```
import os
import glob
import numpy as np
import pandas as pd
import seaborn as sn
from PIL import Image
from collections import Counter
import matplotlib.pyplot as plt


# Function to imprort all .jpg images and store them as numpy arrays in a list
def importing_data(path):
    sample = []
    for filename in glob.glob(path):
        img = Image.open(filename,'r')
        IMG = np.array(img)
        sample.append(IMG)
    return sample


# Loading Train Data
path1 = '/content/drive/MyDrive/Colab Notebooks/Alzheimer_s Dataset/train/NonDemented/*.jp
path2 = '/content/drive/MyDrive/Colab Notebooks/Alzheimer_s Dataset/train/VeryMildDemented
path3 = '/content/drive/MyDrive/Colab Notebooks/Alzheimer_s Dataset/train/MildDemented/*.j
path4 = '/content/drive/MyDrive/Colab Notebooks/Alzheimer_s Dataset/train/ModerateDemented


train_ND = importing_data(path1)
train_VMD = importing_data(path2)
train_MID = importing_data(path3)
train_MOD = importing_data(path4)


print(len(train_ND),"&", (train_ND[0]).shape)
```

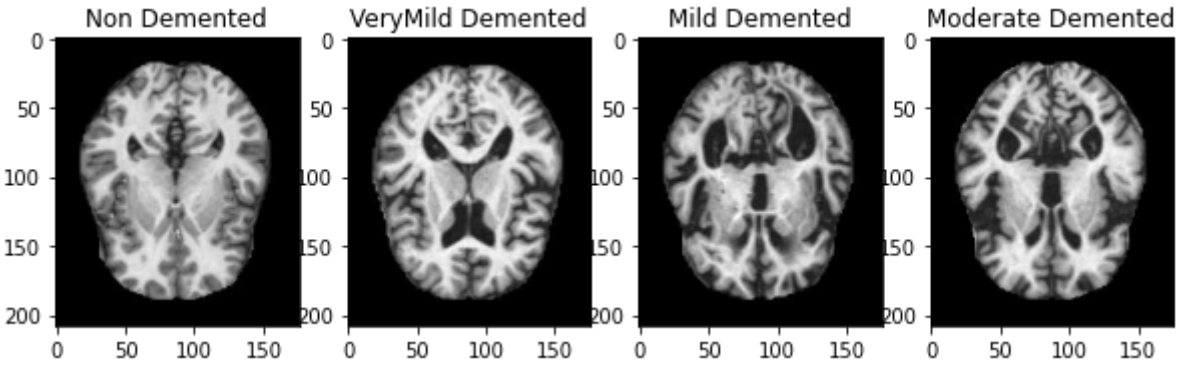
```
2560 & (208, 176)
```

```
print(train_ND[0][120])
```

```
[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0  43  91 110 116 126 134 140 129 124 102  88  98
 118 127 122 116 103  79  54  71 109 112 106 122 120 130 147 158 152 143
 151 167 163 165 165 164 162 162 165 169 189 191 187 177 174 181 186 186
 192 194 198 204 210 213 207 200 197 172 152 165 199 172 131 164 203 194
 158 143 175 203 205 208 209 216 204 188 185 181 175 176 166 175 184 187
 187 183 172 162 164 168 165 155 151 156 158 154 149 160 175 184 181 169
 154 145 127 126 117  87  52  45  51  48  51  55  68  81  80  67  56  53
  51  56  56  28  28   4   2   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
```

```
fig, (ax1, ax2, ax3, ax4) = plt.subplots(1,4, figsize= (10,10))

ax1.imshow(train_ND[0], cmap ="binary_r")
ax1.set_title('Non Demented')
```

```
ax2.imshow(train_VMD[0], cmap ="binary_r")
ax2.set_title('VeryMild Demented')
ax3.imshow(train_MID[0], cmap ="binary_r")
ax3.set_title('Mild Demented')
ax4.imshow(train_MOD[0], cmap ="binary_r")
ax4.set_title('Moderate Demented')
plt.show()
```




```
path5 = '/content/drive/MyDrive/Colab Notebooks/Alzheimer_s Dataset/test/NonDemented/*.jpg
path6 = '/content/drive/MyDrive/Colab Notebooks/Alzheimer_s Dataset/test/VeryMildDemented/
path7 = '/content/drive/MyDrive/Colab Notebooks/Alzheimer_s Dataset/test/MildDemented/*.jp
path8 = '/content/drive/MyDrive/Colab Notebooks/Alzheimer_s Dataset/test/ModerateDemented/
```

```
test_ND = importing_data(path5)
test_VMD = importing_data(path6)
test_MID = importing_data(path7)
test_MOD = importing_data(path8)
```

```
df_train_ND = pd.DataFrame({'image':train_ND, 'label': 'ND'})
df_train_VMD = pd.DataFrame({'image':train_VMD, 'label': 'VMD'})
df_train_MID = pd.DataFrame({'image':train_MID, 'label': 'MID'})
df_train_MOD = pd.DataFrame({'image':train_MOD, 'label': 'MOD'})

df_test_ND = pd.DataFrame({'image':test_ND, 'label': 'ND'})
df_test_VMD = pd.DataFrame({'image':test_VMD, 'label': 'VMD'})
df_test_MID = pd.DataFrame({'image':test_MID, 'label': 'MID'})
df_test_MOD = pd.DataFrame({'image':test_MOD, 'label': 'MOD'})
```

```
df_train_ND.head(4)
```

	image	label
0	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	ND
1	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	ND
2	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	ND
3	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	ND

```
final_data = [df_train_ND, df_train_VMD, df_train_MID, df_train_MOD,df_test_ND, df_test_VM

final_data = pd.concat(final_data)
```

```
final_data.head(10)
```

	image	label
0	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	ND
1	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	ND
2	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	ND
3	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	ND
4	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	ND
5	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	ND
6	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	ND
7	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	ND
8	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	ND
9	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	ND

```
final_data.tail(10)
```

	image	label
2	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	MOD
3	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	MOD
4	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	MOD
5	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	MOD
6	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	MOD
7	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	MOD
8	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	MOD
9	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	MOD
10	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	MOD
11	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	MOD

```
train_data = final_data['image']
labels = final_data['label']

Counter(np.array(labels))
# Data is unbalnced
# daset is strongly unmbalanced, with 2560 images in the Non-Demented class,
```

```
# 1792 images in the VeryMild-Demented class,
# 717 in the Mild-Demented class and just
# 52 images in the Moderate-Demented class.
```

```
Counter({'MID': 896, 'MOD': 64, 'ND': 3200, 'VMD': 2240})
```

```
from sklearn.preprocessing import MinMaxScaler
def normalization(array):

    train_norm = []
    transformer = MinMaxScaler()

    for value in array:
        value = transformer.fit_transform(value)
        train_norm.append(value)

    return train_norm

train_norm = normalization(train_data)
```

```
from sklearn.preprocessing import LabelBinarizer

onehot = LabelBinarizer()
labels = onehot.fit_transform(labels)
print(labels)
```

```
[[0 0 1 0]
 [0 0 1 0]
 [0 0 1 0]
 ...
 [0 1 0 0]
 [0 1 0 0]
 [0 1 0 0]]
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(train_norm, labels,
                                                    test_size = 0.2,
                                                    stratify = labels,
                                                    shuffle = True,
                                                    random_state = 42)
```

```
print('length X_train:', len(X_train))
print('length X_test:',  len(X_test))
print('length y_train:', len(y_train))
print('length y_test:', len(y_test))
```

```
length X_train: 5120
length X_test: 1280
length y_train: 5120
length y_test: 1280
```

```
X_train = np.array(X_train).reshape(5120,208,176,1)
```

```
X_test = np.array(X_test).reshape(1280,208,176,1)

from sklearn.utils import compute_class_weight

y_integers = np.argmax(y_train, axis=1)
class_weights = compute_class_weight('balanced', np.unique(y_integers), y_integers)
d_class_weights = dict(enumerate(class_weights))
print(d_class_weights)
```

```
{0: 1.7852161785216178, 1: 25.098039215686274, 2: 0.5, 3: 0.7142857142857143}
```

```
from keras.metrics import AUC
from keras.models import Sequential
from keras.layers import Dense, BatchNormalization, Dropout, Conv2D , MaxPooling2D, Flatte
from keras.callbacks import EarlyStopping,ModelCheckpoint
from keras.optimizers import RMSprop
def build_model():
    Cnn = Sequential()
    Cnn.add(Conv2D(64,(5,5), activation = 'relu', padding = 'same', strides=(2,2), input_s
    Cnn.add(MaxPooling2D(2))
    Cnn.add(Conv2D(128,(5,5), activation = 'relu', padding = 'same', strides=(2,2)))
    Cnn.add(Conv2D(128,(5,5), activation = 'relu', padding = 'same', strides=(2,2)))
    Cnn.add(Conv2D(256,(5,5), activation = 'relu', padding = 'same', strides=(2,2)))
    Cnn.add(MaxPooling2D(2))
    Cnn.add(Flatten())
    Cnn.add(Dense(64, activation = 'relu'))
    Cnn.add(Dropout(0.4))
    Cnn.add(Dense(32, activation = 'relu'))
    Cnn.add(Dropout(0.4))
    Cnn.add(Dense(4, activation = 'softmax'))
    return Cnn

keras_model = build_model()
keras_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 104, 88, 64)       1664
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 52, 44, 64)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 26, 22, 128)       204928
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 13, 11, 128)       409728
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 7, 6, 256)         819456
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 3, 3, 256)         0
_________________________________________________________________
flatten (Flatten)            (None, 2304)              0
_________________________________________________________________
dense (Dense)                (None, 64)                147520
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
```

```
_________________________________________________________________
dense_1 (Dense)              (None, 32)                2080
_________________________________________________________________
dropout_1 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_2 (Dense)              (None, 4)                 132
=================================================================
Total params: 1,585,508
Trainable params: 1,585,508
Non-trainable params: 0
_________________________________________________________________
```

```
def Model_fit(name):
    keras_model = None
    keras_model = build_model()
    keras_model.compile(optimizer = RMSprop(learning_rate = 1e-4), loss='categorical_cross
    es = EarlyStopping(monitor='val_loss', mode='min', patience=10 , restore_best_weights=
    checkpoint_cb = ModelCheckpoint("AD_Stages_model.h5", save_best_only=True)
    history = keras_model.fit(X_train, y_train, validation_split = 0.1, epochs= 100, batch
    keras_model.save('AD_Stages_model'+str(name)+'.h5')
    return history

def CrossVal(n_fold):
    cv_results = []
    for i in range(n_fold):
        print("Training on Fold: ",i+1)
        cv_results.append(Model_fit(i))
    return cv_results

cv_results = CrossVal(3)

fold1 = cv_results[0]
fold2 = cv_results[1]
fold3 = cv_results[2]
```

```
Training on Fold:  1
Epoch 1/100
461/461 [==============================] - 182s 395ms/step - loss: 1.4711 - acc: 0
Epoch 2/100
461/461 [==============================] - 175s 380ms/step - loss: 1.4790 - acc: 0
Epoch 3/100
461/461 [==============================] - 172s 373ms/step - loss: 1.4807 - acc: 0
Epoch 4/100
461/461 [==============================] - 172s 372ms/step - loss: 1.4720 - acc: 0
Epoch 5/100
461/461 [==============================] - 171s 371ms/step - loss: 1.4523 - acc: 0
Epoch 6/100
461/461 [==============================] - 171s 370ms/step - loss: 1.4316 - acc: 0
Epoch 7/100
461/461 [==============================] - 175s 379ms/step - loss: 1.3577 - acc: 0
Epoch 8/100
461/461 [==============================] - 179s 387ms/step - loss: 1.2737 - acc: 0
Epoch 9/100
461/461 [==============================] - 179s 388ms/step - loss: 1.2092 - acc: 0
Epoch 10/100
461/461 [==============================] - 177s 385ms/step - loss: 1.1462 - acc: 0
Epoch 11/100
```

```
461/461 [==============================] - 177s 384ms/step - loss: 1.0974 - acc: 0
Epoch 12/100
461/461 [==============================] - 177s 384ms/step - loss: 1.0345 - acc: 0
Epoch 13/100
461/461 [==============================] - 178s 385ms/step - loss: 0.9689 - acc: 0
Epoch 14/100
461/461 [==============================] - 176s 381ms/step - loss: 0.8738 - acc: 0
Epoch 15/100
461/461 [==============================] - 177s 385ms/step - loss: 0.7932 - acc: 0
Epoch 16/100
461/461 [==============================] - 177s 384ms/step - loss: 0.7319 - acc: 0
Epoch 17/100
461/461 [==============================] - 178s 385ms/step - loss: 0.6593 - acc: 0
Epoch 18/100
461/461 [==============================] - 178s 385ms/step - loss: 0.5840 - acc: 0
Epoch 19/100
461/461 [==============================] - 179s 388ms/step - loss: 0.4512 - acc: 0
Epoch 20/100
461/461 [==============================] - 181s 393ms/step - loss: 0.4040 - acc: 0
Epoch 21/100
461/461 [==============================] - 178s 387ms/step - loss: 0.3825 - acc: 0
Epoch 22/100
461/461 [==============================] - 177s 385ms/step - loss: 0.2978 - acc: 0
Epoch 23/100
461/461 [==============================] - 176s 381ms/step - loss: 0.2662 - acc: 0
Epoch 24/100
461/461 [==============================] - 172s 374ms/step - loss: 0.2362 - acc: 0
Epoch 25/100
461/461 [==============================] - 172s 373ms/step - loss: 0.2016 - acc: 0
Epoch 26/100
461/461 [==============================] - 170s 369ms/step - loss: 0.2818 - acc: 0
Epoch 27/100
461/461 [==============================] - 172s 374ms/step - loss: 0.1835 - acc: 0
Epoch 28/100
461/461 [==============================] - 172s 373ms/step - loss: 0.2096 - acc: 0
```

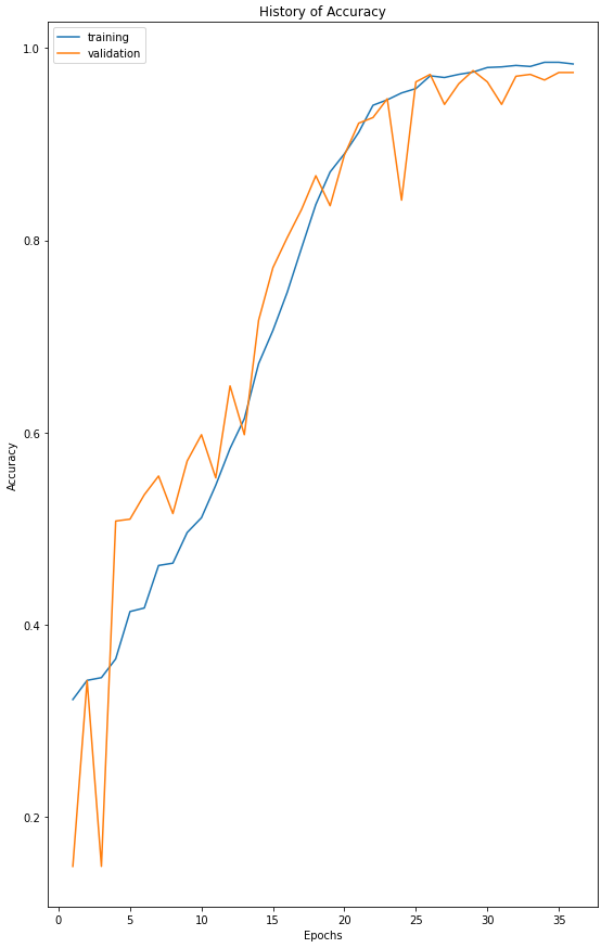
```
#%% CHEKING THE CROSS VALIDATION METRICS

print('Val_Acc Folder 1: ', max(fold1.history['val_acc']))
print('Val_Acc Folder 2: ', max(fold2.history['val_acc']))
print('Val_Acc Folder 3: ', max(fold3.history['val_acc']))
print('--------------------------------')
print('Val_Auc Folder 1: ', max(fold1.history['val_auc']))
print('Val_Auc Folder 2: ', max(fold2.history['val_auc']))
print('Val_Auc Folder 3: ', max(fold3.history['val_auc']))
```

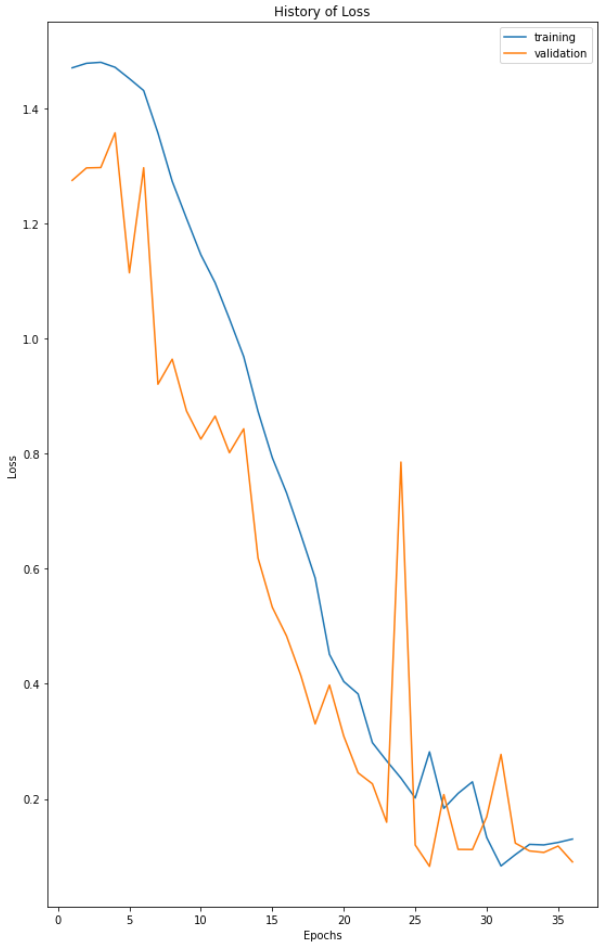
```
Val_Acc Folder 1:  0.9765625
Val_Acc Folder 2:  0.94140625
Val_Acc Folder 3:  0.97265625
--------------------------------
Val_Auc Folder 1:  0.998055100440979
Val_Auc Folder 2:  0.9921443462371826
Val_Auc Folder 3:  0.9965260624885559
```

```
def Train_Val_Plot(acc,val_acc,loss,val_loss):

    fig, (ax1, ax2) = plt.subplots(1,2, figsize= (20,15))
```

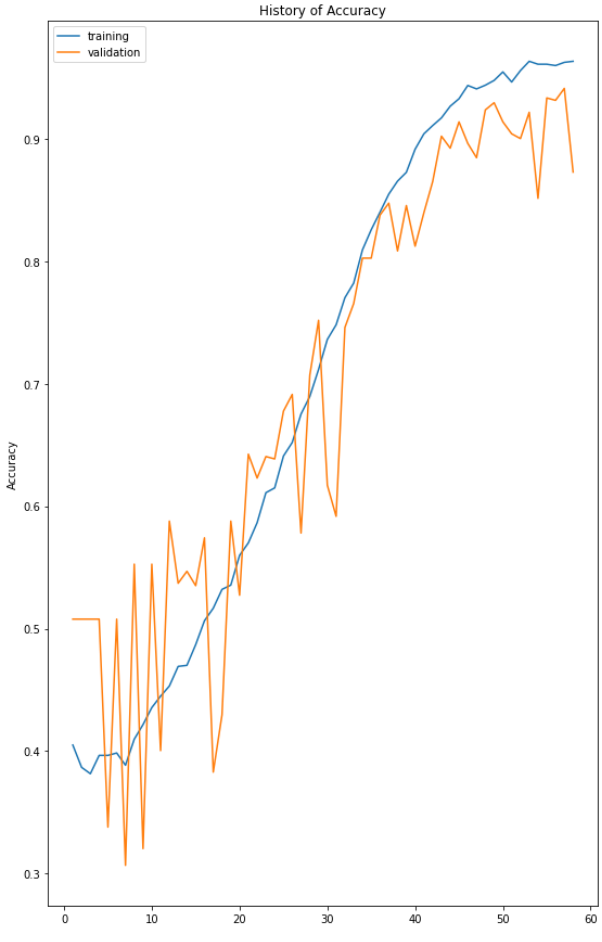
```
    fig.suptitle(" MODEL'S METRICS VISUALIZATION ")

    ax1.plot(range(1, len(acc) + 1), acc)
    ax1.plot(range(1, len(val_acc) + 1), val_acc)
    ax1.set_title('History of Accuracy')
    ax1.set_xlabel('Epochs')
    ax1.set_ylabel('Accuracy')
    ax1.legend(['training', 'validation'])


    ax2.plot(range(1, len(loss) + 1), loss)
    ax2.plot(range(1, len(val_loss) + 1), val_loss)
    ax2.set_title('History of Loss')
    ax2.set_xlabel('Epochs')
    ax2.set_ylabel('Loss')
    ax2.legend(['training', 'validation'])
    plt.show()


Train_Val_Plot(fold1.history['acc'],fold1.history['val_acc'],
               fold1.history['loss'],fold1.history['val_loss'])

Train_Val_Plot(fold2.history['acc'],fold2.history['val_acc'],
               fold2.history['loss'],fold2.history['val_loss'])

Train_Val_Plot(fold3.history['acc'],fold3.history['val_acc'],
               fold3.history['loss'],fold3.history['val_loss'])
```

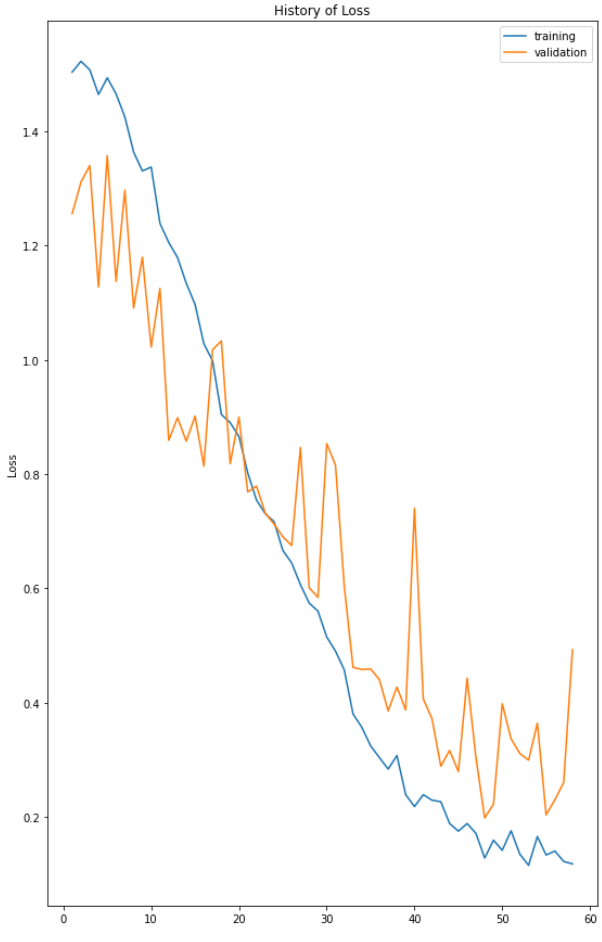
MODEL'S METRICS VISUALIZATION





MODEL'S METRICS VISUALIZATION

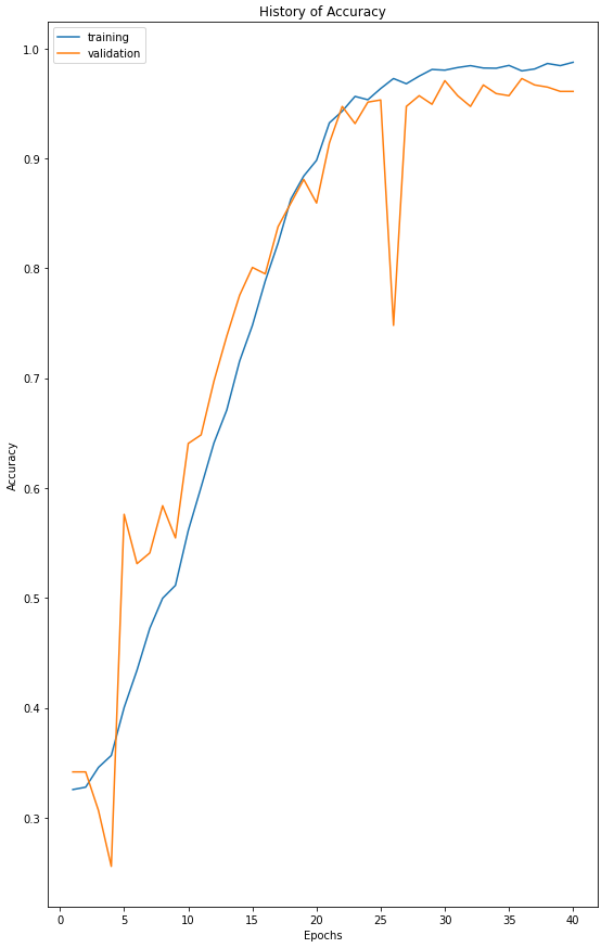


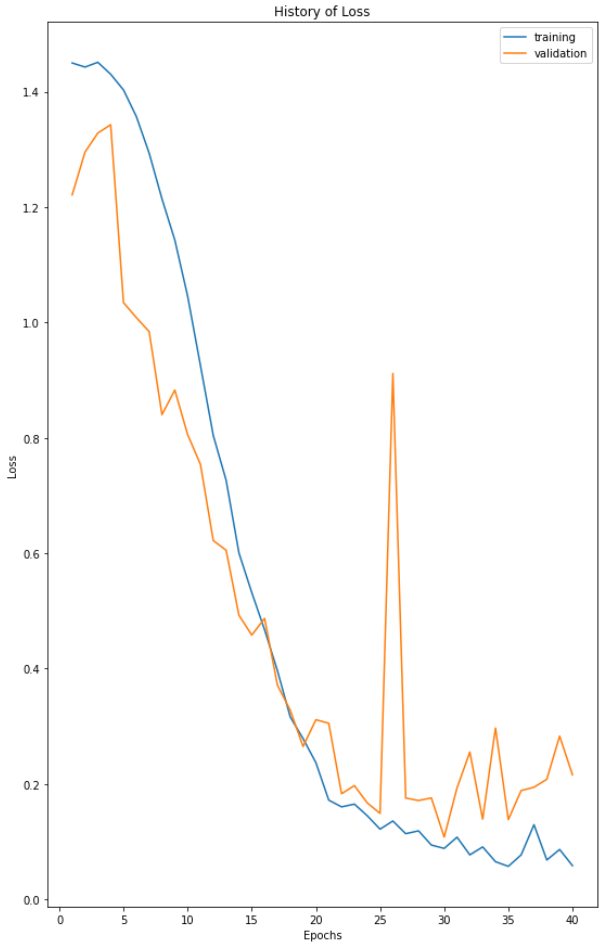


Epochs

Epochs

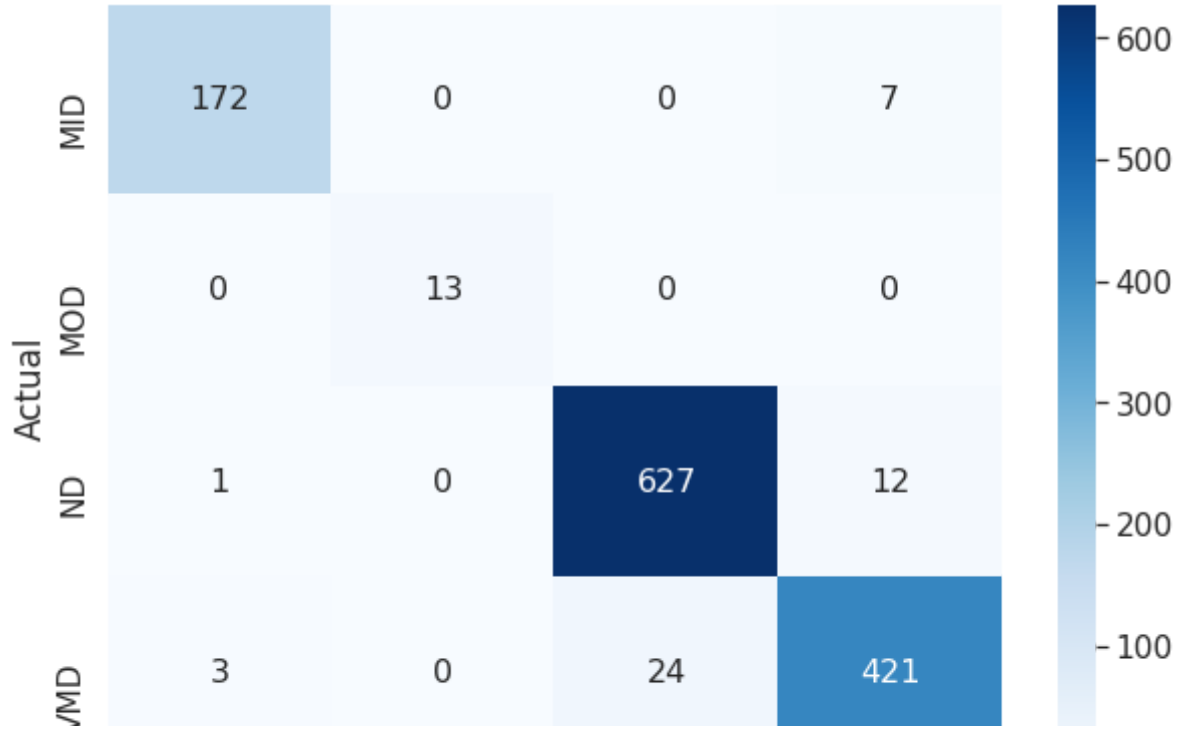
MODEL'S METRICS VISUALIZATION





```
#%% LOADING THE MODEL
import keras
keras_model = keras.models.load_model('AD_Stages_model.h5')
keras_model.compile(optimizer = RMSprop(learning_rate = 1e-4),
                    loss='categorical_crossentropy', metrics =[ 'acc'])


# Prediction on test_set

pred_test = keras_model.predict(X_test, verbose = 1)
pred_test = onehot.inverse_transform(pred_test)
real_val = onehot.inverse_transform(y_test)
pred_test_prb= keras_model.predict_proba(X_test)

     40/40 [==============================] - 12s 291ms/step


from collections import Counter
from sklearn.metrics import confusion_matrix

print(Counter(real_val))
print(Counter(pred_test))

conf_mx = confusion_matrix(real_val, pred_test)
conf_mx

heat_cm = pd.DataFrame(conf_mx, columns=np.unique(real_val), index = np.unique(real_val))
heat_cm.index.name = 'Actual'
heat_cm.columns.name = 'Predicted'
plt.figure(figsize = (10,7))
sn.set(font_scale=1.4) # For label size
sn.heatmap(heat_cm, cmap="Blues", annot=True, annot_kws={"size": 16},fmt='g')# font size
plt.show()
```

```
Counter({'ND': 640, 'VMD': 448, 'MID': 179, 'MOD': 13})
Counter({'ND': 651, 'VMD': 440, 'MID': 176, 'MOD': 13})
```




```
from sklearn.metrics import classification_report
roc_auc = dict()
print(classification_report(real_val, pred_test))
print(roc_auc)
```

```
                  precision    recall  f1-score   support

             MID       0.98      0.96      0.97       179
             MOD       1.00      1.00      1.00        13
              ND       0.96      0.98      0.97       640
             VMD       0.96      0.94      0.95       448

        accuracy                           0.96      1280
       macro avg       0.97      0.97      0.97      1280
    weighted avg       0.96      0.96      0.96      1280

    {}
```

```
import keras
from matplotlib import pyplot
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from numpy import expand_dims
from keras.models import Model
from skimage import color
from skimage import io

for i in range(len(keras_model.layers)):
    layer = keras_model.layers[i]
    # check for convolutional layer
    if 'conv' not in layer.name:
        continue
```

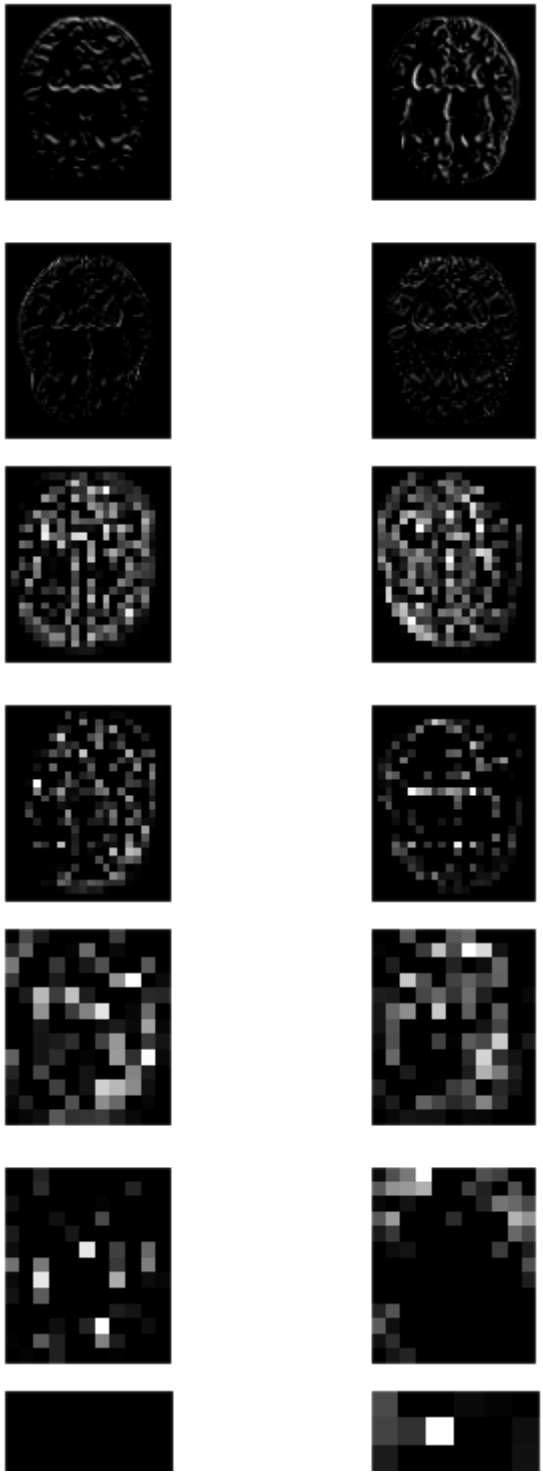
```
    # summarize output shape
    print(i, layer.name, layer.output.shape)
```

```
 0 conv2d_12 (None, 104, 88, 64)
 2 conv2d_13 (None, 26, 22, 128)
 3 conv2d_14 (None, 13, 11, 128)
 4 conv2d_15 (None, 7, 6, 256)
```

```
# Importing an image as example

img = color.rgb2gray(io.imread('/content/drive/MyDrive/Colab Notebooks/Alzheimer_s Dataset
img = expand_dims(img, axis=0)
img
```

```
 array([[[0, 0, 0, ..., 0, 0, 0],
         [0, 0, 0, ..., 0, 0, 0],
         [0, 0, 0, ..., 0, 0, 0],
         ...,
         [0, 0, 0, ..., 0, 0, 0],
         [0, 0, 0, ..., 0, 0, 0],
         [0, 0, 0, ..., 0, 0, 0]]], dtype=uint8)
```

```
#Visualizing our Convolutional outputs

ixs = [0,2,3,4] # The indeces of our Convolutional Layers

outputs = [keras_model.layers[i].output for i in ixs]
model3 = Model(inputs=keras_model.inputs, outputs=outputs)
feature_maps = model3.predict(img)
square = 2
for fmap in feature_maps:
    # plot all 64 maps in an 8x8 squares
    ix = 1
    for _ in range(square):
        for _ in range(square):
            # specify subplot and turn of axis
            ax = pyplot.subplot(square, 2, ix)
            ax.set_xticks([])
            ax.set_yticks([])
            # plot filter channel in grayscale
            pyplot.imshow(fmap[0, :, :, ix-1], cmap='gray')
            ix += 1
    # show the figure
    pyplot.show()
```



```
from google.colab import files
files.download("AD_Stages_model.h5")
```

```
from google.colab import files
files.download("AD_Stages_model0.h5")
```

```
from google.colab import files
files.download("AD_Stages_model1.h5")
```