**import** os

**import** numpy **as** np *# linear algebra*

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

*# Dl framwork - tensorflow, keras a backend*

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

**import** tensorflow.keras.backend **as** K

**from** tensorflow.keras.models **import** Model, Sequential

**from** tensorflow.keras.layers **import** Input, Dense, Flatten, Dropout, BatchNormalization

**from** tensorflow.keras.layers **import** Conv2D, SeparableConv2D, MaxPool2D, LeakyReLU, Activation

**from** tensorflow.keras.optimizers **import** Adam

**from** tensorflow.keras.preprocessing.image **import** ImageDataGenerator

**from** tensorflow.keras.callbacks **import** ModelCheckpoint, ReduceLROnPlateau, EarlyStopping

**from** IPython.display **import** display

**from** os **import** listdir

**from** os.path **import** isfile, join

**from** PIL **import** Image

**import** glob

**from** tensorflow.keras.preprocessing.image **import** ImageDataGenerator

**from** tensorflow.keras.layers **import** Convolution2D

**from** tensorflow.keras.layers **import** MaxPooling2D

**from** tensorflow.keras.layers **import** Flatten

**from** tensorflow.keras.layers **import** Dense

**import** warnings

warnings**.**filterwarnings('ignore')

dir\_name\_train\_Primary\_Endo\_with\_Secondary\_Perio **=** 'dataset/Train/Primary Endo with Secondary Perio'

dir\_name\_train\_Primary\_Periodontal\_Lesion **=** 'dataset/Train/Primary Periodontal Lesion'

dir\_name\_train\_True\_Combined\_Lesions **=** 'dataset/Train/True Combined Lesions'

**def** plot\_images(item\_dir, n**=**6):

all\_item\_dir **=** os**.**listdir(item\_dir)

item\_files **=** [os**.**path**.**join(item\_dir, file) **for** file **in** all\_item\_dir][:n]

plt**.**figure(figsize**=**(80, 40))

**for** idx, img\_path **in** enumerate(item\_files):

plt**.**subplot(7, n, idx**+**1)

img **=** plt**.**imread(img\_path)

plt**.**imshow(img, cmap**=**'gray')

plt**.**axis('off')

plt**.**tight\_layout()

**def** Images\_details\_Print\_data(data, path):

print(" ====== Images in: ", path)

**for** k, v **in** data**.**items():

print("%s:\t%s" **%** (k, v))

**def** Images\_details(path):

files **=** [f **for** f **in** glob**.**glob(path **+** "\*\*/\*.\*", recursive**=True**)]

data **=** {}

data['images\_count'] **=** len(files)

data['min\_width'] **=** 10**\*\***100 *# No image will be bigger than that*

data['max\_width'] **=** 0

data['min\_height'] **=** 10**\*\***100 *# No image will be bigger than that*

data['max\_height'] **=** 0

**for** f **in** files:

im **=** Image**.**open(f)

width, height **=** im**.**size

data['min\_width'] **=** min(width, data['min\_width'])

data['max\_width'] **=** max(width, data['max\_height'])

data['min\_height'] **=** min(height, data['min\_height'])

data['max\_height'] **=** max(height, data['max\_height'])

Images\_details\_Print\_data(data, path)

print("")

print("Trainned data for Primary Endo with Secondary Perio:")

print("")

Images\_details(dir\_name\_train\_Primary\_Endo\_with\_Secondary\_Perio)

print("")

plot\_images(dir\_name\_train\_Primary\_Endo\_with\_Secondary\_Perio, 10)

print("")

print("Trainned data for Primary Periodontal Lesion:")

print("")

Images\_details(dir\_name\_train\_Primary\_Periodontal\_Lesion)

print("")

plot\_images(dir\_name\_train\_Primary\_Periodontal\_Lesion, 10)

print("")

print("Trainned data for True Combined Lesions:")

print("")

Images\_details(dir\_name\_train\_True\_Combined\_Lesions)

print("")

plot\_images(dir\_name\_train\_True\_Combined\_Lesions, 10)

Classifier**=**Sequential()

Classifier**.**add(Convolution2D(32,(3,3),input\_shape**=**(128,128,3),activation**=**'relu'))

Classifier**.**add(MaxPooling2D(pool\_size**=**(2,2)))

Classifier**.**add(Flatten())

Classifier**.**add(Dense(38, activation**=**'relu'))

Classifier**.**add(Dense(3, activation**=**'softmax'))

Classifier**.**compile(optimizer**=**'rmsprop',loss**=**'categorical\_crossentropy',metrics**=**['accuracy'])

train\_datagen**=**ImageDataGenerator(rescale**=**1.**/**255,shear\_range**=**0.2,zoom\_range**=**0.2,horizontal\_flip**=True**)

test\_datagen**=**ImageDataGenerator(rescale**=**1.**/**255)

training\_set**=**train\_datagen**.**flow\_from\_directory('dataset/Train',target\_size**=**(128,128),batch\_size**=**32,class\_mode**=**'categorical')

test\_set**=**test\_datagen**.**flow\_from\_directory('dataset/Test',target\_size**=**(128,128),batch\_size**=**32,class\_mode**=**'categorical')

epochs **=** 10

batch\_size **=** 32

*#### Fitting the model*

history **=** Classifier**.**fit\_generator(

training\_set, steps\_per\_epoch**=**training\_set**.**samples **//** batch\_size,

epochs**=**epochs,

validation\_data**=**test\_set,validation\_steps**=**test\_set**.**samples **//** batch\_size)

**def** graph():

*#Plot training & validation accuracy values*

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

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

plt**.**title('Model accuracy')

plt**.**ylabel('Accuracy')

plt**.**xlabel('Epoch')

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

plt**.**show()

*# Plot training & validation loss values*

plt**.**plot(history**.**history['loss'])

plt**.**plot(history**.**history['val\_loss'])

plt**.**title('Model loss')

plt**.**ylabel('Loss')

plt**.**xlabel('Epoch')

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

plt**.**show()

graph()

**MODULE - 2**

*# Dl framwork - tensorflow, keras a backend*

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

**import** tensorflow.keras.backend **as** K

**from** tensorflow.keras.models **import** Model

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

**from** tensorflow.keras.layers **import** Input

**from** tensorflow.keras.layers **import** Dense

**from** tensorflow.keras.layers **import** Flatten

**from** tensorflow.keras.layers **import** Conv2D

**from** tensorflow.keras.layers **import** MaxPooling2D

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

**from** tensorflow.keras.layers **import** LeakyReLU

**from** tensorflow.keras.layers **import** Activation

**from** tensorflow.keras.optimizers **import** Adam

**from** tensorflow.keras.preprocessing.image **import** ImageDataGenerator

**from** tensorflow.keras.callbacks **import** ModelCheckpoint

**from** tensorflow.keras.callbacks **import** ReduceLROnPlateau

**from** tensorflow.keras.callbacks **import** EarlyStopping

**import** warnings

warnings**.**filterwarnings('ignore')

model **=** Sequential()

*# 1st Convolutional Layer*

model**.**add(Conv2D(filters**=**96, input\_shape**=**(224,224,3), kernel\_size**=**(11,11), strides**=**(4,4), padding**=**'valid'))

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

*# Max Pooling*

model**.**add(MaxPooling2D(pool\_size**=**(2,2), strides**=**(2,2), padding**=**'valid'))

*# 2nd Convolutional Layer*

model**.**add(Conv2D(filters**=**256, kernel\_size**=**(11,11), strides**=**(1,1), padding**=**'valid'))

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

*# Max Pooling*

model**.**add(MaxPooling2D(pool\_size**=**(2,2), strides**=**(2,2), padding**=**'valid'))

*# 3rd Convolutional Layer*

model**.**add(Conv2D(filters**=**384, kernel\_size**=**(3,3), strides**=**(1,1), padding**=**'valid'))

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

*# 4th Convolutional Layer*

model**.**add(Conv2D(filters**=**384, kernel\_size**=**(3,3), strides**=**(1,1), padding**=**'valid'))

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

*# 5th Convolutional Layer*

model**.**add(Conv2D(filters**=**256, kernel\_size**=**(3,3), strides**=**(1,1), padding**=**'valid'))

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

*# Max Pooling*

model**.**add(MaxPooling2D(pool\_size**=**(2,2), strides**=**(2,2), padding**=**'valid'))

*# Passing it to a Fully Connected layer*

model**.**add(Flatten())

*# 1st Fully Connected Layer*

model**.**add(Dense(4096, input\_shape**=**(224**\***224**\***3,)))

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

*# Add Dropout to prevent overfitting*

model**.**add(Dropout(0.4))

*# 2nd Fully Connected Layer*

model**.**add(Dense(4096))

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

*# Add Dropout*

model**.**add(Dropout(0.4))

*# 3rd Fully Connected Layer*

model**.**add(Dense(1000))

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

*# Add Dropout*

model**.**add(Dropout(0.4))

*# Output Layer*

model**.**add(Dense(3))

model**.**add(Activation('softmax'))

model**.**summary()

*# Compile the model*

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

train\_datagen**=**ImageDataGenerator(rescale**=**1.**/**255,shear\_range**=**0.2,zoom\_range**=**0.2,horizontal\_flip**=True**)

test\_datagen**=**ImageDataGenerator(rescale**=**1.**/**255)

training\_set**=**train\_datagen**.**flow\_from\_directory('dataset/Train',target\_size**=**(224,224),batch\_size**=**32,class\_mode**=**'categorical')

test\_set**=**test\_datagen**.**flow\_from\_directory('dataset/Test',target\_size**=**(224,224),batch\_size**=**32,class\_mode**=**'categorical')

epochs **=** 10

batch\_size **=** 32

*#### Fitting the model*

history **=** model**.**fit(

training\_set, steps\_per\_epoch**=**training\_set**.**samples **//** batch\_size,

epochs**=**epochs,

validation\_data**=**test\_set,validation\_steps**=**test\_set**.**samples **//** batch\_size)

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

**def** graph():

*#Plot training & validation accuracy values*

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

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

plt**.**title('Model accuracy')

plt**.**ylabel('Accuracy')

plt**.**xlabel('Epoch')

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

plt**.**show()

*# Plot training & validation loss values*

plt**.**plot(history**.**history['loss'])

plt**.**plot(history**.**history['val\_loss'])

plt**.**title('Model loss')

plt**.**ylabel('Loss')

plt**.**xlabel('Epoch')

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

plt**.**show()

graph()

print("[INFO] Calculating model accuracy")

scores **=** model**.**evaluate(test\_set)

print(f"Test Accuracy: {scores[1]**\***100}")

**MODULE – 3**

**from** tensorflow.keras.callbacks **import** ModelCheckpoint, ReduceLROnPlateau, EarlyStopping

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

**from** tensorflow.keras.layers **import** Convolution2D

**from** tensorflow.keras.layers **import** MaxPooling2D

**from** tensorflow.keras.layers **import** Dense

**import** warnings

warnings**.**filterwarnings('ignore')

Classifier**=**Sequential()

Classifier**.**add(Convolution2D(32,3,3,input\_shape**=**(225,225,3),activation**=**'relu'))

Classifier**.**add(MaxPooling2D(pool\_size**=**(2,2)))

Classifier**.**add(Convolution2D(128,3,3,activation**=**'relu'))

Classifier**.**add(MaxPooling2D(pool\_size**=**(2,2)))

Classifier**.**add(Flatten())

Classifier**.**add(Dense(256, activation**=**'relu'))

Classifier**.**add(Dense(3, activation**=**'softmax'))

Classifier**.**compile(optimizer**=**'rmsprop',loss**=**'categorical\_crossentropy',metrics**=**['accuracy'])

Classifier**.**summary()

**from** tensorflow.keras.preprocessing.image **import** ImageDataGenerator

train\_datagen**=**ImageDataGenerator(rescale**=**1.**/**255,shear\_range**=**0.2,zoom\_range**=**0.2,horizontal\_flip**=True**)

test\_datagen**=**ImageDataGenerator(rescale**=**1.**/**255)

training\_set**=**train\_datagen**.**flow\_from\_directory('dataset/Train',target\_size**=**(225,225),batch\_size**=**32,class\_mode**=**'categorical')

test\_set**=**test\_datagen**.**flow\_from\_directory('dataset/Test',target\_size**=**(225,225),batch\_size**=**32,class\_mode**=**'categorical')

epochs **=** 20

batch\_size **=** 32

Classifier**.**fit\_generator( training\_set, steps\_per\_epoch**=**training\_set**.**samples **//** batch\_size,

epochs**=**epochs,

validation\_data**=**test\_set,validation\_steps**=**test\_set**.**samples **//** batch\_size)

**import** h5py

Classifier**.**save('dental.h5')

**from** keras.models **import** load\_model

model**=**load\_model('dental.h5')

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

**from** tensorflow.keras.preprocessing **import** image

test\_image**=**image**.**load\_img('Primary Periodontal Lesion.jpg',target\_size**=**(225,225))

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

img **=** plt**.**imshow(test\_image)

test\_image**=**image**.**img\_to\_array(test\_image)

test\_image**=**np**.**expand\_dims(test\_image,axis**=**0)

result**=**model**.**predict(test\_image)

result

prediction **=** result[0]

classes**=**training\_set**.**class\_indices

classes

prediction**=**list(prediction)

prediction

classes**=**['Primary Endo with Secondary Perio','Primary Periodontal Lesion','True Combined Lesions']

output**=**zip(classes,prediction)

output**=**dict(output)

output

**if** output['Primary Endo with Secondary Perio']**==**1.0 :

print('Primary Endo with Secondary Perio')

**elif** output['Primary Periodontal Lesion']**==**1.0:

print('Primary Periodontal Lesion')

**elif** output['True Combined Lesions']**==**1.0:

print("True Combined Lesions")

**PyCharm:**

**Views.py**

from django.shortcuts import render

from django.http import HttpResponseRedirect

from django.urls import reverse\_lazy

from django.views.generic import TemplateView

from employee.forms import EmployeeForm

from django.views.generic import DetailView

from employee.models import Employee

class EmployeeImage(TemplateView):

    form = EmployeeForm

    template\_name = 'emp\_image.html'

    def post(self, request, \*args, \*\*kwargs):

        form = EmployeeForm(request.POST, request.FILES)

        if form.is\_valid():

            obj = form.save()

            return HttpResponseRedirect(reverse\_lazy('emp\_image\_display', kwags={'pk': obj.id}))

        context = self.get\_context\_data(form=form)

        return self.render\_to\_response(context)

    def get(self, request, \*args, \*\*kwargs):

        return self.post(request, \*args, \*\*kwargs)

class EmpImageDisplay(DetailView):

    model = Employee

    template\_name = 'emp\_image\_display.html'

    context\_object\_name = 'emp'

def cancer(request):

    result1 = Employee.objects.latest('id')

    import numpy as np

    import tensorflow as tf

    from tensorflow import keras

    import h5py

    models = keras.models.load\_model('C:/Users/SPIRO/Desktop/own/2.WORKING/Periapical\_Xrays/Deploy/employee/dental.h5')

    from tensorflow.keras.preprocessing import image

    test\_image = image.load\_img('C:/Users/SPIRO/Desktop/own/2.WORKING/Periapical\_Xrays/Deploy/media/' + str(result1),

                                target\_size=(225, 225))

    test\_image = image.img\_to\_array(test\_image)

    test\_image = np.expand\_dims(test\_image, axis=0)

    result = models.predict(test\_image)

    prediction = result[0]

    prediction = list(prediction)

    classes=['Primary Endo with Secondary Perio','Primary Periodontal Lesion','True Combined Lesions']

    output = zip(classes, prediction)

    output = dict(output)

    if output['Primary Endo with Secondary Perio'] == 1.0:

        a = "Primary Endo with Secondary Perio"

    elif output['Primary Periodontal Lesion'] == 1.0:

        a = "Primary Periodontal Lesion"

    elif output['True Combined Lesions'] == 1.0:

        a = "True Combined Lesions"

    return render(request, "result.html", {"out": a})