

cv第九次作业

一、作业要求

- 寻找一篇2021/2022年风格迁移的文章
- 翻译其摘要和贡献；对代码主体部分进行注释，截图
- 配置环境，测试自己的图片进行风格迁移的结果，截图

二、作业过程

1、选题

Deng Y, Tang F, Dong W, et al. StyTr²: Image Style Transfer with Transformers[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022: 11326–11336.

2、摘要

图像风格转换的目的是在保持图像原有内容的同时，在风格参照的指导下，使图像具有艺术特征。由于卷积神经网络（CNNs）的局部性，难以提取和维护输入图像的全局信息。因此，传统的神经风格迁移方法面临着内容表征的偏颇。为了解决这一关键问题，本文提出了一种基于变换器的StyTr²方法，将输入图像的长距离依赖性考虑在内，用于图像风格的转换。与用于其他视觉任务的视觉转换器相比，StyTr²包含两个不同的转换器编码器，分别为内容和样式生成域特定序列。在编码器之后，采用多层变换器解码器来根据风格序列对内容序列进行风格化。本文还分析了现有位置编码方法的不足，提出了基于内容感知的位置编码（CAPE），该编码具有尺度不变性，更适合图像风格转换任务。定性和定量实验结果表明，与现有的基于神经网络和基于流的方法相比，StyTr²方法具有更好的性能。有关代码和型号的信息，请访问<https://github.com/diyiyiii/StyTR-2>。

3、贡献

总之，我们的主要贡献包括：

- 一个基于转换器的风格转换框架StyTr²，用于生成风格化结果，并保留输入内容图像的结构和细节。
- 一种内容感知的位置编码方案，具有尺度不变性，适合于风格转换任务。（CAPE, content-aware positional encoding）

综合实验表明，StyTr²优于基线方法，并取得了令人满意的内容结构和样式模式的出色结果。

4、代码注释

代码: <https://github.com/diyiyiii/StyTR-2>

最核心的部分是CAPE (Content-Aware Positional Encoding) 方法，下面就对CAPE部分代码进行注释：

(1) 生成patches的embed

给定输入图像，首先对图像做transform，变成256*256大小：

```
def train_transform():
    transform_list = [
        transforms.Resize(size=(512, 512)),
        transforms.RandomCrop(256),
        transforms.ToTensor()
    ]
    return transforms.Compose(transform_list)
```

然后把输入图像分成L个8*8大小的patches，每个patch的维度是C，即，每个patch由3维变为C维，然后经过一个线性投影层。其实就是经过一个conv(indim:3,outdim:512,kernelsize:8,stride:8),变成512x32x32大小的特征。生成序列特征编码sigma，C是sigma的维度。

```
self.proj =
nn.Conv2d(in_chans, embed_dim, kernel_size=patch_size, stride=patch_size)
        #N 32 32 512
...
x = self.proj(x)
```

(2) CAPE

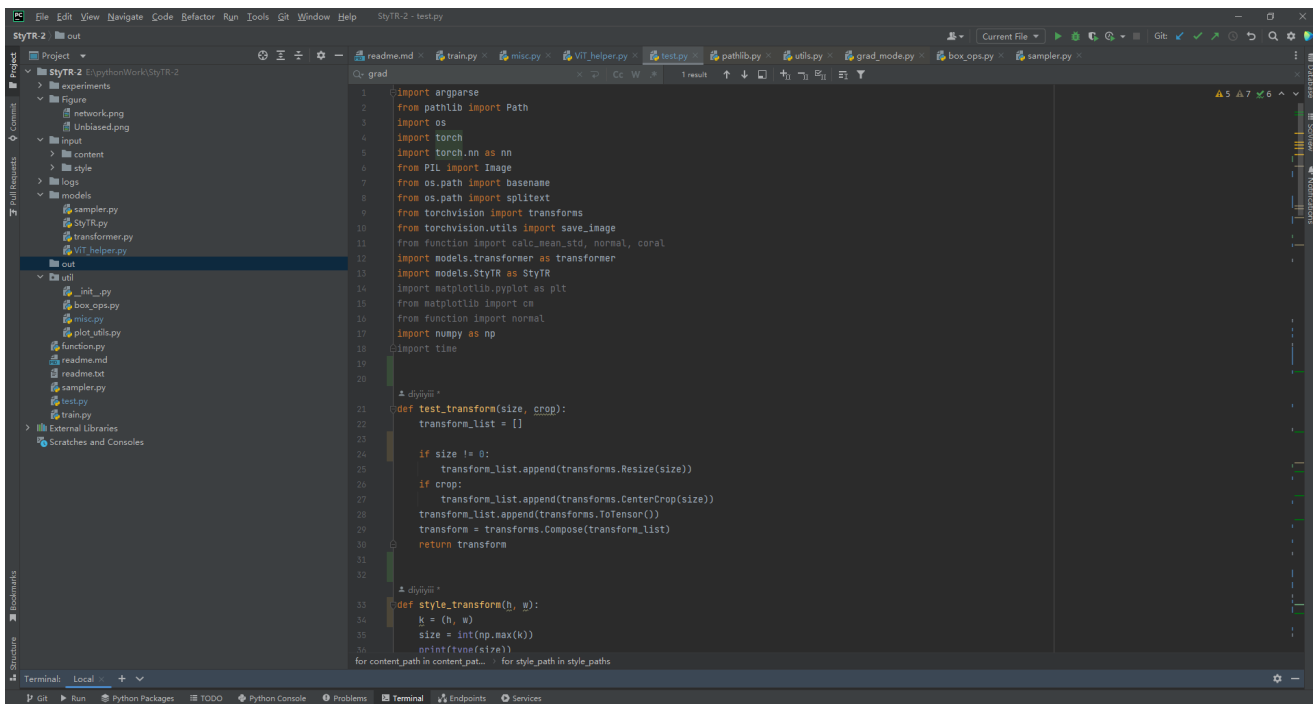
重新缩放为18*18，并通过一个维度不变的1x1卷积用作训练出来的位置编码，然后插值为256x256。

```
self.new_ps = nn.Conv2d(512, 512, (1,1))
self.averagepooling = nn.AdaptiveAvgPool2d(18)
```

```
# content-aware positional embedding
content_pool = self.averagepooling(content)      #18 18 512
pos_c = self.new_ps(content_pool)
pos_embed_c = F.interpolate(pos_c, mode='bilinear', size=
style.shape[-2:])      # N C:512 H:256 W:256
```

5、实验

实验结果：



```
1 import argparse
2 from pathlib import Path
3 import os
4 import torch
5 import torch.nn as nn
6 from PIL import Image
7 from os.path import basename
8 from os.path import splitext
9 from torchvision import transforms
10 from torchvision.utils import save_image
11 from function import calc_mean_std, normal, coral
12 import models.transformer as transformer
13 import models.StyTR as StyTR
14 import matplotlib.pyplot as plt
15 from matplotlib import cm
16 from function import normal
17 import numpy as np
18 import time
19
20
21 # def test_transform(size, crop):
22 #     transform_list = []
23 #
24 #     if size != 0:
25 #         transform_list.append(transforms.Resize(size))
26 #     if crop:
27 #         transform_list.append(transforms.CenterCrop(size))
28 #     transform_list.append(transforms.ToTensor())
29 #     transform = transforms.Compose(transform_list)
30 #     return transform
31
32
33 # def style_transform(h, w):
34 #     k = (h, w)
35 #     size = int(np.max(k))
36 #     print(f'vne(size)')
37 #     for content_path in content_paths:
38 #         for style_path in style_paths
```

Style



Content

