

I:\NewProject2022\PG\BishopData\KerasCnn\venv\Scripts\python.exe I:/NewProject2022/PG/BishopData/KerasCnn/Main.py

2022-05-10 09:28:08.059313: W tensorflow/stream\_executor/platform/default/dso\_loader.cc:64] Could not load dynamic library 'cudart64\_110.dll'; dlerror: cudart64\_110.dll not found

2022-05-10 09:28:08.059569: I tensorflow/stream\_executor/cuda/cudart\_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.

type-15

After saving image:

I:/NewProject2022/PG/BishopData/KerasCnn/Data/type-15/15-11.jpg

15-11.jpg

Found 1505 images belonging to 9 classes.

2022-05-10 09:30:48.658363: W tensorflow/stream\_executor/platform/default/dso\_loader.cc:64] Could not load dynamic library 'nvcuda.dll'; dlerror: nvcuda.dll not found

2022-05-10 09:30:48.659060: W tensorflow/stream\_executor/cuda/cuda\_driver.cc:269] failed call to cuInit: UNKNOWN ERROR (303)

2022-05-10 09:30:48.685063: I tensorflow/stream\_executor/cuda/cuda\_diagnostics.cc:169] retrieving CUDA diagnostic information for host: DESKTOP-9BF8NUN

2022-05-10 09:30:48.685919: I tensorflow/stream\_executor/cuda/cuda\_diagnostics.cc:176] hostname: DESKTOP-9BF8NUN

2022-05-10 09:30:48.736807: I tensorflow/core/platform/cpu\_feature\_guard.cc:151] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations: AVX AVX2

To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.

Model: "sequential"

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

Layer (type) Output Shape Param #

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

conv2d (Conv2D) (None, 198, 198, 16) 448

max\_pooling2d (MaxPooling2D (None, 99, 99, 16) 0

)

conv2d\_1 (Conv2D) (None, 97, 97, 32) 4640

max\_pooling2d\_1 (MaxPooling (None, 48, 48, 32) 0

2D)

conv2d\_2 (Conv2D) (None, 46, 46, 64) 18496

max\_pooling2d\_2 (MaxPooling (None, 23, 23, 64) 0

2D)

conv2d\_3 (Conv2D) (None, 21, 21, 64) 36928

max\_pooling2d\_3 (MaxPooling (None, 10, 10, 64) 0

2D)

conv2d\_4 (Conv2D) (None, 8, 8, 64) 36928

max\_pooling2d\_4 (MaxPooling (None, 4, 4, 64) 0

2D)

flatten (Flatten) (None, 1024) 0

dense (Dense) (None, 128) 131200

dense\_1 (Dense) (None, 9) 1161

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

Total params: 229,801

Trainable params: 229,801

Non-trainable params: 0

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

Epoch 1/25

WARNING:tensorflow:AutoGraph could not transform <function Model.make\_train\_function.<locals>.train\_function at 0x0000025ACDF9FF78> and will run it as-is.

Please report this to the TensorFlow team. When filing the bug, set the verbosity to 10 (on Linux, `export AUTOGRAPH\_VERBOSITY=10`) and attach the full output.

Cause: 'arguments' object has no attribute 'posonlyargs'

To silence this warning, decorate the function with @tf.autograph.experimental.do\_not\_convert

2022-05-10 09:30:52.775250: W tensorflow/core/framework/cpu\_allocator\_impl.cc:82] Allocation of 80289792 exceeds 10% of free system memory.

2022-05-10 09:30:54.013025: W tensorflow/core/framework/cpu\_allocator\_impl.cc:82] Allocation of 80289792 exceeds 10% of free system memory.

1/47 [..............................] - ETA: 2:25 - loss: 2.1536 - accuracy: 0.25002022-05-10 09:30:54.200828: W tensorflow/core/framework/cpu\_allocator\_impl.cc:82] Allocation of 80289792 exceeds 10% of free system memory.

2022-05-10 09:30:54.719541: W tensorflow/core/framework/cpu\_allocator\_impl.cc:82] Allocation of 80289792 exceeds 10% of free system memory.

2022-05-10 09:30:54.936892: W tensorflow/core/framework/cpu\_allocator\_impl.cc:82] Allocation of 80289792 exceeds 10% of free system memory.

47/47 [==============================] - 32s 622ms/step - loss: 2.0977 - accuracy: 0.2492

Epoch 2/25

47/47 [==============================] - 25s 540ms/step - loss: 1.9535 - accuracy: 0.2865

Epoch 3/25

47/47 [==============================] - 26s 544ms/step - loss: 1.4938 - accuracy: 0.4807

Epoch 4/25

47/47 [==============================] - 26s 547ms/step - loss: 1.0026 - accuracy: 0.6544

Epoch 5/25

47/47 [==============================] - 26s 549ms/step - loss: 0.5988 - accuracy: 0.8086

Epoch 6/25

47/47 [==============================] - 28s 599ms/step - loss: 0.3459 - accuracy: 0.8859

Epoch 7/25

47/47 [==============================] - 28s 598ms/step - loss: 0.2855 - accuracy: 0.9117

Epoch 8/25

47/47 [==============================] - 28s 597ms/step - loss: 0.1721 - accuracy: 0.9498

Epoch 9/25

47/47 [==============================] - 26s 551ms/step - loss: 0.1332 - accuracy: 0.9620

Epoch 10/25

47/47 [==============================] - 26s 543ms/step - loss: 0.0638 - accuracy: 0.9817

Epoch 11/25

47/47 [==============================] - 26s 549ms/step - loss: 0.1346 - accuracy: 0.9783

Epoch 12/25

47/47 [==============================] - 26s 545ms/step - loss: 0.0768 - accuracy: 0.9762

Epoch 13/25

47/47 [==============================] - 26s 543ms/step - loss: 0.0657 - accuracy: 0.9830

Epoch 14/25

47/47 [==============================] - 26s 545ms/step - loss: 0.0580 - accuracy: 0.9885

Epoch 15/25

47/47 [==============================] - 26s 542ms/step - loss: 0.0557 - accuracy: 0.9919

Epoch 16/25

47/47 [==============================] - 26s 544ms/step - loss: 0.0511 - accuracy: 0.9885

Epoch 17/25

47/47 [==============================] - 26s 542ms/step - loss: 0.0934 - accuracy: 0.9857

Epoch 18/25

47/47 [==============================] - 26s 543ms/step - loss: 0.0042 - accuracy: 0.9993

Epoch 19/25

47/47 [==============================] - 26s 553ms/step - loss: 0.0615 - accuracy: 0.9885

Epoch 20/25

47/47 [==============================] - 26s 543ms/step - loss: 0.0638 - accuracy: 0.9837

Epoch 21/25

47/47 [==============================] - 25s 542ms/step - loss: 4.5464e-04 - accuracy: 1.0000

Epoch 22/25

47/47 [==============================] - 26s 543ms/step - loss: 2.8777e-05 - accuracy: 1.0000

Epoch 23/25

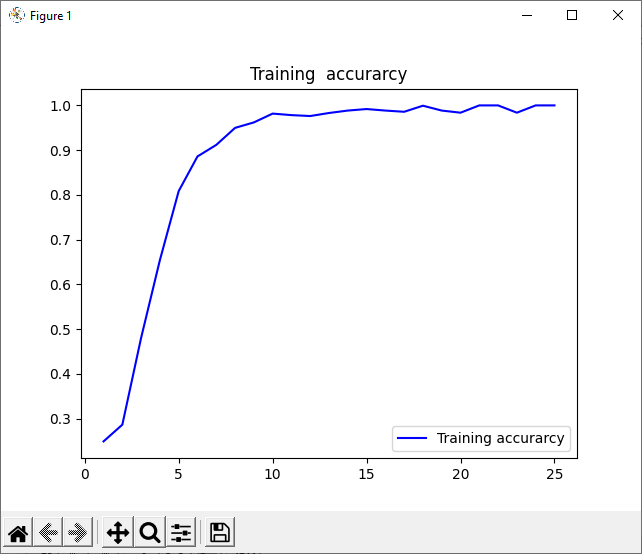
47/47 [==============================] - 26s 546ms/step - loss: 0.1502 - accuracy: 0.9837

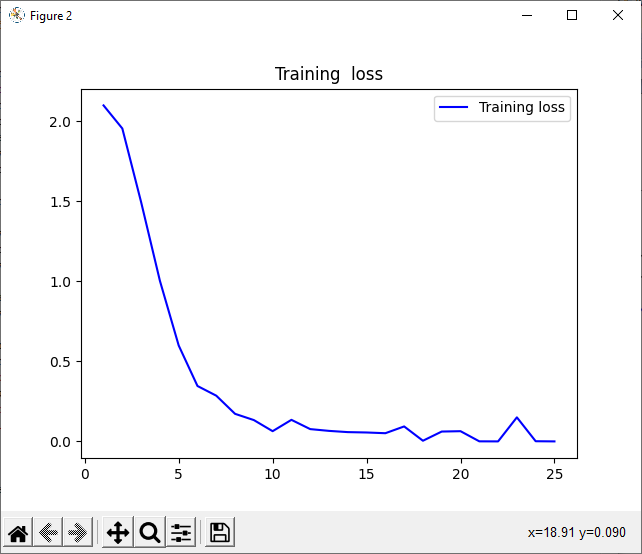
Epoch 24/25

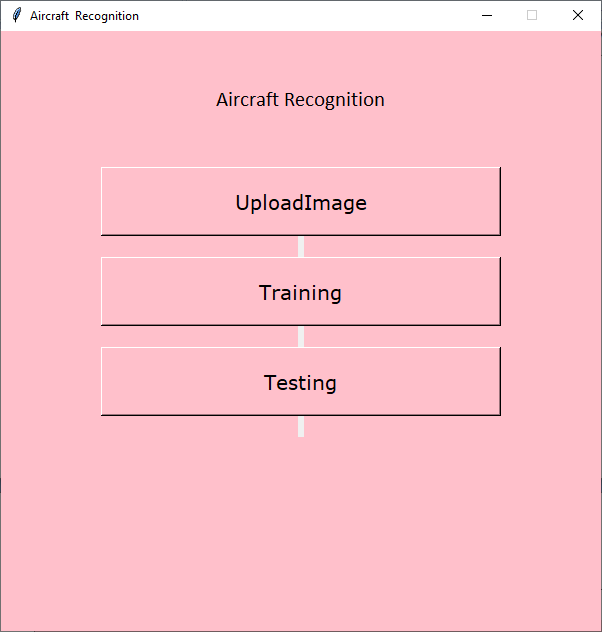
47/47 [==============================] - 26s 552ms/step - loss: 0.0011 - accuracy: 1.0000

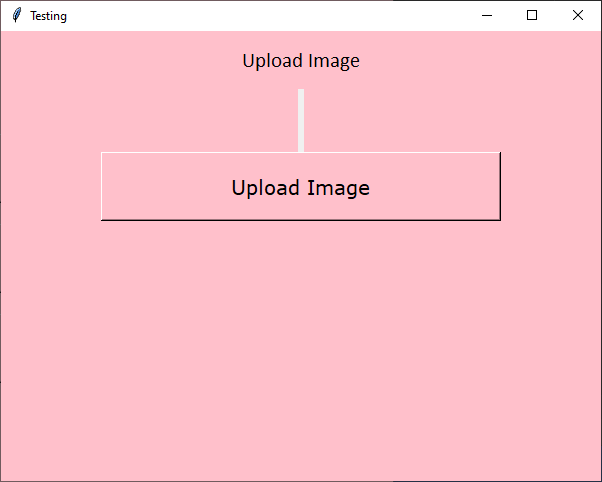
Epoch 25/25

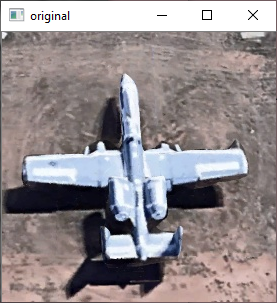
47/47 [==============================] - 26s 543ms/step - loss: 2.8834e-05 - accuracy: 1.0000



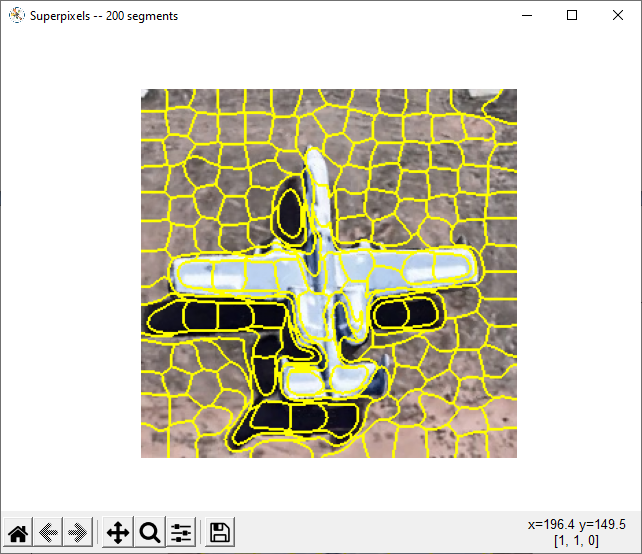
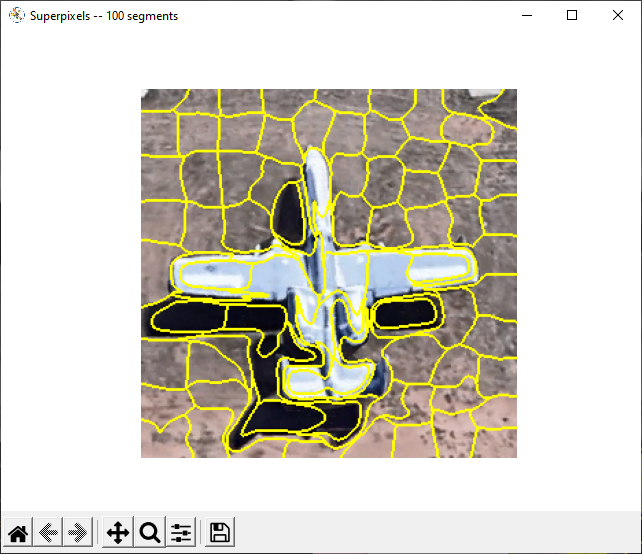


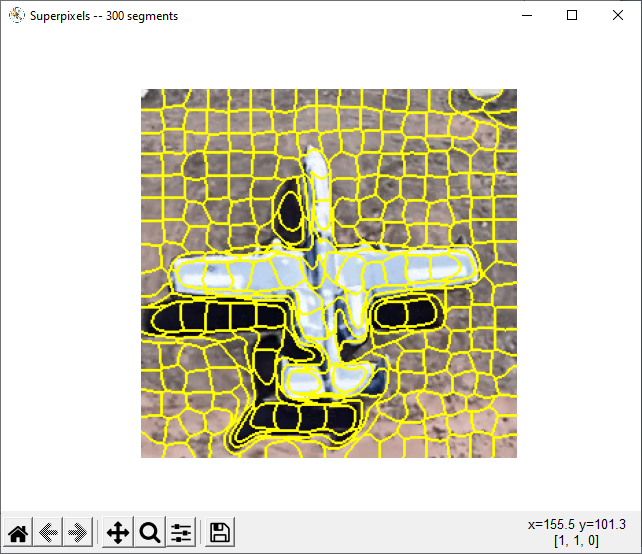


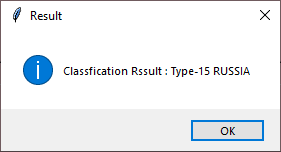












import tensorflow as tf  
import numpy as np  
  
from tkinter import \*  
import os  
from tkinter import filedialog  
import cv2  
import time  
from matplotlib import pyplot as plt  
from tkinter import messagebox  
  
import smtplib  
  
  
  
def endprogram():  
 print ("\nProgram terminated!")  
 sys.exit()  
  
  
  
  
  
  
def file\_sucess():  
 global file\_success\_screen  
 file\_success\_screen = Toplevel(training\_screen)  
 file\_success\_screen.title("File Upload Success")  
 file\_success\_screen.geometry("150x100")  
 file\_success\_screen.configure(bg='pink')  
 Label(file\_success\_screen, text="File Upload Success").pack()  
 Button(file\_success\_screen, text='''ok''', font=(  
 'Verdana', 15), height="2", width="30", bg='pink').pack()  
  
  
global ttype  
  
def training():  
 global training\_screen  
  
 global clicked  
  
 training\_screen = Toplevel(main\_screen)  
 training\_screen.title("Training")  
 # login\_screen.geometry("400x300")  
 training\_screen.geometry("600x450+650+150")  
 training\_screen.minsize(120, 1)  
 training\_screen.maxsize(1604, 881)  
 training\_screen.resizable(1, 1)  
 training\_screen.configure(bg='pink')  
 # login\_screen.title("New Toplevel")  
  
  
  
 Label(training\_screen, text='''Upload Image ''', background="#d9d9d9", disabledforeground="#a3a3a3",  
 foreground="#000000", width="300", height="2",bg='pink', font=("Calibri", 16)).pack()  
 Label(training\_screen, text="").pack()  
  
  
 options = [  
 "type-12",  
 "type-13",  
 "type-14",  
 "type-15",  
 "type-16",  
 "type-17",  
 "type-18",  
 "type-19",  
 "type-20"  
  
 ]  
  
 # datatype of menu text  
 clicked = StringVar()  
  
  
 # initial menu text  
 clicked.set("Aircraft")  
  
 # Create Dropdown menu  
 drop = OptionMenu(training\_screen, clicked, \*options )  
 drop.config(width="30",bg='pink')  
  
 drop.pack()  
  
 ttype=clicked.get()  
  
 Button(training\_screen, text='''Upload Image''', font=(  
 'Verdana', 15), height="2", width="30",bg='pink', command=imgtraining).pack()  
  
  
  
  
def imgtraining():  
 name1 = clicked.get()  
  
 print(name1)  
  
 import\_file\_path = filedialog.askopenfilename()  
 import os  
 s = import\_file\_path  
 os.path.split(s)  
 os.path.split(s)[1]  
 splname = os.path.split(s)[1]  
  
  
 image = cv2.imread(import\_file\_path)  
 #filename = 'Test.jpg'  
 filename = 'Data/'+name1+'/'+splname  
  
  
 cv2.imwrite(filename, image)  
 print("After saving image:")  
  
 gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)  
  
 cv2.imshow('Original image', image)  
 cv2.imshow('Gray image', gray)  
 # import\_file\_path = filedialog.askopenfilename()  
 print(import\_file\_path)  
 fnm = os.path.basename(import\_file\_path)  
 print(os.path.basename(import\_file\_path))  
  
 from PIL import Image, ImageOps  
  
 im = Image.open(import\_file\_path)  
 im\_invert = ImageOps.invert(im)  
 im\_invert.save('lena\_invert.jpg', quality=95)  
 im = Image.open(import\_file\_path).convert('RGB')  
 im\_invert = ImageOps.invert(im)  
 im\_invert.save('tt.png')  
 image2 = cv2.imread('tt.png')  
 cv2.imshow("Invert", image2)  
  
 """"-----------------------------------------------"""  
  
 img = image  
  
 gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
 cv2.imshow('Original image', img)  
 #cv2.imshow('Gray image', gray)  
 #dst = cv2.fastNlMeansDenoisingColored(img, None, 10, 10, 7, 21)  
 dst = cv2.medianBlur(img, 7)  
 cv2.imshow("Nosie Removal", dst)  
  
  
  
  
def fulltraining():  
 import model as mm  
  
  
  
  
  
  
  
def testing():  
 global testing\_screen  
 testing\_screen = Toplevel(main\_screen)  
 testing\_screen.title("Testing")  
 # login\_screen.geometry("400x300")  
 testing\_screen.geometry("600x450+650+150")  
 testing\_screen.minsize(120, 1)  
 testing\_screen.maxsize(1604, 881)  
 testing\_screen.resizable(1, 1)  
 testing\_screen.configure(bg='pink')  
 # login\_screen.title("New Toplevel")  
  
 Label(testing\_screen, text='''Upload Image''', disabledforeground="#a3a3a3",  
 foreground="#000000", width="300", height="2",bg='pink', font=("Calibri", 16)).pack()  
 Label(testing\_screen, text="").pack()  
 Label(testing\_screen, text="").pack()  
 Label(testing\_screen, text="").pack()  
 Button(testing\_screen, text='''Upload Image''', font=(  
 'Verdana', 15), height="2", width="30",bg='pink', command=imgtest).pack()  
  
  
global affect  
def imgtest():  
  
  
 import\_file\_path = filedialog.askopenfilename()  
  
 image = cv2.imread(import\_file\_path)  
 print(import\_file\_path)  
 filename = 'Output/Out/Test.jpg'  
 cv2.imwrite(filename, image)  
 print("After saving image:")  
 #result()  
  
 #import\_file\_path = filedialog.askopenfilename()  
 print(import\_file\_path)  
 fnm = os.path.basename(import\_file\_path)  
 print(os.path.basename(import\_file\_path))  
  
 # file\_sucess()  
  
 print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\nImage : " + fnm + "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  
 img = cv2.imread(import\_file\_path)  
 if img is None:  
 print('no data')  
  
 img1 = cv2.imread(import\_file\_path)  
 print(img.shape)  
 img = cv2.resize(img, ((int)(img.shape[1] / 5), (int)(img.shape[0] / 5)))  
 original = img.copy()  
 neworiginal = img.copy()  
 cv2.imshow('original', img1)  
 gray = cv2.cvtColor(img1, cv2.COLOR\_BGR2GRAY)  
  
 img1S = cv2.resize(img1, (960, 540))  
  
 cv2.imshow('Original image', img1S)  
 grayS = cv2.resize(gray, (960, 540))  
 cv2.imshow('Gray image', grayS)  
  
 dst = cv2.medianBlur(img1,7)  
 cv2.imshow("Nosie Removal", dst)  
  
  
 super()  
 result()  
  
  
  
  
  
def super():  
 from skimage.segmentation import slic  
 from skimage.segmentation import mark\_boundaries  
 from skimage.util import img\_as\_float  
 from skimage import io  
 import matplotlib.pyplot as plt  
  
 filename = 'Output/Out/Test.jpg'  
  
 image = img\_as\_float(io.imread(filename))  
 # loop over the number of segments  
 for numSegments in (100, 200, 300):  
 # apply SLIC and extract (approximately) the supplied number  
 # of segments  
 segments = slic(image, n\_segments=numSegments, sigma=5)  
 # show the output of SLIC  
 fig = plt.figure("Superpixels -- %d segments" % (numSegments))  
 ax = fig.add\_subplot(1, 1, 1)  
 ax.imshow(mark\_boundaries(image, segments))  
 plt.axis("off")  
 plt.savefig(str(numSegments)+".jpg")  
 # show the plots  
 plt.show()  
  
  
  
def result():  
 import warnings  
 warnings.filterwarnings('ignore')  
  
 import tensorflow as tf  
 classifierLoad = tf.keras.models.load\_model("airmodel.h5")  
  
 import numpy as np  
 from keras.preprocessing import image  
  
 global mmmsg  
 #filename1 = "100.jpg"  
 #filename2 = "200.jpg"  
 #filename3 = "300.jpg"  
  
 #cv2.imshow('100 Superpixels', filename1)  
 # cv2.imshow("200 Superpixels", filename2)  
 # cv2.imshow("300 Superpixels", filename3)  
  
 test\_image = image.load\_img('Output/Out/Test.jpg', target\_size=(200, 200))  
 img1 = cv2.imread('Output/Out/Test.jpg')  
 # test\_image = image.img\_to\_array(test\_image)  
 test\_image = np.expand\_dims(test\_image, axis=0)  
 result = classifierLoad.predict(test\_image)  
  
 out = ''  
 pre=''  
 if result[0][0] == 1:  
  
 out = "Type-12 India"  
  
  
  
  
  
  
  
  
 elif result[0][1] == 1:  
  
 out = "Type-13 USA"  
  
 elif result[0][2] == 1:  
  
 out = "Type-14 UAE"  
 elif result[0][3] == 1:  
  
 out = "Type-15 RUSSIA"  
 elif result[0][4] == 1:  
  
 out = "Type-16 CHINA"  
 elif result[0][5] == 1:  
  
 out = "Type-17 INDONESIA"  
 elif result[0][6] == 1:  
  
 out = "Type-18 PARIS"  
 elif result[0][7] == 1:  
  
 out = "Type-19 INDIA"  
 elif result[0][8] == 1:  
  
 out = "Type-20 INDIA"  
  
  
 messagebox.showinfo("Result", "Classfication Rssult : "+str(out))  
  
  
  
 #s = smtplib.SMTP('smtp.gmail.com', 587)  
 #s.starttls()  
 #s.login("sampletest685@gmail.com", "mailtest4")  
 #message = out  
 #s.sendmail("sampletest685@gmail.com", "shabanabana944@gmail.com", message)  
 #s.quit()  
 mmmsg = out;  
  
 import smtplib  
 from email.mime.multipart import MIMEMultipart  
 from email.mime.text import MIMEText  
 from email.mime.base import MIMEBase  
 from email import encoders  
  
 fromaddr = "sampletest685@gmail.com"  
 toaddr = "shabanabana944@gmail.com"  
  
 # instance of MIMEMultipart  
 msg = MIMEMultipart()  
  
 # storing the senders email address  
 msg['From'] = fromaddr  
  
 # storing the receivers email address  
 msg['To'] = toaddr  
  
 # storing the subject  
 msg['Subject'] = "Alert"  
  
 # string to store the body of the mail  
 body = mmmsg  
  
 # attach the body with the msg instance  
 msg.attach(MIMEText(body, 'plain'))  
  
 # creates SMTP session  
 s = smtplib.SMTP('smtp.gmail.com', 587)  
  
 # start TLS for security  
 s.starttls()  
  
 # Authentication  
 s.login(fromaddr, "mailtest4")  
  
 # Converts the Multipart msg into a string  
 text = msg.as\_string()  
  
 # sending the mail  
 s.sendmail(fromaddr, toaddr, text)  
  
 # terminating the session  
 s.quit()  
  
 sendmsg("9597900586",out)  
  
  
  
  
  
  
  
  
  
  
  
  
  
def sendmsg(targetno,message):  
 import requests  
 requests.post("http://smsserver9.creativepoint.in/api.php?username=fantasy&password=596692&to=" + targetno + "&from=FSSMSS&message=Dear user your msg is " + message + " Sent By FSMSG FSSMSS&PEID=1501563800000030506&templateid=1507162882948811640")  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
def main\_account\_screen():  
 global main\_screen  
 main\_screen = Tk()  
 width = 600  
 height = 600  
 screen\_width = main\_screen.winfo\_screenwidth()  
 screen\_height = main\_screen.winfo\_screenheight()  
 x = (screen\_width / 2) - (width / 2)  
 y = (screen\_height / 2) - (height / 2)  
 main\_screen.geometry("%dx%d+%d+%d" % (width, height, x, y))  
 main\_screen.resizable(0, 0)  
 # main\_screen.geometry("300x250")  
 main\_screen.configure(bg='pink')  
 main\_screen.title("Aircraft Recognition")  
  
 Label(text="Aircraft Recognition", width="300", height="5", bg='pink', font=("Calibri", 16)).pack()  
  
 Button(text="UploadImage", font=(  
 'Verdana', 15), height="2", width="30", command=training, highlightcolor="black",bg='pink').pack(side=TOP)  
 Label(text="").pack()  
 Button(text="Training", font=(  
 'Verdana', 15), height="2", width="30", command=fulltraining, highlightcolor="black",bg='pink').pack(side=TOP)  
  
 Label(text="").pack()  
 Button(text="Testing", font=(  
 'Verdana', 15), height="2", width="30", bg='pink', command=testing).pack(side=TOP)  
  
 Label(text="").pack()  
  
 main\_screen.mainloop()  
  
  
main\_account\_screen()

# Part 1 - Building the CNN  
  
# Importing the Keras libraries and packages  
from keras.models import Sequential  
from keras.layers import Convolution2D  
from keras.layers import MaxPooling2D  
from keras.layers import Flatten  
from keras.layers import Dense  
from keras.models import model\_from\_json  
import matplotlib.pyplot as plt  
import warnings  
warnings.filterwarnings('ignore')  
batch\_size = 32  
  
from tensorflow.keras.preprocessing.image import ImageDataGenerator  
  
# All images will be rescaled by 1./255  
train\_datagen = ImageDataGenerator(rescale=1/255)  
  
# Flow training images in batches of 128 using train\_datagen generator  
train\_generator = train\_datagen.flow\_from\_directory(  
 'Data', # This is the source directory for training images  
 target\_size=(200, 200), # All images will be resized to 200 x 200  
 batch\_size=batch\_size,  
 # Specify the classes explicitly  
 classes = ['type-12','type-13','type-14','type-15','type-16','type-17','type-18','type-19','type-20'],  
 # Since we use categorical\_crossentropy loss, we need categorical labels  
 class\_mode='categorical')  
  
import tensorflow as tf  
  
model = tf.keras.models.Sequential([  
 # Note the input shape is the desired size of the image 200x 200 with 3 bytes color  
 # The first convolution  
 tf.keras.layers.Conv2D(16, (3,3), activation='relu', input\_shape=(200, 200, 3)),  
 tf.keras.layers.MaxPooling2D(2, 2),  
 # The second convolution  
 tf.keras.layers.Conv2D(32, (3,3), activation='relu'),  
 tf.keras.layers.MaxPooling2D(2,2),  
 # The third convolution  
 tf.keras.layers.Conv2D(64, (3,3), activation='relu'),  
 tf.keras.layers.MaxPooling2D(2,2),  
 # The fourth convolution  
 tf.keras.layers.Conv2D(64, (3,3), activation='relu'),  
 tf.keras.layers.MaxPooling2D(2,2),  
 # The fifth convolution  
 tf.keras.layers.Conv2D(64, (3,3), activation='relu'),  
 tf.keras.layers.MaxPooling2D(2,2),  
 # Flatten the results to feed into a dense layer  
 tf.keras.layers.Flatten(),  
 # 128 neuron in the fully-connected layer  
 tf.keras.layers.Dense(128, activation='relu'),  
 # 5 output neurons for 5 classes with the softmax activation  
 tf.keras.layers.Dense(9, activation='softmax')  
])  
  
model.summary()  
  
from tensorflow.keras.optimizers import RMSprop  
early = tf.keras.callbacks.EarlyStopping(monitor='val\_loss',patience=5)  
model.compile(loss='categorical\_crossentropy',  
 optimizer=RMSprop(lr=0.001),  
 metrics=['accuracy'])  
  
total\_sample=train\_generator.n  
  
n\_epochs = 25  
  
history = model.fit\_generator(  
 train\_generator,  
 steps\_per\_epoch=int(total\_sample/batch\_size),  
 epochs=n\_epochs,  
 verbose=1)  
  
  
  
  
model.save('airmodel.h5')  
  
  
  
acc = history.history['accuracy']  
  
loss = history.history['loss']  
  
epochs = range(1, len(acc) + 1)  
  
# Train and validation accuracy  
plt.plot(epochs, acc, 'b', label='Training accurarcy')  
  
plt.title('Training accurarcy')  
plt.legend()  
  
plt.figure()  
  
# Train and validation loss  
plt.plot(epochs, loss, 'b', label='Training loss')  
plt.title('Training loss')  
plt.legend()  
plt.show()