机器学习讲义 (L15-B)：卷积神经网络

授课教师：王贝伦 / 助教：张嘉琦，黄旭，谈笑，徐浩卿

1 深度学习概括

深度学习同一般的机器学习方法一样，包括了以下流程：

- 任务：判别预测、生成、强化学习、推导等
- 输入/输出：即数据表示
- 网络结构设计：深度学习特有的步骤，设计网络拓扑结构、确定网络超参数等
- 学习 (训练)：使用不同的损失函数、优化方法，或使用 GPU 和分布式技术提高训练速度
- 验证 (测试)：利用“软件 2.0”概念选择最佳网络

形象地来说，可以把构建深度学习网络的过程看作“搭积木”，由于输入输出都是矩阵形式，我们很容易将不同的网络层堆叠到一起，并且层数可以任意增长。需要修改网络结构时，只需要去除或增加某些网络层，即可达到不一样的效果。

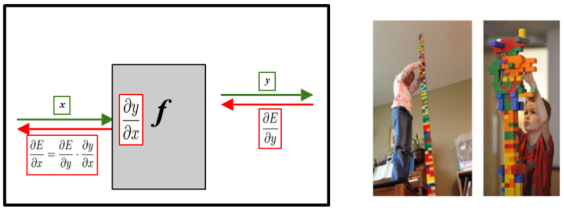

Figure 1: 构建深度神经网络

1.1 软件 2.0

前文提到了软件 2.0 的概念，它实际上是相对于软件 1.0 而言的。软件 1.0 是指传统的程序，如 C++、Java、Python 等高级语言编写的一系列操作指令，最后使计算机完成一些特定的工作。而软件 2.0 是指将深度学习，即深度神经网络，看作一个通用的软件架构。我们的“编程”则是调整网络的结构和超参数，并将输入输出及损失函数的形式规定好，运行时计算机自动进行学习等训练过程，返回一个最佳的参数集合，今后使用软件只需要将输入填入，进行一次前向传播即得到输出结果。

软件 2.0 的编程思维的确有许多优点，如计算简单容易做成集成芯片、性能优秀、易于调整结构等，它也具有一些局限性，如可解释性不高、Debug 工作困难、难以应对特定攻击等等。

2 卷积神经网络 (CNN)

我们将视线移至网络结构设计，本节将介绍一个极其重要的网络块：卷积神经网络 (Convolutional Neural Network)。一个简单的卷积神经网络通常具有卷积层 (Convolutional Layer)、最大池化层 (Maxpooling Layer)、全连接层 (Fully Connected Layer) 等基本组件。如下图所示，对于一个猫狗分类问题，可以将各层如图堆叠构建神经网络。

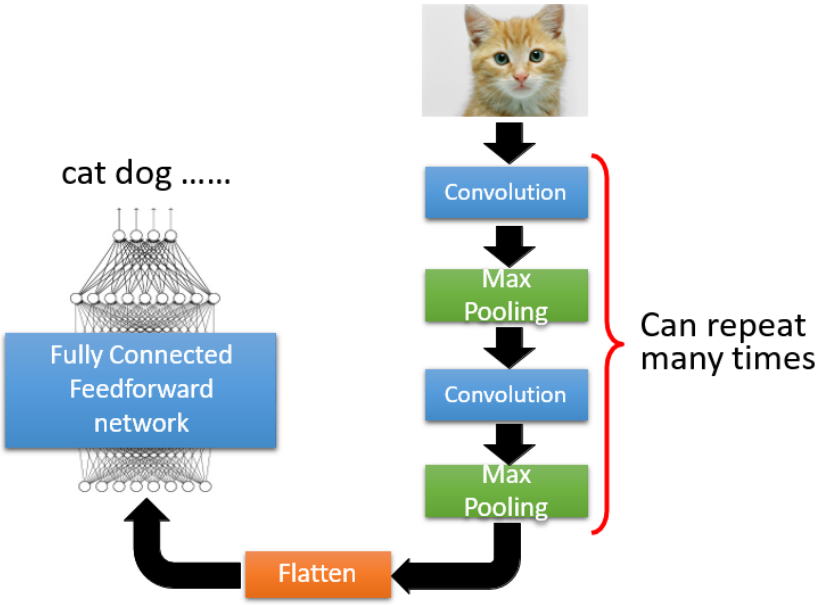

Figure 2: 简单卷积神经网络

2.1 卷积层原理

卷积层是卷积神经网络的核心组件，它利用类似卷积的计算过程构建。对于一个卷积层，假设我们的输入是一个矩阵（在下图的例子中是一个 6×6 的矩阵），我们定义一些过滤器矩阵（也被称为卷积核，图中例子为 2 个 3×3 的矩阵）。其中卷积核的个数、边长为该层的超参数，而卷积核内的元素则是该层的待学习参数。

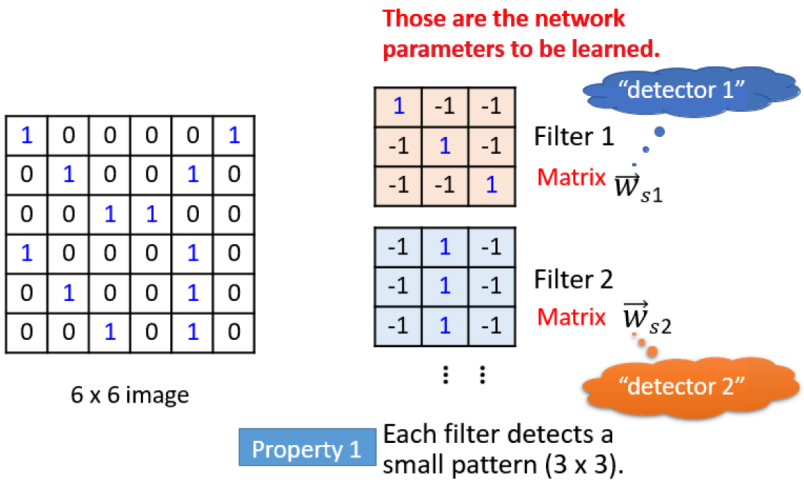

Figure 3: 卷积层示意图

卷积层的计算非常简单，将卷积核放置在输入矩阵的左上角，按元素相乘并将结果相加，得到的就是第一个输出值。之后将卷积核移动一格，得到下一个输出值。

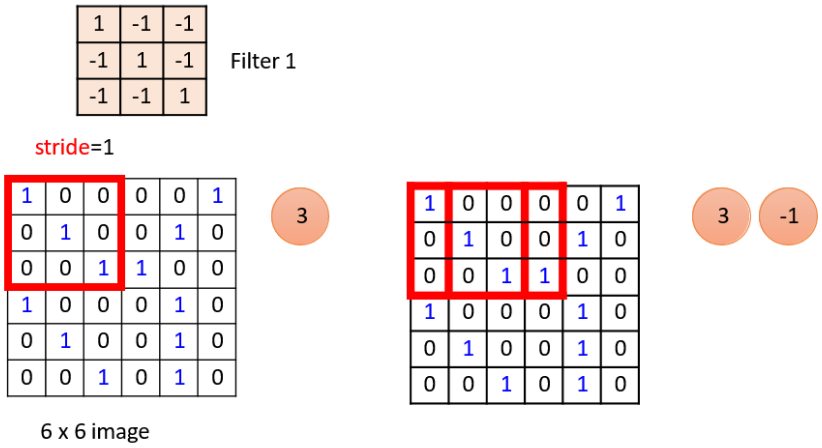

Figure 4: 卷积计算示意图

可以发现，我们移动卷积核不一定要一格一格移动，可以一次移动 2 格，这样会使输出的尺寸缩小一半，当然也可以一次移动多格，这个超参数被称为步幅（stride）。

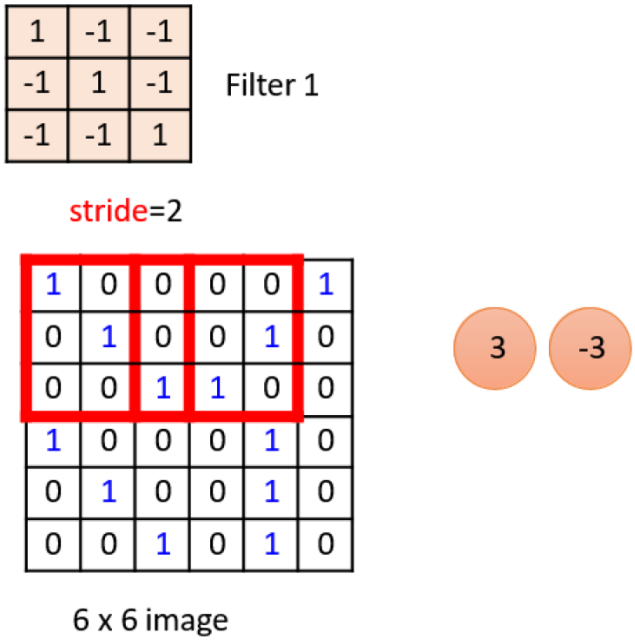

Figure 5: 步幅为 2 时的卷积计算

最后形成一个输出矩阵。对每个卷积核如此操作，我们将得到一系列输出矩阵，或一个输出张量。

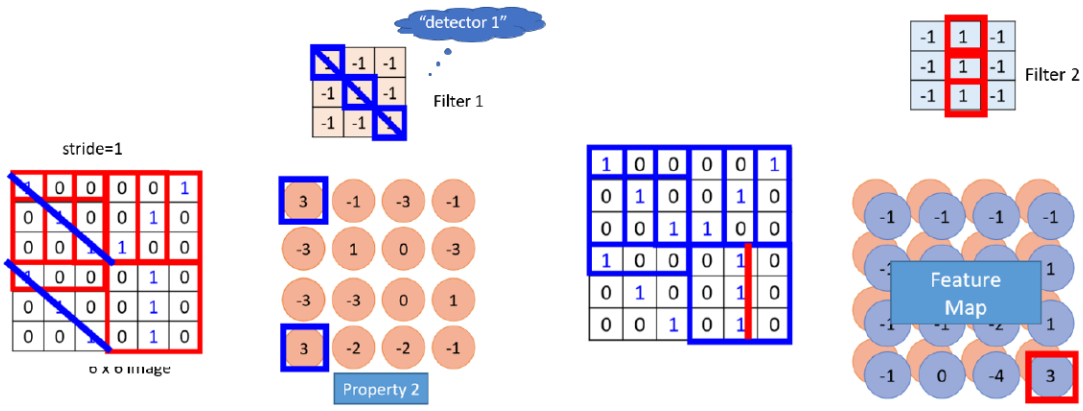

Figure 6: 卷积层输出矩阵示意图

对于输入矩阵是多维数组的情况（如 RGB 输入），我们将卷积核的高度也设置为相同，并进行同样的计算。注意此时对一个卷积核，我们要将 3 层的卷积结果加起来得到一个输出元素，即一个卷积核对应一个输出矩阵。

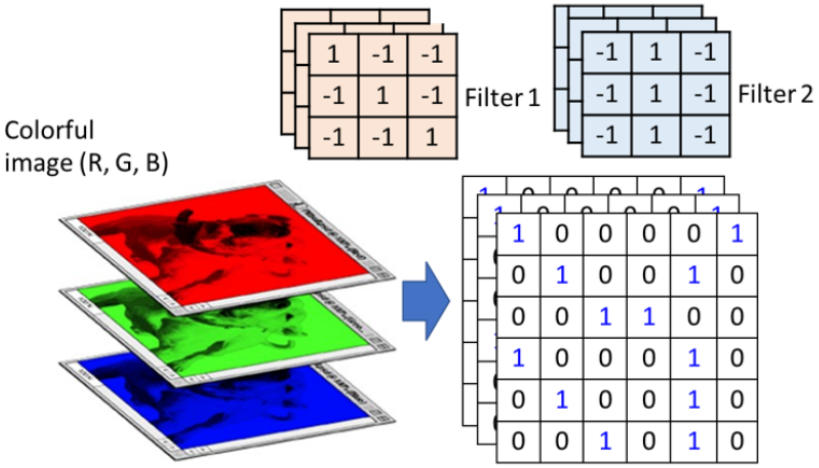

Figure 7: 彩色输入示意图

2.2 其他组件

在一个 CNN 中，通常还有最大池化层、线性整流层、全连接层等组件。

- 线性整流层：这一层将上一层的输入矩阵按元素进行线性整流，即将所有负数值变为 0。这一层一般连接在卷积层之后，我们希望输入输出都是非负值，这是因为现实数据大多数都是非负值，并且非负值易于学习和展示。

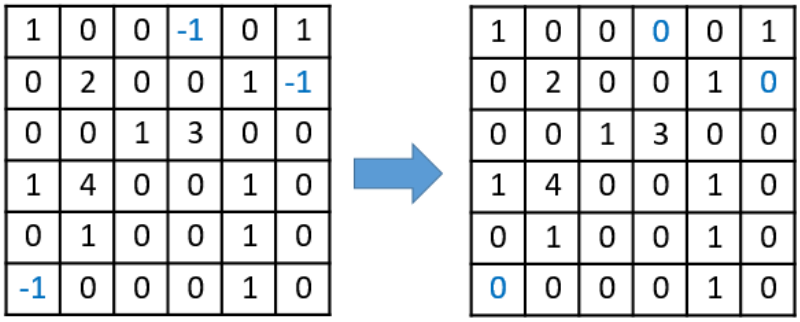

Figure 8: 线性整流层示意图

- 最大池化层：这一层将上一层的输入矩阵切割成等大的小块，取每块中的最大元素作为输出。这层的目的是将卷积层得到的高维张量降维，同时尽量保留特征信息。

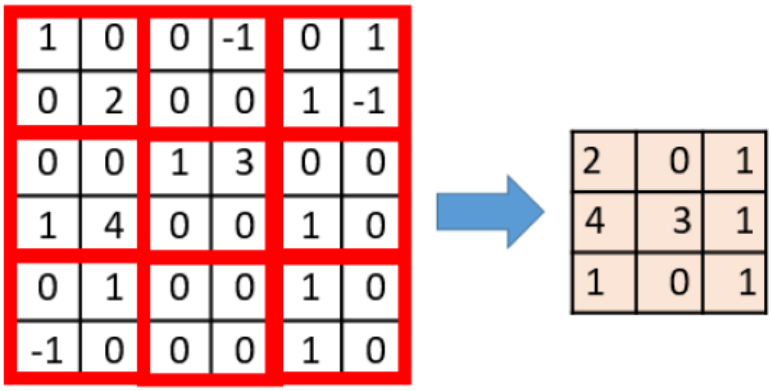

Figure 9: 最大池化层示意图

- 全连接层：这一层是一个全连接神经网络，其结构与之前章节中的一样，此处不再赘述。注意它的输入是将上一层的输出展开为向量得到的。这一层通常是 CNN 的最后一层，它的输入是由最后的卷积层提取出的特征，输出是预测值或者分类分数等标签，事实上是一个逻辑回归。

2.3 特点

CNN 具有一些非常优秀的性质，让它在各种问题（尤其是图像处理工作方面）上表现很好。

- 局部化：卷积核通常提取的是图像的局部特征，而非全局特征。这个特点是由于卷积核自身的大小相对于输入矩阵来说较小，卷积结果只与输入矩阵的局部元素值有关。

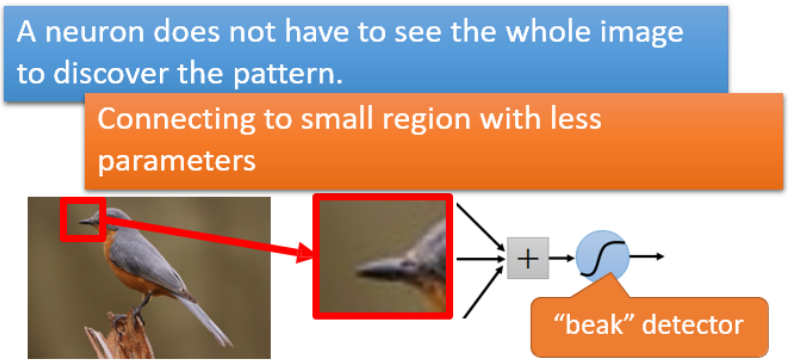

Figure 10: 局部化特点

- 平移不变性：将图像平移，卷积神经网络的输出不会变化太多。注意这个特点与局部化的区别：这个特点是由于卷积核提取了局部特征，而这个特征在矩阵的哪个位置不太重要。

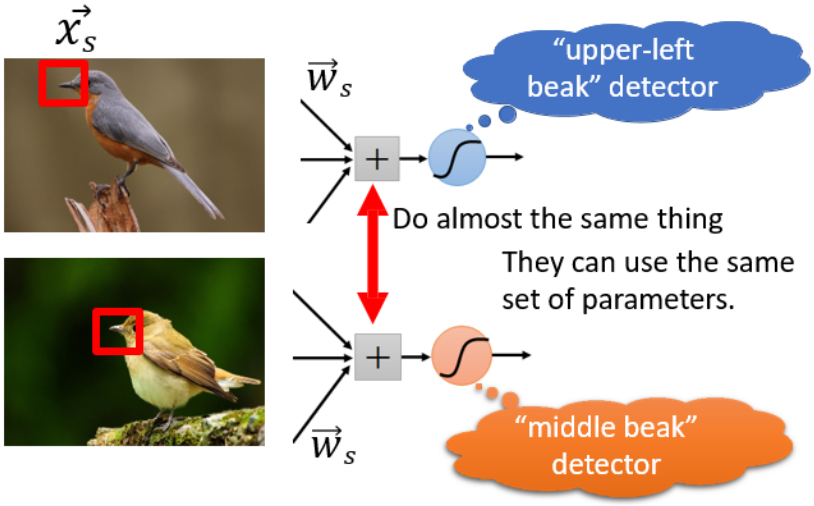

Figure 11: 平移不变特点

- 缩放：将输入缩小一定比例，同时相应改变卷积核的大小，其效果不会变化太多。这是因为 CNN 提取的特征与边界密切相关，缩放图像之后，边界不会变化太多。

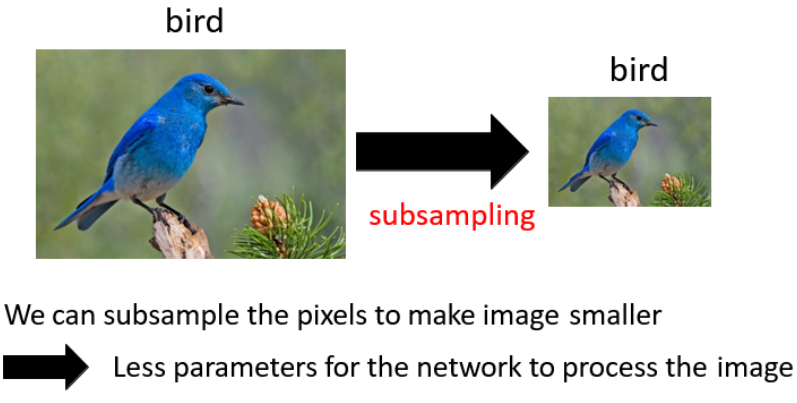

Figure 12: 缩放特点

- 可推广性：对于类似图像的输入，CNN 也可以类似地处理它们：如视频、音频、文本等。

3 卷积网络 VS 全连接网络

全连接网络，即这一层的神经元与上一层的每个神经元都连接。而卷积网络的神经元只与上一层的部分神经元连接，连接个数取决于过滤器（卷积核）的大小。如图13所示，卷积层的每个神经元仅有上层的 9 个神经元相连，因为过滤器大小为 3×3，而全连接层的神经元需要与上层所有的 36 个神经元连接。

全连接网络与卷积网络相比，主要劣势在于参数过多。例如对于一个 6×6 的图像，卷积网络用两个 3×3 的过滤器的话，一共有 $3 \times 3 \times 2 = 18$ 个参数，而同样是两个过滤器，全连接网络需要 $36 \times 2 = 72$ 个参数。当图像较大时，两者的参数个数会有巨大的差距，这使得训练全连接网络几乎不可能。下面从三个方面来说明卷积网络参数较少的原因。

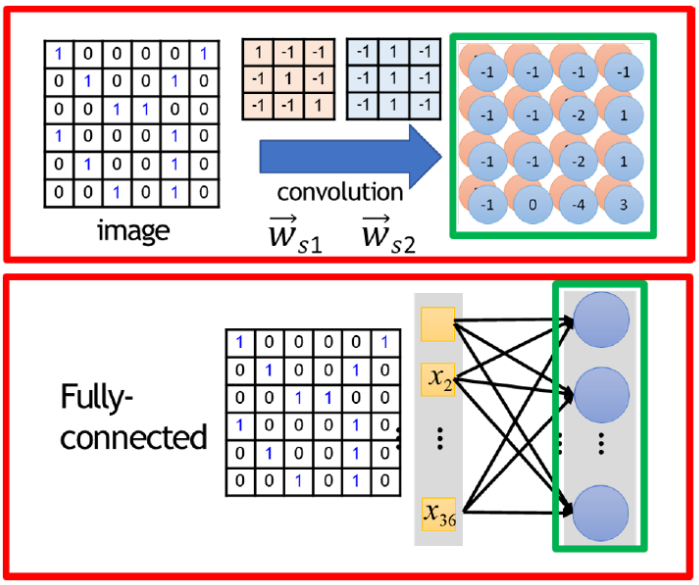

Figure 13: 卷积网络与全连接网络的对比图

局部性 卷积网络的每个神经元不与上一层的所有神经元相连，而是只和一小部分神经元相连，这样可以减少一部分参数。如图14所示，对于一个 3×3 的过滤器，只与上一层 9 个输入元连接，也就只有 9 个参数。

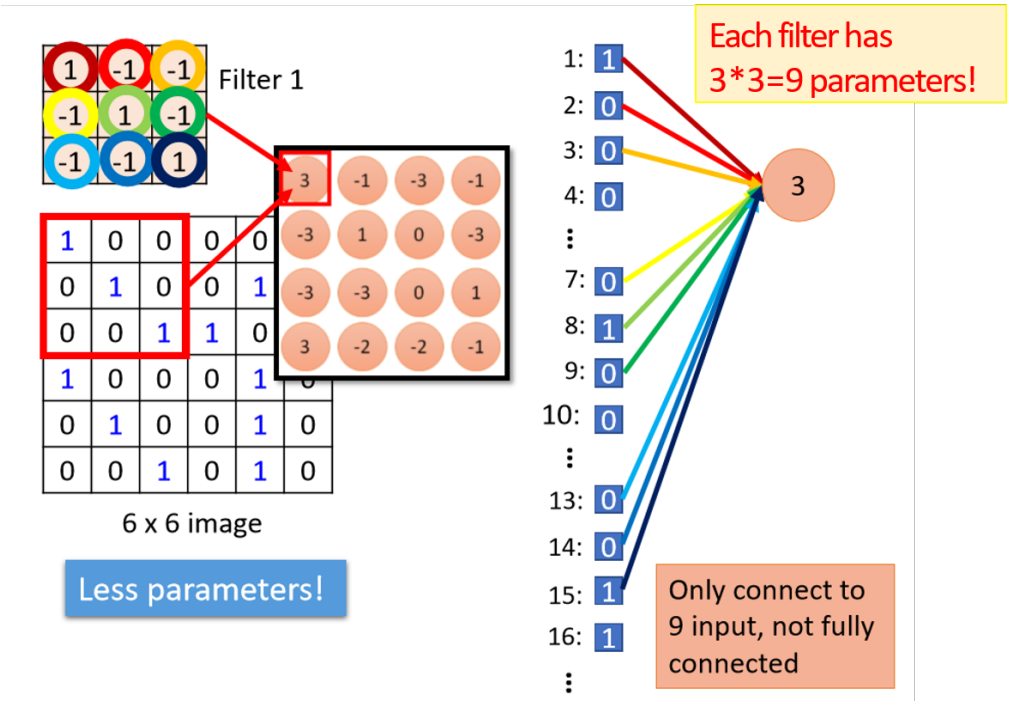

Figure 14: 卷积网络的局部性

平移无关性 平移无关性意思就是参数共享。一组连接共享同一组参数，这样又可以减少许多参数。还是如图14，一个特征图的所有值都是由同一个过滤器计算得出，即使用了同一组参数。

下采样 卷积神经网络中通常使用池化来实现下采样，最常用的是最大池化。它将特征图划分为若干个矩形区域，在每个区域取最大值。如图15所示，通过池化，使特征图缩小，相当于生成了一个新的更小的图片，进一步减少了后续操作的参数个数。

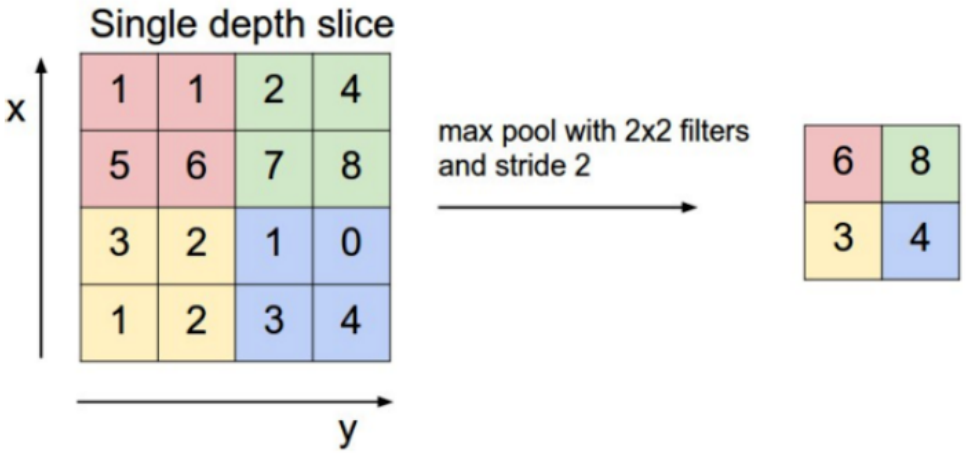

Figure 15: 最大池化实现下采样示意图

4 降维（Flatten）

图像经过一系列卷积池化等操作后，得到若干个提取特征后的小“图像”，降维层将这些图像降成一维，作为后续全连接层的输入。如图16所示，降维层通常作为到全连接层的过渡，后面也可以是一个深度前向传播神经网络。

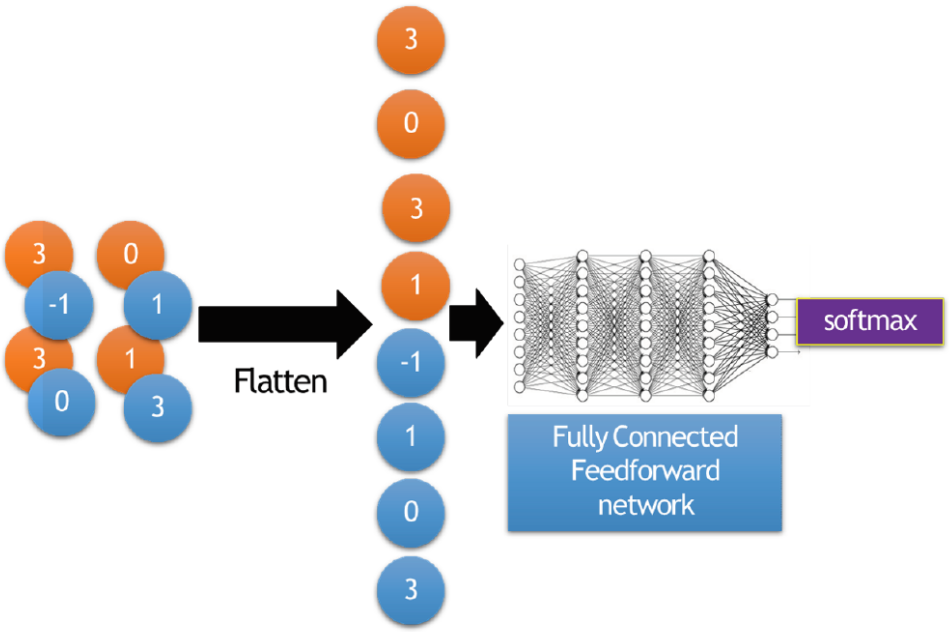

Figure 16: 降维层示意图

5 卷积网络的应用

卷积网络在生活中有许多应用：

- 围棋（AlphaGo）
  围棋的棋盘可以用矩阵表示出来，1 表示放黑子，-1 表示放白子，0 表示没有放棋。将这个矩阵输入已训练好的卷积神经网络，可以输出下一步棋的位置。
- 语音识别（CLDNN）
  使用卷积神经网络分析语音的时频图，过滤器仅在频率方向进行移动。

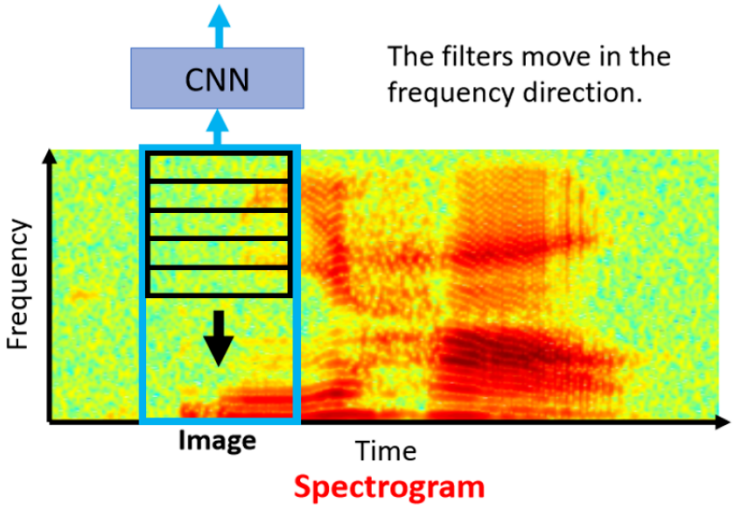

Figure 17: 卷积网络在语音识别中的应用

- 图像分类（ResNet）
  预测给定图像的类别
- 图像检索（AlexNet）
  立足于给出图像的内容，搜索内容相似的图像
- 对象检测（Faster R-CNN）
  检测图像中的物体，并提取每个对象，如图18所示。

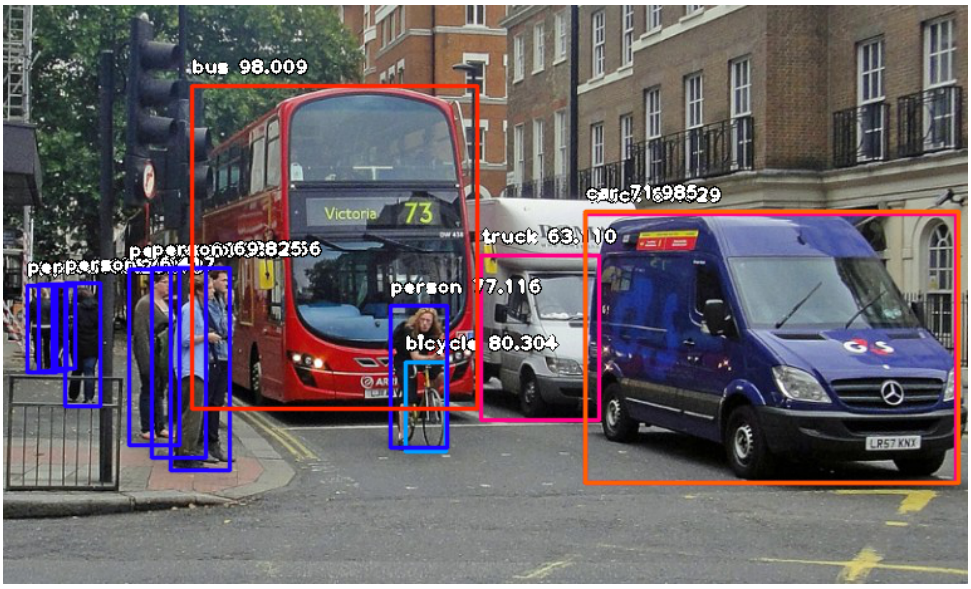

Figure 18: 对象检测

6 残差网络

首先思考一个问题：网络的层数是越多越好吗？理论上，深层网络的效果不应该比浅层网络的差。但是实际上，随着网络层数的增加，网络发生了退化的现象。一开始网络层数增加，在训练集上的误差会逐渐减小，然后趋于饱和，当网络层数继续增加时，训练误差反而会变大。这是由于非线性激活函数 ReLU 的存在，使得输入到输出的过程几乎是不可逆的，这也造成了不可逆的信息损失。如果我们将深层网络比浅层网络多出来的层变成恒等映射（identity mapping）的话，就能让深层网络至少达到浅层网络的效果。这就是残差网络想要解决的问题。通过引入残差块，如图19，在激活函数前，将上一层（或上几层）的输出与本层的输出相加，再将和输入到激活函数作为本层的输出。

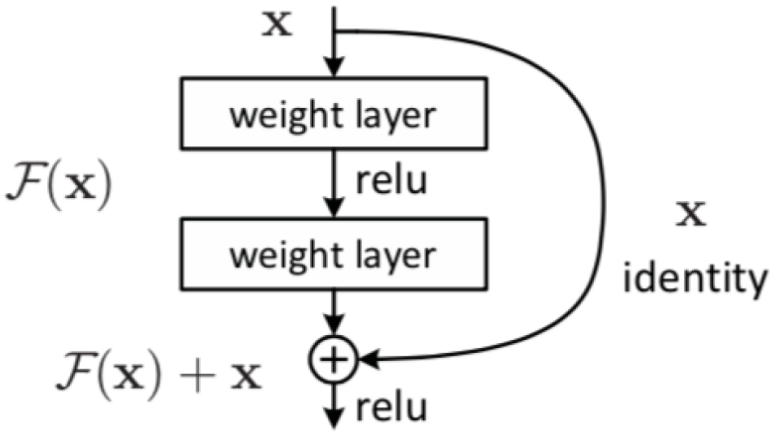

Figure 19: 残差块示意图

使用残差技巧后，优化深层网络也会变得更加容易。著名的残差神经网络 ResNet 模型正是基于残差网络，网络层数多达 152 层，取得了很好的效果。

编程实现

使用 tensorflow.keras 包实现卷积神经网络，数据集为 fashion_mnist，包含 60000 个训练数据和 10000 个测试数据。每张图片内容为一件服装，大小为 28×28，一共 10 个类别。最终测试准确率为 89%

```
import tensorflow as tf
from tensorflow import keras
import numpy as np

# 导入fashion_mnist数据集
# train_images.shape=(60000, 28, 28), test_images.shape=(10000, 28, 28)
fashion_mnist = keras.datasets.fashion_mnist
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()

# 将存储训练和测试图片的数组变成四维，以满足卷积层的输入要求
train_images = train_images.reshape((60000, 28, 28, 1))
test_images = test_images.reshape((10000,28, 28, 1))

# 将图片的数值范围变成[0, 1]
train_images = train_images / 255.0
test_images = test_images / 255.0

# 建立神经网络模型
model = keras.models.Sequential()
# 卷积层，32个过滤器，卷积核大小为5*5，激活函数为relu
model.add(keras.layers.Conv2D(32, (5, 5), activation='relu', input_shape=(28, 28, 1)))
# 池化层，边长为2，步长为2
model.add(keras.layers.MaxPooling2D(2, 2))
# 卷积层，64个过滤器，卷积核大小为3*3，激活函数为relu
model.add(keras.layers.Conv2D(64, (3, 3), activation='relu'))
# 池化层，边长为2，步长为2
model.add(keras.layers.MaxPooling2D(2, 2))
# 降维
model.add(keras.layers.Flatten())
# 全连接层，128个节点，激活函数为relu
model.add(keras.layers.Dense(128, activation='relu'))
# 全连接层，10个节点，激活函数为softmax
model.add(keras.layers.Dense(10, activation='softmax'))
# 查看模型各层的输出大小与参数个数
model.summary()

# 编译模型，使用adam优化器，损失函数为交叉熵
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# 训练模型，每批样本个数为200，共训练5轮
model.fit(train_images, train_labels, batch_size=200, epochs=5)

# 测试
```

```
test_loss, test_acc = model.evaluate(test_images, test_labels, verbose=2)
print('\nTest accuracy:', test_acc)
```

引用

[1]https://imageai.readthedocs.io/en/latest/detection/index.html

[2]https://tensorflow.google.cn/tutorials/keras/classification