```
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
import tensorflow as tf

from tensorflow.keras.utils import to_categorical
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.keras.preprocessing.image import ImageDataGenerator

from sklearn.metrics import classification_report, log_loss, accuracy_score
from sklearn.model_selection import train_test_split
```

```
dir_sp_train = '/content/drive/MyDrive/Parkinsons/drawings/spiral/training'
dir_sp_test = '/content/drive/MyDrive/Parkinsons/drawings/spiral/testing'
dir_wv_train = '/content/drive/MyDrive/Parkinsons/drawings/wave/training'
dir_wv_test = '/content/drive/MyDrive/Parkinsons/drawings/wave/testing'
```

```
Name=[]
for file in os.listdir(dir_sp_train):
    Name+=[file]
print(Name)
print(len(Name))
```

```
['healthy', 'parkinson']
2
```

```
N=[]
for i in range(len(Name)):
    N+=[i]

normal_mapping=dict(zip(Name,N))
reverse_mapping=dict(zip(N,Name))

def mapper(value):
    return reverse_mapping[value]
```

```
dataset_sp=[]
count=0
for file in os.listdir(dir_sp_train):
    path=os.path.join(dir_sp_train,file)
    for im in os.listdir(path):
        image=load_img(os.path.join(path,im), grayscale=False, color_mode='rgb', target_size=(100,100))
        image=img_to_array(image)
        image=image/255.0
        dataset_sp.append([image,count])
    count=count+1

testset_sp=[]
count=0
for file in os.listdir(dir_sp_test):
    path=os.path.join(dir_sp_test,file)
    for im in os.listdir(path):
        image=load_img(os.path.join(path,im), grayscale=False, color_mode='rgb', target_size=(100,100))
        image=img_to_array(image)
        image=image/255.0
        testset_sp.append([image,count])
    count=count+1
```

```
dataset_wv=[]
count=0
for file in os.listdir(dir_wv_train):
    path=os.path.join(dir_wv_train,file)
    for im in os.listdir(path):
        image=load_img(os.path.join(path,im), grayscale=False, color_mode='rgb', target_size=(100,100))
        image=img_to_array(image)
        image=image/255.0
        dataset_wv.append([image,count])
    count=count+1

testset_wv=[]
count=0
for file in os.listdir(dir_wv_test):
    path=os.path.join(dir_wv_test,file)
    for im in os.listdir(path):
        image=load_img(os.path.join(path,im), grayscale=False, color_mode='rgb', target_size=(100,100))
        image=img_to_array(image)
        image=image/255.0
        testset_wv.append([image,count])
    count=count+1
```

```
data_sp,labels_sp0=zip(*dataset_sp)
test_sp,tlabels_sp0=zip(*testset_sp)

data_wv,labels_wv0=zip(*dataset_wv)
test_wv,tlabels_wv0=zip(*testset_wv)
```

```
labels_sp1=to_categorical(labels_sp0)
data_sp=np.array(data_sp)
labels_sp=np.array(labels_sp1)

tlabels_sp1=to_categorical(tlabels_sp0)
test_sp=np.array(test_sp)
tlabels_sp=np.array(tlabels_sp1)
```

```
labels_wv1=to_categorical(labels_wv0)
data_wv=np.array(data_wv)
labels_wv=np.array(labels_wv1)

tlabels_wv1=to_categorical(tlabels_wv0)
```

```
test_wv=np.array(test_wv)
tlabels_wv=np.array(tlabels_wv1)
```

```
trainx_sp,testx_sp,trainy_sp,testy_sp=train_test_split(data_sp,labels_sp,test_size=0.2,random_state=44)
trainx_wv,testx_wv,trainy_wv,testy_wv=train_test_split(data_wv,labels_wv,test_size=0.2,random_state=44)
```

```
print(trainx_sp.shape)
print(testx_sp.shape)
print(trainy_sp.shape)
print(testy_sp.shape)
```

```
(57, 100, 100, 3)
(15, 100, 100, 3)
(57, 2)
(15, 2)
```

```
datagen = ImageDataGenerator(horizontal_flip=True,vertical_flip=True,rotation_range=20,zoom_range=0.2,
                        width_shift_range=0.2,height_shift_range=0.2,shear_range=0.1,fill_mode="nearest")
```

```
pretrained_model3 = tf.keras.applications.DenseNet201(input_shape=(100,100,3),include_top=False,weights='imagenet',pooling='avg')
pretrained_model3.trainable = False

pretrained_model4 = tf.keras.applications.DenseNet201(input_shape=(100,100,3),include_top=False,weights='imagenet',pooling='avg')
pretrained_model4.trainable = False
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/densenet/densenet201_weights_tf_dim_ordering_tf_kernels_notop.h5
74836368/74836368 [==============================] - 0s 0us/step
```

```
inputs3 = pretrained_model3.input
x3 = tf.keras.layers.Dense(128, activation='relu')(pretrained_model3.output)
outputs3 = tf.keras.layers.Dense(2, activation='softmax')(x3)
model3 = tf.keras.Model(inputs=inputs3, outputs=outputs3)
```

```
inputs4 = pretrained_model4.input
x4 = tf.keras.layers.Dense(128, activation='relu')(pretrained_model4.output)
outputs4 = tf.keras.layers.Dense(2, activation='softmax')(x4)
model4 = tf.keras.Model(inputs=inputs4, outputs=outputs4)
```

```
model3.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
model4.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
```

```
his3=model3.fit(datagen.flow(trainx_sp,trainy_sp,batch_size=32),validation_data=(testx_sp,testy_sp),epochs=50)
his4=model4.fit(datagen.flow(trainx_wv,trainy_wv,batch_size=32),validation_data=(testx_wv,testy_wv),epochs=50)
```

```
Epoch 22/50
2/2 [==============================] - 0s 182ms/step - loss: 0.3679 - accuracy: 0.8596 - val_loss: 0.3008 - val_accuracy: 0.8667
Epoch 23/50
2/2 [==============================] - 0s 280ms/step - loss: 0.4341 - accuracy: 0.8772 - val_loss: 0.2539 - val_accuracy: 0.8667
Epoch 24/50
2/2 [==============================] - 0s 271ms/step - loss: 0.3533 - accuracy: 0.8246 - val_loss: 0.2518 - val_accuracy: 0.8667
Epoch 25/50
2/2 [==============================] - 0s 243ms/step - loss: 0.3686 - accuracy: 0.8421 - val_loss: 0.2824 - val_accuracy: 0.8667
Epoch 26/50
2/2 [==============================] - 0s 311ms/step - loss: 0.3176 - accuracy: 0.8596 - val_loss: 0.2986 - val_accuracy: 0.8667
Epoch 27/50
2/2 [==============================] - 1s 308ms/step - loss: 0.4347 - accuracy: 0.8246 - val_loss: 0.2815 - val_accuracy: 0.8667
Epoch 28/50
2/2 [==============================] - 1s 316ms/step - loss: 0.3095 - accuracy: 0.9123 - val_loss: 0.2825 - val_accuracy: 0.8667
Epoch 29/50
2/2 [==============================] - 0s 259ms/step - loss: 0.3272 - accuracy: 0.8947 - val_loss: 0.2785 - val_accuracy: 0.8667
Epoch 30/50
2/2 [==============================] - 0s 178ms/step - loss: 0.3726 - accuracy: 0.8596 - val_loss: 0.3309 - val_accuracy: 0.8667
Epoch 31/50
2/2 [==============================] - 1s 347ms/step - loss: 0.3950 - accuracy: 0.8246 - val_loss: 0.3567 - val_accuracy: 0.7333
Epoch 32/50
2/2 [==============================] - 1s 284ms/step - loss: 0.3056 - accuracy: 0.8246 - val_loss: 0.2607 - val_accuracy: 0.8667
Epoch 33/50
2/2 [==============================] - 0s 231ms/step - loss: 0.2769 - accuracy: 0.8947 - val_loss: 0.2313 - val_accuracy: 0.9333
Epoch 34/50
2/2 [==============================] - 1s 353ms/step - loss: 0.4171 - accuracy: 0.8070 - val_loss: 0.2115 - val_accuracy: 0.8667
Epoch 35/50
2/2 [==============================] - 0s 189ms/step - loss: 0.3999 - accuracy: 0.8246 - val_loss: 0.2802 - val_accuracy: 0.8667
Epoch 36/50
2/2 [==============================] - 0s 209ms/step - loss: 0.4045 - accuracy: 0.8070 - val_loss: 0.3547 - val_accuracy: 0.8000
Epoch 37/50
2/2 [==============================] - 0s 190ms/step - loss: 0.4546 - accuracy: 0.7719 - val_loss: 0.3289 - val_accuracy: 0.8000
Epoch 38/50
2/2 [==============================] - 0s 187ms/step - loss: 0.3786 - accuracy: 0.8246 - val_loss: 0.2201 - val_accuracy: 0.8667
Epoch 39/50
2/2 [==============================] - 0s 175ms/step - loss: 0.3372 - accuracy: 0.8070 - val_loss: 0.2260 - val_accuracy: 0.8667
Epoch 40/50
2/2 [==============================] - 0s 187ms/step - loss: 0.2708 - accuracy: 0.8947 - val_loss: 0.2347 - val_accuracy: 0.8667
Epoch 41/50
2/2 [==============================] - 0s 211ms/step - loss: 0.3437 - accuracy: 0.8421 - val_loss: 0.2733 - val_accuracy: 0.8667
Epoch 42/50
2/2 [==============================] - 0s 166ms/step - loss: 0.3261 - accuracy: 0.8772 - val_loss: 0.2913 - val_accuracy: 0.8000
Epoch 43/50
2/2 [==============================] - 0s 166ms/step - loss: 0.2899 - accuracy: 0.8421 - val_loss: 0.2447 - val_accuracy: 0.8667
Epoch 44/50
2/2 [==============================] - 0s 162ms/step - loss: 0.2358 - accuracy: 0.8947 - val_loss: 0.2381 - val_accuracy: 0.8667
Epoch 45/50
2/2 [==============================] - 0s 187ms/step - loss: 0.2757 - accuracy: 0.8772 - val_loss: 0.2360 - val_accuracy: 0.8667
Epoch 46/50
2/2 [==============================] - 0s 214ms/step - loss: 0.3349 - accuracy: 0.8772 - val_loss: 0.2702 - val_accuracy: 0.8000
Epoch 47/50
2/2 [==============================] - 0s 185ms/step - loss: 0.4661 - accuracy: 0.8070 - val_loss: 0.4051 - val_accuracy: 0.7333
Epoch 48/50
2/2 [==============================] - 0s 163ms/step - loss: 0.3806 - accuracy: 0.8070 - val_loss: 0.3114 - val_accuracy: 0.8000
Epoch 49/50
2/2 [==============================] - 0s 212ms/step - loss: 0.3395 - accuracy: 0.8596 - val_loss: 0.2305 - val_accuracy: 0.9333
Epoch 50/50
2/2 [==============================] - 0s 162ms/step - loss: 0.3429 - accuracy: 0.8421 - val_loss: 0.2315 - val_accuracy: 0.9333
```

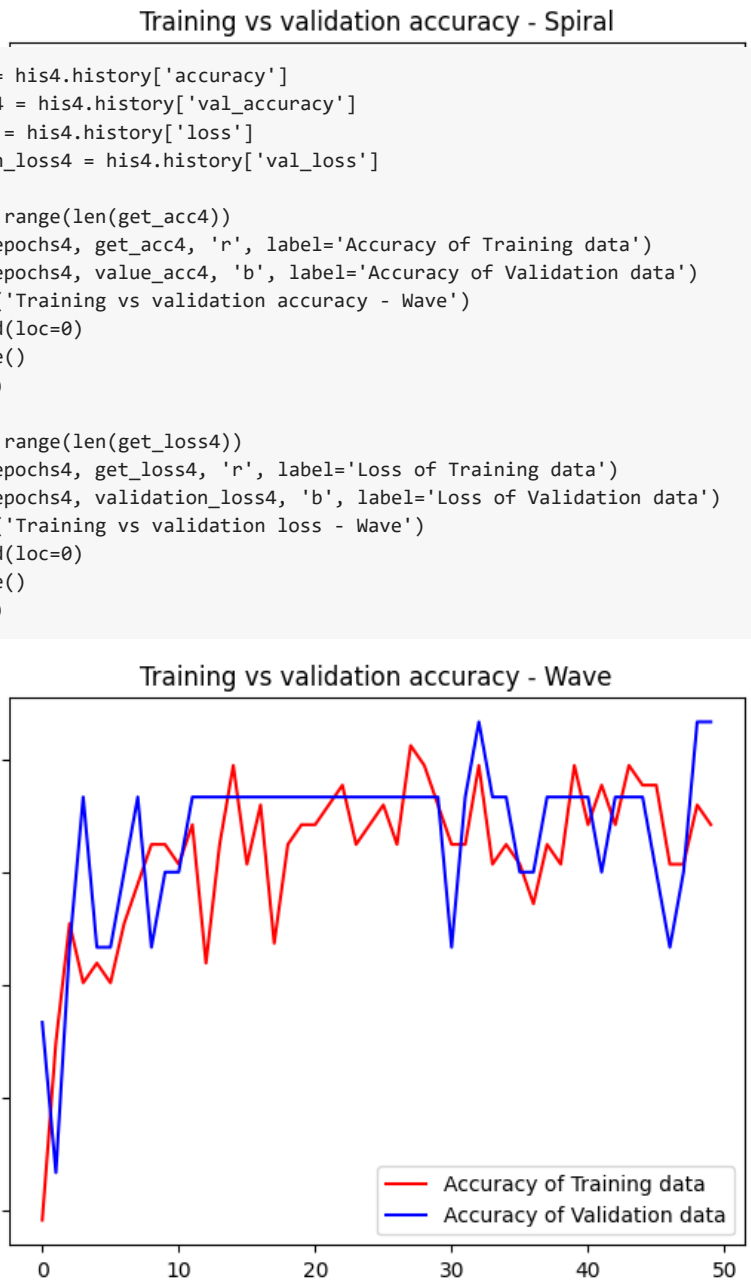
```
#spiral
y_pred_sp=model3.predict(testx_sp)
pred_sp=np.argmax(y_pred_sp,axis=1)
ground_sp = np.argmax(testy_sp,axis=1)
print(classification_report(ground_sp,pred_sp))
```

```
1/1 [==============================] - 3s 3s/step
              precision    recall  f1-score   support

           0       0.75      1.00      0.86         6
           1       1.00      0.78      0.88         9

    accuracy                           0.87        15
   macro avg       0.88      0.89      0.87        15
weighted avg       0.90      0.87      0.87        15
```

```
#wave
y_pred_wv=model3.predict(testx_wv)
pred_wv=np.argmax(y_pred_wv,axis=1)
ground_wv = np.argmax(testy_wv,axis=1)
print(classification_report(ground_wv,pred_wv))
```

```
1/1 [==============================] - 0s 42ms/step
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         6
           1       0.60      1.00      0.75         9

    accuracy                           0.60        15
   macro avg       0.30      0.50      0.37        15
weighted avg       0.36      0.60      0.45        15

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels w
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels w
  _warn_prf(average, modifier, msg_start, len(result))
```

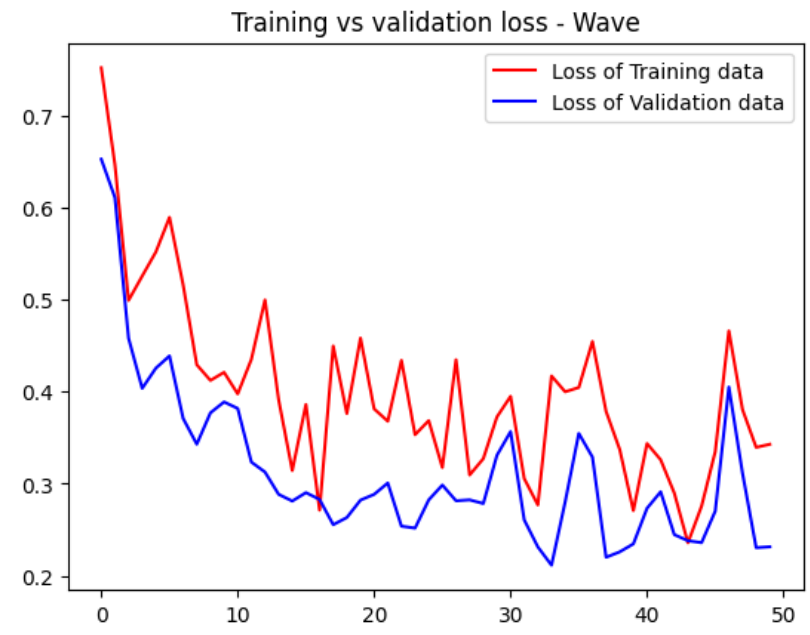
```
get_acc3 = his3.history['accuracy']
value_acc3 = his3.history['val_accuracy']
get_loss3 = his3.history['loss']
validation_loss3 = his3.history['val_loss']

epochs3 = range(len(get_acc3))
plt.plot(epochs3, get_acc3, 'r', label='Accuracy of Training data')
plt.plot(epochs3, value_acc3, 'b', label='Accuracy of Validation data')
plt.title('Training vs validation accuracy - Spiral')
plt.legend(loc=0)
plt.figure()
plt.show()

epochs3 = range(len(get_loss3))
plt.plot(epochs3, get_loss3, 'r', label='Loss of Training data')
plt.plot(epochs3, validation_loss3, 'b', label='Loss of Validation data')
plt.title('Training vs validation loss - Spiral')
plt.legend(loc=0)
plt.figure()
plt.show()
```




```
get_acc4 = his4.history['accuracy']
value_acc4 = his4.history['val_accuracy']
get_loss4 = his4.history['loss']
validation_loss4 = his4.history['val_loss']

epochs4 = range(len(get_acc4))
plt.plot(epochs4, get_acc4, 'r', label='Accuracy of Training data')
plt.plot(epochs4, value_acc4, 'b', label='Accuracy of Validation data')
plt.title('Training vs validation accuracy - Wave')
plt.legend(loc=0)
plt.figure()
plt.show()

epochs4 = range(len(get_loss4))
plt.plot(epochs4, get_loss4, 'r', label='Loss of Training data')
plt.plot(epochs4, validation_loss4, 'b', label='Loss of Validation data')
plt.title('Training vs validation loss - Wave')
plt.legend(loc=0)
plt.figure()
plt.show()
```



```
<Figure size 640x480 with 0 Axes>
```



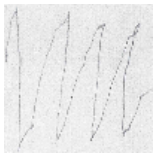

```
<Figure size 640x480 with 0 Axes>
```

```
load_img("/content/drive/MyDrive/Parkinsons/drawings/spiral/testing/parkinson/V03PE07.png",target_size=(100,100))
```

```
image=load_img("/content/drive/MyDrive/Parkinsons/drawings/spiral/testing/parkinson/V03PE07.png",target_size=(100,100))

image=img_to_array(image)
image=image/255.0
prediction_image=np.array(image)
prediction_image=np.expand_dims(image, axis=0)

prediction=model3.predict(prediction_image)
value=np.argmax(prediction)
move_name=mapper(value)
print("Prediction is {}.".format(move_name))
```

```
1/1 [==============================] - 2s 2s/step
Prediction is parkinson.
```

```
load_img("/content/drive/MyDrive/Parkinsons/drawings/wave/testing/parkinson/V03PO01.png",target_size=(100,100))
```



```
image2=load_img("/content/drive/MyDrive/Parkinsons/drawings/wave/testing/parkinson/V03PO01.png",target_size=(100,100))

image2=img_to_array(image2)
image2=image2/255.0
prediction_image2=np.array(image2)
prediction_image2=np.expand_dims(image2, axis=0)

prediction2=model4.predict(prediction_image2)
value2=np.argmax(prediction2)
move_name2=mapper(value2)
print("Prediction is {}.".format(move_name2))
```

```
1/1 [==============================] - 5s 5s/step
Prediction is healthy.
```

```
print(test_sp.shape)
prediction_sp=model3.predict(test_sp)
print(prediction_sp.shape)

PRED_sp=[]
for item in prediction_sp:
    value_sp=np.argmax(item)
    PRED_sp+=[value_sp]

ANS_sp=tlabels_sp0
accuracy_sp=accuracy_score(ANS_sp,PRED_sp)
print(accuracy_sp)
```

```
(30, 100, 100, 3)
1/1 [==============================] - 3s 3s/step
(30, 2)
0.23333333333333334
```

```
print(test_wv.shape)
prediction_wv=model4.predict(test_wv)
print(prediction_wv.shape)

PRED_wv=[]
for item in prediction_wv:
    value_wv=np.argmax(item)
    PRED_wv+=[value_wv]

ANS_wv=tlabels_wv0
accuracy_wv=accuracy_score(ANS_wv,PRED_wv)
print(accuracy_wv)
```

```
(30, 100, 100, 3)
1/1 [==============================] - 0s 48ms/step
(30, 2)
0.13333333333333333
```