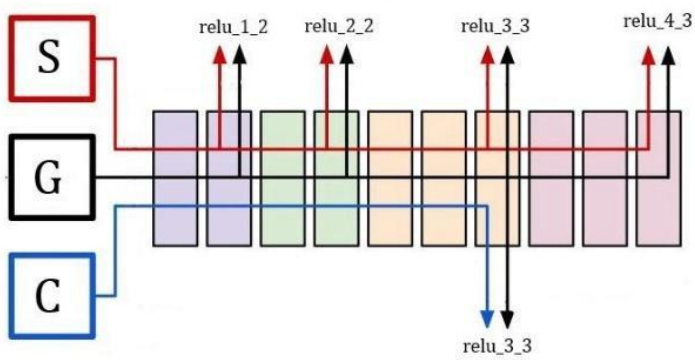


# 计算机科学与技术学院神经网络与深度学习课程实验报告

实验题目：风格迁移		学号：201600301148
日期：4.12	班级：人工智能	姓名：周阳
Email： <a href="mailto:862077860@qq.com">862077860@qq.com</a>		
实验目的： 1. 使用 pytorch 实现 neural style transfer		
实验软件和硬件环境：  CPU：英特尔至强 E5 GPU：NVIDIA GeForce 1060 6G 内存：16G Pycharm Python 3.6		
实验原理和方法： <b>1，风格迁移</b> 一层层的卷积神经网络将图像最原始的像素特征逐渐变成“高层”表示，以方便分类器分类。 这种操作是可以通过，反卷积，反池化等操作还原的。 我们想让一张图像同时拥有原图的内容和风格图像的风格。 从高层特征角度，我们希望我们生成的高层特征的线条，框架与原图相似，但是颜色纹理与风格图像相似。 <b>2，怎样达到这样的效果</b> 定义内容相似：高层表示 feature map(激活层输出)相似		
 <p>通过 VGG 进行特征提取，S 表示 style 图像，G 表示生成图像，C 代表 content 图像</p>		
$L_{\text{content}}(G, C) = \frac{1}{2} \sum_{ij} (G_{ij}^l - C_{ij}^l)^2$		
通过上面这个公式，就可以限制生成图像的内容与内容图像相似了。		

为什么？可能是因为，对于 VGG 来说，这两张图像（G，C）的图像表示相似，代表：这两个图对于 VGG 来说还是一个东西，保证了“内容”的相同。

定义风格：使用 **Gram 矩阵**

$$G_{ij} = \sum_K^{C \times C} F_{ik} F_{jk}$$

Gram 矩阵定义了：通道之间的相关度。可以认为 VGG 的某一层的输出  $c \times w \times h$  的矩阵， $c$  为通道数，他们被激活一定程度上是因为图中同时含有两个通道锁包含的东西。

所以风格被定义为：

$$E_l(G, S) = \frac{1}{4N_l^2 M_l^2} \sum_{ij} (G_{ij}^l - S_{ij}^l)^2$$

$$L_{style}(G, S) = \sum_{ij} (w_l - E_l)^2$$

其中 G，S 都是 gram 矩阵上的相似。

最终 loss

$$L(G, S, C) = \alpha * L_{content}(G, C) + \beta * L_{style}(G, S)$$

实验步骤：（不要求罗列完整源代码）

1，VGG 与 featuremap

```
1 # 定义VGG模型，前向时抽取0, 5, 10, 19, 28层卷积特征
2 class VGGNet(nn.Module):
3     def __init__(self):
4         """Select convl_1 ~ conv5_1 activation maps."""
5         super(VGGNet, self).__init__()
6         self.select = ['0', '5', '10', '19', '28']
7         self.vgg = models.vgg19(pretrained=True).features
8
9     def forward(self, x):
10         """Extract 5 conv activation maps from an input image.
11
12         Args:
13             x: 4D tensor of shape (1, 3, height, width).
14
15         Returns:
16             features: a list containing 5 conv activation maps.
17         """
18         features = []
19         for name, layer in self.vgg._modules.items():
20             x = layer(x) # 依次输出并且截取
21             if name in self.select:
22                 features.append(x)
23         return features
```

2，loss

```
#####
content_loss += torch.mean((f1 - f2)**2)
#####

#####
style_loss += torch.mean((f1 - f3)**2) / (featrue_channel*feature_height * feature_width)
# 两个 gram 矩阵之间的距离为loss 值
#####
```

3，结果



原始图像，风格图像，风格迁移结果

结论分析与体会：

1，风格迁移中的 gram 矩阵定义非常具有借鉴意义。想到使用 gram 矩阵代表图像的 style 确实是一个非常有趣的奇思妙想。

2，neural style 也让我看到了视觉中的问题的直观性，有趣的地方。

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，gram 矩阵的注意点

答：

**1 Gram** 矩阵的计算采用了累加的形式，抛弃了空间信息。一张图片的像素随机打乱之后计算得到的 **Gram Matrix** 和原图的 **Gram Matrix** 一样。所以认为 **Gram Matrix** 所以认为 **Gram Matrix** 抛弃了元素之间的空间信息。

**2 Gram Matrix** 的结果与 **feature maps F** 的尺寸无关，只与通道个数有关，无论 **H,W** 的大小如何，最后 **Gram Matrix** 的形状都是 **CXC**

**3** 对于一个 **C x H x W** 的 feature maps，可以通过调整形状和矩阵乘法运算快速计算它的 **Gram Matrix**。即先将 **F** 调整到 **C X (HW)** 的二维矩阵，然后再计算 **F** 和 **F** 的转置。结果就为 **Gram Matrix**