深度学习-人脸识别和风格迁移

黄海广 副教授

2022年04月

本章目录

1.人脸识别概述

01 人脸识别概述

02 神经风格迁移

1.人脸识别概述

人脸验证（face verification）

- 输入图片，以及某人的ID或者是名字
- 验证输入图片是否是这个人

人脸识别（face recognition）

- 有一个K个人的人脸数据库
- 获取输入图像
- 如果图像是K个人中的某人（或不认识）

人脸聚类（Face Clustering）

在数据库中对人脸进行聚类，直接K-Means即可。

1.人脸识别概述

人脸检测的步骤

- **人脸定位**

确定是否存在人脸,人脸存在的位置、范围等

- **人脸对齐**

把众多人脸图像转换到一个统一角度和姿势

- **确定关键点**

关键点包括:眼角、鼻尖、嘴角等

1.人脸识别概述

人脸检测常用算法(深度学习框架)

- MTCNN算法
- HR
- Face r-CNN
- PyramidBox

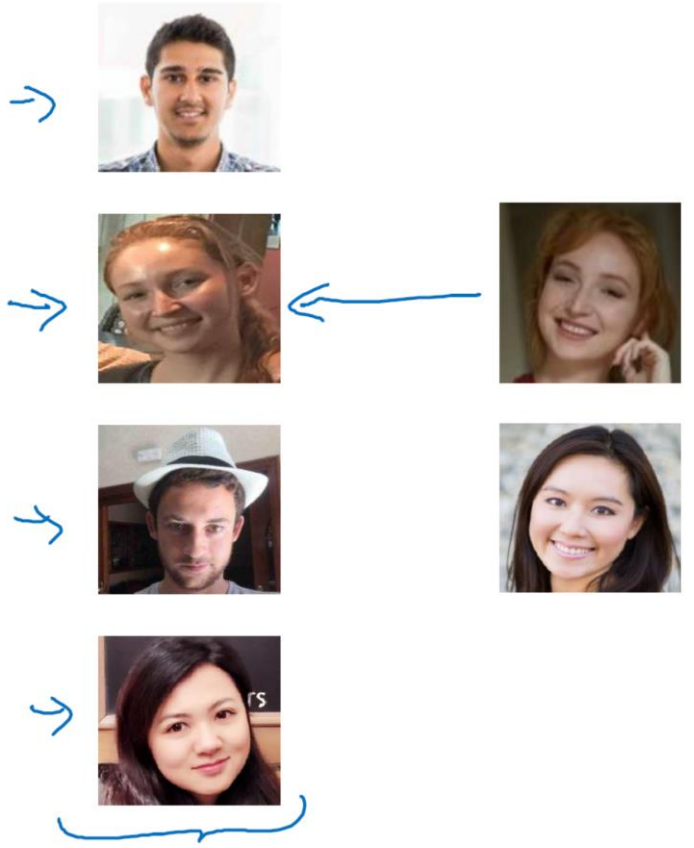
1.人脸识别概述

One-Shot学习

在一次学习问题中，只能通过一个样本进行学习，以能够认出同一个人。大多数人脸识别系统都需要解决这个问题。系统需要做的就是，仅仅通过一张已有的照片，来识别前面这个人确实是她。相反，如果机器看到一个不在数据库里的人所示），机器应该能分辨出她不是数据库中四个人之一。

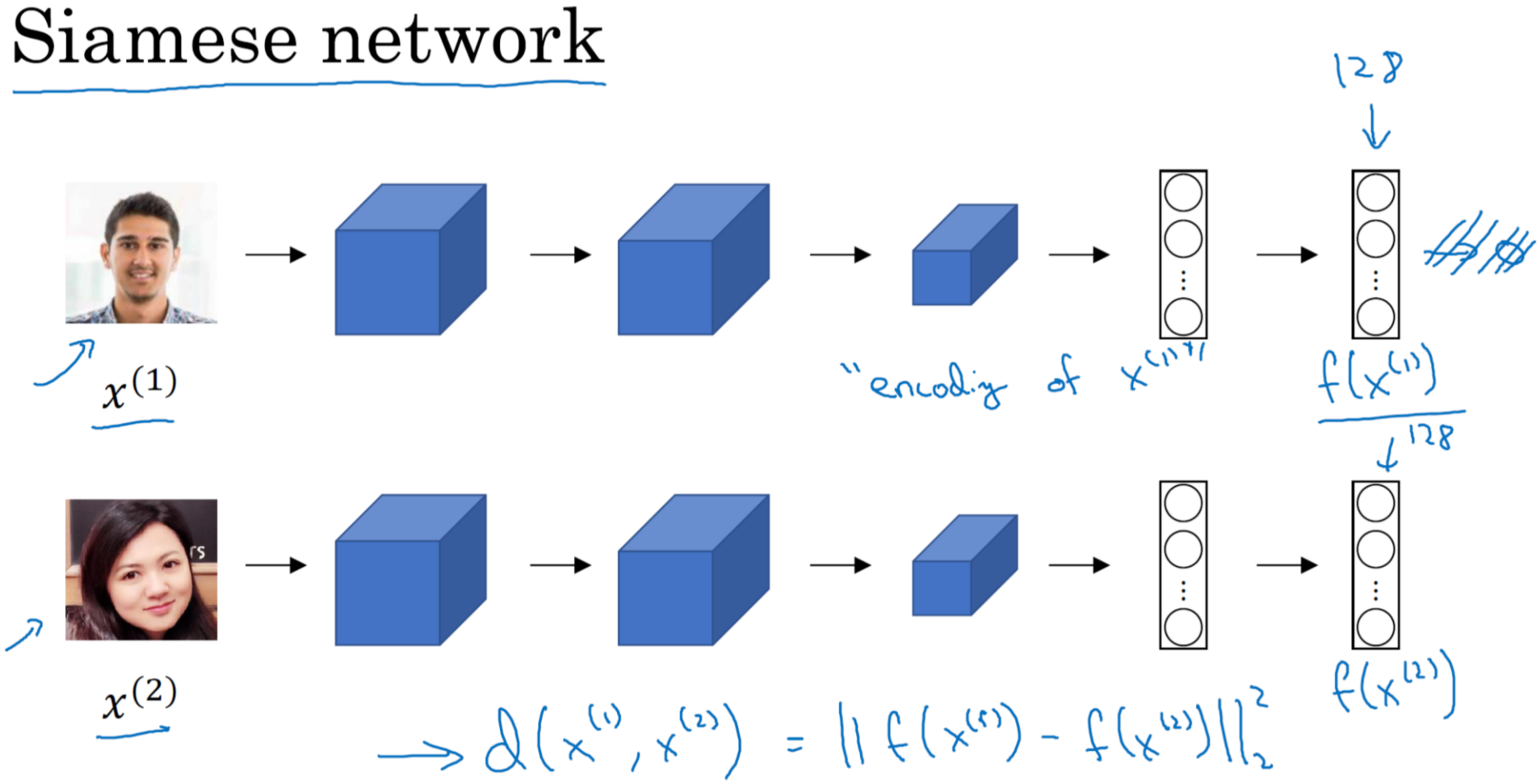
$$d(img1, img2) = degree\ of\ difference\ between\ images$$

只要你能学习这个函数d，通过输入一对图片，它将会告诉你这两张图片是否是同一个人。



1.人脸识别概述

Siamese 网络



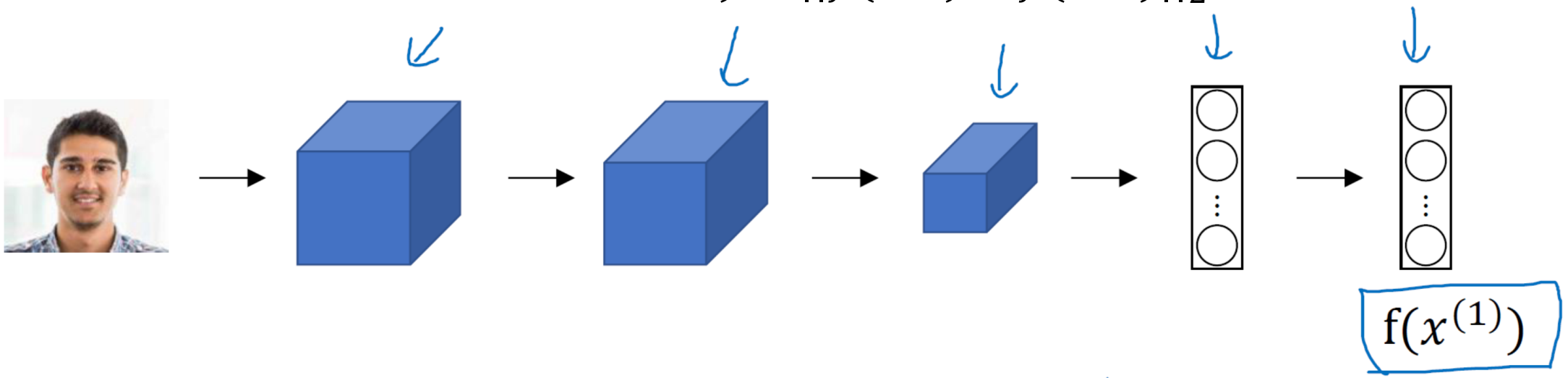
1.人脸识别概述

Siamese 网络

$$d(x^{(i)}, x^{(j)}) = ||f(x^{(i)}) - f(x^{(j)})||_2^2。$$

如果$x^{(i)}, x^{(j)}$是同一个人，则$) = ||f(x^{(i)}) - f(x^{(j)})||_2^2$较小

如果$x^{(i)}, x^{(j)}$不是同一个人，则$) = ||f(x^{(i)}) - f(x^{(j)})||_2^2$较大



1.人脸识别概述

Triplet 损失

三元组损失，它代表你通常会同时看三张图片，你需要看**Anchor**图片、**Postive**图片，还有**Negative**图片，我要把**Anchor**图片、**Positive**图片和**Negative**图片简写成A、P、N。

1.人脸识别概述

Triplet 损失

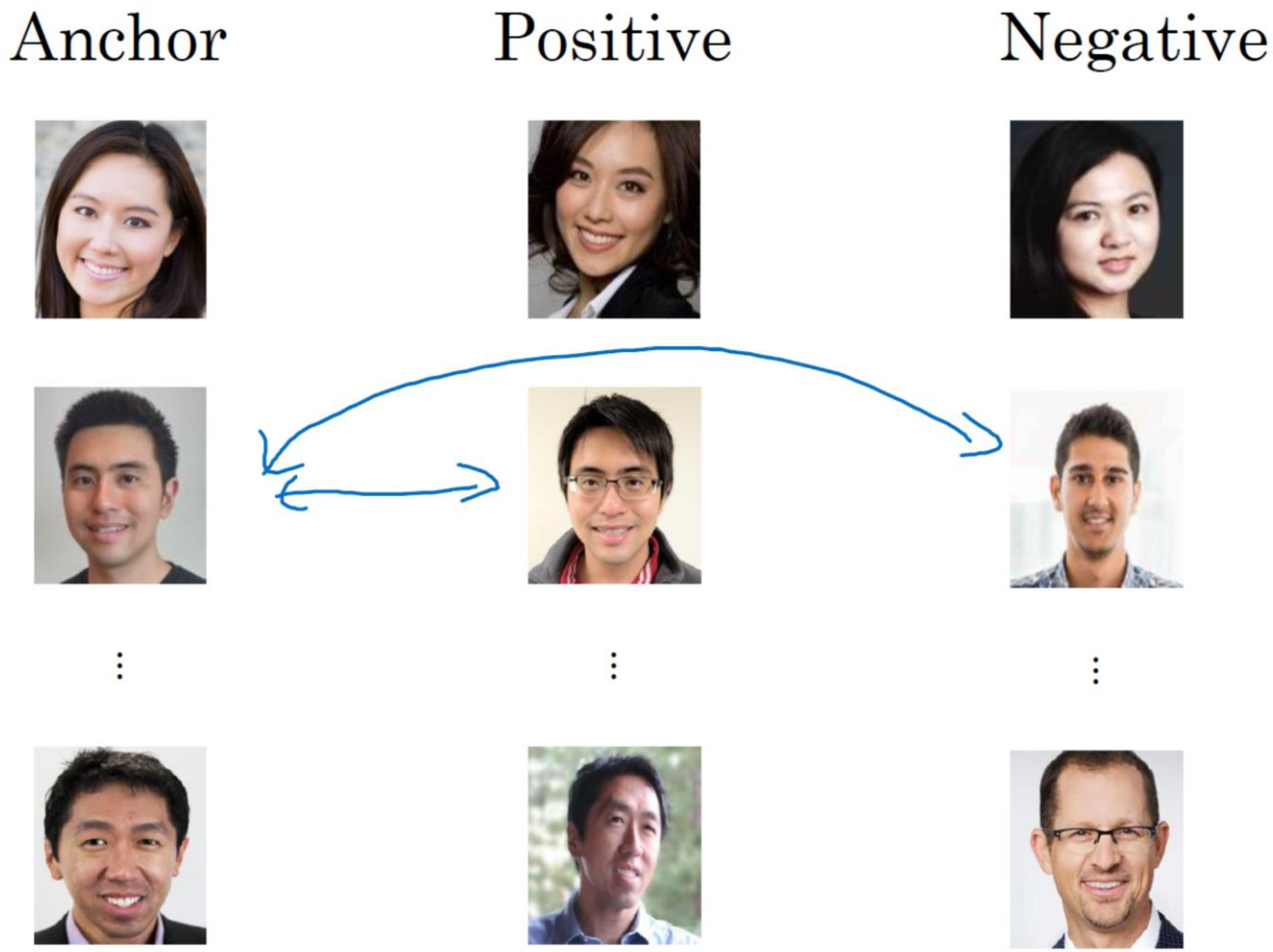
想要$||f(A)-f(P)||^2$，你希望这个数值很小，准确地说，你想让它小于等$f(A)$和$f(N)$之间的距离，或者说是它们的范数的平方（即：$||f(A)-f(P)||^2 \leq ||f(A)-f(N)||^2$）。

$(||f(A)-f(P)||^2)$就是$d(A,P)$，

$(||f(A)-f(N)||^2)$这是$d(A,N)$，你可以把d看作是距离(**distance**)函数，这也是为什么我们把它命名为d。

1.人脸识别概述

Triplet 损失



1.人脸识别概述

Triplet 损失

损失函数的定义基于三元图片组

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

为了定义这个损失函数，我们取这个和0的最大值：

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

假设有1000个不同的人的10000张照片，也就是这1000个人平均每个人10张照片，组成了整个数据集。

如果每个人只有1张照片，那么根本没法训练这个系统。

1.人脸识别概述

Triplet 损失

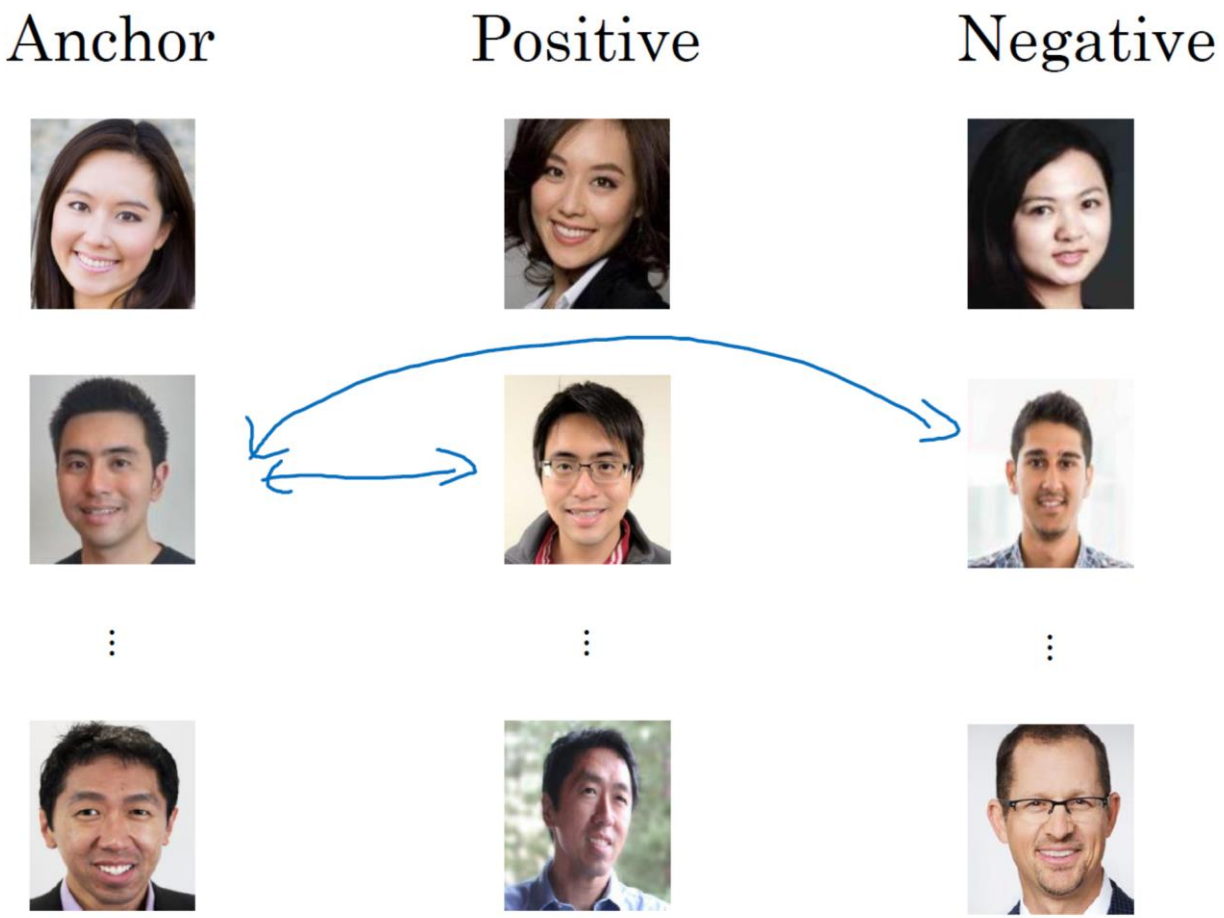
为了构建一个数据集，你要做的就是尽可能选择难训练的三元组A、P和N。具体而言，你想要所有的三元组都满足这个条件

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

学习算法会尽可能地使右边这个式子变大（$d(A,N)$），或者使左边这个式子（$d(A,P)$）变小，这样左右两边至少有一个a的间隔。

1.人脸识别概述

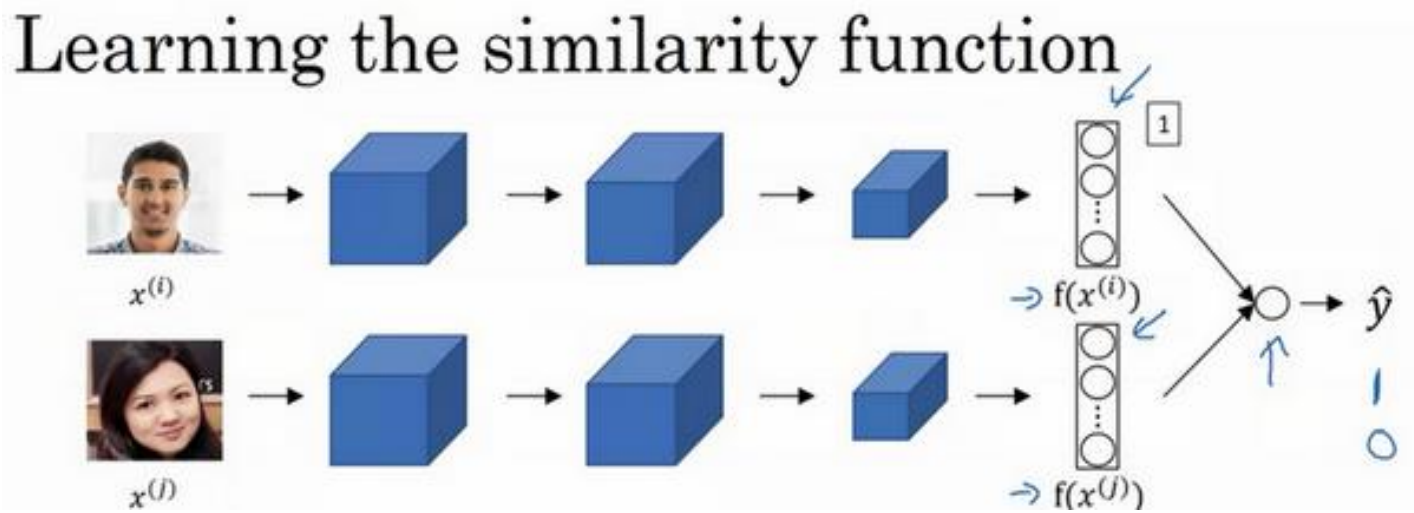
用Triplet 损失训练



1.人脸识别概述

人脸识别与二分类

$$\hat{y} = \sigma(\sum_{k=1}^{128} w_i |f(x^{(i)})_k - f(x^{(j)})_k| + b)$$



符号$f(x^{(i)})_k$代表图片$x^{(i)}$的编码，下标k代表选择这个向量中的第k个元素，

$|f(x^{(i)})_k - f(x^{(j)})_k|$对这两个编码取元素差的绝对值

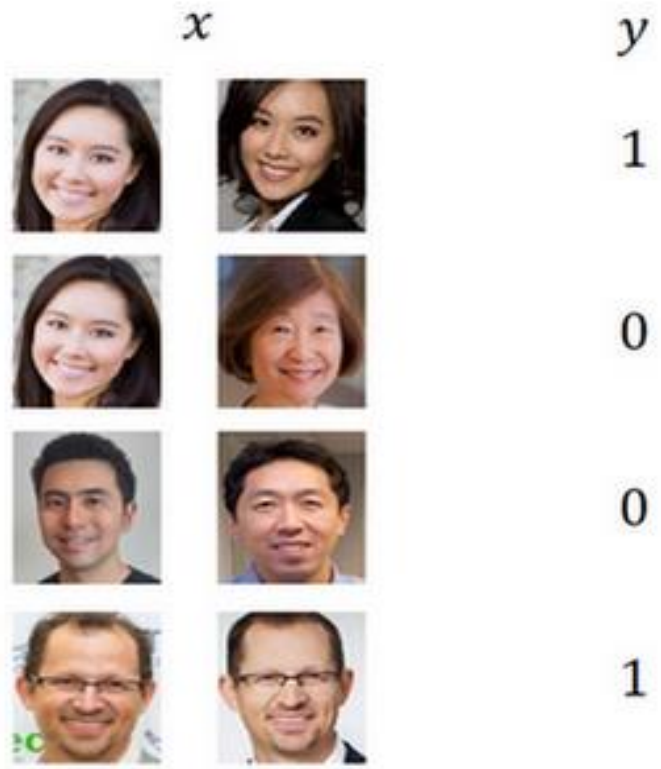
χ^2公式，公式可以是$\chi^2 = \frac{(f(x^{(i)})_k - f(x^{(j)})_k)^2}{f(x^{(i)})_k + f(x^{(j)})_k}$

1.人脸识别概述

用Triplet 损失训练

$$\hat{y} = \sigma(\sum_{k=1}^{128} w_i |f(x^{(i)})_k - f(x^{(j)})_k| + b)$$

我解释一下，符号$f(x^{(i)})_k$代表图片$x^{(i)}$的编码，下标k代表选

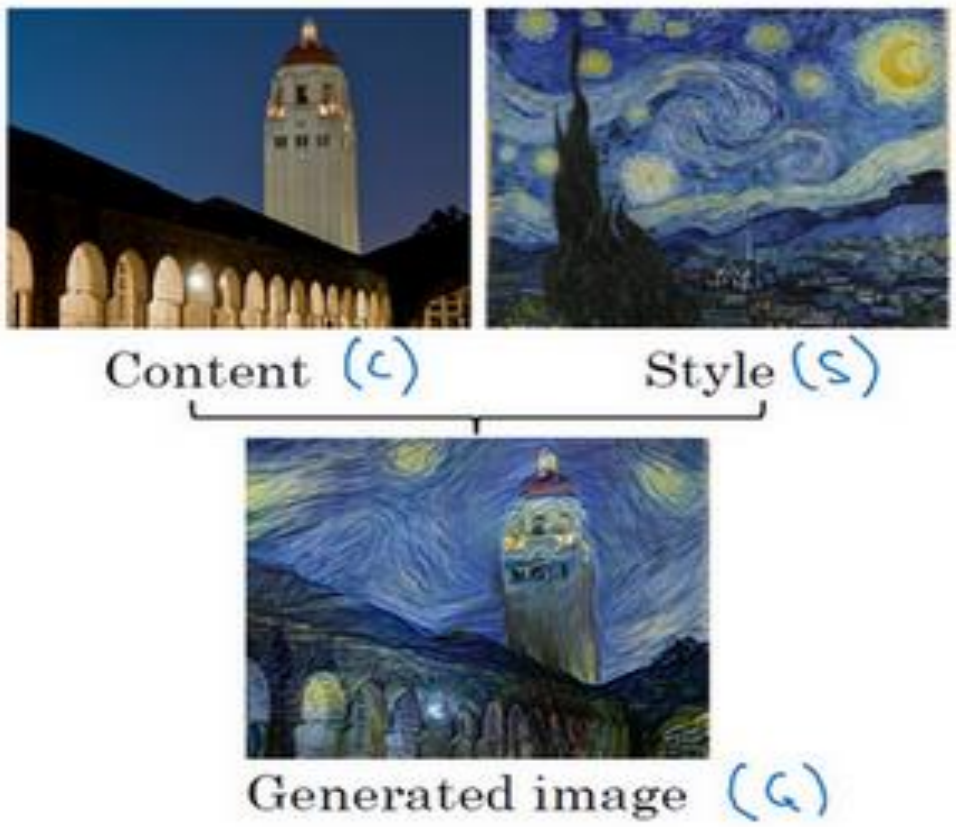


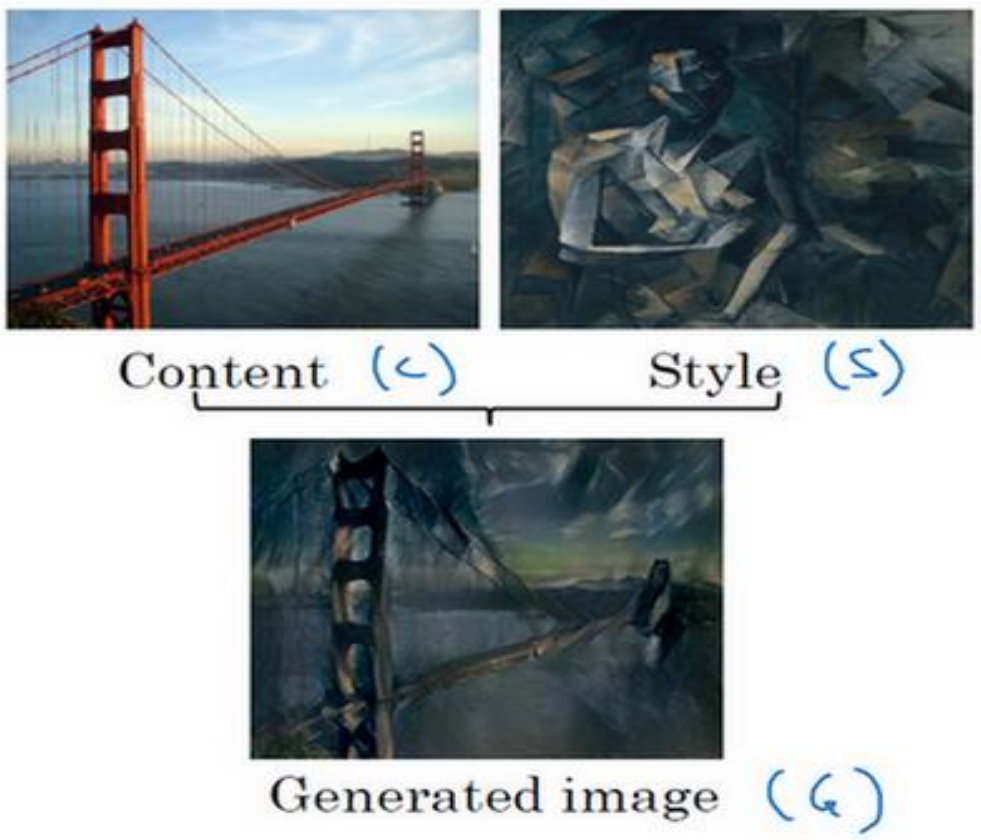
2.神经风格迁移

01 人脸识别概述

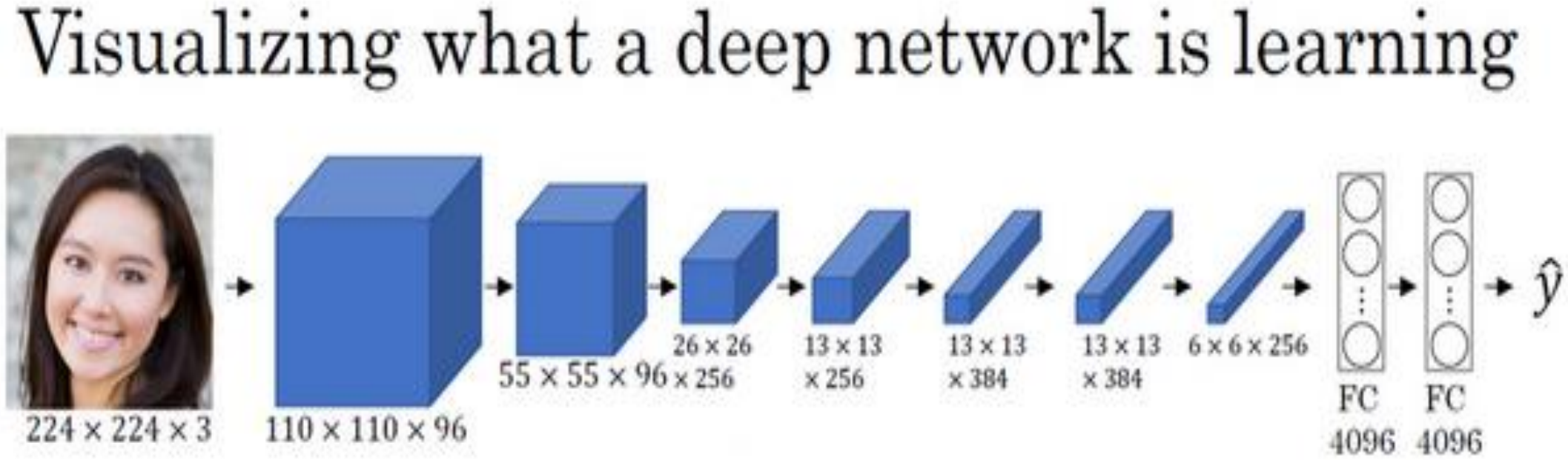
02 神经风格迁移

2.神经风格迁移


Content (c)
Style (s)
Generated image (g)

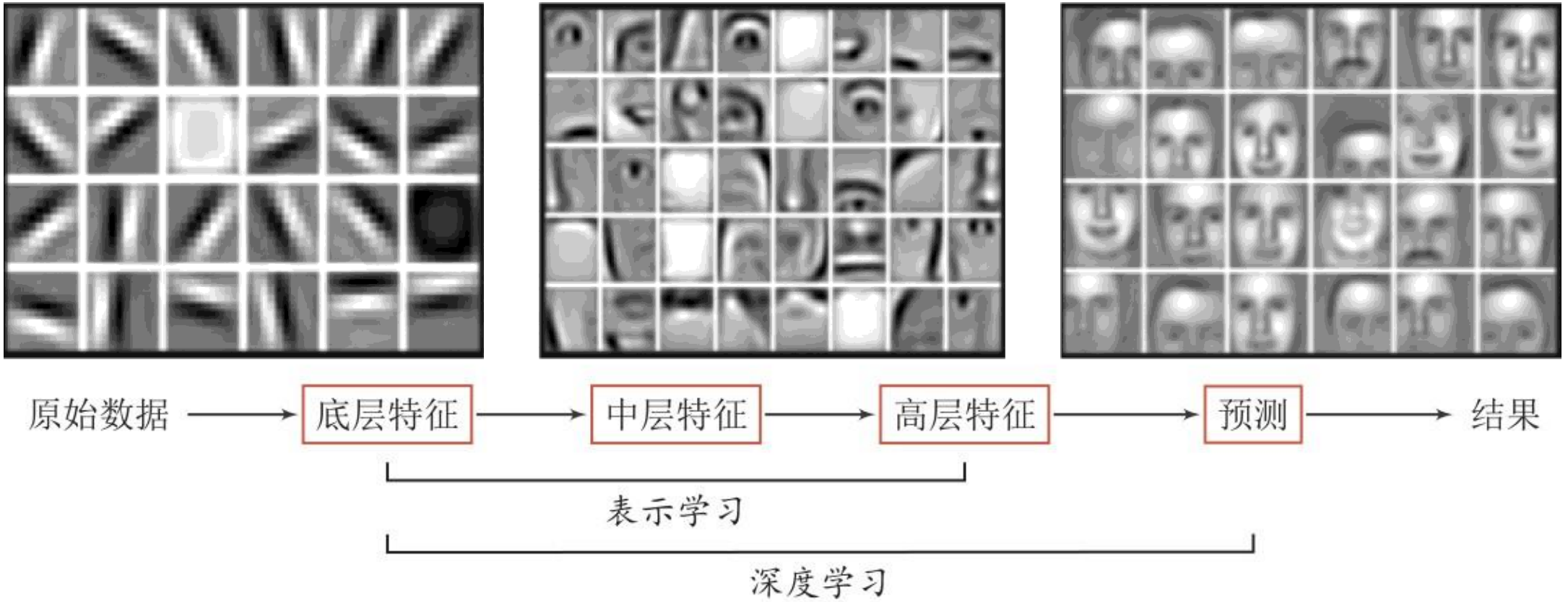

Content (c)
Style (s)
Generated image (g)

2.神经风格迁移


Visualizing what a deep network is learning
224 × 224 × 3
110 × 110 × 96
55 × 55 × 96
26 × 26 × 256
13 × 13 × 256
13 × 13 × 384
13 × 13 × 384
6 × 6 × 256
FC 4096
FC 4096
ŷ

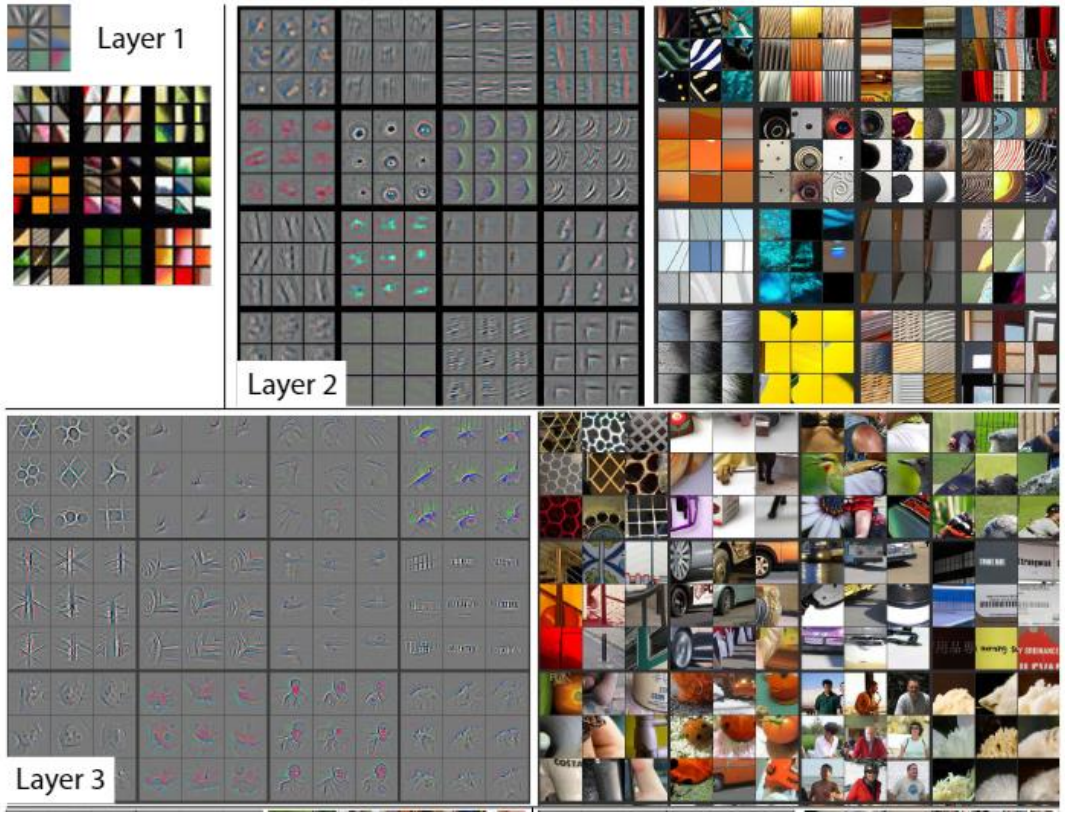
2.神经风格迁移

深度学习=表示学习+浅层学习

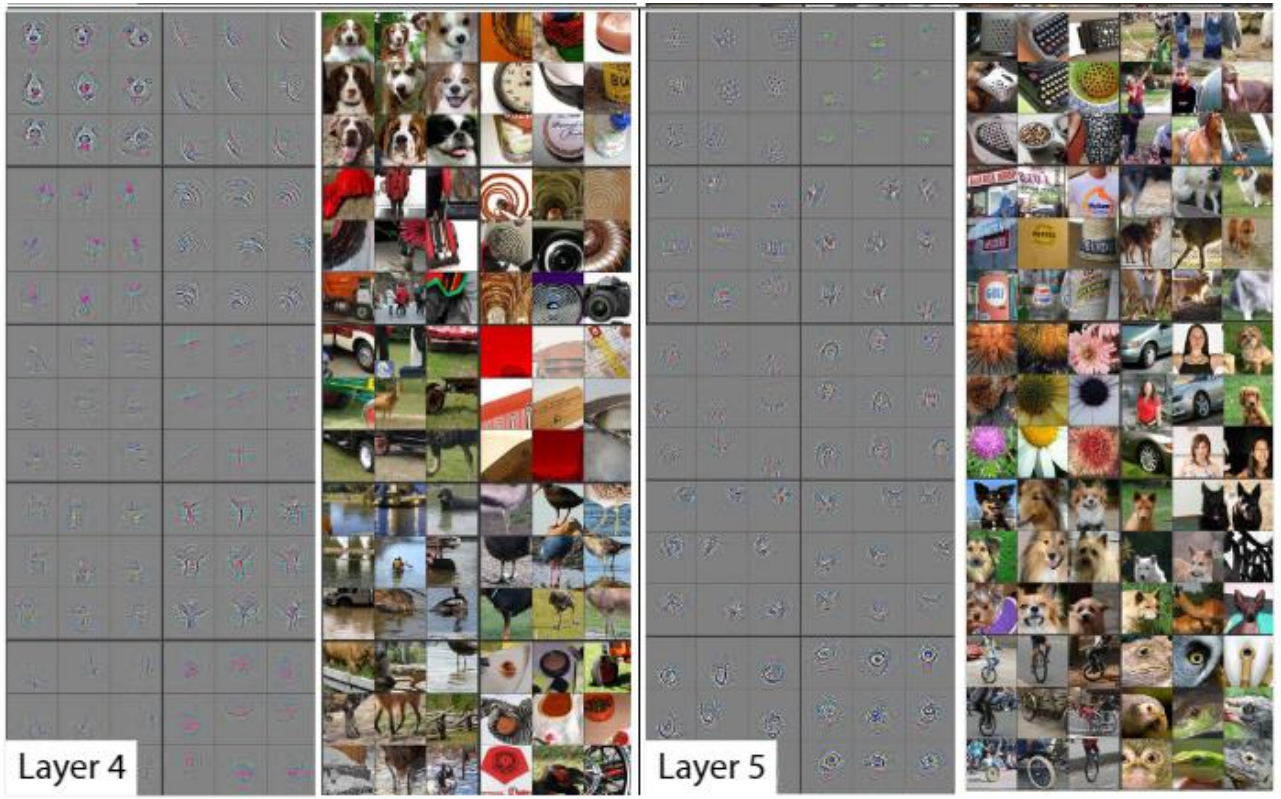


多层卷积能抽取复杂特征

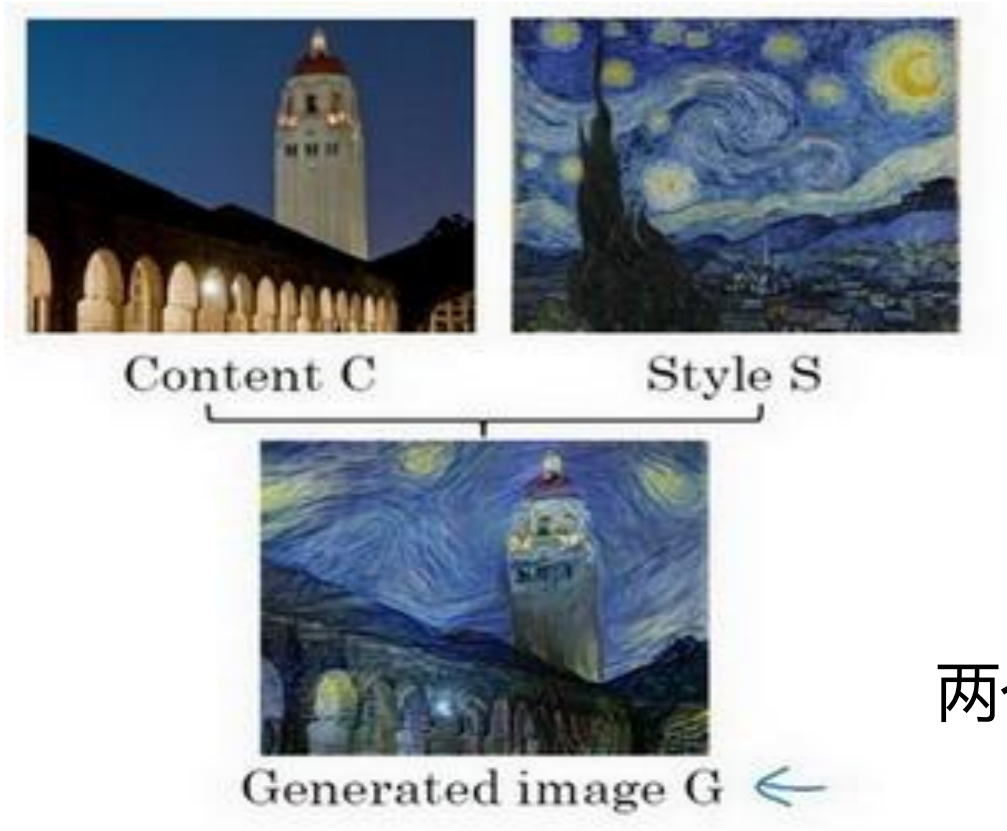
浅层学到的特征为简单的边缘、角点、纹理、几何形状、表面等



深层学到的特征则更为复杂抽象，为狗、人脸、键盘等等



2.神经风格迁移

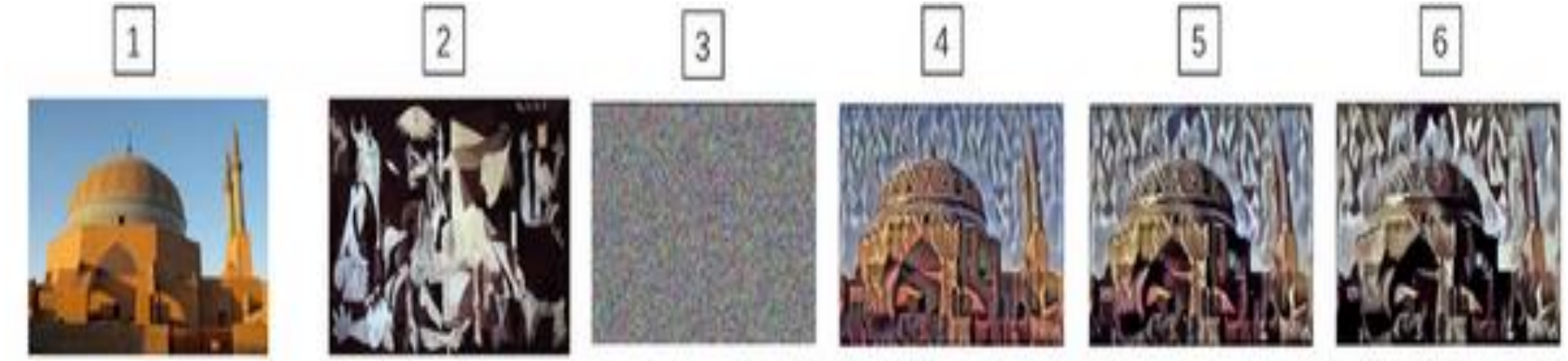


给你一个内容图像C，给定一个风格图片S，而你的目标是生成一个新图片G

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

两个超参数a和β来确定内容代价和风格代价

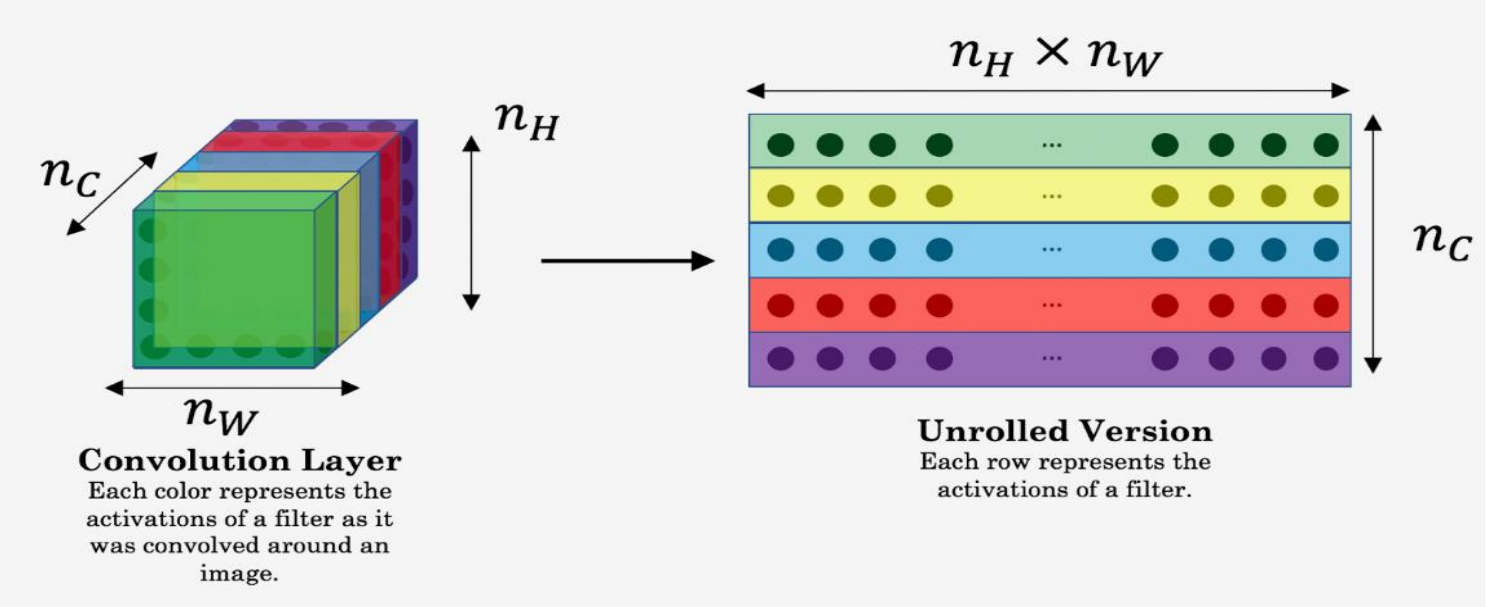
2.神经风格迁移



- 随机初始化生成图像G，如100×100×3，500×500×3，又或者是任何你想要的尺寸。
- 然后使用代价函数$J(G)$，使用梯度下降的方法将其最小化，更新$G := G - \frac{\partial}{\partial G}J(G)$。在这个步骤中，你实际上更新的是图像$G$的像素值，也就是100×100×3，比如**RGB**通道的图片。

2.神经风格迁移

内容代价函数（Content cost function）



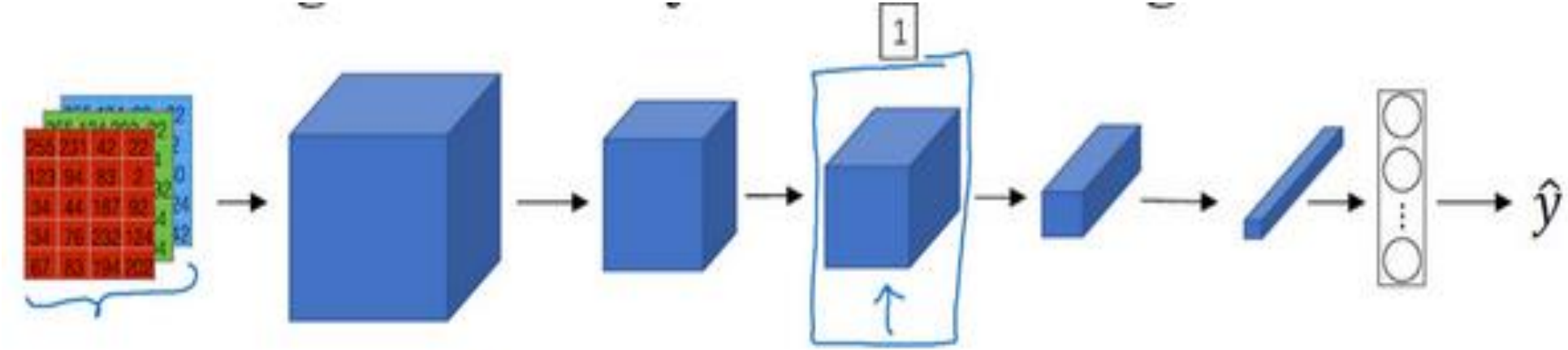
- Say you use hidden layer l to compute content cost.
- Use pre-trained ConvNet. (E.g., VGG network)

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

两个超参数a和β来确定内容代价和风格代价

2.神经风格迁移

风格代价函数（Style cost function）



Say you are using layer l's activation to measure "style."

2.神经风格迁移

content image

louvre museum

+

style image

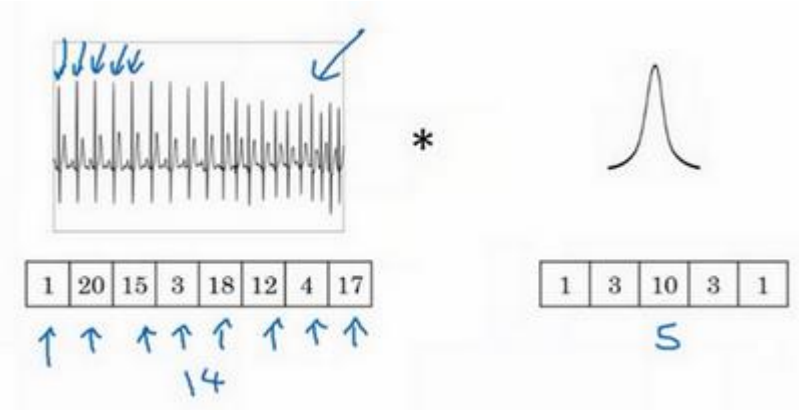
impressionist style painting

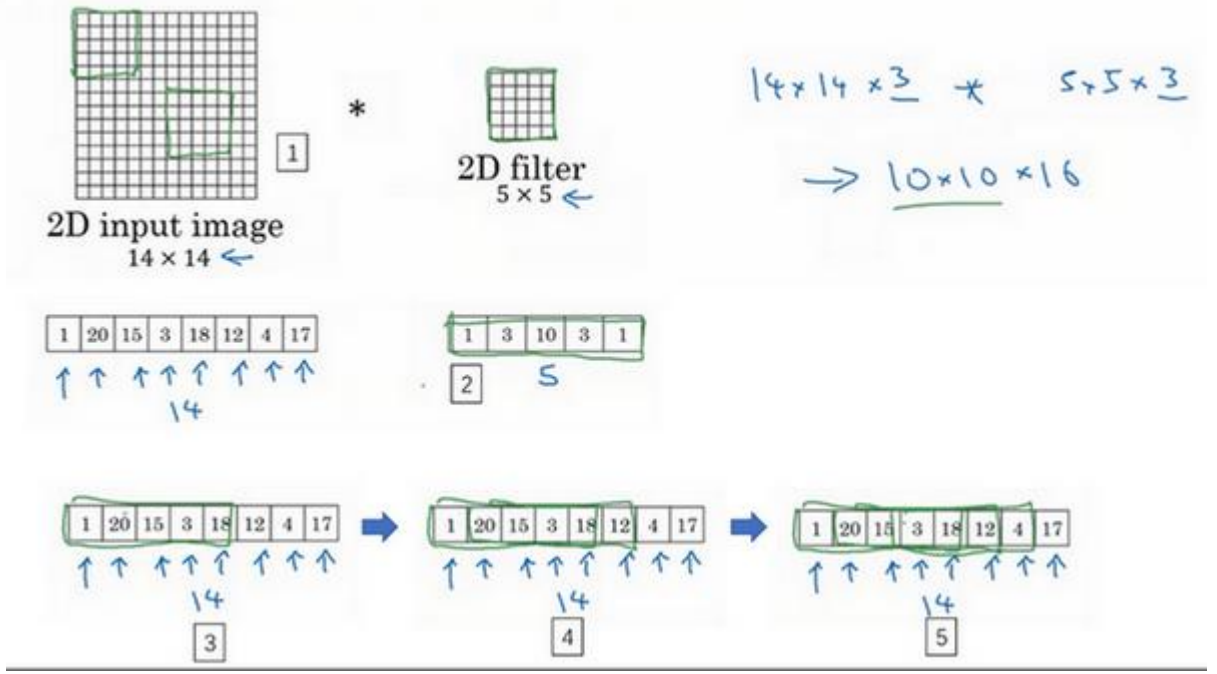
=

generated image

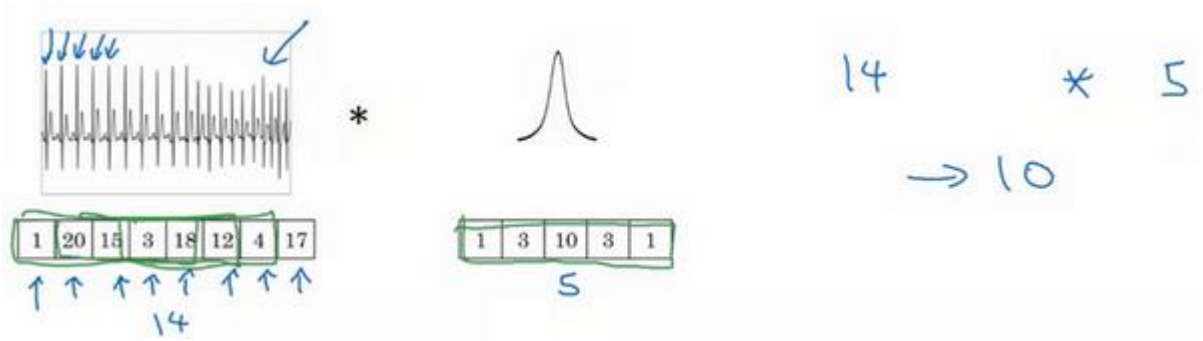
louvre painting
with impressionist style

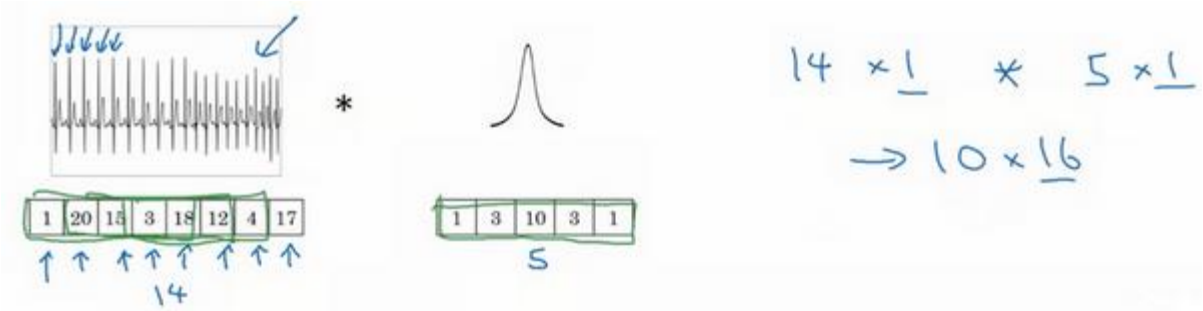
一维到三维推广


1 20 15 3 18 12 4 17
14
1 3 10 3 1
5

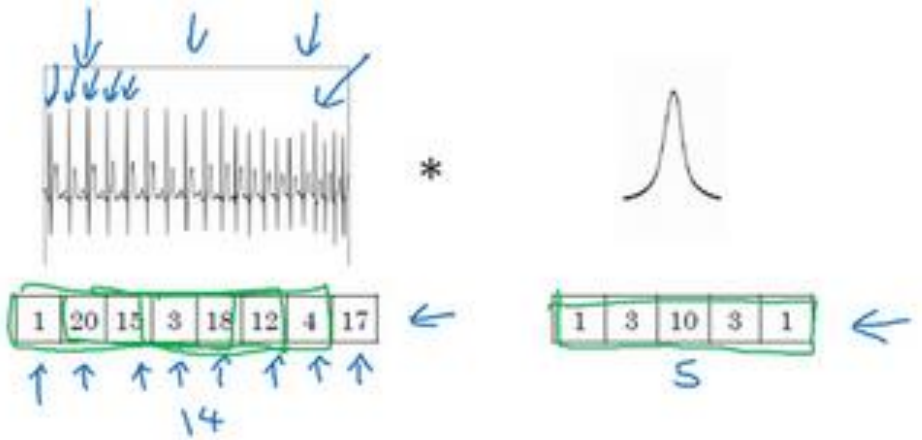

2D input image
14 × 14
1
2D filter
5 × 5
14×14×3 * 5×5×3
→ 10×10×16
1 20 15 3 18 12 4 17
14
1 3 10 3 1
2
5
3
4
5

一维到三维推广


1 20 15 3 18 12 4 17
14
1 3 10 3 1
5
14 * 5
→ 10


1 20 15 3 18 12 4 17
14
1 3 10 3 1
5
14 ×1 * 5 ×1
→ 10 ×16

一维到三维推广



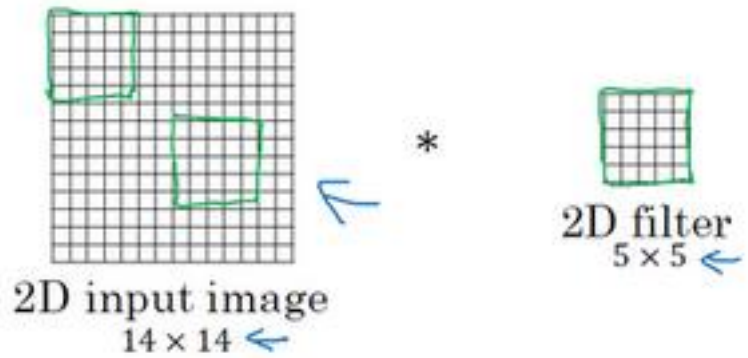
$14 \times 1 \quad * \quad 5 \times 1$

$\rightarrow 10 \times 16$

$10 \times 16 \quad * \quad 5 \times 16$

$\rightarrow 6 * 32$

Andrew Ng



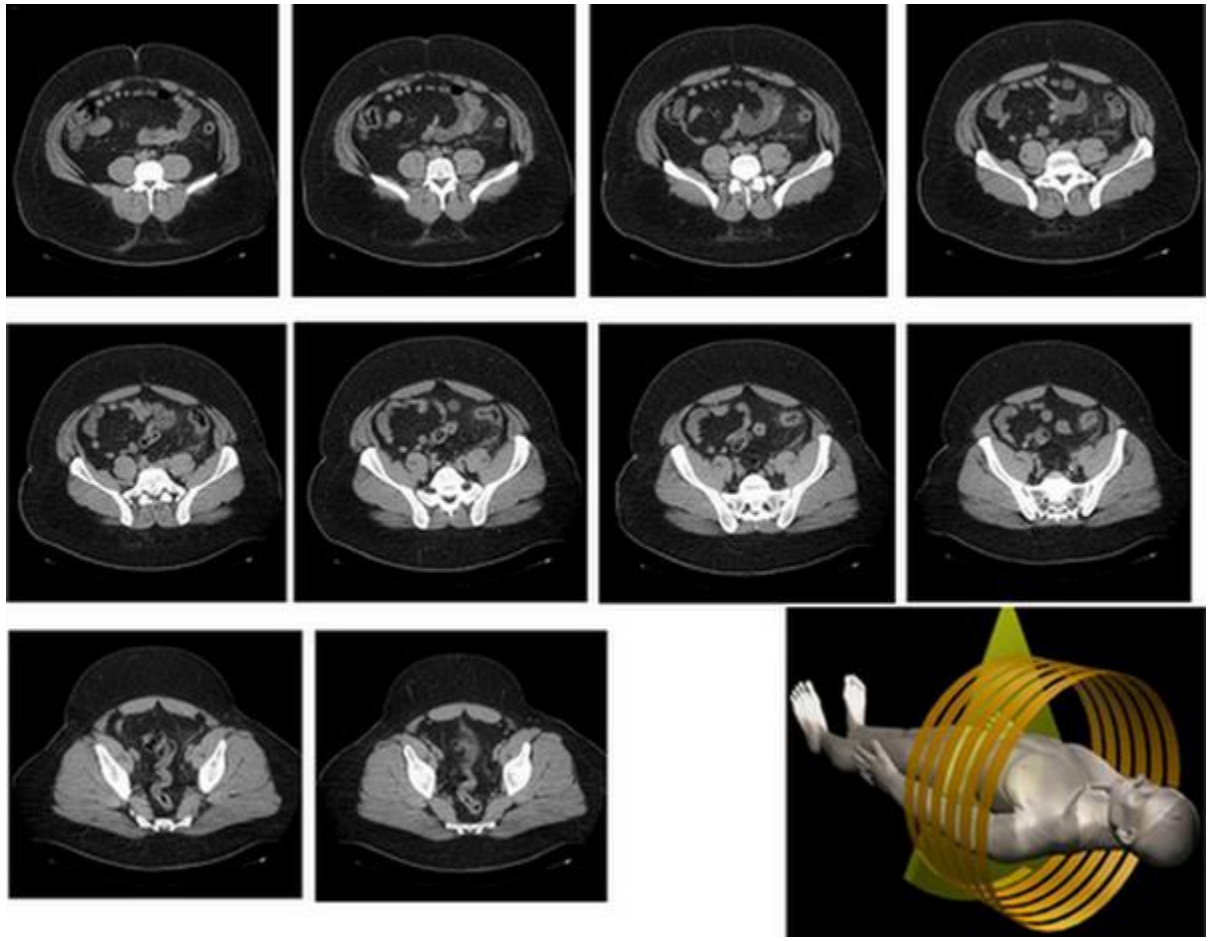
$14 \times 14 \times 3 \quad * \quad 5 \times 5 \times 3$

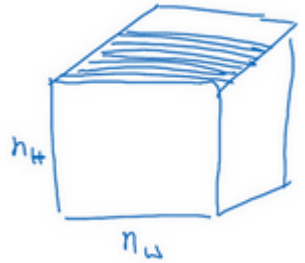
$\rightarrow 10 \times 10 \times 16$

$10 \times 10 \times 16 \quad * \quad 5 \times 5 \times 16$

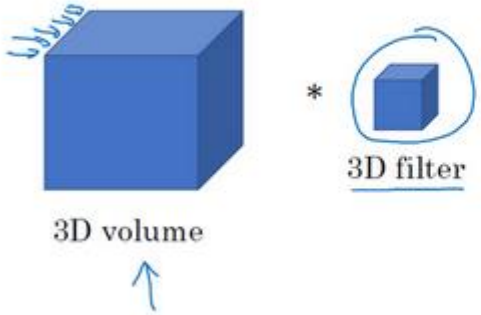
$\rightarrow 6 \times 6 \times 32$

一维到三维推广





3D convolution



n_c

$14 \times 14 \times 14 \times 1$

$* \; 5 \times 5 \times 5 \times 1$ 16 filters

$\rightarrow 10 \times 10 \times 10 \times 16$

$* \; 5 \times 5 \times 5 \times 16$ 32 filters

$\rightarrow 6 \times 6 \times 6 \times 32$

参考文献

1. IAN GOODFELLOW等，《深度学习》，人民邮电出版社，2017
2. Andrew Ng，http://www.deeplearning.ai
3. 吴茂贵等，《PyTorch深度学习：基于PyTorch》，机械工业出版社，2019


谢 谢!