import matplotlib.pyplot as plt import tensorflow as tf from tensorflow.keras import datasets, layers, models

**import** matplotlib.pyplot **as** plt

**import** tensorflow **as** tf

**from** tensorflow.keras **import** datasets, layers, models

C:\Users\LENOVO\AppData\Roaming\Python\Python39\site-packages\scipy\\_\_init\_\_.py:146: UserWarning: A NumPy version >=1.17.3 and <1.25.0 is required for this version of SciPy (detected version 1.26.1

warnings.warn(f"A NumPy version >={np\_minversion} and <{np\_maxversion}"

In [2]:

**import** numpy **as** np

In [3]:

data **=** np.load('cifar-10.npz')

In [4]:

filenames **=** ["x\_train","y\_train","x\_test","y\_test"]

nps **=** []

**for** filename **in** filenames:

nps.append(data[filename])

train\_images,train\_labels,test\_images,test\_labels **=** nps

**Shuffling, resizing images**

In [5]:

**from** tensorflow.keras.utils **import** to\_categorical

train\_images, test\_images **=** train\_images **/** 255.0, test\_images **/** 255.0

y\_train\_cat **=** to\_categorical(train\_labels,10)

y\_test\_cat **=** to\_categorical(test\_labels,10)

In [6]:

train\_images

Out[6]:

array([[[[0.23137255, 0.24313725, 0.24705882],

[0.16862745, 0.18039216, 0.17647059],

[0.19607843, 0.18823529, 0.16862745],

...,

[0.61960784, 0.51764706, 0.42352941],

[0.59607843, 0.49019608, 0.4 ],

[0.58039216, 0.48627451, 0.40392157]],

[[0.0627451 , 0.07843137, 0.07843137],

[0. , 0. , 0. ],

[0.07058824, 0.03137255, 0. ],

...,

[0.48235294, 0.34509804, 0.21568627],

[0.46666667, 0.3254902 , 0.19607843],

[0.47843137, 0.34117647, 0.22352941]],

[[0.09803922, 0.09411765, 0.08235294],

[0.0627451 , 0.02745098, 0. ],

[0.19215686, 0.10588235, 0.03137255],

...,

[0.4627451 , 0.32941176, 0.19607843],

[0.47058824, 0.32941176, 0.19607843],

[0.42745098, 0.28627451, 0.16470588]],

...,

[[0.81568627, 0.66666667, 0.37647059],

[0.78823529, 0.6 , 0.13333333],

[0.77647059, 0.63137255, 0.10196078],

...,

[0.62745098, 0.52156863, 0.2745098 ],

[0.21960784, 0.12156863, 0.02745098],

[0.20784314, 0.13333333, 0.07843137]],

[[0.70588235, 0.54509804, 0.37647059],

[0.67843137, 0.48235294, 0.16470588],

[0.72941176, 0.56470588, 0.11764706],

...,

[0.72156863, 0.58039216, 0.36862745],

[0.38039216, 0.24313725, 0.13333333],

[0.3254902 , 0.20784314, 0.13333333]],

[[0.69411765, 0.56470588, 0.45490196],

[0.65882353, 0.50588235, 0.36862745],

[0.70196078, 0.55686275, 0.34117647],

...,

[0.84705882, 0.72156863, 0.54901961],

[0.59215686, 0.4627451 , 0.32941176],

[0.48235294, 0.36078431, 0.28235294]]],

[[[0.60392157, 0.69411765, 0.73333333],

[0.49411765, 0.5372549 , 0.53333333],

[0.41176471, 0.40784314, 0.37254902],

...,

[0.35686275, 0.37254902, 0.27843137],

[0.34117647, 0.35294118, 0.27843137],

[0.30980392, 0.31764706, 0.2745098 ]],

[[0.54901961, 0.62745098, 0.6627451 ],

[0.56862745, 0.6 , 0.60392157],

[0.49019608, 0.49019608, 0.4627451 ],

...,

[0.37647059, 0.38823529, 0.30588235],

[0.30196078, 0.31372549, 0.24313725],

[0.27843137, 0.28627451, 0.23921569]],

[[0.54901961, 0.60784314, 0.64313725],

[0.54509804, 0.57254902, 0.58431373],

[0.45098039, 0.45098039, 0.43921569],

...,

[0.30980392, 0.32156863, 0.25098039],

[0.26666667, 0.2745098 , 0.21568627],

[0.2627451 , 0.27058824, 0.21568627]],

...,

[[0.68627451, 0.65490196, 0.65098039],

[0.61176471, 0.60392157, 0.62745098],

[0.60392157, 0.62745098, 0.66666667],

...,

[0.16470588, 0.13333333, 0.14117647],

[0.23921569, 0.20784314, 0.22352941],

[0.36470588, 0.3254902 , 0.35686275]],

[[0.64705882, 0.60392157, 0.50196078],

[0.61176471, 0.59607843, 0.50980392],

[0.62352941, 0.63137255, 0.55686275],

...,

[0.40392157, 0.36470588, 0.37647059],

[0.48235294, 0.44705882, 0.47058824],

[0.51372549, 0.4745098 , 0.51372549]],

[[0.63921569, 0.58039216, 0.47058824],

[0.61960784, 0.58039216, 0.47843137],

[0.63921569, 0.61176471, 0.52156863],

...,

[0.56078431, 0.52156863, 0.54509804],

[0.56078431, 0.5254902 , 0.55686275],

[0.56078431, 0.52156863, 0.56470588]]],

[[[1. , 1. , 1. ],

[0.99215686, 0.99215686, 0.99215686],

[0.99215686, 0.99215686, 0.99215686],

...,

[0.99215686, 0.99215686, 0.99215686],

[0.99215686, 0.99215686, 0.99215686],

[0.99215686, 0.99215686, 0.99215686]],

[[1. , 1. , 1. ],

[1. , 1. , 1. ],

[1. , 1. , 1. ],

...,

[1. , 1. , 1. ],

[1. , 1. , 1. ],

[1. , 1. , 1. ]],

[[1. , 1. , 1. ],

[0.99607843, 0.99607843, 0.99607843],

[0.99607843, 0.99607843, 0.99607843],

...,

[0.99607843, 0.99607843, 0.99607843],

[0.99607843, 0.99607843, 0.99607843],

[0.99607843, 0.99607843, 0.99607843]],

...,

[[0.44313725, 0.47058824, 0.43921569],

[0.43529412, 0.4627451 , 0.43529412],

[0.41176471, 0.43921569, 0.41568627],

...,

[0.28235294, 0.31764706, 0.31372549],

[0.28235294, 0.31372549, 0.30980392],

[0.28235294, 0.31372549, 0.30980392]],

[[0.43529412, 0.4627451 , 0.43137255],

[0.40784314, 0.43529412, 0.40784314],

[0.38823529, 0.41568627, 0.38431373],

...,

[0.26666667, 0.29411765, 0.28627451],

[0.2745098 , 0.29803922, 0.29411765],

[0.30588235, 0.32941176, 0.32156863]],

[[0.41568627, 0.44313725, 0.41176471],

[0.38823529, 0.41568627, 0.38431373],

[0.37254902, 0.4 , 0.36862745],

...,

[0.30588235, 0.33333333, 0.3254902 ],

[0.30980392, 0.33333333, 0.3254902 ],

[0.31372549, 0.3372549 , 0.32941176]]],

...,

[[[0.1372549 , 0.69803922, 0.92156863],

[0.15686275, 0.69019608, 0.9372549 ],

[0.16470588, 0.69019608, 0.94509804],

...,

[0.38823529, 0.69411765, 0.85882353],

[0.30980392, 0.57647059, 0.77254902],

[0.34901961, 0.58039216, 0.74117647]],

[[0.22352941, 0.71372549, 0.91764706],

[0.17254902, 0.72156863, 0.98039216],

[0.19607843, 0.71764706, 0.94117647],

...,

[0.61176471, 0.71372549, 0.78431373],

[0.55294118, 0.69411765, 0.80784314],

[0.45490196, 0.58431373, 0.68627451]],

[[0.38431373, 0.77254902, 0.92941176],

[0.25098039, 0.74117647, 0.98823529],

[0.27058824, 0.75294118, 0.96078431],

...,

[0.7372549 , 0.76470588, 0.80784314],

[0.46666667, 0.52941176, 0.57647059],

[0.23921569, 0.30980392, 0.35294118]],

...,

[[0.28627451, 0.30980392, 0.30196078],

[0.20784314, 0.24705882, 0.26666667],

[0.21176471, 0.26666667, 0.31372549],

...,

[0.06666667, 0.15686275, 0.25098039],

[0.08235294, 0.14117647, 0.2 ],

[0.12941176, 0.18823529, 0.19215686]],

[[0.23921569, 0.26666667, 0.29411765],

[0.21568627, 0.2745098 , 0.3372549 ],

[0.22352941, 0.30980392, 0.40392157],

...,

[0.09411765, 0.18823529, 0.28235294],

[0.06666667, 0.1372549 , 0.20784314],

[0.02745098, 0.09019608, 0.1254902 ]],

[[0.17254902, 0.21960784, 0.28627451],

[0.18039216, 0.25882353, 0.34509804],

[0.19215686, 0.30196078, 0.41176471],

...,

[0.10588235, 0.20392157, 0.30196078],

[0.08235294, 0.16862745, 0.25882353],

[0.04705882, 0.12156863, 0.19607843]]],

[[[0.74117647, 0.82745098, 0.94117647],

[0.72941176, 0.81568627, 0.9254902 ],

[0.7254902 , 0.81176471, 0.92156863],

...,

[0.68627451, 0.76470588, 0.87843137],

[0.6745098 , 0.76078431, 0.87058824],

[0.6627451 , 0.76078431, 0.8627451 ]],

[[0.76078431, 0.82352941, 0.9372549 ],

[0.74901961, 0.81176471, 0.9254902 ],

[0.74509804, 0.80784314, 0.92156863],

...,

[0.67843137, 0.75294118, 0.8627451 ],

[0.67058824, 0.74901961, 0.85490196],

[0.65490196, 0.74509804, 0.84705882]],

[[0.81568627, 0.85882353, 0.95686275],

[0.80392157, 0.84705882, 0.94117647],

[0.8 , 0.84313725, 0.9372549 ],

...,

[0.68627451, 0.74901961, 0.85098039],

[0.6745098 , 0.74509804, 0.84705882],

[0.6627451 , 0.74901961, 0.84313725]],

...,

[[0.81176471, 0.78039216, 0.70980392],

[0.79607843, 0.76470588, 0.68627451],

[0.79607843, 0.76862745, 0.67843137],

...,

[0.52941176, 0.51764706, 0.49803922],

[0.63529412, 0.61960784, 0.58823529],

[0.65882353, 0.63921569, 0.59215686]],

[[0.77647059, 0.74509804, 0.66666667],

[0.74117647, 0.70980392, 0.62352941],

[0.70588235, 0.6745098 , 0.57647059],

...,

[0.69803922, 0.67058824, 0.62745098],

[0.68627451, 0.6627451 , 0.61176471],

[0.68627451, 0.6627451 , 0.60392157]],

[[0.77647059, 0.74117647, 0.67843137],

[0.74117647, 0.70980392, 0.63529412],

[0.69803922, 0.66666667, 0.58431373],

...,

[0.76470588, 0.72156863, 0.6627451 ],

[0.76862745, 0.74117647, 0.67058824],

[0.76470588, 0.74509804, 0.67058824]]],

[[[0.89803922, 0.89803922, 0.9372549 ],

[0.9254902 , 0.92941176, 0.96862745],

[0.91764706, 0.9254902 , 0.96862745],

...,

[0.85098039, 0.85882353, 0.91372549],

[0.86666667, 0.8745098 , 0.91764706],

[0.87058824, 0.8745098 , 0.91372549]],

[[0.87058824, 0.86666667, 0.89803922],

[0.9372549 , 0.9372549 , 0.97647059],

[0.91372549, 0.91764706, 0.96470588],

...,

[0.8745098 , 0.8745098 , 0.9254902 ],

[0.89019608, 0.89411765, 0.93333333],

[0.82352941, 0.82745098, 0.8627451 ]],

[[0.83529412, 0.80784314, 0.82745098],

[0.91764706, 0.90980392, 0.9372549 ],

[0.90588235, 0.91372549, 0.95686275],

...,

[0.8627451 , 0.8627451 , 0.90980392],

[0.8627451 , 0.85882353, 0.90980392],

[0.79215686, 0.79607843, 0.84313725]],

...,

[[0.58823529, 0.56078431, 0.52941176],

[0.54901961, 0.52941176, 0.49803922],

[0.51764706, 0.49803922, 0.47058824],

...,

[0.87843137, 0.87058824, 0.85490196],

[0.90196078, 0.89411765, 0.88235294],

[0.94509804, 0.94509804, 0.93333333]],

[[0.5372549 , 0.51764706, 0.49411765],

[0.50980392, 0.49803922, 0.47058824],

[0.49019608, 0.4745098 , 0.45098039],

...,

[0.70980392, 0.70588235, 0.69803922],

[0.79215686, 0.78823529, 0.77647059],

[0.83137255, 0.82745098, 0.81176471]],

[[0.47843137, 0.46666667, 0.44705882],

[0.4627451 , 0.45490196, 0.43137255],

[0.47058824, 0.45490196, 0.43529412],

...,

[0.70196078, 0.69411765, 0.67843137],

[0.64313725, 0.64313725, 0.63529412],

[0.63921569, 0.63921569, 0.63137255]]]])

In [7]:

**import** matplotlib.pyplot **as** plt

In [8]:

class\_names **=** ['airplane', 'automobile', 'bird', 'cat', 'deer','dog', 'frog', 'horse', 'ship', 'truck']

plt.figure(figsize**=**(10,10))

**for** i **in** range(10):

plt.subplot(5,5,i**+**1)

plt.xticks([])

plt.yticks([])

*# plt.grid(False)*

plt.imshow(train\_images[i])

plt.xlabel(class\_names[train\_labels[i][0]])

plt.show()

A group of images of animals

Description automatically generated

In [9]:

**from** tensorflow.keras.layers **import** BatchNormalization, Dropout

In [10]:

model **=** models.Sequential()

model.add(layers.Conv2D(32, (3, 3), activation**=**'relu', input\_shape**=**(32, 32, 3)))

model.add(BatchNormalization())

model.add(layers.MaxPooling2D((2, 2)))

model.add(Dropout(0.25))

model.add(layers.Conv2D(64, (3, 3), activation**=**'relu'))

model.add(BatchNormalization())

model.add(layers.MaxPooling2D((2, 2)))

model.add(Dropout(0.25))

model.add(layers.Conv2D(64, (3, 3), activation**=**'relu'))

model.add(BatchNormalization())

model.add(layers.Flatten())

model.add(layers.Dense(64, activation**=**'relu'))

model.add(layers.Dense(10,activation**=**'softmax'))

model.summary()

Model: "sequential"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv2d (Conv2D) (None, 30, 30, 32) 896

batch\_normalization (Batch (None, 30, 30, 32) 128

Normalization)

max\_pooling2d (MaxPooling2 (None, 15, 15, 32) 0

D)

dropout (Dropout) (None, 15, 15, 32) 0

conv2d\_1 (Conv2D) (None, 13, 13, 64) 18496

batch\_normalization\_1 (Bat (None, 13, 13, 64) 256

chNormalization)

max\_pooling2d\_1 (MaxPoolin (None, 6, 6, 64) 0

g2D)

dropout\_1 (Dropout) (None, 6, 6, 64) 0

conv2d\_2 (Conv2D) (None, 4, 4, 64) 36928

batch\_normalization\_2 (Bat (None, 4, 4, 64) 256

chNormalization)

flatten (Flatten) (None, 1024) 0

dense (Dense) (None, 64) 65600

dense\_1 (Dense) (None, 10) 650

=================================================================

Total params: 123210 (481.29 KB)

Trainable params: 122890 (480.04 KB)

Non-trainable params: 320 (1.25 KB)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

In [11]:

*# from keras import ImageDataGenerator*

In [12]:

model.compile(optimizer**=**'adam',

loss**=**'categorical\_crossentropy',

metrics**=**['accuracy'])

​

*# datagen = ImageDataGenerator(width\_shift\_range=0.1, height\_shift\_range=0.1, horizontal\_flip=True,rotation\_range=20)*

*# it\_train = datagen.flow(train\_images,y\_train\_cat)*

*# steps = int(train\_images.shape[0] / 64)*

*# history=model.fit\_generator(it\_train,epochs=15,steps\_per\_epoch=steps,validation\_data=(test\_images,y\_test\_cat))*

In [13]:

epochs**=**15 *# set this according to your computer*

history**=**model.fit(train\_images, y\_train\_cat,

validation\_data**=**(test\_images,y\_test\_cat),epochs**=**epochs)

Epoch 1/15

1563/1563 [==============================] - 60s 37ms/step - loss: 1.4396 - accuracy: 0.4910 - val\_loss: 1.6897 - val\_accuracy: 0.4464

Epoch 2/15

1563/1563 [==============================] - 67s 43ms/step - loss: 1.0825 - accuracy: 0.6148 - val\_loss: 1.2067 - val\_accuracy: 0.6017

Epoch 3/15

1563/1563 [==============================] - 68s 44ms/step - loss: 0.9405 - accuracy: 0.6691 - val\_loss: 1.1835 - val\_accuracy: 0.6000

Epoch 4/15

1563/1563 [==============================] - 69s 44ms/step - loss: 0.8515 - accuracy: 0.6984 - val\_loss: 0.9006 - val\_accuracy: 0.6902

Epoch 5/15

1563/1563 [==============================] - 69s 44ms/step - loss: 0.7916 - accuracy: 0.7220 - val\_loss: 0.8762 - val\_accuracy: 0.6992

Epoch 6/15

1563/1563 [==============================] - 60s 38ms/step - loss: 0.7450 - accuracy: 0.7378 - val\_loss: 0.9060 - val\_accuracy: 0.6911

Epoch 7/15

1563/1563 [==============================] - 60s 39ms/step - loss: 0.6969 - accuracy: 0.7538 - val\_loss: 1.1760 - val\_accuracy: 0.6347

Epoch 8/15

1563/1563 [==============================] - 60s 39ms/step - loss: 0.6713 - accuracy: 0.7648 - val\_loss: 0.9388 - val\_accuracy: 0.6946

Epoch 9/15

1563/1563 [==============================] - 67s 43ms/step - loss: 0.6403 - accuracy: 0.7748 - val\_loss: 0.8399 - val\_accuracy: 0.7171

Epoch 10/15

1563/1563 [==============================] - 71s 45ms/step - loss: 0.6101 - accuracy: 0.7840 - val\_loss: 1.0119 - val\_accuracy: 0.6745

Epoch 11/15

1563/1563 [==============================] - 68s 44ms/step - loss: 0.5892 - accuracy: 0.7904 - val\_loss: 0.7937 - val\_accuracy: 0.7411

Epoch 12/15

1563/1563 [==============================] - 72s 46ms/step - loss: 0.5727 - accuracy: 0.7991 - val\_loss: 0.8148 - val\_accuracy: 0.7367

Epoch 13/15

1563/1563 [==============================] - 70s 45ms/step - loss: 0.5532 - accuracy: 0.8031 - val\_loss: 0.8641 - val\_accuracy: 0.7252

Epoch 14/15

1563/1563 [==============================] - 77s 49ms/step - loss: 0.5321 - accuracy: 0.8133 - val\_loss: 0.7960 - val\_accuracy: 0.7387

Epoch 15/15

1563/1563 [==============================] - 71s 46ms/step - loss: 0.5177 - accuracy: 0.8155 - val\_loss: 0.9122 - val\_accuracy: 0.7125

In [14]:

test\_loss,test\_acc **=** model.evaluate(test\_images,y\_test\_cat)

print("loss %.3f"**%**test\_loss)

print("acc %.3f"**%**test\_acc)

313/313 [==============================] - 3s 9ms/step - loss: 0.9122 - accuracy: 0.7125

loss 0.912

acc 0.712

In [15]:

predicted\_values **=** model.predict(test\_images)

313/313 [==============================] - 3s 8ms/step

In [16]:

predicted\_values.shape

Out[16]:

(10000, 10)

In [17]:

**import** random

n **=** random.randint(0,9999)

plt.figure(figsize**=**(10,10))

plt.imshow(test\_images[n])

plt.yticks([])

plt.xticks([])

plt.grid(**False**)

plt.title(class\_names[np.argmax(predicted\_values[n])])

Out[17]:

Text(0.5, 1.0, 'dog')

A white dog with its mouth open

Description automatically generated

In [18]:

plt.plot(history.history['accuracy'])

plt.plot(history.history['val\_accuracy'])

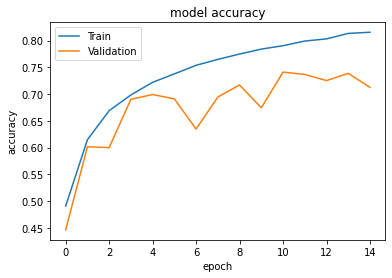
plt.title('model accuracy')

plt.ylabel('accuracy')

plt.xlabel('epoch')

plt.legend(['Train', 'Validation'], loc**=**'upper left')

plt.show()



In [ ]:

​