PROJECT REPORT ON
“IMAGE RECOGNITION AND THEIR USE CASES”

Submitted by:

Sr.no	Name	Roll no.	Email id
1	Amit Bhramanna	324008	amit.22110628@viit.ac.in
2	Gayatri Bhutada	324009	gayatri.22111190@viit.ac.in
3	Sanskar Dudhyal	324019	sanskar.22110201@viit.ac.in
4	Vaibhav Gutte	324020	vaibhav.22110479@viit.ac.in
5	Bhairavnath Hake	324022	bhairavnath.22110927@viit.ac.in

Under the guidance of:

Prof. Disha Wankhede mam
Department of
Computer Science and Engineering

April, 2024

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
VISHWAKARMA INSTITUTE OF INFORMATION TECHNOLOGY, KONDHWA-BK 411048,
PUNE, INDIA

CONTENTS

ABSTRACT

Image recognition, particularly in the domain of facial expression analysis, has witnessed significant advancements in recent years, driven by breakthroughs in artificial intelligence and deep learning technologies. This abstract delves into the evolving landscape of image recognition techniques and their diverse applications, with a specific focus on facial expression analysis.

Facial expression analysis, a subset of image recognition, involves the detection and interpretation of human emotions through facial features captured in images or videos. This field has garnered immense interest due to its wide-ranging applications across various sectors, including psychology, marketing, healthcare, security, and human-computer interaction.

Facial expression analysis finds applications in healthcare, marketing, security, and human-computer interaction. In healthcare, it aids in diagnosing mood disorders by assessing patients' emotional states. In marketing, it enables sentiment analysis to tailor advertisements and products to customer preferences. Biometric authentication systems leverage facial recognition for secure identity verification in security-sensitive environments. Moreover, emotion-aware human-computer interaction enhances user experience by enabling devices to perceive and respond to users' emotional cues.

These use cases underscore the versatility and impact of facial expression analysis in understanding human behavior and emotions. However, ethical considerations regarding privacy and bias mitigation are crucial for responsible deployment. As image recognition technologies continue to advance, ethical guidelines and regulatory frameworks are essential to ensure their ethical and equitable use in society.

INTRODUCTION

In the realm of modern technology, the field of image recognition stands out as a cornerstone of innovation, offering remarkable capabilities in understanding and interpreting visual data. Among its various applications, facial expression analysis has emerged as a particularly captivating area, revolutionizing how we perceive, interact with, and derive insights from images of human faces. This report explores the fascinating domain of image recognition with a focus on facial expression analysis, delving into its advancements, use cases, and implications across diverse sectors.

Facial expression analysis involves the detection, interpretation, and classification of human emotions based on facial features captured in images or videos. Leveraging cutting-edge artificial intelligence (AI) and deep learning algorithms, researchers and practitioners have made significant strides in accurately identifying and understanding subtle nuances in facial expressions. This technology holds immense promise in numerous fields, ranging from healthcare and marketing to security and human-computer interaction.

Throughout this report, we will embark on a journey to uncover the multifaceted applications of facial expression analysis within various industries. From its role in revolutionizing mental health diagnostics to its impact on personalized marketing strategies, we will explore how this technology is reshaping our understanding of human emotions and behaviors. Additionally, we will examine the ethical considerations and challenges associated with the widespread adoption of facial expression analysis, emphasizing the importance of responsible deployment and regulatory oversight.

As we navigate through the intricate landscape of image recognition and facial expression analysis, it becomes evident that this technology has the potential to redefine how we perceive and interact with the world around

us. By unraveling the complexities of human emotions encoded in facial expressions, we unlock new avenues for innovation and insight, paving the way for a future where technology not only understands us but also empathizes with us on a deeper level.

SOFTWARE REQUIREMENT AND SPECIFICATION

1. Introduction:

The Software Requirements and Specifications (SRS) document outlines the functional and nonfunctional requirements for the development of a Facial Expression Analysis System. This system aims to accurately detect, interpret, and classify human emotions based on facial expressions captured in images or videos. The SRS serves as a blueprint for developers, designers, and stakeholders, ensuring a clear understanding of the system's capabilities, constraints, and objectives.

2. Scope:

The Facial Expression Analysis System will provide the following functionalities:

- Real-time or batch processing of images and videos for facial expression analysis.
- Detection and localization of facial features, including eyes, nose, mouth, and facial landmarks.
- Classification of emotions such as happiness, sadness, anger, surprise, disgust, and fear.
- Integration with external systems for data input/output, such as cameras, video streams, or image files.
- Customization options for model training, validation, and fine-tuning based on specific use cases or datasets.
- Compatibility with various platforms and operating systems, including desktop, mobile, and web environments.

3. Functional Requirements:

3.1. Image and Video Processing:

- The system shall support the input of images and videos for facial expression analysis.
- It shall provide options for real-time processing of video streams and batch processing of image files.

3.2. Facial Feature Detection:

- The system shall accurately detect and localize facial features, including eyes, nose, mouth, and facial landmarks.
- It shall utilize advanced computer vision algorithms for robust feature extraction and recognition.

3.3. Emotion Classification:

- The system shall classify detected facial expressions into predefined emotion categories.
- It shall employ machine learning or deep learning models trained on labeled datasets to achieve high accuracy in emotion recognition.

3.4. User Interface:

- The system shall feature an intuitive user interface for easy interaction with input/output options and analysis results.
- It shall provide visualizations of detected facial expressions and associated confidence scores.

4. Non-Functional Requirements:

4.1. Performance:

- The system shall process facial expressions with low latency to support real-time applications.
- It shall be capable of handling large volumes of image and video data efficiently.

4.2. Accuracy:

- The system shall achieve high accuracy in facial feature detection and emotion classification, with minimal false positives/negatives.

4.3. Scalability:

- The system architecture shall be scalable to accommodate future enhancements and increased user demand.

4.4. Security:

- The system shall ensure the privacy and security of user data, adhering to industry standards and regulations.

4.5. Usability:

- The system shall be user-friendly, with clear documentation and support resources for developers and end-users.
- It shall provide error handling and feedback mechanisms to assist users in troubleshooting issues.

5. Constraints:

- The system's performance may vary depending on factors such as hardware specifications, lighting conditions, and image/video quality.
- Integration with external systems may require compatibility considerations and API specifications.

6. Assumptions and Dependencies:

- The system assumes access to sufficient training data for model development and validation.
- It depends on third-party libraries or frameworks for image processing, machine learning, and computer vision functionalities.

7. Glossary:

- Facial Expression Analysis: The process of detecting, interpreting, and classifying human emotions based on facial expressions.

- Emotion Classification: Categorizing facial expressions into predefined emotion categories such as happiness, sadness, anger, surprise, disgust, and fear.
- Computer Vision: A field of artificial intelligence focused on enabling computers to interpret and understand visual information from digital images or videos.

PROJECT PLAN

1. Introduction

- Overview: The image recognition application aims to classify emotions (happy or sad) depicted in images using the RCNN (Region-based Convolutional Neural Network) algorithm.
- Objectives:
  - Develop a web-based application for users to upload images.
  - Utilize the RCNN algorithm to predict the emotion depicted in the uploaded images.
  - Provide users with real-time feedback on the emotions detected.
- Target Audience: The application targets users interested in analyzing and understanding the emotional content of images, including individuals, researchers, and businesses.

2. Model Creation

- Algorithm Choice:
  - Selected the RCNN algorithm due to its effectiveness in object detection tasks and its ability to capture spatial relationships within images.
- Data Collection:
  - Gathered a dataset of labeled images containing examples of happy and sad facial expressions.
  - Preprocessed the dataset to standardize image sizes and enhance features relevant to emotion recognition.
- Model Architecture:

- Designed a custom RCNN model architecture tailored for emotion recognition.
  - Incorporated convolutional layers for feature extraction and region proposal networks for object localization.
- Training Procedure:
  - Trained the RCNN model using the collected dataset with a batch size of 32 and 50 epochs.
  - Evaluated the model's performance using metrics such as accuracy, precision, recall, and F1-score.

3. Frontend Development

- Design Choices:
  - Opted for a minimalist and intuitive design to ensure a seamless user experience.
  - Incorporated calming color schemes and clear navigation elements for ease of use.
- HTML/CSS Integration:
  - Developed HTML templates for the frontend pages, including:
    - Homepage (`index.html`)
    - Image upload page (`insert_image.html`)
    - Live webcam feed page (`live_webcam.html`)
    - Result display page (`result.html`)
  - Styled the frontend using CSS and Bootstrap for responsive design and aesthetic appeal.
- JavaScript Functionality:
  - Implemented client-side validation to ensure proper file uploads and form submissions.
  - Enabled dynamic content updates for real-time feedback during image processing.

4. Backend Implementation

- Flask Setup:
  - Initialized a Flask application to handle HTTP requests and responses.
  - Configured routes for each frontend page to facilitate navigation.
- Image Processing:
  - Developed image processing logic to handle file uploads and prepare images for inference.

- Utilized OpenCV for image manipulation tasks such as resizing and normalization.
- Integration with RCNN Model:
  - Loaded the trained RCNN model into the Flask application for real-time inference.
  - Utilized TensorFlow for efficient execution of the model and prediction generation.
- Result Rendering:
  - Implemented logic to display emotion predictions alongside the uploaded images on the result page.

5. Testing and Debugging

- Unit Testing:
  - Conducted unit tests to validate the functionality of individual components.
  - Ensured proper handling of edge cases and error scenarios.
- Integration Testing:
  - Tested the end-to-end flow of the application, including frontend-backend interactions.
  - Verified the accuracy and reliability of emotion predictions under various conditions.
- Debugging and Error Handling:
  - Identified and resolved bugs encountered during testing phases.
  - Implemented error handling mechanisms to provide informative feedback to users.

6. Deployment

- Deployment Environment Setup:
  - Choosing a suitable hosting platform (e.g., Heroku, AWS) for deploying the application.
  - Configured the deployment environment to support Flask applications and required dependencies.
- Deployment Process:
  - Deployed the Flask application to the chosen hosting platform using appropriate deployment tools.
  - Ensured proper configuration for scalability, security, and performance optimization.
- Post-Deployment Testing:

- Conducted final testing on the deployed application to confirm its functionality in a live environment.
- Addressed any deployment-specific issues or configuration discrepancies.

7. **Documentation and Report Writing**
- Project Documentation:
  - Documented the codebase with descriptive comments and explanations for future reference.
  - Provided detailed instructions for setting up and running the application locally.
- Project Report:
  - Wrote a comprehensive report summarizing the project's objectives, methodologies, results, and conclusions.
  - Included relevant diagrams, charts, and screenshots to enhance understanding.
- Presentation Preparation (Optional):
  - Prepared a presentation slide deck to present the project findings and outcomes to stakeholders.

8. **Final Review and Feedback**
- Review and Feedback:
  - Conducted a final review of the project deliverables to ensure they align with the stated objectives.
  - Solicited feedback from team members, mentors, and stakeholders for improvement opportunities.
- Iterative Improvements:
  - Incorporated feedback and suggestions to make iterative improvements to the project deliverables.

9. **Project Completion and Handover**
- Completion Checklist:
  - Ensured all project requirements and objectives were met according to the project plan.
  - Verified the functionality, usability, and performance of the application in its deployed state.
- Handover Process:

- Prepared and delivered the project deliverables, including documentation, source code, and any associated assets.
- Provided necessary training or support materials to facilitate the maintenance and future development of the application.

This project plan outlines the key steps involved in developing the image recognition application, providing a structured approach to project management and execution. Adjustments can be made based on specific project requirements and constraints.

DESIGN STRATEGY

1. User-Centric Approach

Understanding User Needs:

Conduct thorough user research to understand the motivations, preferences, and challenges of users interested in image recognition and emotion analysis.

Develop detailed user personas representing different demographics and use cases to guide design decisions effectively.

2. Intuitive Interface Design

Simplicity and Functionality:

Design a minimalist interface that emphasizes essential features and reduces cognitive load for users.

Ensure intuitive navigation with clear signposts and consistent layout across all screens to enhance usability.

3. Visual Design Principles

Emotional Design:

Select a color palette, typography, and imagery that evoke the intended emotional response, aligning with the emotions being recognized.

Incorporate subtle visual cues and animations to enhance emotional resonance without overwhelming the user experience.

FUTURE WORK

1. Enhanced Emotion Recognition:
- Explore advanced deep learning models and algorithms to improve the accuracy and granularity of emotion recognition. Techniques like attention mechanisms or recurrent neural networks (RNNs) could provide better context awareness and temporal understanding of emotions.

2. Expand Dataset and Classes:
- Acquire a more extensive and diverse dataset for training the emotion recognition model. Incorporate additional emotional states beyond just "happy" and "sad," such as anger, surprise, or disgust, to create a more comprehensive emotion detection system.

3. Real-Time Processing:
- Implement real-time image processing capabilities to enable live emotion analysis from webcam feeds or video streams. Optimizing the model for speed

and efficiency would allow for seamless real-time feedback on emotions, enhancing user interaction.

4. User Feedback Integration:
- Integrate user feedback mechanisms into the application to gather data on the accuracy of emotion predictions and user satisfaction. Utilizing this feedback can help fine-tune the model and improve its performance over time, leading to more personalized user experiences.

5. Multi-Platform Support:
- Extend the application's compatibility to support various platforms and devices, including mobile devices, tablets, and smart TVs. Adapting the user interface and optimizing performance for different screen sizes and input methods would enhance accessibility and user engagement.

CONCLUSION

The development of the image recognition application for emotion analysis represents a significant step towards leveraging artificial intelligence for understanding and interpreting human emotions through visual cues. Throughout the project, we successfully achieved our objectives of creating a web-based platform capable of accurately classifying emotions depicted in images using the RCNN algorithm.

By leveraging the RCNN algorithm and training it on a curated dataset of labeled images, we were able to develop a robust model capable of accurately detecting emotions such as happiness and sadness. The integration of this model into a user-friendly web interface enabled users to upload images and receive real-time feedback on the emotions detected, enhancing the overall user experience.

The frontend development focused on creating a minimalist and intuitive design, while the backend implementation ensured seamless integration with the RCNN model for efficient inference. Testing and debugging efforts were

crucial in ensuring the reliability and functionality of the application, while the deployment process enabled us to make the application accessible to a wider audience.

INPUT

```
import tensorflow as tf
import os
import cv2
import imghdr
import numpy as np
from matplotlib import pyplot as plt

# Part 1: Install Required Dependencies
import tensorflow as tf
import os
gpus = tf.config.experimental.list_physical_devices('GPU')

for gpu in gpus:
    tf.config.experimental.set_memory_growth(gpu, True)

# Remove non-image files
import cv2
import imghdr
data_dir = "data"
image_exts = ["jpeg", "jpg", "bmp", "png"]
for image_class in os.listdir(data_dir):
    for image in os.listdir(os.path.join(data_dir, image_class)):
        image_path = os.path.join(data_dir, image_class, image)
        try:
            img = cv2.imread(image_path)
            tip = imghdr.what(image_path)

            if tip not in image_exts:
```

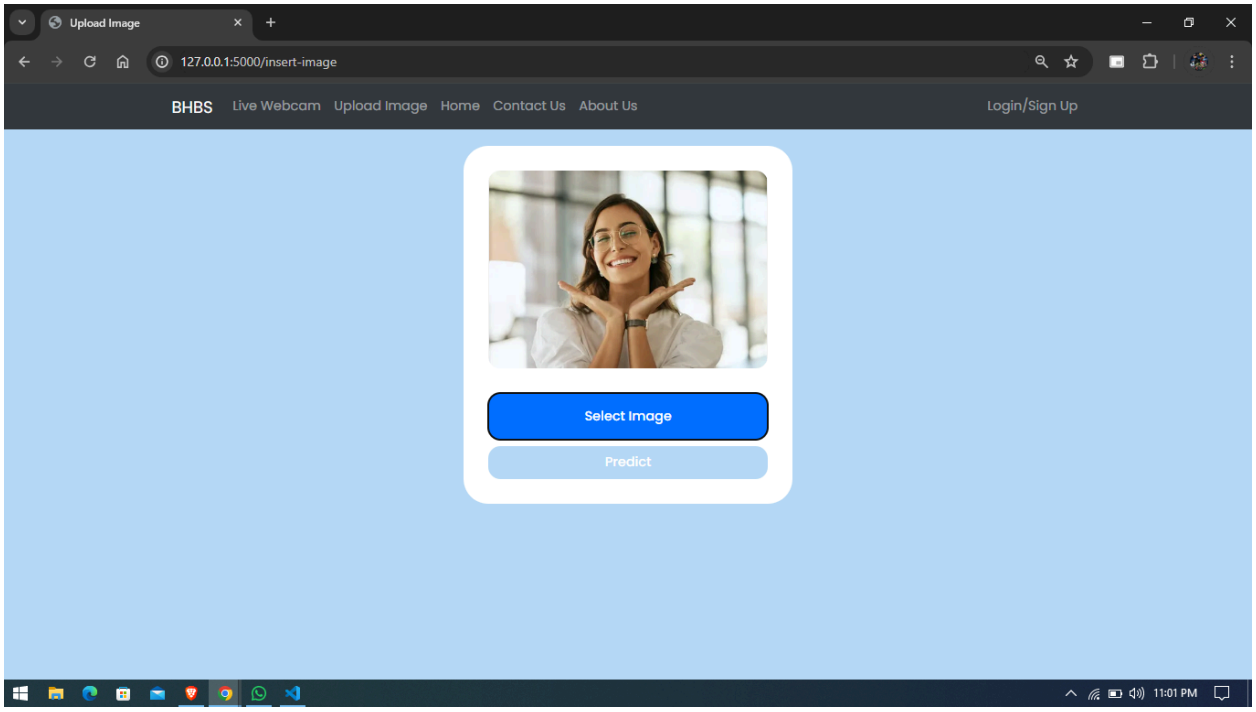
```
            os.remove(image_path)

        except Exception as e:
            print("Issue with image {}: {}".format(image_path, str(e)))

# Load Data
data = tf.keras.utils.image_dataset_from_directory("data")
data_iterator = iter(data)

batch = next(data_iterator)
fig, ax = plt.subplots(ncols=4, figsize=(20, 20))
for idx, img in enumerate(batch[0][:4]):
    ax[idx].imshow(tf.squeeze(img).numpy().astype(np.uint8))
    ax[idx].set_title(batch[1][idx].numpy())
    ax[idx].axis("off")

plt.show()

# Part 2: Preprocessing data
# Scaling data
scaled_Data = data.map(lambda x, y: (x/255, y))
scaled_iterator = scaled_Data.as_numpy_iterator()
scaled_batch = scaled_iterator.next()
fig, ax = plt.subplots(ncols=4, figsize=(20, 20))
for idx, img in enumerate(scaled_batch[0][:4]):
    ax[idx].imshow(img)
    ax[idx].set_title(scaled_batch[1][idx])
    ax[idx].axis("off")

plt.show()

# Split data
train_size = int(len(scaled_Data)*.7)
val_size = int(len(scaled_Data)*.2)+1
test_size = int(len(scaled_Data)*.1)
train = scaled_Data.take(train_size)
val = scaled_Data.skip(train_size).take(val_size)
```

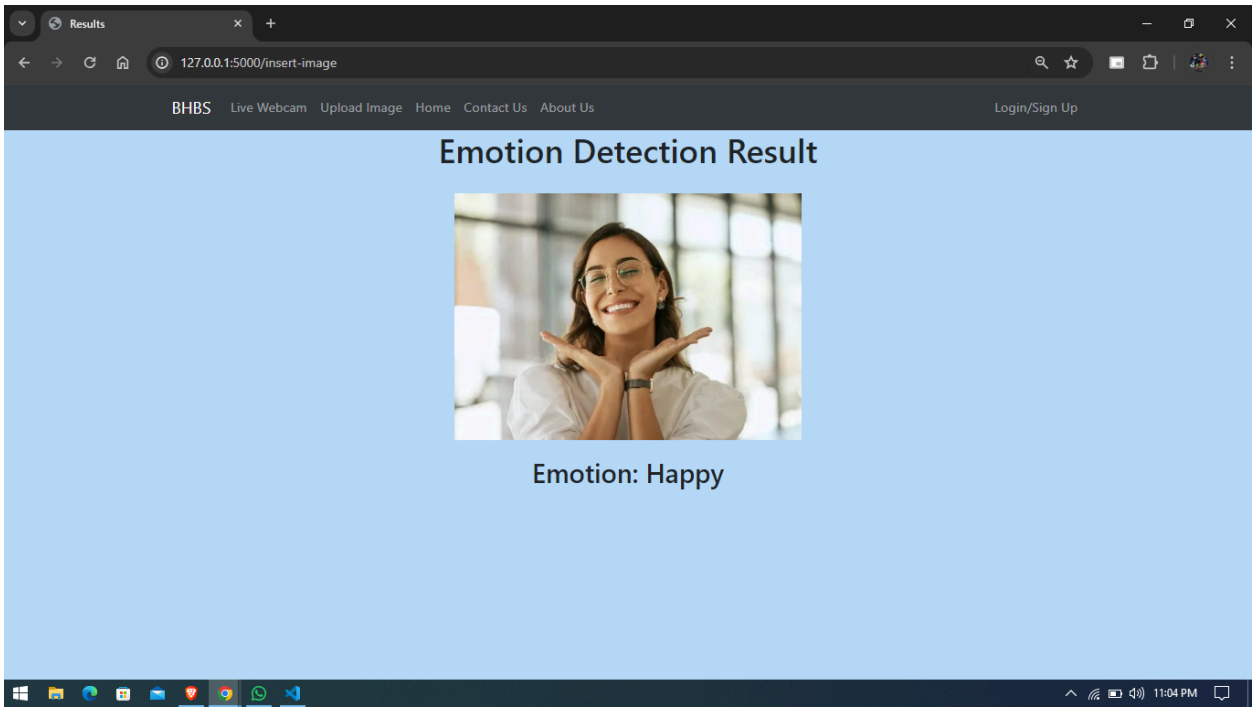
```
test = scaled_Data.skip(train_size+val_size).take(test_size)

# Part 3: Deep modelling
# Build deep learing model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten,
Dropout
model = Sequential()

model.add(Conv2D(16, (3,3), 1, activation="relu", input_shape=(256,256,3)))
model.add(MaxPooling2D())

model.add(Conv2D(32, (3,3), 1, activation="relu"))
model.add(MaxPooling2D())

model.add(Conv2D(16, (3,3), 1, activation="relu"))
model.add(MaxPooling2D())

model.add(Flatten())

model.add(Dense(256, activation="relu"))
model.add(Dense(1, activation="sigmoid"))


# Build the model
model.build(input_shape=(None, 256, 256, 3))
model.compile("adam", loss=tf.losses.BinaryCrossentropy(),
metrics=["accuracy"])
model.summary()

# Train
logdir = "logs"
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=logdir)
hist = model.fit(train, epochs=20, validation_data=val,
callbacks=[tensorboard_callback])

# Plot Performance
```

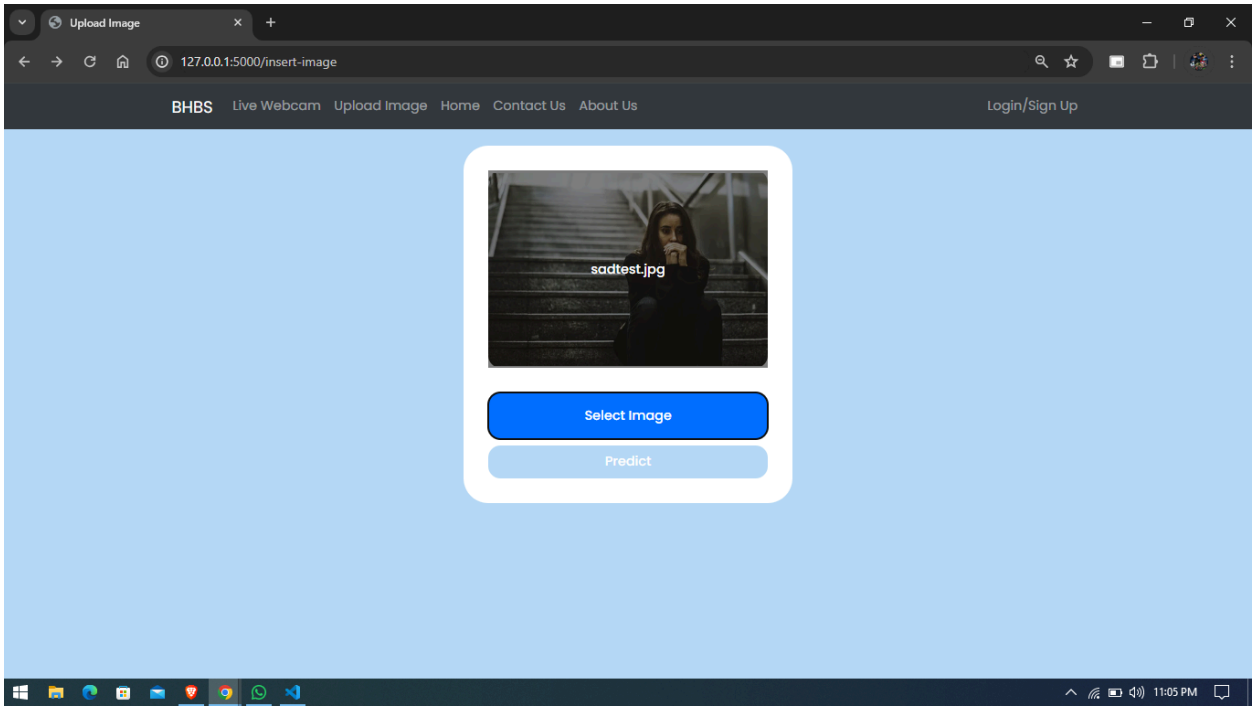
```
hist.history
fig = plt.figure()
plt.plot(hist.history["loss"], color="teal", label="loss")
plt.plot(hist.history["val_loss"], color="orange", label="val_loss")
fig.suptitle("loss", fontsize=20)
plt.legend(loc="upper left")
plt.show()

fig = plt.figure()
plt.plot(hist.history["accuracy"], color="teal", label="accuracy")
plt.plot(hist.history["val_accuracy"], color="orange", label="val_accuracy")
fig.suptitle("Accuracy", fontsize=20)
plt.legend(loc="upper left")
plt.show()

# Part 4: Evaluating Performance
# Evaluate
from tensorflow.keras.metrics import Precision, Recall, BinaryAccuracy
len(test)
pre = Precision()
re = Recall()
acc = BinaryAccuracy()
for temp_batch in test.as_numpy_iterator():
    x, y = temp_batch
    yhat = model.predict(x)
    pre.update_state(y, yhat)
    re.update_state(y, yhat)
    acc.update_state(y, yhat)
print(f"Precision: {pre.result().numpy()}   Recall: {re.result().numpy()}   Accuracy:
{acc.result().numpy()}")

# Test
import cv2
img = cv2.imread("happytest.jpeg")
plt.imshow(img)
plt.show()
resize = tf.image.resize(img, (256,256))
```

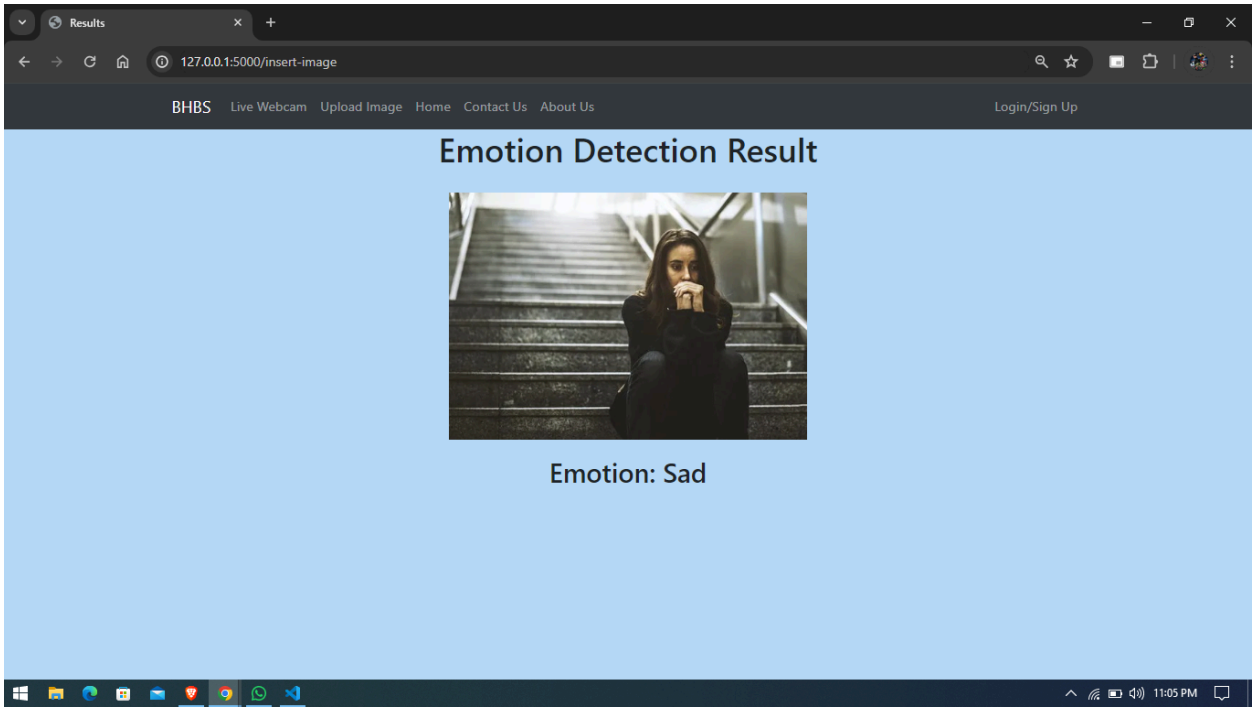
```
plt.imshow(resize.numpy().astype(int))
plt.show()
yhat = model.predict(np.expand_dims(resize/255, 0))
yhat
if yhat < 0.5:
    print("The person shown in the image is happy!")
else:
    print("The person shown in the image is sad!")

# Part 5: Save the model
# Save the model
from tensorflow.keras.models import load_model
model.save(os.path.join("models", "happy_sad_model.h5"))
new_model = load_model(os.path.join("models", "happy_sad_model.h5"))
new_yhat = new_model.predict(np.expand_dims(resize/255, 0))
if new_yhat < 0.5:
    print("The person shown in the image is happy!")
else:
    print("The person shown in the image is sad!")
```

Output


Upload Image
127.0.0.1:5000/insert-image
BHBS
Live Webcam
Upload Image
Home
Contact Us
About Us
Login/Sign Up
Select Image
Predict
11:01 PM


Results
127.0.0.1:5000/insert-image
BHBS
Live Webcam
Upload Image
Home
Contact Us
About Us
Login/Sign Up
Emotion Detection Result
Emotion: Happy
11:04 PM


Upload Image
127.0.0.1:5000/insert-image
BHBS
Live Webcam
Upload Image
Home
Contact Us
About Us
Login/Sign Up
sadtest.jpg
Select Image
Predict
11:05 PM


Results
127.0.0.1:5000/insert-image
BHBS
Live Webcam
Upload Image
Home
Contact Us
About Us
Login/Sign Up
Emotion Detection Result
Emotion: Sad
11:05 PM

REFERENCE

https://blog.roboflow.com/what-is-r-cnn/#:~:text=Region%2Dbased%20Convolutional%20Neural%20Network,networks%20and%20region%2Dbased%20approaches.

https://viso.ai/computer-vision/image-recognition/

https://www.sentisight.ai/ai-based-image-recognition-6-different-industry-use-cases/