计算机视觉第九次作业 - 风格迁移

姓名：岳东旭　　学号：2201212864　　指导老师：张健

1.问题描述

- 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

2.论文阅读-CLIPstyler: Image Style Transfer with a Single Text Condition

2.1摘要

目前的神经架构风格迁移方法都需要参考风格图片，从而将风格图片的纹理信息迁移到内容图片上。然而，在很多实际的场景中，我们可能没有可供参考的风格图片，但是仍然想通过想象来做风格迁移。为了解决这样的应用场景，我们提出了一个新的网络架构，不需要风格图片，仅仅需要目标风格的文本描述就可以完成风格迁移任务。使用预训练CLIP的文本图像编码模型，我们揭示了内容图像的风格的表征只需要一个文本条件。具体而言，我们提出了一种小块式文本-图像匹配损失与多视图增量逼真的纹理转移。扩展实验的结果证实了使用能够反映语义查询文本的逼真的纹理可以成功地进行图像风格迁移。

2.2主要贡献

以往的方法主要是将文本条件的语义信息传递到视觉域，然而存在的问题是由于嵌入模型的性能，语义信息不能被正确的表征，而且对图像风格的迁移只能局限在特定的内容域，比如只能针对人脸图像进行风格迁移，因为这些方法极大地依赖于预训练的生成模型。

针对以上问题，本文的主要贡献如下：

1. 使用最新的CLIP模型做文本嵌入编码。
2. 本文提出训练一个轻量化的CNN网络来表达纹理信息，产生丰富的结果。
3. 提出了一种小块式文本-图像匹配损失与多视图增量逼真的纹理转移。

3.代码主体

先创建不需要计算梯度的特征

```
with torch.no_grad():
    template_text = compose_text_with_templates(prompt, imagenet_templates)  # 把文本变成clip认识的东西哦
    tokens = clip.tokenize(template_text).to(device)  # 分词
    text_features = clip_model.encode_text(tokens).detach()  # clip对文本进行embedding
    text_features = text_features.mean(axis=0, keepdim=True)  # 对所有的79个语句做了均值
    text_features /= text_features.norm(dim=-1, keepdim=True)  # 除了方差

    template_source = compose_text_with_templates(source, imagenet_templates)  # 把源语句也编码一下
    tokens_source = clip.tokenize(template_source).to(device)
    text_source = clip_model.encode_text(tokens_source).detach()
    text_source = text_source.mean(axis=0, keepdim=True)
    text_source /= text_source.norm(dim=-1, keepdim=True)
    source_features = clip_model.encode_image(clip_normalize(content_image, device))  # 用clip把最原始的图片做编码
    source_features /= (source_features.clone().norm(dim=-1, keepdim=True))
```

计算文章中提到的损失函数

```
for epoch in range(0, steps + 1):

    scheduler.step()
    target = style_net(content_image, use_sigmoid=True).to(device)  # 原图像原封不动经过U-net
    target.requires_grad_(True)  # 参数可学习

    target_features = utils.get_features(img_normalize(target), VGG)  # 从U-net生成的特征放到训练好的VGG内，得到那几层的特征图

    content_loss = 0
    # 拿到原图的VGG特征和经过U-net的VGG特征进行对比
    content_loss += torch.mean((target_features['conv4_2'] - content_features['conv4_2']) ** 2)  # 在特征域对比，也就是想让他们的纹理特征是一致的
    content_loss += torch.mean((target_features['conv5_2'] - content_features['conv5_2']) ** 2)

    loss_patch = 0
    img_proc = []
    for n in range(num_crops):
        target_crop = cropper(target)  # 进行随机裁剪
        target_crop = augment(target_crop)  # 随机视角变化
        img_proc.append(target_crop)  # 是一个数据增强后的列表

    img_proc = torch.cat(img_proc, dim=0)
    img_aug = img_proc

    image_features = clip_model.encode_image(clip_normalize(img_aug, device))  # 用clip编码了经过U-net和一系列数据增强变换的图像
    image_features /= (image_features.clone().norm(dim=-1, keepdim=True))  # 做归一化

    img_direction = (image_features - source_features)  # 对比原图和经过U-net数据增强的特征
    img_direction /= img_direction.clone().norm(dim=-1, keepdim=True)

    text_direction = (text_features - text_source).repeat(image_features.size(0), 1)  # 对比文本特征
    text_direction /= text_direction.norm(dim=-1, keepdim=True)
    loss_temp = (1 - torch.cosine_similarity(img_direction, text_direction, dim=1))  # 仿照clip做一个差值的余弦相似度
    loss_temp[loss_temp < args.thresh] = 0  # 相似度已经很高的我们就不惩罚了
    loss_patch += loss_temp.mean()

    glob_features = clip_model.encode_image(clip_normalize(target, device))  # 经过U-net的图像直接做clip编码
    glob_features /= (glob_features.clone().norm(dim=-1, keepdim=True))

    glob_direction = (glob_features - source_features)  # 相当于不是小patch，而是直接做
    glob_direction /= glob_direction.clone().norm(dim=-1, keepdim=True)

    loss_glob = (1 - torch.cosine_similarity(glob_direction, text_direction, dim=1)).mean()

    reg_tv = args.lambda_tv * get_image_prior_losses(target)
```

将所有损失函数汇总,更新优化U-net

```
    total_loss = args.lambda_patch * loss_patch + content_weight * content_loss + reg_tv + args.lambda_dir * loss_glob
    total_loss_epoch.append(total_loss)

    optimizer.zero_grad()
    total_loss.backward()
    optimizer.step()
```

4.环境搭建及实验

环境配置

在基础的深度学习框架下,还需要另外安装CLIP的api

```
pip install clip-anytorch
```

模型训练

本文的模型具有两种训练方式:

1. 输入单张图片训练,可以指定图片的风格,输出结果是单张风格迁移后的图片.
2. 输入是整批次的图片用于训练,可以对各种不同类型的图片进行风格迁移,具有很好的泛化能力,用于测试的同样是整批次的图片.

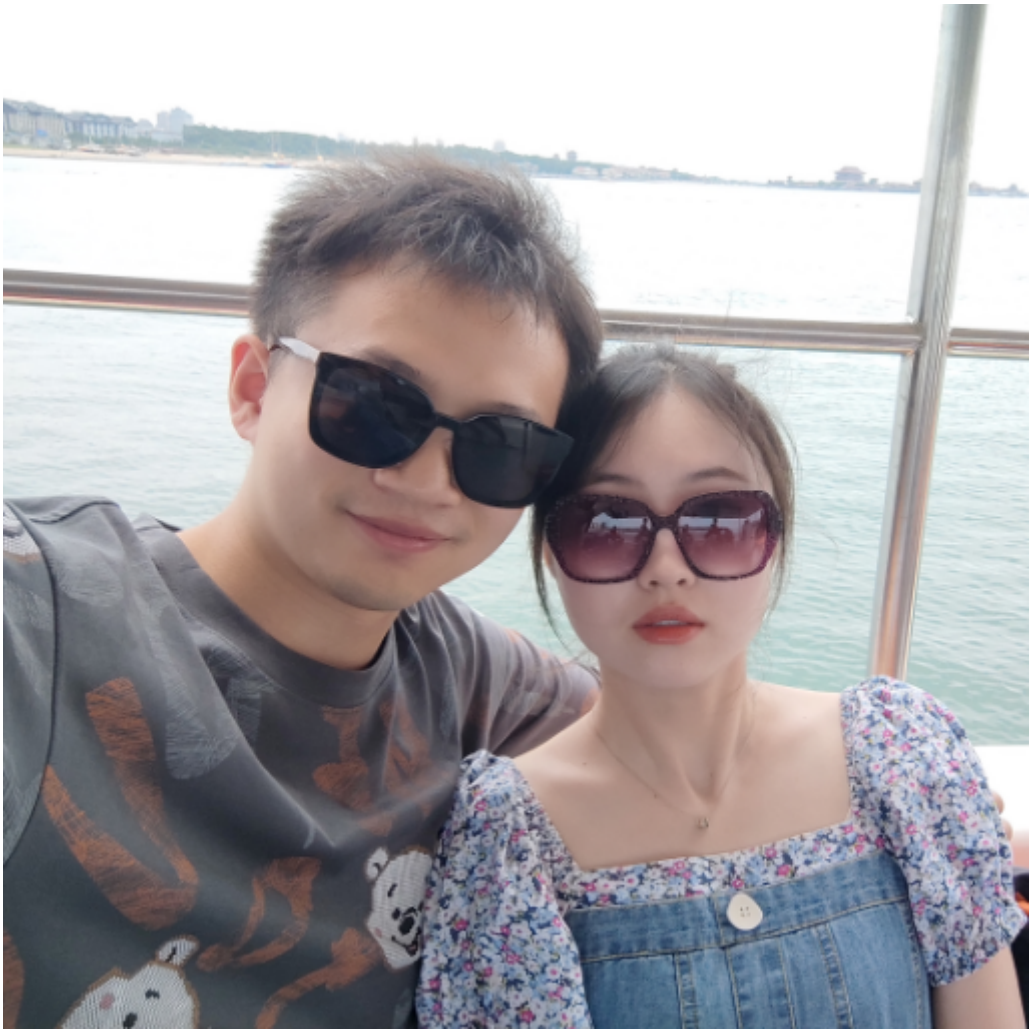
运行命令:

```
python .\train_CLIPstyler.py --content_path .\test_set\yam.png --content_name 'face' --exp_name 'exp3' --text 'Sketch with black pencil'
```

```
patch loss:  0.78173828125
dir loss:  0.37744140625
TV loss:  0.3836619555950165
After 180 criterions:
Total loss:  7561.49072265625
Content loss:  2.393179178237915
patch loss:  0.78076171875
dir loss:  0.34814453125
TV loss:  0.3884744346141815
After 200 criterions:
Total loss:  7523.158203125
Content loss:  2.255099296569824
patch loss:  0.77978515625
dir loss:  0.3291015625
TV loss:  0.39367741346359253
```

共迭代200轮,得到结果:

原图



风格迁移:text(Sketch with black pencil)

text(Mosaic)

text(Watercolor painting)