学习目的

深度学习指标

（1）卷积神经网络

（2）实现猫狗识别

（3）使用api实现人面识别与人面对比

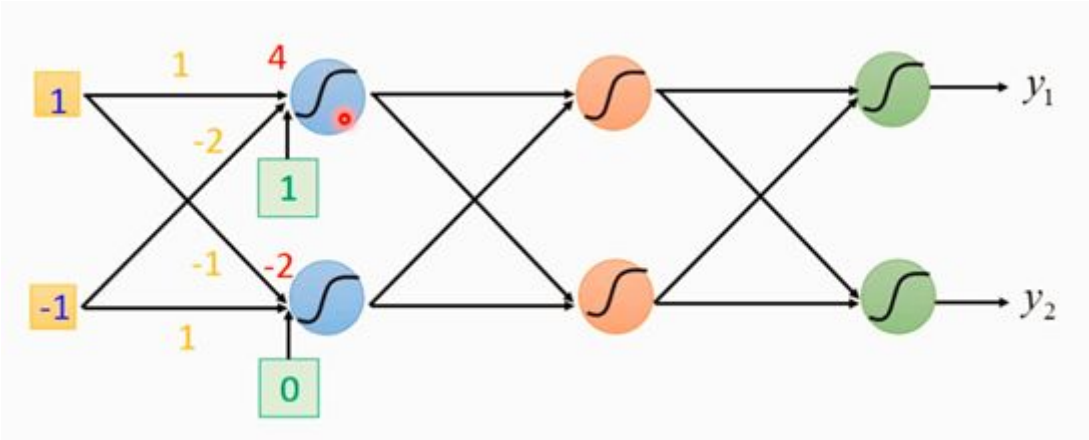
卷积神经网络

卷积神经网络是应用最为广泛的神经网络之一。虽然主战场是图像识别，但是在自然语言处理和语音识别等领域也有着不俗的表现。而在图像识别中，在一些测试中，其表现甚至超越人类。现在我们一起快速了解一下神奇的卷积神经网络。

卷积神经网络分成三个主要部分，分别为：卷积层（convolution layer），池化层（pooling layer）和 全连接层 （fully connected layer）。

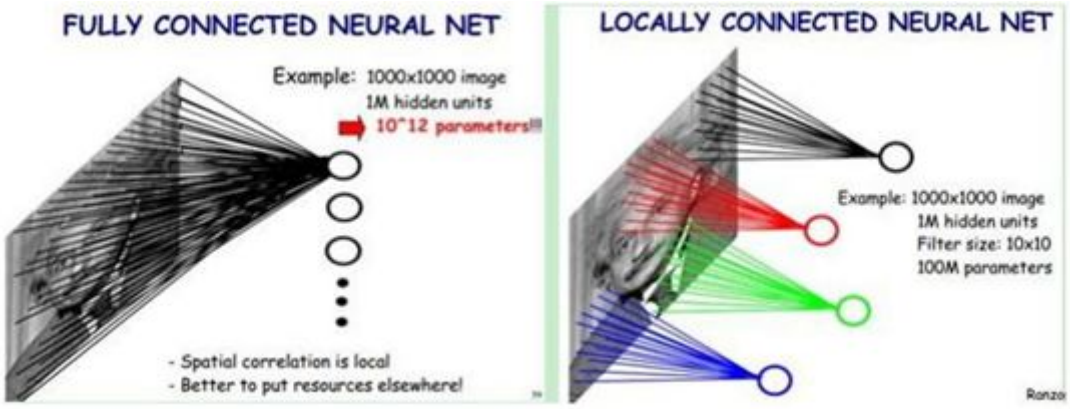
全连接层：

全连接层简单的说就是将层间所有神经元两两连接在一起。如下图所示。每一个神经元代表每一层的输入（蓝色数字）和输出(红色数字)。两者之间的线上的数字(黄色数字)，代表权重。最后还要加上偏差（绿色数字）。运算法则是所有输出乘以权重，然后再加上偏差，得到输出。例如第一行的输出4，是由 (1 X 1+ -1 X-2) + 1得到。通过这样一层一层（一列一列）的迭代，最后我们就有了整个全连接层的输出y1, y2。



卷积层：

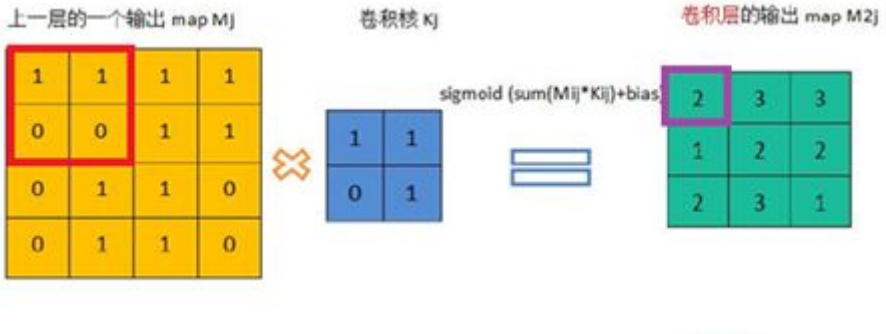
卷积层的思想是，与其像全连接层的每一个输出神经元连接每一个输入，对于输入较多的情况，参数过多效率低下。我们选择每一个输出神经元只链接一个区域的输入，如下图所示。

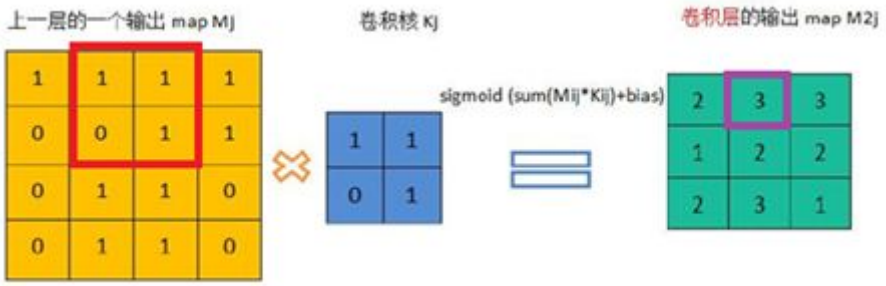


不仅如此，在每个区域中权重（下图蓝色框内数字）的大小是相等的。

具体运算过程如下。输入（红色框内）与权重对位相乘法1 X 1 + 1 X 1 + 0 X 0 + 0 X 1 = 2，得到第一个输出值2（紫色框）。

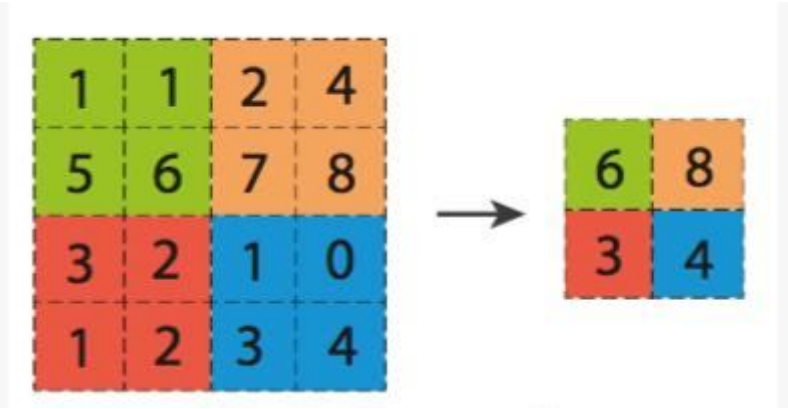
接下来，我们将输入向左移动一格，继续作以上运算 1 X 1 + 1 x 1 + 0 X 0 + 1 X 1 = 3，得到第二个输出值3。当向右移动到达某一排末尾时。向下移动一格到达下一排，并且向左回到排的首列，再继续下一次运算。按照以上规则，遍历整个输入。得到下图中的输出图。





池化层：

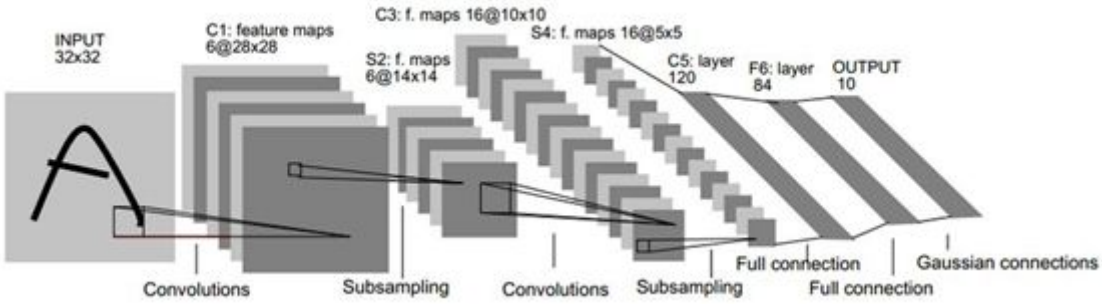
池化层的存在主要目的是除去冗杂信息，增加运算效率。运算法则很简单，直接将每四个相邻输入里面最大的值保留，而删去其他值。如下图所示，在绿，棕，红，蓝四个2X2的区域中，我们只选择其中最大的值保留，得到输出结果。



总览：

一个完整的卷积神经网络由以上三个主要板块构成。下图是一个识别手写英文字母的卷积神经网络。是由 卷积层 + 池化层 + 卷积层 + 池化层 + 全连接层 这样的结构组成。

以此为例，我们的输入是由像素点转化成的数字（比如灰度），如果是32 X 32 大小的图，那就有1024 个的输入，最后有26个输出，26个数字，对应被识别为每一个英文字母的概率。而概率最大的，即是识别结果。



应用于猫狗识别

（1）数据集准备
8000份train_data
2000份test_data

（2）链接选择

①池化层+最大池

②池化层+最大池

③池化层+均值池

④池化层+最大池

⑤池化层+均值池

⑥展平层

⑦Dropout=0.5（防止拟合）

⑧全连接层（relu+sigmoid)

（3）输出结果

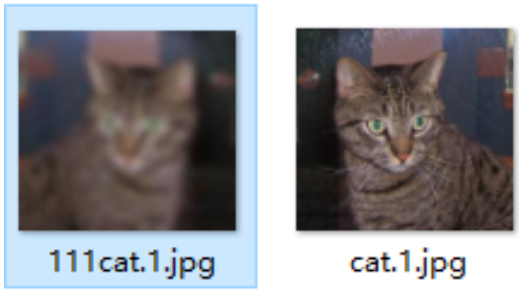
```
- loss: 0.2073 - accuracy: 0.9146 - val_loss: 0.2988 - val_accuracy: 0.8905

- loss: 0.1872 - accuracy: 0.9216 - val_loss: 0.3139 - val_accuracy: 0.8640

- loss: 0.1606 - accuracy: 0.9324 - val_loss: 0.3082 - val_accuracy: 0.8965
```

可见数据集太少依旧导致了精确度不高

之后我利用PYTHON批处理train集，全部镜像加滤镜之后，train变成16000份，val_accuracy达到了略微客观的0.92。



（4）总结

本次使用的是，tensorflow2.20里面自带的keras框架，主要是构造模型与预处理数据集需要花费心思，总得结果我还不是很满意，成果不是很显著，我会继续改进。

基于FACE++的人面对比与人面识别

人面识别

（1）实验数据

input:

output:

```
共检测到4张人脸
第1个人的数据
性别Female
年龄25
微笑率:27.761%

第2个人的数据
性别Female
年龄24
微笑率:85.287%

第3个人的数据
性别Female
年龄34
微笑率:100.0%

第4个人的数据
性别Male
年龄22
微笑率:99.719%


1号人物在图片里的像素位置
top: 345, left: 793, width: 337, height: 337
2号人物在图片里的像素位置
top: 544, left: 412, width: 267, height: 267
3号人物在图片里的像素位置
top: 695, left: 75, width: 246, height: 246
4号人物在图片里的像素位置
top: 339, left: 226, width: 133, height: 133
4个人可以完整识别
```

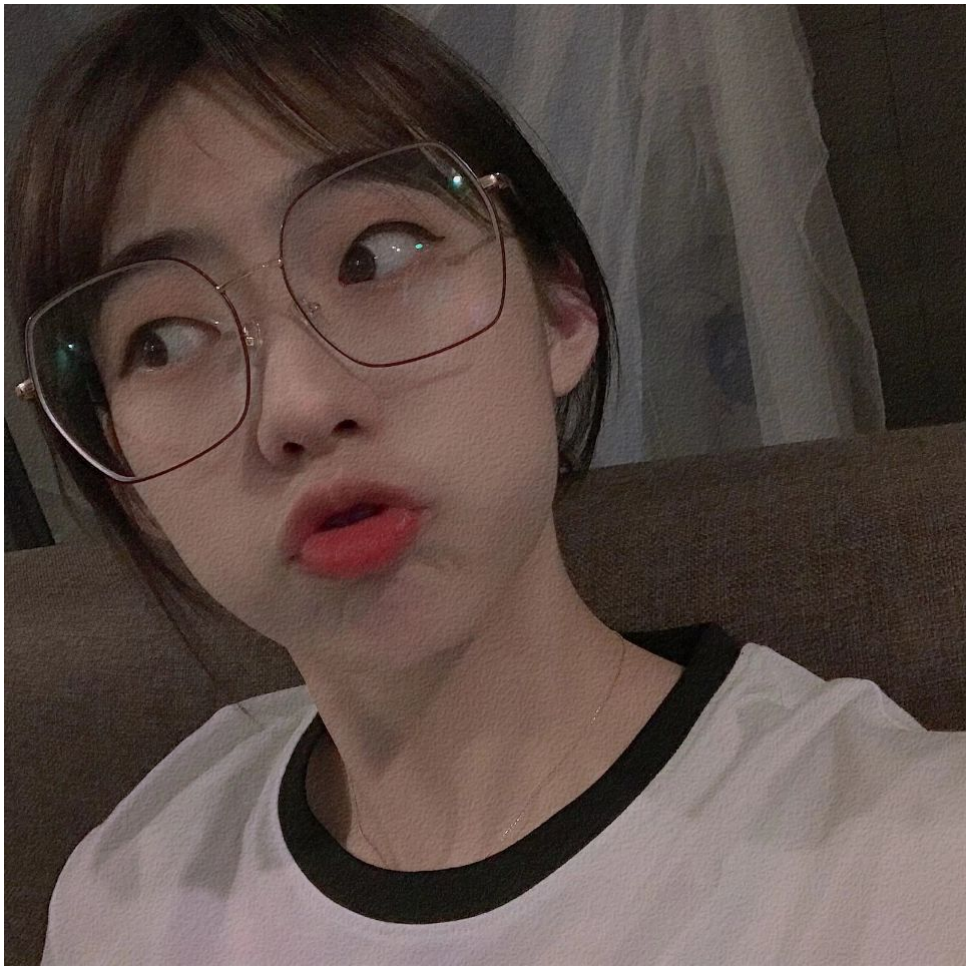
（2）结论

由此可见，此api自动识别所有超过一定图片大小的人脸，输出的width==height，说明识别是使用正方形进行框定人脸，准确度较高。

人面对比

（1）实验数据

input1：



input2（1）：

output:

```
图一中的第1个脸的位置{'width': 500, 'top': 170, 'left': 90, 'height': 500}
图二中的第1个脸的位置{'width': 465, 'top': 185, 'left': 226, 'height': 465}
两张图片的相似度为86.296%
```

input2（2）：

```
图一中的第1个脸的位置{'width': 500, 'top': 170, 'left': 90, 'height': 500}
图二中的第1个脸的位置{'width': 337, 'top': 345, 'left': 793, 'height': 337}
图二中的第2个脸的位置{'width': 267, 'top': 544, 'left': 412, 'height': 267}
图二中的第3个脸的位置{'width': 246, 'top': 695, 'left': 75, 'height': 246}
图二中的第4个脸的位置{'width': 133, 'top': 339, 'left': 226, 'height': 133}
两张图片的相似度为86.698%
```

（2）结论

我遍历过返回值里面的所用属性，最后得出了结论：如果有多人，自动返回的相似度为最高值，但是如果超过97%将返回第二高的数值，可以看下面的一对input

input:　　input:



output:

```
图一中的第1个脸的位置{'width': 70, 'top': 54, 'left': 348, 'height': 70}
图一中的第2个脸的位置{'width': 64, 'top': 61, 'left': 88, 'height': 64}
图二中的第1个脸的位置{'width': 103, 'top': 109, 'left': 166, 'height': 103}
两张图片中最高相似度为57.708%
```

由此可见， > 97%的值通常被认为是同一张图，所以自动避开了重复图片的输入输出，寻找第二高的项目，而经过我个人测试，大部分的测试结果都在80--92之间，这说明除非是同一张图， 不然很难达到相似度97这样的高峰，所以此api的设计可以说是很巧妙。