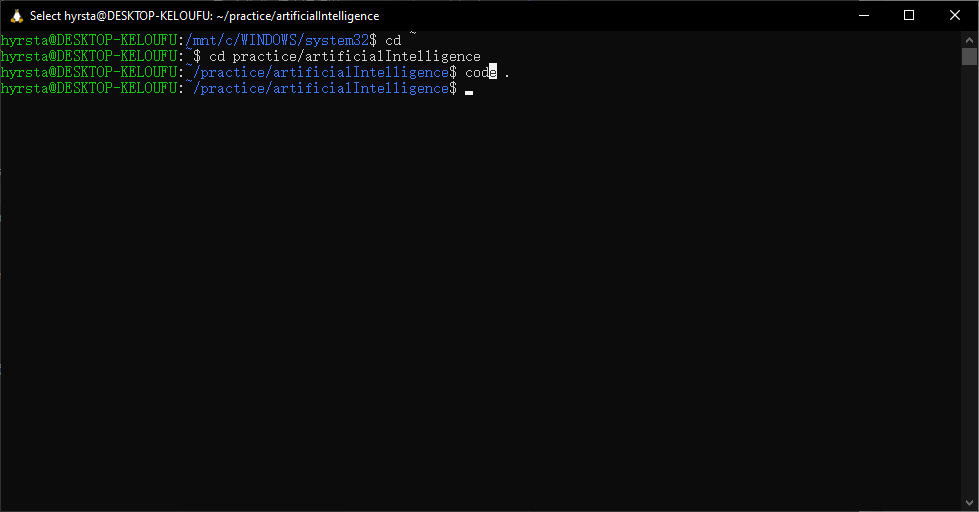
“飞行器图像分类运行说明”

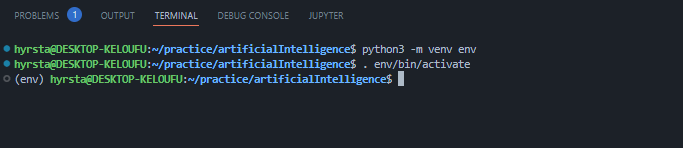
1. SVM分类器运行说明

本大作业我们选择了课程组提供的分类问题，即“飞行器图像分类”。使用WSL1的VSCode作为文本编辑器。

1. 首先在wsl命令行里进入该项目的目录，然后输入“code .”在该目录下打开vscode。

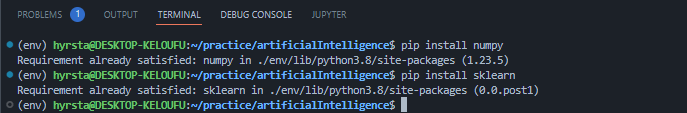


1. 在V.S.Code的终端里输入“python3 -m venv <虚拟环境文件夹名字>”命令创建新的虚拟环境。然后输入“. <虚拟环境文件夹名字>/bin/activate”进入该虚拟环境。



1. 接下来只需要通过pip命令安装对项目有关的库，以下是所有需要安装的库：  
   pip install numpy

pip install sklearn



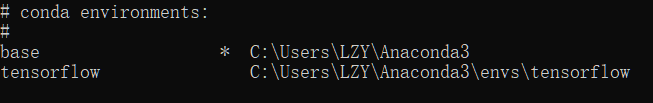
然后把给定的数据集文件夹“飞行器分类数据集”，“project2\_preprocess,ipynb”和“project2\_training.ipynb”移动到VSCode正在运行的目录下。最后按自上而下的顺序运行“project2\_training.ipynb”文件里的所有代码读取并预处理数据集，然后以同样的方式运行“project2\_training.ipynb”文件进行模型的训练和测试。

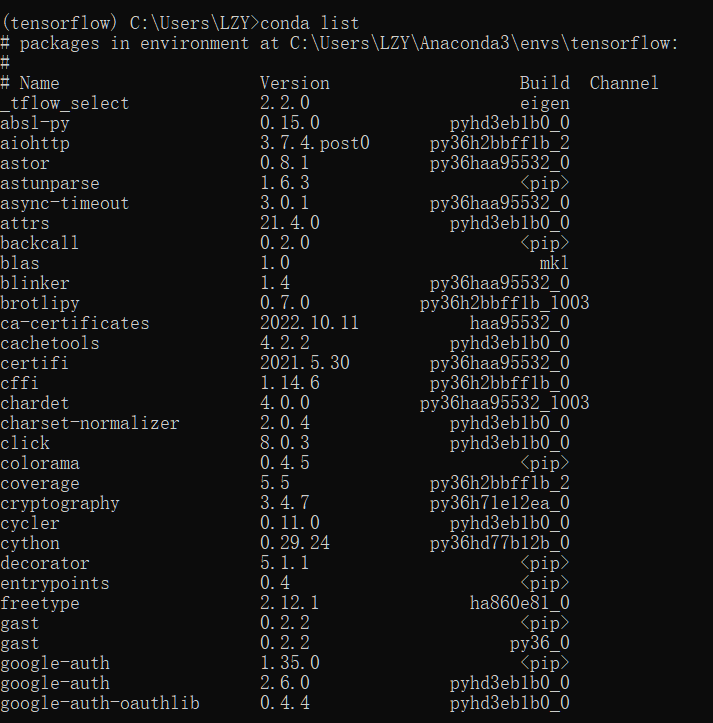
二、CNN分类器运行说明

本次大作业我们选择了课程组提供的分类问题，即“飞行器图像分类”。

使用VSCode作为文本编辑器。

1. 通过Anaconda创建新环境并安装相应的库（通过tensorflow实现CNN）





1. import相关的库

*import* os

*import* glob

*import* cv2

*import* random

*import* numpy *as* np

*import* tensorflow *as* tf

*from* PIL *import* Image

*from* tensorflow.python.keras.layers.convolutional *import* MaxPooling2D, Convolution2D

*from* tensorflow.python.keras.optimizers *import* SGD

*from* sklearn.preprocessing *import* LabelBinarizer

*from* tensorflow.python.keras.layers.core *import* Dense, Dropout, Activation, Flatten

*from* tensorflow.python.keras.models *import* Sequential

*import* argparse

*from* keras *import* regularizers

*from* matplotlib *import* pyplot *as* plt

*from* sklearn.metrics *import* classification\_report

path = "./data/planes/"

3、使用opencv和os遍历图片（注释中本想使用PIL库读取图片解决png无法正常读取问题，后发现RGB通道问题及Convolution2D读取格式问题，选择忽视数据中的png图片），将图片数据，类标，路径分开储存，并将数据/255归一化处理。

*#读取图像*

def read\_img(path):

    cate = [path + x *for* x in os.listdir(path) *if* os.path.isdir(path + x)]

    imgs = []

    labels = []

    fpath = []

*for* idx, folder in enumerate(cate):

*# 遍历整个目录判断每个文件是不是符合*

*for* im in glob.glob(folder + '/\*.jpg'):

            IMG\_SIZE = 224

*# print('reading the images:%s' % (im))*

*try*:

             img = cv2.imread(im)

*# img = Image.open(im).convert("RGB")  # 调用PIL库读取该图像*

*# # 将rgb通道转换为bgr*

*# img = cv2.cvtColor(np.asarray(img), cv2.COLOR\_RGB2BGR)*

*# # print(img.shape)*

             img = cv2.resize(img, (IMG\_SIZE, IMG\_SIZE),interpolation=cv2.INTER\_CUBIC)  *# 图像像素大小一致*

*# print(img.shape)*

             imgs.append(img)  *# 图像数据*

             labels.append(idx)  *# 图像类标*

             fpath.append(path + im)  *# 图像路径名*

*# print(path+im, idx)*

*except*:

*continue*

*return* np.asarray(fpath, np.string\_), np.asarray(imgs, np.float32)/255, np.asarray(labels, np.int32)

1. 使用random.shuffle打乱数据并拆分训练集和测试集为80%和20%，使用LabelBinarizer将类标转换为独热码，使用print进行查看。

*# 读取图像*

fpaths, data, label = read\_img(path)

print(data.shape)  *# (7641, 28, 28, 3)*

*# data=np.transpose(data,(0,3,1,2))*

*# 计算有多少类图片*

num\_classes = len(set(label))

print(num\_classes)

*# 生成等差数列随机调整图像顺序*

num\_example = data.shape[0]

*# print(num\_example)*

arr = np.arange(num\_example)

np.random.shuffle(arr)

data = data[arr]

label = label[arr]

fpaths = fpaths[arr]

*# 拆分训练集和测试集 80%训练集 20%测试集*

ratio = 0.8

s = np.int(num\_example \* ratio)

x\_train = data[:s]

y\_train = label[:s]

fpaths\_train = fpaths[:s]

x\_val = data[s:]

y\_val = label[s:]

fpaths\_test = fpaths[s:]

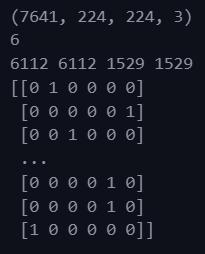
print(len(x\_train),len(y\_train),len(x\_val),len(y\_val))

lb=LabelBinarizer()

y\_train=lb.fit\_transform(y\_train)

y\_val=lb.fit\_transform(y\_val)

print(y\_train)



4、查询相关资料并将简单神经网络与AlexNet模型的准确率进行对比（注释），发现简单神经网络出现了过拟合，准确率低，loss居高不下的问题，在多次改变学习率和epochs无果后，选择了模型更复杂的AlexNet模型。

*#GPU加速*

*import* os

*import* tensorflow *as* tf

os.environ["CUDA\_DEVICES\_ORDER"] = "PCI\_BUS\_IS"

os.environ["CUDA\_VISIBLE\_DEVICES"] = "0"

gpu\_options = tf.GPUOptions(per\_process\_gpu\_memory\_fraction=0.8)

sess = tf.Session(config=tf.ConfigProto(gpu\_options=gpu\_options))

*# # 建立基础的神经网络*

*# model=Sequential()*

*# model.add(Convolution2D(filters = 32,kernel\_size = 5,strides = 1,padding = 'same',input\_shape = (28,28,3)))*

*# # 增加神经网络激活函数*

*# model.add(Activation('relu'))*

*# model.add(tf.keras.layers.Dropout(0.5))*

*# # 池化层利用MaxPooling2D*

*# model.add(MaxPooling2D(*

*#     pool\_size = (2, 2),             # 向下取样*

*#     strides = (2,2),                # 取样跳2个*

*#     padding='same',                 # Padding method*

*# ))*

*# model.add(tf.keras.layers.Dropout(0.5))  # 加上DropOut防止过拟合*

*# model.add(Convolution2D(64, 5, 5, padding='same'))*

*# model.add(Activation('relu'))*

*# model.add(MaxPooling2D(pool\_size = (2, 2), padding='same'))*

*# model.add(Flatten())                  # 将三维层拉直*

*# model.add(Dense(64))                # 全连接层*

*# model.add(Activation('relu'))         # 激励函数*

*# model.add(Dense(6))                  # 输出6个单位*

*# model.add(Activation('softmax'))      # 分类激励函数*

*# LR=0.001*

*# EPOCHS=300*

*# # 定义优化器 学习效率设置为0.001*

*# opt =SGD(lr=LR)*

*#建立AlexNet模型*

def create\_model(optimizer='adam', kernel\_initializer='he\_normal', activation='relu'):

*#第一层卷积*

*#卷积核数量96 尺寸11\*11 步长4 激活函数relu*

*#最大池化 尺寸3\*3，步长2*

    model = Sequential()

    model.add(Convolution2D(filters=96,

                     kernel\_size=11,

                     strides=4,

                     input\_shape=(224, 224, 3),

                     activation=activation))

    model.add(MaxPooling2D(pool\_size=3, strides=2))

*#第二层卷积*

*#卷积核数量256 尺寸5\*5 激活函数relu same卷积*

*#最大池化 尺寸3\*3，步长2*

    model.add(Convolution2D(filters=256,

                     kernel\_size=5,

                     padding='same',

                     activation=activation))

    model.add(MaxPooling2D(pool\_size=3, strides=2))

*#第三层卷积*

*#卷积核数量384 尺寸3 激活函数relu same卷积*

    model.add(Convolution2D(filters=384,

                     kernel\_size=3,

                     padding='same',

                     activation=activation))

*#第四层卷积*

*#卷积核数量384 尺寸3 激活函数relu same卷积*

    model.add(Convolution2D(filters=384,

                     kernel\_size=3,

                     padding='same',

                     activation=activation))

*#第五层卷积*

*#卷积核数量256 尺寸3 激活函数relu same卷积*

*#最大池化 尺寸3\*3，步长2*

    model.add(Convolution2D(filters=256,

                     kernel\_size=3,

                     padding='same',

                     activation=activation))

    model.add(MaxPooling2D(pool\_size=3, strides=2))

*#展平特征图*

    model.add(Flatten())

*#第一个全连接 1024神经元 relu*

    model.add(Dense(1024, activation='relu'))

    model.add(Dropout(0.5))

*#第二个全连接1024神经元 relu*

    model.add(Dense(1024, activation='relu'))

    model.add(Dropout(0.5))

*#第二个全连接 输出6类结果*

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

*# 定义损失函数-*

    model.compile(loss="categorical\_crossentropy",optimizer=optimizer,metrics=["accuracy"])

*return* model

*#创建模型*

optimizer =tf.train.AdamOptimizer(learning\_rate=1e-4)

model = create\_model(optimizer,

                     kernel\_initializer='uniform',

                     activation='relu')

5、使用matplotlib对模型结构可视化。

*from* tensorflow.keras.utils *import*  plot\_model

*from* IPython.display *import* Image *as* IPythonImage

plot\_model(model, to\_file="AlexNet-model.png", show\_shapes=True)

display(IPythonImage('AlexNet-model.png'))

os.environ["PATH"] += os.pathsep + ':C:/Program Files/Graphviz/bin'

*#绘制图形*

def plot\_loss\_accuracy(history):

*# Loss*

    plt.figure(figsize=[8,6])

    plt.plot(history.history['loss'],'r',linewidth=3.0)

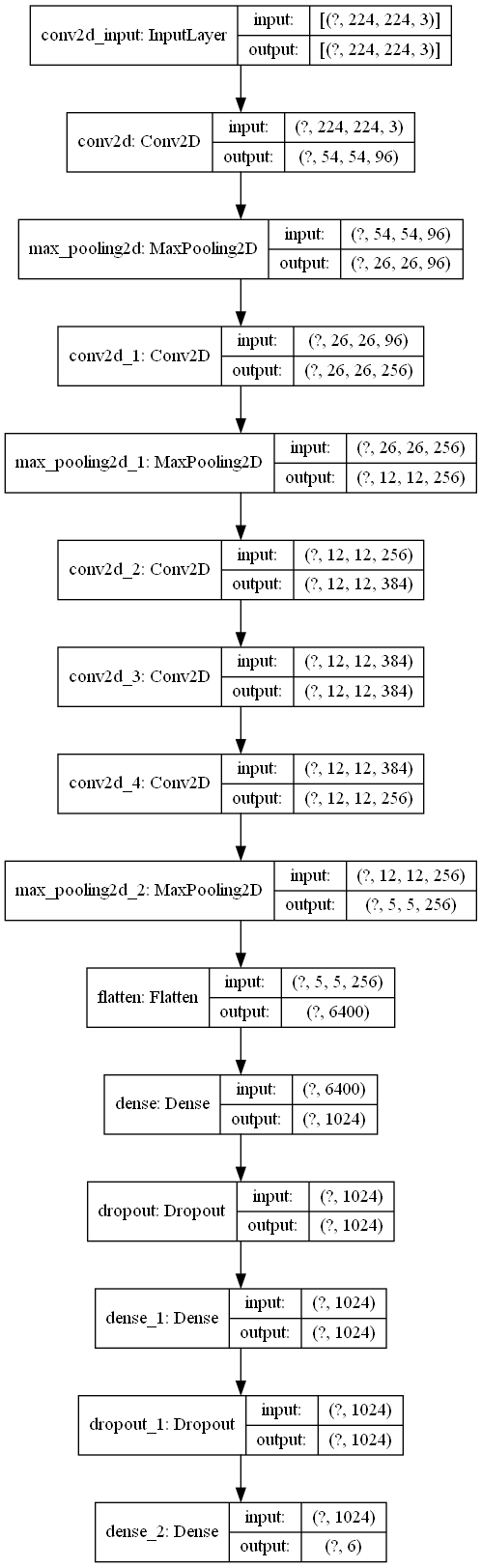
    plt.plot(history.history['val\_loss'],'b',linewidth=3.0)

    plt.legend(['Training loss', 'Validation Loss'],fontsize=18)

    plt.xlabel('Epochs ',fontsize=16)

    plt.ylabel('Loss',fontsize=16)

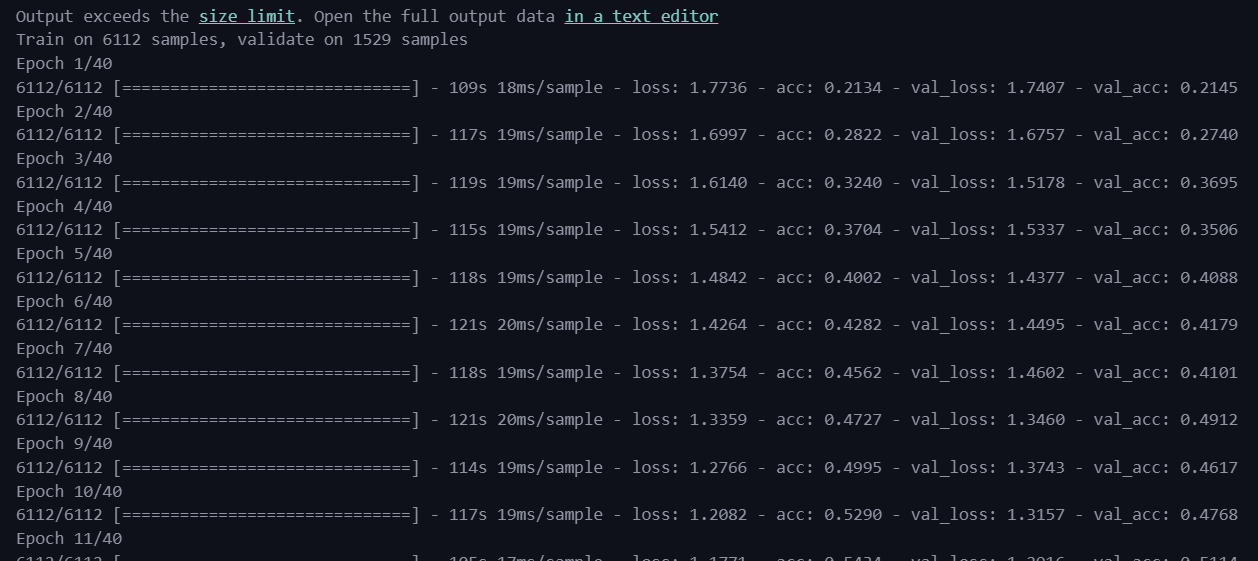
    plt.title('Loss Curves',fontsize=16)



6、训练模型

*# 模型训练*

h=model.fit(x\_train,y\_train,validation\_data=(x\_val,y\_val),epochs=40,batch\_size=64)



1. 通过测试acc,recall,f1-score均达到0.50

*# 模型测试*

predicts=model.predict(x\_val,batch\_size=64)

loss, accuracy = model.evaluate(x\_val, y\_val)

print('\ntest loss: ', loss)

print('\ntest accuracy: ', accuracy)

print(classification\_report(y\_val.argmax(axis=1),

                                predicts.argmax(axis=1), target\_names=["drone","fighter-jet","helicopter","missile","passenger-plane","rocket"]))

