A

MINI PROJECT REPORT

On

Down Syndrome Prediction

As

ISE-2 ACTIVITY-I

UNDER THE SUBJET

DIGITAL IMAGE PROCESSING [1CSPE361]

FOR

TY B. TECH. IN COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ANNASAHEB DANGE COLLAGE OF ENGINEERING AND TECHNOLOGY, ASHTA

(AN AUTONOMOUS INSTITUTE AFFILIATED TO SHIVAJI UNIVERSITY KOLHAPUR)

BY

Mr. Shrinath Dongare(21031093) (3042)

Mr. Prashant Mali(21031096) (3043)

Mr. Jaid Pirjade (21031114) (3049)

UNDER THE GUIDANCE OF

Mrs. A.M. Chougule

DEPARTMENT OF CSE

ACADEMIC YEAR 2024-25

Title: Down Syndrome Prediction

- ## Introduction:

Description of topic:

Down Syndrome prediction involves assessing the risk of a fetus having an extra chromosome 21 through prenatal screening and diagnostic tests. Key methods include first-trimester screening (blood tests and nuchal translucency ultrasound), second-trimester screening (triple or quad screen), and advanced non-invasive prenatal testing (NIPT) analyzing fetal DNA in maternal blood. Diagnostic procedures like chorionic villus sampling (CVS) and amniocentesis provide definitive results. Factors affecting prediction accuracy include maternal age, family history, and ethnicity. Advances in genomic sequencing and AI are enhancing prediction accuracy. Ethical considerations and equitable access to testing are crucial in this domain

Dataset Description :

A dataset for Down Syndrome prediction typically includes prenatal screening results, such as blood test values (PAPP-A, hCG, AFP, estriol, inhibin-A), ultrasound measurements (nuchal translucency), and maternal demographic information (age, ethnicity, family history). Non-invasive prenatal testing (NIPT) data includes fetal DNA sequencing results. Diagnostic test outcomes (CVS, amniocentesis) confirm the presence of Trisomy 21. Additional features may encompass pregnancy outcomes and genetic profiles. The dataset aims to improve predictive models and is used for training AI algorithms to enhance early detection and decision-making in prenatal care

class in the Dataset:

- **Down Syndrome(non standrad)**
    - **Description**: An image of Down Syndrome often shows distinct facial features such as a flat facial profile, upward slanting eyes, a small nose, and a single deep crease across the center of the palm. These physical characteristics are common indicators of the genetic condition. The image may also illustrate associated physical and developmental traits

    - **Image Characteristics**: A Down Syndrome image typically shows a flat facial profile, upward slanting eyes, a small nose, and a single deep crease on the palm.
- **Healthy(standrad)**
    - **Description**: A healthy image generally depicts clear skin, bright eyes, and a well-proportioned body with no visible signs of illness. It conveys vitality, normal growth, and well-being..
    - **Image Characteristics**: A healthy image typically features clear, radiant skin, bright eyes, and a well-balanced body. It reflects overall physical well-being and normal development

Number of Images Related to Each Class

Train data images :

standrad : 196

non standrad : 150

valid :

standrad : 196

non standrad : 150

Test :

standrad : 11

non standrad : 11

Detailed Dataset Information

- **Total Number of Images**: 368
- **Image Resolution**: High-resolution images (e.g., 274*274 pixels) to ensure detailed feature extraction.
- **File Format**: PNG for lossless quality.
- **Annotations**: Each image is labelled with its respective variety name for supervised learning tasks.

- ## Methodology :

Code:

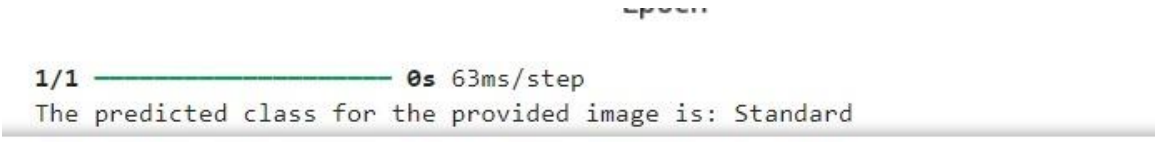
```
import tensorflow as tf
import matplotlib.pyplot as plt
import cv2
import os
import numpy as np
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing import image
from PIL import Image
from keras.models import load_model

# Define the image size and categories
img_height, img_width = 224, 224
categories = ['Standard', 'Nonstandard']

# Data preparation
train = ImageDataGenerator(rescale=1/255)
validation = ImageDataGenerator(rescale=1/255)
train_dataset = train.flow_from_directory(
    r'C:\Users\jaid\OneDrive\Desktop\DIP_Pro\Train2',
    target_size=(img_height, img_width),
    batch_size=9,
    class_mode='categorical'
)
validation_dataset = validation.flow_from_directory(
    r'C:\Users\jaid\OneDrive\Desktop\DIP_Pro\Validation2',
    target_size=(img_height, img_width),
```

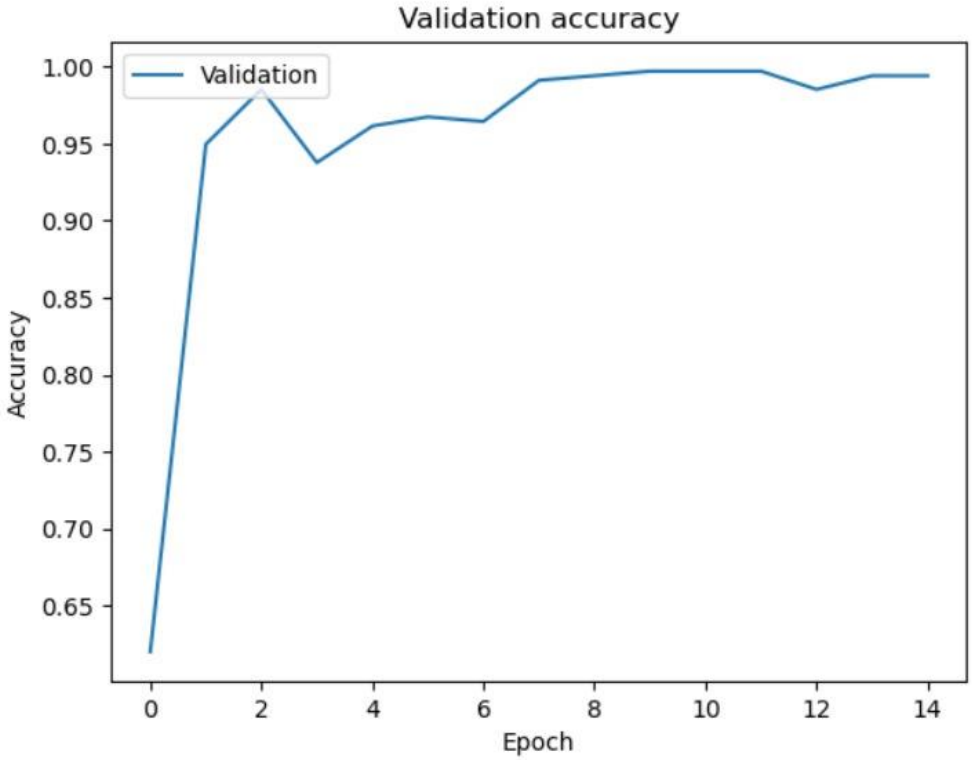
```
    batch_size=9,
    class_mode='categorical'
)

# Define the model
num_classes = 2  # Set this to the correct number of classes
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(16, (3, 3), activation='relu', input_shape=(img_height, img_width, 3)),
    tf.keras.layers.MaxPool2D(2, 2),
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu'),
    tf.keras.layers.MaxPool2D(2, 2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(256, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(num_classes, activation='softmax')
])

# Compile the model
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# Train the model
model_fit = model.fit(train_dataset, steps_per_epoch=8, epochs=15, validation_data=validation_dataset)

# Save the model
```

```
model.save('model.h5')

# Plotting training results
plt.plot(model_fit.history['accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train'], loc='upper left')
plt.show()

plt.plot(model_fit.history['loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train'], loc='upper left')
plt.show()

plt.plot(model_fit.history['val_accuracy'])
plt.title('Validation accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Validation'], loc='upper left')
plt.show()

plt.plot(model_fit.history['val_loss'])
plt.title('Validation loss')
plt.ylabel('Loss')
```

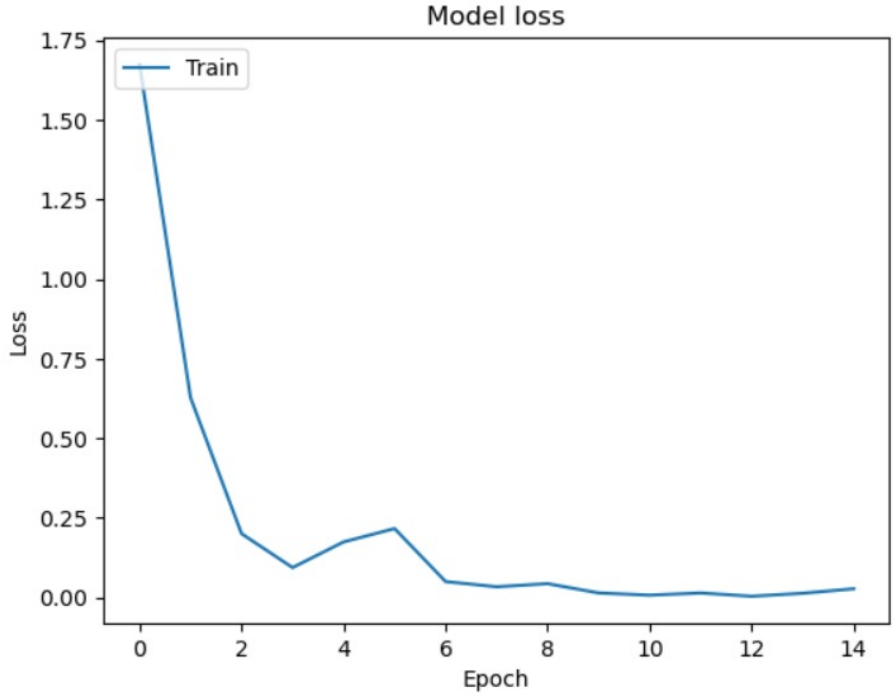
```
plt.xlabel('Epoch')
plt.legend(['Validation'], loc='upper left')
plt.show()

# Function to predict image class
def predict_image_class(image_path):
    img = image.load_img(image_path, target_size=(img_height, img_width))
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0) / 255.0

    predictions = model.predict(img_array)
    predicted_class = categories[np.argmax(predictions)]

    return predicted_class

# Example usage
image_path = r'C:\Users\jaid\OneDrive\Desktop\DIP_Pro\Test2\Standard\17.png' # Update with the path to your test image
predicted_class = predict_image_class(image_path)
print(f'The predicted class for the provided image is: {predicted_class}')
```

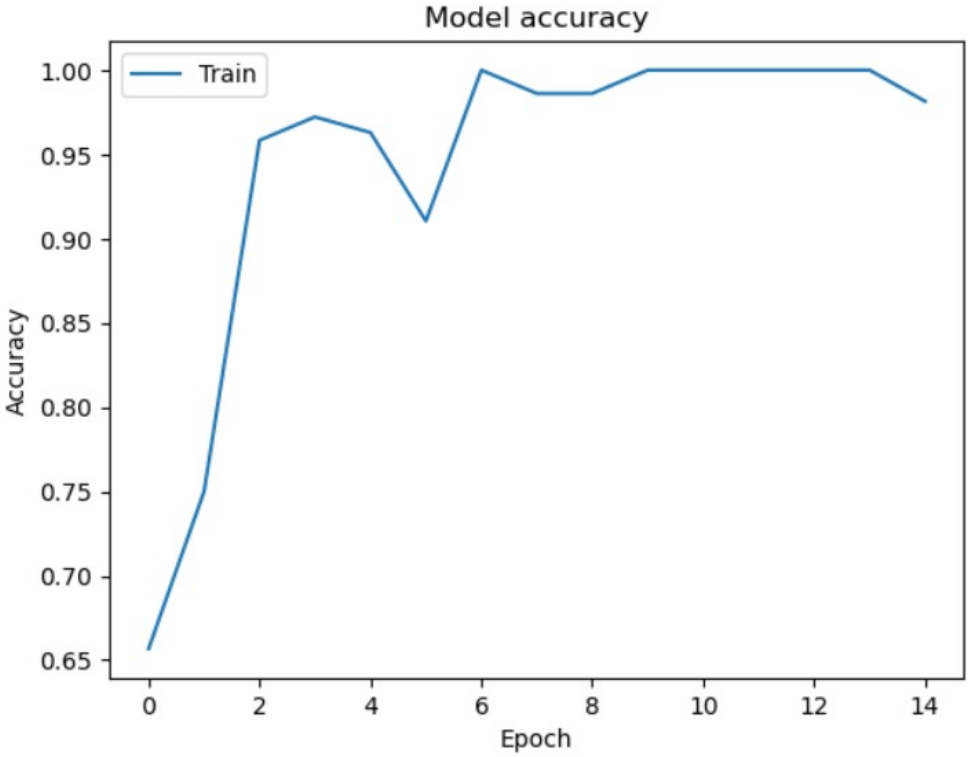
Accuracy and Loss Graphs and table :

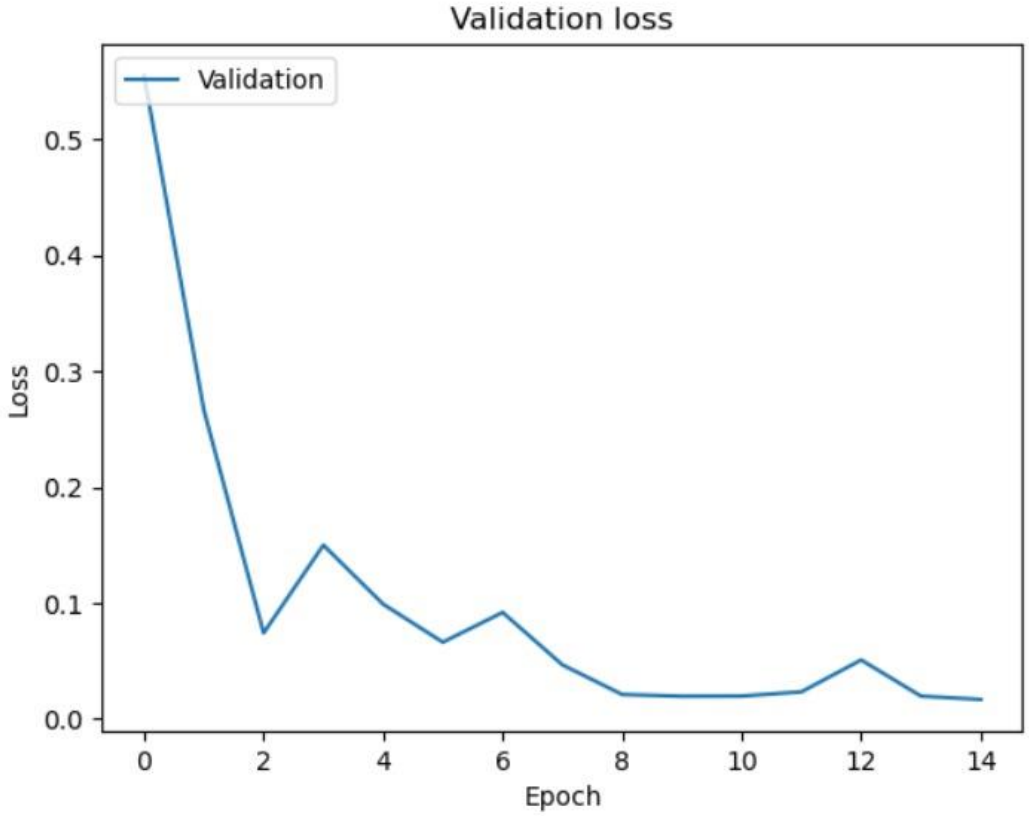



Validation accuracy
Validation
Accuracy
Epoch
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0
2
4
6
8
10
12
14

SSS


Model loss
Train
Loss
Epoch
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
2
4
6
8
10
12
14


Model accuracy
Train
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
Accuracy
0
2
4
6
8
10
12
14
Epoch


Validation loss
Validation
0.5
0.4
0.3
0.2
0.1
0.0
Loss
0
2
4
6
8
10
12
14
Epoch

- **Conclusion :**

  Predicting Down Syndrome involves a combination of prenatal screening, advanced genomic technologies, and diagnostic tests to assess the risk of Trisomy 21. Non-invasive methods like NIPT have significantly improved prediction accuracy. Ethical considerations and equitable access to testing are crucial for informed decision-making. Continued advancements in technology and personalized care are enhancing early detection and prenatal care outcomes