



《工程设计与管理课程设计》报告书

课题名称： 芯片 OCR 识别

学 号： 1810800127

姓 名： 夏小鱼

班 级： 18AI 创新班

所在学院： 计算机学院

时 间： 2021.12

计算机学院教学办公室

2021 年 11 月



目 录

目录

目 录	1
1 课题概述	2
1.1 背景	2
1.2 系统开发环境	2
1.3 设计要求	2
2 基础实验	4
2.1 形状分类实验	4
2.1.1 前言	4
2.1.2 实验目的	4
2.1.3 识别物体形状步骤	5
2.1.4 阈值分割实验	5
2.1.5 形状检测实验	8
2.1.6 形状分类	9
2.2 图像矫正实验	10
2.2.1 前言	10
2.2.2 实验目的	10
2.2.3 边缘点检测	10
2.2.4 轮廓提取	12
2.2.5 透视变换	13
2.3 OCR 识别	14
2.3.1 OCR 字符识别实验	14
2.3.2 EasyOCR 的使用	21
2.3.3 Qtmvp-trigger 的使用与 EasyOCR	23
3 综合实验	27
3.1 配置网络	27
3.1.1 设置 IP 地址	27
3.1.2 修改 DNS	27
3.2 芯片 OCR 识别应用项目	29
3.2.1 概述	29
3.2.2 项目要求	29
3.2.3 实验步骤	30
3.2.4 实验结果	错误!未定义书签。
结束语	28
致 谢	48
学习体会	49
参考文献	50

1 课题概述

1.1 背景

《工程设计与管理课程设计》（简称课程设计）是 AI 创新班的一个综合实践课程，是有别于课程实验的一个独立实践教学环节。课程设计一般在全部理论课程结束后的进行，教学时数为 2 周。

机器视觉是图像分析技术在工厂自动化中的应用，通过使用光学系统、工业数字相机和图像处理工具，来模拟人的视觉能力，并做出相应的决策，最终通过指挥某种特定的装置执行这些决策。一个典型的机器视觉应用系统包括图像捕捉、光源系统、图像数字化模块、数字图像处理模块、智能判断决策模块和机械控制执行模块。机器视觉作为实现工业自动化和智能化的关键核心技术，正成为人工智能发展最快的一个分支。因此，将人工智能理论与机器视觉相结合，设计面向项目驱动的机器视觉综合实验，具有典型性、技术先进性和综合性的特点。将机器视觉综合实验作为《工程设计与管理课程设计》的主要内容，利于学生建立人工智能课程内在知识点的联系，培养学生建立面向复杂工程问题求解能力与创新实践能力。

1.2 系统开发环境

本系统基于 Ubuntu18.04 操作系统，采用 jupyter 和 keras 来开发。

1.3 设计要求

课程设计考核内容包括设计作品、设计报告和答辩三个环节。设计作品包括可运行的源程序（刻录成光盘），系统使用说明，主要程序代码（打印附在课程报告内）。设计报告主要报告系统分析、设计和实现过程，内容如下：

- 1、问题定义及设计要求；
- 2、主要设计内容：详细报告课程设计中所做的主要工作，包括系统或组件的搭建与安装、数据存储、数据预处理、探索性分析、数据分析、数据可视化、算法与模块设计、编程及测试等。
- 3、总结与体会：写出本次课程设计的主要创新点及存在的问题。

4、参考文献：列出所参考的主要文献。

答辩环节要求学生对课程设计全过程思路清晰，关键技术熟练运用，能回答老师的提问。

本次课程设计成绩分三部分，设计作品占 40%，设计报告占 30%，答辩占 30%。评价因素主要有：

- 1、知识点覆盖范围及运用能力；
- 2、算法设计与应用能力；
- 3、图像数据采集-存储-建模-处理流程运用程度；
- 4、系统规模（代码行数）；
- 5、人机交互与表达（用户体验或评价）



2 基础实验

2.1 形状分类实验

2.1.1 前言

形状作为图像检索、目标识别等任务中的一种重要线索，一直是视觉领域中的研究热点与重点。随着计算机认知科学和人工智能的发展，人们对计算机提出了更高的要求，希望计算机能够自动“感知”世界，而感知的重要途径之一就是通过图像。从图像中包含的信息来看，主要含有目标的形状、颜色、纹理等信息。认知学的统计结果表明，相对于颜色和纹理等方面的信息，形状信息更能代表目标物的结构和姿态特征，具有颜色和纹理信息所不具备的特点。在实际应用中，目标物的轮廓、边缘等都可以用形状来表示，还有一些直接利用形状来代表特定含义的图像，如文字、商标、路标、车标等。因此，基于目标形状进行分析识别的方法对视觉领域具有重要的理论意义和应用价值，可以广泛的应用在图像检索、工业自动化、图像配准、产品检测、生物医学工程等领域。

2.1.2 实验目的

- (1)掌握 opencv 中阈值分割的方法;
- (2)掌握 opencv 中提取轮廓的方法;
- (3)学习利用物体外接形状特征来识别不同形状物体;
- (5)使用 opencv 在完成七巧板形状识别。

2.1.3 识别物体形状步骤

读入一幅七巧板图像到 opencv 中作为测试图像，使用阈值分割方法将七巧板区域与背景区域 分割，然后计算七巧板区域中每个区域的外接椭圆，并计算每个外接椭圆的长短轴比值作为 形状识别的特征。在形状识别结束后，再使用面积特征来区分不同形状大小的三角形，具体过程下图所示：



2.1.4 阈值分割实验

打开 jupyter 环境 首先我们打开实验室 docker 环境，进入
`~/share/industry_image_project/base1`， 然后打开 ipython notebook。

接下来选择 `shape_classify.ipynb` 进入到 notebook 环境中。

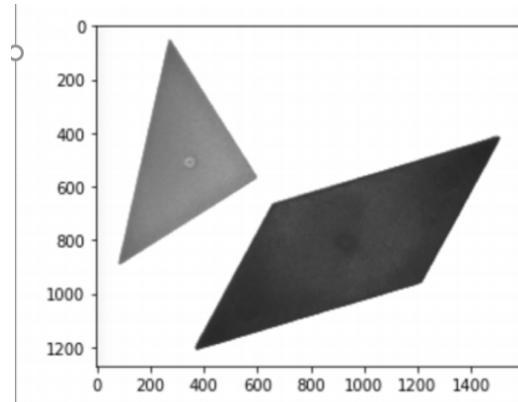
读取图片



```
import cv2
from matplotlib import pyplot as plt
from PIL import Image
import numpy as np
import imutils

im = cv2.imread("im1.png")

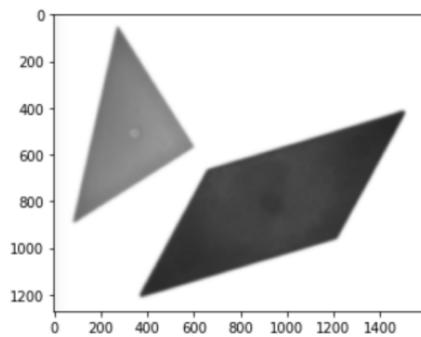
plt.imshow(im)
plt.show()
```



高斯模糊

由于在图像采集的过程中会存在很多噪点，为了方便我们进行阈值分割，我们首先对图像进行模糊，这里采用高斯模糊的方法。这样做的好处就是可以减少异常点的个数。

```
blurred = cv2.GaussianBlur(im, (35, 35), 0)
plt.imshow(blurred)
plt.show()
```

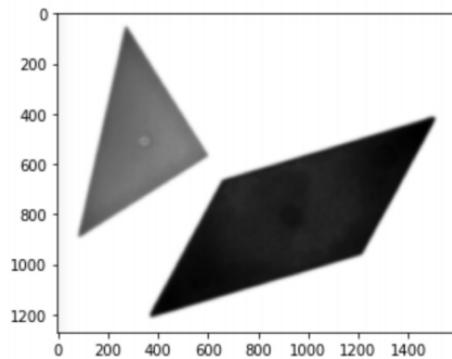




灰度化图像

接下来，我们将彩色图像转换为灰度图像。

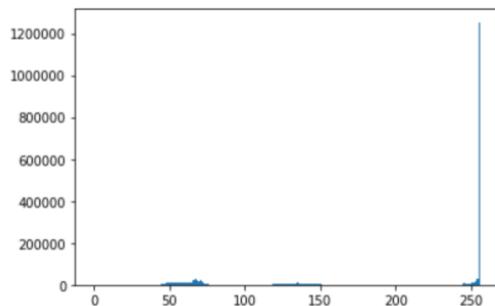
```
gray = cv2.cvtColor(blurred, cv2.COLOR_RGB2GRAY)
plt.imshow(gray, cmap="gray")
plt.show()
```



直方图分析

接下来，我们分析图像的直方图。我们的目标这里是过滤掉比较黑的图像，或者说把颜色浅的留下来。

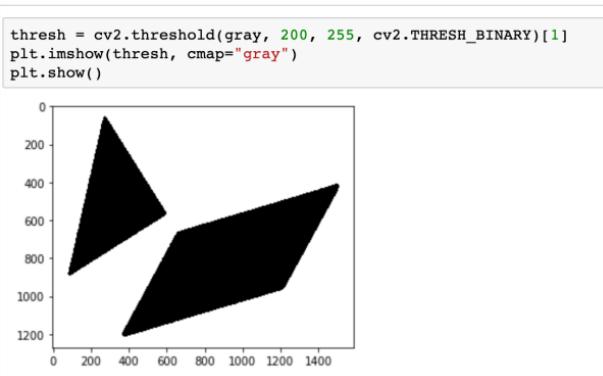
```
plt.hist(gray.ravel(), 256, [0, 256]);
plt.show()
```





阈值分割

这里我们可以调整阈值的大小，观看图像分割的情况，比如说阈值为 150 如下图所示。



2.1.5 形状检测实验

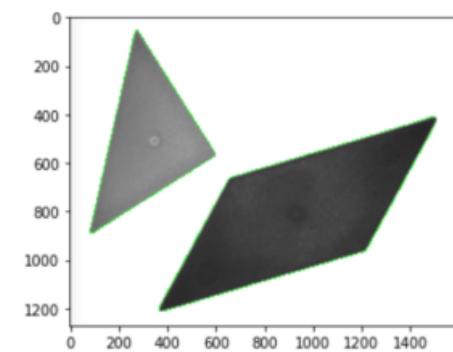
轮廓提取,在上面的实验中，我们已经获取到阈值分割好的图像了。接下来我们可以通过 opencv 中的方法找到所有的轮廓。这里运行代码，效果如下

```
cnts = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
---cv2.CHAIN_APPROX_SIMPLE)

cnts = imutils.grab_contours(cnts)

# loop over the contours
for c in cnts:
---# multiply the contour (x, y)-coordinates by the resize ratio,
---# then draw the contours and the name of the shape on the image
---c = c.astype("float")
---c = c.astype("int")
---cv2.drawContours(im, [c], -1, (0, 255, 0), 2)

plt.imshow(im)
plt.show()
```





2.1.6 形状分类

原理分析：

边数判断：对于我们的七巧板，最简单的一个判断的原则就是判断具有几条边。我们可以通过 opencv 提供的近似多边形的函数来获取每个轮廓的边的个数。

平行四边形与正方形：四边形中，我们这里需要判断平行四边形和正方形。这里有一个简单的方法就是通过外接矩形，判断长和宽的比值。当然，我们也可以通过比较面积，如果外接矩形的面积比轮廓的图形面积大很多，那就肯定是平行四边形了，反之则是正方形。理解原理后，我们可以定义一个形状检测器的类。

```
class ShapeDetector:
    def __init__(self):
        pass

    def detect(self, c):
        # initialize the shape name and approximate the contour
        shape = "unidentified"
        peri = cv2.arcLength(c, True)
        approx = cv2.approxPolyDP(c, 0.04 * peri, True)

        # if the shape is a triangle, it will have 3 vertices
        if len(approx) == 3:
            shape = "triangle"

        # if the shape has 4 vertices, it is either a square or
        # a rectangle
        elif len(approx) == 4:
            # compute the bounding box of the contour and use the
            # bounding box to compute the aspect ratio
            (x, y, w, h) = cv2.boundingRect(approx)
            ar = w / float(h)

            # a square will have an aspect ratio that is approximately
            # equal to one, otherwise, the shape is a rectangle
            shape = "square" if ar >= 0.95 and ar <= 1.05 else "parallelogram"

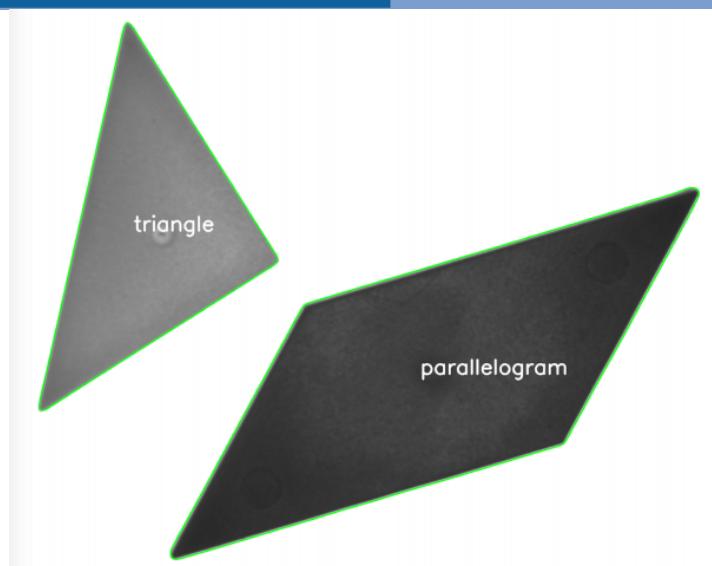
        # return the name of the shape
        return shape
```

最后循环判断每个轮廓的形状，并在原图中画出识别结果。

```
# loop over the contours
for c in cnts:
    # compute the center of the contour, then detect the name of the shape using only the contour
    M = cv2.moments(c)
    cX = int((M["m00"] / M["m00"]))
    cY = int((M["m01"] / M["m00"]))
    shape = sd.detect(c)

    # multiply the contour (x, y)-coordinates by the resize ratio,
    # then draw the contours and the name of the shape on the image
    c = c.astype("float")
    c = c.astype("int")
    cv2.drawContours(im, [c], -1, (0, 255, 0), 2)
    cv2.putText(im, shape, (cX-30, cY), cv2.FONT_HERSHEY_SIMPLEX,
               1.5, (255, 255, 255), 3)

plt.imshow(im)
plt.show()
```



2.2 图像矫正实验

2.2.1 前言

在进行光学扫描时，会因为客观原因，导致扫描的图像位置不正，影响后期的图像处理，因此需对图像进行图像矫正工作。比如说我们用到文档扫描手机 APP，拍一张图片，然后会自动识别出边和角，然后恢复成平面就属于一种图像矫正的应用。本节实验将通过 opencv 来实现一个图像矫正功能。

2.2.2 实验目的

掌握使用opencv 完成图像矫正的方法。

2.2.3 边缘点检测

打开 jupyter 环境 首先我们打开实验室 docker 环境，进入
~/share/industry_image_project/base2，然后打开 ipython notebook。接下来选择
rectify.ipynb 进入到 notebook 环境中。

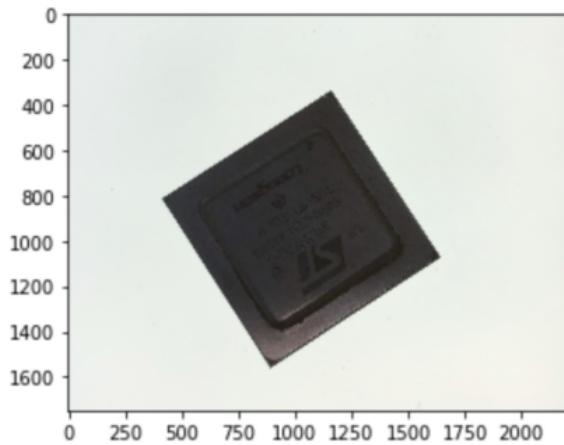
读取图像 我们是要 opencv 读取我们的测试图像



```
import cv2
import numpy as np
from matplotlib import pyplot as plt
import imutils
%matplotlib inline

pathImage = "1.jpg"
img = cv2.imread(pathImage)

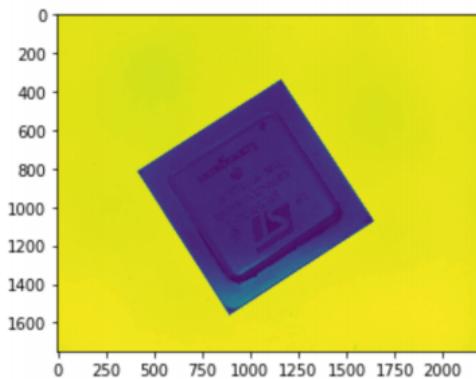
plt.imshow(img)
plt.show()
```



通过 canny 检测算子，检测边缘。

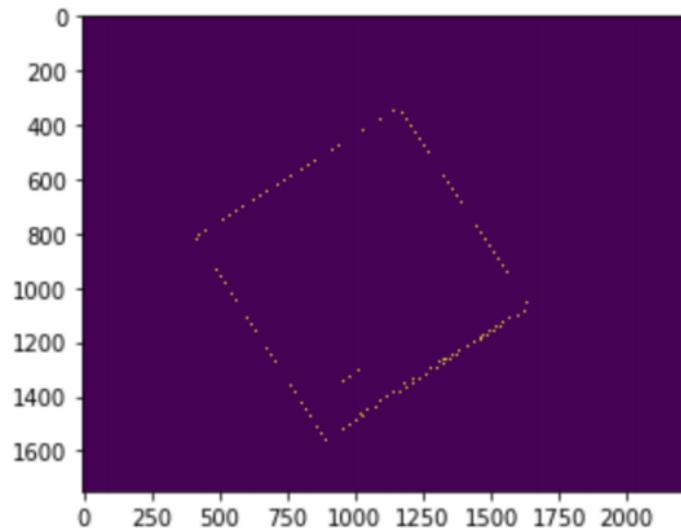
```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) # 转换成灰度图
gray = cv2.GaussianBlur(gray, (5,5),0) # 高斯滤波操作，消除噪音
edged = cv2.Canny(gray, 75, 200) # 边缘检测

plt.imshow(edged)
plt.show()
```





```
plt.imshow(edged)
plt.show()
```



2.2.4 轮廓提取

接下来通过 cv2.findContours 寻找边缘。

```
cnts = cv2.findContours(edged.copy(), cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
cnts = imutils.grab_contours(cnts)
# 按照区域的大小进行排序并获取前5个结果
cnts = sorted(cnts, key = cv2.contourArea, reverse = True)[:5]
```

然后通过多边形近似，可以看到下面得到了 4 条边。

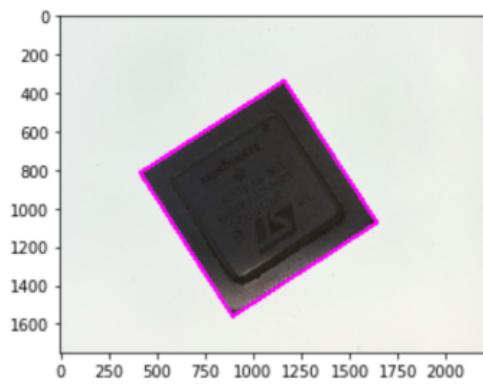
```
# 遍历整个轮廓集合
for c in cnts:
    # 使用多边形近似轮廓
    peri = cv2.arcLength(c, True)
    approx = cv2.approxPolyDP(c, 0.02 * peri, True)
    print(len(approx))
    screenCnt = approx
    if len(approx) == 4:
        screenCnt = approx
        break
```



绘制后如下图所示。

```
screenCnt
array([[[1162, 336]],
       [[ 415, 814]],
       [[ 895, 1560]],
       [[1642, 1073]]], dtype=int32)

cv2.drawContours(img, [screenCnt], -1, (255, 0, 255), 20)
plt.imshow(img)
plt.show()
```



2.2.5 透視变换

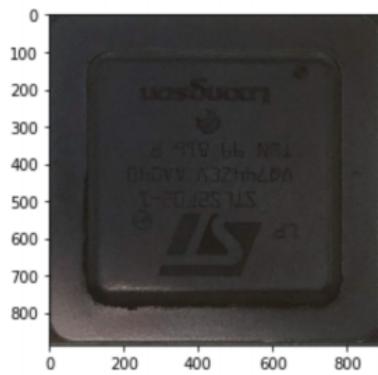
有了边缘的检测，最后通过 4 点透視变换得到矫正后的图片。

```
orig = cv2.imread(pathImage)

from imutils.perspective import four_point_transform

warped = four_point_transform(orig, screenCnt.reshape(4, 2))

plt.imshow(warped)
plt.show()
```



2.3 OCR 识别

2.3.1 OCR 字符识别实验

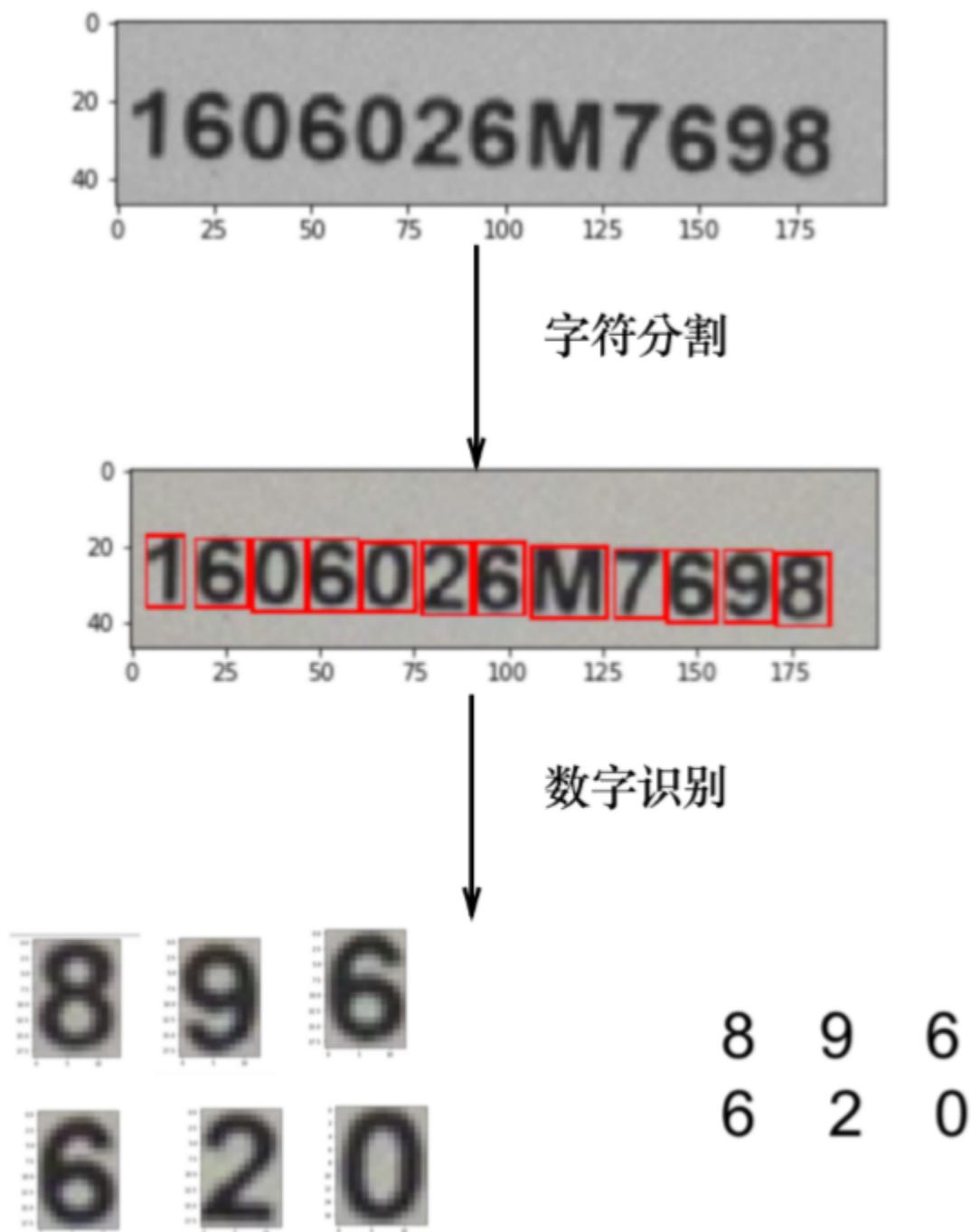
实验内容：随着计算机技术的迅猛发展，信息电子化已近成为一个必然趋势，而文字是信息中最重要的一种载体，其电子化程度决定了信息化的程度。OCR 技术改变了传统的纸质介质资料输入的概念。通过 OCR 技术，用户可以将通过摄像机、扫描仪等光学输入方式得到的报刊、书籍、文稿、表格等印刷品的图像信息转化为可以供计算机识别和处理的文本信息。因此，与传统的手工录入方式相比，OCR 技术大大提供了人们进行资料存储、检索、加工的效率，在银行、证券、保险、税务、公安、军队、交通等众多行业都有广泛的应用和发展。

实验目的：了解机器视觉系统的搭建和构成。掌握光学字符的基本方法。



实验步骤：

实验的实验步骤如下，首先使用基本的 opencv 图像处理算法，将每个字符进行分割，然后通过机器学习的方法对数字进行识别流程。





实验 1：图像基础操作

首先我们打开实验室 docker 环境，进入到

`~/share/industry_image_project/base3`，然后打开 ipython notebook。

选择 ocr.ipynb 进入到 notebook 环境中。

查看测试图片：首先我们来看一下，我们要进行识别的图片。这里测试数据放在章节目录的 `test_data` 下。读取彩色图像接下来我们引入本次课程所需的库文件。

然后通过 opencv2 读取图片，然后通过 matplotlib 工具显示图像。

这里我们读取的是一张彩色的图像。

从以上代码可以看到，我们这张图像的高位 47 个像素，长为 198 像素，然后有三个通道，分别对应着 RGB。

读取灰度图像：对于一般而言的图片识别，其实并需要彩色通道，所以我们这里读取黑白图像就好了。

通过以下代码读取黑白图像。

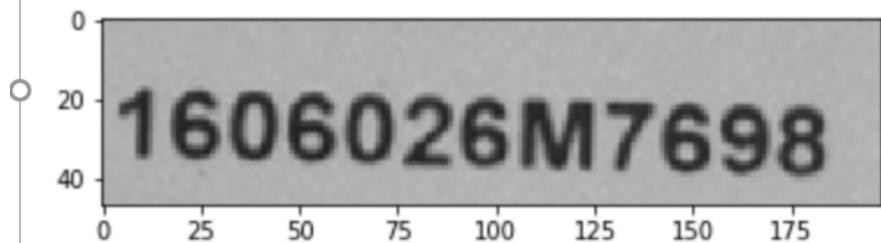
```
im = cv2.imread("test_data/1606026M7698.jpg", 0)
```

```
plt.imshow(im, cmap="gray", norm=NoNorm())
```

```
<matplotlib.image.AxesImage at 0x10fd81668>
```



```
<matplotlib.image.AxesImage at 0x10fd81668>
```



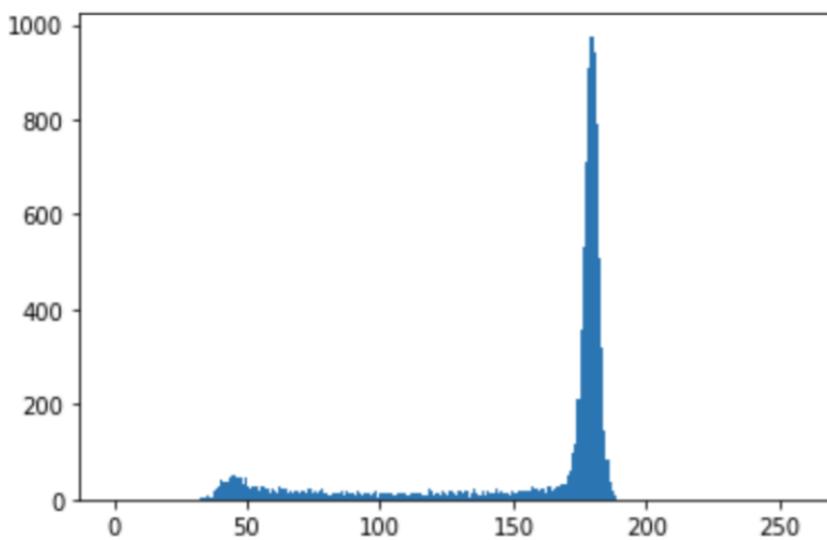
```
im.shape
```

```
(47, 198)
```

图像直方图在图像中，每个像素对应着 8 位，也就是 256 种情况。我们通过直方图可以查看像素灰度的分布情况。

这里 0 的是代表纯黑色，255 代表纯白色。我们这里的图像是浅白色的。可以看到这里背景的灰度大概是 180 左右。

```
plt.hist(im.ravel(), 256, [0, 256])  
plt.show()
```



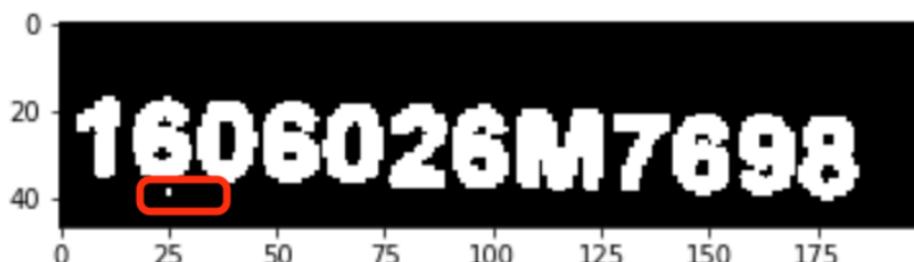
图像二值化一般来说我们不需要考虑 256 个灰度，只要看图像是黑或者白即可。

选取了一个阈值 160，将灰度 160 以下的都设置为纯白白色，160 以上都设置为纯黑色。可以得到如下的效果。

因为我们的数据集的背景色是纯黑的，这里对图像再取一个反，得到以下结果。

二值化后，图像依旧很清晰。图像的腐蚀在上述操作后，我们可以看到图像有些小白点。

我们可以通过图像的腐蚀算法来去除掉这样的小白点。



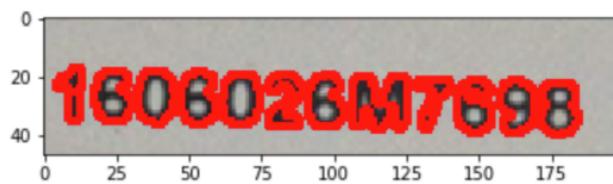
图像的膨腐蚀之后，我们再通过膨胀可以获得和原来差不多大的区域。（通过膨胀算法也可以去除掉小黑点。）

实验 2：字符分割

通过上述操作，通过肉眼我们基本上可以把每个字符都分割开了。接下来，我们来进行字符分割。寻找边缘 opencv 给我们提供了现成的函数 `findContours` 来寻找图像的边缘。

通过提供的函数描一遍边缘，可以看到效果如下。

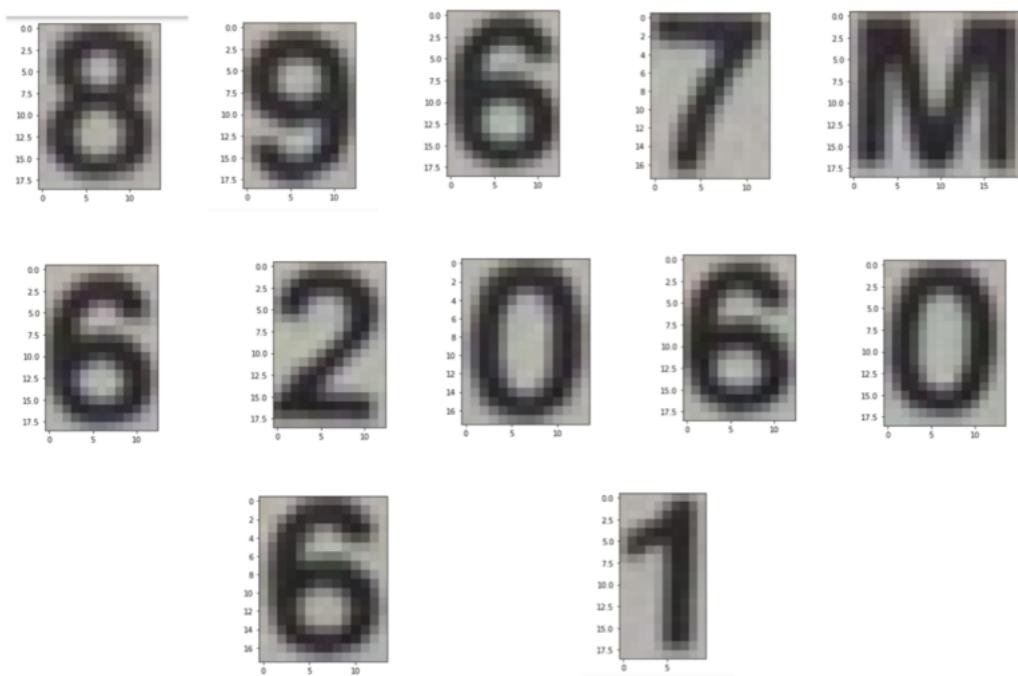
```
plt.imshow(im_origin)  
plt.show()
```



绘制矩形区域当然我们的目的其实并不是获取边缘，而是要分割出一个区域来，这里因为是图像，所以肯定要是一个矩形区域。

切分图像在 python-opencv 中，读取的图像都会是一个 numpy 数组。这里我们通过 python 的切片，可以分割出每个字符。

结果如下：



这里可以看到输出的字符顺序不一定是按照字符的先后顺序。所以在真实系统中，要考虑矩形 区域所处的横、纵坐标，然后再输出结果。

实验 3：数据集构建

接下来进行数据集的构建，我们已经标注好的数据存放在 chars 文件夹中。

获取图片的最大长度和宽度这里因为我们的图并不是同样尺寸的，而机器学习模型要求我们输入的维度必须相同。所以这里可以最大的长度和宽度作为图像的模板。

通过遍历，我们就可以得到每个字符的最大宽度和长度了。

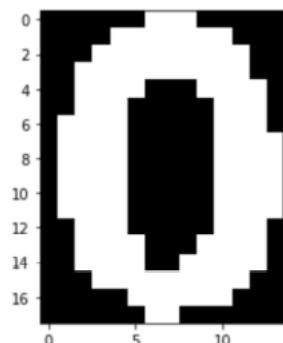


创建数据集模板：这里我们首先取出一张图片来看一下。

```
im_test = cv2.imread("chars/0/ch69_4_0.jpg", 0)
im_test = cv2.threshold(im_test, 160, 255, cv2.THRESH_BINARY)[1] # 2值化一下

print(im_test.shape)
plt.imshow(im_test, cmap="gray", norm=NoNorm())
plt.show()
```

(18, 14)



然后我们创建一个长为 27，宽为 28 的模板，将上面的图片放在模板里居中。

可以得到我们机器学习模型的输入。

生成数据集：

有了上面的函数，这里就可以生成可以用作机器学习的数据集了。

还是遍历所有图片，读取出来的图片首先经过二值化，然后再放到固定模板中，这里再进行一下归一化，因为图片是 256 灰度级的，我们只需要 0 和 1 就可以了。最后我们将 2 维矩阵转为 1 维向量。

这里有个 index2label 变量可以用在之后根据预测的结果，获取对应的字符。

最后通过 train_test_split 来分割训练集和测试集。

实验 4：模型训练

在有了数据之后，模型的训练就很简单了。直接调用 sklearn 的模型即可。

2.3.2 EasyOCR 的使用

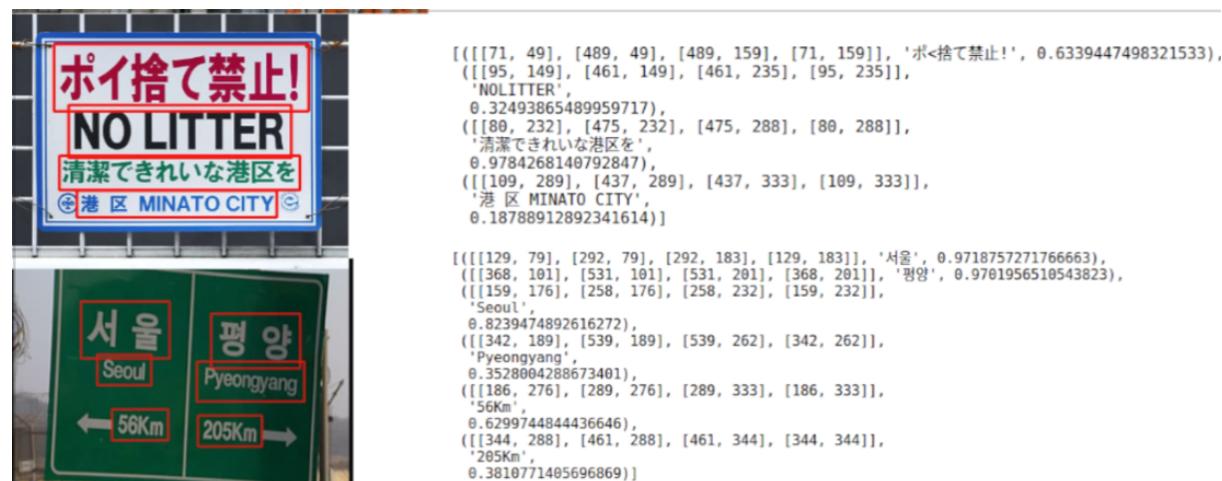
实验内容：

自己从头构建一个 OCR 系统固然能够很好的解决现有的数据，但是这样的 OCR 系统难以适应各种场景。在学习了 OCR 系统的基本原理后，本章实验将介绍一个通用的 OCR 识别工具：EasyOCR。

实验目的掌握使用 EasyOCR 完成光学字符识别。掌握 opencv 中对图片进行旋转。

EasyOCR 介绍 EasyOCR 是一个由开源用户在 [github](#) 上发起的一个项目，截止目前 2021 年年，EasyOCR 已经超过了 1.1 万 [github](#) 用户的关注。

EasyOCR 支持 80 多种语言言，使用 EasyOCR 识别图片的结果如下所示：



EasyOCR 的使用打开 ipython notebook 首先我们打开实验室 docker 环境，进入到 `~/share/industry_image_project/base3`，然后打开 ipython notebook。

接下来选择 `ocr2.ipynb` 进入到 notebook 环境中。

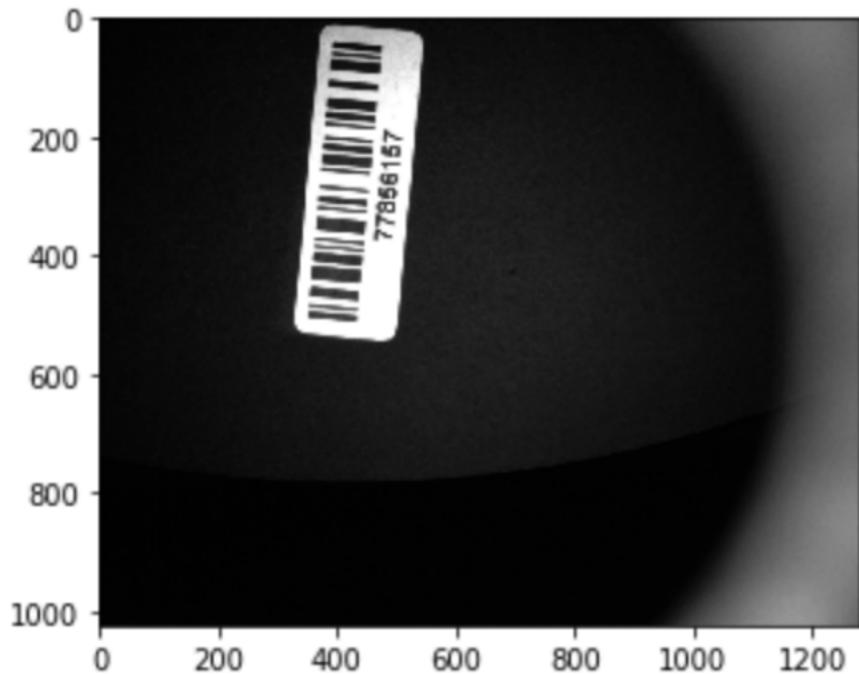
加载模型首先加载模型，通过下面一行代码即可，因为这里做的是数字 OCR，所以语言选英语即可。

读取测试图片接下来通过 opencv 读取一张我们转盘采集的图片。

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
```

```
im = cv2.imread("collect/2.bmp")
plt.imshow(im)
```

```
<matplotlib.image.AxesImage at 0x1317f9d30>
```



可以看到这里的图像不是正着的，我们可以调用 cv2.rotate 对图像进行旋转。

使用 EasyOCR 进行识别

调整好图片后，直接调用 EasyOCR 识别即可。

可以看到这里返回的是一个元组列列表，元组的第一个元素是边框，第二个是 OCR 的文字，最后是一个置信度。

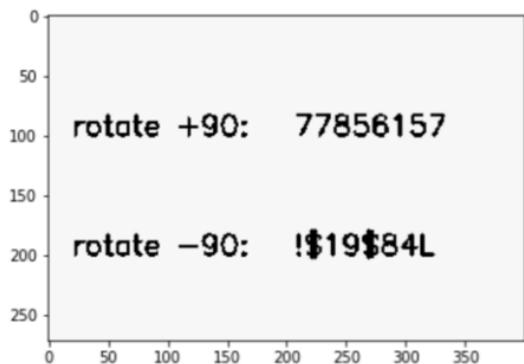
然后因为我们的条形码贴的不都是一个方向的，这里把逆时针旋转的结果也做一下。

结果显示：最后，为了在 QTMVP 程序中显示图像处理的结果，这里将识别的结果放在一个图片中。可以在这张图片中看到顺时针旋转已经逆时针旋转后，OCR 识别的结果。

```
img = cv2.imread('template.png')

font = cv2.FONT_HERSHEY_SIMPLEX # 定义字体
imgzi = cv2.putText(img, "rotate +90: "+rotate_90_txt, (20, 100), font, 0.8, (0, 0, 0), 2)
imgzi = cv2.putText(imgzi, "rotate -90: "+rotate_270_txt, (20, 200), font, 0.8, (0, 0, 0), 2)
plt.imshow(imgzi)
plt.show()

cv2.imwrite("result.png", imgzi)
```



2.3.3 Qtmvp-trigger 的使用与 EasyOCR

实验目的：

学习 qtmvp-trigger 程序的使用实践在实验室 docker 环境下和 qtmvp-trigger 程序通讯的方法

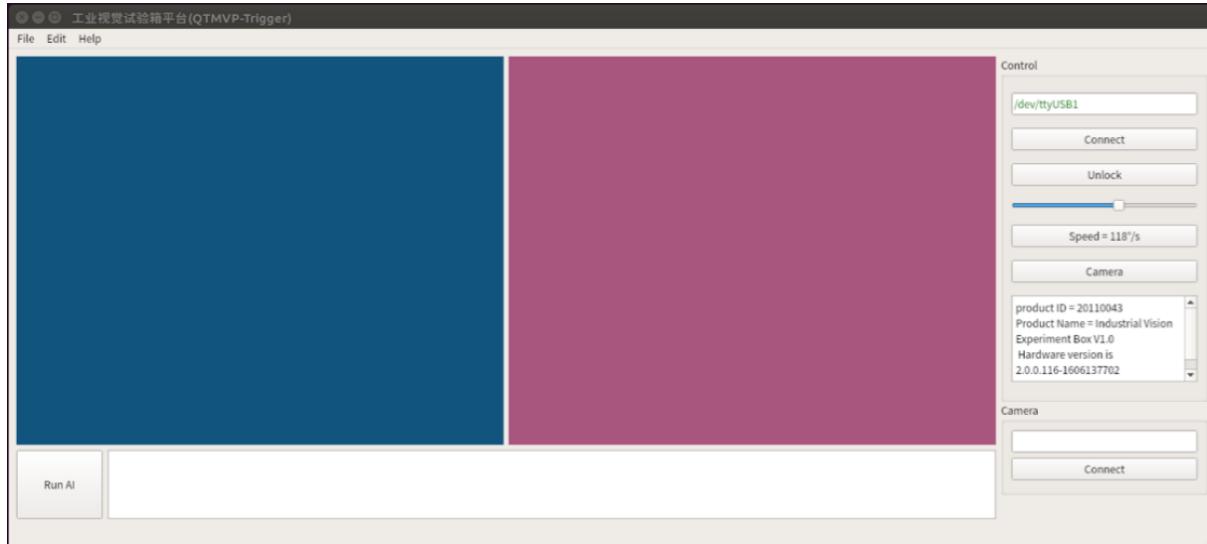
步骤 1：打开 QTMVP-Trigger 程序首先打开终端，然后进入到 qtmvp-trigger 程序的目录中。

```
cd share/qtmvp_trigger
```

接下来通过 sudo 模式启动 qtmvp_trigger 程序。

```
sudo ./qtmvp_trigger
```

打开后界面如下：



这个界面和之前的稍有不同，不过大体是一致的。我们先和之前的操作一样，打开摄像头，配置摄像头 ip，然后测试一下转盘是否可用。

步骤 2：打开 MVS 程序 确保摄像头已经连接好之后，我们打开 MVS 程序。

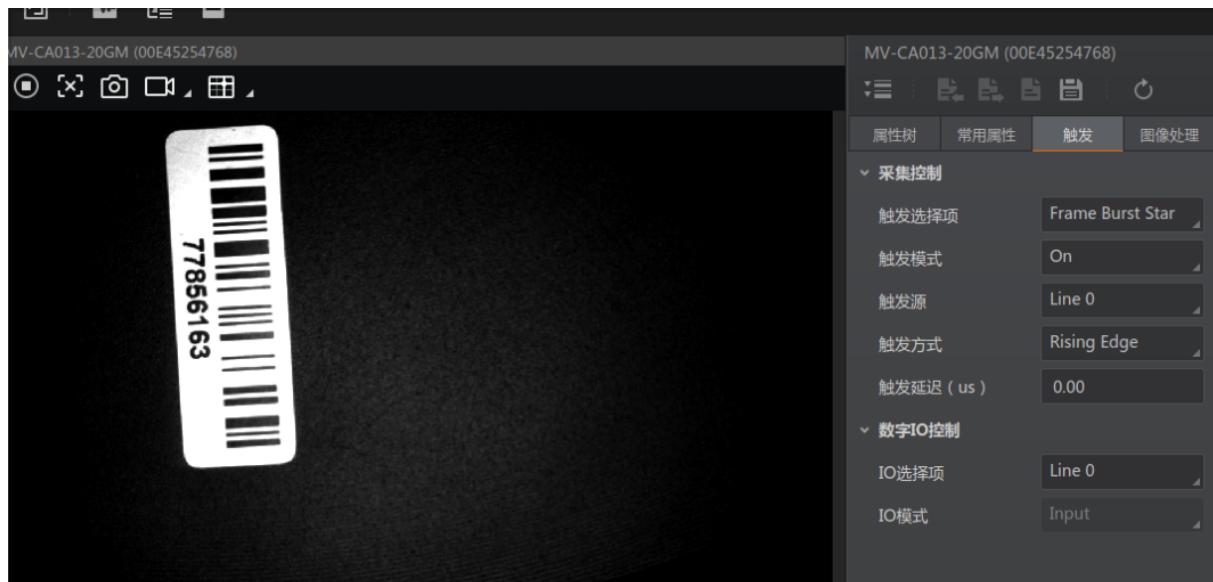
```
cd /opt/MVS/bin ./MVS.sh
```

本次实验需要用到转盘，光源 2（环形光源）。不会用到面光源，这里不要把面光源放在转盘上。

这里将触发模式设置成 On。

然后我们打开转盘（这里转盘的速度最好不要太快）。

并且调整好光源 2。得到一个清晰的图像。



步骤 3：通过实验箱采集图像配置好转盘的图像后，我们就可以关闭 MVS 软件了。

使用 qtmvp-trigger 连接摄像头回到 qtmvp-trigger 程序，我们点击连接摄像头的 connect 按钮。

点击后，我们可以看到一张条形码的图片。

关闭转盘接下来我们再次点击 Speed 按钮关闭转盘。

关闭后，可以看到图像就定格了。接下来我们对这张图片进行识别。

步骤 4：调用算法与 QTMVP 的交互接着 EasyOCR 实验的内容，这里我们 OCR 算法与 QTMVP-Trigger 程序交互的代码存放在 docker 环境中的 /share/industry_image_project/base3 目录下。

输入下面命令便可以启动 qtmvp_trigger 的交互程序。

```
python qtmvp_trigger_ocr.py
```

因为需要加载模型，这里需要等待显示出 OCR model have loaded 才可以进行算法交互。

算法识别效果 打开交互程序后我们再点击 Run AI 按钮，即可获取到算法识别的结果了。



代码说明：测试了效果没问题，我们来看下这个 `python` 文件是如何和 QTMVP-Trigger 程序进行交互的。

我们可以在 `notebook` 中打开 `qtmvp_trigger_ocr.py` 这个文件进行查看。

打开之后可以看到，首先是加载了 EasyOCR 的模型，然后定义了输入图片的路径，输出图片的路径，以及状态文件的路径。

接下来这里定义了一个 `run_ocr` 的函数，也就是这里的 OCR 识别以及结果图片绘制的算法了。

最后是通过状态文件和 QTMVP-Trigger 程序进行交互，这里可以看到是个死循环，首先读取状态文件，当状态文件内容为‘0’时，代表开始处理图片。处理完后，再将状态文件写为‘1’即可。



3 综合实验

3.1 配置网络

3.1.1 设置 IP 地址

首先打开终端，进入到 share 目录下，输入以下命令

```
cd share/network
```

```
cat setup_eth_ip.sh
```

```
master@labox:~$  
master@labox:~$ cd share/network  
master@labox:~/share/network$  
master@labox:~/share/network$ cat setup_eth_ip.sh  
#!/bin/bash  
  
sudo ifconfig eno1 10.12.15.4 netmask 255.255.255.0  
sudo route add default gw 10.12.15.254  
master@labox:~/share/network$
```

这里可以看到设置的 ip 地址为 10.12.15.4，这个最后的 4 是和机器座位编号对应的。我的座位是 41，那么 ip 地址就设为： 10.12.15.41

修改文件可以通过 gedit 软件

```
gedit setup_eth_ip.sh
```

确认没有问题后，运行 setup_eth_ip.sh。

```
bash ./setup_eth_ip.sh
```

3.1.2 修改 DNS

机房的 dns 配置为： 114.114.114.114

输入： sudo gedit /etc/resolvconf/resolv.conf.d/base

查看内容是否如下： nameserver 114.114.114.114

如果内容为空，填写 nameserver 114.114.114.114，再按下 Ctrl+S 键保存关闭即可。最后如果有更新，输入以下命令即可。

```
sudo resolvconf -u
```

3.2 标注平台

3.2.1 注册并登录账号

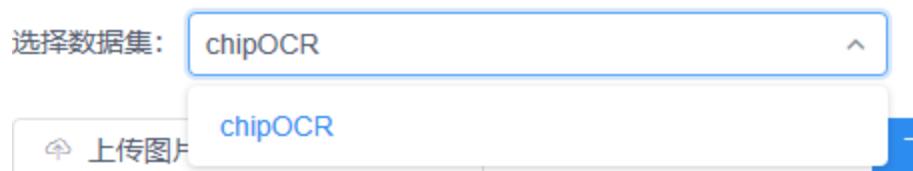
标注平台地址：<http://10.12.15.222:3079>

打开标注平台，进入注册页面。

注册完成后，进入登录页面登录个人账号。

3.2.2 上传数据

选择数据集：首先选择自己要标注的数据集名称。



上传数据分为上传单张图片以及批量上传。

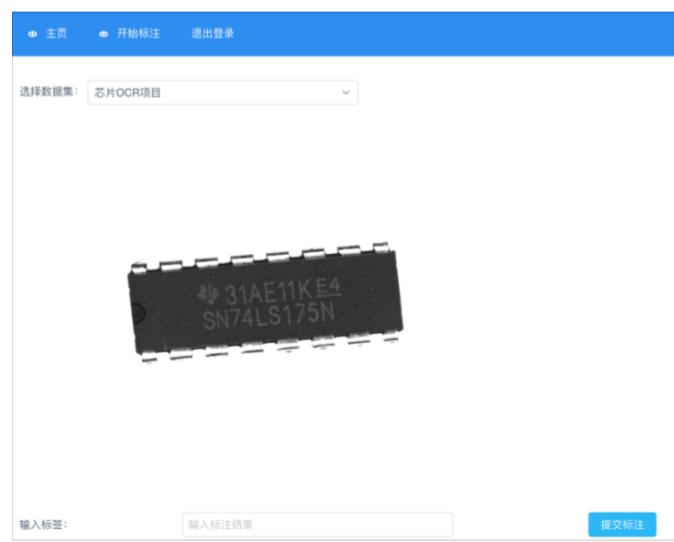
上传单张图片，点击上传图片，然后选择一张图片上传。

批量上传，可以上传一个压缩包，上传后系统会自动解压并将所有图片插入到要标注的图片中。

3.2.3 标注数据

点击最上面的“开始标注”就可以进行标注了。

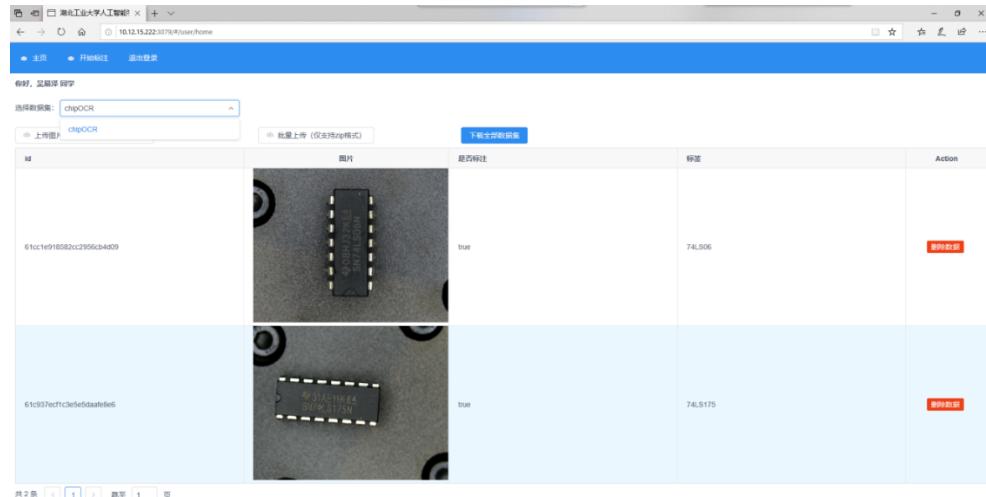
进入标注页面后，页面如下：



输入标签，点击提交标注即完成一张图片的标注。

输入标签： 提交标注

回到主页后，可以看到标注的结果。



序号	图片	是否标注	标签	Action
6fcfce916582cc2956c0409		true	74LS06	<button>删除数据</button>
61c937ecf1c1e5d5aafe8e6		true	74LS175	<button>删除数据</button>

数据是从后往前标注的，如果刚开始标注可以到最后一页查看标注结果。

3.2.4 下载数据

点击下载全部数据集就可以下载所有当前已经标注好的数据（包括其他同学已经标注好的数据）

3.3 芯片 OCR 识别应用项目

3.3.1 概述

字符识别系统主要演示的是芯片对准和芯片上光学字符识别的判定。本系统首先对芯片上的 logo 行训练，每次旋转后根据训练完成芯片对准，然后提取芯片上字符

3.3.2 项目要求

识别芯片上的 logo 的英文字母和数字

准确率不得低于百分之 80，可以尝试多种方法

3.3.3 实验步骤

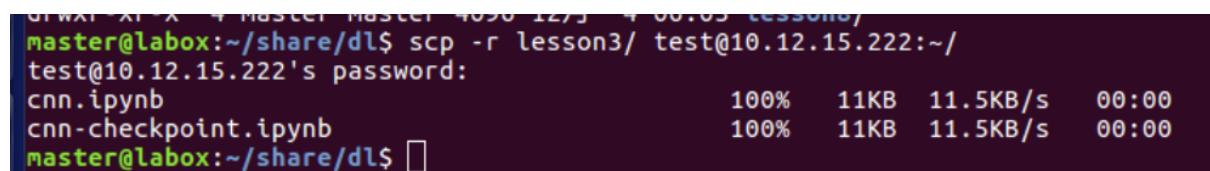
①上传数据：

上传文件的命令如下：

scp 本地地址 用户名@服务器地址:远程地址

如果传输的是一个文件夹，需要添加 -r 参数。

如下所示：



```
master@labox:~/share/dl$ scp -r lesson3/ test@10.12.15.222:~/  
test@10.12.15.222's password:  
cnn.ipynb 100% 11KB 11.5KB/s 00:00  
cnn-checkpoint.ipynb 100% 11KB 11.5KB/s 00:00  
master@labox:~/share/dl$
```

同理如果要下载数据，命令就是参数反过来：

scp 用户名@服务器地址:远程地址 本地地址

②远程登录并打开 Jupyter Notebook：

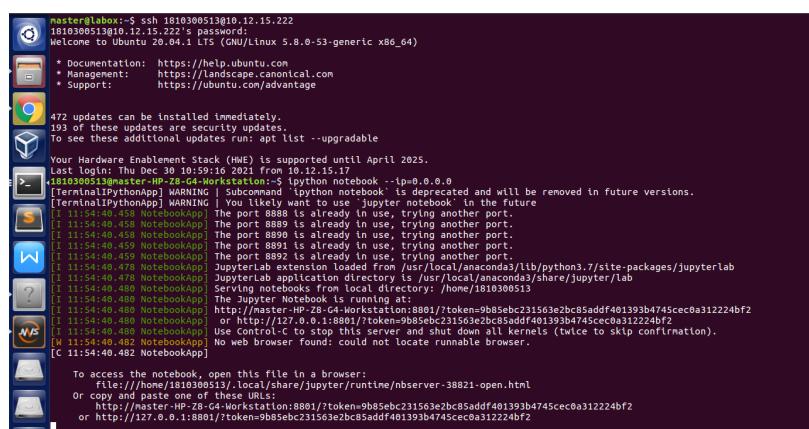
在老师开好 GPU 服务器账号后，可以通过 ssh 命令进行登录，登录到服务器上。

ssh 命令如下：

ssh 1810800127@10.12.15.222

输入密码后，我们就进入到服务器的终端里了。

然后输入 ipython notebook --ip=0.0.0.0



```
master@labox:~$ ssh 1810800127@10.12.15.222  
1810800127@10.12.15.222's password:  
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.8.0-53-generic x86_64)  
  
 * Documentation: https://help.ubuntu.com  
 * Management: https://landscape.canonical.com  
 * Support: https://ubuntu.com/advantage  
  
 472 updates can be installed immediately.  
 193 of these updates are security updates.  
 To see these additional updates run: apt list --upgradable  
  
Your Hardware Enablement Stack (HWE) is supported until April 2025.  
Last login: Thu Dec 30 10:59:11 2021 from 10.12.15.17  
+1810800127@master-HP-Z8-G4-Workstation:~$ ipython notebook --ip=0.0.0.0  
[TerminalPythonApp] WARNING | Starting Jupyter Notebook. This feature is deprecated and will be removed in future versions.  
[TerminalPythonApp] WARNING | You likely want to use "jupyter notebook" in the future.  
[TerminalPythonApp] WARNING | The port 8888 is already in use, trying another port.  
[TerminalPythonApp] WARNING | The port 8889 is already in use, trying another port.  
[TerminalPythonApp] WARNING | The port 8890 is already in use, trying another port.  
[TerminalPythonApp] WARNING | The port 8891 is already in use, trying another port.  
[TerminalPythonApp] WARNING | The port 8892 is already in use, trying another port.  
[TerminalPythonApp] JupyterLab extension loaded from /usr/local/anaconda3/lib/python3.7/site-packages/jupyterlab  
[TerminalPythonApp] JupyterLab application directory is /usr/local/anaconda3/share/jupyter/lab  
[TerminalPythonApp] JupyterLab static files directory is /home/1810800127  
[TerminalPythonApp] The Jupyter Notebook is running at:  
[TerminalPythonApp] http://master-HP-Z8-G4-Workstation:8881/?token=9b85ebc231563e2bc85addf401393b4745cec0a312224bf2  
[TerminalPythonApp] or http://127.0.0.1:8881/?token=9b85ebc231563e2bc85addf401393b4745cec0a312224bf2  
[TerminalPythonApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).  
[TerminalPythonApp] No web browser found; could not locate runnable browser.  
[TerminalPythonApp]
```

打开 JupyterNotebook

在服务器上的 ipython notebook 不会像在本地一样，自动在浏览器中打开。

我们需要把 127.0.0.1 替换成 10.12.15.222，然后再在自己的浏览器中打开。

③处理数据集：

将 dataset 中的一部分数据集划分为测试集，我处理的比例大概为 7:3.

④建立模型：

打开 project1-answer-dl.ipynb

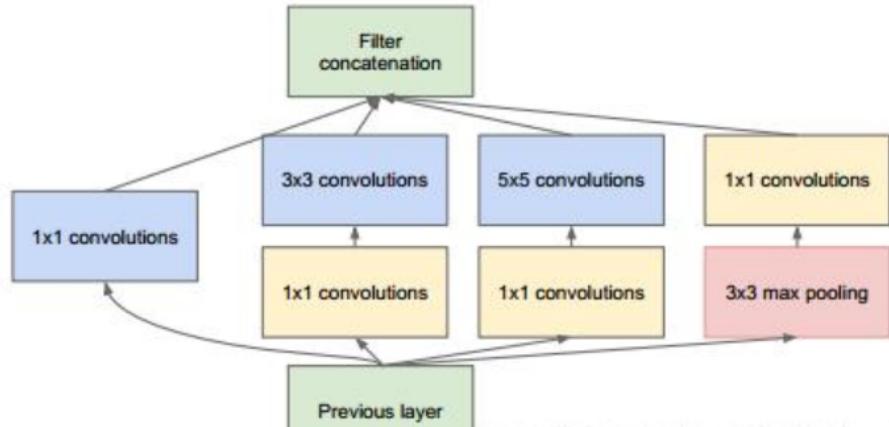
增加测试集读取模块，并且找到神经网络模型模块，选择自己的神经网络模型。

我选择的是 GoogleNet

GoogleNet 结构：

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								

GoogleLeNet 是在 AlexNet、VGG 分类网络之后出现的新的分类网络，其解决的问题获得更加高级的特征且同时减少训练参数。GoogLeNet 共有 22 层（卷积），其中包含 9 个线性堆叠的 Inception 模块，且在最后一个 Inception 模块处使用全局平均池化。



首先对前层特征图进行 1×1 卷积操作，用于降低通道数量，减少训练参数；之后使用不同大小的卷积核进行卷积操作，最后，按通道合并各特征图。

算法特点：

1. 若想获得更加高级的特征，以往的方式是增加网络的深度，而 GoogLeNet 则是通过增加网络的宽度来实现此目的。
2. Inception 模块综合考虑多个卷积核的运算结果，获得输入图像的不同信息，获得更好的图像表征。

建立模型代码展示如下：

```
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function
import os
import numpy as np
import time
import math
import paddle
import paddle.fluid as fluid
import codecs
import logging

from paddle.fluid.initializer import MSRA
from paddle.fluid.initializer import Uniform
from paddle.fluid.param_attr import ParamAttr
from PIL import Image
from PIL import ImageEnhance

train_parameters = {
    "input_size": [3, 224, 224],
    "class_dim": -1, # 分类数，会在初始化自定义 reader 的时候获得
    "image_count": -1, # 训练图片数量，会在初始化自定义 reader 的时候获得
```

```
"label_dict": {},  
  
"data_dir": "data/data2815", # 训练数据存储地址  
  
"train_file_list": "train.txt",  
  
"label_file": "label_list.txt",  
  
"save_freeze_dir": "./freeze-model",  
  
"save_persistable_dir": "./persistable-params",  
  
"continue_train": False, # 是否接着上一次保存的参数接着训练，优先级高于预训练模型  
  
"pretrained": True, # 是否使用预训练的模型  
  
"pretrained_dir": "data/data6593/GoogleNet_pretrained",  
  
"mode": "train",  
  
"num_epochs": 120,  
  
"train_batch_size": 30,  
  
"mean_rgb": [127.5, 127.5, 127.5], # 常用图片的三通道均值，通常来说需要先对训练数据做统计，此处仅取中间值  
  
"use_gpu": True,  
  
"dropout_seed": None,  
  
"image_enhance_strategy": { # 图像增强相关策略  
    "need_distort": True, # 是否启用图像颜色增强  
    "need_rotate": True, # 是否需要增加随机角度  
    "need_crop": True, # 是否要增加裁剪  
    "need_flip": True, # 是否要增加水平随机翻转  
    "hue_prob": 0.5,  
    "hue_delta": 18,
```



```
"contrast_prob": 0.5,  
"contrast_delta": 0.5,  
"saturation_prob": 0.5,  
"saturation_delta": 0.5,  
"brightness_prob": 0.5,  
"brightness_delta": 0.125  
,  
"early_stop": {  
    "sample_frequency": 50,  
    "successive_limit": 3,  
    "good_acc1": 0.92  
,  
"rsm_strategy": {  
    "learning_rate": 0.001,  
    "lr_epochs": [20, 40, 60, 80, 100],  
    "lr_decay": [1, 0.5, 0.25, 0.1, 0.01, 0.002]  
,  
"momentum_strategy": {  
    "learning_rate": 0.001,  
    "lr_epochs": [20, 40, 60, 80, 100],  
    "lr_decay": [1, 0.5, 0.25, 0.1, 0.01, 0.002]  
,  
"sgd_strategy": {
```

```
"learning_rate": 0.001,  
  
"lr_epochs": [20, 40, 60, 80, 100],  
  
"lr_decay": [1, 0.5, 0.25, 0.1, 0.01, 0.002]  
,  
  
"adam_strategy": {  
  
"learning_rate": 0.002  
,  
  
}  
}
```

GoogleNet:

```
class GoogleNet():  
  
    def __init__(self):  
  
        self.params = train_parameters  
  
  
    def conv_layer(self,  
                  input,  
                  num_filters,  
                  filter_size,  
                  stride=1,  
                  groups=1,  
                  act=None,  
                  name=None):  
  
        channels = input.shape[1]
```

```
        stdv = (3.0 / (filter_size**2 * channels))**0.5

    param_attr = ParamAttr(
        initializer=fluid.initializer.Uniform(-stdv, stdv),
        name=name + "_weights")

    conv = fluid.layers.conv2d(
        input=input,
        num_filters=num_filters,
        filter_size=filter_size,
        stride=stride,
        padding=(filter_size - 1) // 2,
        groups=groups,
        act=act,
        param_attr=param_attr,
        bias_attr=False,
        name=name)

    return conv

def xavier(self, channels, filter_size, name):
    stdv = (3.0 / (filter_size**2 * channels))**0.5

    param_attr = ParamAttr(
        initializer=fluid.initializer.Uniform(-stdv, stdv),
        name=name + "_weights")
```

```
return param_attr
```

```
def inception(self,  
             input,  
             channels,  
             filter1,  
             filter3R,  
             filter3,  
             filter5R,  
             filter5,  
             proj,  
             name=None) :  
  
    conv1 = self.conv_layer(  
        input=input,  
        num_filters=filter1,  
        filter_size=1,  
        stride=1,  
        act=None,  
        name="inception_" + name + "_1x1")  
  
    conv3r = self.conv_layer(  
        input=input,  
        num_filters=filter3R,  
        filter_size=1,
```

```
        stride=1,  
  
        act=None,  
  
        name="inception_" + name + "_3x3_reduce")  
  
    conv3 = self.conv_layer(  
  
        input=conv3r,  
  
        num_filters=filter3,  
  
        filter_size=3,  
  
        stride=1,  
  
        act=None,  
  
        name="inception_" + name + "_3x3")  
  
    conv5r = self.conv_layer(  
  
        input=input,  
  
        num_filters=filter5R,  
  
        filter_size=1,  
  
        stride=1,  
  
        act=None,  
  
        name="inception_" + name + "_5x5_reduce")  
  
    conv5 = self.conv_layer(  
  
        input=conv5r,  
  
        num_filters=filter5,  
  
        filter_size=5,  
  
        stride=1,  
  
        act=None,
```

```
name="inception_" + name + "_5x5")  
  
pool = fluid.layers.pool2d(  
  
    input=input,  
  
    pool_size=3,  
  
    pool_stride=1,  
  
    pool_padding=1,  
  
    pool_type='max')  
  
convprj = fluid.layers.conv2d(  
  
    input=pool,  
  
    filter_size=1,  
  
    num_filters=proj,  
  
    stride=1,  
  
    padding=0,  
  
    name="inception_" + name + "_3x3_proj",  
  
    param_attr=ParamAttr(  
  
        name="inception_" + name + "_3x3_proj_weights"),  
  
    bias_attr=False)  
  
cat = fluid.layers.concat(input=[conv1, conv3, conv5, convprj],  
axis=1)  
  
cat = fluid.layers.relu(cat)  
  
return cat  
  
def net(self, input, class_dim=1000):  
  
    conv = self.conv_layer(
```



```
        input=input,
        num_filters=64,
        filter_size=7,
        stride=2,
        act=None,
        name="conv1")

pool = fluid.layers.pool2d(
    input=conv, pool_size=3, pool_type='max', pool_stride=2)

conv = self.conv_layer(
    input=pool,
    num_filters=64,
    filter_size=1,
    stride=1,
    act=None,
    name="conv2_1x1")

conv = self.conv_layer(
    input=conv,
    num_filters=192,
    filter_size=3,
    stride=1,
    act=None,
    name="conv2_3x3")
```

```
pool = fluid.layers.pool2d(  
    input=conv, pool_size=3, pool_type='max', pool_stride=2)  
  
ince3a = self.inception(pool, 192, 64, 96, 128, 16, 32, 32,  
"ince3a")  
  
ince3b = self.inception(ince3a, 256, 128, 128, 192, 32, 96, 64,  
"ince3b")  
  
pool3 = fluid.layers.pool2d(  
    input=ince3b, pool_size=3, pool_type='max', pool_stride=2)  
  
ince4a = self.inception(pool3, 480, 192, 96, 208, 16, 48, 64,  
"ince4a")  
  
ince4b = self.inception(ince4a, 512, 160, 112, 224, 24, 64, 64,  
"ince4b")  
  
ince4c = self.inception(ince4b, 512, 128, 128, 256, 24, 64, 64,  
"ince4c")  
  
ince4d = self.inception(ince4c, 512, 112, 144, 288, 32, 64, 64,  
"ince4d")  
  
ince4e = self.inception(ince4d, 528, 256, 160, 320, 32, 128, 128,  
"ince4e")  
  
pool4 = fluid.layers.pool2d(  
    input=ince4e, pool_size=3, pool_type='max', pool_stride=2)  
  
ince5a = self.inception(pool4, 832, 256, 160, 320, 32, 128, 128,
```

```
"ince5a")
```

```
ince5b = self.inception(ince5a, 832, 384, 192, 384, 48, 128, 128,
                        "ince5b")

pool5 = fluid.layers.pool2d(
    input=ince5b, pool_size=7, pool_type='avg', pool_stride=7)

dropout = fluid.layers.dropout(x=pool5, dropout_prob=0.4)

model = fluid.layers.fc(input=dropout,
                        size=class_dim,
                        act='softmax',
                        param_attr=self.xavier(1024, 1, "out"),
                        name="out",
                        bias_attr=ParamAttr(name="out_offset"))

return model
```

部分代码主要截图：

```
train_parameters = {
    "input_size": [3, 224, 224],
    "class_dim": -1, # 分类数，会在初始化自定义 reader 的时候获得
    "image_count": -1, # 训练图片数量，会在初始化自定义 reader 的时候获得
    "label_dict": {},
    "data_dir": "data/data2815", # 训练数据存储地址
    "train_file_list": "train.txt",
    "label_file": "label_list.txt",
    "save_freeze_dir": "./freeze-model",
    "save_persistable_dir": "./persistable-params",
    "continue_train": False, # 是否接着上一次保存的参数接着训练，优先级高于预训练模型
    "pretrained": True, # 是否使用预训练的模型
    "pretrained_dir": "data/data6593/GoogleNet_pretrained",
    "mode": "train",
    "num_epochs": 120,
    "train_batch_size": 30,
    "mean_rgb": [127.5, 127.5, 127.5], # 常用图片的三通道均值，通常来说需要先对训练数据做统计，此处仅取中间值
    "use_gpu": True,
    "dropout_seed": None,
    "image_enhance_strategy": { # 图像增强相关策略
        "need_distort": True, # 是否启用图像颜色增强
        "need_rotate": True, # 是否需要增加随机角度
        "need_crop": True, # 是否要增加裁剪
        "need_flip": True, # 是否要增加水平随机翻转
        "hue_prob": 0.5,
        "hue_delta": 18,
        "saturation_prob": 0.5,
        "saturation_delta": 32,
        "contrast_prob": 0.5,
        "contrast_delta": 32,
        "sharpness_prob": 0.5,
        "sharpness_delta": 32
    }
}
```



```
class GoogleNet():
    def __init__(self):
        self.params = train_parameters

    def conv_layer(self,
                  input,
                  num_filters,
                  filter_size,
                  stride=1,
                  groups=1,
                  act=None,
                  name=None):
        channels = input.shape[1]
        stdv = (3.0 / (filter_size**2 * channels))**0.5
        param_attr = ParamAttr(
            initializer=fluid.initializer.Uniform(-stdv, stdv),
            name=name + "_weights")
        conv = fluid.layers.conv2d(
            input=input,
            num_filters=num_filters,
            filter_size=filter_size,
            stride=stride,
            padding=(filter_size - 1) // 2,
            groups=groups,
            act=act,
            param_attr=param_attr,
```

```
def inception(self,
              input,
              channels,
              filter1,
              filter3R,
              filter3,
              filter5R,
              filter5,
              proj,
              name=None):
    conv1 = self.conv_layer(
        input=input,
        num_filters=filter1,
        filter_size=1,
        stride=1,
        act=None,
        name="inception_" + name + "_1x1")
    conv3r = self.conv_layer(
        input=input,
        num_filters=filter3R,
        filter_size=1,
        stride=1,
        act=None,
        name="inception_" + name + "_3x3_reduce")
    conv3 = self.conv_layer(
        input=conv3r,
        num_filters=filter3,
        filter_size=3,
        stride=1,
        act=None)
```



```
def inception(self,
               input,
               channels,
               filter1,
               filter3R,
               filter3,
               filter5R,
               filter5,
               proj,
               name=None):
    conv1 = self.conv_layer(
        input=input,
        num_filters=filter1,
        filter_size=1,
        stride=1,
        act=None,
        name="inception_" + name + "_1x1")
    conv3r = self.conv_layer(
        input=input,
        num_filters=filter3R,
        filter_size=1,
        stride=1,
        act=None,
        name="inception_" + name + "_3x3_reduce")
    conv3 = self.conv_layer(
        input=conv3r,
        num_filters=filter3,
        filter_size=3,
        stride=1,
        act=None)
```

训练模型：

```
model=GoogleNet()
model.summary()
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])

: model=GoogleNet()
model.summary()
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

在服务器上运行了 20 个 epoch 后得到了 90+% 的训练集正确率，但是存在 30+ 个 epoch 会出现服务器卡顿的情况。

验证结果：

```
import json

with open ("index2ch.json", "w") as f:
    json.dump(index2ch, f)

im = cv2.imread ("TempPhoto.bep")

im = recitify(im)

dim (150, 50)

resized = cv2.resize(im, dim, interpolation = cv2.INTER_AREA)

Image.fromarray(np.array(resized, dtype=np.uint8))
```

随机拍照测试

```
In [ ]: import json
with open ("index2ch.json", "w") as f:
    json.dump(index2ch, f)
im = cv2.imread ("TempPhoto.bep")
im = recitify(im)
dim (150, 50)
resized = cv2.resize(im, dim, interpolation = cv2.INTER_AREA)
Image.fromarray(np.array(resized, dtype=np.uint8))

In [ ]: res = model.predict(np.array(resized, dtype=np.uint8)).reshape(1, 50, 150, 3)
result = "74LS" + index2ch[res[0].argmax(1)[0]] + index2ch[res[1].argmax(1)[0]] + index2ch(result)
result = result.replace("_", "")
```

在最终答辩与展示过程中，现场演示当场拍照并且准确识别出芯片字母，显示出模型的准确识别。



结束语

此次课设系统模拟工厂化图像识别，主要测试芯片标号识别。在实践过程中，我们通过基础实验熟悉 Linux 系统，了解机器视觉的应用，从图像采集，光源系统，到数字图像处理模块再到图像识别，分类模块，完完全全的实现了一整套流程。

将人工智能与机器视觉相结合，这一过程中有无数种优化方法，但如何找到最优解，最快速最高效地解决实际问题，是我们仍然需要探索和改进的。

致 谢

首先感谢课程设计指导老师的悉心指导，为我们规划任务。合理的设计课设内容，循序渐进，让我们从一知半解到熟悉设备再到自己搭建合适的模型完成课设，这个过程与导师们的课程安排是分不开的。

其次感谢我的同学，大家在熟悉环境的过程中，遇到困难自己解决之后还会主动在课程群里告知，避免同学不停陷入同一个坑里面。这样的分享大大提高了我们整个班级的课设进度。

最后我想感谢学校，为我们新增这一优越的实验环境，相信计算机学院会培养出越来越多的人才！

学习体会

在开始之前，一直感叹人工智能实验室的新增设备，好奇地想知道各个设备如何搭配使用。但是刚开始网络接口的问题就难倒了一大批同学，好不容易解决了，之后又是 GPU 占用的问题，这些看似与实验模型无关的问题不断地打断着实验进度。初期我也在烦恼，但是后来我明白这些问题都是课设的一部分，并不能因为不是核心内容就忽略不管，而是去想出办法解决，就是意想不到的很多小问题在实验中产生，然后我们合理避免，才在实际运用中或者是投入生产中不会出错。

总而言之，经过这次课设，我更加细致耐心地完成各个模块中的小细节，不懂就查，错了就换一种方法，总比呆呆的烦恼要好。实验就是不停的试错的过程！

参考文献

- 【1】Ballester P, Araujo R M. On the performance of GoogLeNet and AlexNet applied to sketches[C]//Thirtieth AAAI Conference on Artificial Intelligence. 2016.
- 【2】Zhong Z, Jin L, Xie Z. High performance offline handwritten chinese character recognition using googlenet and directional feature maps[C]//2015 13th International Conference on Document Analysis and Recognition (ICDAR). IEEE, 2015: 846-850.
- 【3】付若楠. 基于深度学习的目标检测研究[D].北京交通大学,2017.
- 【4】汤鹏杰,谭云兰,许恺晟,李金忠.基于 GoogLeNet 多阶段连带优化的图像描述[J].井冈山大学学报(自然科学版),2016,37(05):47-57.
- 【5】李炳臻,刘克,顾佼佼,姜文志.卷积神经网络研究综述[J].计算机时代,2021(04):8-12+17.DOI:10.16644/j.cnki.cn33-1094/tp.2021.04.003.
- 【6】谢一德. 基于深度卷积神经网络和图像传感器的道路多目标检测研究[D].北京交通大学,2018.
- 【7】赵旭江. 基于卷积神经网络的遥感图像目标检测与识别[D].中国科学技术大学,2018.
- 【8】张选,胡晓娟.基于 GoogLeNet 和 ResNet 的深度融合神经网络在脉搏波识别中的应用[J].计算机系统应用,2019,28(10):15-26.DOI:10.15888/j.cnki.csa.007110.
- 【9】刘子毅. 基于图谱特征分析的农业虫害检测方法研究[D].浙江大学,2017.
- 【10】薛勇,王立扬,张瑜,沈群.基于 GoogLeNet 深度迁移学习的苹果缺陷检测方法[J].农业机械学报,2020,51(07):30-35.