实验六　CIFAR10 图像识别

姓名：李坤璘
班级学号：智能 03 2019202216

一、 实验目的：

1. 掌握 TensorFlow 的使用方法；
2. 利用卷积神经网络对 CIFAR10 数据集进行分类；
3. 掌握 Keras 构建卷积神经网络的方法。

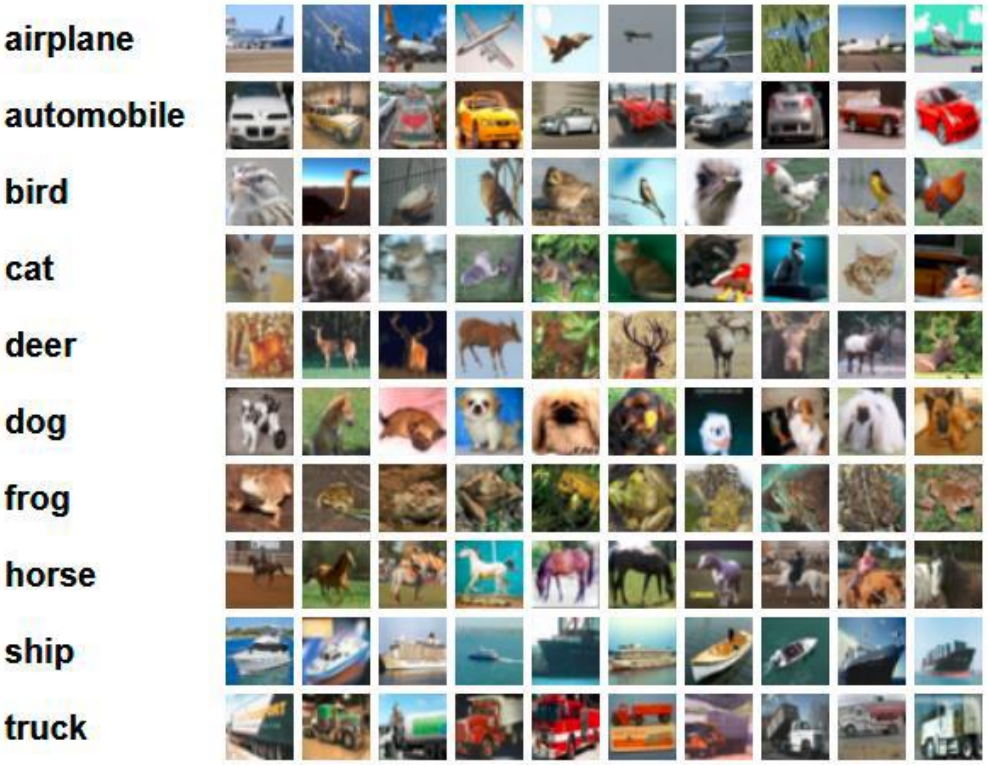
二、 实验条件：

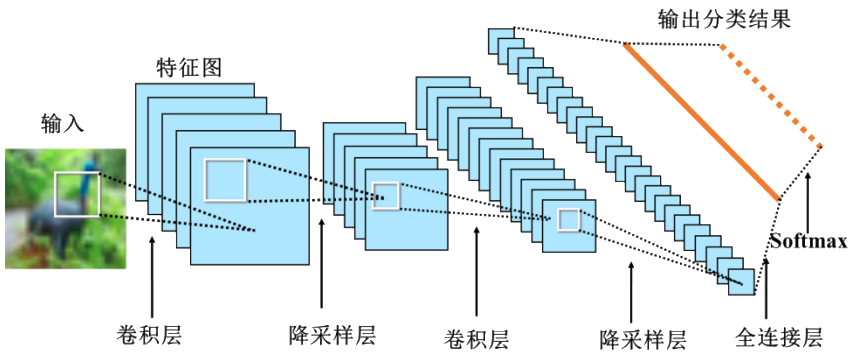
1. PC 微机一台和 Python+TensorFlow 环境。

三、 实验原理：

CIFAR-10是一个用于识别普适物体的小型数据集，它包含了10个类别的RGB彩色图片

图片尺寸：32 x 32
训练图片50000张
测试图片10000张





输入层	卷积层1	降采样层1	卷积层2	降采样层2	全连接层	输出层
32x32图像，通道为3（RGB）	第一次卷积：输入通道：3 输出通道：32 卷积后图像尺寸不变，依然是32x32	第一次降采样：将32x32图像缩小为16x16；池化不改变通道数量，因此依然是32个	第二次卷积：输入通道：32 输出通道：64 卷积后图像尺寸不变，依然是16x16	第二次降采样：将16x16图像缩小为8x8；池化不改变通道数量，因此依然是64个	将64个8x8的图像转换为长度是4096的一维向量，该层有128个神经元	输出层共有10个神经元，对应到0-9这10个类别

四、 实验内容：

1. 数据的导入

```
'''一、数据的导入'''
from keras.datasets import cifar10
import numpy as np
np.random.seed(10)
# 50000 训练集,10000 测试集
# x_train:50000*32*32 y_train:50000 x_test:10000*32*32
y_test:10000
(x_img_train,y_label_train),(x_img_test,y_label_test)=cifar10.loa
d_data()
```

如果是第一次导入数据，那么程序将自动将数据集下载至计算机根目录：

x 系列变量为四维矩阵，存储待处理的图像

```
[2]: print(x_img_train[0,:,:,0]) # 某个图像的某通道
[[ 59  43  50 ... 158 152 148]
 [ 16   0  18 ... 123 119 122]
 [ 25  16  49 ... 118 120 109]
 ...
 [208 201 198 ... 160  56  53]
 [180 173 186 ... 184  97  83]
 [177 168 179 ... 216 151 123]]
```

y 系列变量为列向量，存储待处理图像的标签

```
[3]: print(y_label_train) # 所有标签
```

```
[[6]
 [9]
 [9]
 ...
 [9]
 [1]
 [1]]
```

2. 数据的处理

```
'''二、数据的处理'''
# 标准化
x_img_train_normalize = x_img_train.astype('float32') / 255.0
x_img_test_normalize = x_img_test.astype('float32') / 255.0
# One-Hot 编码
from keras.utils import np_utils
y_label_train_OneHot = np_utils.to_categorical(y_label_train)
y_label_test_OneHot = np_utils.to_categorical(y_label_test)
```

标准化将数据集值归一到[0,1]之间，One-hot 编码将标签映射成二进制向量，方便后续算法的计算：

```
1 print(y_label_train_OneHot)
```

```
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 1.]
 [0. 0. 0. ... 0. 0. 1.]
 ...
 [0. 0. 0. ... 0. 0. 1.]
 [0. 1. 0. ... 0. 0. 0.]
 [0. 1. 0. ... 0. 0. 0.]]
```

3. 建立模型

```
'''三、建立模型'''
from keras.models import Sequential
from keras.layers import
Conv2D,MaxPooling2D,Dense,Dropout,Flatten

# 创建多层顺序连接的神经网络
model = Sequential()
# 卷积层 1，32*32 的图像，共 32 个
model.add(Conv2D(filters=32,kernel_size=(3,3), # 32 个 3*3 的卷积核
                 input_shape=(32,32,3), # 形状:32 高 * 32 宽 * 3 通道
                 activation='relu', # Relu 激活函数
                 padding='same')) # 输入输出尺寸相同
# Dropout 层，随机丢弃 25%输入神经元（置为 0）
model.add(Dropout(0.25))
# 池化层（降采样层） 1，16*16 的图像，共 32 个
model.add(MaxPooling2D(pool_size=(2, 2)))
# 卷积层 2，16*16 的图像，共 64 个
```

```
model.add(Conv2D(filters=64, kernel_size=(3,3),activation='relu',
padding='same')) # 64 个 3*3 的卷积核
model.add(Dropout(0.25))
# 池化层（降采样层） 2，8*8 的图像，共 64 个
model.add(MaxPooling2D(pool_size=(2, 2)))
# 平坦层 8*8*64 个神经元
model.add(Flatten())
model.add(Dropout(rate=0.25))
# 隐藏层（全连接层） 1024 个神经元
model.add(Dense(1024, activation='relu'))
model.add(Dropout(rate=0.25))
# 输出层（全连接层） 对应 0-9 这 10 个类别
model.add(Dense(10, activation='softmax'))
```

整体网络结构如下图所示：

```
[7]  1  model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 32, 32, 32)        896
_________________________________________________________________
dropout (Dropout)            (None, 32, 32, 32)        0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 16, 16, 64)        18496
_________________________________________________________________
dropout_1 (Dropout)          (None, 16, 16, 64)        0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 8, 8, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 4096)              0
_________________________________________________________________
dropout_2 (Dropout)          (None, 4096)              0
_________________________________________________________________
dense (Dense)                (None, 1024)              4195328
_________________________________________________________________
dropout_3 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                10250
...
Total params: 4,224,970
Trainable params: 4,224,970
Non-trainable params: 0
```

4. 训练模型

```
'''四、训练模型'''
# 编译模型（误差函数交叉熵、Adam 梯度下降、指标准确度）
model.compile(loss='categorical_crossentropy',optimizer='adam',
metrics=['accuracy'])
# 训练模型
train_history = model.fit(x_img_train_normalize,
y_label_train_OneHot, # 训练集
```

```
                    validation_split=0.2, # 20%用作验证集
                    epochs=10, batch_size=128, verbose=1) # 10
次迭代训练、每批次 128 张，输出记录
```

这里我们选择迭代训练 10 轮：

```
Epoch 1/10
313/313 [==============================] - 72s 29ms/step - loss: 1.5250 - accuracy: 0.4529 - val_loss: 1.3693 - val_accuracy: 0.5561
Epoch 2/10
313/313 [==============================] - 5s 16ms/step - loss: 1.1548 - accuracy: 0.5896 - val_loss: 1.1389 - val_accuracy: 0.6426
Epoch 3/10
313/313 [==============================] - 5s 16ms/step - loss: 1.0256 - accuracy: 0.6398 - val_loss: 1.0358 - val_accuracy: 0.6643
Epoch 4/10
313/313 [==============================] - 5s 16ms/step - loss: 0.9170 - accuracy: 0.6769 - val_loss: 0.9779 - val_accuracy: 0.6760
Epoch 5/10
313/313 [==============================] - 5s 16ms/step - loss: 0.8258 - accuracy: 0.7088 - val_loss: 0.9117 - val_accuracy: 0.6987
Epoch 6/10
313/313 [==============================] - 5s 16ms/step - loss: 0.7460 - accuracy: 0.7369 - val_loss: 0.8719 - val_accuracy: 0.7175
Epoch 7/10
313/313 [==============================] - 5s 16ms/step - loss: 0.6713 - accuracy: 0.7653 - val_loss: 0.8551 - val_accuracy: 0.7233
Epoch 8/10
313/313 [==============================] - 5s 16ms/step - loss: 0.5932 - accuracy: 0.7932 - val_loss: 0.8095 - val_accuracy: 0.7230
Epoch 9/10
313/313 [==============================] - 5s 16ms/step - loss: 0.5242 - accuracy: 0.8148 - val_loss: 0.8044 - val_accuracy: 0.7304
Epoch 10/10
313/313 [==============================] - 5s 16ms/step - loss: 0.4629 - accuracy: 0.8367 - val_loss: 0.7713 - val_accuracy: 0.7394
```

以上数据存储至 train_history 中：

```
{'loss': [1.525039792060852, 1.154836893081665, 1.0256189107894897, 0.9169905185699463, 0.8257558345794678, 0.7459515333175659, 0.6713104248046875, 0.5931974649429321,

 'accuracy': [0.4528999924659729, 0.5896000266075134, 0.6397749781608582, 0.6768749952316284, 0.7088000178337097, 0.7369250059127808, 0.765250027179718,

'val loss': [1.3693269491195679, 1.1389243602752686, 1.0357707738876343, 0.977938175201416, 0.911689281463623, 0.8718838691711426, 0.8551486134529114,

'val_accuracy': [0.5561000108718872, 0.6425999999046326, 0.6643000245094299, 0.6759999990463257, 0.6987000107765198, 0.7174999713897705, 0.7232999801635742,
```

后续将对以上数据可视化。

5. 测试模型

```
'''五、测试模型'''
scores =
model.evaluate(x_img_test_normalize,y_label_test_OneHot,verbose=0
)
print(scores)
```

```
[0.7738075852394104, 0.7384999990463257]
```

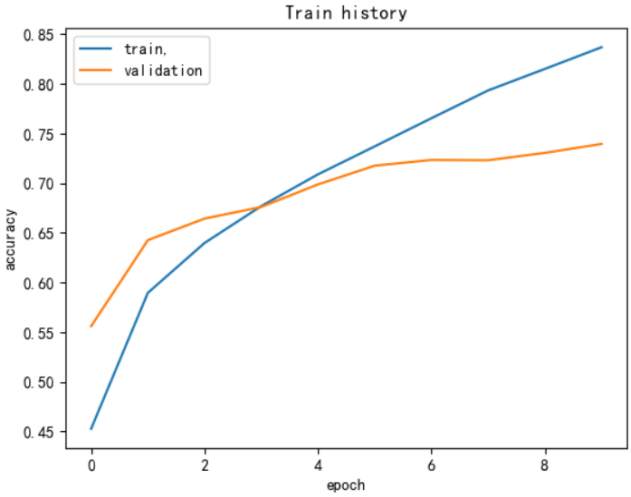
第一个值为最终损失值，第二个值为准确率。

6. 相关信息可视化

（1）准确率

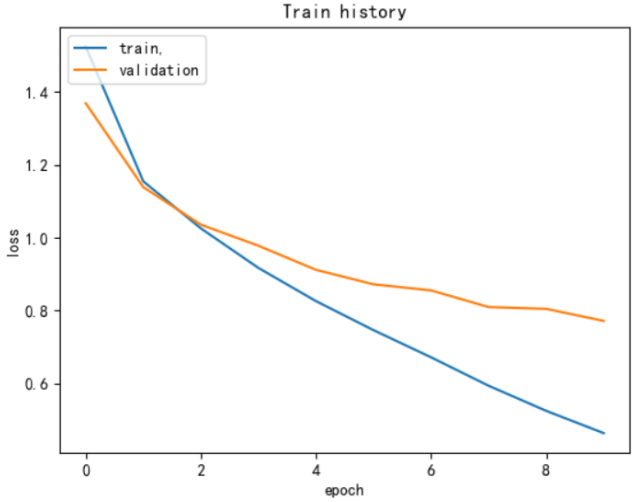
```
'''六、相关信息可视化'''
from matplotlib import pyplot as plt
plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签
plt.rcParams['axes.unicode_minus']=False #用来正常显示负号
def show_train_history(train_history,train,validation):
    plt.plot(train_history.history[train])
```

```
    plt.plot(train_history.history[validation])
    plt.title('Train history')
    plt.ylabel(train)
    plt.xlabel('epoch')
    plt.legend(['train,','validation'],loc='upper left')
# 准确率变化曲线
plt.figure(1)
show_train_history(train_history,'accuracy','val_accuracy')
```



（2）损失率

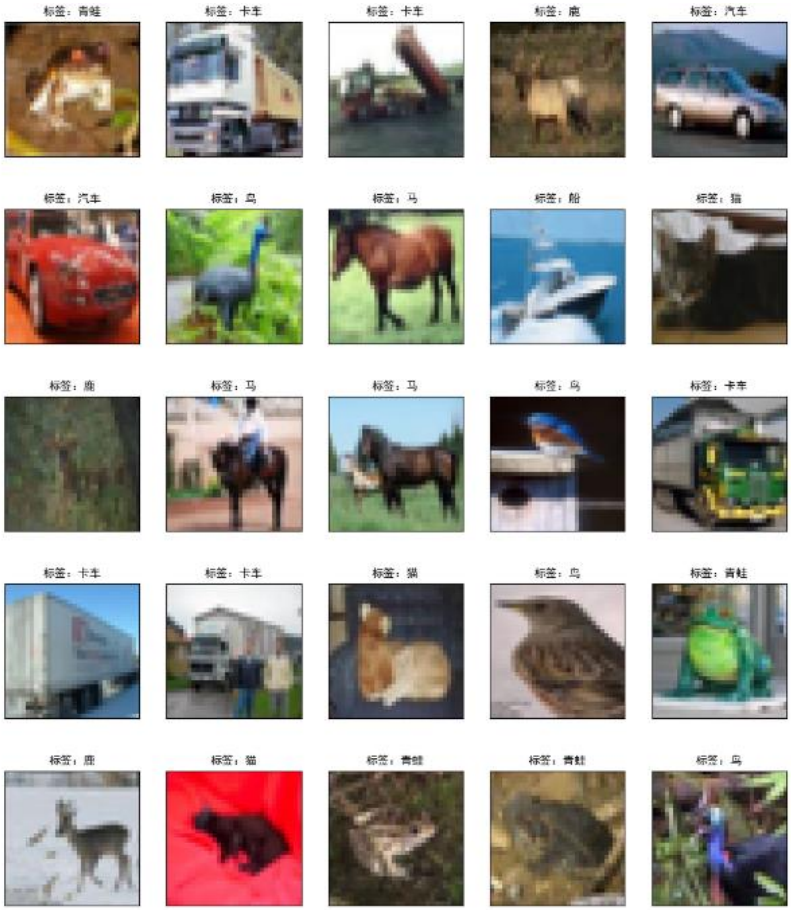
```
plt.figure(2)
show_train_history(train_history,'loss','val_loss')
```



（3）查看一下原始数据集的图像都是什么样的

```
# 输出 25 张原数据集的图像
label_dict={0:'飞机',1:'汽车',2:'鸟',3:'猫',4:'鹿',5:"狗",6:'青蛙
',7:'马',8:'船',9:'卡车'}
#显示几张图片和标签
```

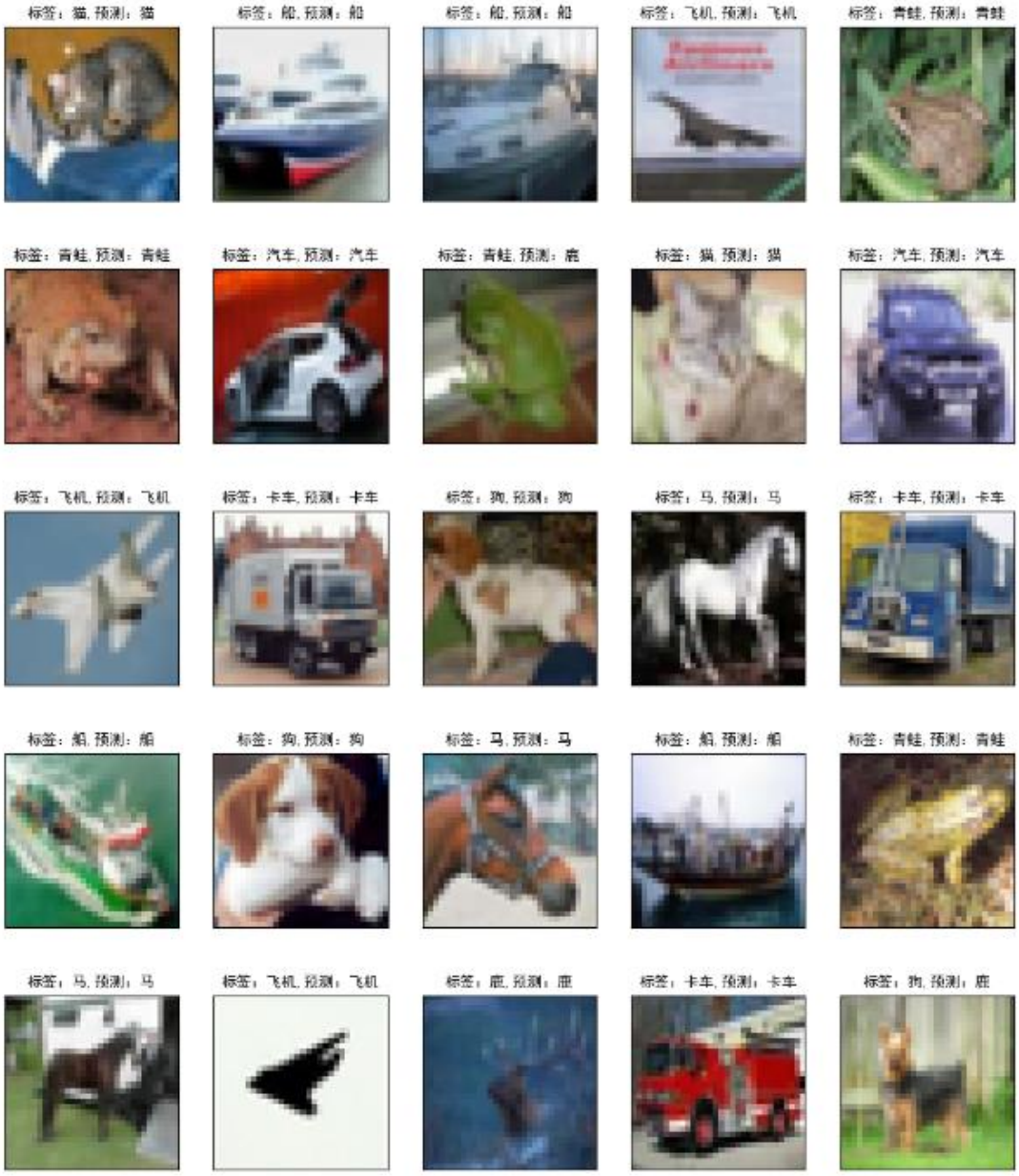
```
def
show_images_labels_prediction(images,labels,prediction,idx,num=10
):
    flig=plt.figure(figsize=(12,14))
    if num>25:
        num=25
    for i in range(0,num):
        ax=plt.subplot(5,5,1+i)
        ax.imshow(images[idx],cmap='binary')
        title='标签: '+str(label_dict[labels[i][0]])
        if len(prediction)>0:
            title+=',预测: '+label_dict[prediction[i]]
        ax.set_title(title,fontsize=10)
        ax.set_xticks([])
        ax.set_yticks([])
        idx+=1
show_images_labels_prediction(x_img_train,y_label_train,[],0,25)
```



（4）显示测试集中预测和真实标签

```
# 显示测试集中预测和真实标签
predicted_probability=model.predict(x_img_test_normalize)
prediction=np.argmax(predicted_probability, axis=-1)
print(prediction)
```

```
show_images_labels_prediction(x_img_test,y_label_test,prediction,
0,25)
```



（5）混淆矩阵

```
# 混淆矩阵
import pandas as pd
# confusion_matrix = pd.crosstab(y_label_test.reshape(-
1),prediction,rownames=['label'],colnames=['predict'])
# print(confusion_matrix)
pd.crosstab(y_label_test.reshape(-
1),prediction,rownames=['label'],colnames=['predict'])
```

[17]:
```
# 混淆矩阵
import pandas as pd
# confusion_matrix = pd.crosstab(y_label_test.reshape(-1),predicti
# print(confusion_matrix)
pd.crosstab(y_label_test.reshape(-1),prediction,rownames=['label']
```

[17]:

predict	0	1	2	3	4	5	6	7	8	9
label										
0	808	14	29	10	17	3	14	14	67	24
1	22	837	11	9	6	5	9	9	31	61
2	79	2	602	32	134	34	65	38	12	2
3	29	6	95	443	126	149	88	42	12	10
4	24	2	55	30	753	20	44	56	14	2
5	12	3	77	126	86	585	41	54	13	3
6	5	4	36	30	53	15	845	3	7	2
7	13	1	37	23	87	34	10	787	4	4
8	61	30	18	11	8	4	6	4	845	13
9	42	86	15	23	7	9	13	29	48	728

五、实验代码及结果

以上为使用 jupyter 编写的 ipynb 程序，完整可运行 py 代码如下：

```
from keras.datasets import cifar10
import numpy as np
from keras.utils import np_utils
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Dense, Dropout,
Flatten
from matplotlib import pyplot as plt
import pandas as pd

plt.rcParams['font.sans-serif'] = ['SimHei']  # 用来正常显示中文标
签
plt.rcParams['axes.unicode_minus'] = False  # 用来正常显示负号

'''一、数据的导入'''
np.random.seed(10)
# 50000 训练集,10000 测试集
# x_train:50000*32*32 y_train:50000 x_test:10000*32*32
y_test:10000
(x_img_train, y_label_train), (x_img_test, y_label_test) =
cifar10.load_data()
```

```
# print(x_img_train[0,:,:,0]) # 某个图像的某通道
# print(y_label_train) # 所有标签

'''二、数据的处理'''
# 标准化
x_img_train_normalize = x_img_train.astype('float32') / 255.0
x_img_test_normalize = x_img_test.astype('float32') / 255.0
# One-Hot 编码
y_label_train_OneHot = np_utils.to_categorical(y_label_train)
y_label_test_OneHot = np_utils.to_categorical(y_label_test)
# print(y_label_train_OneHot)

'''三、建立模型'''
# 创建多层顺序连接的神经网络
model = Sequential()
# 卷积层 1，32*32 的图像，共 32 个
model.add(Conv2D(filters=32, kernel_size=(3, 3),  # 32 个 3*3 的卷积核
                 input_shape=(32, 32, 3),  # 形状:32 高 * 32 宽 * 3 通道
                 activation='relu',  # Relu 激活函数
                 padding='same'))  # 输入输出尺寸相同
# Dropout 层，随机丢弃 25%输入神经元（置为 0）
model.add(Dropout(0.25))
# 池化层（降采样层） 1，16*16 的图像，共 32 个
model.add(MaxPooling2D(pool_size=(2, 2)))
# 卷积层 2，16*16 的图像，共 64 个
model.add(Conv2D(filters=64, kernel_size=(3, 3),
activation='relu', padding='same'))  # 64 个 3*3 的卷积核
model.add(Dropout(0.25))
# 池化层（降采样层） 2，8*8 的图像，共 64 个
model.add(MaxPooling2D(pool_size=(2, 2)))
# 平坦层 8*8*64 个神经元
model.add(Flatten())
model.add(Dropout(rate=0.25))
# 隐藏层（全连接层） 1024 个神经元
model.add(Dense(1024, activation='relu'))
model.add(Dropout(rate=0.25))
# 输出层（全连接层） 对应 0-9 这 10 个类别
model.add(Dense(10, activation='softmax'))

'''四、训练模型'''
# 编译模型（误差函数交叉熵、Adam 梯度下降、指标准确度）
```

```
model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
# 训练模型
train_history = model.fit(x_img_train_normalize,
y_label_train_OneHot,  # 训练集
                          validation_split=0.2,  # 20%用作验证集
                          epochs=15, batch_size=128, verbose=1)  #
10 次迭代训练、每批次 128 张，输出记录
# print(train_history.history)

'''五、测试模型'''
scores = model.evaluate(x_img_test_normalize,
y_label_test_OneHot, verbose=0)
# print(scores)

'''六、相关信息可视化'''


def show_train_history(train_history, train, validation):
    plt.plot(train_history.history[train])
    plt.plot(train_history.history[validation])
    plt.title('Train history')
    plt.ylabel(train)
    plt.xlabel('epoch')
    plt.legend(['train,', 'validation'], loc='upper left')


# 1.准确率变化曲线
plt.figure(1)
show_train_history(train_history, 'accuracy', 'val_accuracy')
# 2.损失率变化曲线
plt.figure(2)
show_train_history(train_history, 'loss', 'val_loss')
# 3.输出 25 张原数据集的图像
label_dict = {0: '飞机', 1: '汽车', 2: '鸟', 3: '猫', 4: '鹿', 5:
"狗", 6: '青蛙', 7: '马', 8: '船', 9: '卡车'}


# 显示几张图片和标签
def show_images_labels_prediction(images, labels, prediction,
idx, num=10):
    flig = plt.figure(figsize=(12, 14))
    if num > 25:
        num = 25
    for i in range(0, num):
```

```
        ax = plt.subplot(5, 5, 1 + i)
        ax.imshow(images[idx], cmap='binary')
        title = '标签: ' + str(label_dict[labels[i][0]])
        if len(prediction) > 0:
            title += ',预测: ' + label_dict[prediction[i]]
        ax.set_title(title, fontsize=10)
        ax.set_xticks([])
        ax.set_yticks([])
        idx += 1


show_images_labels_prediction(x_img_train, y_label_train, [], 0,
25)
# 4.显示测试集中预测和真实标签
predicted_probability = model.predict(x_img_test_normalize)
prediction = np.argmax(predicted_probability, axis=-1)
# print(prediction)
show_images_labels_prediction(x_img_test, y_label_test,
prediction, 0, 25)
# 5.混淆矩阵
confusion_matrix = pd.crosstab(y_label_test.reshape(-1),
prediction, rownames=['label'], colnames=['predict'])
print(confusion_matrix)
# confusion_matrix.head()
# 6.查看完整神经网络的构架层次
model.summary()
# 7.准确率
print("准确率: {:.4f}%".format(scores[1] * 100))
plt.show()
```

六、实验总结

这次实验基于 Keras 框架，利用 CIFAR-10 数据集建立卷积神经网络（CNN）实现对分类问题的预测。

①首先通过数据的导入，获取训练集和测试集，并对数据进行了标准化处理和 One-Hot 编码；

②其次，在建立模型时，利用 Conv2D 卷积层对图像进行卷积操作，MaxPooling2D 池化层对降采样处理，Flatten 层对图像进行降维，Dense 层对数据进行全连接操作，Dropout 层通过随机丢弃神经元来防止过拟合；

③然后，在模型训练过程中，利用 compile 函数定义误差函数、梯度下降的优化器和指标准确度，并用 fit 函数将训练数据、验证数据、迭代次数、批次大小作为参数传入，进行模型的训练；

④接着，利用评估函数 evaluate 来评估训练好的模型准确率；

⑤最后，利用 matplotlib 库进行可视化处理，分别绘制准确率和损失率的变化

曲线，输出数据集的图片，查看测试集预测结果并对其进行混淆矩阵分析，查看神经网络的层次结构，并输出实验的准确率结果。

通过本次实验，我掌握了卷积神经网络的基本知识和实现方式，熟悉使用 Keras 框架构建卷积神经网络进行分类的方法，以及对图像的标准化处理和分类问题的解决方法。同时，通过对实验结果的可视化展示，也能够更好地了解数据集的特点和模型的表现，进一步优化和改进模型效果。