Date	08 November 2022
Team ID	PNT2022TMID37911
Project Name	AI Based Localization and Classification of Skin Disease with Erythema

```
#import random

#import cv2

 #from keras.preprocessing import image

 #import scipy.misc as sm

 #from keras.utils import to_categorical from

 keras.models import Model from keras.layers import

 Dense, GlobalAveragePooling2D from keras.optimizers

 import SGD#, Adam

 from keras.applications.resnet50 import ResNet50 from
 keras.preprocessing.image import ImageDataGenerator

 #import numpy as np

 #import os

 #from matplotlib import pyplot

 #from sklearn.preprocessing import LabelEncoder

 # from keras.preprocessing.image import flow_from_directory

 #from keras.preprocessing.image import img_to_array

 #from sklearn.preprocessing import LabelBinarizer

 #from sklearn.model_selection import train_test_split

 #import matplotlib.pyplot as plt

 #from imutils import paths

 #import scipy.misc as sm

 #from keras.models import model_from_json

 data = ['C:/Users/ankur/.spyder-py3/autosave/data']

 labels = []

 IMAGE_DIMS = (224,224,3) print("1”)

"""count=0

ls1=os.listdir('color1')

dic1={} for idx,i in

enumerate(ls1): dic1[i]=idx
```

```
ls2=os.listdir('color1/'+i)
for j in ls2:
      #im1=np.asarray(sm.imread('color/'+i+'/'+j))
      #temp=np.zeros((len(im1),len(im1[0]),len(im1[0][0]) ))
 count=count+1 print(count) print(dic1)
 X=np.zeros((count,224,224,3)
 ) Y=np.zeros((count,1)) vap=0
 for idx,i in enumerate(ls1):
 dic1[i]=idx
 ls2=os.listdir('color1/'+i) for j
 in ls2:
 img = image.load_img('color1/'+i+'/'+j, target_size=(224,
 224)) #im1=np.asarray(sm.imread('color1/'+i+'/'+j)) img =
 image.img_to_array(img) print(img[0]) print(img.shape)
 #X[vap,:,:,:]=im1 #Y[vap,0]=idx vap=vap+1
 """
 # imagePaths = sorted(list(paths.list_images("color")))
 # i=0 #
 print("2")

 # for imagePath in imagePaths:
 # load the image, pre-process it, and store it in the data list
 # img = image.load_img(imagePath,target_size=(224,224))
 # img = img_to_array(img)
 # data.append(img)
 # """im0=np.asarray(image)
 # data[i,:,:,:]=im0"""
 # extract set of class labels from the image path and update the
 # labels list
 # l = label = imagePath.split(os.path.sep)[-2]
 # labels.append(l)
 # print("3")
 # data = np.array(data, dtype="float") / 255.0
 # ltb=labels = np.array(labels)
 # print(labels[16])
 # lb = LabelBinarizer() # labels =
 lb.fit_transform(labels)

 """ train_labels =
 os.listdir("color") le =
 LabelEncoder()
 le.fit([tl for tl in train_labels])
 le = LabelEncoder() le_labels =
 le.fit_transform(ltb)
 """
 # (trainX, testX, trainY, testY) = train_test_split(data,
 # labels, test_size=0.3, random_state=42)


 # print("4")
 # print(trainX.shape)
 """" ind_train =
 random.sample(list(range(trainX.shape[0])), 20) trainX =
 trainX[ind_train] trainY = trainY[ind_train]
 # test data ind_test =
 random.sample(list(range(testX.shape[0])), 5) testX =
 testX[ind_test] testY = testY[ind_test] def
 resize_data(data):
   data_upscaled = np.zeros((data.shape[0], 320, 320, 3)) for i, img in
 enumerate(data): large_img = cv2.resize(img, dsize=(320, 320),
 interpolation=cv2.INTER_CUBIC) data_upscaled[i] = large_img return
```

```
data_upscaled # resize train and test data x_train_resized = resize_data(trainX)
x_test_resized = resize_data(testX)
"""

# y_train_hot_encoded = to_categorical(trainY)

# y_test_hot_encoded = to_categorical(testY)
"""for i in range(0,len(trainY)):
print(y_train_hot_encoded[i])
print("\n") """

aug = ImageDataGenerator(rotation_range=25, width_shift_range=0.1,
height_shift_range=0.1,     shear_range=0.2,     zoom_range=0.2,
horizontal_flip=True, fill_mode="nearest")



train_generator=aug.flow_from_directory(
[10:35, 11/8/2022] Irin: directory=r"C:/Users/ankur/.spyder-py3/autosave/data/train",
target_size=(224,224), color_mode="rgb", batch_size=64, class_mode="categorical",
shuffle=True, seed=None
)
valid_generator=aug.flow_from_directory(
directory=r"C:/Users/ankur/.spyder-
py3/autosave/data/test", target_size=(224,224),
color_mode="rgb", batch_size=64,
class_mode="categorical",
shuffle=True,
seed=None
)
```

```
def model(base_model):


print("5")

# get layers and add average pooling layer

x = base_model.output x =

GlobalAveragePooling2D()(x)


# add fully-connected layer

x = Dense(512, activation='relu')(x)

# add output layer

predictions = Dense(7, activation='softmax')(x)

model = Model(inputs=base_model.input, outputs=predictions)

# fname = "weights.hdf5"

# model.load_weights(fname) # freeze

pre-trained model area's layer for layer

in base_model.layers: layer.trainable =

False

# update the weight that are added

# model.compile(optimizer='rmsprop', loss='categorical_crossentropy')

# model.fit(x_train, y_train,epochs=4)
# choose the layers which are updated by training

layer_num = len(model.layers) print(layer_num,"

number of layers") for layer in

model.layers[:int(layer_num * 0.7)]:

layer.trainable = False


for layer in model.layers[int(layer_num * 0.7):]:

layer.trainable = True


# update the weights
```

```
model.compile(optimizer=SGD(lr=1e-4,decay=1e-6, momentum=0.9), loss='categorical_crossentropy',
metrics=['accuracy'])

"""history = model.fit_generator(

aug.flow(x_train,           y_train,),

validation_data=(testX,      testY),

steps_per_epoch=len(trainX),

epochs=5, verbose=1)"""

STEP_SIZE_TRAIN=train_generator.n//train_generator.batch_size

STEP_SIZE_VALID=valid_generator.n//valid_generator.batch_size

history=model.fit_generator(generator=train_generator,

steps_per_epoch=STEP_SIZE_TRAIN,

validation_data=valid_generator, validation_steps=STEP_SIZE_VALID,

# use_multiprocessing=True,

# workers=3,

# verbose=2,

epochs=100

)

# print(model.evaluate_generator(generator=valid_generator))

model_json = model.to_json() with open("C:/Users/ankur/.spyder-

py3/autosave/model.json", "w") as json_file:

json_file.write(model_json) # serialize weights to HDF5

model.save_weights("model.h5") print("Saved model to

disk") fname="C:/Users/ankur/.spyder-

py3/autosave/weights1.hdf5"

model.save_weights(fname,overwrite=True)


# prediction

#img =
image.load_img(r'C:\Users\WASD\Desktop\hoga\color\Pepper,bell_Bacterial_spot\29.jpg',target_size=
(224,224))

#img = image.img_to_array(img)

#img=np.expand_dims(img,axis=0)
```

```
 #predictedclass = model.predict(img)

 # print(train_generator.class_indices)

 # predictedclass


 #for i in train_generator.class_indices:

#          if train_generator.class_indices[i] == np.argmax(predictedclass):

#                   print(i)

#                   break




 # history = model.fit(x_train, y_train, epochs=7,batch_size=10)i

 # pyplot.plot(history.history['loss'])

 # pyplot.plot(history.history['val_loss'])

 # pyplot.title('model train vs validation loss')

 # pyplot.ylabel('loss')

 # pyplot.xlabel('epoch')

 # pyplot.legend(['train', 'validation'], loc='upper right')

 # pyplot.show()


 # print(model.summary())


 return history





 res_50_model = ResNet50(weights='imagenet', include_top=False)

 history=model(res_50_model)
```