

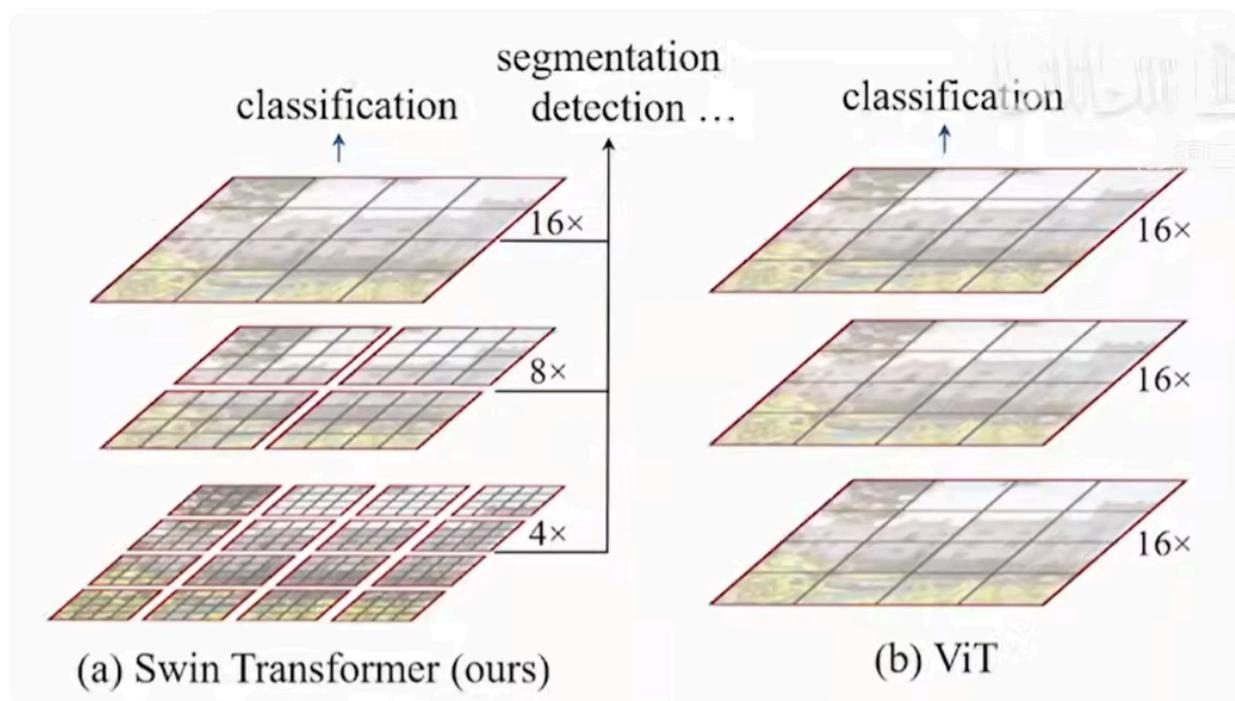
# Swin Transformer

## 1.motivation

VIT只做了分类，下游任务检测和分割留给后人

swin transformer证明了transformer确实能在视觉领域通吃

## 2.Introduction

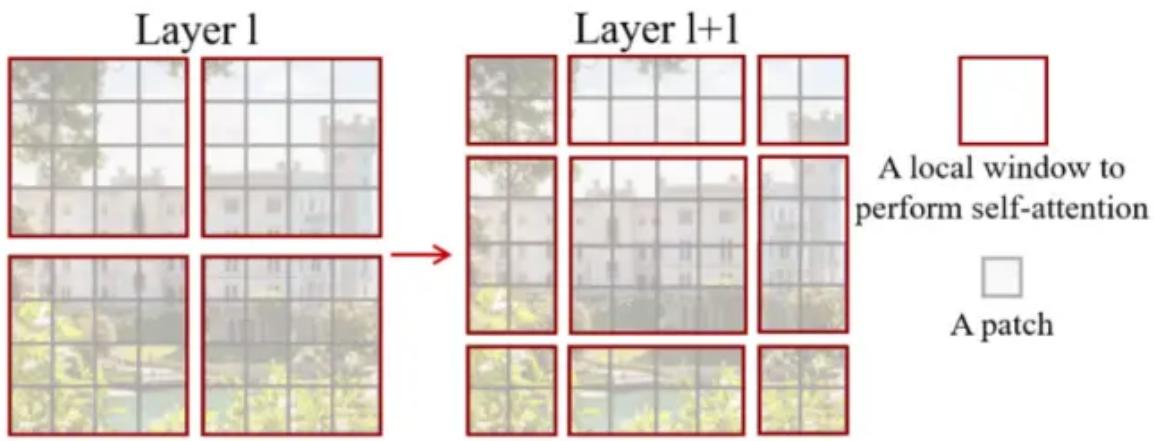


**VIT :**

- 全局自注意力，全局建模能力强，但多尺寸特征把握弱
- 计算复杂度平方，高分辨率下序列长，计算复杂度难以承受

**Swin Transformer**

- 小窗口内自注意力
  - 窗口大小固定->自注意力固定->图片面积扩大 $x$ 倍，窗口数量扩大 $x$ 倍：线性关系
  - 局部相似性：利用Locality的inductive bias
  - Patch Merging：CNN通过Pooling 增大感受野 提取多尺寸特征，Swin transformer 把相邻小patch合并大patch 感受野增大 抓住多尺寸特征
- 移动窗口Shift windows操作

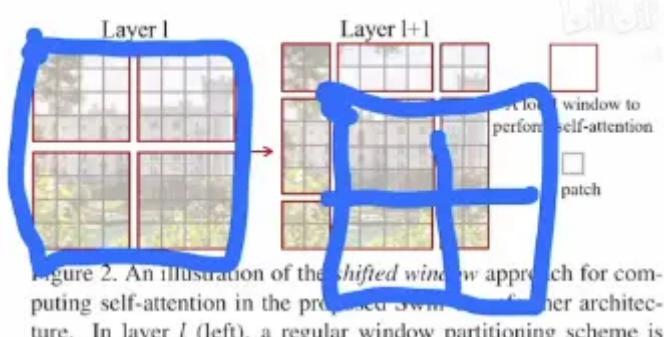
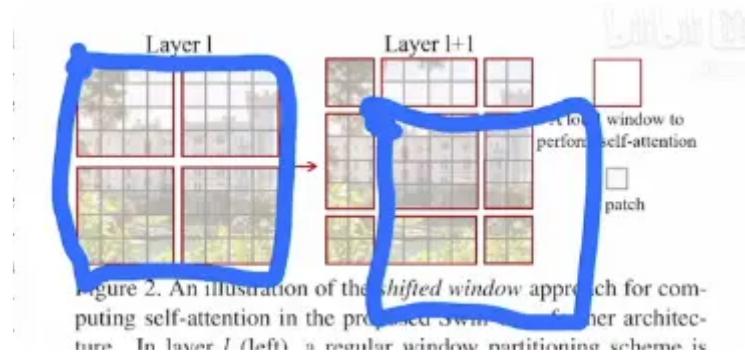


灰色的小patch 4\*4大小

红色的框 窗口 Swin transformer里面一个窗口默认有 $7 \times 7 = 49$ 个patch

那么最初有 $224/7/4=8$ 就是8\*8个窗口

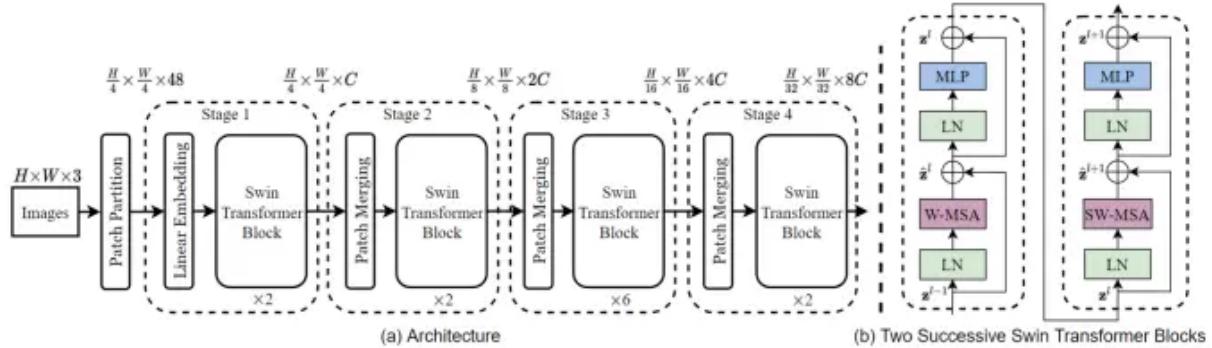
## Shift 操作



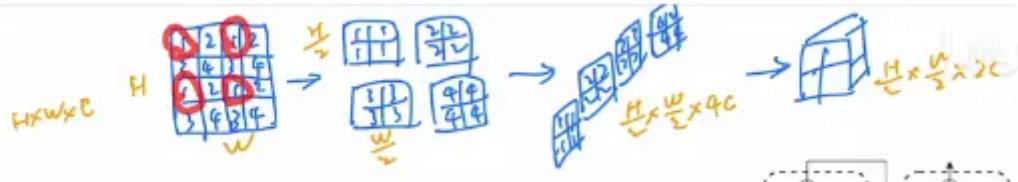
通过shift：原来patch只能与同个window自注意力，通过shift能够与其他窗口patch交互，配合patch merging 每个patch的感受野相当可观

### 3. Method

#### Major Process



- Images 224\*224\*3
- Patch Partition: 一个patch是4\*4 现在图片变成56\*56\*48(48是向量维度)
- Linear Embedding: 把向量维度变成预设值HyperPram c(tiny设成96) 前面拉直 变成3136 那么变成了3136\*96 (一次卷积完成 等价于VIT里面的 Patch Projection)
- Swin transformer block 基于窗口自注意力计算 暂时看作黑盒(transformer输入输出尺寸不变), 那么出来也是56 5696
- Patch Merging:



将downsample的样本同一个位置进行拼接( 实际上就是做了4次pooling然后给你接在一起了 )

变成四个 $h/2$ 、 $w/2$ 的张量

然后在c的维度拼接  $h/2 * w/2 * 4c$

为了等同于CNN 我们只想让c维度翻倍, 用1\*1卷积变成 $h/2 * w/2 * 2c$  的一个张量

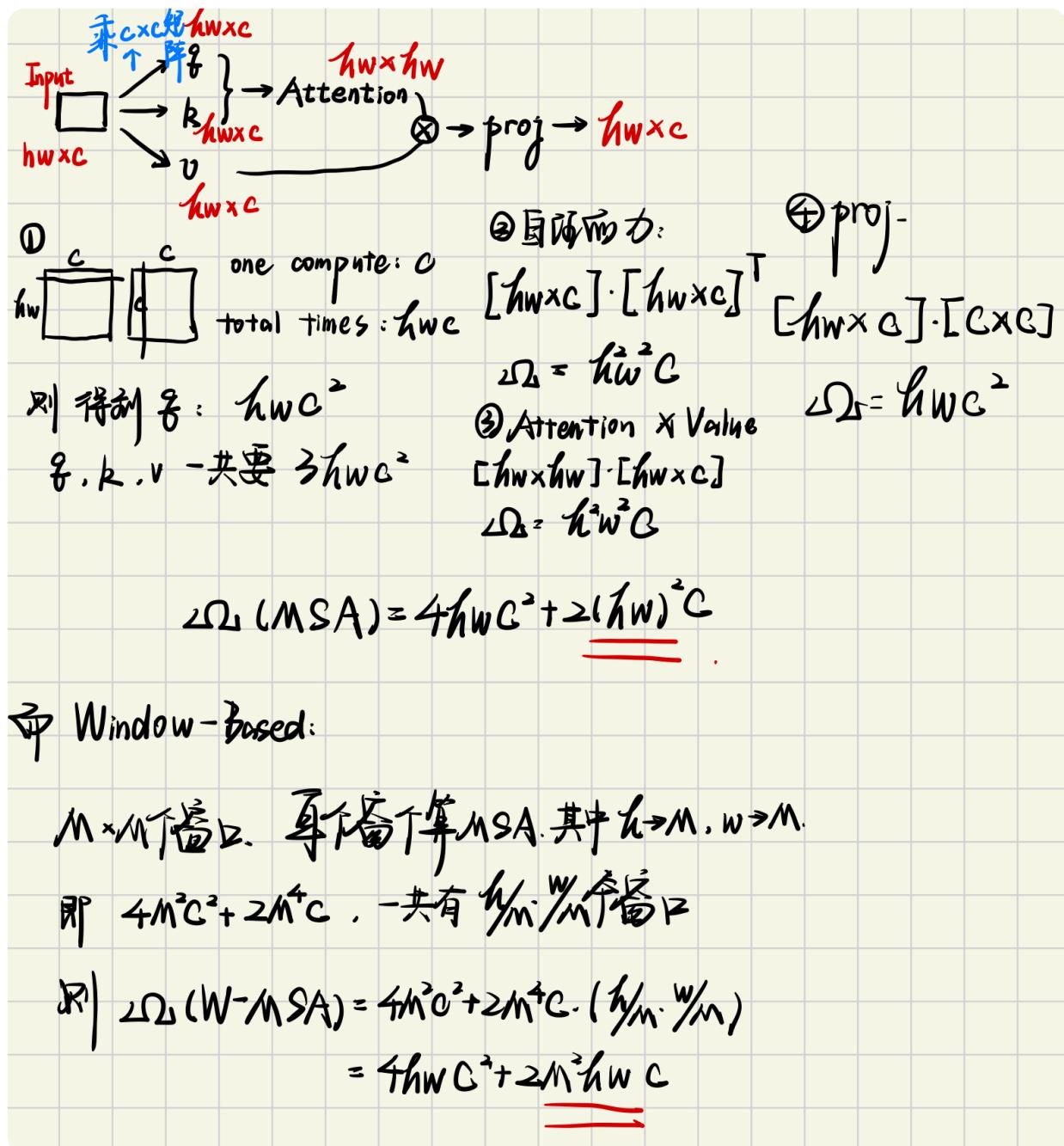
- 输出的大小就从56\*56\*96变成了28\*28\*192, 经过stage2中的 Transformer block, 还是28\*28\*192
- stage3, 4同理 维度降成14\*14\*384以及7\*7\*768
- 如果做分类 给你flatten一下1\*768, 然后又变成了1\*1,000

# Shifted Window based Self Attention

复杂度分析

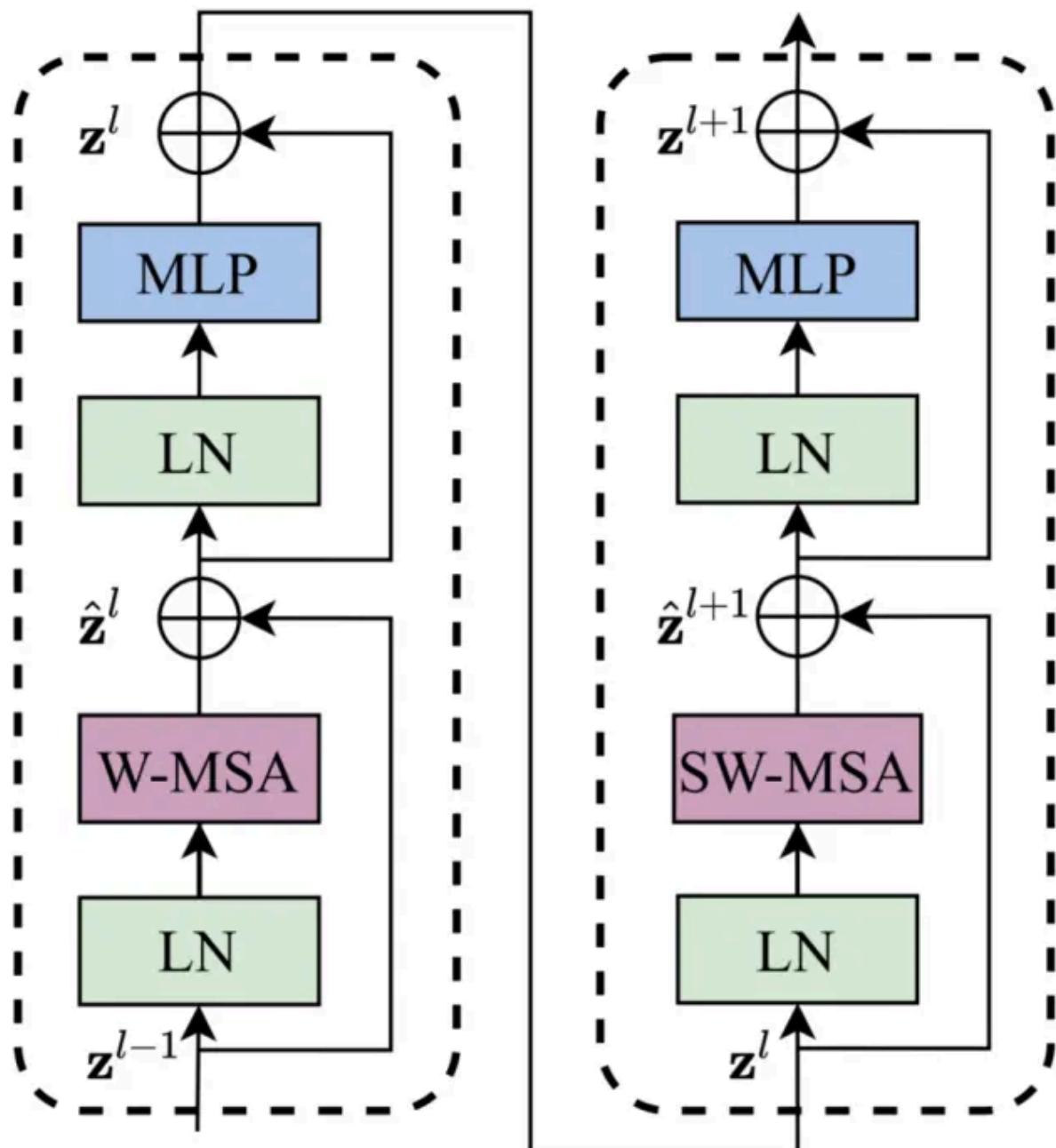
$$\Omega(\text{MSA}) = 4hwC^2 + 2(hw)^2C, \quad (1)$$

$$\Omega(\text{W-MSA}) = 4hwC^2 + 2M^2hwC, \quad (2)$$



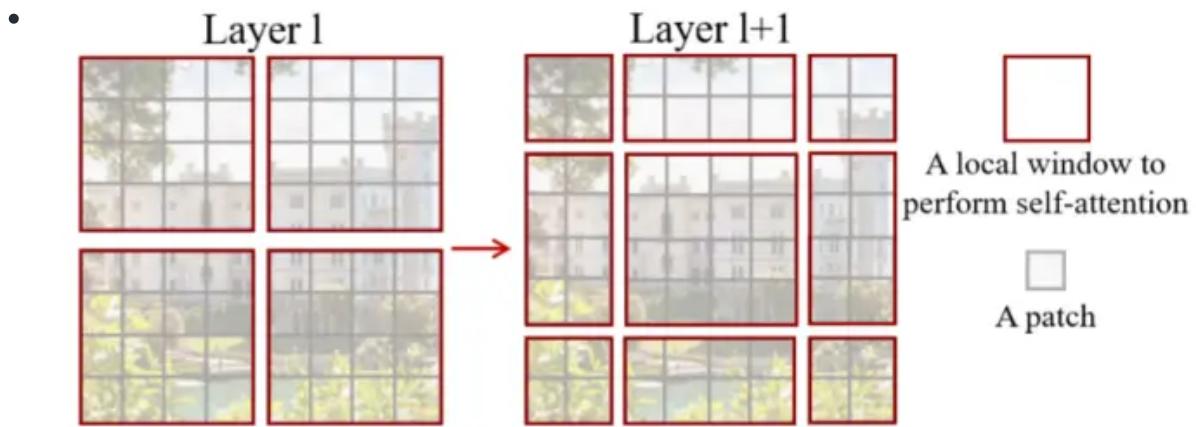
模型架构

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



先基于窗口做MSA 然后基于移动窗口做MSA

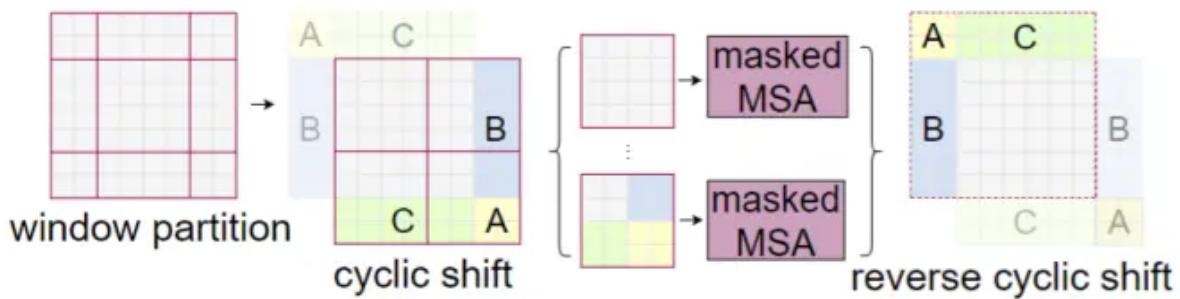
# Masking



移动窗口下 每个patch大小不一 快速运算需要压成一个patch 但是窗口大小不一做不到

考虑Padding会增加复杂度(原来4个窗口 现在9个窗口)

## Solving Masking



循环移位 将左上角移到右下角来 然后分成四宫格 算完了我们再给他换回去

? 原来离很远的部分现在紧挨着 理论上不该做自注意力

solving: Masking

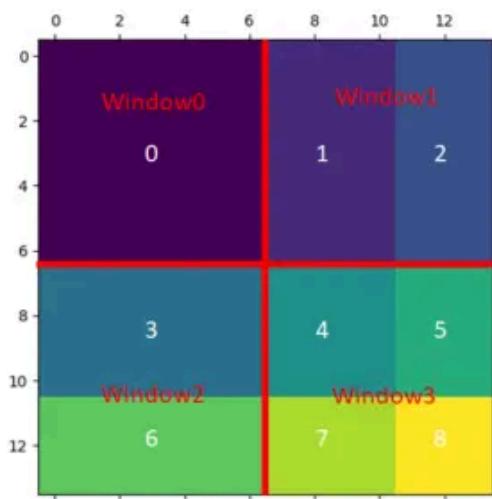
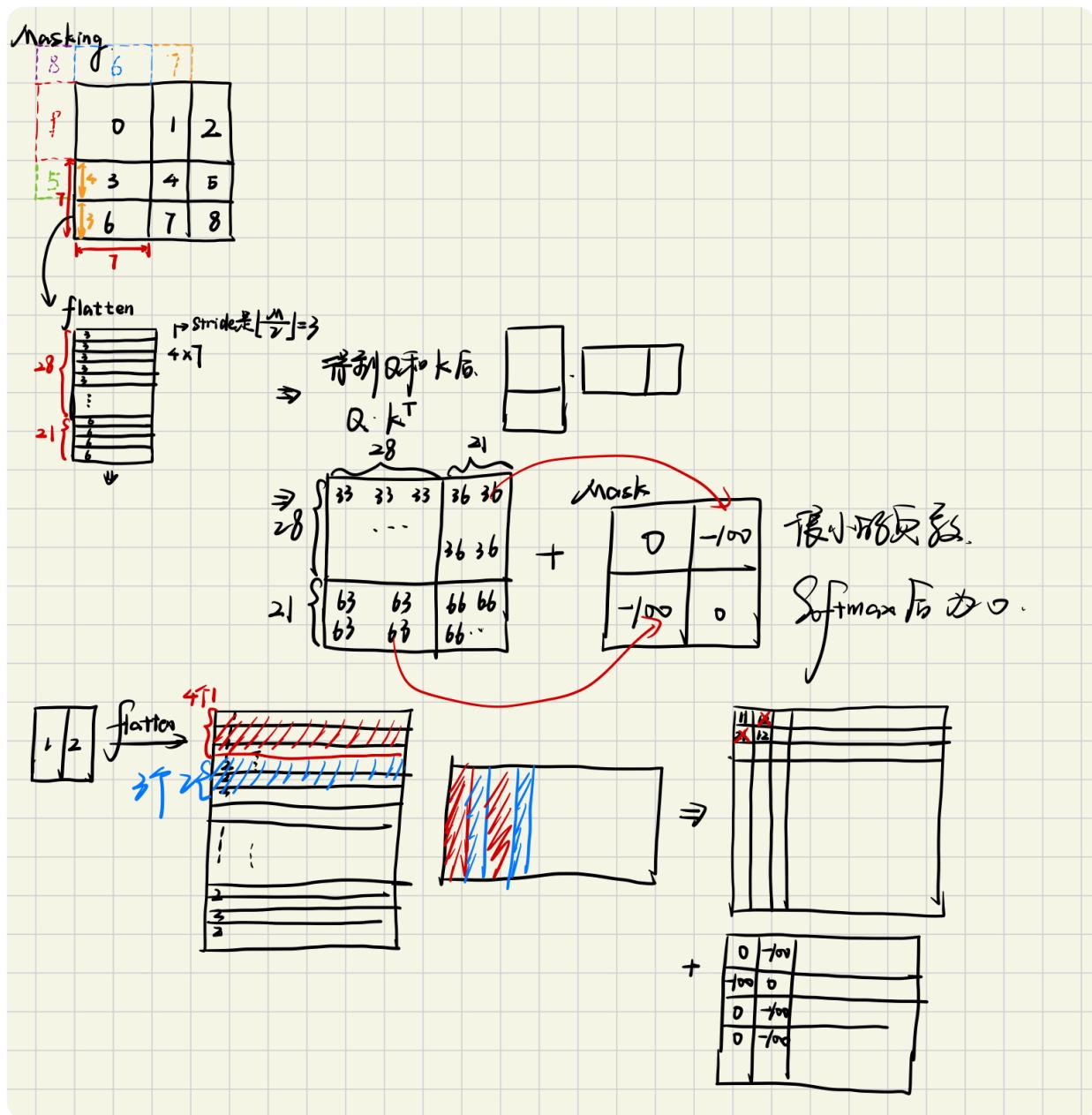
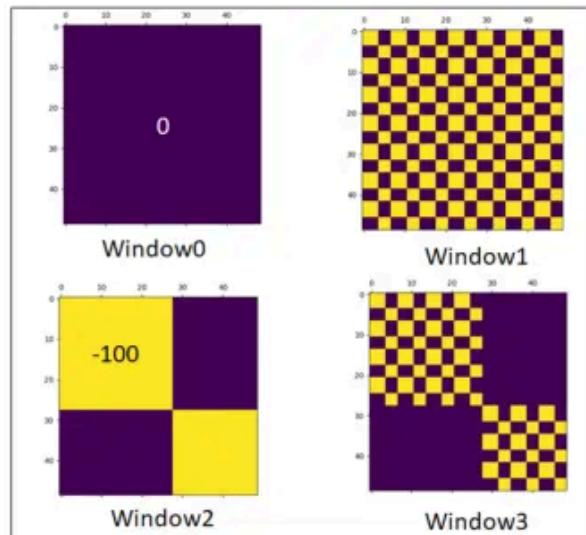


Image Mask  
(14x14, window 7x7, shift 3)



Attn Mask

## 4.Others

变体

- Swin Tiny (复杂度跟resnet-50差不多)
- Swin Small (跟Resnet-101差不多)
- Swin Base
- Swin Large

差别：

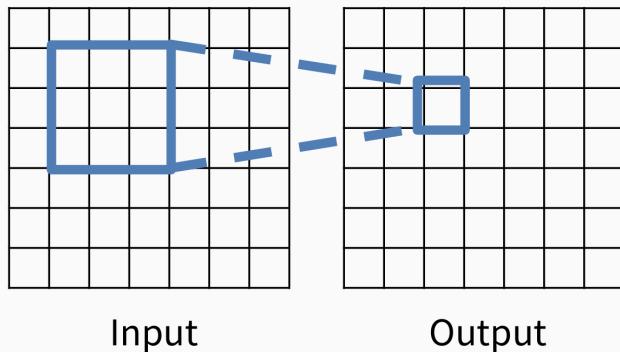
- 一个是向量维度的大小  $c$
- 另一个是每个 stage 里到底有多少个 transform block
- ...
- Swin-T:  $C = 96$ , layer numbers = {2, 2, 6, 2}
- Swin-S:  $C = 96$ , layer numbers = {2, 2, 18, 2}
- Swin-B:  $C = 128$ , layer numbers = {2, 2, 18, 2}
- Swin-L:  $C = 192$ , layer numbers = {2, 2, 18, 2}

### 概念回顾

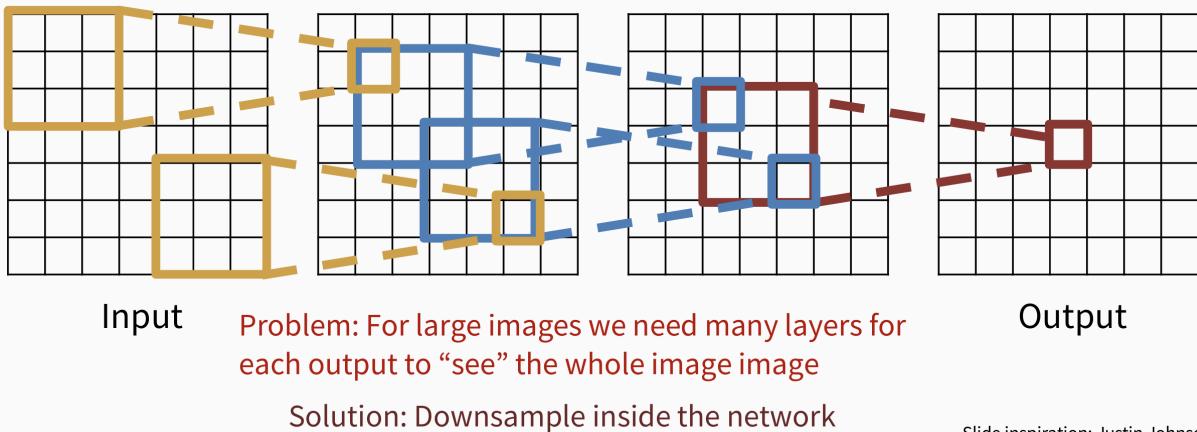
感受野：

#### Receptive Fields

For convolution with **kernel size K**, each element in the output depends on a  $K \times K$  receptive field in the input



Each successive convolution adds  $K - 1$  to the receptive field size  
With  $L$  layers the receptive field size is  $1 + L * (K - 1)$



Slide inspiration: Justin Johnson

## 下采样率

**下采样率** (Downsampling Rate) 指的是通过某种操作 (如池化或卷积步长) 对输入特征图 (Feature Map) 的空间维度 (高度和宽度) 进行缩小的比例。其核心目的是逐步减少特征图的分辨率，从而降低计算量、增加感受野 (Receptive Field)，并提取更高层次的抽象特征。

### 实现方式：

- **池化操作 (Pooling) :**
  - **最大池化 (Max Pooling)** 或 **平均池化 (Average Pooling)** 是典型的下采样方法。
  - 池化的核 (Kernel Size) 和步长 (Stride) 决定下采样率：
    - 若使用  $2 \times 2$  池化核，步长为 2，则下采样率为 **2**。
- **卷积步长 (Strided Convolution) :**
  - 当卷积的步长 (Stride) 大于 1 时，也会导致下采样。
  - 例如，步长为 2 的卷积会将特征图尺寸减半，下采样率为 **2**。