目录

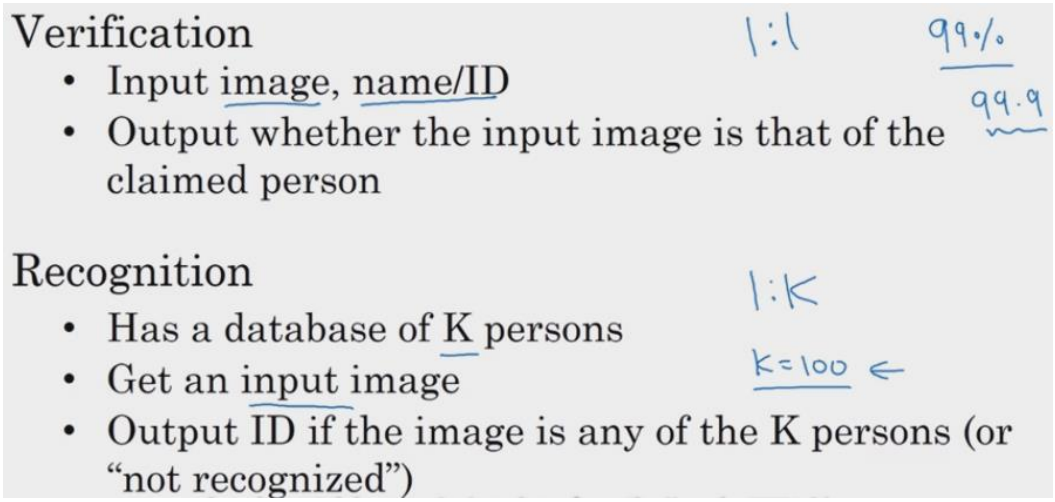
Week4 人脸识别和神经风格转换

4.1 什么是人脸识别

现在为了让人脸识别技术变得更智能，通常还会额外加入活体识别（Liveness detection）技术。例如，一个外来人员使用公司内部人员的照片对准摄像头，来企图欺骗人脸识别系统，但是活体识别就能认出到底镜头前的是真人，还是照片。活体识别技术可以通过监督学习来实现：

Face verification/Face recognition：

通常，Verification 可看做一对一的问题，只用判断当前照片的人是不是输入的人名。而 Recognition 是一对多的问题，它需要判断当前人是不是系统内部的某个人：



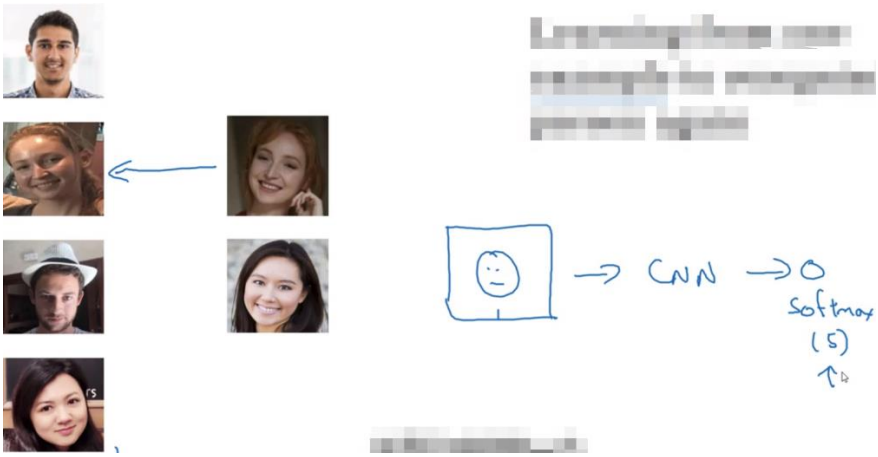
后续，我们将先构建一个人脸认证系统。当它准确率足够高时，再用它去构建人脸识别系统。

4.2 One-shot 学习

人脸识别面临的一个问题就是一次学习问题，这意味着在绝大多数人脸识别应用系统中，你需要通过某人的仅仅一张样本图片，就能后续去识别这个人。历史上 DL 在处理单样本问题上取得的效果并不好。

传统方法：

对于该 one-shot 学习问题，若使用传统 CNN 网络来实现：把待识别的人脸输入 CNN，经过各种卷积池化操作后，得到 softmax 的输出。假设系统数据库中采集了 4 个人的照片，现要输入待识别照片，输出是 5 维向量（分别代表当前人脸是四个人之一的概率+谁都不是的概率）。



缺陷 1：由于这是 One-shot 问题，故数据集的容量一定不大，不足以训练一个精良的 CNN 网络，所以得到的 CNN 识别率堪忧；

缺陷 2：假设现在公司新入职一个员工，那么他的照片将加入系统的样本库，还将导致输出 y 将变成 6 维。故 CNN 网络的结构也随之改变，又需要重新训练。

学习相似度函数：

为了使系统更有健壮性和可扩展性，现引入相似度函数 d(img1, img2)来衡量两张图片的差异。并设置一个阈值 τ，当差异小于 τ 时则认为两张图对应同一个人。由此可通过学习该相似度函数，来实现人脸认证。

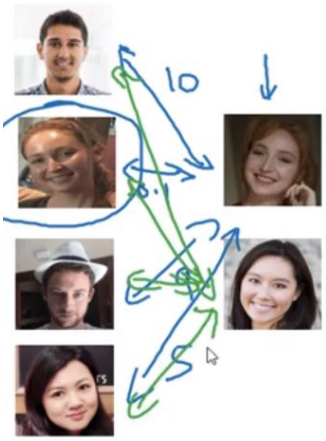


图 1：计算该人脸与数据库中每个样本的相似度，会发现它与数据集第 2 张图的 d 值很小，那么可认为识别成功！

图 2：由于该人脸与数据集中每个样本的差异度 d 都很大，都超过了阈值 τ，则认为该人不属于公司内部人员，则认证失败！

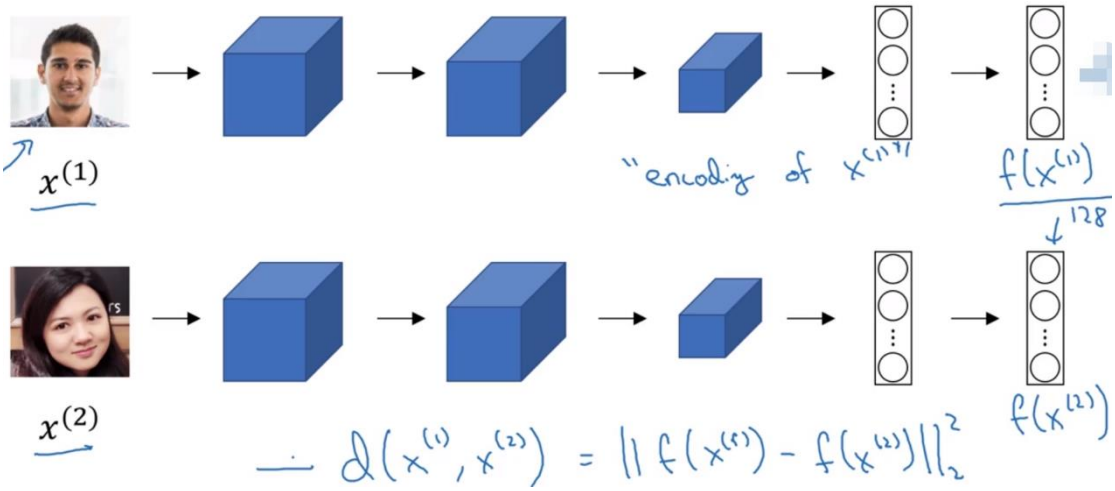
核心：观察以上可知，实现了 One-shot 检测的核心是学习一个良好的相似度函数 d(img1, img2)。a.因为即使数据集很小，但相似度函数可以准确地告诉我们它与数据集中的每个人是不是同一个人，来实现人脸识别；b.假如有新同事入职，我们只用把他的照片加入数据集即可一劳永逸，不会像上文一样带来任何额外的工作。

4.3 Siamese Network

Siamese 网络：

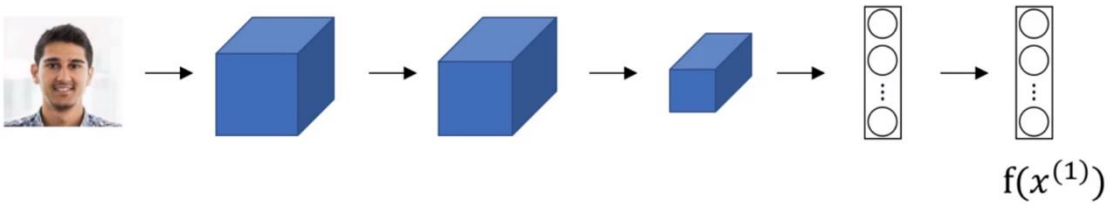
通过 Siamese 网络[1]我们可以很方便地实现相似度函数。设输入图片为 x，它经过 CNN 网络一系列卷积、池化、全连接层后得到一个向量 f(x)，该向量（假设为 128 维）可以认为是输入图片的特征向量，是输入 x 的 encoding。

现要衡量两张图片的相似度，可以直接用两个特征向量的距离来表示，即 $d(x^{(1)}, x^{(2)}) = \| f(x^{(1)}) - f(x^{(2)}) \|_2^2$.



训练：

由于 Siamese 网络的参数决定了输入 x 的编码 f(x)，当两张图片对应同一个人时，我们希望这两个输出向量的距离尽量小；同理，当两张图对应不同人时，我们希望这两个输出向量距离尽量大。



Parameters of NN define an encoding $f(x^{(i)})$ 128

Learn parameters so that:

If $x^{(i)}, x^{(j)}$ are the same person, $\|f(x^{(i)}) - f(x^{(j)})\|^2$ is small.

If $x^{(i)}, x^{(j)}$ are different persons, $\|f(x^{(i)}) - f(x^{(j)})\|^2$ is large.

故我们可以用 BP 算法来更新总误差对每个参数的梯度，并确保满足以上的限制条件，就可训练得想要的网络。

[1]Taigman et. al., 2014, DeepFace closing the gap to human level performance.

4.4 Triplet 损失

要想通过学习神经网络的参数来得到优质的人脸编码（encoding），方法之一是定义三元组损失函数（Triplet loss function），然后应用 BP 和梯度下降来更新参数。

学习目标：

如图，在 Triplet 中，我们会同时看三张图片 A, P, N（Anchor, Positive, Negative）。我们希望：A 与 P 的距离尽量小，而 A 与 N 的距离尽量大，即 d(A, P) – d(A, N) <= 0。值得注意的是，这里为了让算法将二者的区别变得更大，我们不仅希望差值仅仅小于 0，更希望它小于一个负数，这样才更能体现出正负样本的区别：

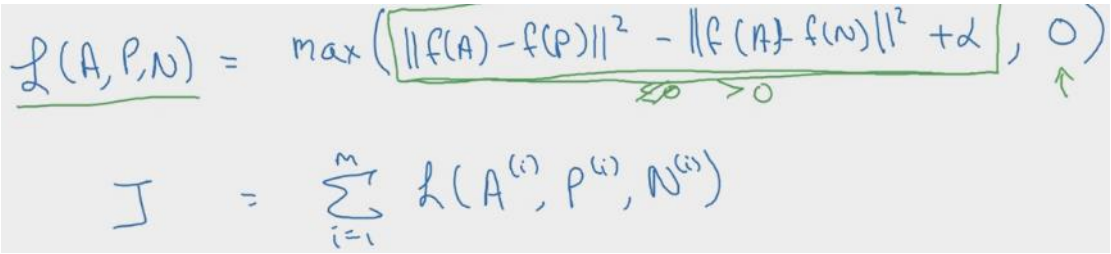
Want: $\underbrace{\|f(A)-f(P)\|^2}_{d(A,P)} + \alpha \leq \underbrace{\|f(A)-f(N)\|^2}_{d(A,N)}$

$\underbrace{\|f(A)-f(P)\|^2}_{0} - \underbrace{\|f(A)-f(N)\|^2}_{0} + \alpha \leq 0$ 小于0 (margin)

这里，超参数 α 也称为间距（margin），概念与 SVM 中的间距非常相似。间距越大则要求算法的精确度越高。

代价函数：

给定三张图片 A, P, N，定义 loss 如下：



观察可知：a.当间距不足即第一项大于 0 时，则 loss 等于第一项的值，表示该三元组图片还存在误差，需要继续优化参数来使它们的间距更大；b.当间距足够即第一项小于 0 时，则 loss 为 0，表示间距足够，算法能足以区分这 3 张图片的内容，故不会造成误差，从而不会影响参数继续改变。

算法在整个数据集的误差=所有样本三元组误差之和。由此再用 BP 可得总误差对每个参数的梯度，再调用梯度下降/优化算法，即可得到良好的网络，从而产生良好的图片编码（encoding）。

三元组的选择：

理论上，我们要求 A 和 P 是同一个人，而 A 和 N 不是同一个人。而实际上由于每个人长相差异本身就比较大，所以公式 d(A, P) – d(A, N) <= α 实际很容易满足，这就导致训练后网络并不会学到什么有价值的东西。

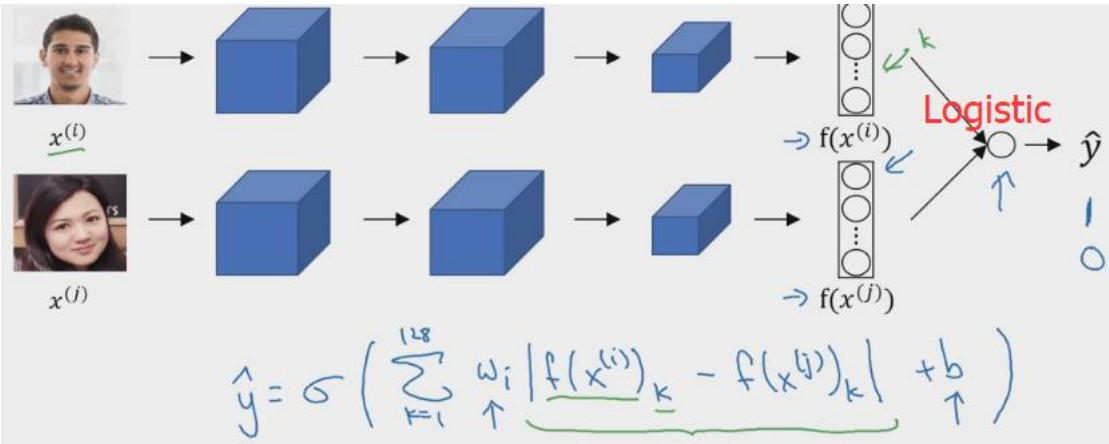
所以我们应尽可能选择难度较大的(A, P, N)三元组[1]，如 d(A, P) ≈ d(A, N)。这样，由于正负样本的区别不是很大，算法会尽可能地去学习，使得二者的差距达到 α。由此训练后的网络将具有更高的精确度。

[1]Schroff et al., 2015, FaceNet: A unified embedding for face recognition and clustering

4.5 人脸认证与二分类

Triplet 是一种学习参数的方法，但我们还可以把人脸认证问题转换成二分类问题来学习参数。

与前文不同的是，这里我们把两张图片的特征向量 $f(x^{(1)})$和 $f(x^{(2)})$输入给 logistic 单元，最终得到一维输出 y（0/1），分别代表是/不是同一个人。其中，logistic 单元的输入是 $f(x^{(1)})$和 $f(x^{(2)})$每个维度差值的绝对值（也可是平方）的线性变换：

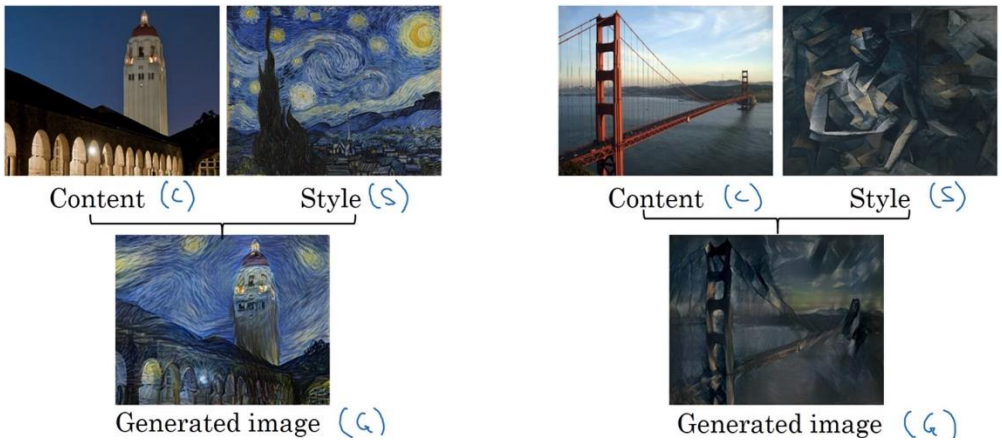


提高识别速度的技巧：

当门禁处有个待识别的人脸，不用把它和数据库的每个照片都运用以上流程。因为每次都重复计算数据集中人脸的特征是没必要的，我们可以提前存好数据集中每个员工的人脸特征{f($x^{(1)}$), f($x^{(2)}$), , f($x^{(n)}$)}，然后每当有门禁出有人到来时，仅计算待识别人的特征向量，依次与上述预先计算好的特征向量比对即可。

4.6 什么是神经风格转换

输入一张内容图片（Content）和一张风格图片（Style），神经风格转换的任务是在保留图片 C 主体内容的前提下，把 C 的画风转换成 S 的画风：



4.7 深度卷积网络在学什么

本节将用可视化的方法展示卷积网络中深度较大的层真正在做什么，这将有助于理解如何实现神经风格转换。

可视化[1]Layer1：

对第一层的某个单元：选择 9 个使该单元激活值最大的图片块，或许得到左上角 3*3 的图片块，结果表示该 unit 或许对左斜的线条比较明显。

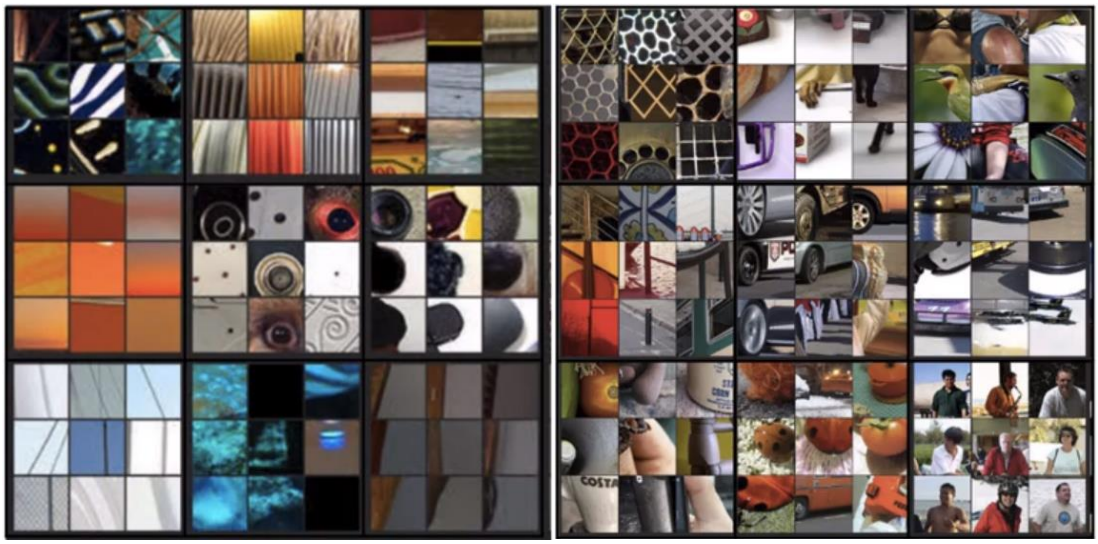
对第一层其他单元执行同样操作，得到的图片块可能如下。由此可见，第一层神经元往往只学一些比较低级的特征，如颜色、轮廓、线条等：

可视化 Layer2 和 Layer3：

第二层：学到的特征明显更为复杂，例如垂线，圆形轮廓等；

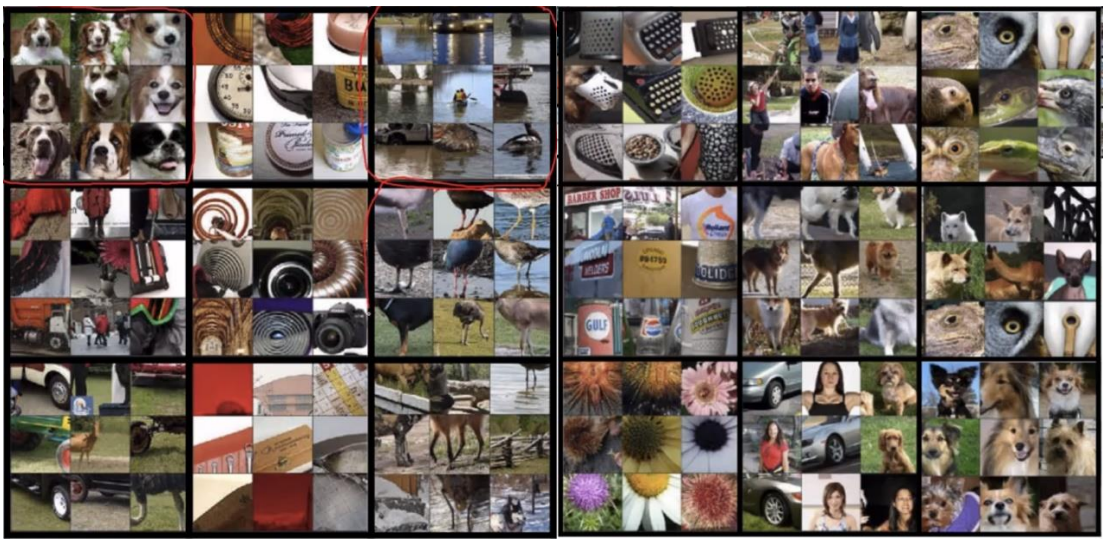
第三层：图 5 表示该单元对左下角敏感，所以学到了许多车轮；图 9 识别到了人的特征；图 1 识别到了斑点条纹的线条。

综上，第三层的特征明显比第一二层的特征更高级：



可视化 Layer4 和 Layer5：

第四层：图 1 几乎成了一个狗的分类器，而且该类狗的外形还比较相似；图 3 学到了水的特征，因为表面都有水的波纹；图 5 学到了螺旋状物体的特征；图 6 学到了鸟类脚的特征。本层的特征更复杂，更细致：



[1]Zeiler and Fergus., 2013, Visualizing and understanding convolutional networks.

4.8 代价函数

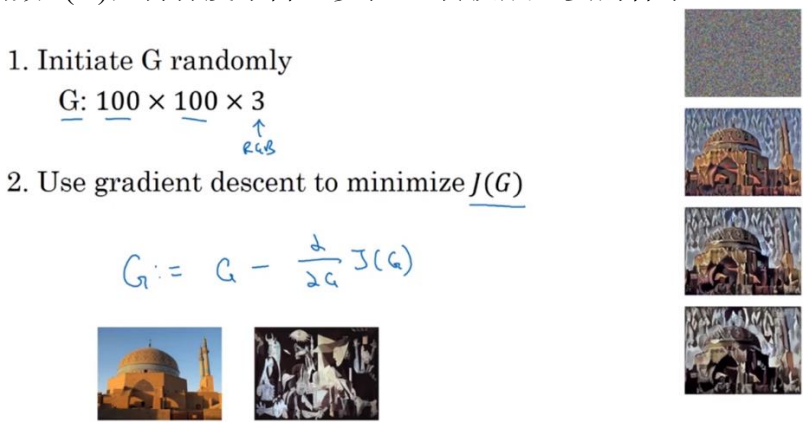
同样的思路，我们用一个代价函数[1]衡量生成图片 G 与输入图片 C、S 的融合程度/图像好坏，通过求得参数的梯度，再用梯度下降对参数进行优化，我们就能得到任何想要的图片。

代价函数 J(G)分为两部分：$J_{content}(C, G)$用来衡量 C 和 G **内容**的相似程度，$J_{style}(S, G)$用来衡量 S 和 G **风格**的相似程度。最后，在二者前面分别乘两个超参数 α 和 β 来衡量二者的权重（想要内容更多地还原，还是风格更多地还原）：

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

生成图像：

首先随机初始化 G，得到如题的噪声图；其次计算 G 与输入 C, S 的误差，得到代价函数 J(G)；再梯度下降逐步把 G 转换成想要的样子：



[1]Gatys et al., 2015. A neural algorithm of artistic style. Image on slide generated by Justin Johnson.

4.9 内容代价函数

使用隐藏层 L 来计算 content cost，如果 L 选得很小，那么生成的图片在像素上就会很接近输入图片 C；如果 L 选得很大，那么可能会问图片里是否会有狗，它会确保生成图片里也有一只狗。所以 L 不会选的太浅也不会选的太深。

其次我们会使用预训练好的 ConvNet（如 VGG），并令 $a^{[L](C)}$和 $a^{[L](G)}$分别为图片在隐藏层 L 上的激活值：

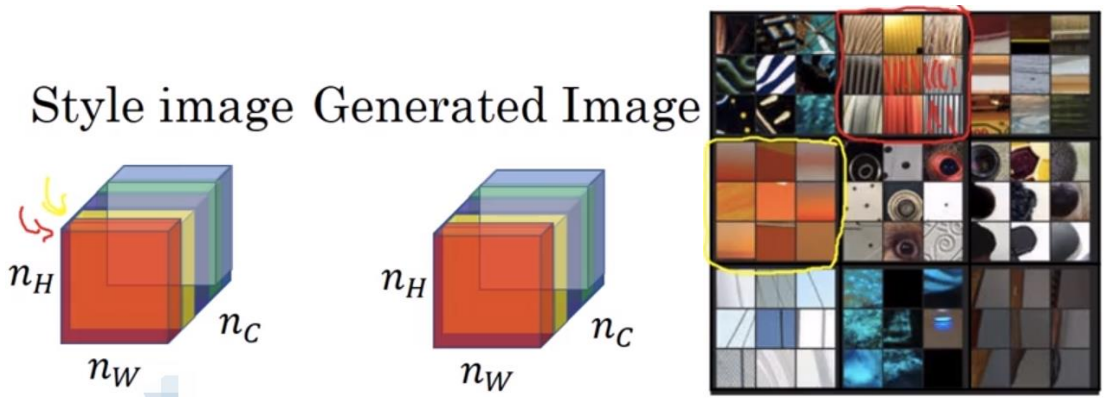
If $a^{[l](C)}$ and $a^{[l](G)}$ are similar, both images have similar content

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

4.10 风格代价函数

图片“风格”的含义：

假设用隐藏层 L 的激活值来衡量“风格”，则定义“style”为 L 层各个通道之间激活值的**相关系数**（Correlation）。因为很多时候画风是指在不改变图片内容的前提下，变换它整体的颜色风格（类似加滤镜），而不同的通道就恰好对应的不同颜色。

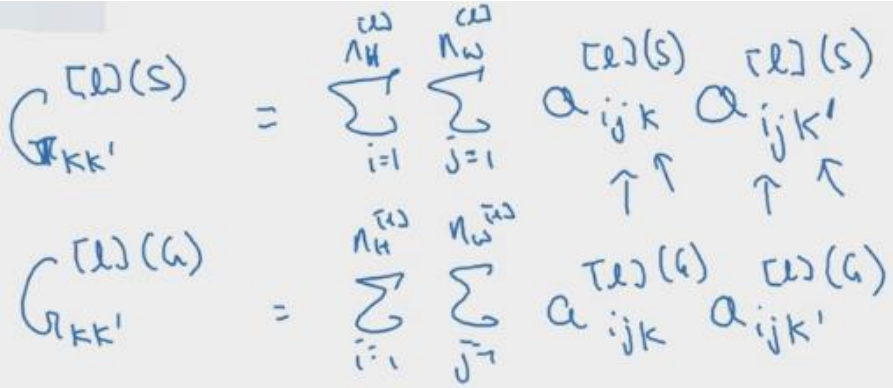


如上，取图片的前两个红色、黄色通道，假设它们分别代表第 2、4 个图片块。如果这两个通道相关（Correlated），则代表该图片中，出现竖直条纹的地方很大概率将呈现橘黄色！反之亦然。

所以，对于输出图片 G，我们也可以检查其前两个红色、黄色通道，看二者是否相关。如果也相关，那么就可以说图片 G 和图片 S 的“风格”很相近。

风格矩阵 style matrix：

设 $a^{[l]}_{i,j,k}$ 为隐藏层 l 在(i, j, k)位置处的激活值，其中 i, j, k 分别代表图片宽度、高度和通道数。设风格矩阵 $G^{[l](S)}$表示关于第 l 层隐藏层和风格图片 S，它是一个 $n_c * n_c$ 的矩阵，其中 n_c 表示通道数：



风格代价函数：

通过以上公式，我们可以分别得到图 S 和 G 中，风格矩阵的元素，遍历每个 k 即可得风格矩阵。这是一个非标准的协相关函数，因为它没有减去平均值。若风格相近，该求和值会很大；若不相关，则该值很小。

最终，我们可定义风格代价函数 $J_{style}^{[L]}(S, G) = ||G^{[L](S)} - G^{[L](G)}||_F$ 为两个矩阵的 Frobenius 范数，即对应元素差值平方的和：

$$J_{style}^{[l]}(S,G) = \frac{1}{\left(2n_H^{[l]}n_W^{[l]}n_C^{[l]}\right)^2}\sum_k\sum_{k'}(G_{kk'}^{[l](S)} - G_{kk'}^{[l](G)})$$

如果要对各层都使用风格代价函数，则可把代价函数定义为各层代价之和，并用额外的超参数 $\lambda^{[L]}$ 来表示各层的权重。这能让我们的网络转换的风格，既包含低级特征（如边缘、颜色），又包含高级的特征：

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

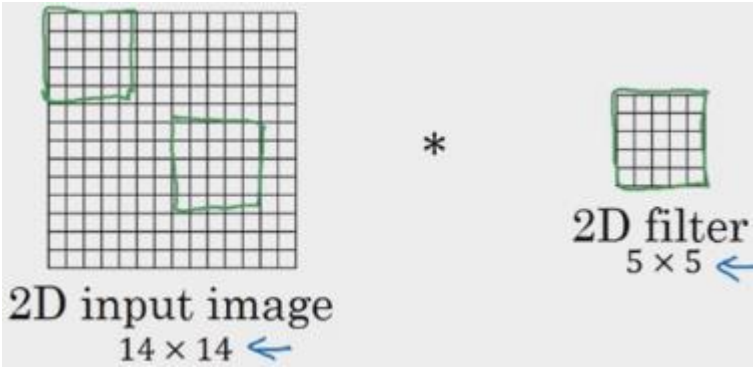
综上，总的代价函数同时结合 $J_{content}$ 和 J_{style} 可得 J(G)，用梯度下降对其训练，后续可以得到很好的生成图片 G。

4.11 一维和三维的推广

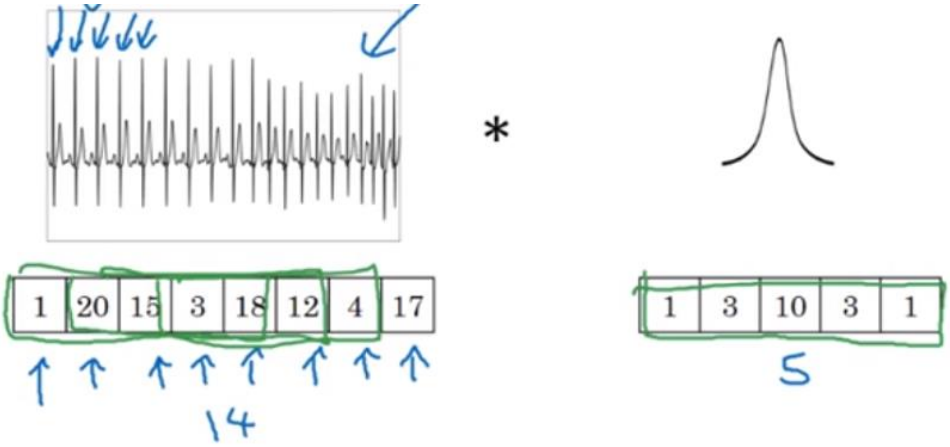
目前我们用卷积网络都处理的是二维数据（如图片），本节将介绍如何在一维和三维数据上进行卷积。

不同维度上的卷积：

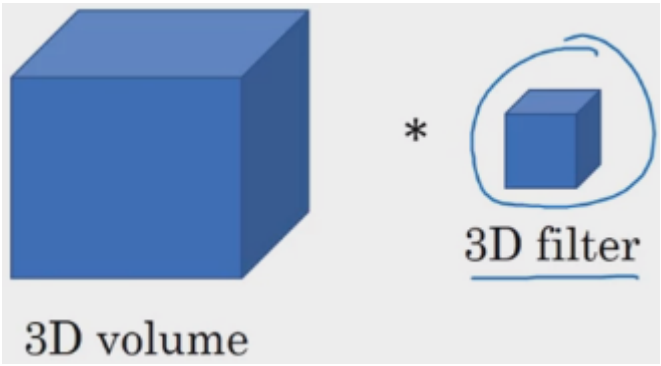
2D 卷积：此处设输入图片为 14*14*3，设 kernel 的维度为 5*5*3（注：此处与以往有所不同，kernel 在通道上也有维度），设#kernel 为 16，则输出为 10*10*16：



1D 卷积：其实就是序列数据，经常也会用 RNN 来处理，则此时 kernel 也是一维的。在卷积过程中，这个细长的 kernel 将横着在一维数据上依次扫描、卷积：



3D 卷积：设输入为 14*14*14*1（通道为 1），kernel 为 5*5*5*1（通道为 1），设 kernel 数量为 16 个，则卷积后输出为 5*5*5*16：



3D 卷积在医疗图像中用的比较多（CT 图包含大脑各个维度的二维切片图），也可以用于处理视频，因为视频可认为是二维图片在时间维度上的叠加。