1 特殊应用

1.1 人脸识别

人脸验证（Face Verification）和**人脸识别（Face Recognition）**的区别：

- 人脸验证：一般指一个一对一问题，只需要验证输入的人脸图像是否与某个已知的身份信息对应；
- 人脸识别：一个更为复杂的一对多问题，需要验证输入的人脸图像是否与多个已知身份信息中的某一个匹配。

一般来说，由于需要匹配的身份信息更多导致错误率增加，人脸识别比人脸验证更难一些。

1.1.1 One-Shot 学习

人脸识别所面临的一个挑战是要求系统只采集某人的一个面部样本，就能快速准确地识别出这个人，即只用一个训练样本来获得准确的预测结果。这被称为 **One-Shot** 学习。

有一种方法是假设数据库中存有 N 个人的身份信息，对于每张输入图像，用 Softmax 输出 N+1 种标签，分别对应每个人以及都不是。然而这种方法的实际效果很差，因为过小的训练集不足以训练出一个稳健的神经网络；并且如果有新的身份信息入库，需要重新训练神经网络，不够灵活。

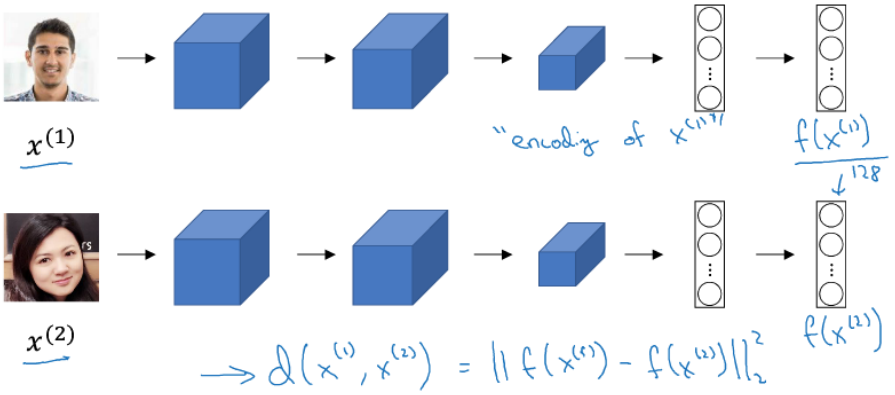
因此，我们通过学习一个 Similarity 函数来实现 One-Shot 学习过程。Similarity 函数定义了输入的两幅图像的差异度，其公式如下：

$$Similarity = d(img1, img2) \tag{1}$$

可设置一个超参数 τ，当 $d(img1, img2) \leq \tau$，则两幅图片为同一人，否则为不同

1.1.2 Siamese 网络

实现 Similarity 函数的一种方式是使用 Siamese 网络，它是一种对两个不同输入运行相同的卷积网络，然后对它们的结果进行比较的神经网络。

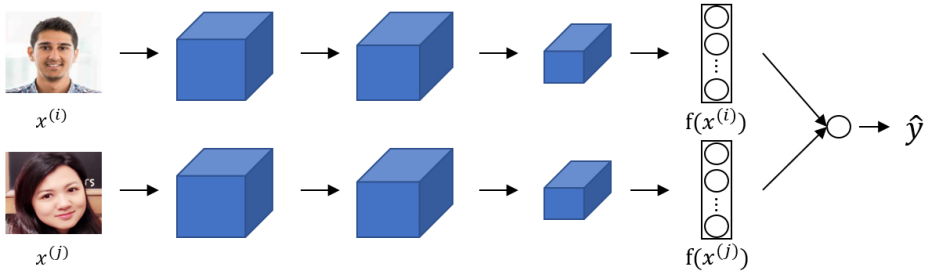


上图的示例中，将图片 $x^{(1)}$、$x^{(2)}$ 分别输入两个相同的卷积网络中，经过全连接后不再进行 Softmax，得到它们的特征向量 $f(x^{(1)})$、$f(x^{(2)})$。此时 Similarity 函数就被定义为这两个特征向量之差的 2 范数：

$$d(x^{(1)}, x^{(2)}) = || f(x^{(1)}) - f(x^{(2)}) ||_2^2 \tag{2}$$

二分类结构

利用一对相同的 Siamese 网络，可以将人脸验证看作二分类问题。



如上图中，输入的两张图片 $x^{(i)}$、$x^{(j)}$，经过卷积网络后分别得到m维的特征向量 $f(x^{(i)})$、$f(x^{(j)})$，将它们输入一个逻辑回归单元，最后输出的预测结果中用 1 和 0 表示相同或不同的人。

其中对最后的输出结果 $\hat{y}$，如果使用的逻辑回归单元是 sigmoid 函数，表达式就会是：

$$\hat{y} = \sigma(\sum_{k=1}^{K} w_k |f(x^{(i)})_k - x^{(j)})_k| + b) \tag{3}$$

其中，w_k 和 b 都是通过梯度下降算法迭代训练得到的参数。上述计算表达式也可以用另一种表达式代替：

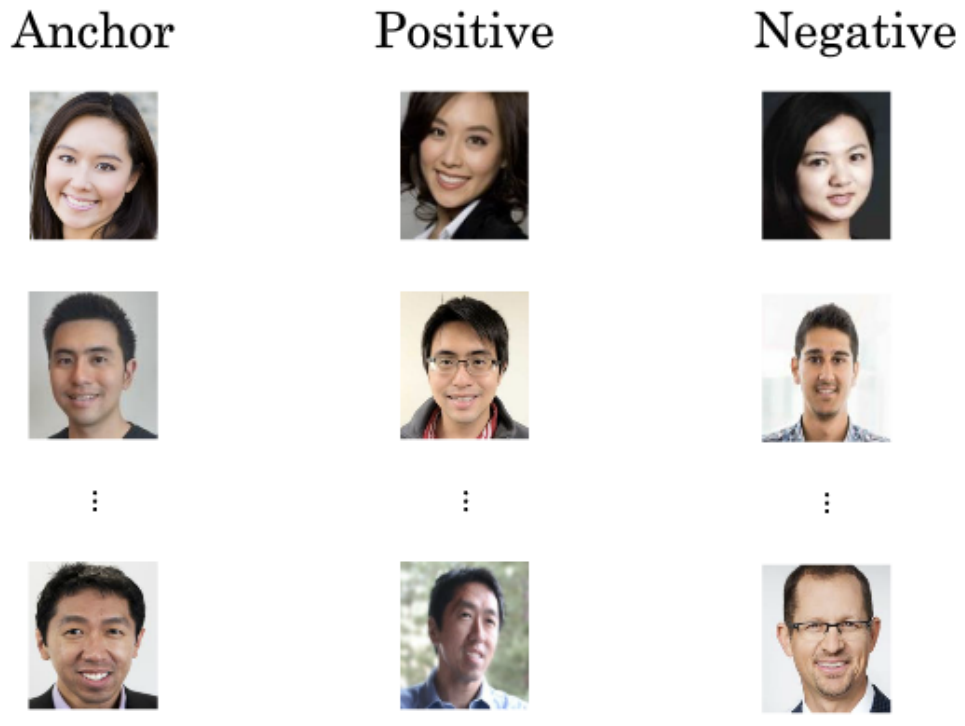
$$\hat{y} = \sigma(\sum_{k=1}^{K} w_k \frac{(f(x^{(i)})_k - f(x^{(j)})_k)^2}{f(x^{(i)})_k + f(x^{(j)})_k} + b) \tag{4}$$

其中，$\frac{(f(x^{(i)})_k - f(x^{(j)})_k)^2}{f(x^{(i)})_k + f(x^{(j)})_k}$ 被称为 χ 方相似度。

以上所述内容，都来自 Taigman 等人 2014 年发表的论文 DeepFace closing the gap to human level performance (http://www.cs.wayne.edu/~mdong/taigman_cvpr14.pdf) 中提出的 DeepFace。

1.1.3 Triplet 损失

Triplet 损失函数用于训练出合适的参数，以获得高质量的人脸图像编码。“Triplet”一词来源于训练这个神经网络需要大量包含 Anchor（靶目标）、Positive（正例）、Negative（反例）的图片组，其中 Anchor 和 Positive 需要是同一个人的人脸图像。



对于这三张图片，应该有：

$$||f(A) - f(P)||_2^2 + \alpha \leq ||f(A) - f(N)||_2^2 \tag{5}$$

其中，α 被称为间隔（margin），用于确保 $f()$ 不会总是输出零向量（或者一个恒定的值）。那么 Triplet 损失函数的定义：

$$L(A, P, N) = max(||f(A) - f(P)||_2^2 - ||f(A) - f(N)||_2^2 + \alpha, 0) \tag{6}$$

其中，因为 $||f(A) - f(P)||_2^2 - ||f(A) - f(N)||_2^2 + \alpha$的值需要小于等于 0，因此取它和 0 的更大值。

这样，训练这个神经网络就需要有大量经过特定组合的包含Anchor、Postive、Negative的图片组。且使用m个训练样本，代价函数将是：

$$\begin{aligned} J &= \sum_{i=1}^{m} L(A^{(i)}, P^{(i)}, N^{(i)}) \\ &= \sum_{i=1}^{m} [|| f(A^{(i)}) - f(P^{(i)}) ||_2^2 - || f(A^{(i)}) - f(N^{(i)}) ||_2^2 + \alpha] \end{aligned} \tag{7}$$

通过梯度下降最小化代价函数。

在选择训练样本时，随机选择容易使 Anchor 和 Positive 极为接近，而 Anchor 和 Negative 相差较大，以致训练出来的模型容易抓不到关键的区别。因此，最好的做法是人为增加 Anchor 和 Positive 的区别，缩小 Anchor 和 Negative 的区别，促使模型去学习不同人脸之间的关键差异。

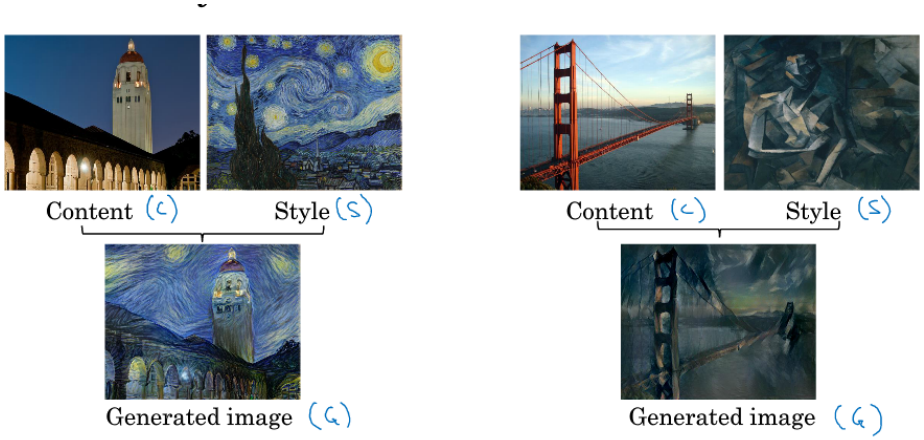
Triplet 损失的相关内容来自 Schroff 等人 2015 年在论文 FaceNet: A unified embedding for face recognition and clustering (https://arxiv.org/pdf/1503.03832.pdf) 中提出的 FaceNet，更细节可以参考论文内容。

注意

无论是对于使用 Triplet 损失函数的网络，还是二分类结构，为了减少计算量，可以提前计算好编码输出 $f(x)$ 并保存。这样就不必存储原始图片，并且每次进行人脸识别时只需要计算测试图片的编码输出。

1.2 神经风格转换

神经风格迁移（Neural Style Tranfer）是将参考风格图像的风格转换到另一个输入图像中，如下图所示。

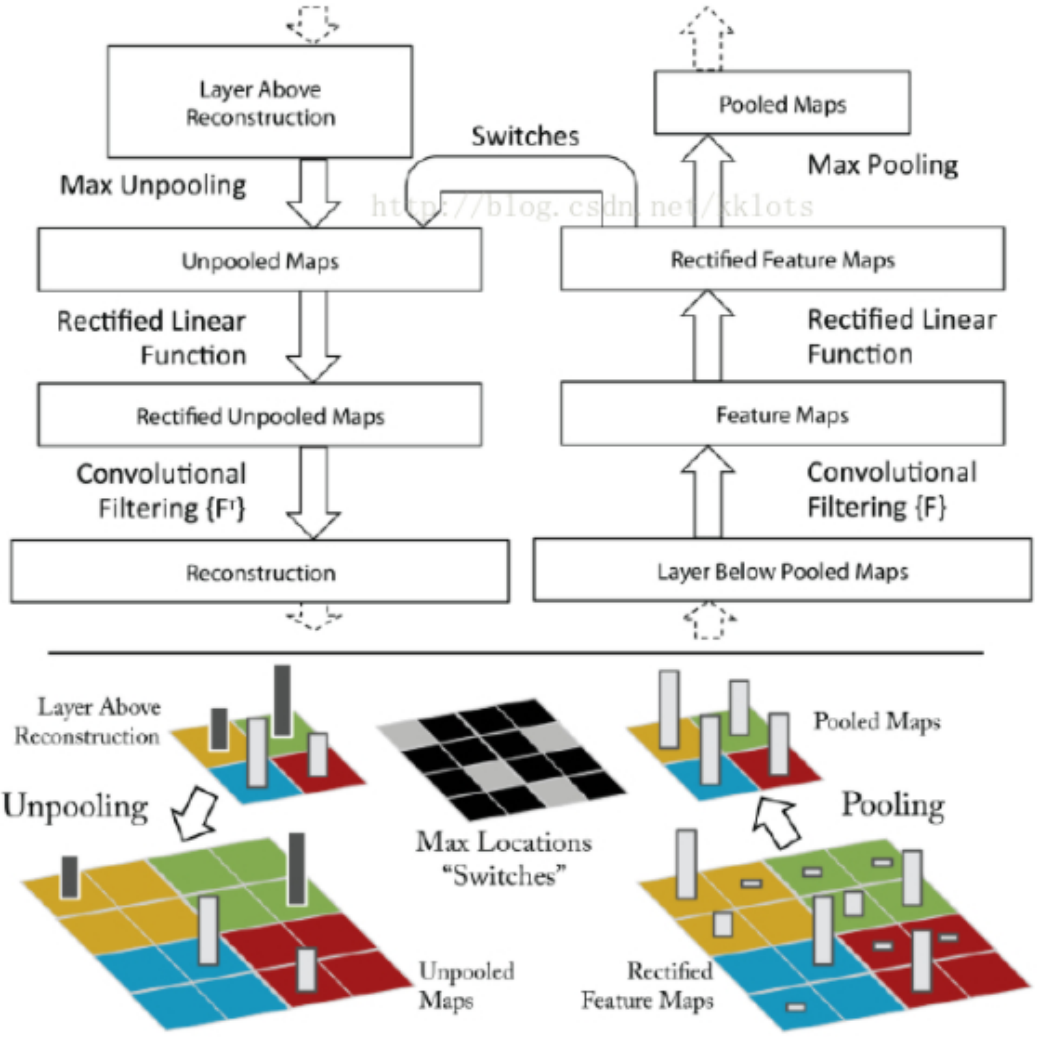


其中待转换的图片标识为 C（Content），某种风格的图片为 S（Style），转换后的图片为 G（Generated）。

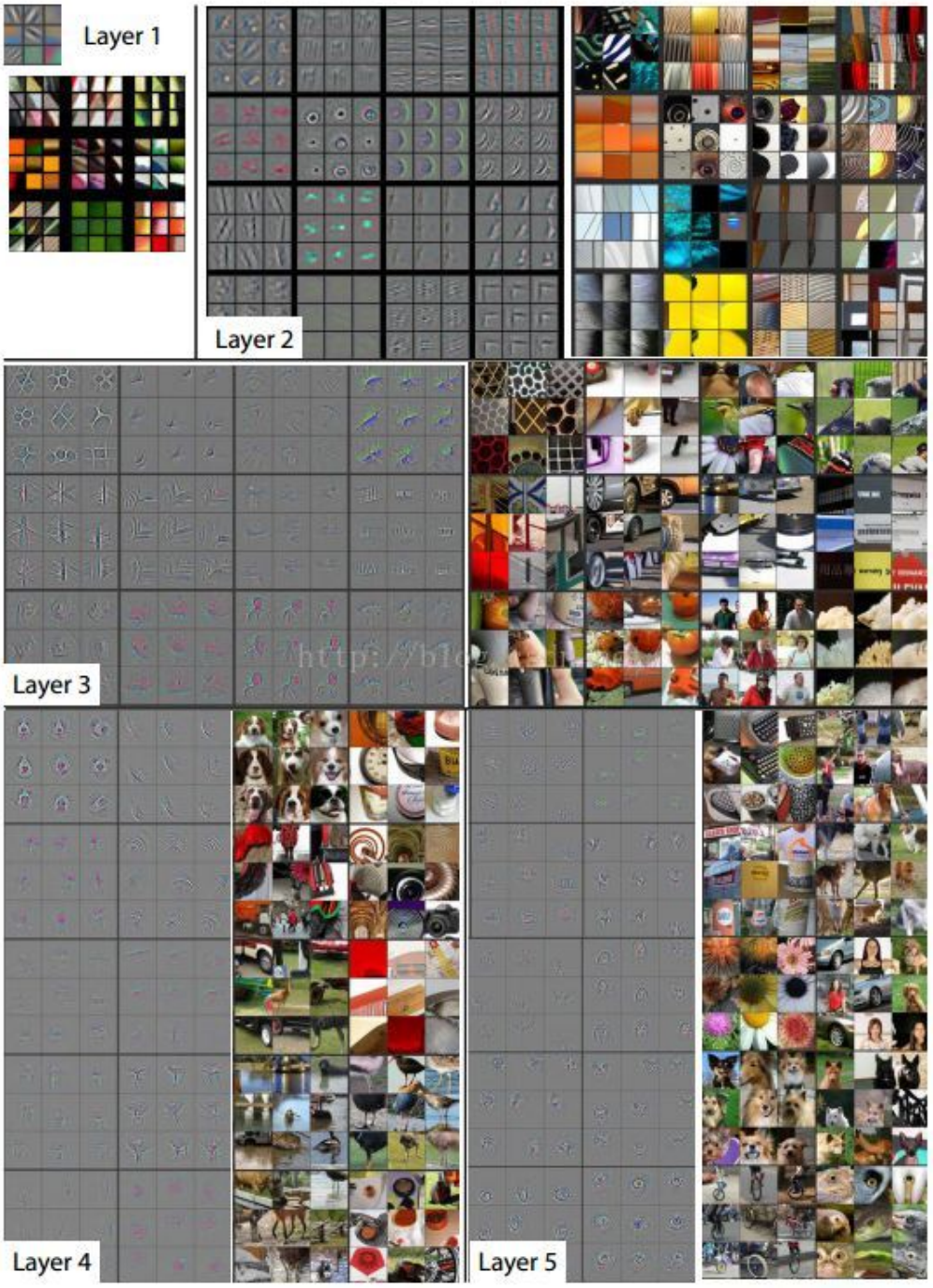
1.2.1 理解 CNN（可视化特征）

要理解利用卷积网络实现神经风格转换的原理，首先要理解在输入图像数据后，一个深度卷积网络从中都学到了些什么。

2013 年 Zeiler 和 Fergus 在论文 Visualizing and understanding convolutional networks (https://arxiv.org/pdf/1311.2901.pdf) 中提出了一种将卷积神经网络的隐藏层特征进行可视化的方法。



上图展示是一个 AlexNet 中的卷积、池化以及最后的归一化过程，以及实现隐藏层可视化的反卷积网络中的 Unpooling、矫正以及反卷积过程。论文中将 ImageNet 2012 中的 130 万张图片作为训练集，训练结束后提取到的各个隐藏层特征如下图：



从中可以看出，浅层的隐藏单元通常学习到的是边缘、颜色等简单特征，越往深层，隐藏单元学习到的特征也越来越复杂。

1.2.2 代价函数

实现神经风格转换，需要定义一个关于生成的图像 G 的代价函数 $J(G)$，以此评判生成图像的好坏的同时，用梯度下降法最小化这个代价函数，而生成最终的图像。

神经风格迁移生成图片 G 的代价函数如下：

$$J(G) = \alpha \cdot J_{content}(C, G) + \beta \cdot J_{style}(S, G) \tag{8}$$

其中内容代价函数 $Jcontent(C, G)$ 度量待转换的 C 和生成的 G 的相似度，风格代价函数 $Jstyle(S, G)$ 则度量某风格的 S 和生成的 G 的相似度，α、β 是用于控制相似度比重的超参数。

神经风格迁移的算法步骤如下：

- 随机生成图片 G 的所有像素点；
- 使用梯度下降算法使代价函数最小化，以不断修正 G 的所有像素点。

1.2.2.1 内容代价函数

$J_{content}(C,G)$ 的计算过程如下：

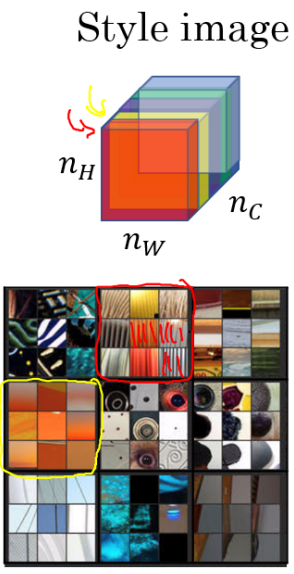
- 使用一个预训练好的 CNN（例如 VGG）；
- 选择一个隐藏层 l 来计算内容代价。l 太小则内容图片和生成图片像素级别相似，l 太大则可能只有具体物体级别的相似。因此，l 一般选一个中间层；
- 设 $a^{(C)[l]}$、$a^{(G)[l]}$ 为 C 和 G 在 l 层的激活，则有：

$$J_{content}(C,G) = \frac{1}{2}||(a^{(C)[l]} - a^{(G)[l]})||^2 \tag{9}$$

$a^{(C)[l]}$ 和 $a^{(G)[l]}$ 越相似，则 $Jcontent(C,G)$ 越小。

1.2.2.2 风格代价函数

定义风格代价函数 $Jstyle(S,G)$ 前，首先提取出 S 的“风格”。通过之前的理解 CNN 内容，将 S 也输入那个预先训练好的卷积神经网络中，就可以将其所谓的“风格”定义为神经网络中某一层或者几个层中，各个通道的激活项之间的相关系数。如下图所示为网络中的某一层，假设其中前两个红、黄色通道分别检测出了下面对于颜色圈出的特征，则这两个通道的相关系数，就反映出了该图像所具有的“风格”。



更进一步解释为：每个通道提取图片的特征不同，比如标为红色的通道提取的是图片的垂直纹理特征，标为黄色的通道提取的是图片的橙色背景特征。那么计算这两个通道的相关性，相关性的大小，即表示原始图片既包含了垂直纹理也包含了该橙色背景的可能性大小。通过 CNN，“风格”被定义为同一个隐藏层不同通道之间激活值的相关系数，因其反映了原始图片特征间的相互关系。

对于风格图像 S，选定网络中的第 l 层，则相关系数以一个 gram 矩阵的形式表示：

$$G^{[l](S)}_{kk'} = \sum_{i=1}^{n_H^{[l]}} \sum_{j=1}^{n_W^{[l]}} a^{[l](S)}_{ijk} a^{[l](S)}_{ijk'} \tag{10}$$

其中，i 和 j 为第 l 层的高度和宽度；k 和 k' 为选定的通道，其范围为 1 到 $n_C^{[l]}$；$a^{[l](S)}_{ijk}$ 为激活。

同理，对于生成图像 G，有：

$$G^{[l](G)}_{kk'} = \sum_{i=1}^{n_H^{[l]}} \sum_{j=1}^{n_W^{[l]}} a^{[l](G)}_{ijk} a^{[l](G)}_{ijk'} \tag{11}$$

因此，第 l 层的风格代价函数为：

$$J_{style}^{[l]}(S,G) = \frac{1}{(2n_H^{[l]}n_W^{[l]}n_C^{[l]})^2}\sum_k\sum_{k'}(G_{kk'}^{[l](S)} - G_{kk'}^{[l](G)})^2 \tag{12}$$

如果对各层都使用风格代价函数，效果会更好。因此有：

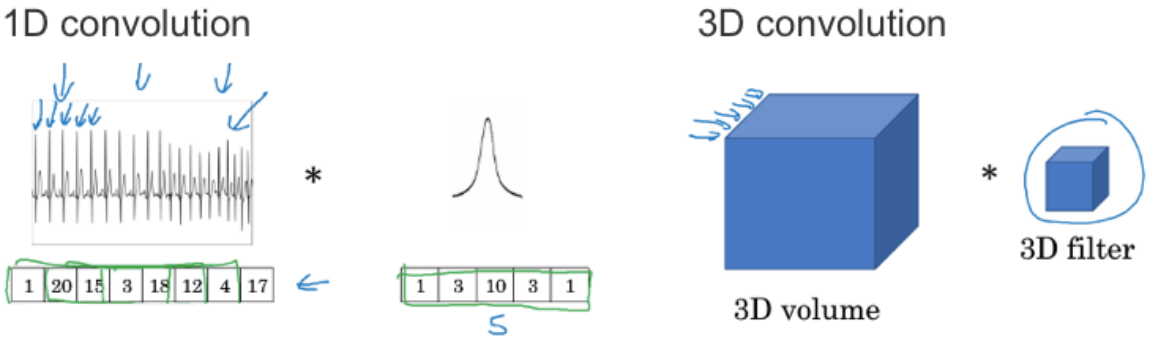
$$J_{style}(S,G) = \sum_l \lambda^{[l]} J_{style}^{[l]}(S,G) \tag{13}$$

其中，λ 是用于设置不同层所占权重的超参数。

这样一种神经风格转换的实现方法，来自 2015 年 Gatys 等人发表的论文 A Neural Algorithm of Artistic Style (https://arxiv.org/abs/1508.06576)。

1.3 扩展至一维和三维

之前我们处理的都是二维图片，实际上卷积也可以延伸到一维和三维数据。我们举两个示例来说明。



EKG 数据（心电图）是由时间序列对应的每个瞬间的电压组成，是一维数据。一般来说我们会用 RNN（循环神经网络）来处理，不过如果用卷积处理，则有：

- 输入时间序列维度：14 x 1
- 滤波器尺寸：5 x 1，滤波器个数：16
- 输出时间序列维度：10 x 16

而对于三维图片的示例，有

- 输入 3D 图片维度：14 x 14 x 14 x 1
- 滤波器尺寸：5 x 5 x 5 x 1，滤波器个数：16
- 输出 3D 图片维度：10 x 10 x 10 x 16