Image Style Transfer with Transformers

张瀚文，2201212865，信息工程学院

2022 年 11 月 29 日

1 论文摘要及贡献

摘要 :由于 CNN 的局部感知域性和空间不变性,输入图像的全局信息难以提取和维护。因此，传统的神经网络风格传递方法通常是有偏差的，对于同一幅参考风格图像，通过多次运行风格迁移过程可以观察到内容泄漏。为了解决这个关键问题，该文提出了一种基于 Transformer 的方法，即 StyTr2，将输入图像的长期依赖关系考虑到无偏风格传输中。 与用于其他视觉任务的视觉转换器不同，我们的 StyTr2 包含两个不同的转换器编码器，分别为内容和样式生成特征序列。在编码器之后，采用多层 Transformer 解码器，根据样式序列对内容序列进行风格化。

主要贡献 :经典的基于深度学习的图像风格迁移，样式转换方法使用多层 cnn 来学习样式和内容表示。由于卷积层的接收域有限，CNN 无法处理长距离依赖关系。输入图像难以获得全局信息,这是图像风格传递任务的关键。本文将将 Transformer 应用于图像风格迁移，可以通过自注意力机制帮助模型有更大的感受野，从而获得一个图像全局归纳的信息，且可以避免了 content leak 的产生。

- 提出一个基于 Transformer 的风格转换框架,以减少内容泄漏并实现无偏的风格化；
- 提出一种内容感知的位置编码机制，该机制是尺度不变的，适用于视觉生成任务；

2 代码主体分析

输入 content_img 和 style_img,然后首先要进行 split the content and style images into patches 得到 content1 和 style1 ,这一步通过 class PatchEmbed 实现核心代码。

当使用基于 Transformer 的模型时，需要在输入序列中加入位置编码(PE)以获取结构信息。此论文提出了基于图像语义的位置编码，这一改进基于以下两个想法：

- 在传统的位置编码中：两个 patch 之间的位置相对关系仅仅与它们之间的距离有关。而对于图像生成任务，在计算位置编码时，我们应该考虑图像的语义。
- 当输入图像的尺寸呈指数增长时,传统的正弦位置编码是否仍然适用于视觉任务? 如下所示当调整输入图像的大小时，相同语义的 patches (blue blocks)之间的相对关系会发生巨大的变化，这可能不适合视觉任务中多大小的输入。

```python
class PatchEmbed(nn.Module):
    """ Image to Patch Embedding
    """
    def __init__(self, img_size=256, patch_size=8, in_chans=3, embed_dim=512):
        super().__init__()
        img_size = to_2tuple(img_size)
        patch_size = to_2tuple(patch_size)
        num_patches = (img_size[1] // patch_size[1]) * (img_size[0] // patch_size[0])
        self.img_size = img_size
        self.patch_size = patch_size
        self.num_patches = num_patches

        self.proj = nn.Conv2d(in_chans, embed_dim, kernel_size=patch_size, stride=patch_size)
        self.up1 = nn.Upsample(scale_factor=2, mode='nearest')

    def forward(self, x):
        B, C, H, W = x.shape
        x = self.proj(x)

        return x
```

接下来就是把 content1 和 style1 分别 flatten 经过一个 Transformer Encoder 得到 content2 和 style2。Transformer Encoder 核心就是多头注意力机制。`flatten` 操作放在 class Transformer 中。

```python
def forward(self, style, mask , content, pos_embed_c, pos_embed_s):

    # content-aware positional embedding
    content_pool = self.averagepooling(content)
    pos_c = self.new_ps(content_pool)
    pos_embed_c = F.interpolate(pos_c, mode='bilinear',size= style.shape[-2:])

    ###flatten NxCxHxW to HWxNxC
    style = style.flatten(2).permute(2, 0, 1)
    if pos_embed_s is not None:
        pos_embed_s = pos_embed_s.flatten(2).permute(2, 0, 1)

    content = content.flatten(2).permute(2, 0, 1)
    if pos_embed_c is not None:
        pos_embed_c = pos_embed_c.flatten(2).permute(2, 0, 1)
```

然后输入 content2 和 style2 到这个 Transformer decoder，得到 hs。这个 Transformer decoder 核心部分还是多头自注意力机制。

```python
class TransformerDecoderLayer(nn.Module):

    def __init__(self, d_model, nhead, dim_feedforward=2048, dropout=0.1,
                 activation="relu", normalize_before=False):
        super().__init__()
        # d_model embedding dim
        self.self_attn = nn.MultiheadAttention(d_model, nhead, dropout=dropout)
        self.multihead_attn = nn.MultiheadAttention(d_model, nhead, dropout=dropout)      diyiiyiii, 9个月前 • Adc
        # Implementation of Feedforward model
        self.linear1 = nn.Linear(d_model, dim_feedforward)
        self.dropout = nn.Dropout(dropout)
        self.linear2 = nn.Linear(dim_feedforward, d_model)

        self.norm1 = nn.LayerNorm(d_model)
        self.norm2 = nn.LayerNorm(d_model)
        self.norm3 = nn.LayerNorm(d_model)
        self.dropout1 = nn.Dropout(dropout)
        self.dropout2 = nn.Dropout(dropout)
        self.dropout3 = nn.Dropout(dropout)

        self.activation = _get_activation_fn(activation)
        self.normalize_before = normalize_before
```

3 配置环境测试

环境按照作者所提供的要求安装，但需要注意在 colab 测试时在 pytorch 1.4.0 下未能测试成功，升级到 pytorch 1.6.0 后成功运行，指令为：pip install torch==1.6.0+cpu torchvision==0.7.0+cpu -f https://download.pytorch.org/whl/torch_stable.html

Requirements

- python 3.6
- pytorch 1.4.0
- PIL, numpy, scipy
- tqdm

```
[7] pip install torch==1.6.0+cpu torchvision==0.7.0+cpu -f https://download.pytorch.org/whl/torch_stable.html

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Looking in links: https://download.pytorch.org/whl/torch_stable.html
Collecting torch==1.6.0+cpu
  Downloading https://download.pytorch.org/whl/cpu/torch-1.6.0%2Bcpu-cp37-cp37m-linux_x86_64.whl (154.6 MB)
     |████████████████████████████████| 154.6 MB 45 kB/s
Collecting torchvision==0.7.0+cpu
  Downloading https://download.pytorch.org/whl/cpu/torchvision-0.7.0%2Bcpu-cp37-cp37m-linux_x86_64.whl (5.1 MB)
     |████████████████████████████████| 5.1 MB 44.5 MB/s
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from torch==1.6.0+cpu) (1.21.6)
Requirement already satisfied: future in /usr/local/lib/python3.7/dist-packages (from torch==1.6.0+cpu) (0.16.0)
Requirement already satisfied: pillow>=4.1.1 in /usr/local/lib/python3.7/dist-packages (from torchvision==0.7.0+cp
Installing collected packages: torch, torchvision
  Attempting uninstall: torch
    Found existing installation: torch 1.4.0+cpu
    Uninstalling torch-1.4.0+cpu:
      Successfully uninstalled torch-1.4.0+cpu
  Attempting uninstall: torchvision
    Found existing installation: torchvision 0.5.0+cpu
    Uninstalling torchvision-0.5.0+cpu:
      Successfully uninstalled torchvision-0.5.0+cpu
```

- 运行指令：!python test.py --content content.jpg --style style.jpg --output out
- 运行结果：从左到右依次为 content，style，style-transfer-image