Swin Transformer: Hierarchical Vision Transformer using Shifted Windows #论文

Metadata

- Authors:: Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, Baining Guo
- 作者机构::
- Keywords:: Computer-Science---Computer-Vision-and-Pattern-Recognition, Computer-Science---Machine-Learning
- Date:: 2021-08-17
- 状态:: #待读
- 对象:: Swin Transformer
- 方法:: 层级式、移动窗口
- 分类:: CV
- 内容:: 利用了CNN的模型的先验知识。
- PDF Attachments
  - Liuet al*Swin Transformer2021*.pdf

Abstract

This paper presents a new vision Transformer, called Swin Transformer, that capably serves as a general-purpose backbone for computer vision. Challenges in adapting Transformer from language to vision arise from differences between the two domains, such as large variations in the scale of visual entities and the high resolution of pixels in images compared to words in text. To address these differences, we propose a hierarchical Transformer whose representation is computed with \textbf{S}hifted \textbf{win}dows. The shifted windowing scheme brings greater efficiency by limiting self-

attention computation to non-overlapping local windows while also allowing for cross-window connection. This hierarchical architecture has the flexibility to model at various scales and has linear computational complexity with respect to image size. These qualities of Swin Transformer make it compatible with a broad range of vision tasks, including image classification (87.3 top-1 accuracy on ImageNet-1K) and dense prediction tasks such as object detection (58.7 box AP and 51.1 mask AP on COCO test-dev) and semantic segmentation (53.5 mIoU on ADE20K val). Its performance surpasses the previous state-of-the-art by a large margin of +2.7 box AP and +2.6 mask AP on COCO, and +3.2 mIoU on ADE20K, demonstrating the potential of Transformer-based models as vision backbones. The hierarchical design and the shifted window approach also prove beneficial for all-MLP architectures. The code and models are publicly available at~\url{https://github.com/microsoft/Swin-Transformer}.

理论

- Swin Transformer 可以作为一个通用的骨干网络，不仅可以做分类，还可以做密集预测型任务。

内容

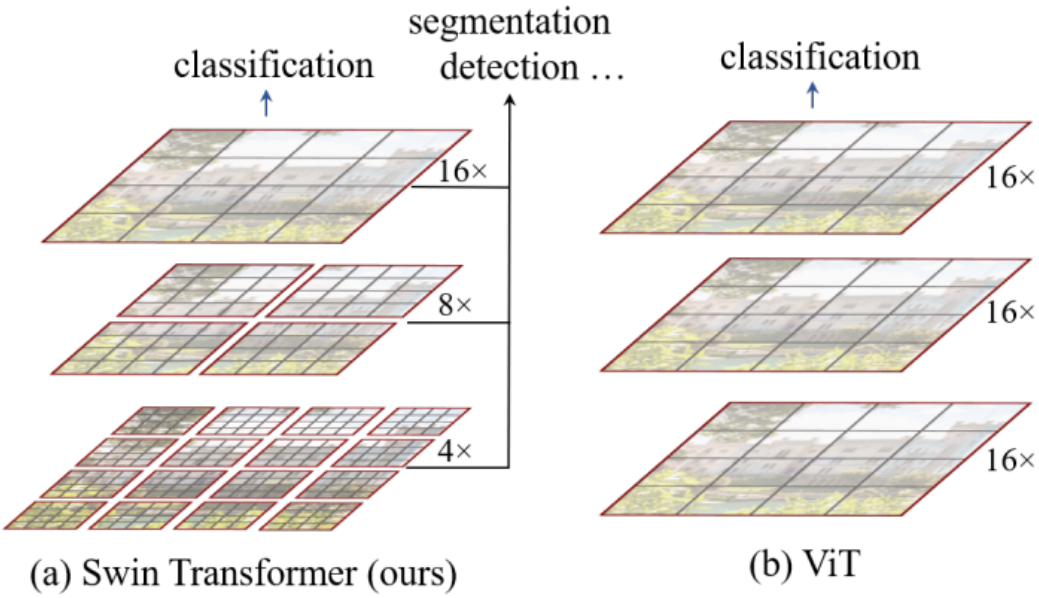


Figure 1. (a) The proposed Swin Transformer builds hierarchical feature maps by merging image patches (shown in gray) in deeper layers and has linear computation complexity to input image size due to computation of self-attention only within each local window (shown in red). It can thus serve as a general-purpose 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.

-
- ViT 对多尺寸特征的掌握程度就会相对弱些。
  - 自始自终处理16倍下采样率过后的特征，不适合处理密集预测型任务。
  - 始终在整图上建模（计算全局的自注意力），计算复杂度与图像的长度平方成正比。
- Swin Transformer 小窗口之内计算自注意力，而不是在整张图。
  - 只要窗口大小固定，那么自注意力的复杂度就是固定的。
  - 复杂度随图片大小成线性增长。
    - 本质：利用了locative 的先验偏置。
  - patch merging：合成相邻的patch （类似于CNN中的pooling）得到多尺度的特征图。

方法

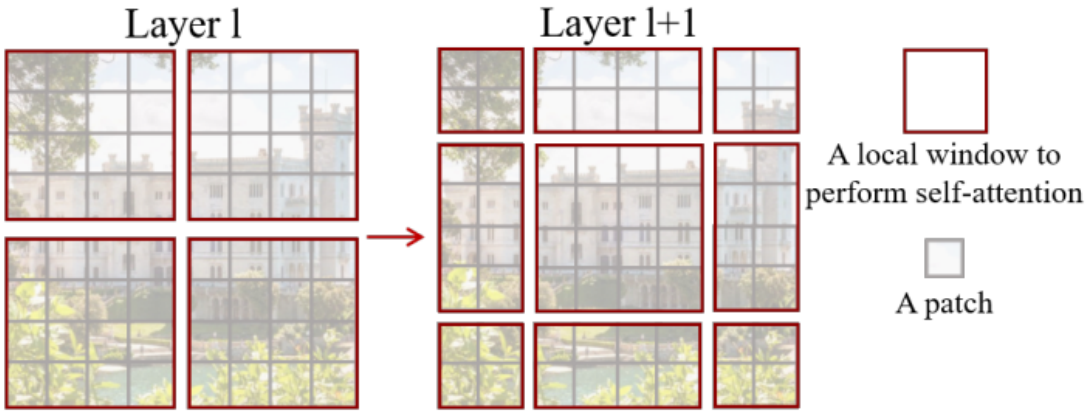


Figure 2. An illustration of the *shifted window* approach for computing self-attention in the proposed Swin Transformer architecture. In layer l (left), a regular window partitioning scheme is adopted, and self-attention is computed within each window. In the next layer $l+1$ (right), the window partitioning 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.

-
- Shift 操作，使得不同窗口在不同层之间可以有cross windows connection，然后结合patch emerging操作，使得局部感受野可以近似全局的感受野，这使得计算复杂度大幅度下降。
  - 先做一次窗口多头自注意力，再做shifted window的多头注意力。这两个结构联合起来才是swintransformer连在一起作为一个单元。
- 掩码方式，降低运算量。**介绍了一种高效批次的自注意力运算**

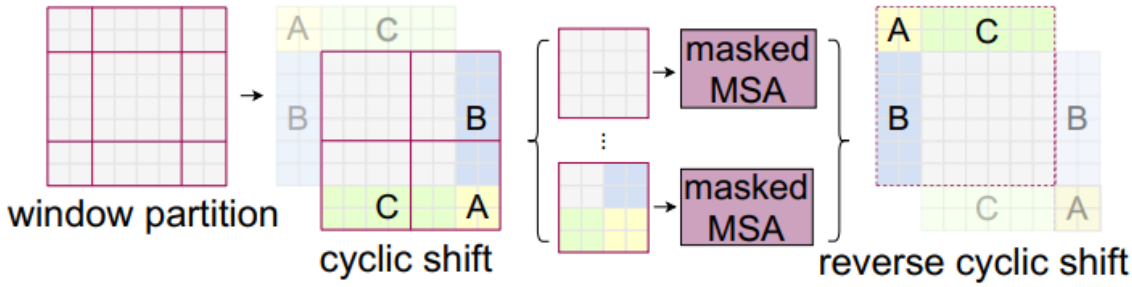


Figure 4. Illustration of an efficient batch computation approach for self-attention in shifted window partitioning.

-
- 循环移位之后，解决了窗口数量变多，大小不一致的问题。
- cyclic shift 之后再通过掩码操作，从而能让同一窗口中，只经过一次前向操作，计算出不同区域的分别自注意力来。（本质是图中含不同颜色的的区块不应一起做自注意力）

- 最后再循环回移。这里每一个移动的区域需要标记一些序号，（图中使用了不同颜色）用以标记不同不同窗口内，不同颜色（标号）区域不应该一起做自注意力计算。
- 自注意力中，各个区域块向量拉直，与自身转置后相乘，求解自注意力矩阵。
- 在自注意力矩阵经过softmax之前，需要加上一个掩码矩阵。

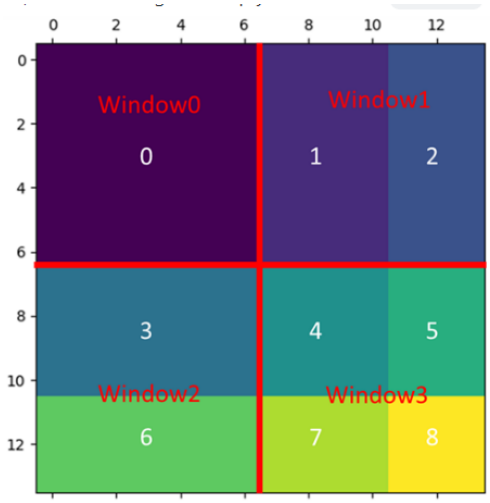
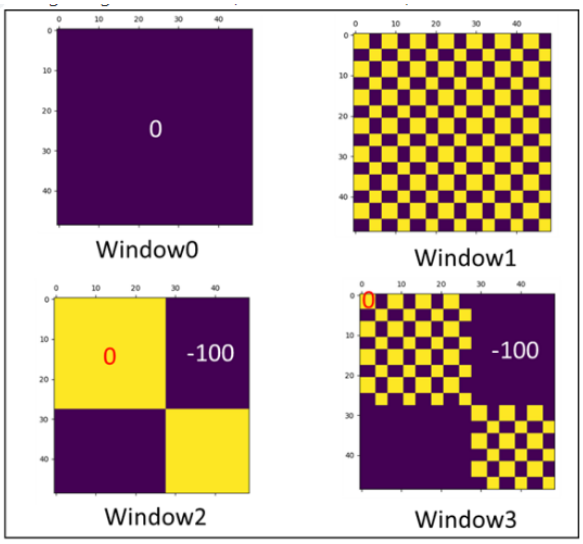


Image Mask
(14x14, window 7x7, shift 3)



Attn Mask

- softmax的时候会把很小的负数变成零，所以掩码的时候只需要在对应不同区域的（非对角区域块）加上大的负数，使得他在经过softmax的时候输出接近零即可。

-```

python demo of mask

```
import torch

import matplotlib.pyplot as plt


def window_partition(x, window_size):
    """
    Args:
        x: (B, H, W, C)
        window_size (int): window size
    Returns:
        windows: (num_windows*B, window_size, window_size, C)
    """
    B, H, W, C = x.shape
    x = x.view(B, H // window_size, window_size, W //
```

```
    window_size, window_size, C)
        windows = x.permute(0, 1, 3, 2, 4,
    5).contiguous().view(-1, window_size, window_size, C)
        return windows


    window_size = 7
    shift_size = 3
    H, W = 14, 14
    img_mask = torch.zeros((1, H, W, 1))  # 1 H W 1
    h_slices = (slice(0, -window_size),
                slice(-window_size, -shift_size),
                slice(-shift_size, None))
    w_slices = (slice(0, -window_size),
                slice(-window_size, -shift_size),
                slice(-shift_size, None))
    cnt = 0
    for h in h_slices:
        for w in w_slices:
            img_mask[:, h, w, :] = cnt
            cnt += 1

    mask_windows = window_partition(img_mask, window_size)  # nW,
    window_size, window_size, 1
    mask_windows = mask_windows.view(-1, window_size *
    window_size)

    attn_mask = mask_windows.unsqueeze(1) -
    mask_windows.unsqueeze(2)
    attn_mask = attn_mask.masked_fill(attn_mask != 0,
    float(-100.0)).masked_fill(attn_mask == 0, float(0.0))

    plt.matshow(img_mask[0, :, :, 0].numpy())
    plt.matshow(attn_mask[0].numpy())
    plt.matshow(attn_mask[1].numpy())
    plt.matshow(attn_mask[2].numpy())
    plt.matshow(attn_mask[3].numpy())

    plt.show()
```

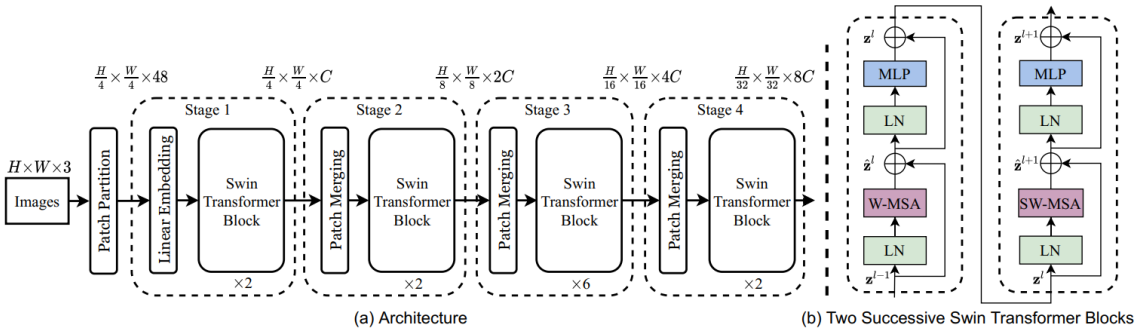


Figure 3. (a) The architecture of a Swin Transformer (Swin-T); (b) two successive Swin Transformer Blocks (notation presented with Eq. (3)). W-MSA and SW-MSA are multi-head self attention modules with regular and shifted windowing configurations, respectively.

-
- Images 224×224×3
- patch partition 56×56×48
- Linear Embedding 56×56×96 → 3136×96
- Swin Transformer block
- Patch merging
    - 下采样特征图
    - 空间大小减半，通道数乘二（与CNN等价起来）
- 在这里没有画最后的检测头/分类头
    - 只画了网络的主干部分
- 披着CNN皮的Transformer架构。

结论

- 多层级式→ patch merging，移动窗口。相对于输入图像大小具有线性计算复杂度。

亮点

- 展望：
    - 作为Swin Transformer的关键要素，基于窗户的自我注意事项被证明是有效且有效的，我们也期待研究其在自然语言处理中的使用。
    - 假如能够实现，那么将是模型大一统方面的里程碑。

灵感

- 移动窗口可以用来解决窗口之间互相通讯。
- 采用了掩码的方式提高移动窗口计算效率。
- 采用相对位置编码。

Zotero links

- DOI: 10.48550/arXiv.2103.14030