目　录

一、写在前面

如果让你设计一个可以识别图像的神经网络，你会怎么做？

我之前问过自己这个问题，思来想去，我的答案是：我可能不知道如何下手。

突然有一天，当我把 Resnet50 这个网络的所有算法都写完一遍之后，我突然觉得，只要我深入了解了这些算法的原理，或许这个网络我也能设计出来（后知后觉的大话而已）。

于是，我有了一个大胆的想法。

接下来，我会花 10+篇文章的内容，从头开始，聊聊一个图像识别网络是怎么工作的，每一步算法的原理，以及相关的背景知识。

你可能会想，看懂这些需要什么知识呢？

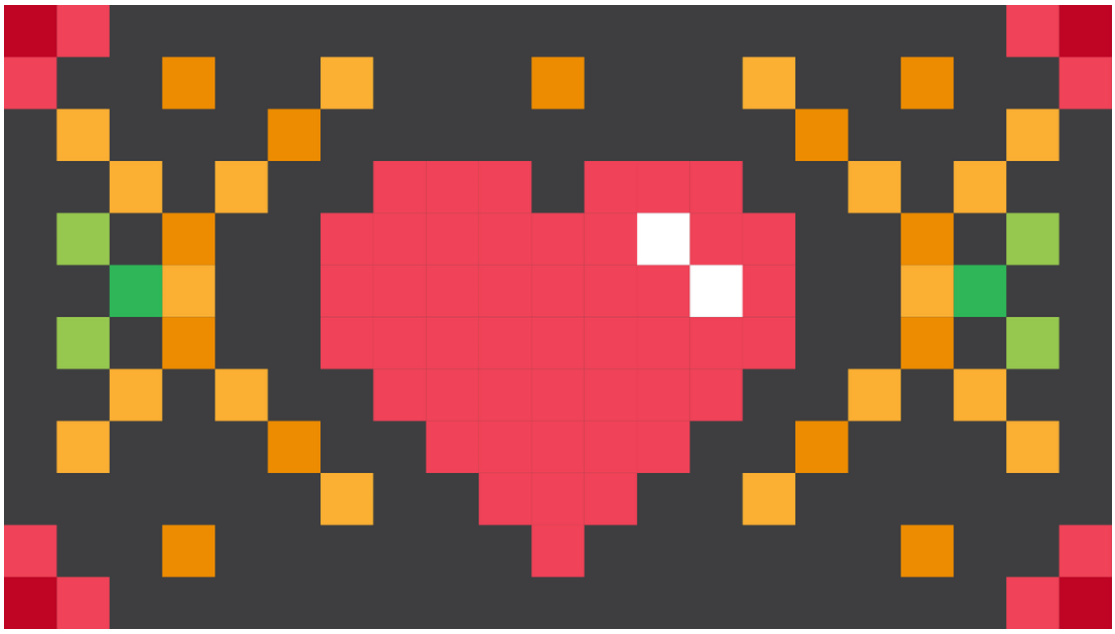
其实不需要太深奥的数学知识。

我会尽可能把每一步写的通俗易懂，这个过程中我也会搜集一些资料，也是一个不断完善自己知识体系的过程。

文中聊得网络，就是大家都比较熟悉的，被玩烂的、作为各大 AI 芯片厂商性能标杆的 Resnet50。

个人水平有限，文中如有错误，欢迎联系我指正。

二、从像素说起

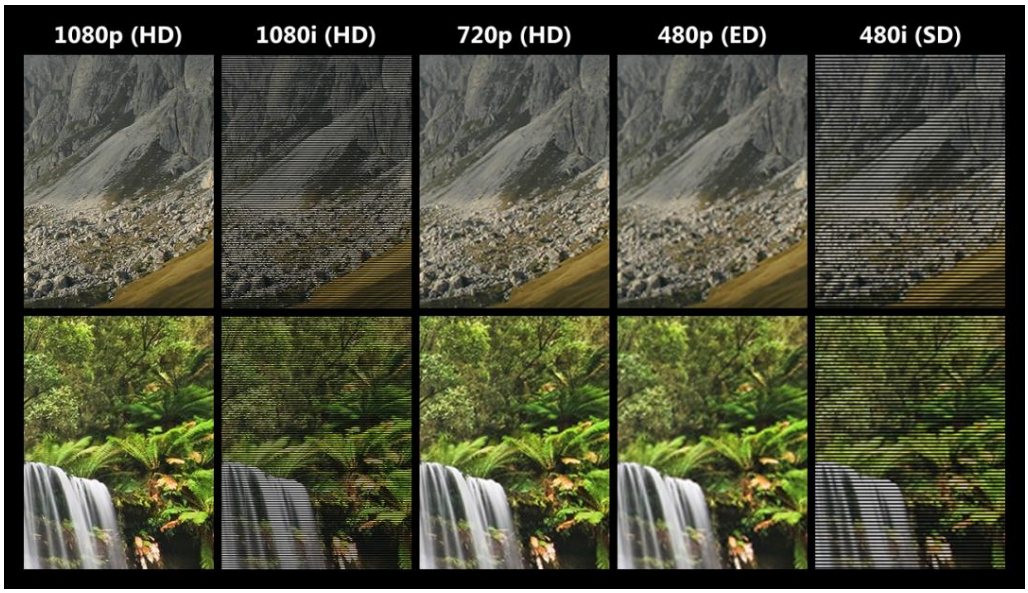


要实现图像识别，最离不开的，就是像素。

其实我们都知道，图像是由像素组成的。

实际上，神经网络计算，算的就是像素之间的关系，以及这些关系背后可能隐藏的图片信息。相机摄像头像素 2000 万，拍出来的照片肯定比像素 1000 万的要清晰我们更容易看到图片中的物体是什么。

这是为什么？因为像素越多，像素之间的关系（色彩，轮廓）越丰富，我们所能看到的信息就越多，自然而然获取到的信息就多。

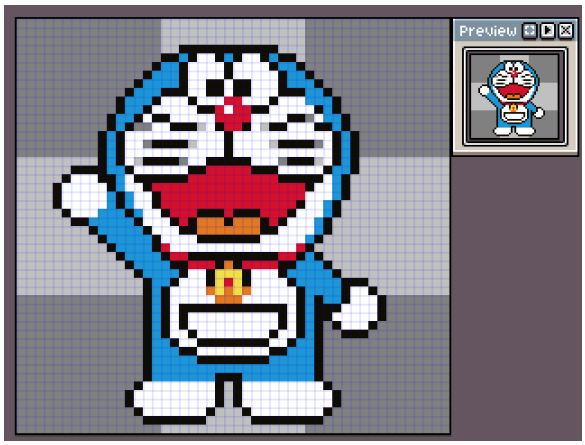


一张 1080p 的图片，我们可以更容易辨别出图像中的物体是山还是水。这是因为更多的像素会给眼睛更丰富的图片细节

但是，你有没有发现，当我们去看一张图片时，我们绝对不是盯着某一个像素或某几个像素看，而是看了整个图像的大部分区域，或者说，大部分像素！

因为只有看到了大部分的图片，才能知道图中是座山。正所谓，聚沙成山！绝不是少了一粒沙，山就不是山，多了一粒沙，就变成了山。

像素局部性



上图哆啦 A 梦，虽然不是很清晰，像素点数也很少，但一眼望去，依然可以分清是哆啦 A 梦，甚至，用手捂住一半的图像，依然可以。

这是因为人眼对于图像信息的识别，是建立在对像素局部性分析的基础上的。

所谓局部性，通俗点说，就是眼睛或大脑会将相邻的像素或大片的像素连接起来分析，从而组合成嘴巴，然后是耳朵，最后是哆啦 A 梦。

神经网络识别图片大致就是这样的原理，它模拟的，就是人们看到图片之后的信息处理过程。当我们盯着一个图片看时，我们首先会获取到图片的细节特征。比如哆啦 A 梦红色的大嘴巴。

但是如果仅仅盯着大嘴巴，又反而让人有一种“只缘身在此山中”的感觉，看不到图片的全貌。

因此还需要看一下图像的轮廓。于是，我们的眼睛看图片大致有以下两个过程：

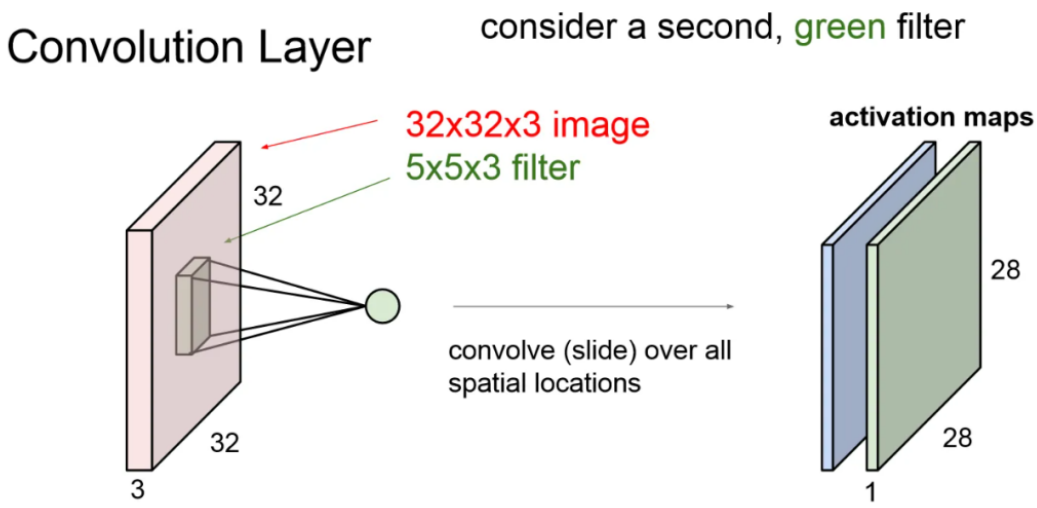
- 瞳孔放大，盯着某一处细节（如大嘴巴）看
- 瞳孔缩小，模糊的看一张图片的大致轮廓

两个过程获取的信息叠加。Bingo，看清楚了，是哆啦 A 梦！那么神经网络是否可以模拟这种瞳孔放大、缩小的方式呢？

很幸运，可以！

那就是卷积算法。

卷积



卷积的计算示意图

卷积核的设计，就可以很直观的模拟这种获取图片信息的方法。

人们通过调整卷积核的大小，来达到瞳孔张开、缩小的目的。并且大量的实验和论文表明，卷积这一针对图像局部性识别的算法，可以非常有效的模拟人眼识别物体的过程。

关于卷积算法以及卷积核的设计原理，后面会专门来讲，因为卷积这一算法，在图像处理领域，太重要了。我们现在继续沿着像素这一话题讲下去。

色彩分量 RGB

回到像素这一话题。

你有没有想过，为什么一张图片会是彩色的。

学过摄影的小明同学可能这时会回答：因为图片是由 RGB 三种颜色来表示

的，每个像素实际是不同的 R/G/B 分量的叠加，混合起来，就表示成了不同的颜色。

回答正确。

三张分别表示 R/G/B 分量的图片，合成一张彩色图片

对于上面一张 RGB 的图，我们人眼可以很直观的看到红色和蓝色，可以察觉到一张图片的色彩和轮廓。那么，如果让计算机来处理图片，他又是如何知道色彩和轮廓的呢？

其实对于计算机来说，一张图片只是一堆数据，计算机是无法知道这堆数据代表的是什么。

这就需要人为的给这堆数据一种表示方法，让计算机知道，哦，这 1/3 的数据是红色分量，这 1/3 的是蓝色分量，这些数据（像素）组合起来，可能代表的是个“帽子”。

怎么做呢？

通过设计数据在计算机中的存储方式来解决。

数据在计算机的存储中，最常见的存储方式是连续存储的。比如 C 语言，定

义一个数组，那么数组在内存中的位置是连续的。

int data[10] = {0,1,2,3,4,5,6,7,8,9};

内存怎么理解，它就是一排连着的门牌号的公寓宿舍。门牌号为 101 里面住着的，是 data 的第一个数据 0。门牌号 102 里面住着的，是 data 的第二个数据 1，...，以此类推。

哈利波特的女贞路 4 号，内存也是类似于门牌号规则，数据就像人一样，存储在以地址为标识的一个个的内存地址上（房子）

只不过，在计算机存储器中，没有门牌号，有的都是地址。

这个时候，计算机根本就不关心数据是啥，计算机用到的时候，就把数据从内存对应的地址中取出来用。

计算机是如何取像素数据的

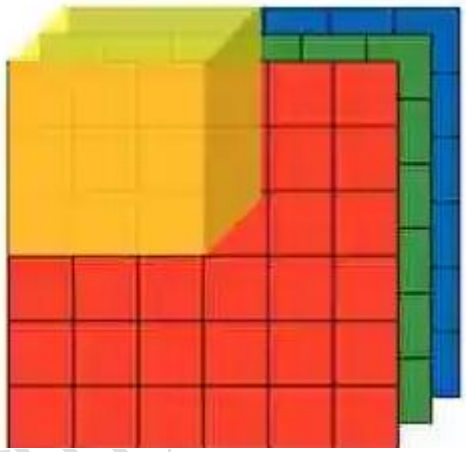
人们为数据存储设计一种格式，告诉计算机，这堆数据大概是什么样的。

只有这样，通过这种约定的方式，计算机才能正确的取到 R 分量或者 B 分

量。

对于一张图片来说，最常见的两个参数是长和宽，一般用 H（height）和 W(width) 来表示，那么 RGB 三个分量，看作是 3 个通道（channel），一般用 C 来表示。

如此一来，一张长宽分别是 224 像素*224 像素的 RGB 图像，就可以用 HWC = [224, 224, 3]来表示。两张类似的图片就用 NHWC = [2, 224, 224, 3]表示，其中 N 代表图片张数。



一张图片的抽象数据表示

友好的数据表示方法，可以减少大量的计算复杂度。虽然这样表示不太利于人们的直观理解，但是计算机处理这种数据是十分方便的。

在目前主流的深度学习框架中，对于图片的数据格式，基本都支持了 NHWC 或 NCHW 这种数据摆放格式。说到底，都是为了更高效地进行图片数据的处理和运算。

总结一下

这一章，从像素说起，说到了像素具有局部连接性的，人眼识别图像也是通过获取像素的局部连接性信息来完成的。

幸运的是，卷积这一算法，可以很好的模拟这一过程。

最后，为了使计算机更高效的处理图片数据，引出 NHWC 的图片数据表示方法，所以，之后我们说图片，不仅仅局限于图片的长和宽，还多了一个维度信息，那就是 channel。

加餐

熟悉 OpenCV 或者计算机视觉的同学，可能对于上面的 RGB 分量中的女神很熟悉。没错，在很多的教程中，这位女神不止一次的出场。

Lena

这位女士名叫 Lena。

电气电子工程师学会图像处理汇刊 （IEEE Transactions on Image Processing）主编曾在 1996 年 1 月出版的一期中解释道，Lena 的流行，因为她是一张好的测试图片，其中包含了很多细节，平滑区域，阴影和纹理。

当然，另外一个原因就是漂亮美女的图片自然受到男性居多的研究领域的欢迎。

三、图像的色彩空间

上一章从像素开始，聊到了 RGB 这一常见的色彩空间模型。

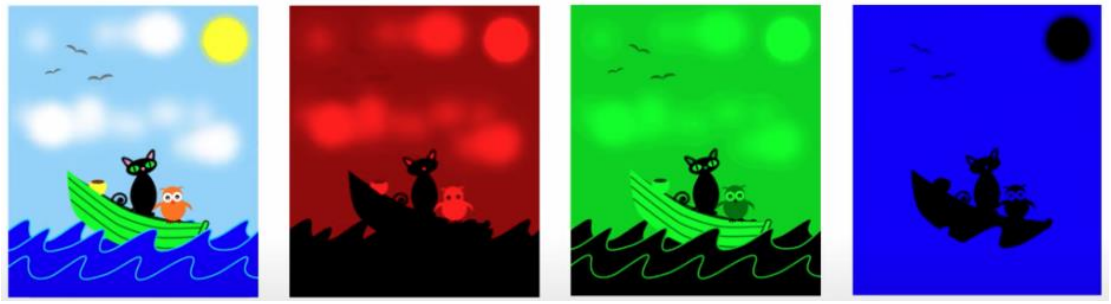
之所以还想继续聊聊 RGB 以及另一种色彩空间模型-YUV ，不是说想要以后去学摄影，学学如何需要调节色度、曝光和饱和度啥的。

而是在图像识别的深度学习任务中，RGB 以及 YUV 这些概念，总是会时不时的出现一下，让枯燥无味且高度抽象的深度学习算法，突然之间，变得具体一些，光鲜一些。

RGB

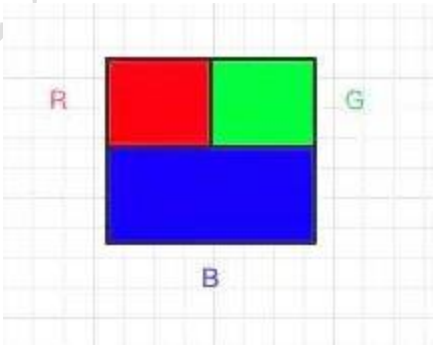
Red，Green，Blue（RGB）是我们最常见的图像表示方法。

这个非常好理解，三原色的融合，几乎可以构造出所有需要的颜色。三张 RGB 分量图片的融合，就可以构成一幅色彩斑斓的图片。



原图与 R G B 三个分量图片

平时我们说，分辨率为 1920*1080 的图片，它代表的是在长宽两个方向上，有 1920 * 1080 个像素，但是，在色彩这个方向上，还有 3 个通道（channel），也就是 RGB，往往被我们忽略。我们看到的一个像素点的颜色，在计算系统中，并不是简单的由一个数值来表示的，而是由 RGB 三个分量的三个数值来表示的。



一个像素点的 RGB 表示

因此，想要计算一张 1920*1080 的图片的大小，或者说计算这张图片在计算机内存中所占用的大小时，不能仅仅用图片的长度乘以宽度这么算，还需要考虑通道数。

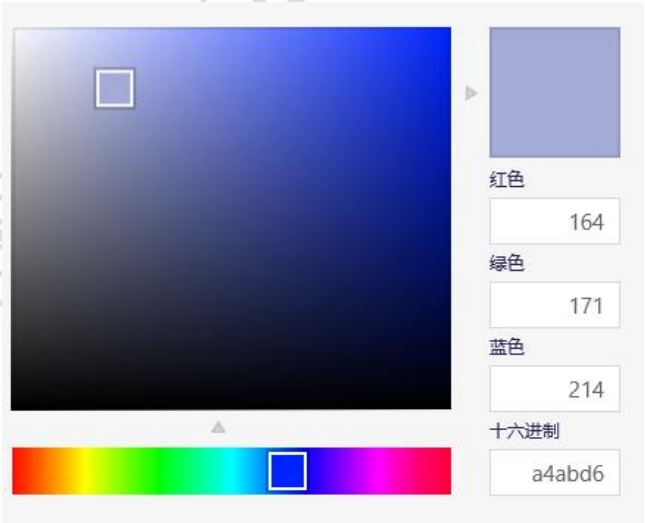
数值表示和图片大小计算

我们可能用过画图这一软件来调过颜色。

通过简单的设置红色，绿色，蓝色的数值，就可以在调色板中得到一个颜色。有没有注意到，无论红色，还是绿色，还是蓝色，其表示的数值都没有超过 255。

为什么？

因为像素值，在计算机语言中，是用一个 int8 的数据来表示的。而 int8，指的是 8bit 无符号整数，其能表示的范围就是 0 - 255。



画图面板：自从几年前微软宣布停止更新画图软件之后，画图就越来越少的出现在人们的视野中，你或许可以打开电脑看看，左下角的菜单里，是否还有画图软件，就像当年 window xp 系统被停用一样，慢慢的就会消失在人们的记忆里

说到这，我们就可以算一算，一张 1920 * 1080 的 RGB 图像，在计算机的表

示中，到底占多少的内存？

很简单，长宽方向上每个像素由 3 个通道组成，每个通道由一个 8 bit 的数值表示，一个 8 bit 数值代表一个字节 (Byte)。因此，一张 1920 * 1080 的 RGB 图像，占计算机存储大小为

1920 * 1080 * 3 * 1 Bytes = 6075 KB = 5.9 MB

5.9 MB 的内存占用！

大么？和目前动辄几个 G 的手机内存相比，不算大。

小么？和边缘侧图像识别终端内存，比如摄像头里的嵌入式芯片内存相比，又不算小。

更何况在公路违法拍照的摄像头场景下，在车流量很大的时候，需要实时处理的图片，可远远不止一张！

那怎么办？

有没有办法可以在进行图像处理时，减少图片的数据量，从而减少图片大小和内存占用呢？有，YUV 就是其中一种办法。

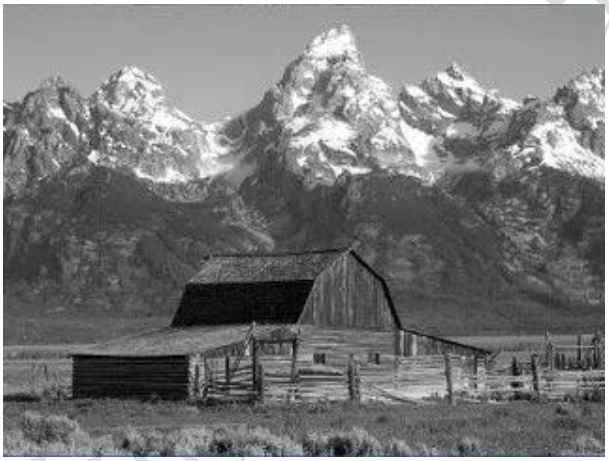
YUV

YUV 是将亮度信息和颜色信息进行编码的一种颜色空间表示方法。和 RGB 类似，YUV 也使用 3 个字母维度来表示颜色。

为了简单点，我们暂时将这 3 个值称为 Y，U 和 V。(事实上，YUV 的称呼很多，比如 Y'CbCr，也很细节，这里不多描述，我们只要知道它是另外一种表示颜色的方法就可以。)

Y 代表亮度，U 代表色彩度，V 代表饱和度。

原图



仅有 Y 分量也就是亮度，黑白图片

仅有 U 分量，只有色度

仅有 V 分量，只有饱和度

上面的几张图片，除了原图之外，我们可能更加倾向于使用只有 Y 分量的图片，也就是那张黑白图像。因为即使没有色彩，但是它的轮廓以及明亮程度，也足以让我们分辨出图片中的物体。

其视觉效果远远好于其他两个分量的图片。相反，只有色度和饱和度的图片，反而变得模糊不堪。

这就是问题所在！

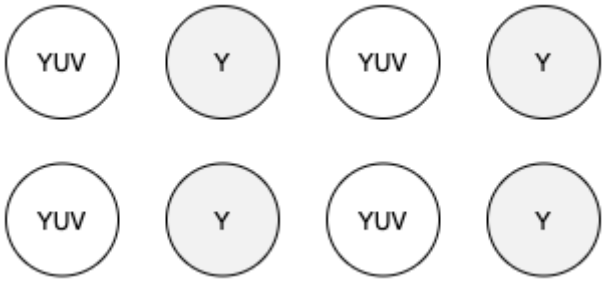
人眼对于亮度具有更高的敏感度，而对色度和饱和度反而显得迟钝一些。说到这，有没发现什么？

既然人眼对于色度和饱和度的反应不敏感，那就没有必要把所有的色度和饱和度信息都放在图片里了啊。

举个例子，色度和饱和度我隔一个像素放一个，剩下的像素没有饱和度，不就可以了么。

没错，是可以，而且效果很好。

这就是 YUV 的不同编码。



YUV422 编码

实际上，YUV 的编码方式有很多种，比如 YUV444, YUV422 等。

大致意思就是，保留全部的 Y 分量（人眼最敏感的亮度分量），但是只保留部分的 U/V 分量（人眼不敏感），以此来减少图片的占用，但又不失重要信息。

还记得上面的 1920*1080 的 RGB 图片的内存占用么，为 5.9MB。如果用 YUV444 的编码，结果也同样是 5.9MB， 因为 YUV444 也是全采样，所有的亮度、色度、饱和度信息都保留了。而如果采用 YUV422 的编码，相当于 U 分量减少一半，V 分量减少一半。那么最终的图片占用大小就变成了

1920 * 1080 * (1 + 0.5 + 0.5) Bytes = 4050KB = 3.9MB

只有 3.9MB，减少了 1/3 的内存占用！

是不是好很多？更多关于 YUV 的编码知识，有兴趣的同学可以 Google。如果不做相关课题，可以不用深究。我们只需要知道，YUV 这一色彩编码方法，在保留亮度这一人眼最敏感信息的基础上，通过降低其他人眼不太需要的信息，可以来达到降低图片大小的目的。就足够了！

YUV 编码的用途

原始图片，channel 数代表的是 RGB 通道，可以理解为原始图片具有的三个

特征。可在深度学习网络中，随着网络深度的增加，图片的 channel 数会不断的增大。

就拿 Resnet50 这个网络来说，最后面的一层图片，channel 数已经增大到了 2048。这时 channel 代表的信息，早已不再是 RGB 这种基础的特征了。

而是通过了大量的神经网络训练，代表了图片的多种分类特征，比如是猫的特征还是狗的特征。这个后面会详细说。

YUV 这种编码方法，可以用在图片的上下采样中。通过降低或增加通道数，实现图片的上下采样，以此来实现图片的增大或减少，但又不损失太多我们希望保留的信息。

总结一下

这章聊了聊 RGB、YUV 两种颜色空间，以及 YUV 可能的用途和它的优势。

为什么聊这么多关于图片的东西，因为在深度学习处理图像的任务中，图片是原材料。正所谓知己知彼，百战不殆。了解了图片这一深度学习的原材料之后，我们就可以更加高效的来完成图片数据的处理和分析。就可以开始图像识别的算法之旅了！

加餐

公元 663 年，唐代诗人王勃游历南昌，登滕王阁而做序，大笔一挥，豪气万

丈。当时的王勃，傲立在滕王阁楼顶时，看到的应该是怎样的一副壮美图景，才使得中国文坛留下了一句千古绝唱。

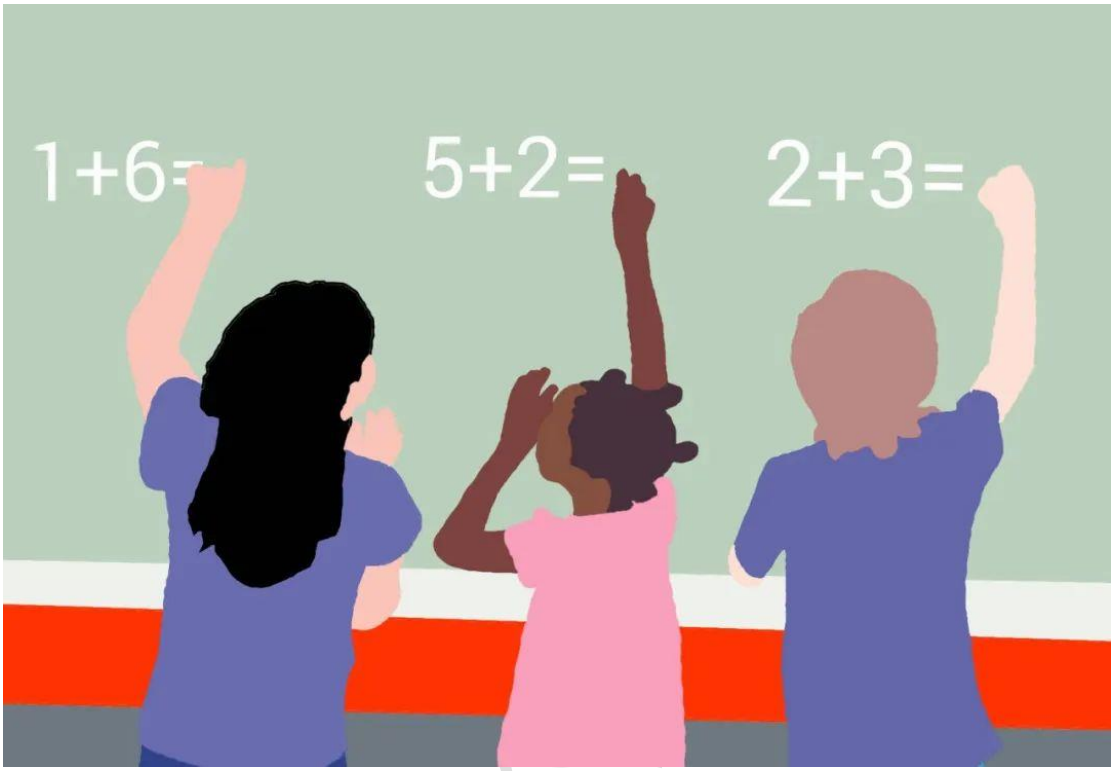
潦水尽而寒潭清， 烟光凝而暮山紫。

当时的王勃，应该怎么也不会想到，1000 年后的今天，聪明华夏后人，将"暮山紫"这一颜色进行了编码。

从此，暮山紫，不只存在于人们的想象中，而是精确地存在了计算机里。

暮山紫-RGB：163 171 214 | 图片来源：微博@一条

注：动图 pdf 不显示，可点击这里查看



四、初识卷积

就像上一章说的那样，图片是做深度学习任务的原材料，就像是做饭，不了解原材料的特性，怎么能快速高效的做出一顿美味的大餐？

下面开始，想聊聊卷积。但是不聊公式，只想聊一下，卷积这一算法是如何工作的，以及它的一些原理。

人脑是怎么记住东西的？

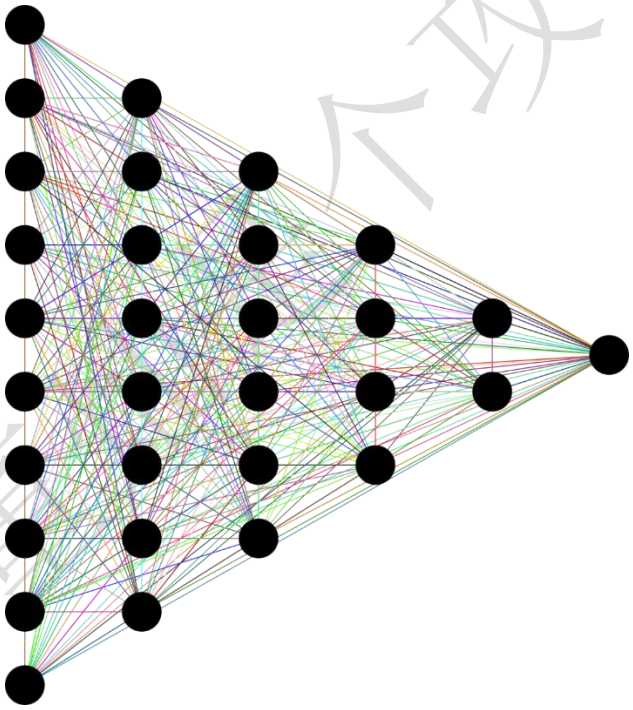
在说卷积之前，先务虚一下，说说 AI 的记忆。或许你已经听说过很多 AI 故事了，比如大名鼎鼎的阿尔法狗大战柯洁。但是，你有没有想过一个问题。

阿尔法狗确实是学会了下棋，但是它下棋的记忆到底是什么样的？存在什么地方呢？

高中生物老师教过我们，人脑中有大量的脑神经元。每个脑神经元都可以看做是一个小的记忆体，神经元之间通过树突连接起来。整个大脑的神经元，可以说是一张十分复杂的网络。

人脑处理信息，就是利用这个复杂的网络处理信息，并最终得到一个结果。通过神经元网络，我们才能知道，眼睛看到的是一只猫，还是是一只狗。

稍微简化一下大脑神经元的复杂结构成如下的网络。



每个黑点代表一个神经元脑细胞，每个神经元都有自己负责记忆的东西。

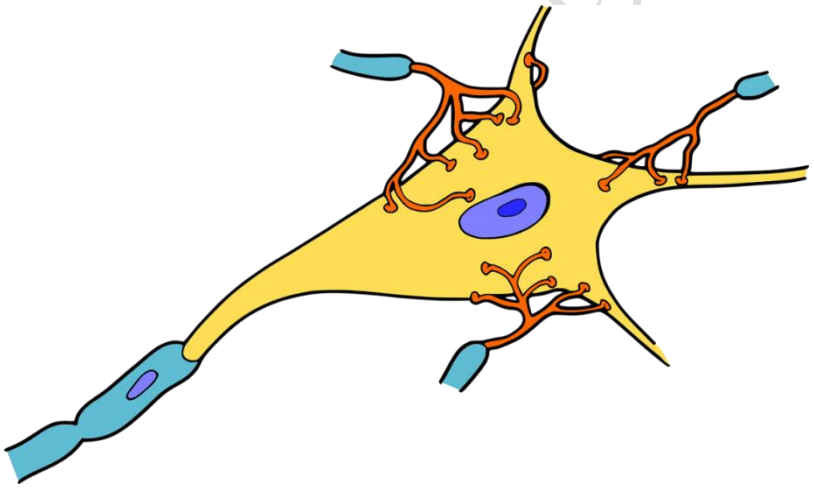
当我们看到一张画着猫的图片的时候，图片信息通过视神经传给大脑神经元，于是，信息到达了最左边一排竖着的黑点（神经元）。

神经元的激活与静止

假如一个黑点（神经元）之前见过猫，那么这个黑点就会把信息往后传，此时神经元处于激活状态。

假如一个黑点从来没见过猫，那么这个黑点（神经元）就啥也不知道，啥也不做，此时神经元处于静止状态。

像不像初中课堂上，老师问了你一个超难的问题，而你不知道的时候，你也只能站着，可怜又无助，啥也不会做？没错，神经元如果没见过猫，他啥也不会做！



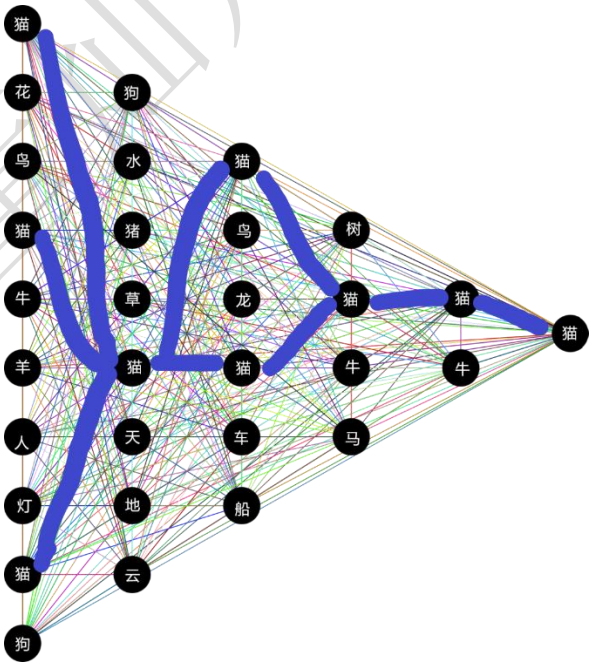
神经元

图片的信息就这样，一层一层的通过“见过猫且确信它是一只猫的”神经元往后传递，直到在最后输出一个结果。

这是一只猫。

这个过程叫做大脑的推理。

整个推理过程你应该注意到了一件事。所有的黑点（神经元），都可能是有记忆的，只不过记得东西各有不同，有的认识猫，有的认识狗，就像下面这样。



所有认识猫的神经元都会让信息通过，其他不认识猫的神经元都静止了。但

是只要信息能传到最后，人脑最终就可以得出一个结论，这就是一只猫。

那神经元的这些记忆是怎么获取的呢？当然是训练！

人们在日常生活中不断地训练大脑，时刻观察着周围的事物。见得多了，就会了！

训练，人工智能获取记忆

那么计算机又该怎么模拟这个记忆过程呢？答案很简单：因为计算机只会计算，那就让它计算好了。

如果某个黑点认识猫，有什么办法可以把“这是一只猫”这一信息传递到后面呢？乘以 1 啊，任何数乘以 1 都是它自己，一只猫乘以 1 也还是他自己。

如果某个黑点压根没见过猫，有什么办法可以什么都不做呢？乘以 0 啊，任何数乘以 0 都是 0，信息也就没了，一只猫乘以零，猫也就没了。

于是乎。

在深度学习的网络中，每个黑点（神经元）都有一个与之对应的数字（实际的网络中，不是 0 或者 1 这样简单的数字，而是一对复杂的数字，这里仅仅是为了说明示意），这些数字，在深度学习中，我们称之为权值。

神经元可以通过与权值的加权计算来判断是否让某一信息经过神经元，到达下一层。权值乘以输入的信息（猫），然后经过激活函数去激活（类似于人脑神经元的激活）。

如果能成功激活，那么信息就往下传。

如果没有成功激活，信息就在此丢失。

当然神经网络中的权值不是简单的 0 或 1，所以经过激活函数计算出来的只是一个概率值，也就是说黑点（神经元）觉得它是一只猫的概率。最终如果得到95%的概率觉的它是一只猫，那基本就是一只猫。

这个权值，就是 AI 的记忆。

这个权值，就是 AI 在训练的过程中学到的东西：千百万次计算得出的最优解。

这个权值，可以保证，只要 AI 在训练过程中看过猫，那么新的猫咪来的时候，猫咪乘以权值有很高的概率能通过激活函数，确保神经元被激活。为什么可以这么确定呢。

因为 AI 的训练过程早已经模拟了成千上万次“识猫”的过程了。

权值就是训练出来的！就像我们的记忆被训练出来的一样！...

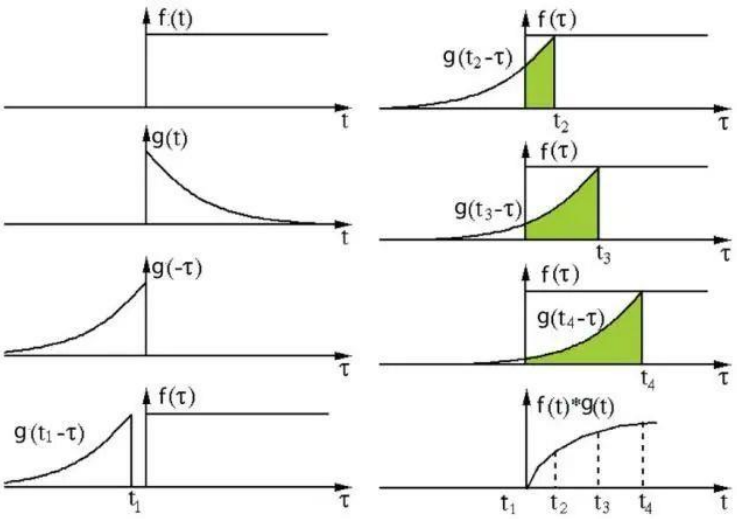
而卷积这一算法，就天然存在一个记忆体，或者说权值。

那就是卷积核。

卷积-Convolution

首先不要被这个名字吓到了。

不管数学好不好的同学，看到卷积的第一反应，可能是记得有一个卷积公式，貌似可以进行信号处理。

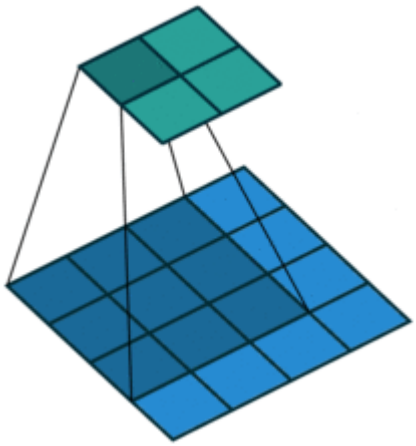


信号处理中的卷积过程

一个代表卷积核的曲线在原始信号曲线上滑来滑去，得到不同的输出。在什么地方学过来着？好像是时频转换的时候，又好像不是。(当然不是！)

但是，不用回忆之前的知识，不用管它！

因为，深度学习中的卷积，和信号处理中的卷积，有相似之处，但又不完全一样。深度学习中的卷积，完完全全模拟的，就是人眼看物体的过程！



卷积过程示意图

注：动图 pdf 不显示，可点击这里查看

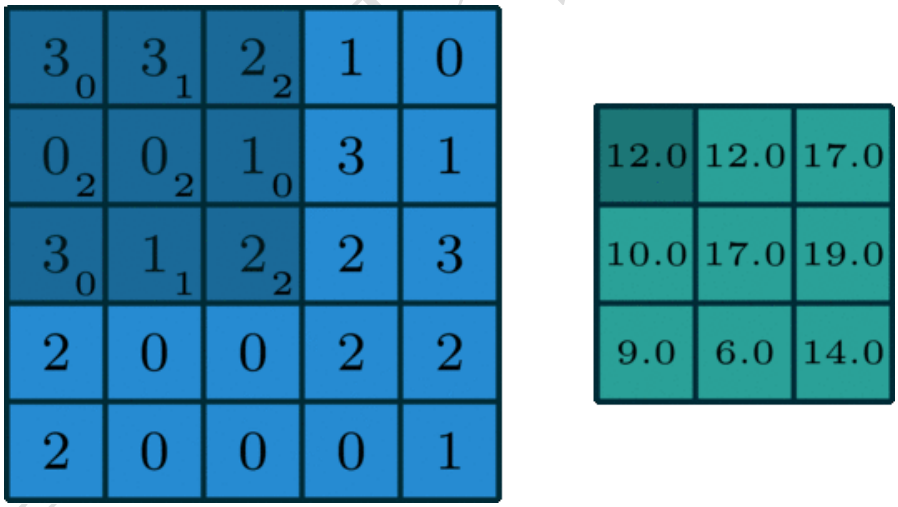
卷积模拟人眼

上图是深度学习中卷积的示意图。还记得之前说过的么，图片是由像素组成的。示意图下方的 4x4 的像素方格就是卷积需要处理的图片（模拟人眼观看图片的过程）。示意图上方的 2x2 的像素方格就是卷积的输出（人眼看到图片之后得出的结论）。

那么卷积核在哪？4x4 方格上移动的灰色阴影，3x3 的像素方格就是卷积核！

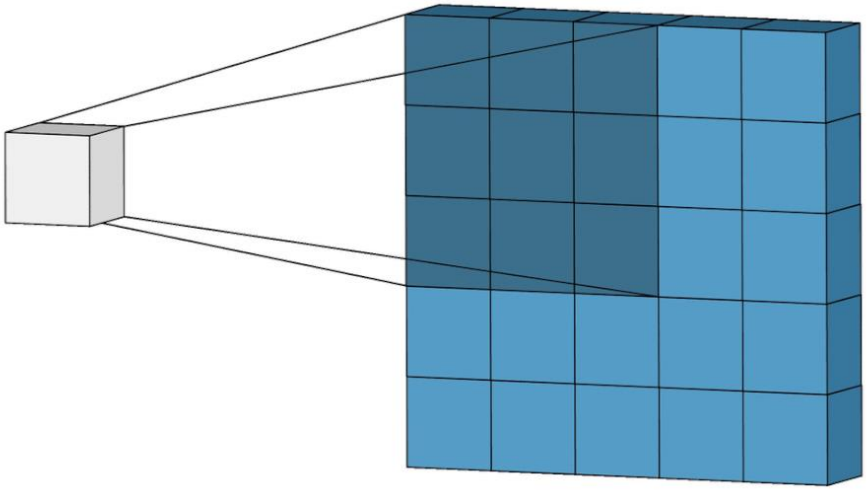
可以理解为人眼此时聚焦看到的区域（称之为**感受野**，人眼的视野），只不过，这个示意图中每次看到的都是一个 3x3 的像素方格！

而卷积过程，就是用 3x3 的卷积核，去逐步扫描图片。横着扫完竖着扫。每扫一次，就将逐个像素点的值相乘然后加一起，得到一个输出。



注：动图 pdf 不显示，可点击这里查看

再换个更直观的角度看一眼。



注：动图 pdf 不显示，可点击这里查看

上图第一次扫到的是左上角的 9 个点，与卷积核中 9 个点逐点相乘，然后相加，就得到了输出图片的左上角的一个点的值。

卷积，就是这么简单的过程。

我们可以通过调整卷积核的大小，比如把上图 3x3 的卷积核扩大到 5x5，来控制“人眼”看到的图片范围，从而获取到不同的图片信息。

当然，在实际神经网络中，存在这个各种各样的卷积变种。

科学家或工程师们通过设计不同的卷积核以及卷积每次移动的多少等参数，来实现不同的功能。

但卷积操作万变不离其宗！

总结一下

这一章主要聊了聊几个概念。

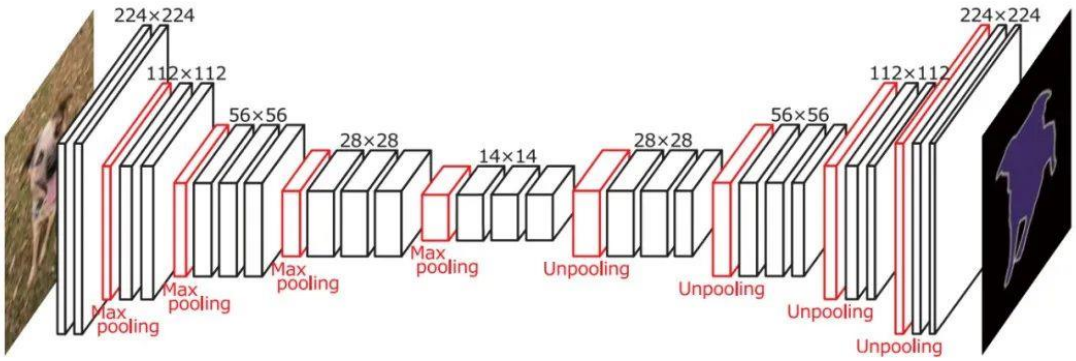
AI 之所以能够记住它所学的东西，关键在于神经网络有权值这一参数的存

在，它的存在就类似人脑的记忆。并且，权值和人脑的记忆一样，都是通过大量的实践来训练出来的。

而卷积这一算法，天然就存在一个权值参数，称之为卷积核，人们可以通过设计卷积核的大小，调整希望神经网络“看到”的图片的视野，也叫作感受野，从而不同的卷积获取到不同的信息。

实际上，卷积这一算法，除了本文说的利用“感受野”获取到不同图片区域的信息，从而将图片在长宽两个维度的尺寸缩放之外，还存在 channel 维度的升降。而这，才是卷积这一算法的核心，称之为特征提取。

下一章，聊一聊卷积是如何完成一张“画着猫的图片”的特征提取的。



五、卷积的特征提取

上一章聊到了卷积这一算法。

通俗点讲，卷积就是模仿的人眼识图的过程，以“感受野”的视角去扫描图片，从而获取不同区域的图片信息。

但其实，这并不是卷积算法的核心思想。卷积的核心，是通过设计多个卷积核，同时对一张图片进行卷积操作，以完成不同特征的提取。

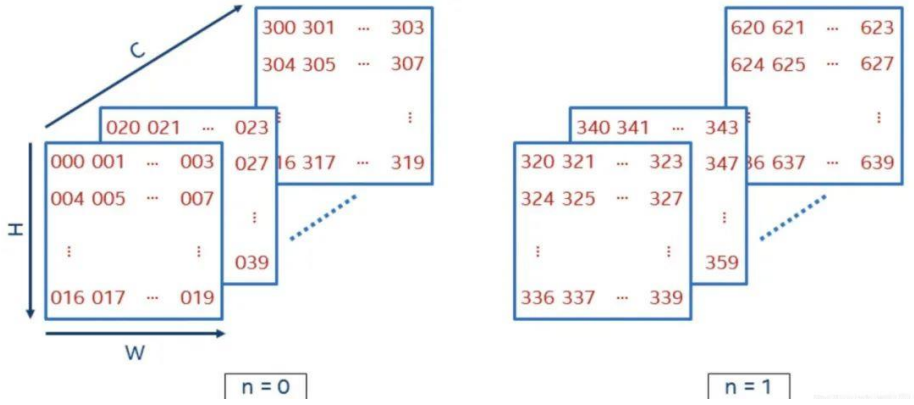
本章重点围绕特征提取这一概念，聊一聊卷积。

卷积的数学描述

不想看数学描述的同学可以略过，不影响后面的阅读。

有了之前文章的铺垫，这里说一张图片的尺寸是 [n, h, w, c]，应该不陌生了，其中，

- n 代表的是图片的张数。
- h 代表的是图片的高度，通俗的讲，高度方向上有多少像素。
- w 代表的是图片的宽度，通俗的讲，宽度方向上有多少像素。
- c 代表图片的通道数，例如 RGB 图片，c 等于 3。



图片的维度

不论是图片，还是卷积核，其数学描述都是具有 n,h,w,c 四个维度的数据。

因此，对于卷积算法而言，输入图片尺寸为 [n, hi, wi, c]（下标 i 代表 input，

输入)，卷积核尺寸为 [kn, kh, kw, c]，输出图片尺寸为[n, ho, wo, kn]（下标 o 代表 output)。

有没有发现，输出图片的 channel 数与输入图片的 channel 数不一致！

输出图片的 channel 数与卷积核的个数一致！

图片的特征

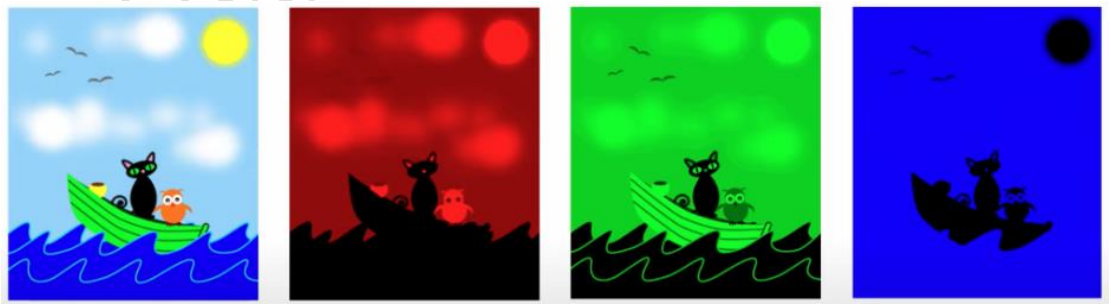
这意味着什么？还记得么，channel 代表的是图片的特征，如果我们想让图片呈现出 100 个特征，怎么办？

用卷积，使用 100 个卷积核计算，输出图片就具有 100 个特征！卷积算法，可以通过设计卷积核的个数，随意的提取图片的不同的数量的特征！

说的数学一点，卷积算法，就是通过线性变换，将图片映射到特征空间！

有点绕？没关系，只需要知道，卷积的核心，是提取图片的特征就行了。那么，特征怎么理解呢？

图像特征主要有图像的颜色特征、纹理特征、形状特征和空间关系特征。

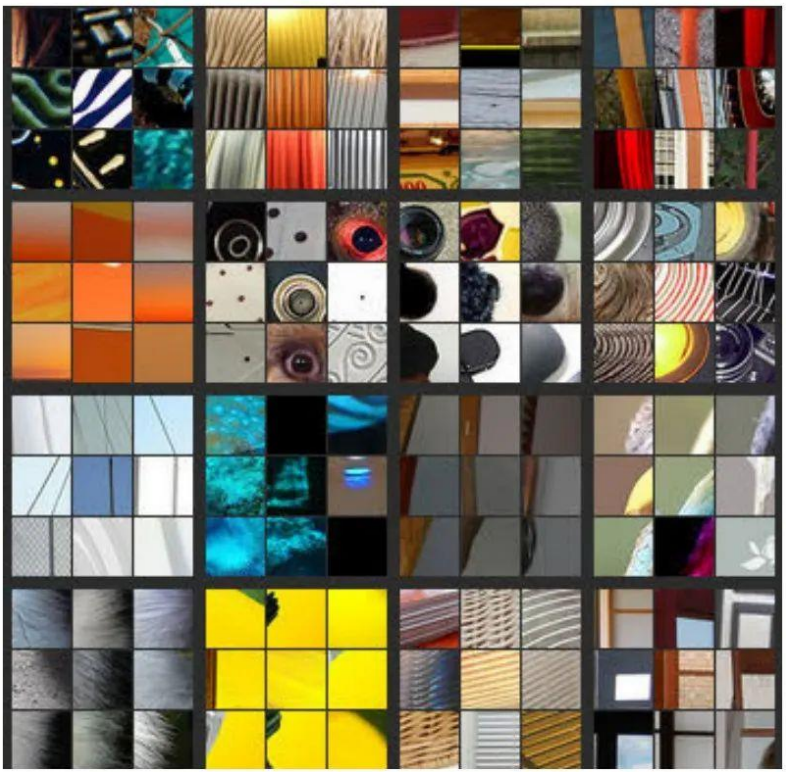


RGB 图像和它的 3 个特征通道

RGB 图片有 3 个通道，可以说有 3 个颜色特征，分别为红色，绿色和蓝色。

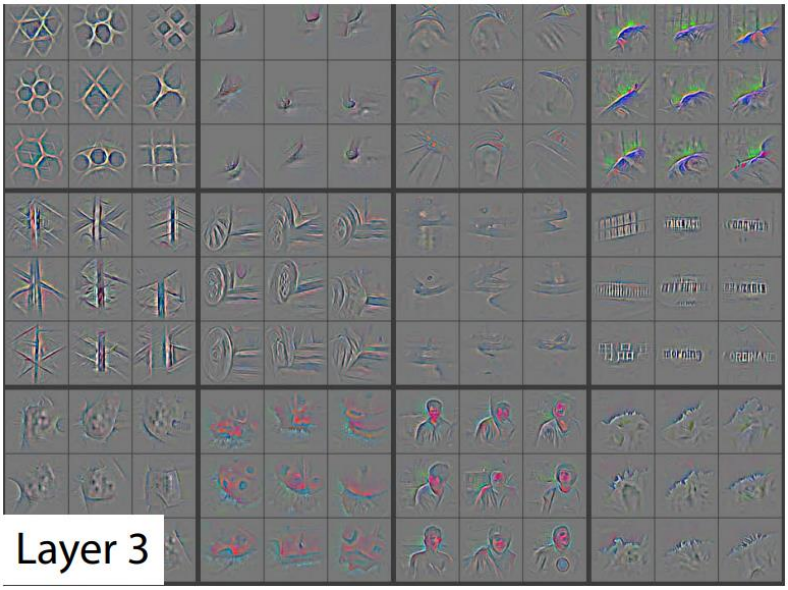
那么纹理特征，形状特征和空间特征又是什么意思呢？纹理特征就是图片的

纹理，比如下面这样。



图片中的纹理特征

形状特征就是图片中物体的形状，比如下面这样。

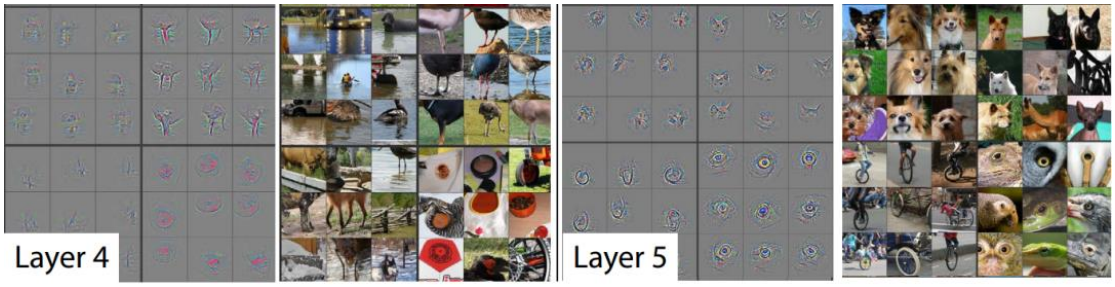


图片中的纹理特征

那么卷积这一算法在神经网络的训练过程中学习到这些特征了么？

答案是肯定的！

卷积不仅学到了这些特征，而且还学到了更多人们不太好描述的特征，这些特征对于人类来说可能毫无意义，但对于神经网络来说，确实十分重要的。



Resnet50 中各层卷积学到的特征，可视化

上图是著名的论文《Visualizing and Understanding Convolutional Networks》中的截图，文中提出通过反卷积这一算法，以可视化的视角，形象的展示卷积神经网络在训练过程中到底看到了什么。

所谓反卷积，通俗的理解就是卷积的逆运算。

可以看到，随着神经网络深度的不断加深，卷积提取到的特征逐渐清晰起来。由浅层次的纹理特征，逐步到深层次的形状特征！比如，在 Layer 4 中已经可以看到狗狗的形象！

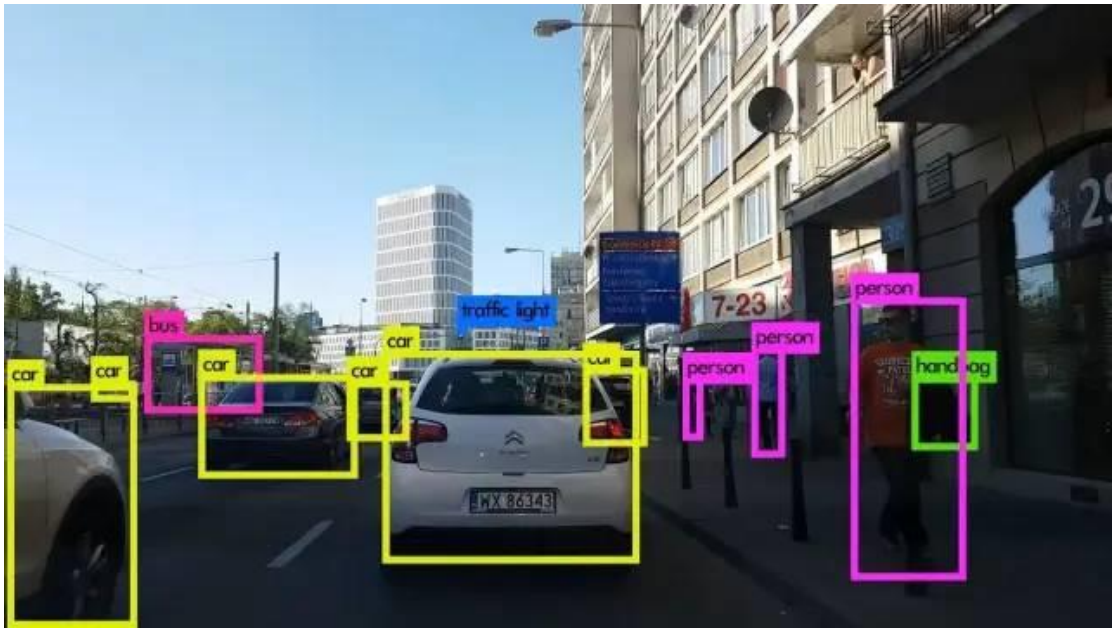
事实上，我们希望神经网络展现出来的是，看到一张画着小猫的图片，里面有一个代表猫的特征通道，该通道最终得分最高。

说到这里，即使你是一个 AI 算法小白，那也应该对卷积有了一些感性的认识。如果你希望了解到更多细节的东西，后面会逐步进行拆解。

Resnet 网络这篇文章的最后，介绍一下 Resnet 网络。就在昨天，深度学习大牛孙剑博士因病去世，在人工智能行业掀起轩然大波。孙剑博士领导研究的 Resnet 网络（残差网络），曾多次荣获图片分类大赛冠军，开启了一个卷积图像分类的热潮。

而本文重点拆解的就是 Resnet50 网络，这是一个图像分类网络。

所谓图像分类，就是它可以将一张图片进行分类。猫就是猫，狗就是狗，飞机就是飞机，大树就是大树。与图像分类不同的，还有图像检测网络。比如物体识别，需要在一张图片上准确的标注出物体是啥以及物体的位置。



图像检测

这些网络里都大量使用了卷积这一算法。因此这些网络我们也可以称之为卷积神经网络（Convolution Neural Network, CNN)。除了 CNN，还有循环神经网络（Recurrent Neural Network, RNN），之前写过一篇关于 LSTM 的文章 LSTM 中提到的 LSTM 就是一种 RNN 结构。

回到 Resnet50 这一卷积神经网络，这一网络由 50 个卷积层前后连接而成，因此叫 Resnet50，除此之外，还有 Resnet18，Resnet101 等，大致网络结构相似，只是卷积的层数不同。

为什么会有不同的卷积层数呢？神经网络在学习的时候，每一层学习到的特征是不同的，就比如第一层，它的输入只有 3 个特征，输出有 64 个特征，至于

这 64 个特征代表的是什么，可能连神经网络自己也说不清，它就只管学习。

一直到最后一层有 2048 个特征，到了最后一层，可以比较形象的这么比喻：最后一层共 2048 个特征，实际上已经代表了 2048 种物体的分类了。

针对一张图片是猫的原始输入，2048 个特征中，只有猫这一特征最后的得分最高，因此，网络会把它推理成猫。

这就是卷积算法的核心，特征提取。

六、残差结构

桃树、杏树、梨树，你不让我，我不让你，都开满了花赶趟儿。红的像火，粉的像霞，白的像雪。花里带着甜味儿；闭了眼，树上仿佛已经满是桃儿、杏儿、梨儿。花下成千成百的蜜蜂嗡嗡地闹着，大小的蝴蝶飞来飞去。野花遍地是：杂样儿，有名字的，没名字的，散在草丛里，像眼睛，像星星，还眨呀眨的。

朱自清在写《春》的时候，或许也没有完全认清春天的所有花，以至于写出了“有名字的，没名字的，散在草丛中”这样的句子。

如今，时代变了。

人手一部手机的我们，遇到不认识的花，随时随地就可以打开手机百度识图功能来完成识图。“杂样儿的，有名字的，有名字的，有名字的，有名字的 ... 都散落在手机里，像眼睛，像星星，还眨呀眨的”！而让我们如此轻松加愉悦的完成识图功能的，便是手机背后运行的大量卷积神经网络，或者说是 CNN 网络。

上一章聊了聊卷积的核心思想，那就是特征提取，文章最后也介绍了 Resnet50 这一图像分类网络。

我打算接着上篇末尾介绍 Resnet50 的逻辑，继续聊聊这个图像分类网络，以及它的思想。

为什么叫 Resnet50

研究 AI 网络的人拥有网络命名权。比如我研究出来一个网络，效果很好，要发一篇论文来介绍这个网络，论文中需要给网络起个名字，并且希望这个名字可以流传很广。那么，简单、好记同时又能概括网络思想的名字肯定是首选。

Resnet50 就是这样的名字，这个网络的核心思想，就藏在名字里

Res + net + 50，Res 是 Residual （残差）的缩写，50 指的是整个网络中有 50 个卷积层。

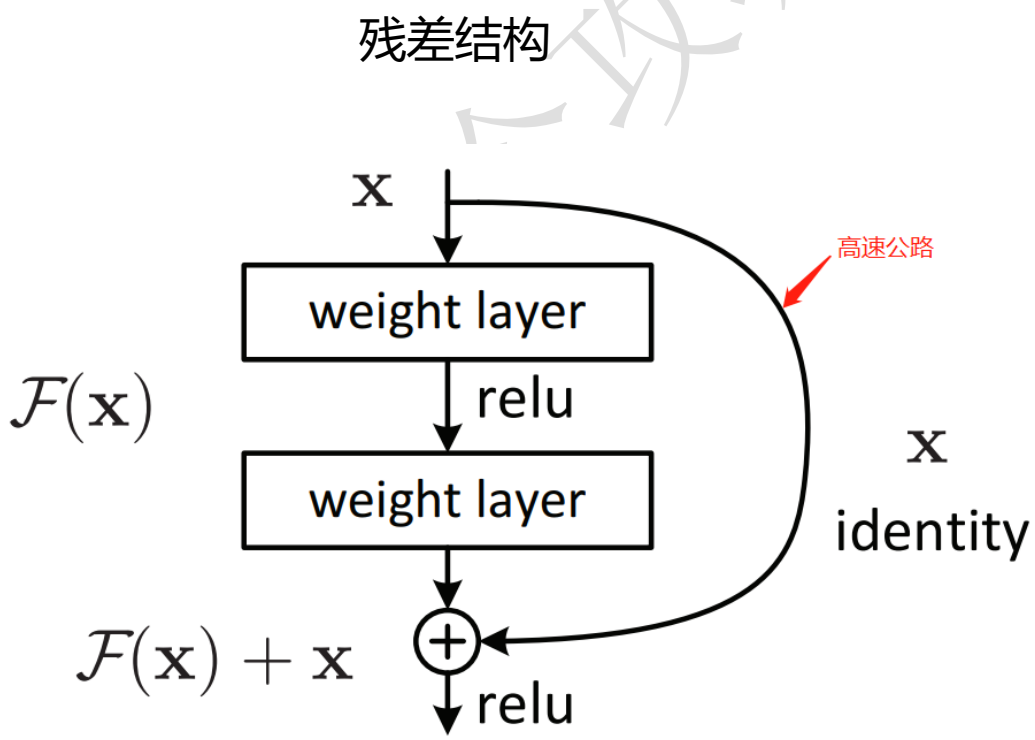
下图是 Resnet50 的网络结构图，可以看到，从第一层到最后一层，总共 50 个卷积算法。

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	第49层 7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×8
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23	[1×1, 256; 3×3, 256; 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax 第50层				

48层

Resnet50 的网络结构拆解，共 50 个卷积层

那么 Res（Residual）残差又是个什么东西呢?



所谓残差结构，其实就是在正常的神经网络中，增加一个 short cut 分支结构，也称为高速公路。

比如上图中，左侧是正常的卷积层，一层层往下传，在右侧增加一条连线，使得整个网络结构形成了一个残差结构。这样，网络的输出不再是单纯卷积的输

出 F(x)，而是卷积的输出和前面输入的叠加 F(x) + X。

为什么要增加残差结构

在前面说过，深度卷积神经网络在网络深度不断加深的过程中 ，神经网络会学到不同的特征。但是，能无限制地加深么？比如使用 1000 层卷积层进行网络的训练的。

答案显然是不行的。

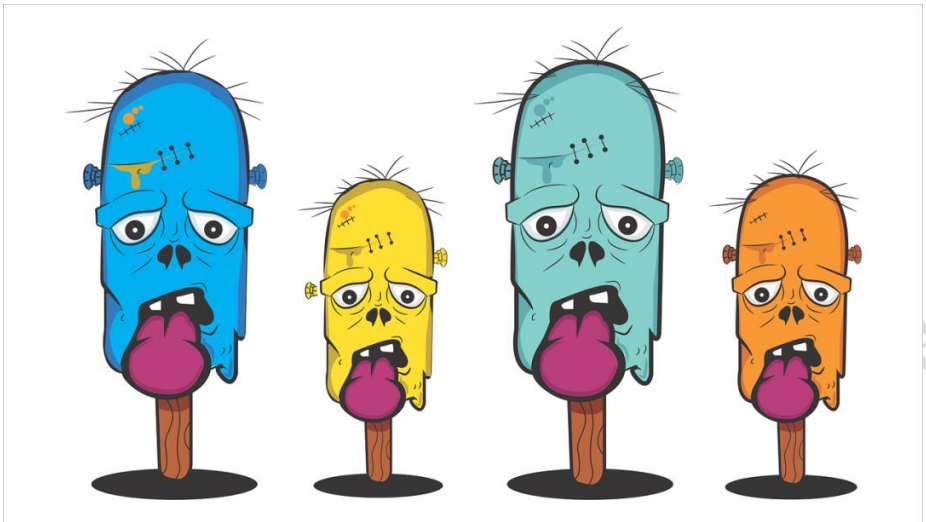
原因在于神经网络训练的过程是不断与目标值进行拟合的过程，直到拟合的误差降低到人们的预期，代表着神经网络训练完毕，一个会识图的 AI 就诞生了。

但是在实际训练过程中，数据的传递除了从网络前端往后传之外，还需要将最后一层与目标值的误差传回到网络前端，从而进行下一轮的训练，得到更小的误差，这一过程称为神经网络的反向传播。

在往回传的过程中，由于误差本身就很小，如果卷积层数过多，在经过激活函数时，很容易发生误差传着传着就消失了，称为梯度消失。

梯度消失的原因有很多种，不好的激活函数、过深的网络层数等都有可能导致误差消失。想象一下，上一轮训练结果的误差传不回来，下一轮如何在上一轮的基础上进行进一步优化训练？结果就会导致怎么训练神经网络最终的结果都

无法收敛。AI 根本训练不出来！

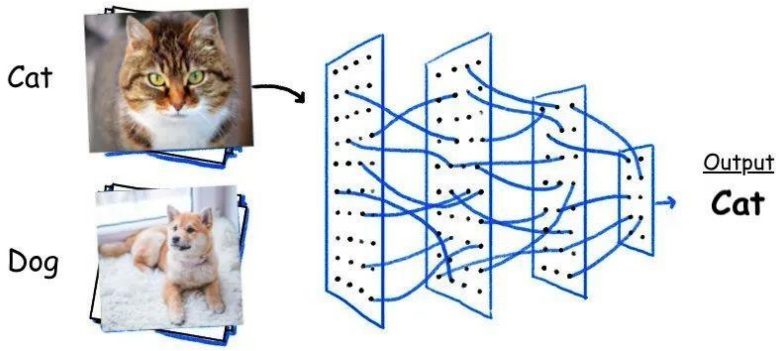


残差结构这个时候就可以发挥作用！

想象一下，这个高速公路的存在，可以使得输入数据无损地通过。如果左侧卷积层学习到的数据不够好，那么叠加上无损通过的原始数据，依然保留了原始数据，不至于丢掉原始数据。

而如果左侧卷积层学习到的效果很好，那么依然会保留着学习到的数据，下面的卷积层依然可以在这些数据基础上进一步学习优化。

反向传递也是一样，高速公路的存在，可以确保即使很小的误差也能传递过来，从而避免了梯度消失的发生。说回 Resnet50，这个网络就是通过 50 层卷积的计算，外加残差结构连接，来完成图像分类的。



实际上，目前各大公司直接使用 Resnet50 进行图像分类是很少的，大多数公司会在这个网络的基础上，结合自家公司的业务场景进行改造，或者直接借鉴 Resnet50 的网络设计思想，重新设计新的网络，以期获得更加高效的识图效果。看到这，你或许能够了解，当我们打开百度识图完成图像识别时，它的背后，可能不是 Resnet50 这一网络，但肯定是有卷积和残差这两个算法！

Resnet——简单，暴力，有效

Resnet50 网络的结构其实说简单，它很简单，而且算法思想也很简洁，就是 50 层卷积的计算，依据卷积局部感受野这一特性，抽取出图像的不同特征，通过最后一层卷积（或者叫做全连接）将图片进行分类。这样的网络设计，分类效果很好，使得 Resnet50 多次在图像分类大赛中夺冠！

Resnet50 除了大量使用了卷积这一算法之外，一个简单暴力的残差结构的应用，使得该网络无论在训练还是推理过程中，其效果都极为出彩！

从此，残差这一结构，受到了人们的关注，以至于，有人开始专门研究不同层之间的残差连接。

一句话，Resnet50 的核心是卷积和残差，卷积的核心是特征抽取。

加餐——大数吃小数

反向传播中，一个很小的误差，在反向传播经过激活函数（比如 Sigmoid 激活函数）时，有可能结果为零，这就是上面我们说的梯度消失。

但实际上，在计算机的科学计算中，同样存在一个很有趣的事情，使得很小的数在参与计算的过程中，并不起作用。这就是大数吃小数。什么意思呢？

如果你用一个超大的数，去加上一个很小的数，你会发现，结果和你的预期是不一致的。比如我有以下简单的 C 语言代码，一个超大数是 20000000， 一个很小数是 1，两个数相加。

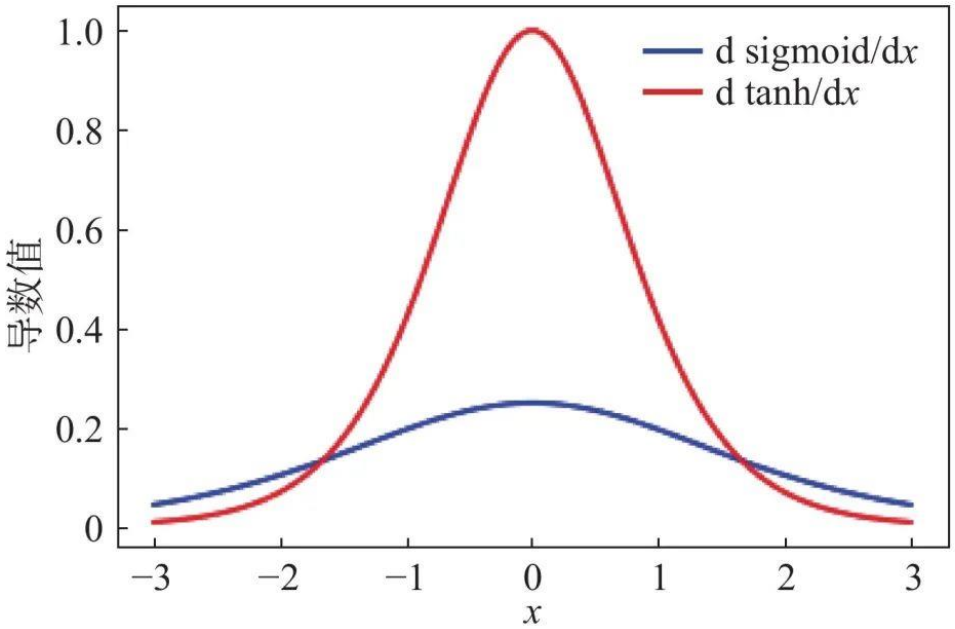
```
#inlcude "stdio.h"
int main() {
   float big = 20000000;
   float small = 1;
   printf("big = %f\n small = %f\nbig+small = %f\n", big, small, big + small);
}
```

```
big = 20000000.000000
small = 1.000000
big + small = 20000000.000000
```

你可能期望得到的结果是 20000001，但是结果却依然是 20000000！很明显，小的数字 1 被大数 20000000 吃掉了！你可以运行上述代码实际测试一下。为什么会有大数吃小数这一现象呢？

这就跟数据在内存的存储格式有关了，感兴趣的同学可以查看 IEEE754 标准的浮点数在内存中的表示。

如果过去、现在或者未来，你发现你的程序出现了这种不可思议的错误，尤其是在你做好几万次加法循环的时候，不要怀疑出现了幻觉，也不要怀疑人类科技被智子锁死了，翻一翻计算机标准，你会豁然开朗的。



七、激活函数

上一章说到了 Resnet 网络的残差结构。也就是人们俗称的高速公路。

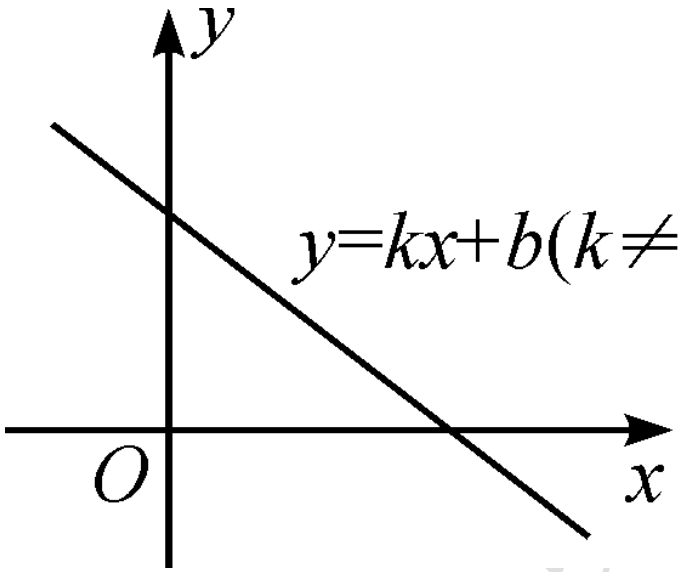
看过那一章的同学或许有印象，一层层搭建而成的神经网络，由卷积层、激活层、池化层等组成。而每一个卷积层的后面，都会跟着一个激活层。

在 Resnet50 中，激活函数用的是 Relu 激活函数。那为什么在神经网络中，每一层卷积后面都需要跟着一个激活函数呢？

非线性

敲黑板，划重点，为了非线性。

我们都学过线性关系，最简单的 y = kx + b，画出来就是一条直线。这个函数就是一个线性函数，称 y 和 x 是线性关系。



线性函数

如果这个时候，又有一个线性关系 z = hy + d，那么，可以通过如下的线性变换，得到变量 z 和 x 同样也是线性关系！

z = hy + d = h(kx+b) + d = hkx + hb + d = Ax + B

其中：A = hk， B = hb + d。

所以，不管有多少个线性关系，只要在数学上首尾相连，最终都可以等效成一个线性关系！

而在深度学习任务中，如分类任务，具有线性关系的模型其分类效果是不好的，甚至是很差的。

因为卷积算法是由大量的乘法和加法组成，所以，卷积算法也是线性的！

这就导致，由大量卷积算法组成的卷积神经网络（CNN），如果没有非线性因素的引入，会退化成一个简单的线性模型。这就使得多层卷积失去了意义。

比如，Resnet50 网络中的 50 层卷积，就会退化为一个卷积。而在神经网络

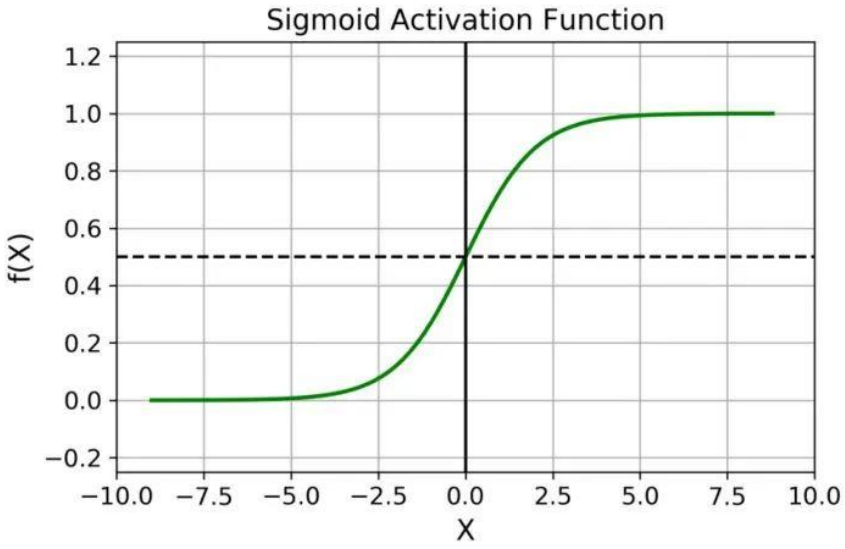
中，使用多层卷积的一个重要目的，就是利用不同卷积核的大小，来抽取不同卷积核尺度下的图像特征。

因此，在神经网络设计时，在每层的卷积后面，都增加一个非线性函数，就可以完成两个卷积层的线性隔离，确保每个卷积层完成自己的卷积任务。

目前常见的激活函数，主要有 Sigmoid、tanh、Relu 等。Resnet50 中的激活函数就是 Relu。

下面主要介绍下这三个函数。

sigmoid



Sigmoid 函数的图像看起来像一个 S 形曲线。公式为：

f(z) = 1/(1+ e^-z)

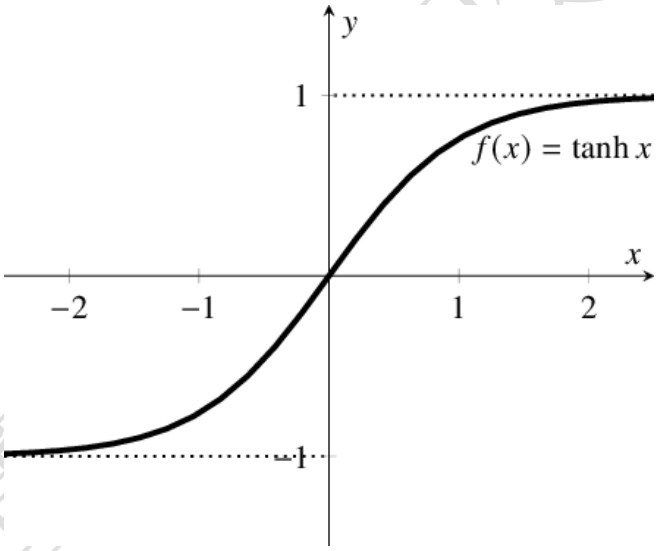
Sigmoid 在神经网络中使用，是有一些优点的，主要体现在：

- Sigmoid 函数的输出范围是 0 到 1。由于输出值限定在 0 到 1，因此它

对每个神经元的输出进行了归一化；

- 用于将预测概率作为输出的模型。由于概率的取值范围是 0 到 1，因此 Sigmoid 函数非常合适；
- 梯度平滑，避免「跳跃」的输出值；
- 函数是可微的。这意味着可以找到任意两个点的 sigmoid 曲线的斜率；
- 明确的预测，即非常接近 1 或 0；

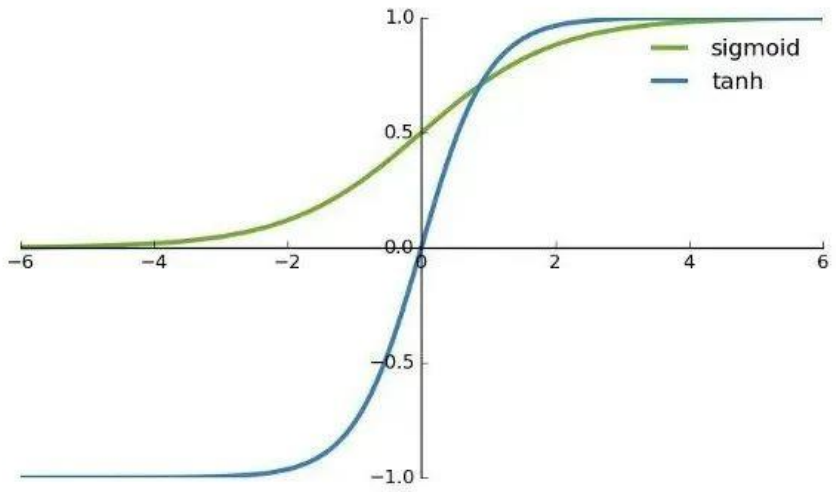
tanh



tanh 激活函数的图像也是 S 形，表达式如下：

$$f(x) = tanh(x) = \frac{2}{1+e^{-2x}} - 1$$

tanh 是一个双曲正切函数。tanh 函数和 sigmoid 函数的曲线相对相似。但是它比 sigmoid 函数更有一些优势。

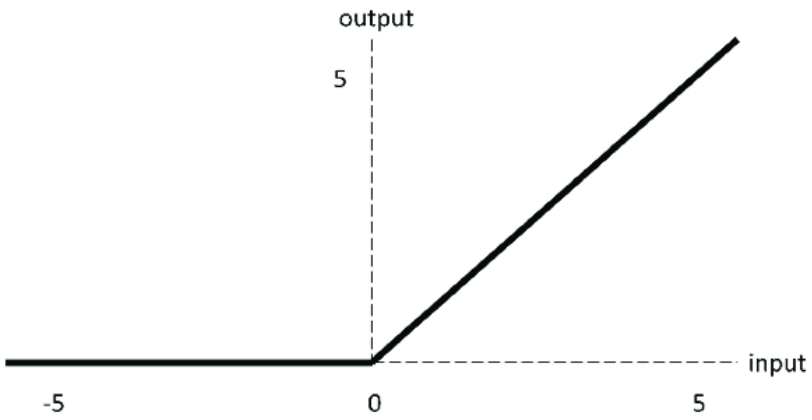


sigmoid 函数和 tanh 函数对比

首先，当输入比较大或者比较小时，函数的输出几乎是平滑的并且梯度较小，这不利于权重更新。二者的区别在于输出间隔，tanh 的输出间隔为 1，并且整个函数以 0 为中心，比 sigmoid 函数更好；

在 tanh 图中，负输入将被强映射为负，而零输入被映射为接近零。

Relu



ReLU 激活函数图像如上图所示，函数表达式如下：

$$\sigma(x) = \begin{cases} max(0,x) & ,x >= 0 \\ 0 & ,x < 0 \end{cases}$$

ReLU 函数是深度学习中较为流行的一种激活函数，相比于 sigmoid 函数和 tanh 函数，它具有如下优点：

- 当输入为正时，不存在梯度饱和问题。
- 计算速度快得多。ReLU 函数中只存在线性关系，因此它的计算速度比 sigmoid 和 tanh 更快。

当然，它也有缺点：Dead ReLU 问题。当输入为负时，ReLU 完全失效，在正向传播过程中，这不是问题。有些区域很敏感，有些则不敏感。但是在反向传播过程中，如果输入负数，则梯度将完全为零，sigmoid 函数和 tanh 函数也具有相同的问题；

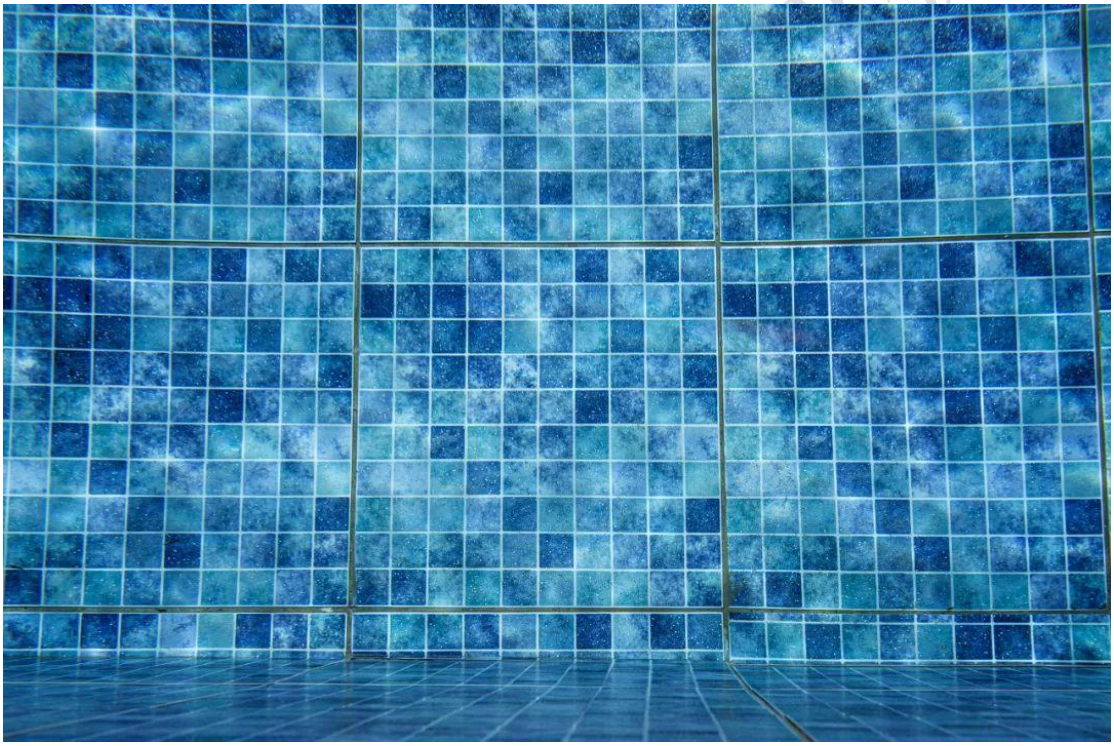
我们发现 ReLU 函数的输出为 0 或正数，这意味着 ReLU 函数不是以 0 为中心的函数。

除了上面的 3 种激活函数之外，还有很多其他激活函数，比如 Relu 函数就有很多变种，如 PRelu、LeakyRelu 等。

每种激活函数，都在一种或几种特定的深度学习网络中有优势。判断一个激活函数的好与坏，绝不仅仅是从函数的数学表达式上来判断，而是需要在实际的深度学习网络中不断地实验和实践，来找到最适合这个网络的激活函数。

总结一下

之所以在神经网络中添加激活函数，一个重要的原因是给网络模型增加非线性因素。目前已有的激活函数有很多种，每种激活函数，不论从数学原理上还是从适用的 AI 模型上，都有各自的优缺点，需要根据网络特点和适用场景，进行实验，选择最适合这个模型的激活函数。



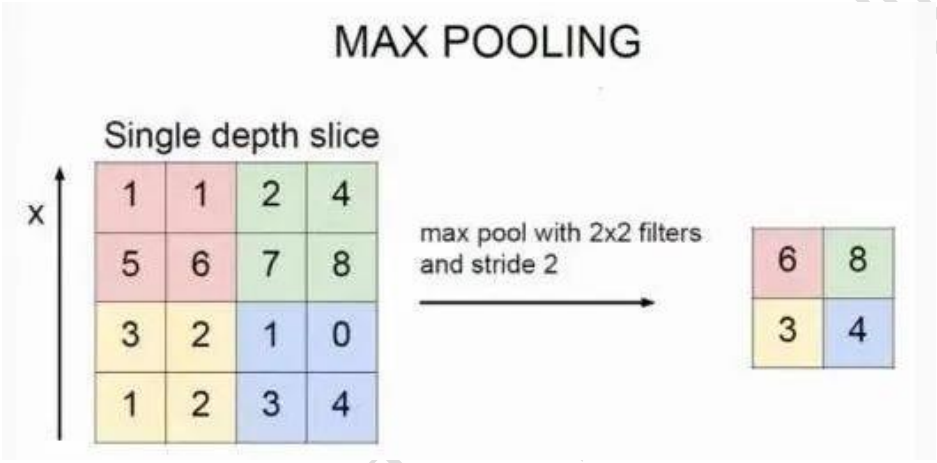
八、池化

在之前写的文章 Resnet50 网络结构中，展示了一张 Resnet50 的网络结构。在该网络结构中，存在一个最大池化层和一个全局平均池化层，而在其他的 CNN 网络中，也会时而看到池化层的出现。那么，什么是池化层呢？在 CNN 网络中，

池化层又能起到什么作用？

池化一般接在卷积过程后。

池化，也叫 Pooling，其本质其实就是采样，池化对于输入的图片，选择某种方式对其进行压缩，以加快神经网络的运算速度。这里说的某种方式，其实就是池化的算法，比如最大池化或平均池化。



最大池化

池化过程类似于卷积过程。

上图表示的就是对一个图片邻域内的值，用一个 2x2 的池化 kernel，步长为 2 进行扫描，选择最大值输出，称为最大池化。

最大池化 MaxPool 常用的参数为 kernel = 2, stride = 2 ，这样的参数处理效果就是输出图片的高度、宽度减半，通道数不变。

还有一种叫平均池化，和最大池化类似，就是将取区域内最大值改为求这个区域的平均值。

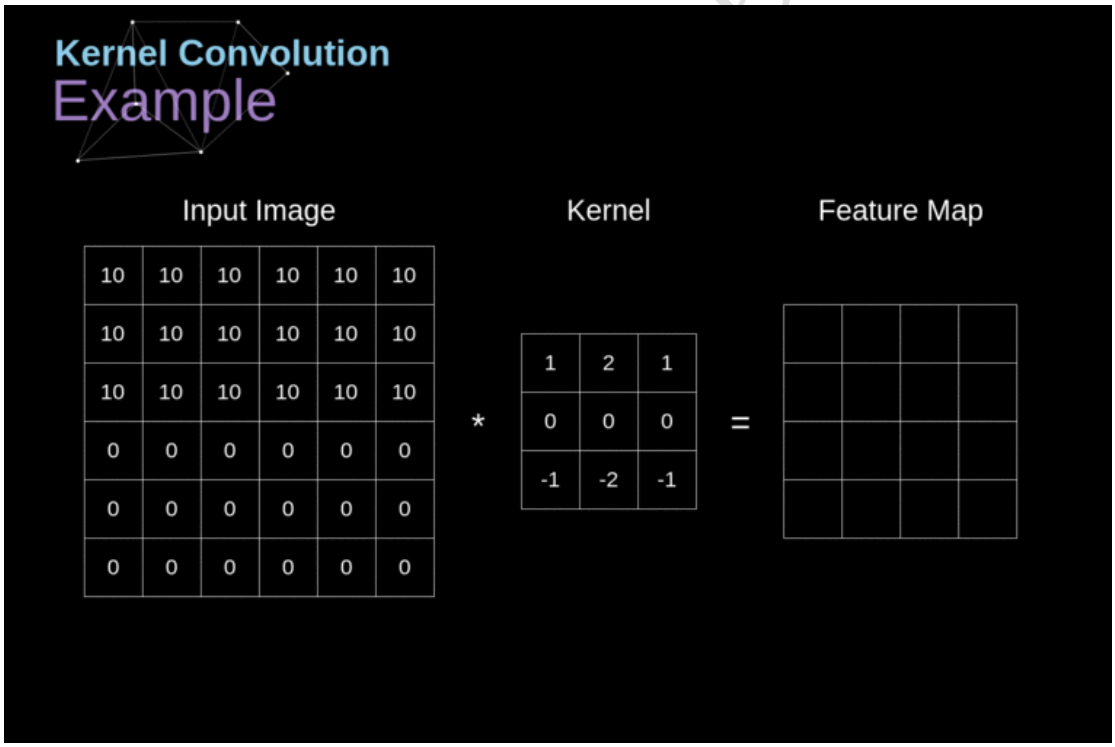
和卷积类比

和卷积类比，池化操作也有一个核(kernel)，但它不是卷积核。

池化的核只负责框定某次池化计算需要的图片的范围，核里面并没有数据参与计算，也就是说，在训练过程中，池化层不像卷积层那样，需要学习权重。

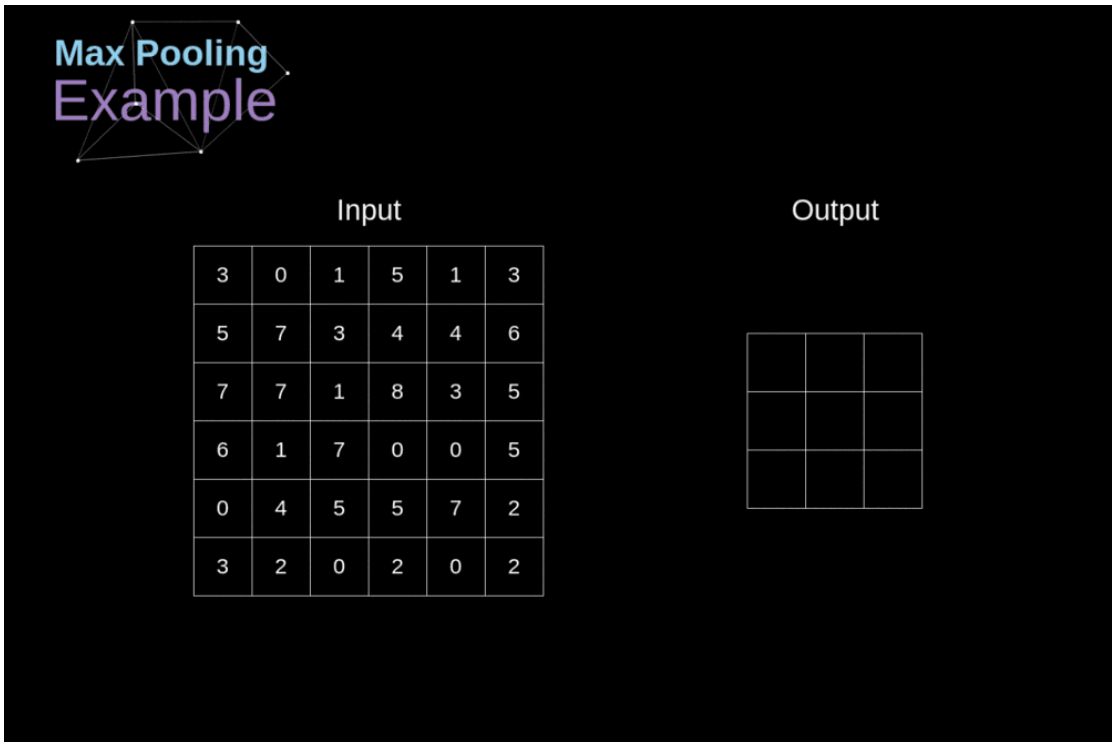
另一个与卷积不同的是，在卷积的计算中，需要 channel 维度的数据累加，而池化层的 channel 维度的数据不需要累加，每个 channel 中的数据是独立的，这也导致，池化的运算复杂度比卷积简单很多

下图是卷积和池化的示意图，通过两张图，大致可以看出两者的不同。



卷积示意图

注：动图 pdf 不显示，可点击这里查看



池化示意图

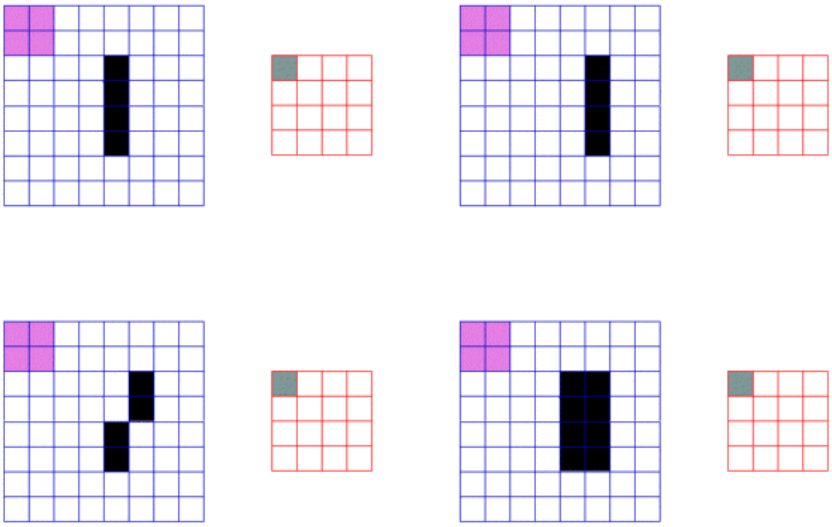
注：动图 pdf 不显示，可点击这里查看

神经网络中为什么需要池化层

特征不变性

池化层支持了一定的平移、旋转、拉伸不变性，这个特性有点抽丝剥茧的意思，用小特征对大特征进行精简。

如下图，通过池化操作，图片中的黑色特征在输出图片中，仍然被保留了下来，虽然有些许的误差。



池化的特征不变性

注：动图 pdf 不显示，可点击这里查看

降维

如上的例子，图片经过池化的操作，可以减小图片的尺寸，同时又可以保留相应特征，所以主要用来降维。

防止过拟合

由于池化层没有需要学习的参数，因此，在训练的过程中，可以在一定程度上防止过拟合的发生。

降低模型计算量

池化的操作，会在保留原始图片特征不变的情况下，将图片尺寸缩小，从而减少整个模型的计算量。

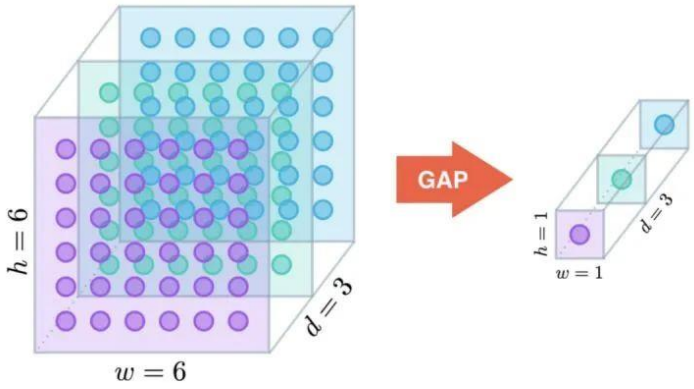
在神经网络的训练和推理过程中，一个维度的计算量减倍，往往会带来一个数量级的性能提升，尤其是在训练过程动辄迭代成千上万次的训练场景中。

使用池化算法，在减少图片的宽和高尺寸的同时，也会给模型的训练和推理带来更优异的性能提升。

加餐

除了上述最大池化和平均池化外，池化还有很多变种。最常见的一种变体就是全局池化。

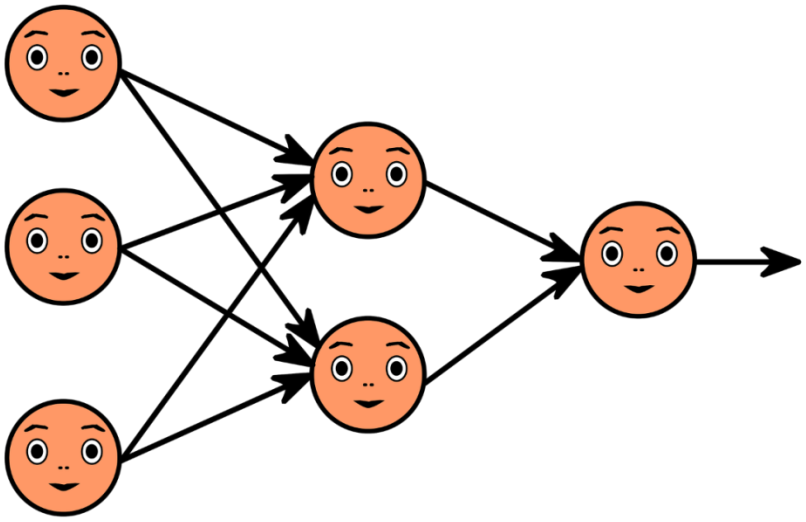
全局池化的 kernel 大小和图片大小一样，因此最终输出的图片大小就是一个点。这种全局池化操作，后面一般用来接全连接层，从而进行分类。如 Resnet50 最后一层全连接层前，就是一个全局平均池化层。



九、全连接

前面的一章说到了 CNN 网络中的池化层，池化层一般接在卷积层后面，用来完成特征图的降维和特征融合操作。

在 CNN 网络的最后，一般还会有一个全连接层（Fully Connected Layer）。CNN 中为什么还需要一个全连接层呢，它的作用是什么？这篇文章会告诉你答案。

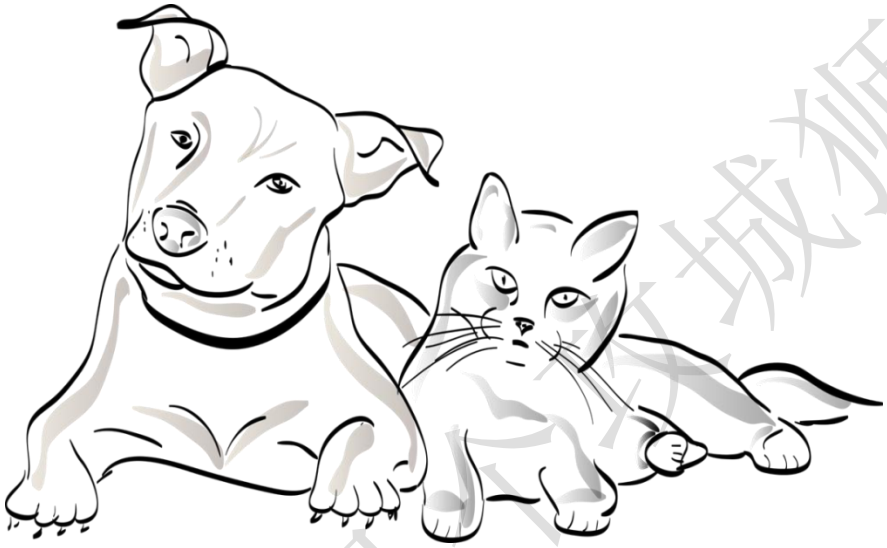


全连接层，指的是每一个结点都与上一层的所有结点相连（示意图如上图所示），用来把前面几层提取到的特征综合起来。由于其全连接的特性，一般全连接层的参数也是最多的。

第四章提到卷积的作用是完成图像的特征提取，那提取出了特征之后，还是无法根据提取的一堆特征来完成图像的识别。因为卷积层提取出来的特征太多了！

举个例子，一张画着猫咪的图片，经过几十层卷积的特征提取，很有可能已经提取出了几十个甚至上百个特征，那我们如何根据这几十上百个特征来最终确认，这是一只猫呢？

把上面的问题细化并且简化一下，不说几十上百个特征，就说卷积层只提取了 3 个特征：分别是鼻子，耳朵和眼睛。实际上，有鼻子、耳朵和眼睛这三个特征的动物有很多，我们并不能只根据某个动物有鼻子、耳朵和眼睛，就把它简单的认为是一只猫。



那么就需要一种方法，把鼻子、耳朵和眼睛这三个特征进一步融合，使得神经网络看到这三个特征的融合集合之后，可以区分这是一只猫而不是一只狗。

上面的例子比较简单，实际网络中卷积提取的特征远远不止 3 个，而是成百上千个，将这些特征进一步融合的算法，就是全连接。

或者说，全连接，可以完成特征的进一步融合。使得神经网络最终看到的特征是个全局特征（一只猫），而不是局部特征（眼睛或者鼻子）。知乎上对于这个问题有个比较形象的回答，大意是说：

假设你是一只蚂蚁，你的任务是找小面包。这时候你的视野比较窄，只能看到很小一片区域，也就只能看到一个大面包的部分。
当你找到一片面包之后，你根本不知道你找到的是不是全部的面包，所以你们所有的蚂蚁开了个会，互相把自己找到的面包的信息分享出来，通过开会分享，最终你们确认，哦，你们找到了一个大面包。

上面说的蚂蚁开会的过程，就是全连接，这也是为什么，全连接需要把所有的节点都连接起来，尽可能的完成所有节点的信息共享。

说到这，大概就能理解全连接的作用了吧。

卷积和全连接

其实有两首诗可以很形象的概括卷积和全连接的作用。

我们知道卷积是对图像的局部区域进行连接，通过卷积核完成的是感受野内的长宽方向以及 channel 方向的数据连接。因此，卷积操作，提取的特征是局部特征。也就是说，卷积是“不是庐山真面目，只缘身在此山中”。

而全连接层呢？它的每次完成的是所有 channel 方向的连接，它看到的是全局特征。全连接是“不畏浮云遮望眼，自缘身在最高层”。

除此之外，卷积和全连接在算法上是可以转换的。通常情况下，在进行全连接的计算时，可以把它等效于卷积核为 1x1 的卷积运算。

总结一下

全连接的作用，说的学术专业一点，就是把卷积层学到的特征空间映射到样本标记空间。说的通俗易懂点，就是把卷积学到的一堆特征互相融合一下，变成样本（比如一只猫）的代表。

在使用 Resnet50 对 ImageNet2012 数据集进行分类时，最终完成某个图片的分类，全连接层会输出一个值。其中每个值都会对应一个索引，在 ImageNet 中，

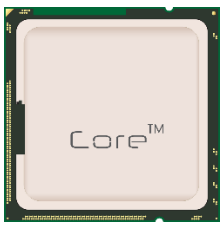
索引 281-287 都代表猫。比如 282 这个索引，代表的是一只虎猫，而这个值，就是把所有的虎猫的特征进行了融合后计算而来的。

- 281 n02123045 猫, tabby, tabby cat
- 282 n02123159 猫, tiger cat
- 283 n02123394 猫, Persian cat
- 284 n02123597 猫, Siamese cat, Siamese
- 285 n02124075 猫, Egyptian cat
- 286 n02125311 猫, cougar, puma, catamount, mountain lion, painter, panther, Felis concolor
- 287 n02127052 猫, lynx, catamount

加餐

看到这，可能有人会问，既然全连接层处理的特征比卷积层信息更丰富，那为什么在 CNN 网络中进行图像识别和分类时，我们还大量的使用卷积而不全部使用全连接呢？

答案很简单。全连接由于连接了上一层所有的节点，需要的模型参数更多，计算更密集。一个普通的卷积神经网络，如果用全连接来实现，你可以试试，分分钟挤爆你的 CPU，甚至你的显卡。



十、SoftMax 分类

很多同学在做深度学习时，都会遇到难以理解的算法，SoftMax 肯定是其中一个。初学者大都对它一知半解，只知道 SoftMax 可以用来做分类，输出属于某个类别的概率。

但是，为什么要用 SoftMax 呢？这个算法又是如何将神经网络推理的数值，转换为一个类别的分类的呢？

应用场景

假设要使用神经网络做图片分类。现在有 3 个类别：猫，狗，人。给你下面一张图片，神经网络需要在这 3 个类别中选出一个。

上图人眼一看就知道是猫咪，但是神经网络需要通过计算才知道。好，我们使用 Resnet50 这一分类网络进行推理运算。算到最后面的全连接层时，全连接输出了 3 个数值，分别为 2，1，0.1。

看过前面文章的同学可能知道，全连接输出的数值，代表了这一分类的得分。

现在我们假设这三个分类的得分分别为：

分类	得分
猫	2
狗	1
人	0.1

猫得了 2 分，狗得了 1 分，人得了 0.1 分。单看这个结果，我们大概知道，因为猫的得分最高，那最终神经网络会认为这张图片是一只猫。

错了！错在哪？至少两点。

第一，神经网络最终选择某一分类，依据的不是得分，而是概率也就是说，最终神经网络会选择一个概率最高的分类作为它识别的结果。

为什么要把得分转为概率呢？因为多分类模型中，输出值为概率更利于反向推导和模型的迭代，概率之间更好的计算距离，而数值之间的计算的距离是无含义的。所以，我们需要一种方法，将上面的得分转换为概率。

第二，得分是神经网络经过了几十层卷积运算计算出来的例子中猫的得分是 2，狗的得分是 1，人的得分是 0.1，我们可以比较肯定的说，因为猫的得分最高，而且比狗和人都高很多，肯定就是猫。

但实际中，有很大的可能算出的猫的得分是 2.1，狗的得分是 1.9，人的得分是 0.1。这个时候，我们可能就没有像刚才那么肯定了。因为猫的得分和狗的得分相差很少，而且两者都很高！

这也是为什么，很多神经网络最终都会以 TOP1 和 TOP5 的识别准确度来衡量神经网络的精度。

由于上述两个原因的存在，人们想到了 SoftMax 算法。而这个算法，也几乎完美地解决了这两个问题。

为什么叫 SoftMax 以及它的实现原理

不知你有没有想过，为什么这个算法叫 SoftMax 呢？Soft 是软的意思，与之对应肯定有 HardMax。

而 HardMax，可以理解为我们平时认知的 Max。比如给你两个数（3, 4), 那么这两个数的 HardMax(3,4) 结果就是 4。这个逻辑，小学生学会了 10 以内的加减法都知道。

但正如上面所说，SoftMax 不一样，它是要处理多个类别分类的问题。并且，需要把每个分类的得分值换算成概率，同时解决两个分类得分值接近的问题。

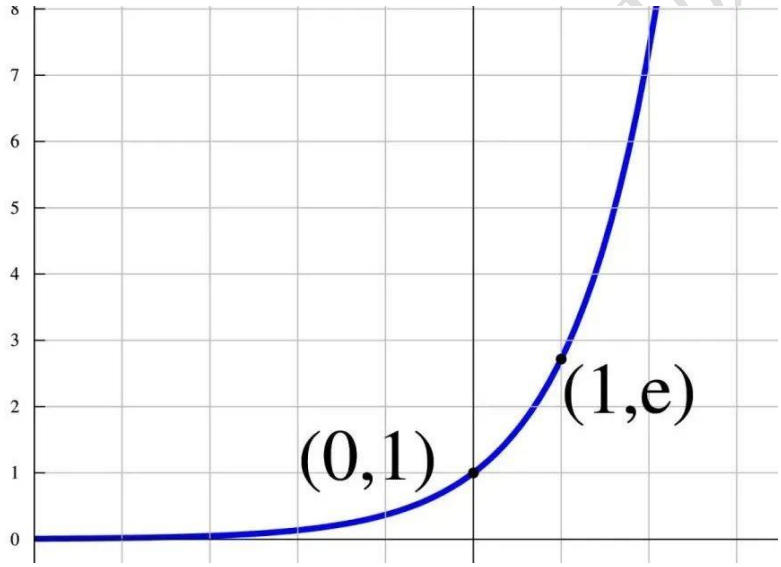
先从公式上看，SoftMmax 是怎么做到的。公式中，每个 z 就对应了多个分类的得分值。SoftMax 对得分值进行了如下处理：

- 以 e 为底数进行了指数运算，算出每个分类的 eZi，作为公式的分子
- 分母为各分类得分指数运算的加和。

- 根据公式很自然可以想到，各个分类的 SoftMax 值加在一起是 1，也就是 100%。所以，每个分类的 SoftMax 的值，就是将得分转化为了概率，所有分类的概率加在一起是 100%。

这个公式很自然的就解决了从得分映射到概率的问题。那，它又是怎么解决两个得分相近的问题的呢？

其实也很简单，重点在选择的指数操作上。我们知道指数的曲线是下面的样子。



指数曲线，恒大于零，并且在正半轴，离零越远，增长越快（指数增长）

指数增长的特性就是，横轴变化很小的量，纵轴就会有很大的变化。所以，从 1.9 变化到 2.1，经过指数的运算，两者的差距立马被的拉大了。从而，我们可以更加明确的知道，图片的分类应该属于最大的那个。下面是将猫、狗、人三个分类经过 SoftMax 计算之后得到的概率。

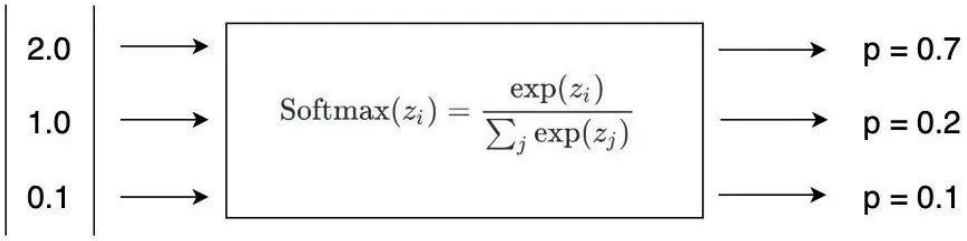
分类	得分	softmax

猫	2	70%
狗	1	20%
人	0.1	10%

可以看到，分类是猫的概率遥遥领先。所以，神经网络在经过 softmax 层之后，会以 70%的概率，认为这张图片是一张猫。

这就是 SoftMax 的底层原理。

指数让得分大的分类最终的概率更大，得分小的分类最终的概率更小，而得分为负数的分类，几乎可以忽略。



加餐

SoftMax 其实也是一种激活函数，它广泛的应用于多分类任务中。在二分类任务重，其实有个函数也被广泛的使用，它就是 Sigmoid，可以查看第六章了解 Sigmoid 函数。

比如，有个朋友告诉我，在一些互联网的广告或者商品推荐（比如某宝的猜你喜欢）中，曾经广泛的使用 Sigmod 函数来预测点击的可能性，如果 Sigmoid 函数的输出值越大，那么说明这个内容被用户点击的可能性就越大

想想吧，我们逛淘宝的每次点击，背后都有一个函数在分析你的行为，你还会点击么？

免责声明

本文旨在深度学习相关技术交流，非商业目的。

作者撰写本文的目的是技术交流和学习总结。本文文字和表达以及相关算法引申基本都为原创，但由于要展示一些算法的可视化（比如卷积的动图），会从网络上引用部分图像，作者已经尽量减少对于他人资料的直接引用，但在整理文章的过程中，不可避免的会查阅一些资料，也会不可避免的出现一些引用链接的丢失问题。

在查阅资料的过程中，发现了很多国内外优秀的技术博客和科普文章，他们将自己对于算法的理解，对于行业的洞见，甚至实验代码都放在了网上，免费供人阅读学习，正是由于这些免费的学习资料，才使得学习不再是难事。

如果文中某些内容是您原创，并且本文下方并未给您署名引用，或您希望署名引用，或您希望删除，或其他需求，请联系我。

同时，非常希望并感谢您将自己原创文章、图片、代码等用于学术交流或科普传播，一起为深度学习贡献一份力量。

最后，限于作者白天要上班，文章基本都是晚上下班后和周末抽时间写的，时间精力有限。文中一些表述难免有疏漏，同时，作者为了尽可能用通俗易懂的语言说清楚一些算法原理，会导致某些算法上的表述并不严谨，但不妨碍我们对一些算法有一个感性认识。

如果您有更好的想法，也欢迎联系我，一起学习交流。

版本管理

版本	时间	备注
V0.1	2022/08/13	第一版整理完成，共 18500 字。主要内容： 像素、卷积、残差、池化、全连接、激活、分类器。
欢迎关注公众号：董董灿是个攻城狮，获取最新文章。 		

部分参考资料

1. https://www.zhihu.com/question/401688068/answer/1295651905 作者： James Ken (池化层特征不变性)
2. https://zhuanlan.zhihu.com/p/78760534 作者：G-kdom （池化层）
3. https://medium.com/analytics-vidhya/activation-functions-all-you-need-to-know-355a850d025e 作者：Sukanya Bag （激活函数）
4. 微博@一条 （暮山紫动图）
5. https://www.zhihu.com/question/41037974/answer/150552142,作者:田 star（全连接“蚂蚁开会”）
6. https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1，作者：Irhum Shafkat （卷积动图）