姓名

- 张进华

学号班级

- 智能19 201900150221

实验日期

- 2021-11-4

实验题目 Style Transfer

> 实现使用卷积神经网络的图像风格迁移，一般的想法是拍摄两个图像，并产生一个反映一个内容但另一个艺术“风格”的新图像。首先制定一个损失函数来匹配深度网络特征空间中每个图像的内容和风格，然后对图像本身的像素执行梯度下降。用作特征提取器的深度网络是 `SqueezeNet`，这是一个在 `ImageNet` 上训练过的小模型，体积小且效率高

实验步骤

步骤一 Content Loss

通过将损失函数组合生成一张反映一张图的内容和另一张图的风格的图片，惩罚图片内容的偏移和风格的偏移，然后使用这个混合损失函数进行梯度下降，不是对模型的参数，而是对源图的像素值
内容损失评估了生成图像的特征图与源图像的特征图有多大区别，只关心网络其中一层的内容表达，将对重构空间为一维的特征图进行运算，计算内容损失部分在整体损失中的损失权值。
内容损失计算公式：

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

内容损失实现关键代码如下：

```
def content_loss(content_weight, content_current, content_original):
    # Compute the content loss for style transfer.
    temp = (content_current - content_original) ** 2
    content_loss = content_weight * torch.sum(temp)
    return content_loss
```

测试内容损失显示结果如下：

```
]  ✓ 0.1s

   Maximum error is 0.000
```

步骤二 Style Loss

首先，计算一个用于表示每个滤波器的相关程度的Gram矩阵G，其中F与之前一样。Gram矩阵是一个协 方差矩阵的近似，我们希望生成图片的激励状态与风格图片的激励状态相匹配，使(近似)协方差相匹配是 其中的一种方法。你能用很多种方法完成，但Gram矩阵很好计算且在实践中有一个好结果，所以它很 不错。 给定特征图形状为，Gram矩阵形状是， 它的项为:

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

假设 Gℓ是由当前图片的特征图算得的Gram矩阵， Aℓ是源图片的特征图算得的Gram矩阵， wℓ是可调权重，那么 ℓ层的风格损失可以用两Gram矩阵的加权欧几里得距离表示:

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

通常我们会计算一组层 LL的风格损失而不是单独一层 ℓ；完整风格损失是每一层风格损失的和:

$$L_s = \sum_{\ell \in \mathcal{L}} L_s^{\ell}$$

计算Gram矩阵关键代码如下:

```
def gram_matrix(features, normalize=True):
    """
    Compute the Gram matrix from features.
    """
    shape = features.size()
    features = features.view([shape[0],shape[1],-1])

    transpose_features = features.clone()

    transpose_features = transpose_features.permute(0,2,1)

    result = torch.matmul(features, transpose_features)
    if normalize:
        result = result / (shape[0]*shape[1]*shape[2]*shape[3])
    return result
```

显示结果如下:

```
    8
    9  gram_matrix_test(answers['gm_out'])
[8] ✓ 0.5s

</> Maximum error is 0.000
```

然后实现style_loss 函数，关键代码如下:

```
def style_loss(feats, style_layers, style_targets, style_weights):
    """
    Computes the style loss at a set of layers.
    """
    style_losses = 0
    for i in range(len(style_layers)):
        idx = style_layers[i]
        style_losses += content_loss(style_weights[i],
                                     gram_matrix(feats[idx]),
                                     style_targets[i])
    return style_losses
```

显示结果如下:

```
19
20  style_loss_test(answers['sl_out'])
✓ 0.1s
```

```
Error is 0.000
```

步骤三 Total-variation regularization

实际上对图像的平滑是有帮助的。我们可以通过在损失函数中添加一个用于处理像素值毛刺或者说整体偏差的项来实现。 你可以通过求每个像素与其周围相邻(水平或垂直)像素的差的平方和的累计值计算整体偏差。在这里我们 求3个输入通道(RGB)的整体方差正则化的和，并用参数对其加权。

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

TV损失的定义实现关键代码如下:

```
def tv_loss(img, tv_weight):
    """
    Compute total variation loss.
    """
    shape = img.size()
    row_cur = img[:, :, :-1, :]
    row_lat = img[:, :, 1:, :]
    col_cur = img[:, :, :, :-1]
    col_lat = img[:, :, :, 1:]
    row_result = row_lat - row_cur
    col_result = col_lat - col_cur
    row_result = row_result * row_result
    col_result = col_result * col_result
    result = tv_weight * (torch.sum(row_result) + torch.sum(col_result))
    return result
```

显示结果如下:

```
11
12  tv_loss_test(answers['tv_out'])
```

[12] ✓ 0.6s

```
Error is 0.000
```

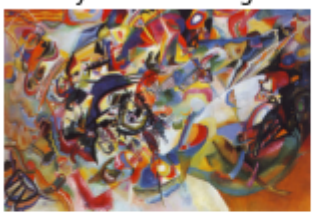
步骤五 Generate some pretty pictures

实现效果如下：

第一组

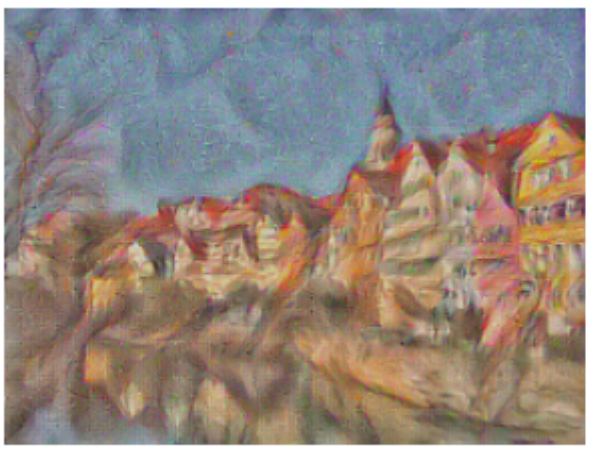
Content Source Img.

Style Source Img.



```
Iteration 0
```

```
Iteration 100
```



Iteration 199

** 第二组**

</>

Content Source Img.

Style Source Img.

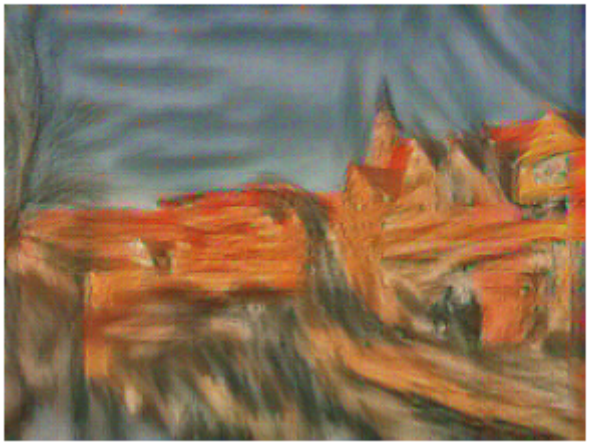
```
Iteration 0
```

</>

```
Iteration 100
```

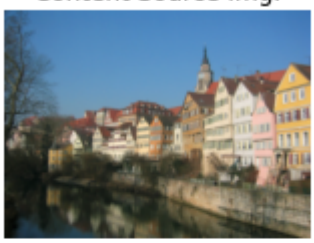
</>

Iteration 199



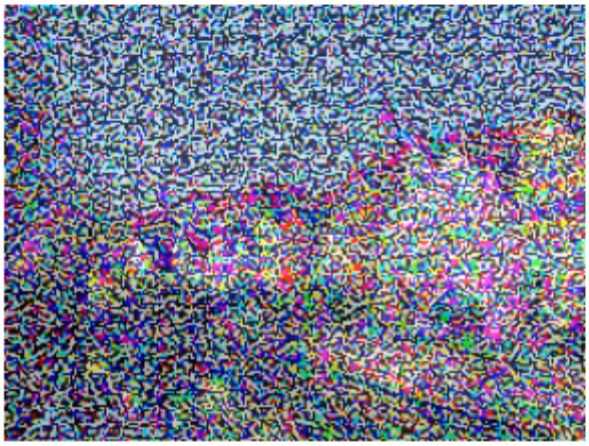
** 第三组**

Content Source Img.

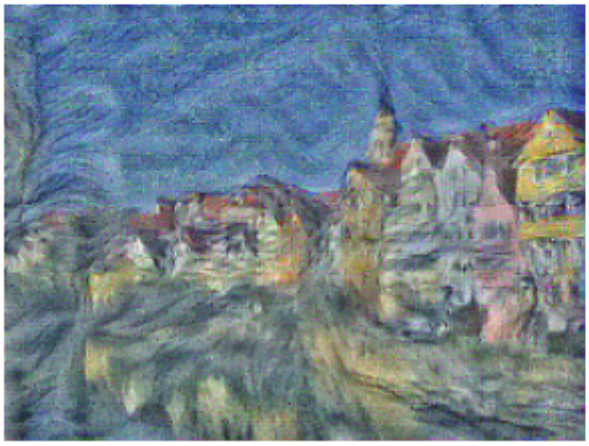


Style Source Img.

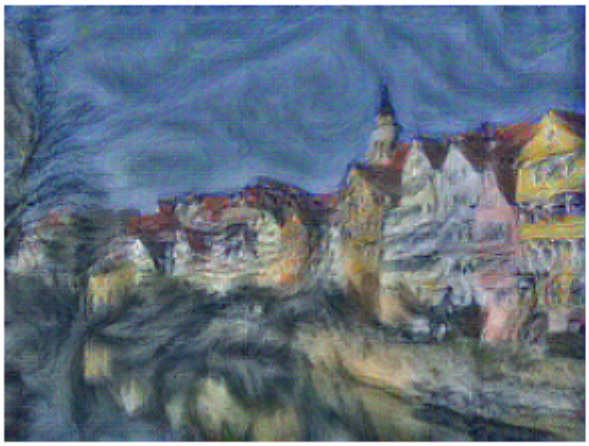
```
Iteration 0
```



```
Iteration 100
```



```
Iteration 199
```

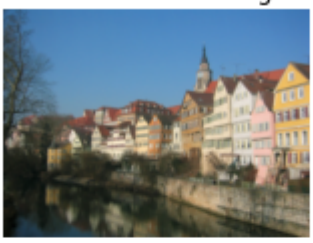


步骤六 Feature Inversion

为了理解卷积神经网络学习识别的特征的类型，尝试通过图像的特征表达重构一张图片。可以通过在预训练网络中做图像梯度实现,设置特征权值为 0，并用随机噪点取代内容源图像进行初始化初始图像，可通过图像的特征表达重构一张内容源图像。

实现效果如下：

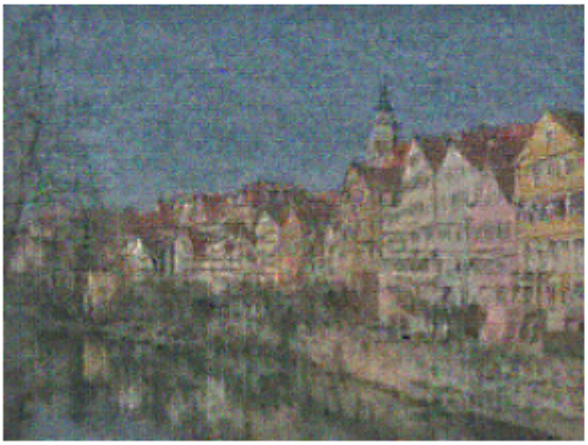
Content Source Img.



Style Source Img.

```
Iteration 0
```

```
Iteration 100
```



```
Iteration 199
```

结论分析与体会：

在本次实验中，学习使用了预训练好的模型进行图像风格转换，学习了由三个部分组 合的损失函数的计算。

损失函数分为三个部分的和： ①内容损失 ②风格损失 ③整体多样性损失。

内容损失部分负责使输出图像的 content 更加接近输入的 content image，风格损失部 分负责使输出图像的 style 更加接近输入的 style image。同时我们通过加入正则化项对图像 进行了平滑处理操作，了解了如何利用深度学习神经网络进行图像风格转化，知道了损失函 数如何设置以及每个部分的作用，收获很大。