CS205 C/ C++ Program Design

Final Project Of CNN

Name：孔令凯　　SID：11912026

Part1.代码简介

本项目将被分为三部分介绍，其分别为：

① 图片的导入与初步处理；

② 卷积部分；

③ Max Pooling 部分的简述与部分判别思路；

项目代码已上传至 GitHub 网站，网址如下：

https://github.com/SickDawn/Final_Project_Of_CPP

注：因本项目并未得出合理的最终结果，即完成对于图片的人脸识别检测，所以将把重心放在研究对图片卷积的处理方法及对其结果的处理思路，使其能够更加合理地在后续判别中得到应用；

Part2.代码思路及方法详述

① 对于图片的导入与初步处理：

1. 采用 opencv 的 imread 方式导入图片，并以 Mat 的形式储存；
2. 对于 RGB 三色通道图片，将对其采用分离三色通道的形式，分别得到其三个颜色通道对应的灰度表达的单色道的 Mat 形式；

```
Mat imageForRed = projectConvolution.split_Redcolor(image);
Mat imageForGreen = projectConvolution.split_Greencolor(image);
Mat imageForBlue = projectConvolution.split_Bluecolor(image);
```

② 卷积部分：

1. 通过对预设的 kernal 的导入使得图片能够进行对应的卷积；

```
Mat kernal_3 = (Mat_<float>(3, 3) << 0.111, 0.111, 0.111,
    0.111, 0.111, 0.111,
    0.111, 0.111, 0.111);
```

```
Mat resultForKernal_3;
projectConvolution.load_kernal(kernal_3);
projectConvolution.convolute(image, resultForKernal_3,1);
imshow("用3*3核进行卷积", resultForKernal_3);
```

（其中 projectConvolution 为预设卷积用类，可自由导入卷积所用的 kernal 的值）

2. 支持各种类型的 kernal 和同 kernal 下的不同 stride 的卷积；

```
//5*5均值卷积核
Mat kernal_5 = (Mat_<float>(5, 5) << 0.04, 0.04, 0.04, 0.04, 0.04,
    0.04, 0.04, 0.04, 0.04, 0.04,
    0.04, 0.04, 0.04, 0.04, 0.04,
    0.04, 0.04, 0.04, 0.04, 0.04,
    0.04, 0.04, 0.04, 0.04, 0.04);
```

```
Mat resultForKernal_3_Step2;
projectConvolution.load_kernal(kernal_3);
projectConvolution.convolute(image, resultForKernal_3_Step2, 2);
imshow("用3*3核进行2步卷积", resultForKernal_3_Step2);

Mat resultForKernal_5;
projectConvolution.load_kernal(kernal_5);
projectConvolution.convolute(image, resultForKernal_5, 1);
imshow("用5*5核进行卷积", resultForKernal_5);
```

（红色部分改变了卷积的 stride 值）

3. 构造函数来自动补齐图像边框（即完成 padding=1 的部分），可保证卷积后保持图像大小不变；

```
void Final_Convolution::fillImage(const cv::Mat& image, cv::Mat& result)
{
    result = Mat::zeros(2 * dy + image.rows, 2 * dx + image.cols, image.type());
    Rect real_roi_of_image = Rect(dx, dy, image.cols, image.rows);
    Mat real_mat_of_image = result(real_roi_of_image);
    image.copyTo(result(real_roi_of_image));
}
```

（其中 result 为卷积后的结果，image 为卷积前的原图）

4. 对 RGB 图片分离颜色通道后的图片进行卷积，可得到对应图片的激活图，用于训练 cnn，给后续判别带来提升；

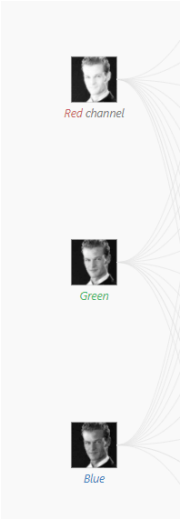
③ Max Pooling 部分的简述与部分判别思路：

1. 对于所得的激活图，当训练样本达到一定数量后，可较为精准地识别人脸；
2. 其判别过程可为：对已有样本的人脸进行分析后找出近似人脸图的滤波器（kernal），导入 projectConvolution 函数，然后对待识别目标所对应的单色通道灰度图进行依次滤波并找出其激活图；
3. 对于判别过程的详细解释将于下个部分做出图文解释与对其合理性给与测试数据并分析；

Part3.项目结果展示及结论分析

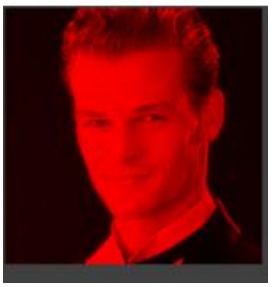
① 对于图片的导入与初步处理：

1. 对图片的 RGB 三色通道的分离以及对应灰度表达结果：



（左图为导入网站模拟的结果，右图为项目运行结果，该步骤结果基本相似）

2. 项目支持更加直观地反映各颜色通道的结果：



（红色图片的缩略图显示似乎有 bug，要单独查看，否则会有白条）

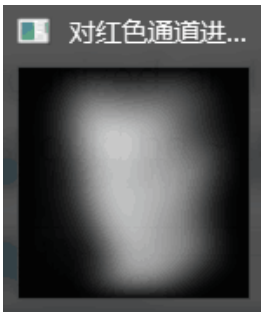
② 卷积部分：

1. 对未分离色道的原图进行卷积的结果（包含不同 kernal 和相同 kernal 不同 stride）：
   注：以下结果均已补齐结果图片的尺寸，及保证了卷积后图片尺寸与原图一致。

（最右图 stride 值为 2）

2. 对于单色通道的灰度图的卷积分析：

结论一：可用各单色道进行多次卷积后所得结果作为激活图，用于获取较为粗略的 kernal 来对图片进行初步的人脸识别，从而提速判别；



（以上结果分别为红色通道原图，卷积 1 次，10 次和 50 次的结果）

可以看出，对于 50 次卷积的结果图，红色通道的图仍保持人脸的基本轮廓，而此时其 Mat 所包含的数据量较原单色通道图相比以十分小。因此，可以通过对 50 次卷积后的激活图提取 kernal 来对新图进行滤波，若此时所得结果均不符合，即可判断新图大概率不含人脸。

对于蓝色通道结果基本类似，如下：

结论二：对于提取初步激活图所采用的 kernal 可以使用 5*5 等甚至更大的 kernal 来提取，尽管提取结果会不如 3*3 等的结果精致，但不影响初步判别。并且由于 kernal 的 size 变大，滤波次数降低，利于提速。

如下为蓝色通道 kernal_5*5 的卷积结果（依次为原图，1 次卷积，10 次）：

以下则为 kernal_3*3 时的卷积结果：

同结论一，由于降低了 kernal 提取的精度，能够提高初步判断的速度，而且对比结论一中卷积图可得，其对图片画质的影响比提高卷积次数所产生的影响小；

③ Max Pooling 部分的简述与部分判别思路：

1. 在对图片的整体卷积结果判别完毕后，可筛选出符合初步预期（即至少存在人脸大致轮廓的图片）；

   结论三：对符合初步预期的图片，只需要对人脸轮廓内部的图片信息进行判断识别即可。

   由于每次卷积不会改变图片的大小（已由 fillImage 函数修补回去了），因此可以得出人脸的大致轮廓在图中的坐标，此时只需对原图中相应坐标之内的数据进行判断即可。

   注：即使人脸轮廓并非规则的图形，但可将其范围大致用一个矩形圈出来。该步骤尽管可能损失判断精度，但在训练资料充足的情况下，一般不会产生过大的偏差。

   结论四：判断轮廓内部信息时，一般可对五官做一次识别过程，其判断过程依旧可参照结论一、二、三中所述。

2. 对于最终所得结论，无需将原图（待识别的图片）整张进行卷积、max pooling 直到得出最终数字结果，只要需要在上述过程的最后（即对五官的判断结束后），再返回所得结论即可。所以可得：

   结论五：依据上述方法，无需对整张图片进行反复卷积和合成，只需要对图片各个部分依次进行判断并且最终通过后，即可得出结论。

此操作增加了额外的判断过程，即判别人脸位置、五官位置等操作，此步骤会增加项目运行时间，但此操作减少了对无关区域的反复卷积计算，在图片尺寸较大时，能够获得不错的收益。