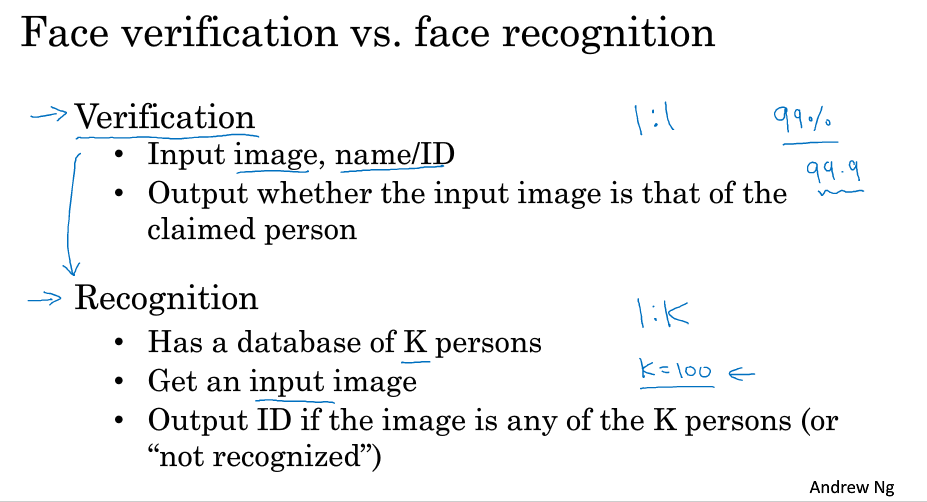
第四章 特殊应用：人脸识别

4.1 什么是人脸识别

活体检测可以使用监督式学习实现，判断是不是一个真人。

Face verification：有一张输入图片以及某人的ID或是姓名，验证输入图片是否是这个人，1对1问题，只需要弄明白这个人和他的身份相符。

Face recognition: 人脸识别比人脸验证复杂，它是一对多的问题，假设有一个100人的数据库，给你输入图片，必须去判断这个图片是不是100人中的一个。也就是说，人脸验证作为人脸识别系统的一部分，必须保证很高的正确率才能使人脸识别过程的效果好。

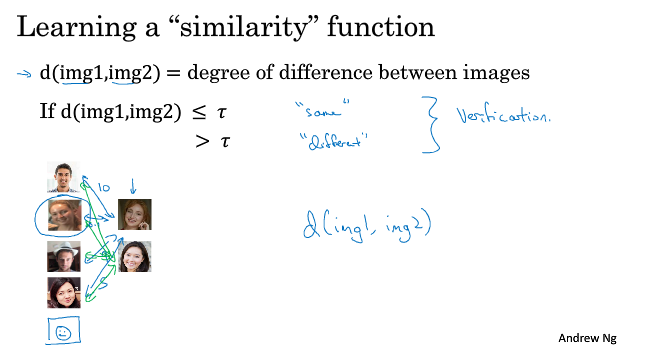


首先我们应该学习构造一个人脸验证的基本模块，如果准确率够高，就可以用在人脸识别系统中了。人脸识别之所以难是要解决“一次学习one shot”问题。

4.2 One shot学习

需要解决一次学习问题，意味着在绝大多数人脸识别应用中，需要通过一个人脸样例，就能去识别这个人，而当深度学习只有一个训练样例时，它的表现并不好。

也许可以的做法，将这个人脸照片输入到卷积网络，使用softmax单元来输出4种，或者说5种标签，分别对应数据库有的所有的4个人或者4个人都不是。但实际上这样效果并不好，因为如此小的训练集，不足以去训练一个稳健的神经网络，并且如果你的员工多了一个，难道又要重新训练你的网络去输出6个类别吗？

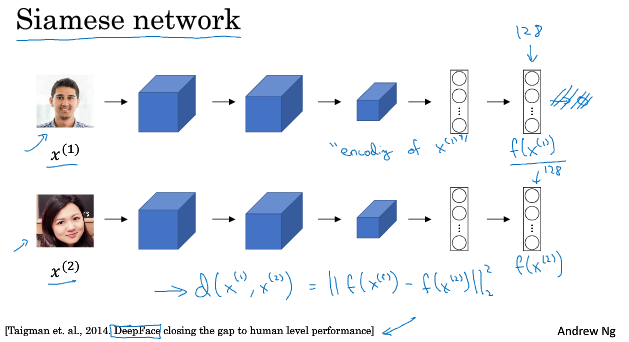


所以要使人脸识别完成一次学习，并且有好的效果，要做的就是学习Similarity函数，也就是说，让神经网络学习d这个函数，表示输入两张图片，输入两个图片之间的差异。如果两张同一个人的照片，希望输出很小的值。这也是人脸验证的过程。要将它应用于识别任务，要做的就是使用图片，用d函数去比较和数据库中的图片，通过计算，就能判断，如果和每一个人比较输出都很大，就说明这个人不在数据库中。

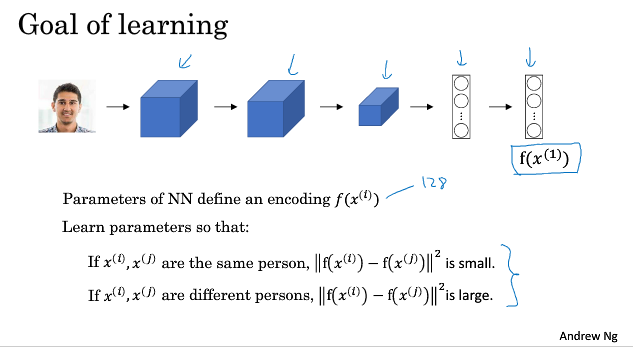
所以d函数通过输入两张图片，将让你能够解决一次学习问题。

4.3 Siamese网络

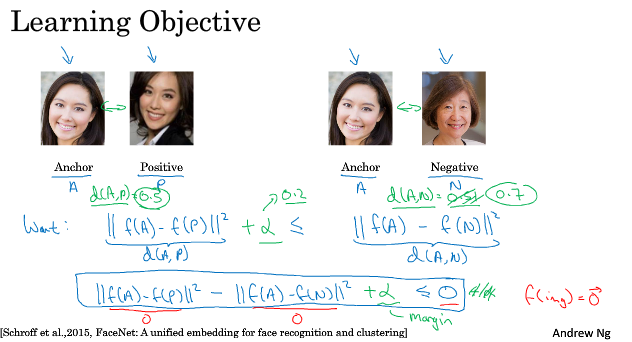
实现两张图片对比判断相似度的功能的一个方式就是用Siamese网络。首先一张图片进入卷积网络，通过全连接层进入计算，得到了输入图像的编码，,取一张图像然后表示成一个128维的向量，建立一个识别系统，就是把另一张图片喂给有同样参数的同样的神经网络，也得到一个不同的128维向量，然后使用d函数计算两个向量的距离定义为两个图像的编码之差的范数。所以就是对于两个图片运行相同的卷积神经网络，然后比较他们，这就是Siamese网络的运行原理。



如何训练这个网络呢，两者有一样的参数，所以实际只需要训练一个网络，它计算得到的编码可以用于函数d，判断是不是一个人。所以神经网络的参数定义了一种编码,要做的就是学习参数，使得如果两个图片和是同一个人两个编码的距离就小。



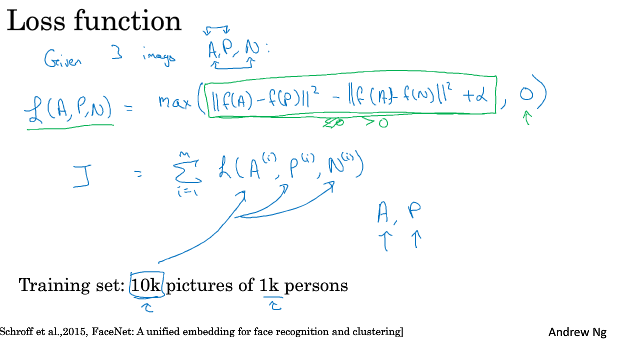
4.4 Triplet损失



三元组损失函数应用到梯度下降，可以使方向传播过程学到好的参数。什么意思，就是我们应该使用一个三元组，有三张照片，一张anchor，一张positive表示是本人，一张negative不是本人，分别用A，P，N表示，所以我们将三张图片放到同一个网络之后，得到三个向量，分别计算(anchor,positive)与(anchor,negative)之间的距离，根据我们的目标，前者距离小，后者距离大，所以,

但是有种特殊情况会满足这个式子，但是不满足我们的目的，就是当三个向量都为0或者三个向量输出一样的情况，此时0=0，所以我们增加一个间隔参数，目的就是避免这种情况发生。

目的则是使不是同一个人的两张照片计算出来距离尽可能的大，用这样的目标去训练神经网络。拉大两个元组之间的距离。

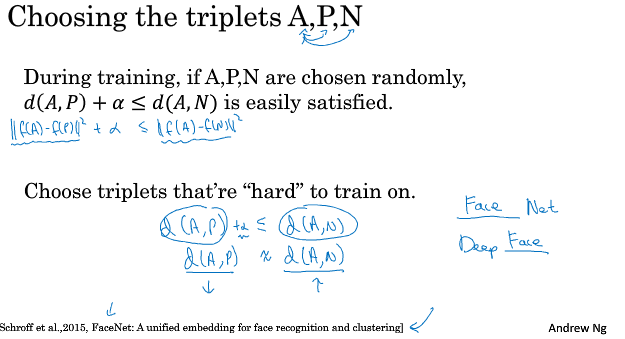


接下来定义loss function，基于三元图片，

公式的意思是取前者和0之间的最大值，只要前者小于或等于0，那么损失就是0。所以我们的目标就是使现在的差值比0小，符合提议所以损失值为0，如果前者大于0，取最大值，最终得到前者计算出来的结果，就得到一个正的损失值，所以通过最小化这个损失函数，得到的效果就是使两组对象的差值为0。

上面是基于一个三元组的损失函数定义，那么对于一个训练集来说，

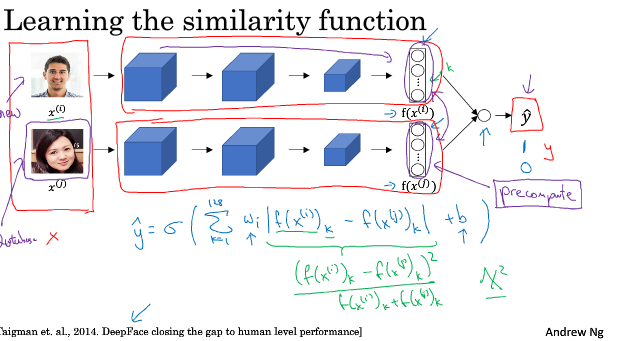
所以假如你有1000个人的照片，那么就是利用这10000张照片生成很多不同的三元组，然后训练你的学习算法，对这种代价函数使用梯度下降，代价函数就是定义在数据集里的三元组图片上的。对一个人肯定需要好几张照片，如果每个人只有一张，那么是没法训练。当训练完这个网络之后，可以应用到一次学习上。



如何选择APN三元组呢，如果是随机选择两张一个人的，和与这个人不是同一人的照片，最后想要满足（A，P）<=(A, N)是很好达到条件的，那么对网络的训练意义就不大，所以我们应该选择尽量比较难的三元组，也就是说d(A,P)比较接近d(A,N),这样你的算法才会尽量使两者的差距变大，至少有这么大的差值。并且这样的难的三元组还能增加你的学习算法的计算效率。

4.5 面部验证和二分分类

Triplet loss 是学习人脸识别卷积网络参数的好办法，还有其他学习参数的方法，如何将人脸识别当成一个二分分类的问题。就是使用一对Simaese网络，并且分别输入一张图片，都得到一个128维的向量以后，输入到一个逻辑回归单元中，计算loss，如果是同一个的照片，就输出1，不是就输出0。就这样最后将人脸识别问题转化成了二分分类的问题。



最后的逻辑单元怎么处理，最后输出

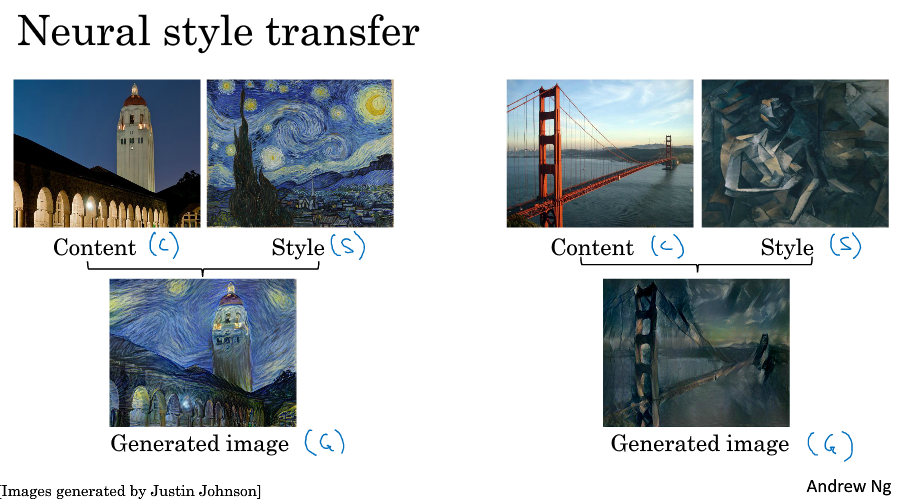
就像普通逻辑回归一样，在这128维向量上训练合适的权重，用来预测两张图片是否是一个，来获得预测是0还是1。

有一个可以加速训练的技巧就是，当我们把照片存在数据库，一个员工进入识别的时候，不需要每次都去计算那128维的编码，可以计算好存放起来。当有一个新员工进入时，使用一次Simaese网络计算128维编码，然后使用它和预先计算好的进行比较得到预测值。

总结，把人脸验证当做一个监督学习，创建一个只有成对图片的训练集，使用反向传播训练网络，变成一个二分分类的问题，这样的效果也很好。希望学习到在一次学习时，需要用什么来训练人脸验证或者人脸识别系统。

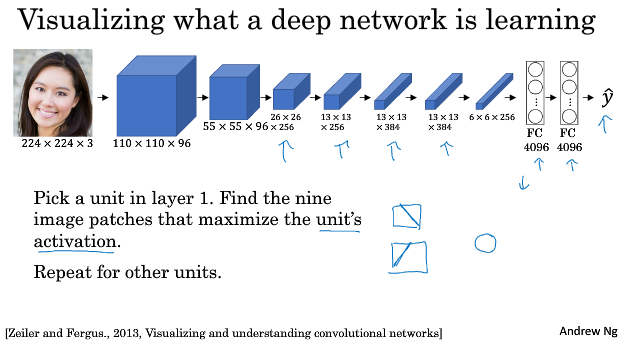
4.6 什么是神经风格转换

C表示内容图像，S表示风格图像，G表示生成的图像，为了实现神经风格迁移，你需要知道卷积网络提取的特征，在不同的神经网络深层的浅层的。



4.7 深度卷积网络在学什么

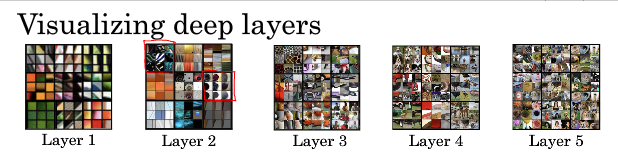
可视化和理解卷积网络，这一篇论文主要是通过可视化的方法探究卷积网络的每一层的神经元到底对应的（或者说学习到）的特征到底是什么样子的。思想就是构建一个逆向的卷积神经网络，但是不包括训练过程，使用原网络各个层的feature map当作输入，逆向生成像素级图片。这里的逆向卷积神经网络，其实采用了反池化，反激活和反卷积这样一个步骤，最后把本来一张13\*13大小的特征图(conv5大小为13\*13)，放大回去，最后得到一张与原始输入图片一样大小的图片(227\*227)。



关于可视化卷积的理解的博客：

<https://www.cnblogs.com/hellcat/p/7149348.html>

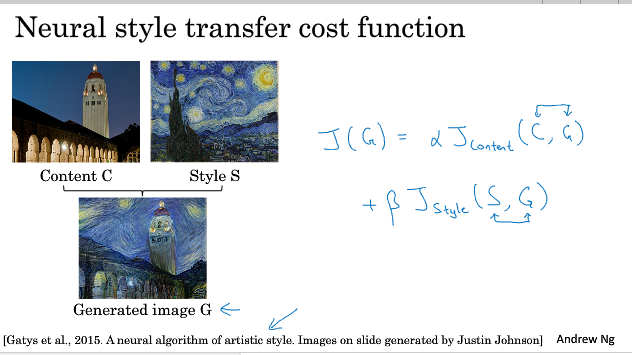
如图加入我们训练了一个卷积神经网络，Alex网络，希望看到每一层的计算结果，可以这么做，从第一层的隐藏单元开始，假设遍历了训练集，然后发现一些图片或者是图片块，最大化地激活了那个运算单元，换句话说，将你的训练集经过神经网络，然后弄明白哪一张图片最大限度地激活特定的单元，第一层的隐藏单元只能看到小部分卷积神经。比如选择一个神经单元发现有9个图片最大化地激活了这个单元，可能找到这样的9个图片块。 以此类推。在卷积网络的更深层次可以看到图片的更大的图片块。



结论，从layer 1、layer 2学习到的特征基本上是颜色、边缘等低层特征；layer 3则开始稍微变得复杂，学习到的是纹理特征，比如上面的一些网格纹理；layer 4学习到的则是比较有区别性的特征，比如狗头；layer 5学习到的则是完整的，具有辨别性关键特征。层次越深，检测到的模式越复杂。

4.8 代价函数

要构建一个神经风格迁移系统，就要为生成的图像定义一个代价函数，通过最小化代价函数，可以生成你想要的任何图像。J(G)用来评判某个生成图像的好坏，使用梯度下降法去最小化J(G)，而生成这个图像。

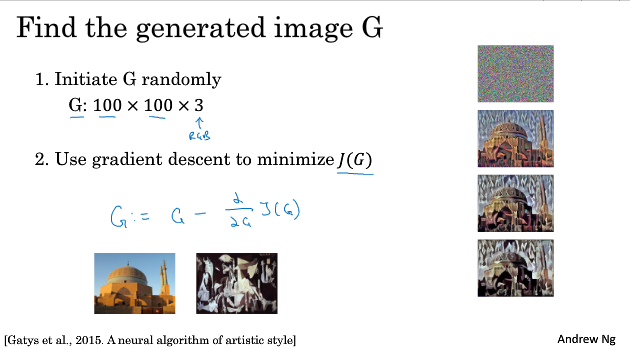


如何定义J(G)呢，分为两个部分，第一个为内容的代价，关于内容图像和生成图像的代价，用来度量生成图像的内容和原始图像内容有多相似，然后加上另一部分，一个风格代价函数，用来计算S和G的风格有多相似，最后用两个超参数来确定这两部分之间的权重。

具体的算法步骤如下：

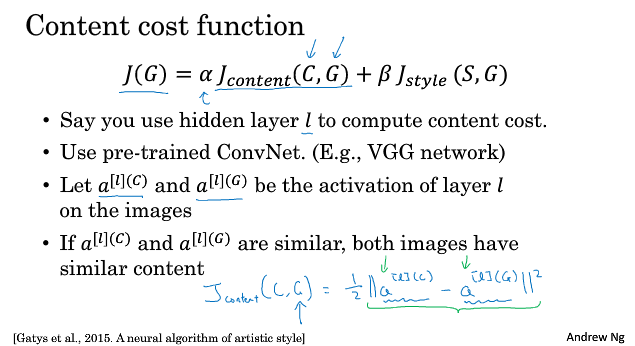
1. 随机初始化生成图像G，自己定义大小如100\*100\*3
2. 使用梯度下降法最小化

实际上更新的是图像G的像素值。



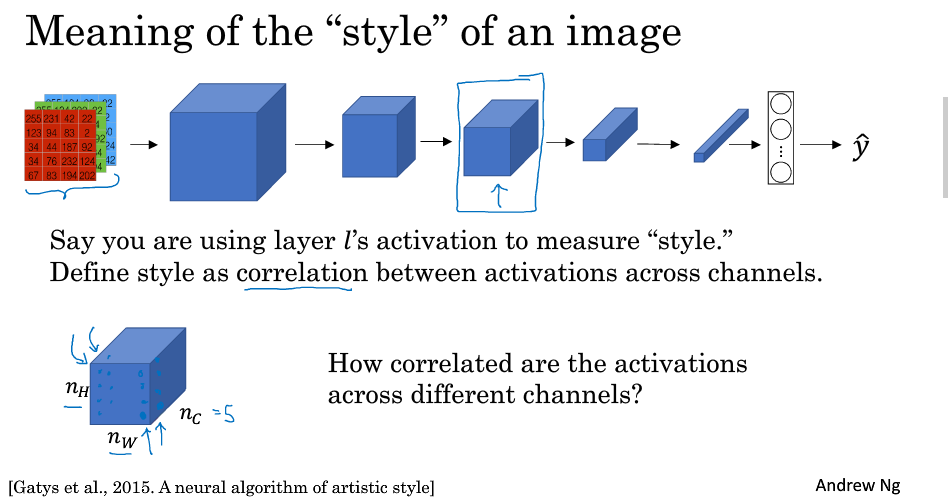
4.9 内容代价函数

其实就是计算了内容图片C和生成图片G内容上有多相似，激活值是否相似可以反映到内容上是否相似，前面的是归一项，也可以不要影响不大，因为这些都可以由代价函数J(G)里面的超参数来调整，注意激活值都是展开成向量的，才计算范数。

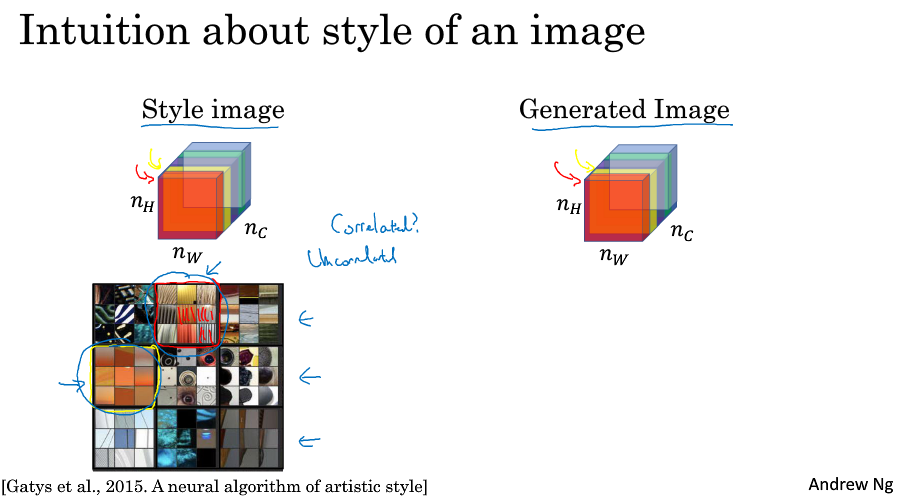


4.10 风格损失函数

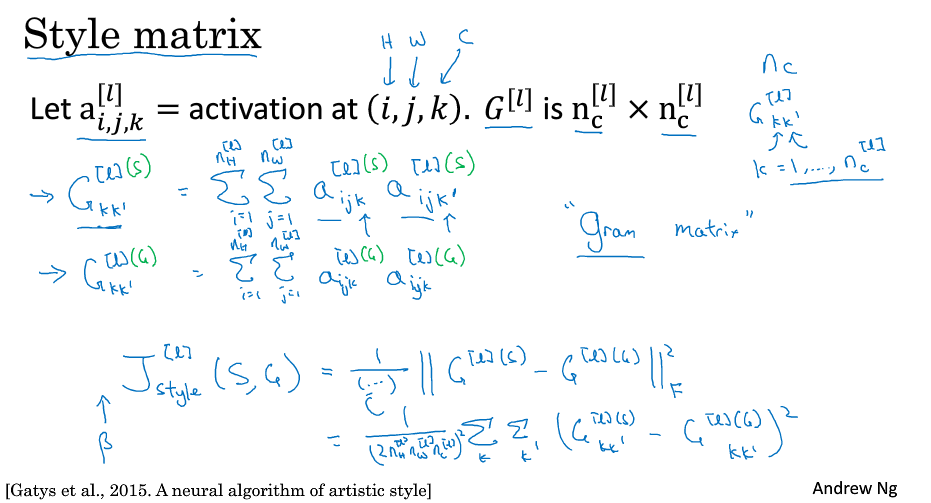
在卷积网络运算的过程中，我们应该选择第l层的激活值来侧量风格，为图片的风格定义一个深度侧量，这个l值不能选的太大，也不能选的太小，然后将style定义为l层各个通道之间激活项的相关系数。为什么可以这么定义呢，因为之前可视化卷积网络中，我们学到每一层的隐藏单元，在卷积网络里面可以理解为每一个通道，都是在计算一些特征的，使用反卷积可以看到学到了什么东西，假设一个隐藏单元学习到的是关于垂直边缘的探测，另一个隐藏单元学习到关于色彩，那么探究一张图像的风格不就是看垂直线的地方是不是颜色相同，或者不出现那些颜色，也就是判断学到的内容的相关性来定义风格。

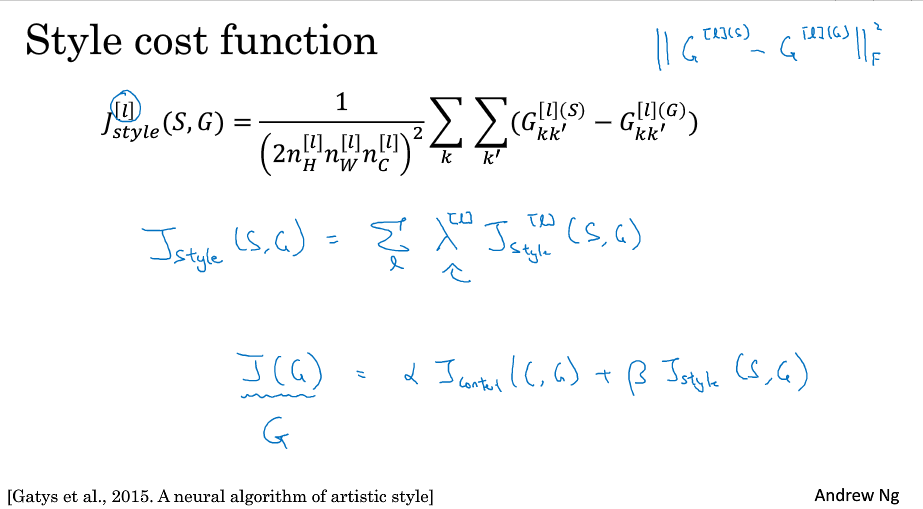


首先我们得知道如何定义一个图像的风格，图像是三维的，nh\*nw\*nc，高\*宽\*通道数，在每一个通道是上是一个二维矩阵，所以我们定义一个图像的风格是通过计算每两两通道之间的相关系数。两两通道计算，所以最后的得到的有nc\*nc个数，就做成一个矩阵，所以对一个图像，我们有风格矩阵，它的维度是。



下面关于这个风格矩阵怎么计算也很容易理解了，里面的每个元素表示的则是第k通道和k`通道的相关系数值。接下来相关系数怎么计算呢，就是两个通道之间所有对应位置的两个数的乘积加和。这里不是标准的计算方式，因为也没有减去平均值什么的。所以通过这些计算，我们就能计算出G生成图像和S风格图像各自的风格矩阵。最后就能定义风格代价函数部分了，就是两个矩阵的对应元素的范式。前面有个系数归一化，但是其实总代价函数的参数αβ已经计算了。





所以

这是关于第l层上的风格代价计算，也可以定义风格代价为把每个层的结果都加在一起，那么就还需要对每个层定义权重，额外的超参数，这样使你能够在神经网络中使用不同的层，包括之前的一些可以测量类似边缘这样的低级特征的层，以及之后的一些能测量高级特征的层，使我们的神经网络在计算风格时候，能够同时考虑到这些低级和高级特征的相关系数。在这样的基础上，定义超参数可以得到更合理的选择。

最后得到关于风格转化网络的总体代价函数：

4.11 一维到三维推广

1D数据就比如时间序列数据，2D比如图片，虽然是个RRB三个通道，但是其实是平面图，使用的过滤器跟通道数必须相同，而3D数据比如说人体某个器官模型，另一个维度可以理解成是切片，使用过滤器的时候必须是3维的，然后过滤器的个数决定了最后输出内容为4个维度。

