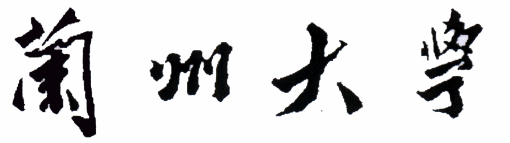
****

卷积神经网络实现垃圾分类

**课程:** 机器学习

**班级:** 2018级四班

**姓名:**

**学号:** 320180939790

## **日期**: 2021/07/05

目录

**[一、 卷积神经网络简介](#_Toc1258825542_WPSOffice_Level1)** **[1](#_Toc1258825542_WPSOffice_Level1)**

**[二、 卷积神经网络结构](#_Toc2144178165_WPSOffice_Level1)** **[1](#_Toc2144178165_WPSOffice_Level1)**

[1. 卷积层](#_Toc2144178165_WPSOffice_Level2) [2](#_Toc2144178165_WPSOffice_Level2)

[2. 池化层](#_Toc1848909174_WPSOffice_Level2) [3](#_Toc1848909174_WPSOffice_Level2)

[3. 全连接层](#_Toc1295912651_WPSOffice_Level2) [4](#_Toc1295912651_WPSOffice_Level2)

**[三、 垃圾分类模型](#_Toc1848909174_WPSOffice_Level1)** **[5](#_Toc1848909174_WPSOffice_Level1)**

**[四、 参考文献](#_Toc1295912651_WPSOffice_Level1)** **[11](#_Toc1295912651_WPSOffice_Level1)**

# 卷积神经网络简介

卷积神经网络与传统神经网络都是由对权重和偏置进行学习的神经元构成，神经元接收输入，执行点积并选择一个非线性激活函数。输入为原始像素图片，输出类别概率。但与传统神经网络不同的是，卷积神经网络假设输入的都是图片，卷积网络各层的神经元具有三个维度，即宽度、高度、深度，使用参数共享机制大大减少网络中的参数数量。这样，使用较少的参数，训练出更高效的模型，还可以有效的避免过拟合。

# 卷积神经网络结构

一个卷积神经网络由一系列层组成，每一层的三维输入特征体通过一个可微函数变换为经过神经元激活的三维输出特征体。用于构造卷积网络的三类层：卷积层、池化层和全连接层。

输入层输入原始像素值的图片，假设为16\*16\*3；

卷积层计算神经元和与神经元局部连接的输入层的点积，深度变为卷积核的个数，假设卷积核有32个，将得到16\*16\*32的特征体；

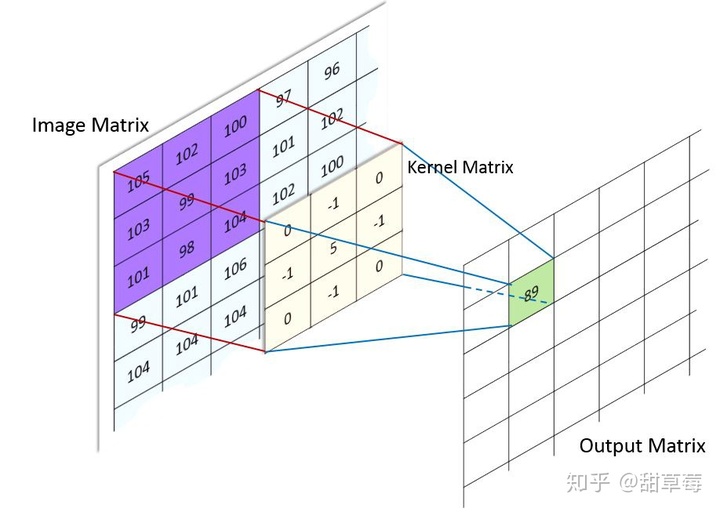
ReLu层不会改变特征体的大小。卷积之后，通常会加入偏置, 并引入非线性激活函数；

池化层对特征体的高度和宽度进行降采样操作，适当减少参数数量，避免国拟合，即得到尺寸为8×8×32的特征体；

全连接层计算类别概率，得到1\*1\*10（类别个数）的特征体，与普通神经网络相同，神经元与前面特征体全连接。

## 卷积层

卷积层是卷积神经网络的核心，执行大量的计算。卷积层的参数由一组可以学习的卷积核组成。卷积核是一个二维区域，正向传播时，卷积核沿着输入特征体的宽度和高度进行卷积，计算卷积核矩阵和卷积核对应位置特征体的输入点积。沿着宽度和高度计算完后，得到二维激活映射图。这使得卷积核看到某种类型的视觉特征时就会被激活。例如下图：105\*0+102\*（-1）+100\*0+103\*（-1）+99\*5+103\*（-1）+101\*0+98\*（-1）+104\*0 = 89。



（图片来自知乎）

从上图可以看出每个神经元只与输入特征体的部分相连接，特别的，沿着宽度和高度是局部连接，而沿着深度方向却是全连接。

卷积层有四个超参数，分别是卷积核的尺寸和数量、步长和零填充。卷积核的个数即深度，每个卷积核都可以学习输出中的不同特征。卷积核沿着宽度和高度滑动的单位距离为步长。用0填充输入图片的边界四周为零填充，可以用它来精确地保留输入特征体的二维面的尺寸。

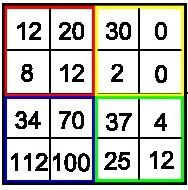
根据输入图片尺寸、卷积核大小以及步长可以计算输出大小：假设输入大小为I，卷积核大小为K，步长为S，零填充为P（填充0，P=1；反之为0），输出尺寸为（I - F +2P）/S + 1。

卷积层另一个重要特征是参数共享。参数共享用于控制参数数量。如果使用全连接，仅在卷积网络一层上就可能出现数量庞大的参数。所以假设特征不仅可以在（a,b）位置上用于计算，还可以用于位置（c,d）上的计算，这样就产生了深度，限制每个深度切片中的神经元具有相同的权重和偏置。这样就大大减少了参数的数量，同时有效避免了过拟合。

## 池化层

池化是一种降采样操作，用来减少特征空间、减少网络中的参数数量以及降低计算量，且因此也可以控制过拟合。

常见的池化方式有最大值池化、平均值池化和L2-正则池化。

（图片来自知乎）

|  |  |
| --- | --- |
| 20 | 30 |
| 112 | 37 |

* 最大值池化保留四个像素点中的最大值，得到

|  |  |
| --- | --- |
| 13 | 8 |
| 79 | 20 |

* 平均池化取四个像素点中的平均值，得到

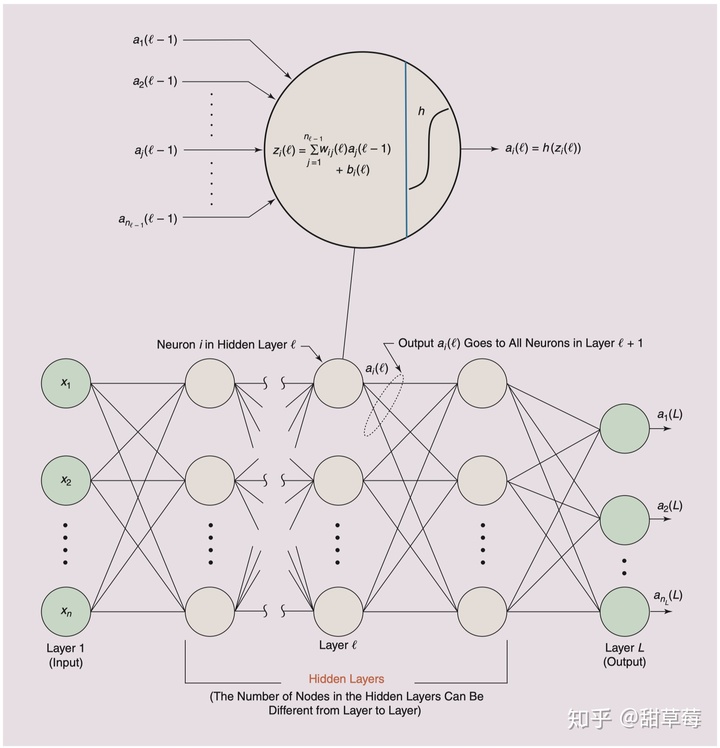
|  |  |
| --- | --- |
| 78948 | 1800 |
| 2.4\* | 105786 |

* L2池化保留均方值

池化层在输入特征体的每个深度切片上独立运行，调整其二维面的尺寸。最常见的池化层的形式具有大小为2×2的过滤器，在输入特征体的每个深度切片上分别沿着宽度和高度执行以2为步长的降采样操作，以丢弃75%的特征值。

## 全连接层

全连接层中的神经元与前一层的全部特征值完全连接。因此其输出特征可以通过矩阵乘法和偏置来进行计算。



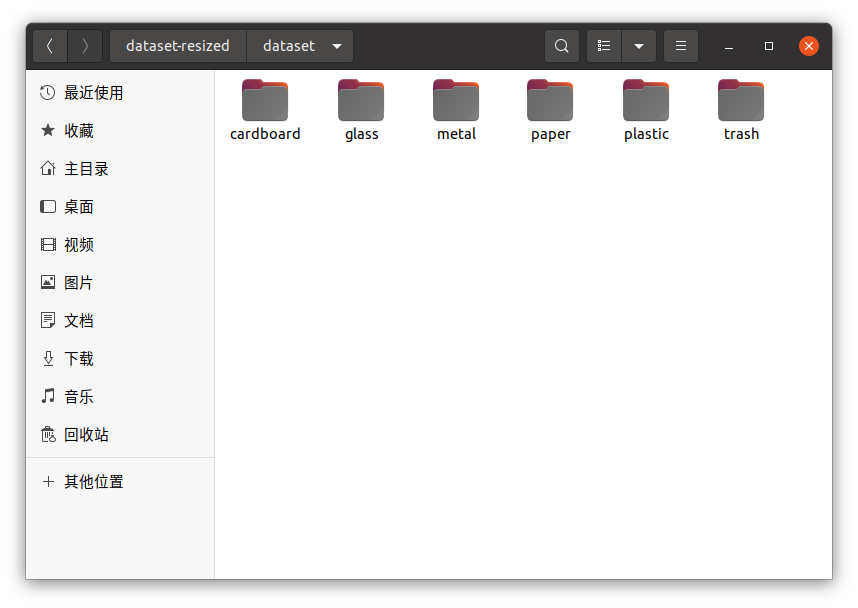
（图片来自知乎）

# 垃圾分类模型

我使用CNN进行垃圾的分类。

1. 读取数据。

原始数据为jpg图片，每张图片放入了不同的代表垃圾种类的文件夹中。使用glob和os读取数据。



【代码】

filepath = "dataset"

img = glob.glob(os.path.join(filepath, '\*/\*.jpg'))

1. 由于原始数据量较少，使用ImageDataGenerator进行图像增广。

【代码】

train\_datagen = ImageDataGenerator(

rotation\_range=45,#数据提升时图片随机转动的角度

rescale=1./255,#重放缩因子

width\_shift\_range=0.1,#图片宽度的某个比例，数据提升时图片水平偏移的幅度

height\_shift\_range=0.1,#图片高度的某个比例，数据提升时图片竖直偏移的幅度

shear\_range=0.1,#剪切强度（逆时针方向的剪切变换角度）

zoom\_range=0.1,#随机缩放的幅度

horizontal\_flip=True,#进行随机水平翻转

vertical\_flip = True,#进行随机竖直翻转

fill\_mode='constant',#当进行变换时超出边界的点将根据本参数给定的方法进行处理

validation\_split=0.1)

test\_datagen = ImageDataGenerator(

rescale=1./255,

validation\_split=0.1)

1. 训练，通过flow\_from\_directory方法。规定图片尺寸为（300，300），一批数据大小为16，"categorical" 返回2D one-hot 编码标签。

【代码】

train = train\_datagen.flow\_from\_directory(

filepath, target\_size=(300, 300), batch\_size=16,

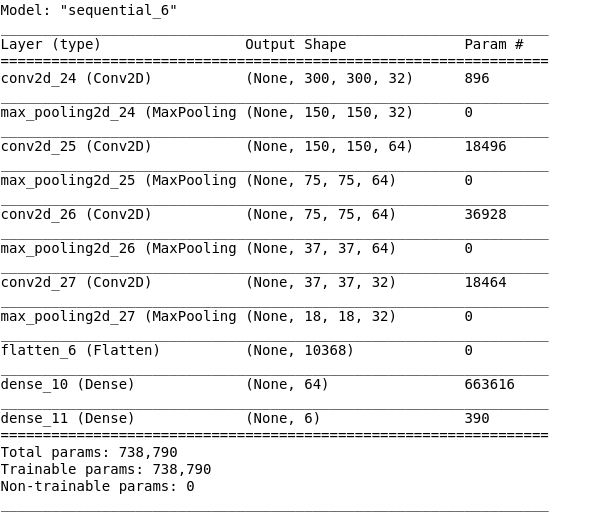
class\_mode='categorical', subset='training', seed=1)

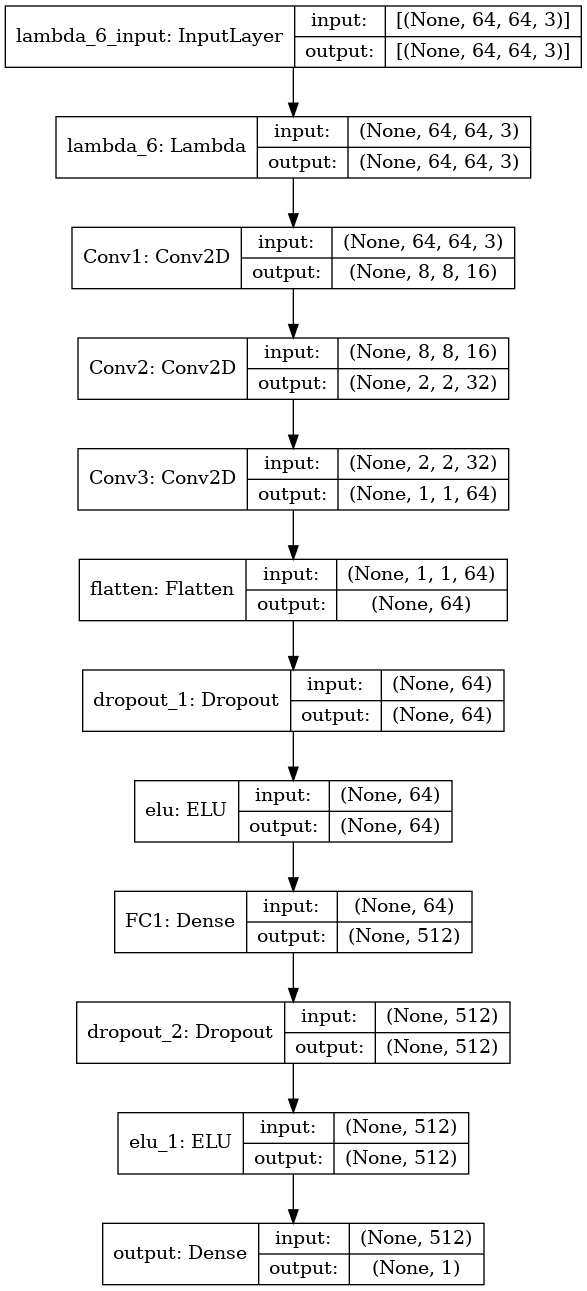
validation = test\_datagen.flow\_from\_directory(

filepath, target\_size=(300, 300), batch\_size=16,

class\_mode='categorical', subset='validation', seed=1)

1. 构建模型。
2. 使用32个过滤器，卷积窗口为3\*3，使用零填充，激活函数为relu，输入为300\*300\*3经过计算输出尺寸为300\*300\*32。
3. 池化。池化窗口为2\*2。输入为300\*300\*32经过计算输出尺寸为150\*150\*32。
4. 卷积。使用64个过滤器，卷积窗口为3\*3，使用零填充，激活函数为relu，输入为150\*150\*32经过计算输出尺寸为150\*150\*64。
5. 池化。池化窗口为2\*2。输入为150\*150\*64经过计算输出尺寸为75\*75\*64。
6. 使用64个过滤器，卷积窗口为3\*3，使用零填充，激活函数为relu，输入为75\*75\*64经过计算输出尺寸为75\*75\*64。
7. 池化。池化窗口为2\*2。输入为75\*75\*64经过计算输出尺寸为37\*37\*64。
8. 使用32个过滤器，卷积窗口为3\*3，使用零填充，激活函数为relu，输入为37\*37\*64经过计算输出尺寸为37\*37\*32。
9. 池化。池化窗口为2\*2。输入为37\*37\*32经过计算输出尺寸为18\*18\*32。
10. 扁平层的作用将张量转换为相同数量 1 维向量18\*18\*32 = 10368
11. 分类器采用两个全连接器。最后一层全联接层也是输出层，采用了 softmax 作为激活层，因此，输出长度为6的向量，对应6个分类的预测概率值。





【代码】

model = Sequential()

model.add(Conv2D(filters=32, kernel\_size=3, padding='same', activation='relu', input\_shape=(300, 300, 3)))

model.add(MaxPooling2D(pool\_size=2))

model.add(Conv2D(filters=64, kernel\_size=3, padding='same', activation='relu'))

model.add(MaxPooling2D(pool\_size=2))

model.add(Conv2D(filters=64, kernel\_size=3, padding='same', activation='relu'))

model.add(MaxPooling2D(pool\_size=2))

model.add(Conv2D(filters=32, kernel\_size=3, padding='same', activation='relu'))

model.add(MaxPooling2D(pool\_size=2))

model.add(Flatten())

model.add(Dense(64, activation='relu'))

model.add(Dense(6, activation='softmax'))

1. 训练时用adma优化器、categorical\_crossentropy损失函数和准确率评测标准为acc。

【代码】

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['acc'])

1. 使用 Python 生成器（或 Sequence 实例）逐批生成的数据，按批次训练模型。

【代码】

model.fit\_generator(train, epochs=100, steps\_per\_epoch=2276//32,validation\_data=validation,

validation\_steps=251//32)

1. 全部代码和数据集我放到了github仓库上，详细

# 参考文献

* 斯坦福CS231n课程
* 知乎（https://www.zhihu.com/topic/20043586/intro）