计算机科学与技术学院神经网络与深度学习课程实验报告

实验题目： Homework 4		学号：201900130024
日期：2021.11.5	班级： 数据 19	姓名： 刘士渤
Email：liuburger@qq.com		

实验目的：

完成 Style Transfer

实验软件和硬件环境：

VScode JupyterNoteBook

联想拯救者 Y7000p

实验原理和方法：

CNN

实验步骤：（不要求罗列完整源代码）

1. 补全 StyleTransfer-Pytorch.ipynb：

- content_loss：

  content_current 和 content_original 的形状是一样的，从哪个获取 shape 都行；

  F 是 current image 的特征矩阵，P 是 source image 的特征矩阵，二者都是 Cl×Ml 型，其中 Ml=Hl×Wl；

```
_,C_l,H_l,W_l=content_current.shape
F=content_current.view(C_l,H_l*W_l)# Ml=Hl*Wl
P=content_original.view(C_l,H_l*W_l)
```

  根据公式：

$$L_c = w_c \times \sum_{i,j}(F_{ij}^{\ell} - P_{ij}^{\ell})^2$$

  得到 loss，因为 content_current 和 content_original 都是 Pytorch 的张量，所以求和要用 torch.sum（）。

```
loss=content_weight*torch.sum((F-P)**2)
return loss
```

- gram_matrix：
  Fl 是一个这样的矩阵（其中其中 Ml=Hl×Wl）：

$$F^\ell \text{ of shape } (C_\ell, M_\ell),$$

而 feature 又是(N, C, H, W) 的矩阵，参数-1 可以让 F 的第三个维度=N·C·H·W/(N·C)=H·W:

```
N,C,H,W=features.shape
F=features.view(N,C,-1)# N,C,M
```

Fl 的转置是（Ml，Cl）的，所以调换 F 第二维和第三维：

```
F_T=F.permute(0,2,1)
```

根据 G 的定义：

$$G_{ij}^\ell = \sum_k F_{ik}^\ell F_{jk}^\ell$$

矩阵相乘：

```
gram=F.matmul(F_T)
```

如果 normalize 是 True，除以（H·W·C）：

```
if normalize:
    gram/=H*W*C
return gram
```

- style_loss：
  style_layers 是 feats 的索引，feats 是图像每一层的特征，feats[style_layers[i]]得到图像某一层的特征；
  需要将 feats[style_layers[i]]转化为 gram matrix，style_targets 本身就是 gram matrix，所以不需要转化；
  根据公式：

$$L_s^\ell = w_\ell \sum_{i,j} \left(G_{ij}^\ell - A_{ij}^\ell\right)^2$$

  得出 loss：

```
loss=0.
for i in range(len(style_layers)):
  loss+=style_weights[i]*torch.sum((gram_matrix(feats[style_layers[i]])-style_targets[i])**2)

return loss
```

- tv_loss：
  根据公式：

$$L_{tv} = w_t \times \left(\sum_{c=1}^{3}\sum_{i=1}^{H-1}\sum_{j=1}^{W}(x_{i+1,j,c} - x_{i,j,c})^2 + \sum_{c=1}^{3}\sum_{i=1}^{H}\sum_{j=1}^{W-1}(x_{i,j+1,c} - x_{i,j,c})^2\right)$$

  第一项是在 H 维度上后面减前面，第二项是在 W 维度上后面减前面；
  所以被减数是[1,H)（或[1,W)），减数是[0,H-1)（或[0,W-1)）；
  得出 loss：

```
tv1=torch.sum((img[:,:,1:,:]-img[:,:,:-1,:])**2)
tv2=torch.sum((img[:,:,:,1:]-img[:,:,:,:-1])**2)
loss=tv_weight*(tv1+tv2)
return loss
```

结论分析与体会：

1. 没有随机初始化的星空风格：

随机初始化的星空风格：

可以看出：没有随机初始化的图片更接近星空的风格，随机初始化的图片更接近原图片。

2. 除了生成样例的图像外，我还生成了自己的头像的风格迁移：

Content Source Img.

Style Source Img.

Style Source Img.

Style Source Img.

Style Source Img.

就实验过程中遇到和出现的问题，你是如何解决和处理的，自拟 1－3 道问答题：

1. 如果选择 GPU 加速、没有安装 cuda 的话，会报这样的错：

```
RuntimeError: Cannot initialize CUDA without ATen_cuda library. PyTorch splits its backend into
nd a CUDA library; this error has occurred because you are trying to use some CUDA functionalit
oaded by the dynamic linker for some reason.  The CUDA library MUST be loaded, EVEN IF you don'
DA library! One common culprit is a lack of -INCLUDE:?warp_size@cuda@at@@YAHXZ in your link arg
te dynamic library dependencies if you don't depend on any of their symbols.  You can check if
r binary to see if there is a dependency on *_cuda.dll library.
```

选择 CPU 加速不会报错，但是生成图像会非常慢，所以我最终还是选择了 GPU 加速。

使用 conda install pytorch torchvision torchaudio cudatoolkit=10.2 -c pytorch 命令将 pytorch 安装到 python 为 3.6 的虚拟环境，然后选择 GPU 加速就不会报错了。

GPU 加速效果十分显著，之前运行 20min 还没完成的图片，加速后 2 秒就可以。

2. 原本以为参数中的 size 是图像的边长，并且要求图像必须是正方形。

```
params1 = {
    'content_image' : 'styles/tubingen.jpg',
    'style_image' : 'styles/composition_vii.jpg',
    'image_size' : 192,
    'style_size' : 512,
    'content_layer' : 3,
    'content_weight' : 5e-2,
    'style_layers' : (1, 4, 6, 7),
    'style_weights' : (20000, 500, 12, 1),
    'tv_weight' : 5e-2
}
```

但查看图像属性之后发现图像的宽高并不是 size 中的大小，也不是正方形，这才放心地生成自己的图片。

content-weight 越小，style 越明显；当然 style-weights 越大，style 也越明显。