《图像处理与模式识别》实验指导书(Python-Opencv版)

目录

实验环境配置

一、Anaconda安装

1.下载
选择1--官网
选择2--镜像网站安装（推荐）

2.安装
具体安装细节可以参考这个网页：https://zhuanlan.zhihu.com/p/75717350

二、虚拟环境配置

1.配置虚拟环境的原因
在实际项目开发中，通常会根据自己的需求去下载各种相应的框架库，但是可能每个项目使用的框架库并不一样，或使用框架的版本不一样，那就需要我们根据需求不断的更新或卸载相应的库。如果直接对Python环境操作，会对开发环境和项目造成很多不必要的麻烦，管理也相当混乱。

2.配置虚拟环境步骤
步骤一：用管理员状态打开CMD
步骤二：输入命令 conda create --name env_name python=3.9（这里的env_name是这个虚拟环境的命名，虚拟环境建立在anaconda安装路径下的envs文件夹里　　面，3.9是python的版本，可以根据需求修改）
步骤三：激活虚拟环境，通过输入activate+虚拟环境名实现
步骤四：在虚拟环境中，根据项目需求进行包的下载，而这些操作都会在anaconda的envs对应的虚拟环境中修改，不会影响实际的环境
步骤五：完成项目操作后，可以通过deactivate退出虚拟环境

注：包的安装与卸载
安装：pip install pakage_name
卸载：pip uninstall pakage_name

实验一 图像色彩、代数、几何变换

一、实验意义及目的

（1）了解和掌握图像处理工具软件，熟悉相关图像处理函数，并为下一步编程进行图像处理打下基础。
（2）理解色彩的概念，掌握图像代数运算，几何变换方法。

二、实验内容

打开一幅彩色图像Image1，对其进行下列变换：
（1）将Image1红绿色彩互换，并显示效果；
（2）将Image1灰度化为gray，并显示灰度化后图像；
（3）采用不同的插值方法实现gray的旋转、放大变换；
（4）打开另一幅彩色图像Image2，和Image1进行代数运算，要求运用拼接、加减乘除等多种技术。
（5）使用python中matplotlib.pyplot进行图像显示。（未提供代码）
（6）将蔬菜RGB图像转换到HSI空间，变换HSI空间各通道值，看图像效果。（未提供代码）

三、相关函数介绍

(1)import cv2
功能：导入opencv模块

(2)cv2.imread（）
功能：实现多种类型图像文件的读取
调用格式：
cv2.imread（filename,flags）
filename是图片存储的具体路径，flag的值定义了如何读取这幅图片（可以根据自己的需要对参数进行设置）
flag = -1，8位深度，原通道
flag = 0， 8位深度，1通道
flag = 1， 8位深度，3通道
flag = 2， 原深度， 1通道
flag = 3， 原深度， 3通道
flag = 4， 8位深度，3通道

(3)cv2.namedWindow（）
功能：新建一个窗口，并且可以指定窗口的类型
调用格式：
cv2.namedWindow（winname,flags）
第一个参数是窗口名字
第二个参数一般有4种选择
cv2.WINDOW_NORMAL:窗口大小可改变
cv2.WINDOW_AUTOSIZE：无法调整窗口大小，该大小受所显示图像的限制。
cv2.WINDOW_FREERATIO：自适应比例。
cv2.WINDOW_KEEPRATIO：尊重图像的比例。

(4)cv2.imshow（）
功能：显示图像
调用格式：
cv2.imshow（winname，mat）
第一个参数是窗口名字，第二个参数是需要显示的图片

(5)cv2.waitKey（）
功能：定义窗口的显示时间
调用格式：
cv2.waitKey(delay)
参数=NONE & 0表示一直显示，其他数值表示显示的毫秒数

(6)cv2.destroyAllWindows（）
功能：删除窗口,括号里不指定任何参数，则删除所有窗口，删除特定的窗口，往括号输入特定的窗口值。

(7)cv2.cvtColor（）
功能：颜色空间转换
调用格式：
cv2.cvtColor(input_image，flags)
第二个参数常使用的转换的格式有：
cv2.COLOR_BGR2GRAY:将BGR格式转换成灰度格式
cv2.COLOR_BGR2HSV :将BGR格式转换成HSV格式
cv2.COLOR_BGR2HLS :将BGR格式转换成HLS格式
cv2.COLOR_RGB2HSV :将RGB格式转换成HSV格式

(8)cv2.getRotationMatrix2D（）
功能：获得仿射变换矩阵M
调用格式：
cv2.getRotationMatrix2D(center, angle, scale)
这里的三个参数 分别是中心位置，旋转角度（正值为逆时针旋转，负值为顺时针旋转）和缩放程度

(9)cv2.warpAffine（）
功能：利用变换矩阵对图像进行如旋转、仿射、平移等变换
调用格式：
cv2.warpAffine(src, M, dsize,dst,flags,borderMode,borderValue)
参数说明：
src 输入图像
M 变换矩阵，一般反映平移或旋转的关系，为InputArray类型的2×3变换矩阵。
dsize 输出图像的大小
flags 插值方法的组合（int 类型）
borderMode 边界像素模式（int 类型）
borderValue 边界填充值; 默认情况下，它为0，也就是边界填充默认是黑色。

```
其中flags表示插值方式，有以下取值
cv2.INTER_LINEAR     线性插值(默认)
cv2.INTER_NEAREST    最近邻插值
cv2.INTER_AREA       区域插值
cv2.INTER_CUBIC      三次样条插值
cv2.INTER_LANCZOS4   Lanczos插值
```

(10)cv2.resize（）

功能：图像缩放

调用格式：

cv2.resize(InputArray src, OutputArray dst, Size, fx, fy, interpolation)

参数说明：

InputArray src　输入图片

OutputArray dst　输出图片

Size　　输出图片尺寸

fx, fy　　沿x轴，y轴的缩放系数

interpolation 插入方式

```
interpolation 选项所用的插值方法：
INTER_NEAREST    最近邻插值
INTER_LINEAR     双线性插值（默认设置）
INTER_AREA       使用像素区域关系进行重采样。
INTER_CUBIC      4x4像素邻域的双三次插值
INTER_LANCZOS4    8x8像素邻域的Lanczos插值
```

(11)cv2.flip（）

功能：使图像进行翻转

调用格式：

cv2.flip(src, flipcode)

第一个参数代表需要操作的图像

第二个参数定义翻转方式，具体为

flipcode=1　　水平翻转

flipcode=0　　垂直翻转

flipcode=-1　　水平垂直翻转

(12)cv2.hconcat（）

功能：水平方向拼接

调用格式：

cv2.hconcat([image1,image2])

image1,image2指需要进行拼接的图片

(13)cv2.vconcat（）

功能：垂直方向拼接

调用格式：

cv2.vconcat([image1,image2])

四、参考代码

```
#部分代码需要根据实际情况更改
#红绿通道互换
import cv2

image = cv2.imread('E:\\pictures\\peppers.jpg')
#image = cv2.imread('1.jpg')
image1 = image.copy()
image[:,:,1] = image1[:,:,2]
image[:,:,2] = image1[:,:,1]
cv2.namedWindow('changed picture',cv2.WINDOW_NORMAL)
cv2.imshow('changed picture',image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
#灰度化
import cv2

image = cv2.imread('E:\\pictures\\peppers.jpg')
gray = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
cv2.namedWindow('rgb to gray', cv2.WINDOW_NORMAL)
cv2.imshow('rgb to gray',gray)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
#图像旋转
import cv2

image = cv2.imread('E:\\pictures\\peppers.jpg',0)
imageInformation = image.shape
height = imageInformation[0]
width = imageInformation[1]
M = cv2.getRotationMatrix2D((height*0.5,width*0.5),15,0.8)
# 这里有三个参数 分别是中心位置，旋转角度，缩放程度
dst = cv2.warpAffine(image,M,(height,width),cv2.INTER_NEAREST)
cv2.namedWindow('rotate',cv2.WINDOW_NORMAL)
cv2.imshow("rotate",dst)
cv2.waitKey(0)
```

```
-1
```

```
#图像缩放（缩放到原来的2.5倍）
import cv2

image = cv2.imread('E:\\pictures\\peppers.jpg',0)
imageInformation = image.shape
height = imageInformation[0]
width = imageInformation[1]
image1 = cv2.resize(image,(int(height*2.5),int(width*2.5)))
cv2.namedWindow('result',cv2.WINDOW_NORMAL)
cv2.imshow("result",image1)
cv2.waitKey(0)
```

```
-1
```

```
#最近邻插值法缩放（缩放到原来的3倍）
import cv2

image = cv2.imread('E:\\pictures\\peppers.jpg',0)
imageInformation = image.shape
height = imageInformation[0]
width = imageInformation[1]
image1 = cv2.resize(image,(0,0),fx=3,fy=3,interpolation = cv2.INTER_NEAREST)
cv2.namedWindow('result',cv2.WINDOW_NORMAL)
cv2.imshow("result",image1)
cv2.waitKey(0)
```

```
#图像镜像与拼接
import cv2

image = cv2.imread('E:\\pictures\\lotus.bmp')
flip_horizontal = cv2.flip(image , 1)
flip_vertical = cv2.flip(image , 0)
flip_hv = cv2.flip(image , -1)
image1 = cv2.hconcat([image , flip_horizontal])
image2 = cv2.hconcat([flip_vertical , flip_hv])
image3 = cv2.vconcat([image1 , image2])
cv2.namedWindow('result',cv2.WINDOW_NORMAL)
cv2.imshow('result',image3)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

部分程序运行结果如下：

五、实验要求

1.熟悉图像处理相关函数，读懂参考代码；
2.尝试进行单步调试，查看每一步运行的结果；
3.试着将实验中的图片换成别的图片看实验效果；
4.图像插值时尝试其他插值方法的效果；
5.提前编写程序实现实验内容中未提供代码的要求功能；
6.实验报告中给出peppers.jpg 、lotus.bmp的分辨率

六、实验考核

预习准备情况（1/5）+现场考核（3/5）+实验报告（1/5）

七、实验报告

实验结束后，撰写实验报告，实验报告主题部分应包括：算法原理、程序流程、算法各部分主要函数代码以及功能注释、运行结果四部分，按照各部分撰写情况打分。

实验二 图像增强

一、实验意义及目的

（1）进一步掌握图像处理工具，熟悉相关软件的的图像处理函数。
（2）掌握各种图像增强方法。

二、实验内容

打开一幅彩色图像Image1，使用图像处理函数，对其进行下列变换：
（1）将Image1灰度化为gray，统计并显示其灰度直方图；
（2）对gray进行分段线性变换；
（3）对gray进行直方图均衡化；
（4）对gray进行伪彩色增强；
（5）对gray添加噪声并平滑；
（6）对gray利用Sobel算子锐化。
（7）对gray使用canny算子、形态学梯度进行锐化。（未提供代码）
（8）对gray使用傅里叶变换在频域里进行低通、高通滤波实现平滑与锐化。（未提供代码）

三、相关函数介绍

（1）cv2.calcHist（）
功能：统计图像的直方图
调用格式：
cv2.calcHist(images, channels, mask, histSize, ranges,hist，accumulate)
参数说明：
images　输入图像
channels 计算直方图的通道。
Mask　一般用None，表示处理整幅图像。
histSize 表示这个直方图分成多少份（即多少个直方柱）。
ranges　直方图中各个像素的值的范围

最后是两个可选参数，由于直方图作为函数结果返回了，所以第六个hist就没有意义了（待确定） 最后一个accumulate是一个布尔值，用来表示直方图是　否叠加。

（2）cv2.equalizeHist（）
功能：进行直方图均衡化
调用格式：
cv2.equalizeHist(src,dst)

（3）cv2.applyColorMap（）
功能：调用伪彩色模式
调用格式：
cv2.applyColorMap(src, colormap，dst)

（4）cv2.medianBlur（）
功能：对图像进行中值滤波
调用格式：
cv2.medianBlur（src,ksize,dst）

（5）cv2.GaussianBlur（）
功能：对图像进行高斯滤波
调用格式：
cv2.GaussianBlur(src, ksize, sigmaX, dst,sigmaY, borderType)
src　　输入图像
dst　　输出图像
ksize　　高斯内核大小
sigmaX　　沿X轴（水平方向）的内核标准偏差。
sigmaY　　沿Y轴（垂直方向）的内核标准偏差。 如果sigmaY = 0，则将sigmaX值用于sigmaY
borderType　在将内核应用于图像边界时指定图像边界。 可能的值是cv.BORDER_CONSTANT cv.BORDER_REPLICATE cv.BORDER_REFLECT cv.BORDER_WRAP cv.BORDER_REFLECT_101 cv.BORDER_TRANSPARENT cv.BORDER_REFLECT101 cv.BORDER_DEFAULT cv.BORDER_ISOLATED

（6）cv2.Sobel（）
功能：利用Sobel算子进行图像梯度计算
调用格式：
cv2.Sobel(src,ddepth,dx,dy,ksize,scale,delta,borderType)
参数说明：
src　　输入图像
ddepth　输出图像的深度（可以理解为数据类型），-1表示与原图像相同的深度
dx,dy　当dx=1,dy=0时，求的是x方向的一阶导数；当dx=0,dy=1时，求的是y方向的一阶导数（如果同时为1，通常得不到想要的结果）
ksize　Sobel算子的大小，必须是1,3,5或者7,默认为3
scale　将梯度计算得到的数值放大的比例系数，效果通常使梯度图更亮，默认为1
delta　在将目标图像存储进多维数组前，可以将每个像素值增加delta，默认为0
borderType　决定图像在进行滤波操作时边沿像素的处理方式，默认为BORDER_DEFAULT

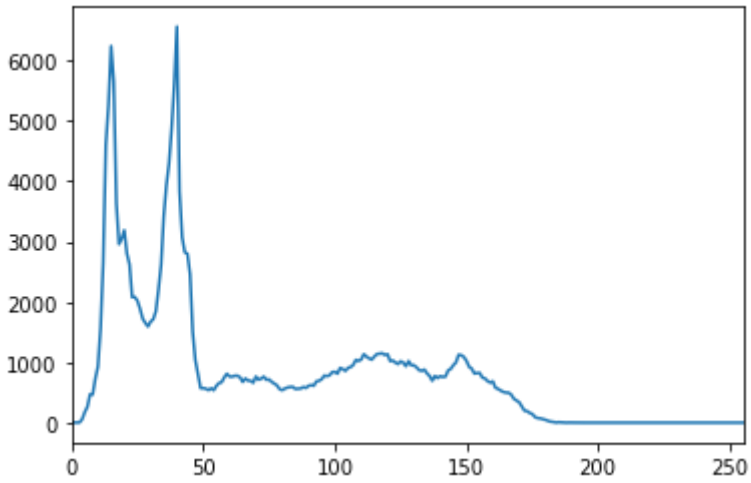
（7）cv2.convertScaleAbs（）
功能：实现图像深度的转换
调用格式：
cv2.convertScaleAbs（src,dst,alpha,beta）
其中，alpha表示比例因子，beta为保存新图像（数组）前可以增加的值

（8）cv2.addWeighted（）

功能：实现图像的加权和融合
调用格式：
cv2.addWeighted(src1, alpha, src2, beta, gamma, dst=None, dtype=None)
参数说明：
src1, src2　需要融合相加的两幅大小和通道数相等的图像
alpha　src1的权重
beta　src2的权重
gamma　修正系数，不需要修正设置为0
dst　可选参数，输出结果保存的变量，默认值为None，如果为非None，输出图像保存到dst对应实参中，其大小和通道数与输入图像相同，　图像的深度由dtype参数或输入图像确认
dtype　可选参数，输出图像数组的深度，即图像单个像素值的位数（如RGB用三个字节表示，则为24位），选默认值None表示与源图像保持一致

四、参考代码

```
#灰度直方图
import cv2
from matplotlib import pyplot as plt
import numpy as np


image = cv2.imread('E:\\pictures\\lotus.bmp',0)
hist = cv2.calcHist([image], [0], None, [256], [0, 255])
plt.plot(hist)
plt.xlim([0, 255])
plt.show()
```




```
#分段线性变换
import cv2
import numpy as np

image = cv2.imread('E:\\pictures\\lotus.bmp',0)
h,w = image.shape[:2]
out = np.zeros(image.shape, np.uint8)
for i in range(h):
```

```
    for j in range(w):
        pix = image[i][j]
        if pix < 10:
            out[i][j] = 0
        elif pix < 180:
            out[i][j] = pix
        else:
            out[i][j] = 255

cv2.namedWindow('result', 0)
cv2.imshow('result' , out)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
#直方图均衡化
import cv2

image = cv2.imread('E:\\pictures\\lotus.bmp',0)
equalizedimage = cv2.equalizeHist(image)
cv2.namedWindow('equalizedimage', cv2.WINDOW_NORMAL)
cv2.imshow('equalizedimage', equalizedimage)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
#伪彩色增强
import cv2

image = cv2.imread('E:\\pictures\\lotus.bmp',0)
color_image = cv2.applyColorMap(image, cv2.COLORMAP_JET)
cv2.imshow("color_image", color_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
#椒盐噪声3*3中值滤波
import cv2
import numpy as np
import random

#椒盐噪声
def sp_noise(image,prob):
    thres = 1 - prob
    output = np.zeros(image.shape, np.uint8)
    for i in range(image.shape[0]):
        for j in range(image.shape[1]):
            rdn = random.random()#生成0-1之间的随机值
            if rdn < prob:
                output[i][j] = 0
            elif rdn > thres:
                output[i][j] = 1
            else:
                output[i][j] = image[i][j]
    return output
```

```
image = cv2.imread('E:\\pictures\\lotus.bmp',0)
image1 = sp_noise(image, 0.04)
image2 = cv2.medianBlur(image1, 3)
image3 = cv2.hconcat([image, image1])
cv2.namedWindow('result', cv2.WINDOW_NORMAL)
cv2.imshow('result', image3)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
#高斯滤波
import cv2
import numpy as np
import random

def gaussian_noise(image,mean,var):
    image = np.array(image/255,dtype=float)
    noise = np.random.normal(mean,var**0.5,image.shape)
    out = image+noise
    if out.min() < 0:
        low_clip = -1
    else:
        low_clip = 0
    out = np.clip(out,low_clip,1.0)
    out = np.uint8(out * 255)
    return  out

image = cv2.imread('E:\\pictures\\lotus.bmp',0)
out1 = gaussian_noise(image,0,0.01)
gaussianblurimage = cv2.GaussianBlur(out1, (3,3) , 3,3)
final = cv2.hconcat([gaussianblurimage,out1])
cv2.namedWindow('result', cv2.WINDOW_NORMAL)
cv2.imshow('result', final)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
#Sobel梯度图像
import cv2

image = cv2.imread('E:\\pictures\\lotus.bmp',0)
sobel_x = cv2.Sobel(image, cv2.CV_64F, 1, 0, 3)
sobel_y = cv2.Sobel(image, cv2.CV_64F, 0, 1, 3)
sobel_x = cv2.convertScaleAbs(sobel_x)
sobel_y = cv2.convertScaleAbs(sobel_y)
sobel_xy = cv2.addWeighted(sobel_x, 0.5, sobel_y, 0.5, 0)
cv2.namedWindow('result', cv2.WINDOW_NORMAL)
cv2.imshow('result', sobel_xy)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
#Sobel锐化图像
import cv2
```

```
image = cv2.imread('E:\\pictures\\lotus.bmp',0)
sobel_x = cv2.Sobel(image, cv2.CV_64F, 1, 0, 3)
sobel_y = cv2.Sobel(image, cv2.CV_64F, 0, 1, 3)
sobel_x = cv2.convertScaleAbs(sobel_x)
sobel_y = cv2.convertScaleAbs(sobel_y)
sobel_xy = cv2.addWeighted(sobel_x, 0.5, sobel_y, 0.5, 0)
result = image+sobel_xy
cv2.namedWindow('result', cv2.WINDOW_NORMAL)
cv2.imshow('result', result)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

程序的运行效果如下：

五、实验要求

1. 熟悉相关图像处理函数，读懂参考代码； 提前编写程序实现实验内容中未提供代码的要求功能；
2. 在此基础上，变换参数，或设计不同的方法，如更改伪彩色增强方法为热金属编码或彩虹编码，更改噪声参数，设计不同的滤波方法等，充分发挥个人主动性完成实验内容，不能完全照搬参考代码。
3. 使用所学的不同锐化方法锐化图像，并对比效果。

六、实验考核

预习准备情况（1/5）+现场考核（3/5）+实验报告(1/5)

七、实验报告

实验结束后，撰写实验报告，实验报告主题部分应包括：算法原理、程序流程、算法各部分主要函数代码以及功能注释、运行结果四部分，按照各部分撰写情况打分。

实验三 图像分割与描述

一、实验意义及目的

（1） 进一步掌握图像处理工具，熟悉图像处理函数。
（2） 掌握图像分割方法，熟悉常用图像描述方法。

二、实验内容

打开一幅图像Image，使用图像处理函数，对其进行下列变换：
（1）将Image灰度化为gray，对其进行阈值分割转换为BW；
（2）对BW进行数学形态学滤波；
（3）对BW进行边缘跟踪，用红色线在图中标出；

（4）计算各区域边界点的傅里叶描绘子并用四分之一点重建边界;
（5）使用多种方法分割图像（未提供代码）。
（6）选做加分：尝试使用深度学习的方法进行图像分割；（未提供代码）

三、相关函数介绍

（1）cv2.threshold（）
功能：对图像进行阈值处理
调用格式：
cv2.threshold (src, thresh, maxval, type)
src 源图片，必须是单通道
thresh 阈值，取值范围0～255
maxval 填充色，取值范围0～255
type 阈值类型，主要是以下5种

```
cv2.THRESH_BINARY        二进制阈值化，非黑即白
cv2.THRESH_BINARY_INV    反二进制阈值化，非白即黑
cv2.THRESH_TRUNC         截断阈值化 ，大于阈值设为阈值
cv2.THRESH_TOZERO        阈值化为0 ，小于阈值设为0
cv2.THRESH_TOZERO_INV    反阈值化为0 ，大于阈值设为0
```

（2）cv2.getStructuringElement（）
功能：返回指定形状和尺寸的结构元素
调用格式：
cv2.getStructuringElement（shape,ksize,anchor）
shape表示内核的形状，有三种形状可以选择
矩形：MORPH_RECT;交叉形：MORPH_CROSS;椭圆形：MORPH_ELLIPSE;
第二和第三个参数分别是内核的尺寸以及锚点的位置。对于锚点的位置，有默认值Point（-1,-1），表示锚点位于中心点。

（3）cv2.morphologyEx（）
功能：进行各类形态学的变化
调用格式：
cv2.morphologyEx(src, op, kernel)
第二个参数主要有5种选择
cv2.MORPH_OPEN 开运算 ,先腐蚀后膨胀的过程。开运算能够除去孤立的小点，毛刺和小桥，而总的位置和形状不便
cv2.MORPH_CLOSE 闭运算，先膨胀后腐蚀的过程。闭运算能够填平小孔，弥合小裂缝，而总的位置和形状不变。
cv2.MORPH_GRADIENT 形态学梯度，能够突出团块的边缘，保留物体的边缘轮廓。
cv2.MORPH_TOPHAT 顶帽，将突出比原轮廓亮的部分。
cv2.MORPH_BLACKHAT 黑帽，将突出比原轮廓暗的部分。

(4)cv2.findContours()
功能：寻找图像轮廓
调用格式：
cv2.findContours（image,mode,method）
mode表示轮廓检索模式，有4种选择
RETR_EXTERNAL 只检测外轮廓
RETR_LIST 检测的轮廓不建立等级关系

RETR_CCOMP　建立两个等级的轮廓，上面的一层为外边界，里面的一层为内孔的边界信息。如果内孔内还有一个连通物体，这个物体的边界也在顶　　层。
RETR_TREE　建立一个等级树结构的轮廓

method表示轮廓逼近方法，有3种选择
cv2.CHAIN_APPROX_NONE　存储轮廓所有点的信息，相邻两个轮廓点在图象上也是相邻的；
cv2.CHAIN_APPROX_SIMPLE　压缩水平方向，垂直方向，对角线方向的元素，只保留该方向的终点坐标；
cv2.CHAIN_APPROX_TC89_L1　使用teh-Chinl chain 近似算法保存轮廓信息。

（5）cv2.drawContours（）
功能：绘制图像轮廓
调用格式：
cv2.drawContours(image, contours, contourIdx, color, thickness=None, lineType=None)
第一个参数image表示目标图像，
第二个参数contours表示输入的轮廓组，每一组轮廓由点vector构成，
第三个参数contourIdx指明画第几个轮廓，如果该参数为负值，则画全部轮廓，
第四个参数color为轮廓的颜色，
第五个参数thickness为轮廓的线宽，如果为负值或CV_FILLED表示填充轮廓内部，
第六个参数lineType为线型，

（6）np.fft.fft（）
功能：实现快速傅里叶变换
调用格式：
np.fft.fft（signal）

四、参考代码

```
#二值化图像
import cv2

image = cv2.imread('E:\\pictures\\plane.jpg', 0)
ret,BWimage = cv2.threshold(image, 127, 255, cv2.THRESH_BINARY)
cv2.namedWindow('result', cv2.WINDOW_NORMAL)
cv2.imshow('result', BWimage)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
#形态学滤波
import cv2

image = cv2.imread('E:\\pictures\\plane.jpg', 0)
ret, BWimage = cv2.threshold(image, 127, 255, cv2.THRESH_BINARY)
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))
LBimage1 = cv2.morphologyEx(BWimage, cv2.MORPH_OPEN, kernel)
LBimage2 = cv2.morphologyEx(BWimage, cv2.MORPH_CLOSE, kernel)
LBimage3 = cv2.morphologyEx(BWimage, cv2.MORPH_GRADIENT, kernel)
LBimage4 = cv2.morphologyEx(BWimage, cv2.MORPH_TOPHAT, kernel)
LBimage5 = cv2.morphologyEx(BWimage, cv2.MORPH_BLACKHAT, kernel)
A = cv2.hconcat([LBimage1, LBimage2, LBimage3 ])
B = cv2.hconcat([LBimage4, LBimage5,  BWimage])
C =cv2.vconcat([A, B])
cv2.namedWindow('result', cv2.WINDOW_NORMAL)
```

```
cv2.imshow('result', C)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
#绘制轮廓
import cv2

image = cv2.imread('E:\\pictures\\plane.jpg')
image1 = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
ret, BWimage = cv2.threshold(image1, 127, 255, cv2.THRESH_BINARY_INV)
contours, hierarchy = cv2.findContours(BWimage, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_NONE)
cv2.drawContours(image, contours, -1, (0, 0, 255), 1)
cv2.namedWindow('result',  cv2.WINDOW_NORMAL)
cv2.imshow('result', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
#重建的图像
import cv2
import numpy as np

gray = cv2.imread('E:\\pictures\\plane.jpg',0)
image = cv2.imread('E:\\pictures\\plane.jpg')
ret, BWimage = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY_INV)
contours, hierarchy = cv2.findContours(BWimage, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_NONE)
contour_array = contours[0]
contours_complex = np.empty(contour_array.shape[:-1], dtype=complex)
contours_complex.real = contour_array[:,:,0]
contours_complex.imag = contour_array[:,:,1]

for i in range(1,len(contours_complex.real),4):
    contours_complex.real[i,0] = contour_array[i,:,0]

for j in range(1,len(contours_complex.imag),4):
    contours_complex.imag[j,0] = contour_array[j,:,1]

fourier_result = np.fft.fft(contours_complex)
contour_reconstruct = np.fft.ifft(fourier_result)
contour_reconstruct =
np.array([contour_reconstruct.real,contour_reconstruct.imag])
contour_reconstruct = np.transpose(contour_reconstruct)
contour_reconstruct = np.expand_dims(contour_reconstruct, axis = 1)
cv2.drawContours(image,contour_reconstruct.astype(int),-1, (0, 0, 255), 1)
cv2.namedWindow("contour_reconstruct",0)
cv2.imshow("contour_reconstruct", image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

程序运行结果如下：

五、实验要求

1.熟悉图像处理相关函数，读懂参考代码；
2.在此基础上，变换参数，或设计不同的方法，如设计不同的滤波方法等，充分发挥个人主动性完成实验内容，不能完全照搬参考代码。
3.使用三种及以上方法分割图像，并对比效果。

六、实验考核

预习准备情况（1/5）+现场考核（3/5）+实验报告(1/5)

七、实验报告

实验结束后，撰写实验报告，实验报告主题部分应包括：算法原理、程序流程、算法各部分主要函数代码以及功能注释、运行结果四部分，按照各部分撰写情况打分。

实验四 图像综合处理小设计实验

一、实验意义及目的

充分利用所学各种图像处理技术，实现对图像的综合处理，加深对基础知识的理解和应用。

二、实验内容

在下列题目中任意选择3个题目完成实验，提前预先编好程序。建议实验时，自己收集与提供的图片类似图片测试所编程序的效果，效果不佳再改进程序。
1．机器视觉图像的目标与背景的分割与提取主要要求：对输入图像可以达到目标和背景的分割
建议方法：
（1） 将已知图像进行消噪处理；
（2） 对彩色图像进行目标和背景分析；
（3） 通过阈值法将图像进行分割；
（4） 确定目标的位置。
使用上述方法完成功能后，可尝试使用深度学习的方案；

2.基于Sobel算子完成对图像的搜索
主要要求：从图像中检索出飞机图像，要求五架飞机全部找到。
建议方法：
（1） 对待检测图片进行预处理（灰度化、二值化）；
（2） 对图像进行边缘提取；
（3） 改进算子，使图像达到标准对照图像效果。
使用上述方法完成功能后，可尝试使用深度学习的方案；

3. 硬币检测及计数
主要要求：白色背景上扔几枚不同面值的硬币，拍摄图像，通过图像处理，获取硬币的数目及金额。
建议方法：
（1） 图像分割；
（2） 边缘检测、滤波去燥；
（3） 连通区域检测，判断硬币个数；

（4） 边缘图像通过霍夫变换检测圆，进而检测硬币金额。
使用上述方法完成功能后，可尝试使用深度学习的方案；

4. 基于彩色阈值变换的香蕉彩色图像分割
主要要求：利用香蕉和其它水果及其背景颜色在R、G、B分量上的差异进行识别,根据香蕉和其它水果在R、G、B分量的二值化处理，获得特征提取的有效区域，然后提取特征，达到提取香蕉的目的。
建议方法：
（1） 分别截取图像中香蕉和其它水果的感兴趣区域，并将其转换为R分量、G分量、B分量的灰度图像对各个分量进行灰度值统计，得到灰度直方图；
（2） 设定阈值对其进行二值化；
（3） 特征提取。根据香蕉、草莓、奇异果的特征统计出各自的个数。 (图片可自己在网上下载符合条件的)
使用上述方法完成功能后，可尝试使用深度学习的方案；

三、实验要求

提前预先编好程序，实验完成时教师会提问问题。每个题目使用多种方法完成任务。

四、实验考核

实验预习实现（2/5）+现场考核（2/5）+实验报告(1/5)

五、实验报告

实验结束后，撰写实验报告，实验报告主题部分应包括：算法原理、程序流程、算法各部分主要函数代码以及功能注释、运行结果四部分， 按照各部分撰写情况打分。

实验五 九宫格人车识别

一、实验意义及目的

充分利用所学各种图像处理技术，实现对图像的综合处理，加深对基础知识的理解和应用。

二、实验内容

至少使用2种方法，自动识别下列九宫格图像中小人的位置，并将小人分割出来。

建议方法：
1）将图像二值化后，提取分割九个宫格九宫格图像，根据小人和小车黑色像素点的不同识别出小人的图像；
2）使用角点特征来匹配识别小人，再分割确定其位置；
3）使用目标检测的方法寻找小人，再分割确定其位置；

三、实验要求

提前预先编好程序，实验完成时教师会提问问题，使用多种方法完成任务并对比效果。

四、实验考核

实验预习实现（2/5）+现场考核（2/5）+实验报告(1/5).

五、实验报告

实验结束后，撰写实验报告，实验报告主题部分应包括：算法原理、程序流程、算法各部分主要函数代码以及功能注释、运行结果四部分，按照撰写情况打分。

实验六 手写数字或服饰识别

一、实验意义及目的

了解Pytorch深度学习框架，充分利用所学各种图像处理技术，实现对图像的综合处理，加深对基础知识的理解和应用。

二、实验内容

搭建简单神经网络，对MNIST或Fashion-MNIST数据集进行训练并进行测试识别。

建议方法：
基本神经网络方法，卷积神经网络方法

三、实验要求

（1）了解相关pytorch函数，读懂CNN识别与神经网络识别的参考代码；
（2）给代码加上注释，若由于环境等问题不能直接运行，进行相关配置与解决；
（3）参考代码进行代码补全；
（4）尝试修改lr,batch_size等参数，观察不同训练效果
（5）尝试自己学习并使用其他损失函数和优化器，观察不同训练效果
（6）尝试自己添加一些网络层，观察不同训练效果

四、参考代码

```
#导入相关库
import torch
from torch import nn
from torch.utils.data import DataLoader
from torchvision import datasets
from torchvision.transforms import ToTensor
import matplotlib.pyplot as plt
import numpy as np
import os
import torch.nn.functional as F
import torch.optim as optim
```

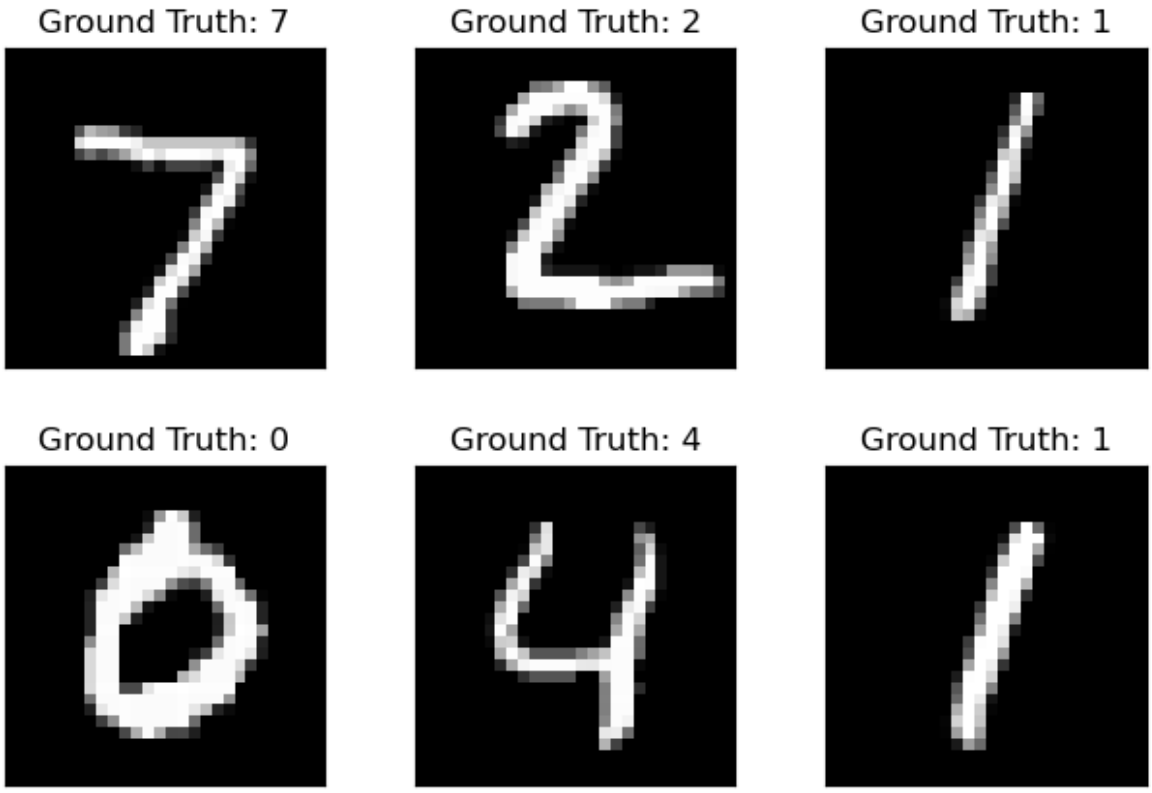
```
#参数定义
#dataset = "FashionMNIST" #选择所使用的数据集"MNIST"or"FashionMNIST"
dataset = "MNIST"
#Net = "LeNet5" #选择所使用的网络"LeNet5"or "LinearNet"
Net = "LinearNet" #选择所使用的网络"LeNet5"or "LinearNet"
batch_size = 64 #定义批处理大小
lr=0.01 #设置learning rate学习率
epochs = 5 #指定训练迭代次数
save_path = "./" #模型保存路径
Early_Stopping = 0 #选择是否使用Early Stopping训练模式，训练时根据精度的变化率来控制训练迭代代数
```

```
#下载数据集
if dataset=="MNIST":
    # 从torchvision下载训练集.
    training_data = datasets.MNIST(
        root="data",
        train=True,
        download=True,
        transform=ToTensor(),
    )

    # #从torchvision下载测试集.
    test_data = datasets.MNIST(
        root="data",
        train=False,
        download=True,
        transform=ToTensor(),
    )
    classes = [ "0", "1", "2", "3", "4", "5", "6", "7", "8", "9"]
else:
    # 从torchvision下载训练集.
    training_data = datasets.FashionMNIST(
        root="data",
        train=True,
        download=True,
        transform=ToTensor(),
    )

    # #从torchvision下载测试集.
    test_data = datasets.FashionMNIST(
        root="data",
        train=False,
        download=True,
        transform=ToTensor(),
    )
    classes = [ "T-shirt/top", "Trouser", "Pullover", "Dress", "Coat",
"Sandal","Shirt", "Sneaker", "Bag", "Ankle boot"]
```

```
D:\ProgramData\Anaconda3\envs\pytorch\lib\site-
packages\torchvision\datasets\mnist.py:498: UserWarning: The given NumPy array is
not writeable, and PyTorch does not support non-writeable tensors. This means you
can write to the underlying (supposedly non-writeable) NumPy array using the
tensor. You may want to copy the array to protect its data or make it writeable
before converting it to a tensor. This type of warning will be suppressed for the
rest of this program. (Triggered internally at
..\torch\csrc\utils\tensor_numpy.cpp:180.)
  return torch.from_numpy(parsed.astype(m[2], copy=False)).view(*s)
```

```
# 创建数据加载器.
train_dataloader = DataLoader(training_data, batch_size=batch_size)
test_dataloader = DataLoader(test_data, batch_size=batch_size)

#观察数据样本
for X, y in test_dataloader:
    print(f"Shape of X [N, C, H, W]: {X.shape}")
    print(f"Shape of y: {y.shape} {y.dtype}")
    fig = plt.figure()
    for i in range(6):
      plt.subplot(2,3,i+1)
      plt.tight_layout()
      plt.imshow(X[i][0], cmap='gray', interpolation='none')
      plt.title("Ground Truth: {}".format(classes[int(y[i])]))
      plt.xticks([])
      plt.yticks([])
    plt.show()
    break
```

```
Shape of X [N, C, H, W]: torch.Size([64, 1, 28, 28])
Shape of y: torch.Size([64]) torch.int64
```




```
# 自动选择cpu或gpu用于训练.
device = (
    "cuda"
    if torch.cuda.is_available()
    else "cpu"
)
print(f"Using {device} device")
```

```
Using cpu device
```

```
#定义模型

if Net == "LinearNet":
    class LinearNet(nn.Module):
        def __init__(self):
            super().__init__()
            self.flatten = nn.Flatten()
            self.linear_relu_stack = nn.Sequential(
                nn.Linear(28*28, 512),
                nn.ReLU(),
                nn.Linear(512, 512),
                nn.ReLU(),
                nn.Linear(512, 10)
            )

        def forward(self, x):
```

```
            x = self.flatten(x)
            logits = self.linear_relu_stack(x)
            return logits

    model = LinearNet().to(device)

else:
    class CNN_1(nn.Module):
        def __init__(self):
            super(CNN_1, self).__init__()
                nn.Conv2d(
                ## 卷积核大小为羽3，输入通道为__，输出通道为13，步长为1;
                ## paddingHzero padding
                ##要求经过conv1的输出空间维度与输入的空间维度相同
                in_channels =       ,
                out_channels =          ,
                kernels_size =           ,
                stride=                 ,
                padding=            ,
            )
            ##激活函数+最大池化
            ,
            ,

            self.conv2d = nn.Sequential(
            ## 1. 卷积核大小为3*3，输入通道为10，输出通道为10，padding 方法为same，
padding大小为？？？步长为??
            ## 2.自行选择激活函数
            ## 3. 池化




            )
            ## 添加残差连接模块
            ##############残差模块设计部分################




            ##############################################
            self.conv3 = nn.Sequential(
                ##自行设计卷积模块


            )
            self.out1 = nn.Linear(3*3*29, 10, bias = True)
            ## 在下方添加Dropout以及其他代码


        def forward(self, x):
            ## 请将余下代码补充完整


    model = CNN_1().to(device)
```

```
print(model)
```

```
LinearNet(
  (flatten): Flatten(start_dim=1, end_dim=-1)
  (linear_relu_stack): Sequential(
    (0): Linear(in_features=784, out_features=512, bias=True)
    (1): ReLU()
    (2): Linear(in_features=512, out_features=512, bias=True)
    (3): ReLU()
    (4): Linear(in_features=512, out_features=10, bias=True)
  )
)
```

```
#为了训练模型，我们需要一个损失函数和一个优化器
loss_fn = nn.CrossEntropyLoss()#定义损失函数
optimizer = torch.optim.???????(model.parameters(), ????????)#定义优化器

#在单个训练循环中，模型对训练数据集进行预测（分批提供给它），并反向传播预测误差以调整模型的参数
def train(dataloader, model, loss_fn, optimizer):
    size = len(dataloader.dataset)
    model.train()
    for batch, (X, y) in enumerate(dataloader):
        X, y = X.to(device), y.to(device)

        # Compute prediction error
        pred = model(X)
        loss = loss_fn(pred, y)

        # Backpropagation
        loss.backward()
        optimizer.step()
        optimizer.zero_grad()

        if batch % 100 == 0:
            loss, current = loss.item(), (batch + 1) * len(X)
            print(f"loss: {loss:>7f}  [{current:>5d}/{size:>5d}]")

#对照测试数据集检查模型的性能，以确保它正在学习
def test(dataloader, model, loss_fn):
    size = len(dataloader.dataset)
    num_batches = len(dataloader)
    model.eval()
    test_loss, correct = 0, 0
    with torch.no_grad():
        for X, y in dataloader:
            X, y = X.to(device), y.to(device)
            pred = model(X)
            test_loss += loss_fn(pred, y).item()
            correct += (pred.argmax(1) == y).type(torch.float).sum().item()
    test_loss /= num_batches
```

```
    correct /= size
    print(f"Test Error: \n Accuracy: {(100*correct):>0.1f}%, Avg loss:
{test_loss:>8f} \n")
    return test_loss

#实现Early Stopping训练
class EarlyStopping:
    def __init__(self, save_path, patience=2, verbose=False, delta=0.03):
        """
        Args:
            save_path : 模型保存路径
            patience (int): 设置将连续几次训练迭代纳入Early Stopping考评
            verbose (bool): 如果是 "True"，则为每次验证损失的优化值打印一条信息
            delta (float): 前后两次训练迭代的最小变化阈值，小于该阈值则认为模型优化幅度有
限，将该次迭代计入patience，
                           数量达到patience则提前停止训练。
        """
        self.save_path = save_path
        self.patience = patience
        self.verbose = verbose
        self.counter = 0
        self.best_score = None
        self.early_stop = False
        self.val_loss_min = np.Inf
        self.delta = delta

    def __call__(self, val_loss, model):

        score = -val_loss

        if self.best_score is None:
            self.best_score = score
            self.save_checkpoint(val_loss, model)
        elif score < self.best_score + self.delta:
            self.counter += 1
            print(f'EarlyStopping counter: {self.counter} out of
{self.patience}')
            if self.counter >= self.patience:
                self.early_stop = True
        else:
            self.best_score = score
            self.save_checkpoint(val_loss, model)
            self.counter = 0

    def save_checkpoint(self, val_loss, model):
        '''当验证损失降低时，保存模型'''
        if self.verbose:
            print(f'Validation loss decreased ({self.val_loss_min:.6f} -->
{val_loss:.6f}).  Saving model ...')
        path = os.path.join(self.save_path, 'best_network.pth')
        torch.save(model.state_dict(), path)    # 这里会存储迄今最优模型的参数
        self.val_loss_min = val_loss
```

```python
#训练过程是在所定义的几个迭代上进行的。在每次迭代，模型学习参数以做出更好的预测。我们在每次迭代打印模型的准确性和损失；我们希望看到精度随着迭代次数的增加而增加，而损失随着迭代次数的减少而减少
early_stopping = EarlyStopping(save_path)
for t in range(epochs):
    print(f"Epoch {t+1}\n-------------------------------")
    train(train_dataloader, model, loss_fn, optimizer)
    test_loss = test(test_dataloader, model, loss_fn)
    if Early_Stopping:
        early_stopping(test_loss, model)
        if early_stopping.early_stop:
            print("Early stopping")
            break #跳出迭代，结束训练
print("Done!")
```

```
Epoch 1
-------------------------------
loss: 2.310157  [   64/60000]
loss: 2.255705  [ 6464/60000]
loss: 2.219243  [12864/60000]
loss: 2.015800  [19264/60000]
loss: 1.852811  [25664/60000]
loss: 1.544494  [32064/60000]
loss: 1.164786  [38464/60000]
loss: 1.102608  [44864/60000]
loss: 0.855882  [51264/60000]
loss: 0.716411  [57664/60000]
Test Error:
 Accuracy: 83.3%, Avg loss: 0.661168

Epoch 2
-------------------------------
loss: 0.739424  [   64/60000]
loss: 0.552977  [ 6464/60000]
loss: 0.556154  [12864/60000]
loss: 0.530490  [19264/60000]
loss: 0.468894  [25664/60000]
loss: 0.451964  [32064/60000]
loss: 0.342593  [38464/60000]
loss: 0.503526  [44864/60000]
loss: 0.463082  [51264/60000]
loss: 0.470099  [57664/60000]
Test Error:
 Accuracy: 88.9%, Avg loss: 0.391432

Epoch 3
-------------------------------
loss: 0.417350  [   64/60000]
loss: 0.315711  [ 6464/60000]
loss: 0.325021  [12864/60000]
loss: 0.406835  [19264/60000]
loss: 0.326629  [25664/60000]
loss: 0.374817  [32064/60000]
loss: 0.239529  [38464/60000]
loss: 0.420350  [44864/60000]
loss: 0.383979  [51264/60000]
```

```
loss: 0.425964  [57664/60000]
Test Error:
 Accuracy: 90.5%, Avg loss: 0.330049

Epoch 4
-------------------------------
loss: 0.316552  [   64/60000]
loss: 0.268695  [ 6464/60000]
loss: 0.246773  [12864/60000]
loss: 0.371894  [19264/60000]
loss: 0.275253  [25664/60000]
loss: 0.335943  [32064/60000]
loss: 0.203020  [38464/60000]
loss: 0.382334  [44864/60000]
loss: 0.338214  [51264/60000]
loss: 0.400558  [57664/60000]
Test Error:
 Accuracy: 91.4%, Avg loss: 0.299417

Epoch 5
-------------------------------
loss: 0.263625  [   64/60000]
loss: 0.250378  [ 6464/60000]
loss: 0.206313  [12864/60000]
loss: 0.352884  [19264/60000]
loss: 0.243308  [25664/60000]
loss: 0.310380  [32064/60000]
loss: 0.183422  [38464/60000]
loss: 0.358683  [44864/60000]
loss: 0.302009  [51264/60000]
loss: 0.379364  [57664/60000]
Test Error:
 Accuracy: 92.0%, Avg loss: 0.278075

Done!
```

```python
#保存模型的一种常见方法是序列化内部状态字典(包含模型参数)
torch.save(model.state_dict(), "model.pth")
print("Saved PyTorch Model State to model.pth")
```

```
Saved PyTorch Model State to model.pth
```

```
#加载模型的过程包括重新创建模型结构并将状态字典加载到其中。
model = LinearNet().to(device) if Net == "LinearNet" else LeNet5().to(device)
model.load_state_dict(torch.load("model.pth"))

model.eval()
x, y = test_data[0][0], test_data[0][1]
with torch.no_grad():
    x = x.to(device)
    pred = model(x)
    predicted, actual = classes[pred[0].argmax(0)], classes[y]
    print(f'Predicted: "{predicted}", Actual: "{actual}"')
```

```
Predicted: "7", Actual: "7"
```

五、实验考核

实验预习实现（2/5）+现场考核（2/5）+实验报告(1/5).

#六、实验报告

实验结束后，撰写实验报告，实验报告主题部分应包括：算法原理、程序流程、算法各部分主要函数代码以及功能注释、运行结果四部分，按照撰写情况打分。