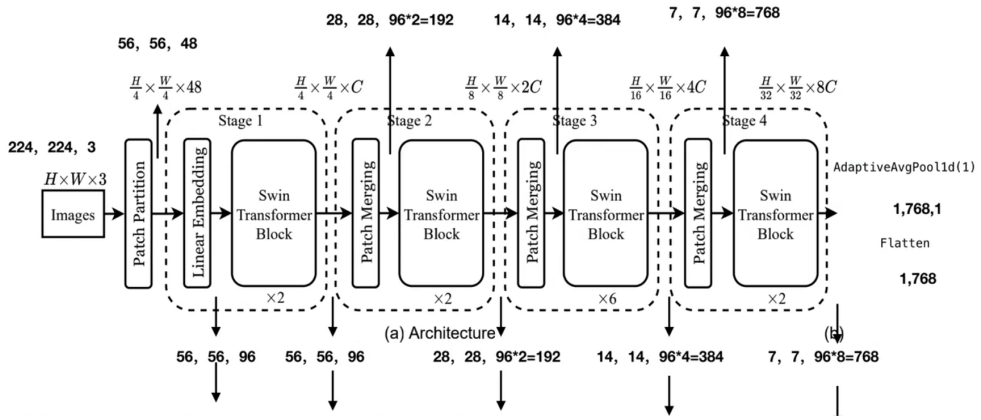

图5

3.Swin Transformer blocks

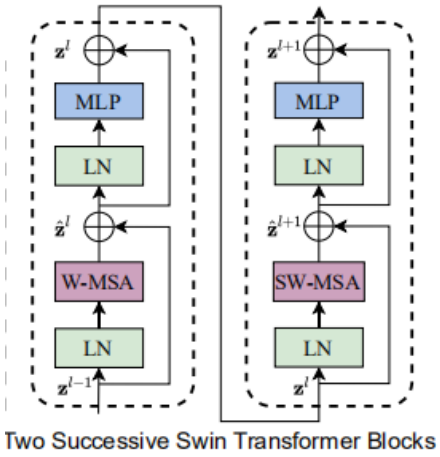

图6

- **Swin Transformer block**相比于**Vision Transformer block(ViT)**,将标准多头自注意力模块（MSA，Multi-Head Self Attention）替换为**基于规则/移位窗口的多头自注意力模块 (W-MSA / SW-MSA)**且其他部分保持不变。
- 如图6所示：一个**Swin Transformer block**由前一**基于规则窗口的多头自注意力模块W-MSA**和后一个**基于移位窗口的多头自注意力模块SW-MSA**构成，前后各接一个**LayerNorm(LN)层**，且在后接的LN层后还要接一个应用有**Gelu非线性激活函数**(似乎是NLP领域的当前最佳,尤其在Transformer模型中表现最好)的**两层MLP**（Multilayer Perceptron,多层感知器，包含一系列全连接层和非线性激活函数来对特征进行变换，最终得到一个经过非线性映射的新特征表示)
- 残差链接分别在W—MSA和SW-MSA模块中被分别运用两次

4.基于移位窗口的自注意力机制

4.1 传统CV版本Transformer复杂度计算

- 基于标准的**Transformer**架构变化而形成的对图像分类的适应版本，如**Vision Transformer**架构，都执行**全局自注意力**,即计算了每个token与其他所有tokens之间的注意力得分(**Attention map**),其具有**相对于token数N的二次计算复杂度**$O(N^2D)$,D为token向量长度/嵌入维度，即上文中$p^2 \times C$.

Multi-Head Attention

$$\begin{aligned}\text{MultiHead}(Q,K,V) &= \text{Concat}(\text{head}_1,\ldots,\text{head}_\text{h})W^O \\ \text{where}\quad \text{head}_\text{i} &= A(QW_i^Q, KW_i^K, VW_i^V)\end{aligned}$$

对于multi-head attention，假设有 h 个head，这里 h 是一个常数，对于每个head，首先需要把三个矩阵分别映射到 d_q, d_k, d_v 维度。这里考虑一种简化情况： $d_q = d_k = d_v = \frac{d}{h}$ 。(对于dot-attention计算方式， d_k 与 d_v 可以不同)。

- 输入线性映射的复杂度： $n \times d$ 与 $d \times \frac{d}{h}$ 运算，忽略常系数，复杂度为 $\mathcal{O}(nd^2)$ 。
- Attention操作复杂度：主要在相似度计算及加权和的开销上， $n \times \frac{d}{h}$ 与 $\frac{d}{h} \times n$ 运算，复杂度为 $\mathcal{O}(n^2 d)$
- 输出线性映射的复杂度：concat操作拼起来形成 $n \times d$ 的矩阵，然后经过输出线性映射，保证输入输出相同，所以是 $n \times d$ 与 $d \times d$ 计算，复杂度为 $\mathcal{O}(nd^2)$

故最后的复杂度为： $\mathcal{O}(n^2 d + nd^2)$

图7

- 基于上述计算，当$N >> D$时，$O(MSA) = O(N^2D)$,过于复杂
- 结论：**传统变体不适用与许多需要大量tokens的密集预测/高分辨率图像表示等高计算量视觉问题**

4.2 swin Transformer复杂度计算

- 为高效建模，**Swin Transformer block**中以不重叠的方法均匀划分图像得到各个窗口，在**非重叠的局部窗口中计算自注意力**，取代全局注意力
- 已知$D = 2C$,我们设**每个非重叠局部窗口都包含$N = M \times M$个Patch tokens**,则**基于具有$N = H \times W$个patch tokens的图像窗口的MSA模块**和**基于非重叠的规则/移动局部窗口的W-MSA/SW-MSA模块**的计算复杂度分别如下图所示：

$$\begin{aligned}\Omega(\text{MSA}) &= 4hwC^2 + 2(hw)^2C, \\ \Omega(\text{W-MSA}) &= 4hwC^2 + 2M^2hwC,\end{aligned}$$

图8

- 其中
  1.MSA关于Patch tokens数$N = h \times w$具有**二次复杂度**$O(MSA) = O(N^2D) = O((hw)^2D)$（**共有$hw$个patch tokens,基于多头自注意力机制，每个token在全局内计算hw次,每次要计算D个元素相乘**）
  2.W-MSA/SW-MSA，当M固定的时候（假设为一个较小的数字），关于Patch tokens数$N = H \times W$具有**线性复杂度**$O(W\text{—}MSA/SW\text{—}MSA) = O((M^2)hwC)$（**共有$hw$个patch tokens,基于非重叠窗口的自注意力机制，每个token在各自的局部窗口内计算M^2次,每次要计算D个元素相乘**）
- 巨大的hw对于全局自注意力的计算是难以承受的，而**基于非重叠的规则/移动局部窗口的自注意力机制（W-MSA/SA-MSA）**具有良好的扩展性

4.3 在连续块中的移动窗口划分

- ***基于非重叠的规则局部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，**Swin Transformer**提出一个**移位窗口划分方法**，该方法在连续**Swin Transformer block**中的两种划分/分区配置间交替。

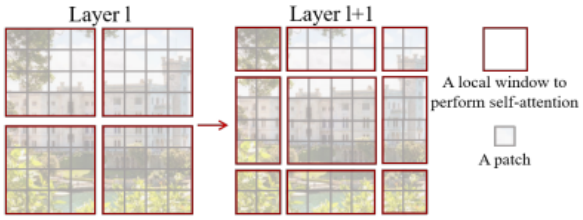

Figure 2. An illustration of the *shifted window* approach for computing self-attention in the proposed Swin Transformer architecture. In layer l (left), a regular window partitioning scheme is adopted, and self-attention is computed within each window. In the next layer $l+1$ (right), the window partitioning is shifted, resulting in new windows. The self-attention computation in the new windows crosses the boundaries of the previous windows in layer l, providing connections among them.

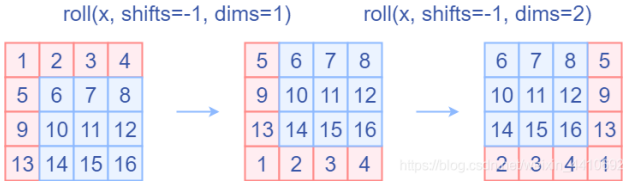

图9

- 如图9，首个模块使用**基于非重叠的规则局部窗口的自注意力机制（W-MSA）**,从左上角像素开始，将8×8的图像均匀划分为2×2个大小为4×4的规则窗口（此时$N = 16 = M \times M = 4 \times 4$)
- 然后，第二个模块使用**基于非重叠的移动局部窗口的自注意力机制（SW-MSA）**,令前一层的规则窗口向左上**循环移位（$\frac{M}{2}, \frac{M}{2}$)**个像素，如图9所示

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

图10

- 通过采用移位窗口划分方法，如图6的**两个连续 Swin Transformer Blocks**的计算可表示为图10,其中$\hat{z}^l$和z^l分别表示了第l个**Swin Transformer block**的**W-MSA/SA-MSA模块**输出特征和**MLP模块**输出特征

4.4 SW-MSA中的masked MSA机制

- 一个关于移动窗口的问题是：用上述策略进行移动后不会产生更多的窗口编号，但**有些属于同一编号窗口的尺寸将小于**$M \times M$。一个解决办法是，**将更小的窗口填充至**$M \times M$，**且在计算注意力时masked掉填充值。**当规则划分的窗口数很小的时候，如前文举例的2×2，由该朴素方法所带来的计算量增长是相当可观的（$2 \times 2 = 4 \rightarrow 3 \times 3 = 9$,扩大2.25倍）
- 另一个关于移动窗口的问题是：**在移动窗口后，批窗口所在位置不变，但其内部却可能由特征图中不相邻的子窗口组成**，这可能会导致不同窗口的自注意力计算混合到一起,而我们在计算Attention Map时，希望**仅留下具有相同index的Query和Key的计算结果**，而**忽略不同index 的Query和Key的计算结果**。因此，需要采用**masked MSA 机制将自注意力计算限制在各子窗口内，最后通过逆循环移位方法将每个窗口的自注意力结果返回**。通过合理设置Mask，可使**Shifted Window Attention(SW-MSA)** 在与**Basic Window Attention(W-MSA)** 窗口个数相同的情况下达到等价的计算结果，如图11所示

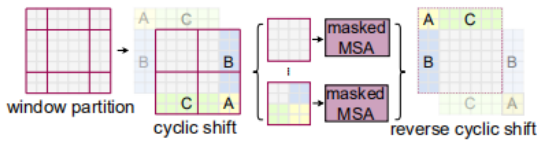

Figure 4. Illustration of an efficient batch computation approach for self-attention in shifted window partitioning.

图11

下面是一个简单的示例：

首先，对Shift Window后的每个窗口都赋予index，并执行roll操作(window_size=2, shift_size=1),如下所示：

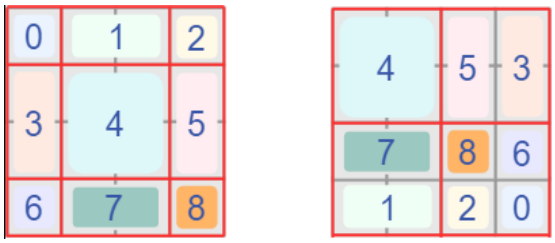

图12

假设窗口大小为2×2,则**图中应共有四个规则窗口**，若在原本的规则窗口中直接进行自注意力计算会使得 5号/3号 子窗口的自注意力计算混在一起，类似的混算还包括 7号/1号 子窗口和 8号/6号/2号/0号 子窗口的纵向或横向等，这些**移动后产生的窗口大多在原规则窗口中不相邻。**所以需采用**masked MSA 机制**：**先正常计算自注意力，再进行mask操作将不需要的注意力图置0，从而将自注意力计算限制在各子窗口内。**如下图所示：

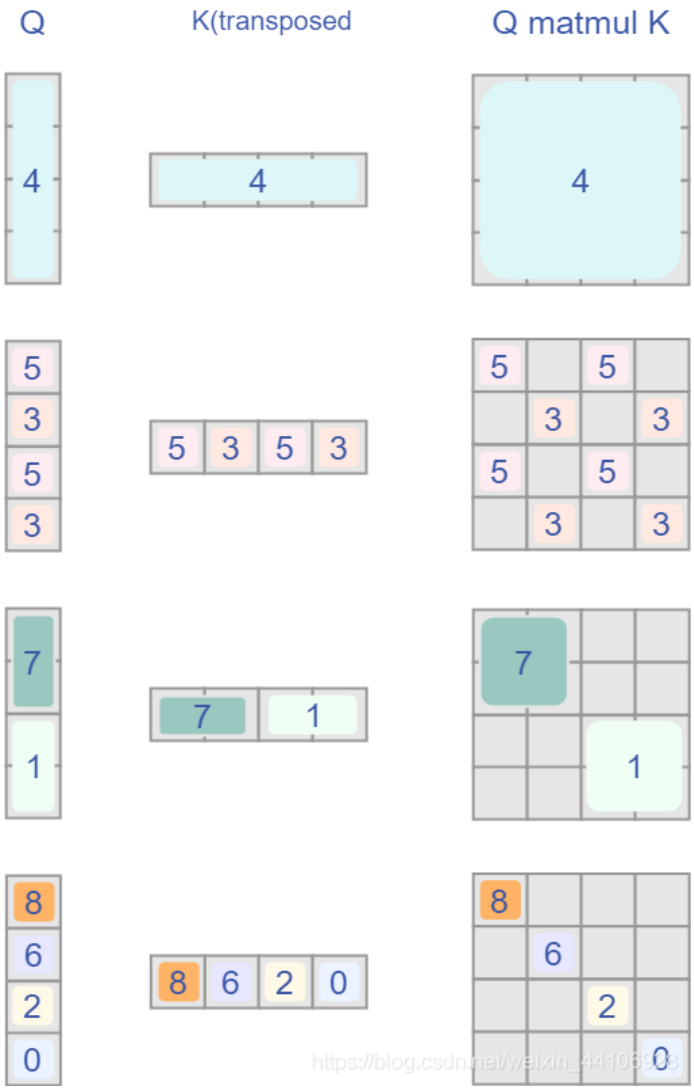

图13

Value和Query的shape一致(4×1),以上方法计算的shape=(4×4)的Q、K乃至Attention Map与Value相乘时，依然能够得到正确位置的运算结果，即(4×4)·(4×1)=(4×1)。

而若要在原始的四个窗口下得到正确计算结果，则必须给**Attention Map**加入一个**Mask**(如上图灰色 patch)

5.相对位置编码

- 在计算自注意力时，我们在计算相似度的过程中对每个head加入**相对位置编码**，$B \in \mathbb{R}^{\mathbb{M}^2 \times \mathbb{M}^2}$,如下所示：

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

图14

- 其中,$Q, K, V \in \mathbb{R}^{\mathbb{M}^2 \times d}$分别为**Query,Key和Value矩阵,$d$为Query/key维度，$M^2$为（局部）窗口内的Patch数。**$Q, K$相乘得到**Attention map**,它的shape=($numWindows \times B, numHeads, WindowSize \times WindowSize, WindowSize \times WindowSize$)。对于**Attention**

map,**以不同的像素点作为原点，则各像素点位置/坐标随之变化**，这也是相对位置编码的设计基础

- 因为沿各轴的相对位置均处于[-M+1,M-1]范围内，我们参数化一个更小尺寸的偏置矩阵$\hat{B} \in \mathbb{R}^{2M-1 \times 2M-1}$,且$B$中的值均取自$\hat{B}$
- 由前文可以知道，每个非重叠规则局部窗口包含$N = M \times M$个patch tokens,假设现在某个局部窗口的$WindowSize = M = 2$,那么分别以**左上角像素点和右上角像素点为原点**的**初始相对位置编码**如下图所示：

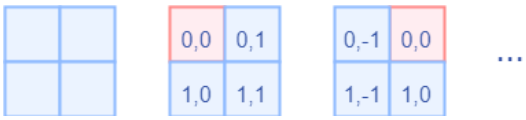

图15

- 以图15所展示该窗口中**不同像素点所对应的初始化相对位置编码规则**为基础，我们可以将该窗口中每个像素点都进行初始化，然后展开成为$1D$向量，堆叠到一起后进行一些操作得到最终的**相对位置索引矩阵**，并注册一个不参与网络学习的**relative_position_index**,其作用是**根据最终得到的相对位置索引找到对应的可学习的相对位置编码，并根据相对位置索引矩阵的shape构成相对位置编码矩阵B，然后在后续的注意力机制中与Attention map逐元素相加**
- 需要注意的是，每个局部窗口所得出的**Attention map**与此方法得到的**相对位置索引矩阵**的**shape**是相同的，这也是二者可以逐元素相加的前提

下图是得到**相对位置索引矩阵**方法的示例图：

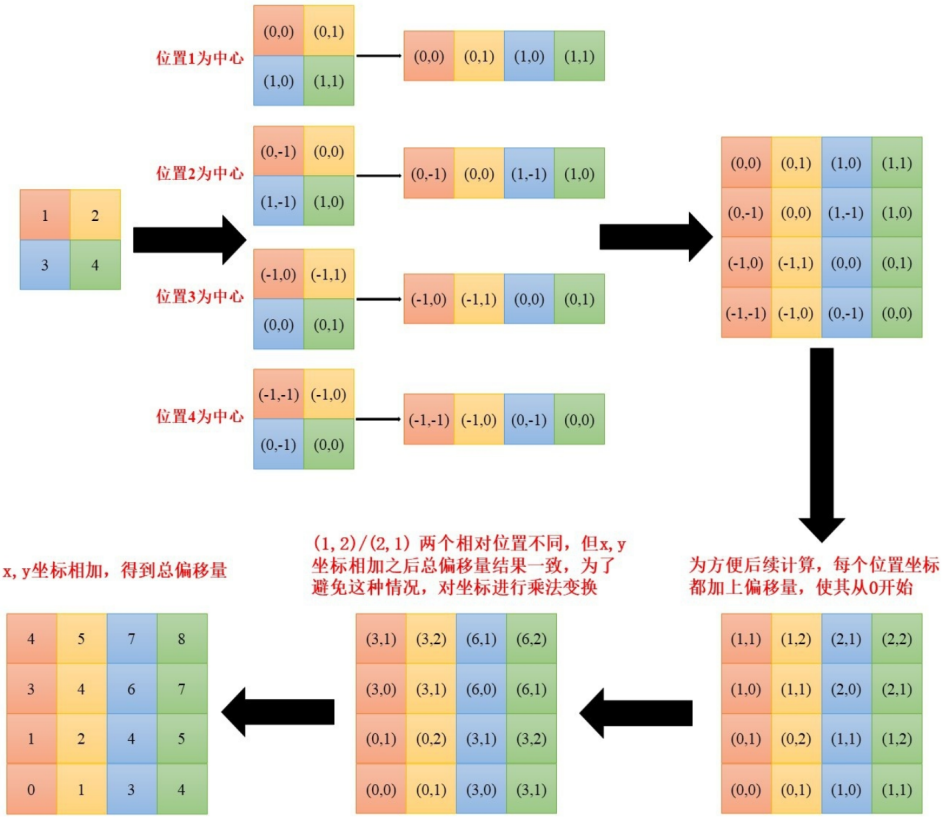

图16

- 经过实验，使用相对位置编码的效果**显著优于不使用位置编码和使用绝对位置编码**，可以显著改善模型性能。有趣的是，进一步向输入添加**绝对位置编码**却会略微降低性能