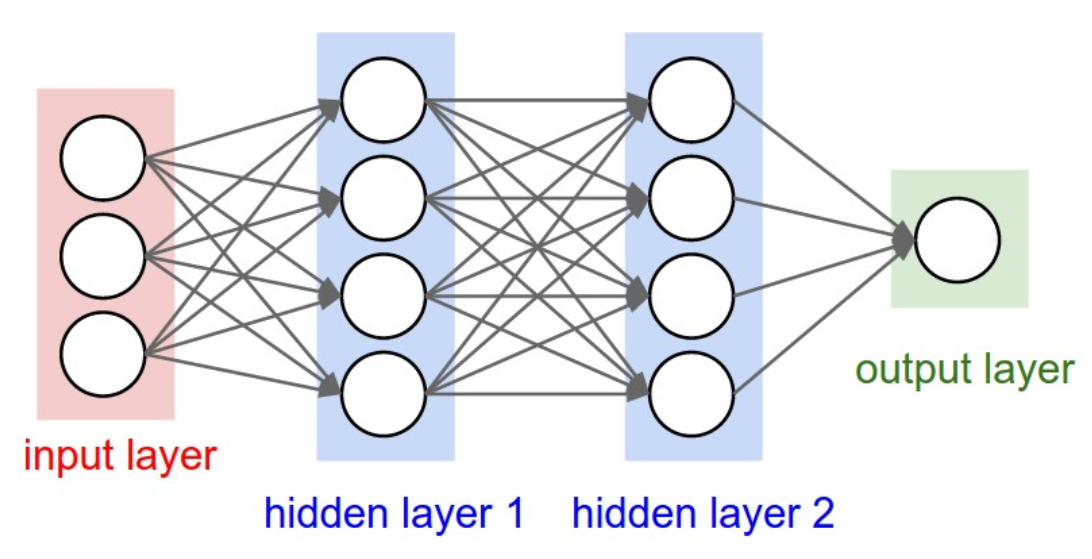
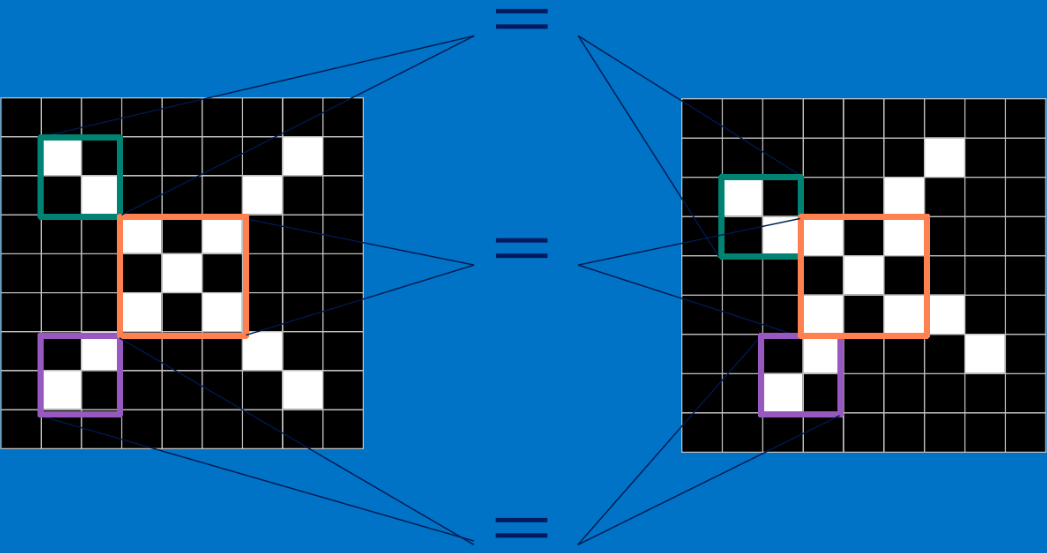
CNN



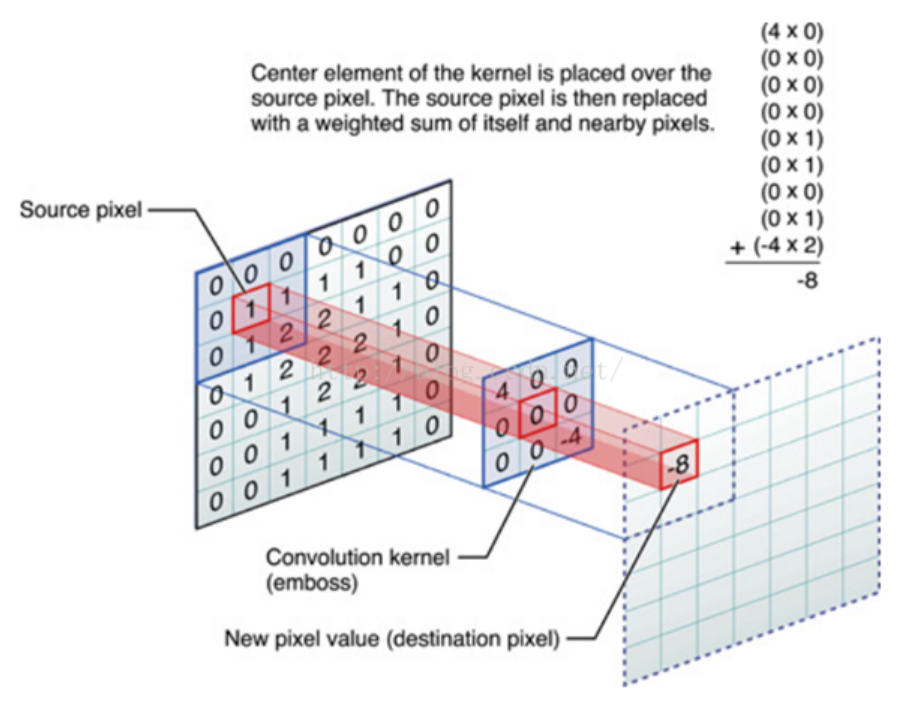
1. cnn如何进行识别

CNN是把未知图案和标准X图案一个局部一个局部的对比



### **2 什么是卷积**

对图像（不同的数据窗口数据）和滤波矩阵（一组固定的权重：因为每个神经元的多个权重固定，所以又可以看做一个恒定的滤波器filter）做****内积****（逐个元素相乘再求和）的操作就是所谓的『卷积』操作

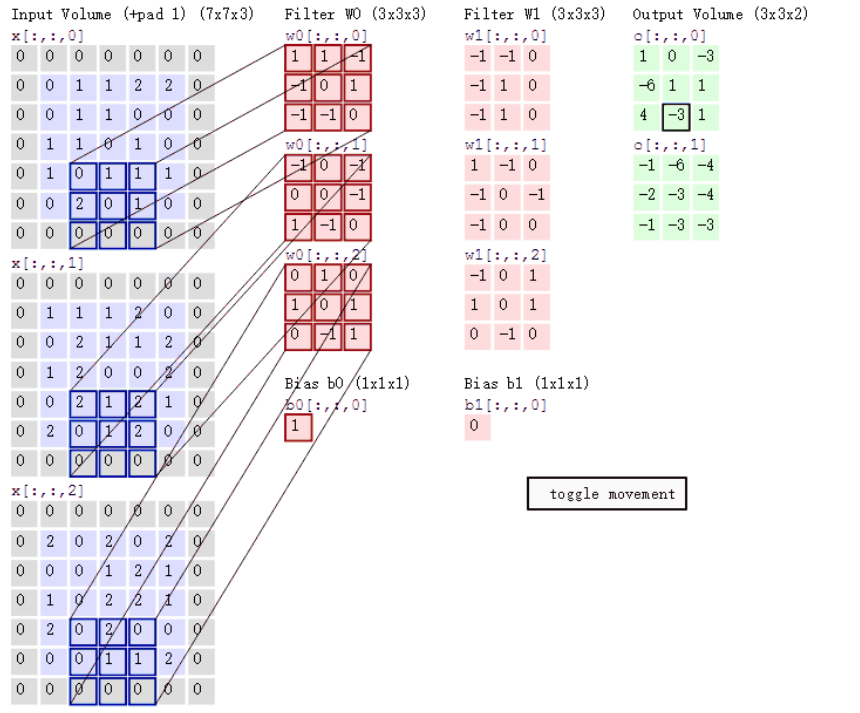


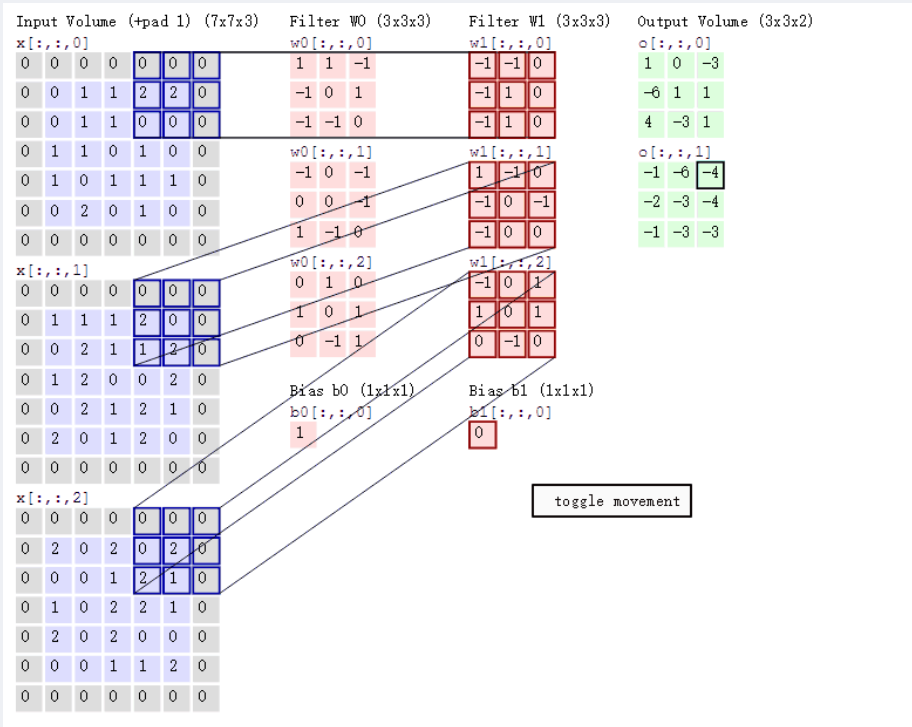
在CNN中，滤波器filter（带着一组固定权重的神经元）对局部输入数据进行卷积计算。每计算完一个数据窗口内的局部数据后，数据窗口不断平移滑动，直到计算完所有数据。这个过程中，有这么几个参数：

　　a. 深度depth：神经元个数，决定输出的depth厚度。同时代表滤波器个数。

　　b. 步长stride：决定滑动多少步可以到边缘。

　　c. 填充值zero-padding：在外围边缘补充若干圈0，方便从初始位置以步长为单位可以刚好滑倒末尾位置，通俗地讲就是为了总长能被步长整除。





数据窗口滑动，导致输入在变化，但中间滤波器Filter w0的权重（即每个神经元连接数据窗口的权重）是固定不变的，这个权重不变即所谓的CNN中的****参数（权重）共享****机制

### **3 池化pool层**

池化，简言之，即取区域平均或最大

一是减少处理特征图的元素个数，二是窗口覆盖原始输入的比例越来越大

特征中往往编码了某种模式，或概念在特征图的不同位置是否存在，平均值往往会抹消这一特点。

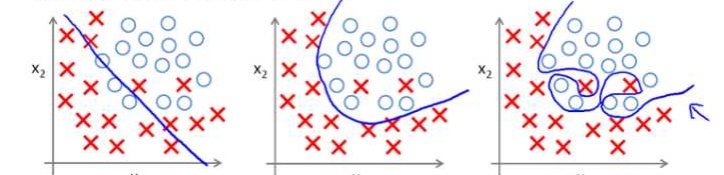
4 **数据增强**

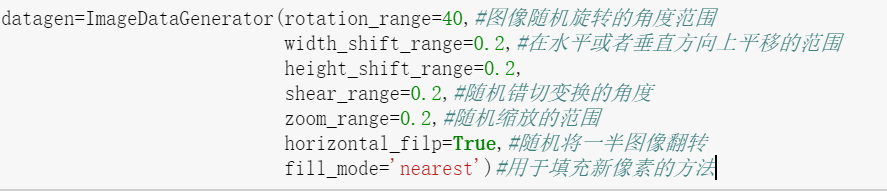
最大池化层：具有一定的内置平移不变性，当平移时，最大值依旧在窗口，则结果不变。

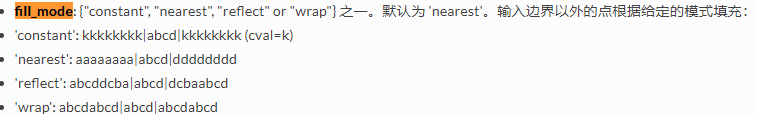
数据增强：**避免过拟合**

数据增强是什么？

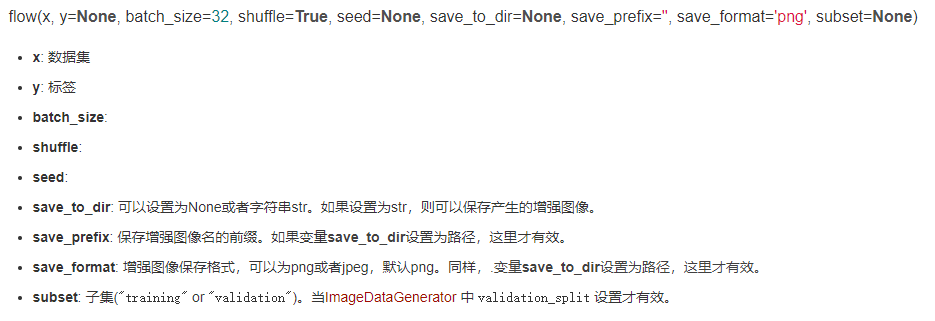
数据增强，是指对(有限)训练数据通过某种变换操作，从而生成新数据的过程。











### **fit\_generator**



运行机器学习算法时，很多人一开始都会[有意无意](https://www.baidu.com/s?wd=%E6%9C%89%E6%84%8F%E6%97%A0%E6%84%8F&tn=24004469_oem_dg&rsv_dl=gh_pc_csdn" \t "https://blog.csdn.net/sinat_26917383/article/details/_blank)将数据集默认直接装进[显卡显存](https://www.baidu.com/s?wd=%E6%98%BE%E5%8D%A1%E6%98%BE%E5%AD%98&tn=24004469_oem_dg&rsv_dl=gh_pc_csdn" \t "https://blog.csdn.net/sinat_26917383/article/details/_blank)中，如果处理大型数据集（例如图片尺寸很大）或是网络很深且隐藏层很宽，也可能造成显存不足。

这个情况随着工作的深入会经常碰到，解决方法其实很多人知道，就是分块装入。以keras为例，默认情况下用fit方法载数据，就是全部载入。换用fit\_generator方法就会以自己手写的方法用yield逐块装入

fit\_generator(self, generator, steps\_per\_epoch, epochs=1, verbose=1, callbacks=**None**, validation\_data=**None**, validation\_steps=**None**, class\_weight=**None**, max\_q\_size=10, workers=1, pickle\_safe=**False**, initial\_epoch=0)

利用Python的生成器，逐个生成数据的batch并进行训练。生成器与模型将并行执行以提高效率。例如，该函数允许我们在CPU上进行实时的数据提升，同时在GPU上进行模型训练

函数的参数是：

generator：生成器函数，生成器的输出应该为：

一个形如（inputs，targets）的tuple

一个形如（inputs, targets,sample\_weight）的tuple。所有的返回值都应该包含相同数目的样本。生成器将无限在数据集上循环。每个epoch以经过模型的样本数达到samples\_per\_epoch时，记一个epoch结束

steps\_per\_epoch：整数，当生成器返回steps\_per\_epoch次数据时计一个epoch结束，执行下一个epoch

epochs：整数，数据迭代的轮数

verbose：日志显示，0为不在标准输出流输出日志信息，1为输出进度条记录，2为每个epoch输出一行记录

validation\_data：具有以下三种形式之一

生成验证集的生成器

一个形如（inputs,targets）的tuple

一个形如（inputs,targets，sample\_weights）的tuple

validation\_steps: 当validation\_data为生成器时，本参数指定验证集的生成器返回次数

class\_weight：规定类别权重的字典，将类别映射为权重，常用于处理样本不均衡问题。

sample\_weight：权值的numpy array，用于在训练时调整损失函数（仅用于训练）。可以传递一个1D的与样本等长的向量用于对样本进行1对1的加权，或者在面对时序数据时，传递一个的形式为（samples，sequence\_length）的矩阵来为每个时间步上的样本赋不同的权。这种情况下请确定在编译模型时添加了sample\_weight\_mode='temporal'。

workers：最大进程数

max\_q\_size：生成器队列的最大容量

pickle\_safe: 若为真，则使用基于进程的线程。由于该实现依赖多进程，不能传递non picklable（无法被pickle序列化）的参数到生成器中，因为无法轻易将它们传入子进程中。

initial\_epoch: 从该参数指定的epoch开始训练，在继续之前的训练时有用。

# **5 迁移学习**

迁移学习是指对提前训练过的神经网络进行调整，以用于新的不同数据集。

取决于以下两个条件：

* 新数据集的大小，以及
* 新数据集与原始数据集的相似程度

使用迁移学习的方法将各不相同。有以下四大主要情形：

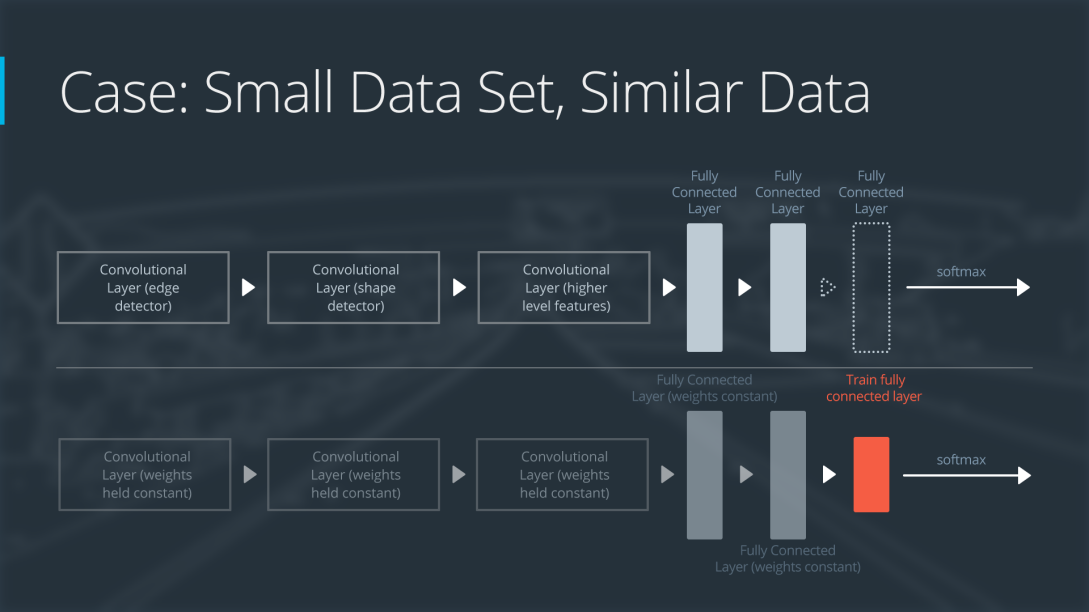
1. 新数据集很小，新数据与原始数据相似
2. 新数据集很小，新数据不同于原始训练数据
3. 新数据集很大，新数据与原始训练数据相似
4. 新数据集很大，新数据不同于原始训练数据

如果新数据集很小，并且与原始训练数据相似：

* 删除神经网络的最后层级
* 添加一个新的完全连接层，与新数据集中的类别数量相匹配
* 随机化设置新的完全连接层的权重；冻结预先训练过的网络中的所有权重
* 训练该网络以更新新连接层的权重

为了避免小数据集出现过拟合现象，原始网络的权重将保持不变，而不是重新训练这些权重。

因为数据集比较相似，每个数据集的图片将具有相似的更高级别特征。因此，大部分或所有预先训练过的神经网络层级已经包含关于新数据集的相关信息，应该保持不变

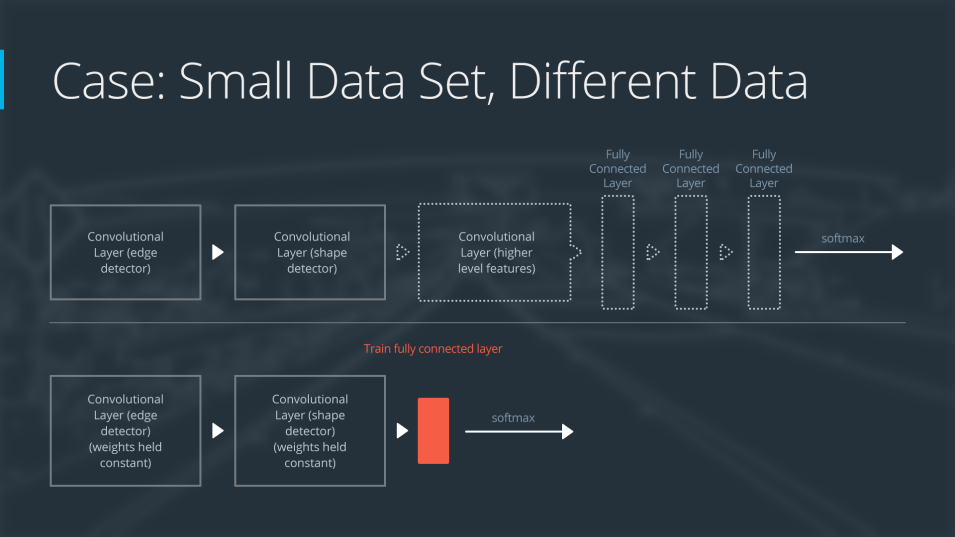


如果新数据集很小，并且与原始训练数据不同：

* 将靠近网络开头的大部分预先训练过的层级删掉
* 向剩下的预先训练过的层级添加新的完全连接层，并与新数据集的类别数量相匹配
* 随机化设置新的完全连接层的权重；冻结预先训练过的网络中的所有权重
* 训练该网络以更新新连接层的权重

因为数据集很小，因此依然需要注意过拟合问题。要解决过拟合问题，原始神经网络的权重应该保持不变，就像第一种情况那样。

但是原始训练集和新的数据集并不具有相同的更高级特征。在这种情况下，新的网络仅使用包含更低级特征的层级。

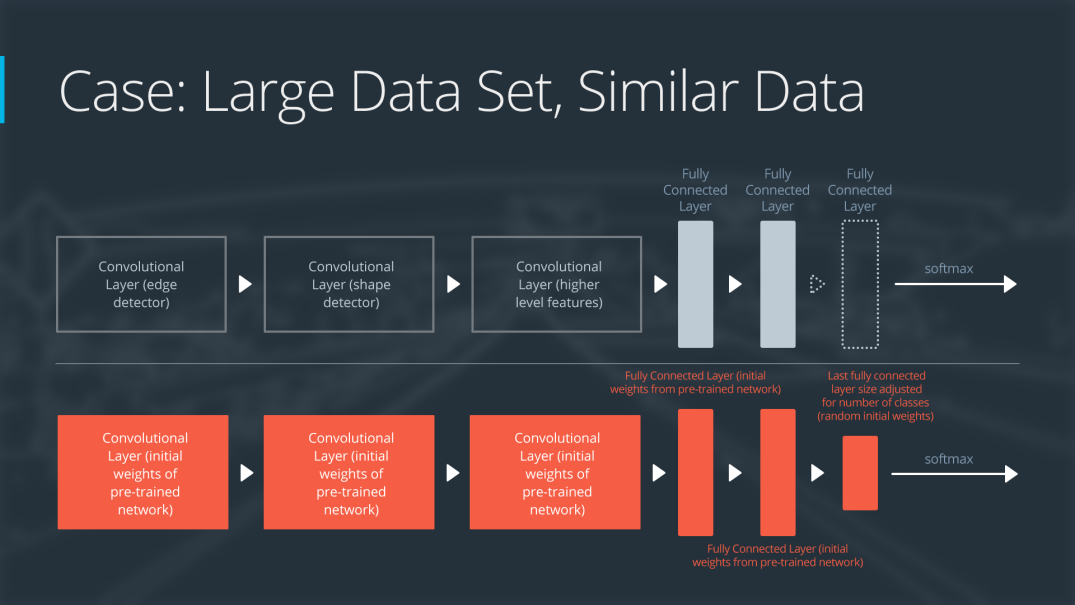


如果新数据集比较大型，并且与原始训练数据相似：

* 删掉最后的完全连接层，并替换成与新数据集中的类别数量相匹配的层级
* 随机地初始化新的完全连接层的权重
* 使用预先训练过的权重初始化剩下的权重
* 重新训练整个神经网络

训练大型数据集时，过拟合问题不严重；因此，你可以重新训练所有权重。

因为原始训练集和新的数据集具有相同的更高级特征，因此使用整个神经网络。

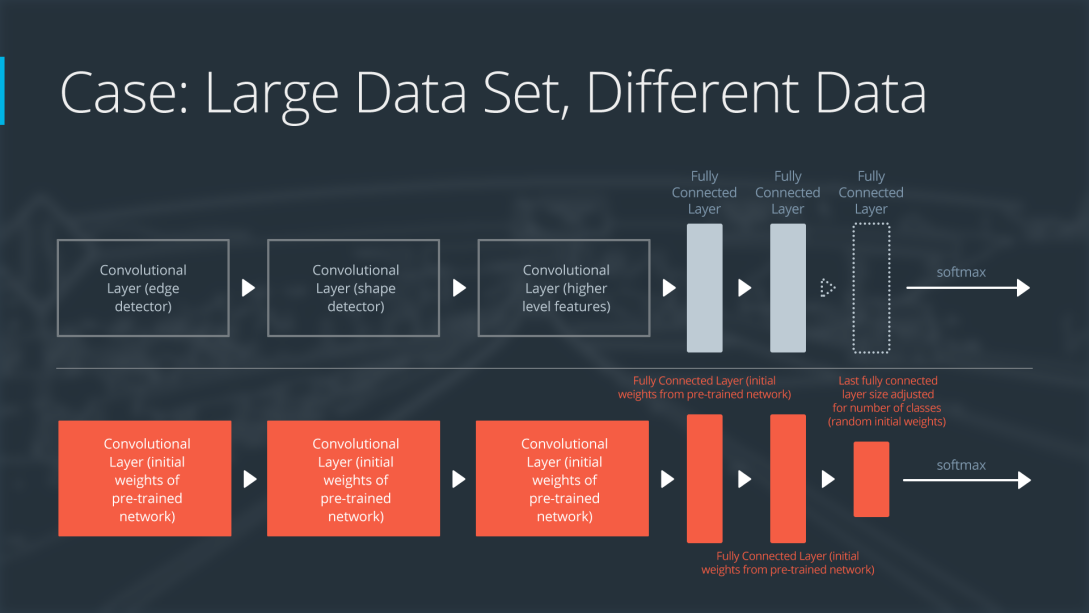


如果新数据集很大型，并且与原始训练数据不同：

* 删掉最后的完全连接层，并替换成与新数据集中的类别数量相匹配的层级
* 使用随机初始化的权重重新训练网络
* 或者，你可以采用和“大型相似数据”情形的同一策略

虽然数据集与训练数据不同，但是利用预先训练过的网络中的权重进行初始化可能使训练速度更快。因此这种情形与大型相似数据集这一情形完全相同。

如果使用预先训练过的网络作为起点不能生成成功的模型，另一种选择是随机地初始化卷积神经网络权重，并从头训练网络。

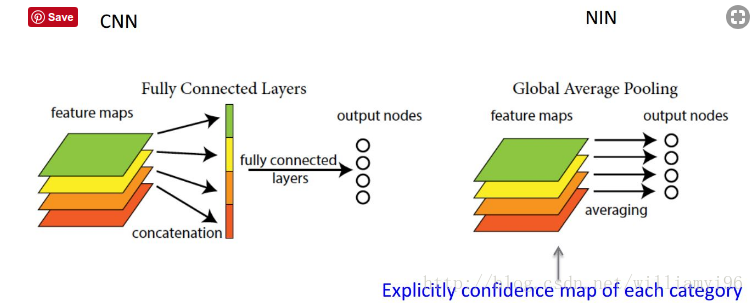


# GAP全局池化替代全连接层的分析

全连接层有一个非常致命的弱点就是参数量过大，特别是与最后一个卷积层相连的全连接层。一方面增加了Training以及testing的计算量，降低了速度；另外一方面参数量过大容易过拟合。虽然使用了类似dropout等手段去处理，但是毕竟dropout是hyper-parameter。

全连接层将卷积层展开成向量之要针对每个feature map进行分类，GAP的思路就是将上述两个过程合二为一，一起做了。

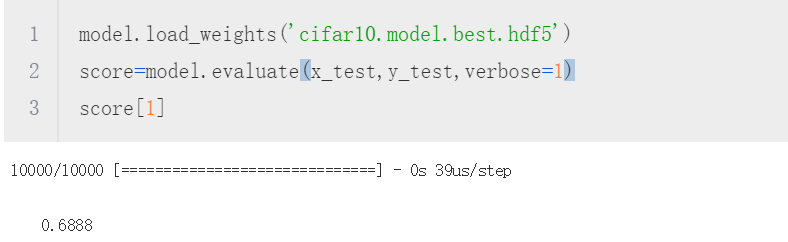
用 feature map 直接表示属于某个类的 confidence map，比如有10个类，就在最后输出10个 feature map，每个feature map中的值加起来求平均值，这十个数字就是对应的概率或者叫置信度。然后把得到的这些平均值直接作为属于某个类别的 confidence value，再输入softmax中分类



**6.结果分析**

使用极算云跑出来的结果

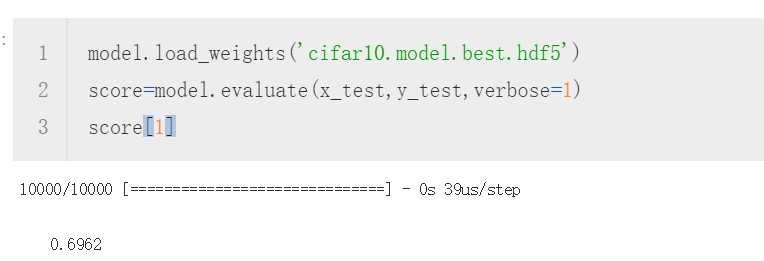
CNN——cifar10



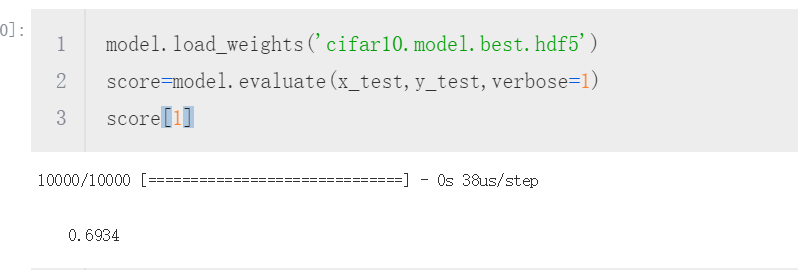
修改参数观察结果

1. 最后的全连接层尽量进行完整训练，因为训练的次数不是很多，所以最后全连接层过拟合的可能性较低。
2. 修改卷积层的filters数量，比如把第一层改到了64，结果最后准确率下降到0.09左右，前面是提取一些低级特征，后面是一些高级的特征，简单理解下就是低级的特征分辨率要低一些，高级的特征要复杂多变所以要精细些。
3. 使用GPA后准确度提高到49% 看来是有过拟合的现象出现











**caffe因为使用opencv处理图片的缘故，一般训练出来的模型是读取BGR图像的**

## 6 迁移学习 补：

## 什么是/为什么要迁移学习？

迁移学习(Transfer learning) 顾名思义就是就是把已学训练好的模型参数迁移到新的模型来帮助新模型训练。考虑到大部分数据或任务是存在相关性的，所以通过迁移学习我们可以将已经学到的模型参数（也可理解为模型学到的知识）通过某种方式来分享给新模型从而加快并优化模型的学习效率不用像大多数网络那样从零学习（starting from scratch，tabula rasa）。

