MDS381 - Specialization Project

System Development Phase

Submitted by:

Gloria Sara Joji (2248429)

Jeniya Richu Jacob (2248431)

Sneha (2248439)

3MDS B

Date of submission: 03/10/2023

Submitted to:

Dr. Mahalakshmi J

Department of Computer Science

CHRIST (Deemed to be University), Yeshwanthpur Campus

```
# Importing necessary libraries
import pandas as pd
import numpy as np
import cv2
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import matplotlib.patheffects as PathEffects
from PIL import Image, ImageDraw, ImageOps
%matplotlib inline
import seaborn as sns
import random
import os
import gc

from sklearn.manifold import TSNE
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras import models, Sequential
from tensorflow.keras import optimizers

from keras.layers.core import Dense, Dropout, Flatten
from keras.layers.convolutional import Conv2D, MaxPooling2D,
SeparableConv2D

from tensorflow.keras.applications.vgg16 import VGG16

from keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing.image import img_to_array, load_img

from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report,
confusion_matrix
from sklearn.metrics import roc_curve, auc
from sklearn.metrics import precision_recall_curve
```

Getting 'Autistic' and 'Non-Autistic' train images from respective file names of train data

```
train_non_autistic = []
train_autistic = []
for i in os.listdir(train_dir):
    if 'Non_Autistic' in ("input/autism-image-
data/AutismDataset/train/{}".format(i)):
        train_non_autistic.append(("input/autism-image-
data/AutismDataset/train/{}".format(i)))
    else:
        train_autistic.append(("input/autism-image-
data/AutismDataset/train/{}".format(i)))
```

```
# Getting test images from test data file path
test_imgs = ["input/autism-image-
data/AutismDataset/test/{}".format(i) for i in
os.listdir(test_dir)]


# Concatenate 'Autistic'  and 'Non-Autistic' images and shuffle
them as train_images
train_imgs = train_autistic + train_non_autistic
random.shuffle(train_imgs)
```

Set the dimensions for images

```
nrows = 150
ncolumns  = 150
channels = 3

# Read and process the images: Function returns X,y. X - list of
resized images, y - list of labels for the images

def read_and_process_image(list_of_images):
    X = []
    y = []

    for image in list_of_images:
        X.append(cv2.resize(cv2.imread(image, cv2.IMREAD_COLOR),
(nrows, ncolumns), interpolation = cv2.INTER_CUBIC))
        if 'Non_Autistic' in image:
            y.append(0)
        else:
            y.append(1)

    return X,y
```

Function for pre-processing images for input to t-sne algorithm

```
def process_data_tsne(list_of_images):
    nrows = 150
    ncolumns  = 150
    channels = 3

    X = []
    y = []

    for image in list_of_images:
        X.append(cv2.resize(cv2.imread(image,
cv2.COLOR_BGR2GRAY), (nrows, ncolumns)))
```

```
        if 'Non_Autistic' in image:
            y.append(0)
        else:
            y.append(1)

    X = np.asarray(X)
    y = np.asarray(y)
    X = X.reshape(2540, 150*150*3)

    return X,y
```

We use pre-trained Convolutional Neural Network(CNN) model VGG16 that has been trained on a large number of image data for image classification task.

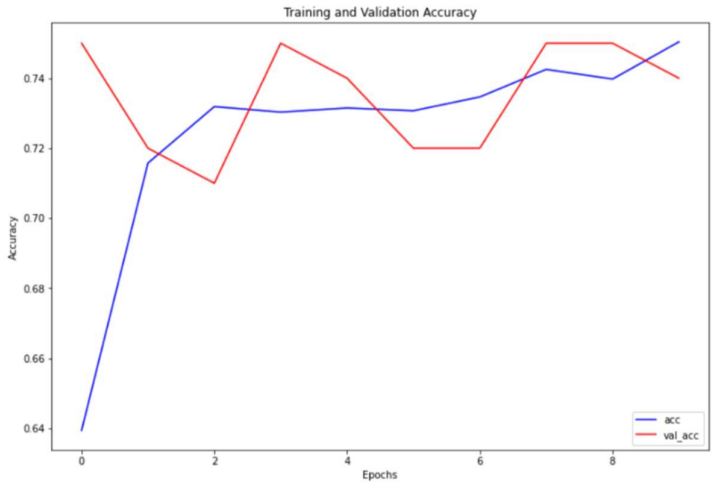
```
base_model =
VGG16(include_top=False,weights='imagenet',input_shape=(150,150,3
))
```

Model training

```
# Tune the model based on the validation loss
early_stopping =
keras.callbacks.EarlyStopping(monitor="val_loss", patience=3)

# Train the model
history = model.fit(train_generator,
                              epochs=30,
                              validation_data=val_generator,
                              callbacks=[early_stopping],
                              workers=4,
                              use_multiprocessing=False
                             )
```

```
80/80 [==============================] - 21s 142ms/step - loss: 0
.8141 - acc: 0.6394 - val_loss: 0.5278 - val_acc: 0.7500
Epoch 2/30
80/80 [==============================] - 12s 145ms/step - loss: 0
.5514 - acc: 0.7157 - val_loss: 0.5284 - val_acc: 0.7200
Epoch 3/30
80/80 [==============================] - 11s 133ms/step - loss: 0
.5362 - acc: 0.7319 - val_loss: 0.5320 - val_acc: 0.7100
Epoch 4/30
80/80 [==============================] - 12s 146ms/step - loss: 0
.5362 - acc: 0.7303 - val_loss: 0.5126 - val_acc: 0.7500
Epoch 5/30
80/80 [==============================] - 11s 134ms/step - loss: 0
.5300 - acc: 0.7315 - val_loss: 0.5178 - val_acc: 0.7400
Epoch 6/30
```

```
80/80 [==============================] - 11s 132ms/step - loss: 0
.5264 - acc: 0.7307 - val_loss: 0.5067 - val_acc: 0.7200
Epoch 7/30
80/80 [==============================] - 12s 147ms/step - loss: 0
.5092 - acc: 0.7346 - val_loss: 0.5012 - val_acc: 0.7200
Epoch 8/30
80/80 [==============================] - 11s 135ms/step - loss: 0
.5020 - acc: 0.7425 - val_loss: 0.5224 - val_acc: 0.7500
Epoch 9/30
80/80 [==============================] - 12s 144ms/step - loss: 0
.5112 - acc: 0.7398 - val_loss: 0.5076 - val_acc: 0.7500
Epoch 10/30
80/80 [==============================] - 11s 134ms/step - loss: 0
.5071 - acc: 0.7504 - val_loss: 0.5296 - val_acc: 0.7400
```

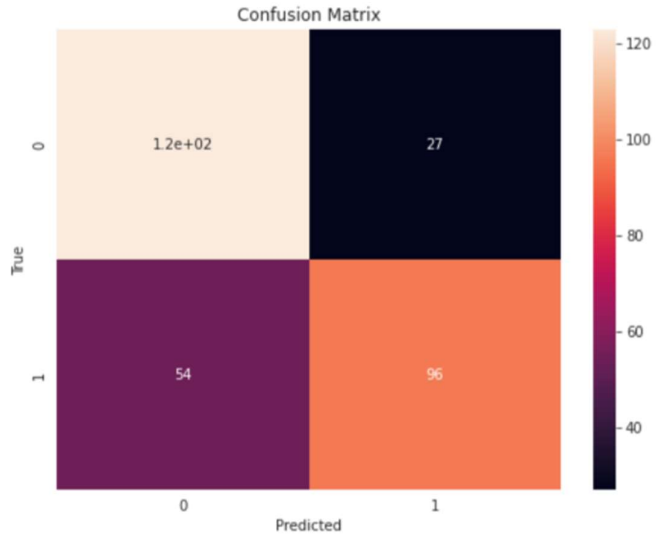


Model Validation

```
# Generating Classification report for model's performance in
each class
cl_report = classification_report(y_test, predictions)
print(cl_report)
```

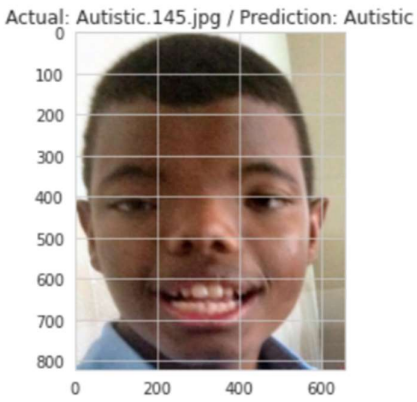
```
                precision    recall  f1-score   support

           0       0.69      0.82      0.75       150
           1       0.78      0.64      0.70       150

    accuracy                           0.73       300
   macro avg       0.74      0.73      0.73       300
weighted avg       0.74      0.73      0.73       300
```

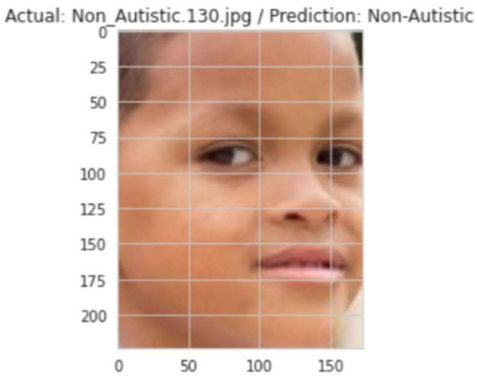
Confusion matrix



Predictions against some test images

```
plt.figure(figsize=(4,4))
for val, i in enumerate(test_imgs[:10]):
    img = mpimg.imread(i)
    imgplot = plt.imshow(img)
    plt.title('Actual: ' + os.path.basename(i) + ' / Prediction:
' +  f"{'Autistic' if predictions[val] == 1 else 'Non-
Autistic'}")
    plt.show()
```



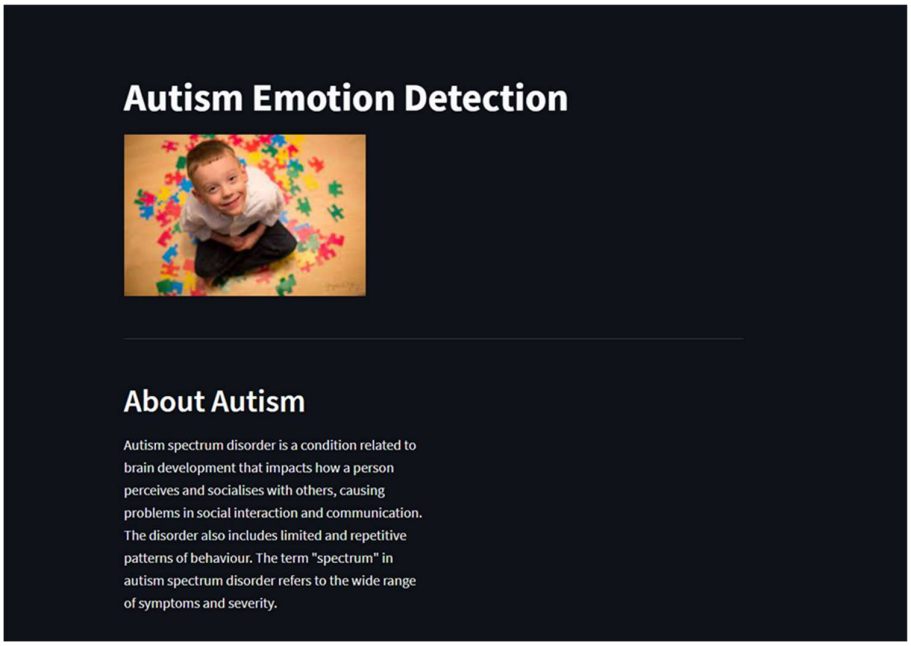


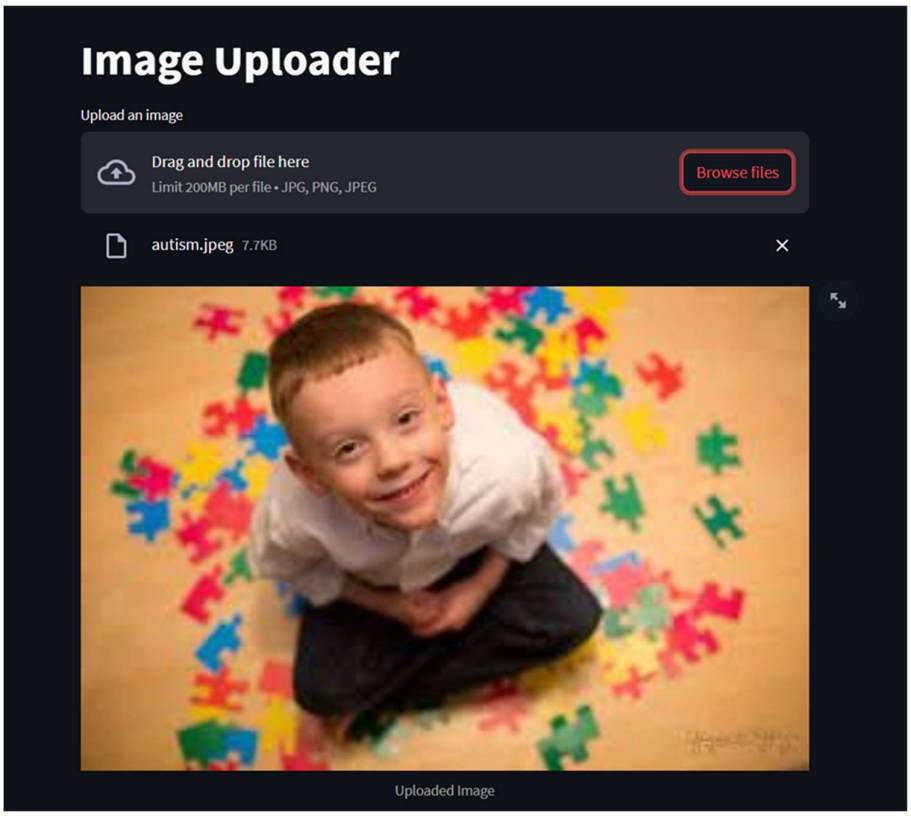
Streamlit code for website creation:

```
import streamlit as st

# Title and information about Autism
st.set_page_config(page_title="Autism")
st.title("Autism Emotion Detection")
st.image("autism.jpeg")
with st.container():
        st.write("----")
        left_column, right_column =st.columns(2)
        with left_column:
                st.header("About Autism")
                st.write("""Autism spectrum disorder is a condition related to brain development that impacts how a person perceives
and socialises with others, causing problems in social interaction and communication. The disorder also includes limited and repetitive
patterns of behaviour. The term "spectrum" in autism spectrum disorder refers to the wide range of symptoms and severity.""")

from PIL import Image

# To upload image
st.title("Image Uploader")

# File uploader widget
uploaded_image = st.file_uploader("Upload an image", type=["jpg", "png", "jpeg"])

if uploaded_image is not None:
    # Display the uploaded image
    image = Image.open(uploaded_image)
    st.image(image, caption="Uploaded Image", use_column_width=True)

    # You can perform additional operations on the image here



# Feedback form
st.title("Feedback Form")
st.write("We value your feedback. Please let us know your thoughts!")

# Text input for user's name
name = st.text_input("Your Name:")
```

```
# Text area for feedback message
feedback = st.text_area("Feedback:")

# Button to submit feedback
if st.button("Submit Feedback"):
    # Save the feedback to a file, database, or take any other desired action
    # In this example, we'll simply display the feedback
    st.success(f"Thank you, {name}, for your feedback:")
    st.write(feedback)
```

Demo Model Website:





A feedback form for improvisation.