



deeplearning.ai

吴恩达 DeepLearning.ai 课程提炼笔记 (4-4) 卷积神经网络 -- - 人脸识别和神经风格迁移

Ng卷积神经网络部分，完结撒花！以下为在Coursera上吴恩达老师的 deeplearning.ai 课程项目中，第四部分《卷积神经网络》第四周课程“**特殊应用：人脸识别和神经风格迁移**”关键点的笔记。本次笔记几乎涵盖了所有视频课程的内容。在阅读以下笔记的同时，强烈建议学习吴恩达老师的视频课程，视频请至 Coursera 或者 网易云课堂。

Part 1：人脸识别

1. 人脸验证和人脸识别

人脸验证 (Verification) :

- Input: 图片、名字/ID;
- Output: 输入的图片是否是对应的人;
- 1 to 1 问题。

人脸识别 (Recognition) :

- 拥有一个具有K个人的数据库;
- 输入一副人脸图片;
- 如果图片是任意这K个人中的一位，则输出对应人的ID。

人脸识别问题对于人脸验证问题来说，具有更高的难度。如对于一个验证系统来说，如果我们拥有 **99%** 的精确度，那么这个验证系统已经具有了很高的精度；但是假设在另外一个识别系统中，如果我们把这个验证系统应用在具有K个人的识别系统中，那么系统犯错误的机会就变成了K倍。所以如果我们想在识别系统中得到更高的精度，那么就需要得到一个具有更高精度的验证系统。

2. one shot learning

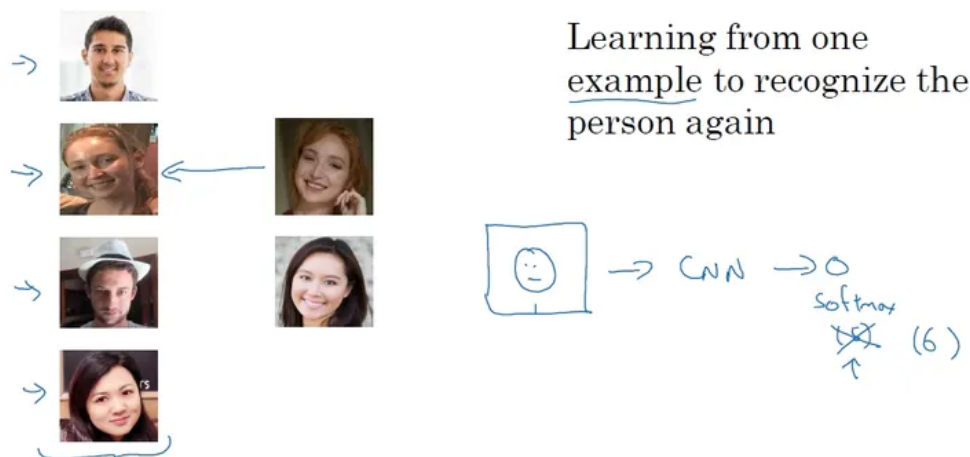
对于大多数的人脸识别系统都存在的一个问题就是one shot learning。



什么是 one shot learning:

对于一个人脸识别系统，我们需要仅仅通过先前的一张人脸的图片或者说一个人脸的样例，就能够实现该人的识别，那么这样的问题就是 one shot 问题。对于存在于数据库中的人脸图片，系统能够识别到对应的人；而不在数据库中的人脸图片，则系统给出无法通过识别的结果。

One-shot learning



对于one shot learning 问题，因为只有单个样本，是不足以训练一个稳健的卷积神经网络来进行不同人的识别过程。而且，在有新的样本成员加入的时候，往往还需要对网络进行重新训练。所以我们不能以传统的方法来实现识别系统。

Similarity 函数:

为了能够让人脸识别系统实现一次学习，需要让神经网络学习 **Similarity** 函数:

- $d(img1, img2)$: 两幅图片之间的差异度
- 输入: 两幅图片
- 输出: 两者之间的差异度
- 如果 $d(img1, img2) \leq \tau$, 则输出 "same" ;

如果 $d(img1, img2) > \tau$, 则输出 "different" .

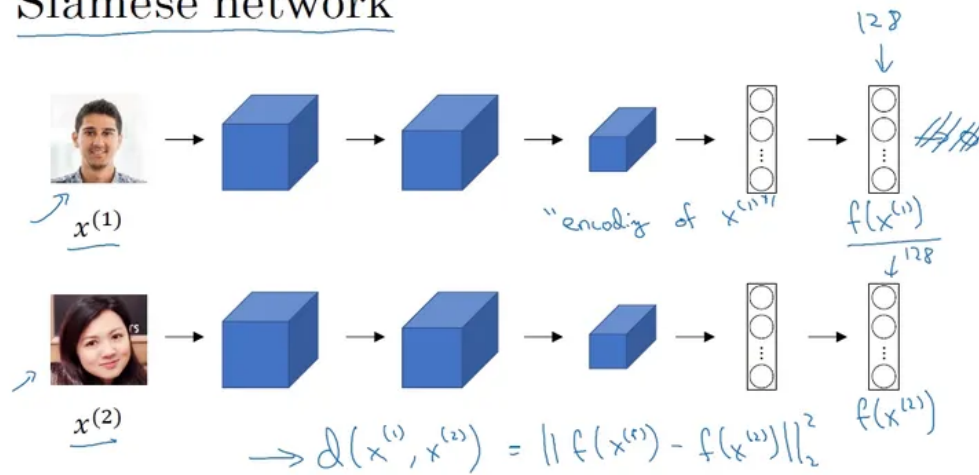
对于人脸识别系统，通过将输入的人脸图片与数据库中所拥有的图片成对输入Similarity函数，两两对比，则可解决one shot problem。如果有新的人加入团队，则只需将其图片添加至数据库即可。

3. Siamese 网络

利用Siamese 网络来实现 Similarity 函数。

构建网络:

Siamese network



对于一个卷积神经网络结构，我们去掉最后的softmax层，将图片样本1输入网络，最后由网络输出一个N维的向量（图中实例以128表示），这N维向量则代表输入图片样本1的编码。将不同人的图片样本输入相同参数的网络结构，得到各自相应的图片编码。

Similarity 函数实现：

将Similarity 函数表示成两幅图片编码之差的范数：

$$d(x_1, x_2) = ||f(x_1) - f(x_2)||_2^2$$

那么也就是说：

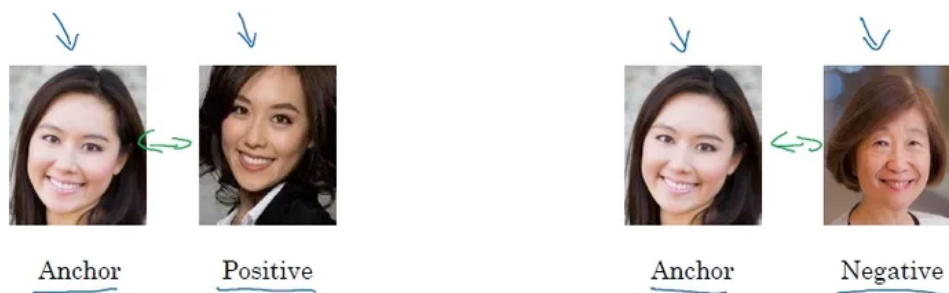
- 我们的神经网络的参数定义了图片的编码；
 - 学习网络的参数，使我们得到好的Similarity 函数：
- 如果 x_1, x_2 是同一个人的图片，那么得到的 $||f(x_1) - f(x_2)||^2$ 很小；
 - 如果 x_1, x_2 不是同一个人的图片，那么得到的 $||f(x_1) - f(x_2)||^2$ 很大。

4. Triplet 损失

如何通过学习神经网络的参数，得到优质的人脸图片的编码？方法之一就是定义 Triplet 损失函数，并在其之上运用梯度下降。

学习目标：

为了使用Triplet 损失函数，我们需要比较成对的图像（三元组术语）：



- Anchor (A) : 目标图片；
- Positive (P) : 与Anchor 属于同一个人的图片；
- Negative (N) : 与Anchor不属于同一个人的图片。

对于Anchor 和 Positive，我们希望二者编码的差异小一些；对于Anchor 和Negative，我们希望他们编码的差异大一些。所以我们的目标以编码差的范数来表示为：

$$d(A, P) = \|f(A) - f(P)\|^2 \leq \|f(A) - f(N)\|^2 = d(A, N)$$

也就是：

$$\|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 \leq 0$$

上面的公式存在一个问题就是，当 $f(A) = f(P) = f(N) = 0$ 时，也就是神经网络学习到的函数总是输出0时，或者 $f(A) = f(P) = f(N)$ 时，也满足上面的公式，但却不是我们想要的目标结果。所以为了防止出现这种情况，我们对上式进行修改，使得两者差要小于一个较小的负数：

$$\|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 \leq -\alpha$$

一般将 α 写成 $+\alpha$ ，称为“margin”，即：

$$\|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 + \alpha \leq 0$$

不同 margin 值的设置对模型学习具有不同的效果，margin 的作用就是拉大了 Anchor与 Positive 图片对 和 Anchor与Negative 图片对之间的差距。

Triplet 损失函数：

Triplet 损失函数的定义基于三张图片：Anchor、Positive、Negative。

$$L(A, P, N) = \max(\|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 + \alpha, 0)$$

整个网络的代价函数：

$$J = \sum_{i=1}^m L(A^{(i)}, P^{(i)}, N^{(i)})$$

假设我们有一个10000张片的训练集，里面是1000个不同的人的照片样本。我们需要做的就是从这10000张训练集中抽取图片生成 (A,P,N) 的三元组，来训练我们的学习算法，并在Triplet 损失函数上进行梯度下降。

注意：为了训练我们的网络，我们必须拥有Anchor和Positive对，所以这里我们必须有每个人的多张照片，而不能仅仅是一张照片，否则无法训练网络。

三元组 (A,P,N) 的选择：

在训练的过程中，如果我们随机地选择图片构成三元组 (A,P,N)，那么对于下面的条件是很容易满足的：

$$d(A, P) + \alpha \leq d(A, N)$$

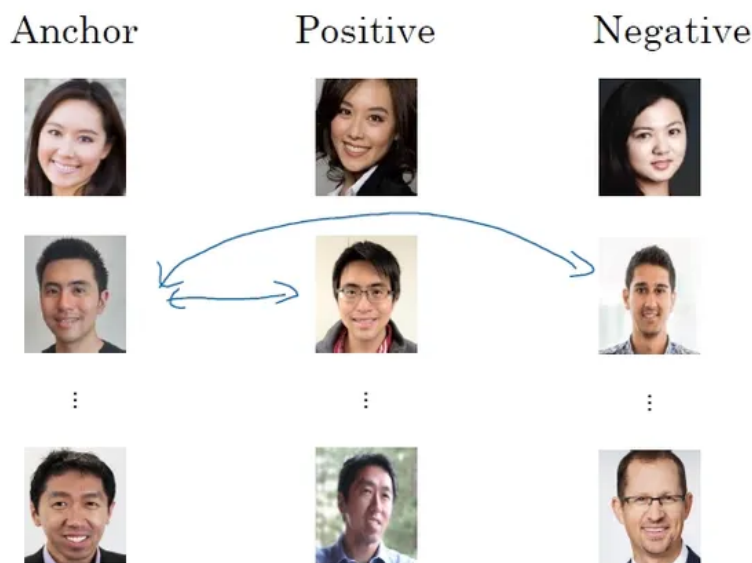
所以，为了更好地训练网络，我们需要选择那些训练有“难度”的三元组，也就是选择的三元组满足：

$$d(A, P) \approx d(A, N)$$

- 算法将会努力使得 $d(A, N)$ 变大，或者使得 $d(A, N) + \alpha$ 变小，从而使两者之间至少有一个 α 的间隔；
- 增加学习算法的计算效率，避免那些太简单的三元组。

最终通过训练，我们学习到的参数，会使得对于同一个人的图片，编码的距离很小；对不同人的图片，编码的距离就很大。

Training set using triplet loss



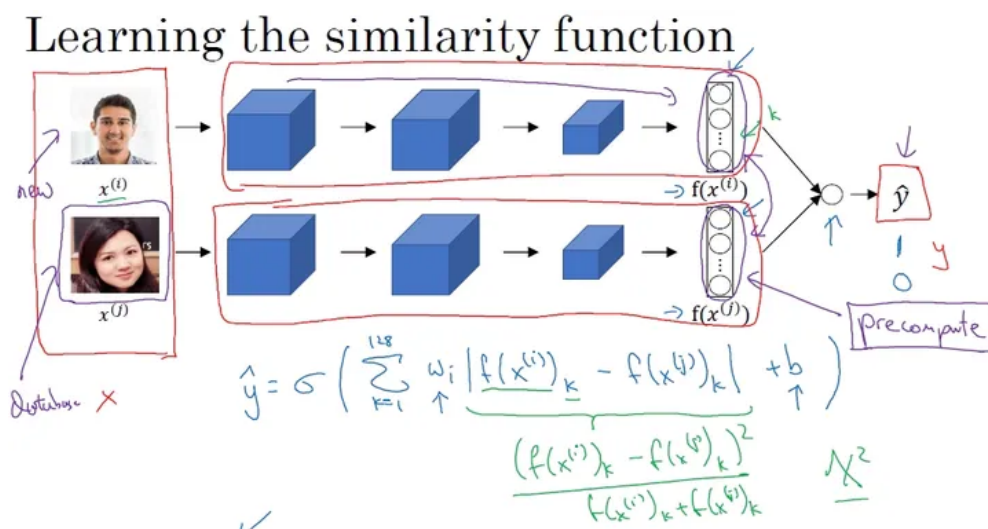
对于大型的人脸识别系统，常常具有上百万甚至上亿的训练数据集，我们并不容易得到。所以对于该领域，我们常常是下载别人在网上上传的预训练模型，而不是从头开始。

5. 脸部验证和二分类

除了利用 Triplet 损失函数来学习人脸识别卷积网络参数的方法外，还有其他方式。我们可以将人脸识别问题利用Siamese网络当成一个二分类问题，同样可以实现参数的学习。

Siamese 二分类改进：

对两张图片应用Siamese 网络，计算得到两张图片的N维编码，然后将两个编码输入到一个 logistic regression 单元中，然后进行预测。如果是相同的人，那么输出是1；如果是不同的人，输出是0。那么这里我们就将人脸识别的问题，转化为一个二分类问题。



对于最后的sigmoid函数，我们可以进行如下计算：

$$\hat{y} = \sigma \left(\sum_{k=1}^N w_k |f(x^{(i)})_k - f(x^{(j)})_k| + b \right)$$

其中, $f(x^{(i)})$ 代表图片 $x^{(i)}$ 的编码, 下标 k 代表选择N维编码向量中的第 k 个元素。

我们以两个图片编码向量对应元素之间的差值作为特征输入到logistic regression 的单元中, 增加参数 w_i 和 b , 通过训练得到合适的参数权重和偏置, 进而判断两张图片是否为同一个人。

同时输入逻辑回归单元的特征可以进行更改, 如还可以是:

$$\frac{(f(x^{(i)})_k - f(x^{(j)})_k)^2}{f(x^{(i)})_k + f(x^{(j)})_k}$$





上式也被称为 χ 方公式, 有时也称为 χ 方相似度。

在实际的人脸验证系统中, 我们可以对数据库的人脸图片进行预计算, 存储卷积网络得到的编码。当有图片进行识别时, 运用卷积网络计算新图片的编码, 与预计算保存好的编码输入到逻辑回归单元中进行预测。这样可以提高我们系统预测的效率, 节省计算时间。

总结:

利用Siamese 网络, 我们可以将人脸验证当作一个监督学习, 创建成对的训练集和是否同一个人的输出标签。

Face verification supervised learning

x		y	
		1	"Same"
		0	"Different"
		0	
		1	

我们利用不同的图片对, 使用反向传播的算法对Siamese网络进行训练, 进而得到人脸验证系统。

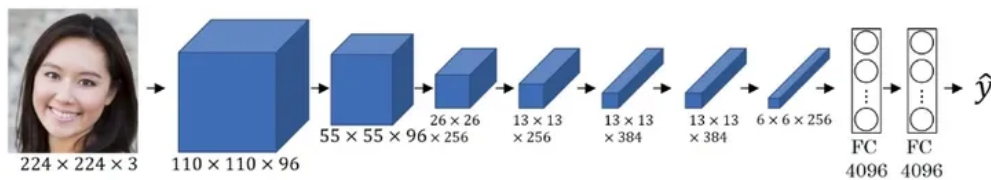
Part 2: 神经风格迁移

6. 深度网络学习内容可视化

如何可视化:

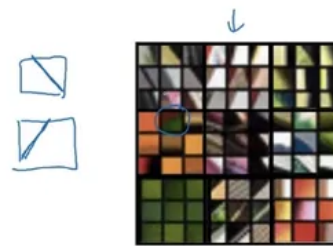
假设我们训练了一个卷积神经网络如下所示:

Visualizing what a deep network is learning



Pick a unit in layer 1. Find the nine image patches that maximize the unit's activation.

Repeat for other units.



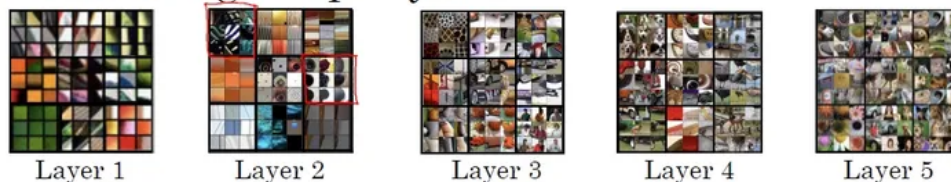
我们希望看到不同层的隐藏单元的计算结果。依次对各个层进行如下操作：

- 在当前层挑选一个隐藏单元；
- 遍历训练集，找到最大化地激活了该运算单元的图片或者图片块；
- 对该层的其他运算单元执行操作。

对于在第一层的隐藏单元中，其只能看到卷积网络的小部分内容，也就是最后我们找到的那些最大化激活第一层隐层单元的是一些小的图片块。我们可以理解为第一层的神经元通常会寻找一些简单的特征，如边缘或者颜色阴影等。

各层网络可视化：

Visualizing deep layers



对于卷积网络的各层单元，随着网络深度的增加，隐藏层计算单元随着层数的增加，从简单的事物逐渐到更加复杂的事物。

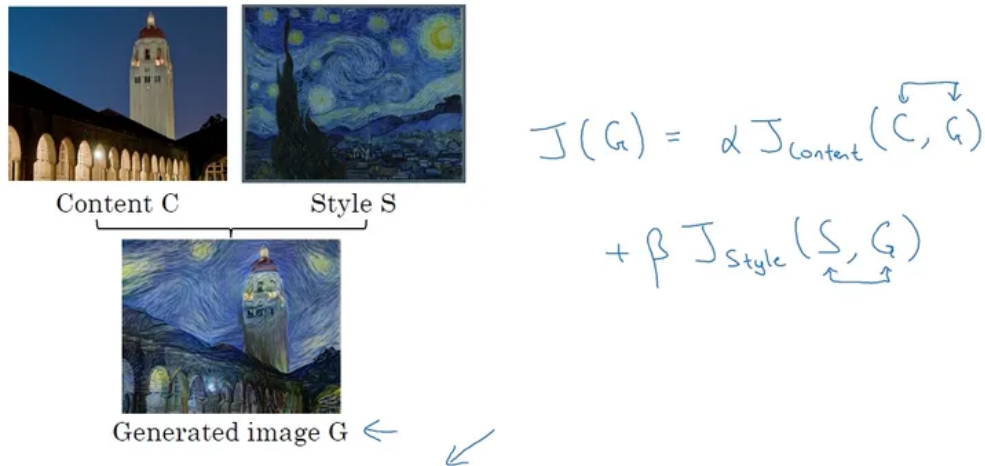
7. 神经风格迁移代价函数

代价函数：

为了实现神经风格迁移，我们需要为生成的图片定义一个代价函数。

对于神经风格迁移，我们的目标是由内容图片C和风格图片S，生成最终的风格迁移图片G：

Neural style transfer cost function



所以为了实现神经风格迁移，我们需要定义关于G的代价函数J，以用来评判生成图片的好坏：

$$J(G) = \alpha J_{\text{content}}(C, G) + \beta J_{\text{style}}(S, G)$$

其中

- $J_{\text{content}}(C, G)$ 代表生成图片G的内容和内容图片C的内容的相似度；
- $J_{\text{style}}(S, G)$ 代表生成图片G的内容和风格图片S的内容的相似度；
- α 、 β 两个超参数用来表示以上两者之间的权重。

执行过程：

- 随机初始化生成图片G，如大小为 $100 \times 100 \times 3$ ；
- 使用梯度下降算法最小化上面定义的代价函数 $J(G)$ ， $G := G - \frac{\partial}{\partial G} J(G)$ ；

Find the generated image G

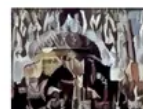
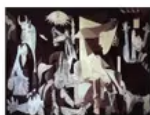
1. Initiate G randomly

$$G: 100 \times 100 \times 3$$

↑
RGB

2. Use gradient descent to minimize $J(G)$

$$G := G - \frac{\partial}{\partial G} J(G)$$



对于上图的内容图片C和风格图片S，通过梯度下降算法一次次的训练，我们可以由初始的噪声图片得到最终的风格迁移图片G。

8. 内容代价函数 (Content cost)

- 假设我们使用隐藏层 l 来计算内容代价。（如果选择的 l 太小，那么代价函数就会使得我们的生成图片G在像素上非常接近内容图片；然而用很深的网络，那么生成图片G中就会产生与内容

- 图片中所拥有的物体。所以对于 l 一般选在网络的中间层，既不深也不浅）；
- 使用一个预训练的卷积网络。（如，VGG或其他）；
 - 令 $a^{[l](C)}$ 和 $a^{[l](G)}$ 分别代表内容图片C和生成图片G的 l 层的激活值；
 - 如果 $a^{[l](C)}$ 和 $a^{[l](G)}$ 相似，那么两张图片就有相似的内容；

定义内容代价函数如下：

$$J_{content}(C, G) = \frac{1}{2} \|a^{[l](C)} - a^{[l](G)}\|^2$$

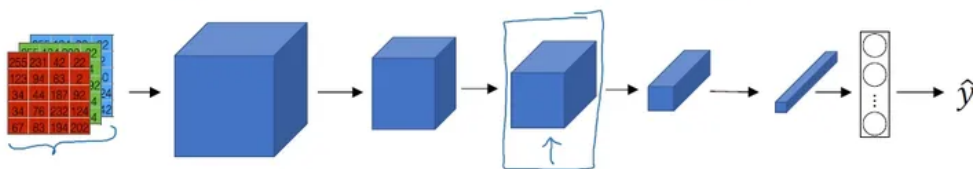
在对代价函数运行梯度下降算法时，会激励这里的内容代价函数，努力使得生成图片G隐含层 l 的激活值和内容图片C隐含层 l 的激活值相似。

9. 风格代价函数 (Style cost)

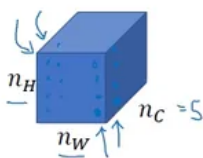
“Style” 的含义：

对于一个卷积网络中，我们选择网络的中间层 l ，定义 “Style” 表示 l 层的各个通道激活项之间的相关性。

Meaning of the “style” of an image



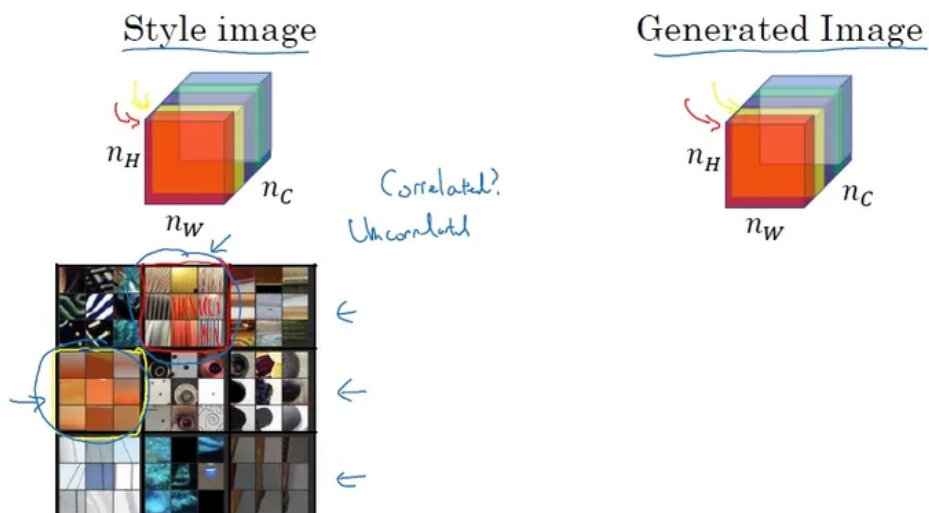
Say you are using layer l 's activation to measure “style.”
Define style as correlation between activations across channels.



How correlated are the activations across different channels?

相关性大小的度量：

Intuition about style of an image



上面是我们选出的 l 层的激活项，对于不同的通道值，代表不同的神经元所学习到的特征，这里假如红色的通道可以找到图片中含有垂直纹理特征的区域，黄色通道可以找出橙色的区域。

而相关性大小的含义就是，如假设中，图片出现垂直纹理特征的区域显示橙色可能的大小。

我们将相关系数应用到风格图片S和生成图片G的对应通道上，就可以度量风格图片和生成图片的相似度。

Style 矩阵：

- 令 $a_{i,j,k}^{[l]}$ 表示在 (i,j,k) 位置的激活值，其中i、j、k分别代表激活值的高、宽、通道；
- $G^{[l]}$ 是一个 $n_c^l \times n_c^l$ 大小的矩阵：

$$G_{kk'}^{[l](S)} = \sum_{i=1}^{n_h^{[l]}} \sum_{j=1}^{n_w^{[l]}} a_{i,j,k}^{[l](S)} a_{i,j,k'}^{[l](S)}$$

$$G_{kk'}^{[l](G)} = \sum_{i=1}^{n_h^{[l]}} \sum_{j=1}^{n_w^{[l]}} a_{i,j,k}^{[l](G)} a_{i,j,k'}^{[l](G)}$$

上面的矩阵在线性代数中又称为Gram 矩阵，这里称为风格矩阵。

代价函数：

$$J_{style}^{[l]}(S, G) = \frac{1}{2n_h^{[l]}n_w^{[l]}n_c^{[l]}} \|G^{[l](S)} - G^{[l](G)}\|_F^2 = \frac{1}{2n_h^{[l]}n_w^{[l]}n_c^{[l]}} \sum_k \sum_{k'} (G_{kk'}^{[l](S)} - G_{kk'}^{[l](G)})^2$$

内容代价函数和风格代价函数前面的归一化可以加也可以不加，因为总体的代价函数前面有权重系数。

如果对各层都使用风格代价函数，那么会让结果变得更好：

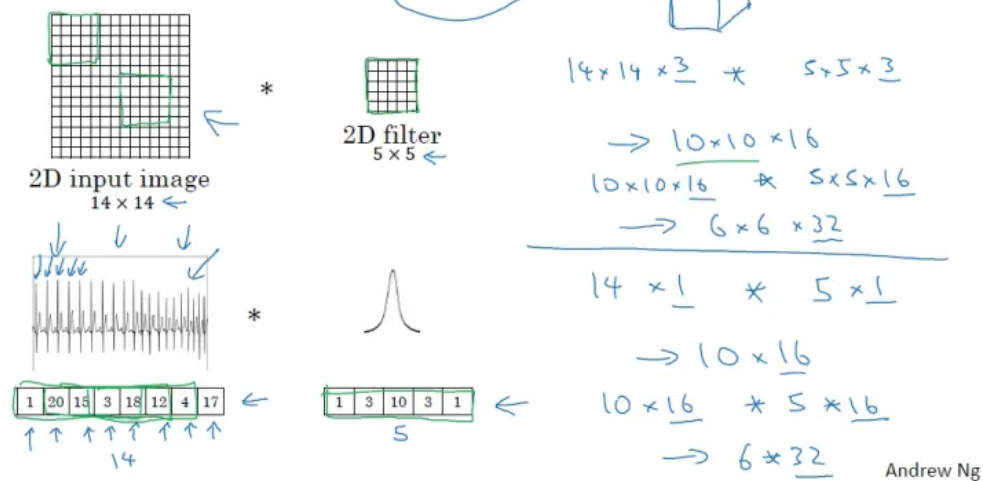
$$J_{style}(S, G) = \sum_l \lambda^{[l]} J_{style}^{[l]}(S, G)$$

10. 1D to 3D 卷积

在我们上面学过的卷积中，多数是对图形应用2D的卷积运算。同时，我们所应用的卷积运算还可以推广到1D和3D的情况。

2D和1D卷积：

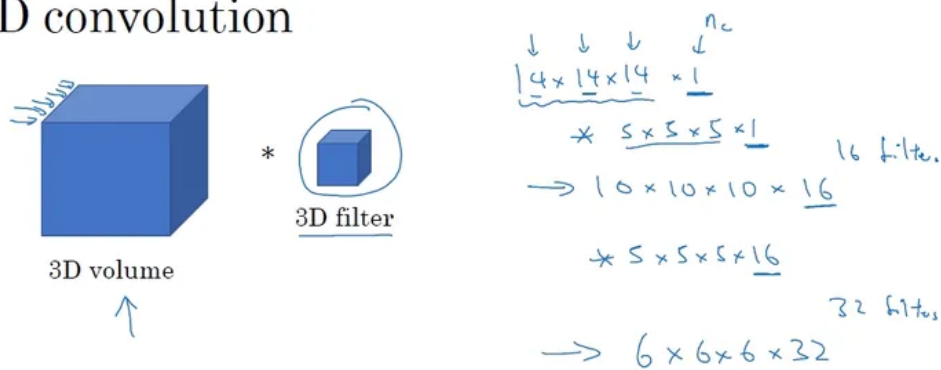
Convolutions in 2D and 1D



- 2D卷积: $14 \times 14 \times 3 * 5 \times 5 \times 3 \rightarrow 10 \times 10 \times n_c$;
- 1D卷积: $14 \times 1 * 5 \times 1 \rightarrow 10 \times n_c$ 。

3D卷积:

3D convolution



- 3D卷积: $14 \times 14 \times 14 \times 1 * 5 \times 5 \times 5 \times 1 \rightarrow 10 \times 10 \times 10 \times n_c$;
- 3D数据: 如医疗CT扫描中的即可产生身体的3D模型; 电影切片也属于3D数据。

发布于 2017-11-28 18:05