# 深度学习：homework5

张驰　　前沿交叉学科研究院

2019 年 5 月 26 日

## 1　背景知识：利用深度神经网络进行纹理合成

纹理合成可视作利用随机噪声 $\hat{x}$ 来近似真实纹理图片的过程，对 $\hat{x}$ 进行训练和优化，直到从深度神经网络中提取的特征具有与源纹理图像 $x$ 相同的统计量，由于纹理的每个定义都是固定的，所以特征的统计数据应该与空间信息无关。在本实验中，我们使用 Gram 矩阵来描述源纹理图像和生成纹理图像的空间不可知统计量。将 Gram 矩阵定义为特征不同通道之间的相关性，其公式为：

$$G_{ij}^{l} = \sum_{m,n} F_{i,m,n}^{l} F_{j,m,n}^{l} \tag{1}$$

$F_{i,m,n}^{l}$ 表示第 l 层，第 i 个特征图的 (m,n) 像素点。这个矩阵的含义是同一层级 conv 的不同特征之间的像素值点乘然后累加，得到的是每两个不同特征图之间的相关系数值。

相应地，构建损失函数为

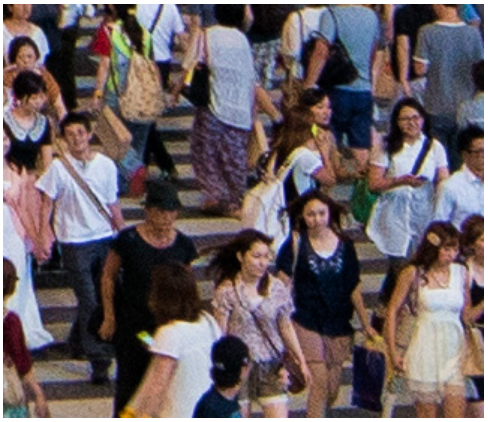
$$Loss = \sum_{l} w_l \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G_{ij}^{l} - \widehat{G_{ij}^{l}})^2 \tag{2}$$

$G\ \widehat{G}$ 表示纹理图像与生成图像的 Gram 矩阵，$N_l, M_l$ 分别表示第 l 层特征图的通道数和对应的每张特征图的像素总数，$w_l$ 表示第 l 层的权重 (在 Baseline 代码中，我们对所有层都设置为 1)。

公式中除以 M 方 N 方是保证 loss 对每个像素的改变与像素值在同一个数量级上。图 1 是实验中测试的 4 张图片：

(a) red-peppers256

(b) robot



(c) shibuya

(d) stone

图 1: 实验测试图片

# 2 实现 Gram 矩阵和 Loss 函数

## 2.1 实验准备

实验一中要求我们实现 Gram 矩阵和 Loss 函数的代码，相应的代码如下：

```
def get_l2_gram_loss_for_layer(noise, source, layer):
  noise_layer = getattr(noise, layer)
  source_layer = getattr(source, layer)

  n_filter=noise_layer.shape[-1].value
  h_filter=noise_layer.shape[1].value
  w_filter=noise_layer.shape[2].value

  re_noise = tf.reshape(noise_layer, shape=(-1, n_filter))
```

```
    tran_noise = tf.transpose(tf.reshape(noise_layer, shape=(-1, n_filter)))
    Matrix_noise = tf.matmul(tran_noise, re_noise)

    re_source = tf.reshape(source_layer, shape=(-1, n_filter))
    tran_source_ = tf.transpose(tf.reshape(source_layer, shape=(-1, n_filter)))
    Matrix_source = tf.matmul(tran_source_, re_source)

    diff_G = tf.reduce_sum(tf.square(Matrix_noise - Matrix_source))

    weight_filter = 1.0
    # loss = weight_filter * diff_G / tf.cast(4 * (n_filter ** 2) * (h_filter * w_filter) ** 2,
                                              dtype=tf.float32)
    loss = weight_filter * diff_G /  tf.cast(4 * (h_filter * w_filter) ** 2, dtype=tf.float32)
return loss
```

注意到，这里我将论文要求的 loss 函数进行了修改，没有除以 N 方，只相应地除了 M 方，得到的结果最好，具体的演示将在下一节介绍。

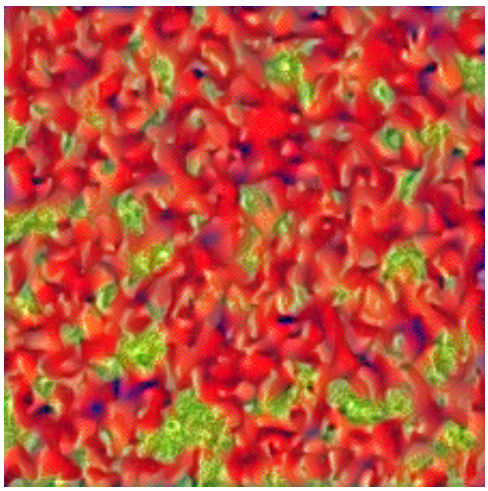
## 2.2 实验分析

实验运行结果，图 2 是没有除以 N 方的红辣椒图和石头图:

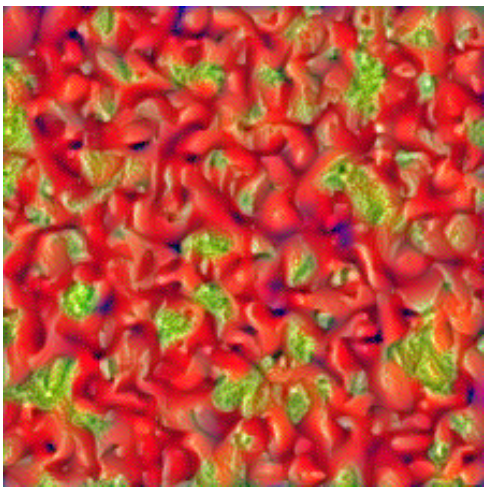
(a) 红辣椒　　(b) 石头

图 2: 未添加 N 方的最佳生成图片

根据实验结果可以发现生成的新图片与原图是比较相似的，验证了代码的正确性。但由于修改了 loss 函数的表达式，我们相应的测试一下修改分母的其它项时，结果会发生什么变化，结果如图 3 所示：



(a) 未添加 M 方红辣椒



(b) 未添加 M 方 N 方红辣椒

图 3: 修改分母其它项时最佳生成图片

对比结果可以看出，修改分母其它项时，生成的图片效果都比较差，只要当分母去掉 N 方的时候，效果是最好的。就该情况提出为什么论文中未去掉 N 方，而本实验却不得不去掉 N 方的原因：可能是原论文中输入没有作归一化，即像素范围在 0-255 之间，而本实验做了归一化，像素范围在 0-1 之间，因此结果有差异，但若如此，则分母不一定是除 N 方来保证 loss 的梯度与像素 x 在同一数量级，具体原因尚且未研究出来。下面的实验暂且以去掉 N 方的代码作为 Baseline。

# 3 测试非纹理图片的效果

## 3.1 实验准备

非纹理图片为机器人和 shibuya 图片，它们和上一个实验使用的图片明显不同的是，没有局部纹理信息，整体非常光滑。

## 3.2 实验分析

实验结果如图 2 所示：

(a) 机器人

(b) shibuya

图 4: 未添加 N 方的最佳生成图片

由实验结果可以判断，对于非纹理图片，模型生成的过程中会将特征信息打散然后任意组合，出现以上情况，由此可判断本模型适用于有纹理的照片。

# 4 用更少的卷积层来测试

## 4.1 实验准备

在前面的实验中，我们使用的是所有卷积层来计算 Gram 矩阵的损失函数，包含有大量的参数，前后信息呼应，效果显著。在本次实验中我们将尝试更少的卷积层对结果会有怎样的影响，并找到效果十分显著的一种选择，这里我将测试以下三种情况：

- 1. 保证当前卷积层之前的卷积层至少有一个存在的前提下，减少卷积层的个数。
- 2. 只保留前几个卷积层，后面的全舍弃。
- 3. 只保留后面的卷积层，前面的全舍弃。

对应的代码为：

```
GRAM_LAYERS= ['conv1_1', 'conv2_1', 'conv3_1', 'conv4_1' , 'conv5_1']
GRAM_LAYERS= ['conv1_1', 'conv1_2' ,'conv2_1', 'conv2_2']
GRAM_LAYERS= ['conv4_1', 'conv4_2', 'conv4_3' , 'conv5_1', 'conv5_2', 'conv5_3']
```

该实验的目的是探索浅层卷积层和深层卷积层对模型生成图片结果的影响，以及分别对结果影响的程度有多少。

### 4.2 实验分析

实验结果如图 2 所示：

(a) 每层保留一个卷积层　(b) 只保留浅层卷积　(c) 只保留深层卷积

图 5: 不同方案实验结果

根据以上实验结果，可得出：1. 保证每层至少保留一个卷积层的实验得到的结果基本和 Baseline 方法差别不大。2. 只保留浅层卷积的实验结果有纹理和局部信息。3. 只保留深层卷积层的实验结果能得到明显的轮廓，但石头的纹理效果很差。

因此，我们可以推测深度卷积层获取的是局部细节信息 (石头的轮廓)，浅层获取的是纹理的抽象信息 (石头的纹理)。仅仅依靠浅层或者深层的卷积层的 GRAM 矩阵计算，都不能对结果有较明显的改进，以上实验 2,3 的结果是很直观的例子，必须要保证 Gram 矩阵既有深层卷积的轮廓细节信息，又有浅层卷积的纹理抽象信息，结合起来才能生成类似原图的效果。对应地方案 1 中每一层都只取一个卷积层（5 层）的效果基本和所有 13 层的结果差不多，验证了我们的猜想。

## 5 寻找 Gram 矩阵的替代方式

### 5.1 实验准备

前面的实验中，我们都是依靠 Gram 矩阵来进行实验，Gram 矩阵可以看做 feature 之间的偏心协方差矩阵（即没有减去均值的协方差矩阵，但是在我们的实验中减去了），在 feature map 中，每个数字都来自于一个特定滤波器在特定位置的卷积，因此每个数字代表一个特征的强度，而 Gram 计算的实际上是两两特征之间的相关性，哪两个特征是同时出现的，哪两个是此消彼长的等等，同时，Gram 的对角线元素，还体现了每个特征在图像中出现的量，因此，Gram 有助于把握整个图像的大体风格。有了表示风格的 Gram Matrix，要度量两个图像风格的差异，只需比较他们 Gram Matrix 的差异即可。

本次实验尝试使用生成图片和原图片特征的 Earth mover's distance 来度量，EMD 为归一化的从一个分布变为另一个分布的最小代价, 可以用来测量两个分布 (multi-dimensional distributions) 之间的距离。其公式表达为：

$$Loss = \sum_{l} w_l \sum_{i} (sorted(F_i) - sorted(\hat{F}_i))^2 \tag{3}$$

对应的修改的代码部分如下所示：

```
def get_l2_EMD_loss_for_layer(noise, source, layer):
    noise_layer = getattr(noise, layer)
    source_layer = getattr(source, layer)

    n_filter = noise_layer.shape[-1].value
    h_filter = noise_layer.shape[1].value
    w_filter = noise_layer.shape[2].value

    tran_noise = tf.transpose(tf.reshape(noise_layer, shape=(-1, n_filter)))
    tran_source_ = tf.transpose(tf.reshape(source_layer, shape=(-1, n_filter)))
    feature_noise=tf.nn.top_k(tran_noise,h_filter*w_filter)[0]
    feature_source=tf.nn.top_k(tran_source_,h_filter*w_filter)[0]

    feature_sum = tf.reduce_sum(tf.square(feature_noise - feature_source))
    weight = 1.0
    loss = weight * feature_sum
    return loss
```

## 5.2 实验分析

实验结果如图 6 所示：

(a) EMD 红辣椒　　(b) EMD 石头

图 6: 利用 EMD 来代替 Gram 矩阵生成图片

对比实验结果可以发现，EMD 方法基本上也能实现和原图片类似的样子，但是整体效果上，即图片的清晰度稍微比 Gram 矩阵得到的结果差，但是效果依然可观。

# 6 改变不同层的 Gram 矩阵的权重

## 6.1 实验准备

在前面的实验中，我们设置不同层的 w 为 1，即每层的权重相同。在本实验中，我们将按以下方式改变权重，来观察实验结果有何变化。

- 1. 权重逐渐增大。
- 2. 权重逐渐减小。

对应的修改的代码部分如下所示：

```
    weight_filter = 1.0/h_filter  #M逐渐减少，即越后面权重越大
weight_filter = 1.0/n_filter #N逐渐增大，即越后面权重越小
```

## 6.2 实验分析

实验结果如图 7 所示：

(a) 后面层权重增加　　(b) 后面层权重减少

图 7: 不同权重改变方式生成的图片

对比实验结果可以发现，后面层权重逐渐得到的结果 (图 a) 比后面层权重逐渐减少得到的图片 (图 b) 效果更好，更立体，更真实，轮廓更分明一点。图 b 由于浅层卷积层权重更大，导致深层的局部细节信息不够清晰，更侧重整体结构一点。因此可以得出结论，深层的卷积层对图像真实性的贡献更大一点。

# 参考文献


[1] Gatys L A , Ecker A S , Bethge M . Texture Synthesis Using Convolutional Neural Networks[J]. 2015.

[2] https://blog.csdn.net/cicibabe/article/details/70991588

[3] https://blog.csdn.net/wangyang20170901/article/details/79037867/