OVERVIEW

- **ABSTRACT**
- **INTRODUCTION**
- **LITERATURE AND SURVEY**
- **METHODOLOGY**
- **CODE**
- **RESULT AND ANALYSIS**
- **CONCLUSION AND FUTURE WORK**
- **REFERENCES**

ABSTRACT

This project explores a machine learning approach for real-time age detection using facial images captured by a webcam. By employing facial recognition and a linear regression model, it aims to estimate a person's age based on their facial features.

Objective: The primary goal is to develop a system capable of predicting an individual's age by analysing facial images in real-time. This is achieved through a webcam, which provides a live feed for age estimation.

Data and Preprocessing: The training process is handled by the script **train_age_model.py**, which preprocesses the images, splits the dataset into training and testing subsets, and trains a linear regression model. The dataset, contained in **data.csv**, includes facial images labeled with corresponding ages. This data underwent thorough preprocessing, including normalization and feature extraction, to enhance the model's accuracy.

Model: A linear regression model was chosen for its simplicity and effectiveness in mapping facial features to age. The model was trained on the processed dataset and has been saved as **age_detection_model.pkl** for deployment.

Real-Time Application: The project includes a script for live age detection, which utilizes OpenCV for the trained model for age prediction. This allows for real-time age estimation from a webcam feed. The project includes a script (**real_time_age_detection.py**) for deploying the age detection model in real-time.

The project demonstrates the practical application of linear regression in age detection and highlights the integration of real-time facial analysis for various potential uses.

INTRODUCTION

Age detection is a fundamental task in various domains such as facial recognition, demographic analysis, and targeted marketing. It plays a crucial role in understanding user behavior, providing personalized services, and ensuring age-appropriate content delivery.

In recent years, with the proliferation of digital technologies and the advent of deep learning techniques, age detection has become increasingly accurate and efficient. One such technique that has gained prominence is linear regression.

Linear regression is a simple yet powerful statistical method used for modeling the relationship between a dependent variable and one or more independent variables. In the context of age detection, linear regression can be employed to predict a person's age based on certain features or attributes.

The basic premise of using linear regression for age detection is to train a model using a dataset containing samples of individuals along with their corresponding ages. The model learns the underlying patterns and relationships between the input features and the target variable .

SOME FORMULAES ON LINEAR REGRESSION :-

- Y = mx + c , where m denotes the slope (coef) , and c denotes the intercept .
- **Simple Linear Regression:**
  - Model: $y = \beta_0 + \beta_1 x + \varepsilon$
  - Slope (β_1): $\beta_1 = \frac{\sum_{i-1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i-1}^{n}(x_i - \bar{x})^2}$
  - Intercept (β_0): $\beta_0 = \bar{y} - \beta_1 \bar{x}$
  - Residuals (ε_i): $\varepsilon_i = y_i - (\beta_0 + \beta_1 x_i)$
- **Multiple Linear Regression:**
  - Model: $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_m x_m + \varepsilon$
  - Coefficients (β): $\boldsymbol{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$
  - Residuals (ε): $\varepsilon = \mathbf{y} - \mathbf{X}\boldsymbol{\beta}$

LITERATURE SURVEY

1. Introduction

Age detection is a critical task with applications in various fields such as healthcare, education, and social sciences. Incorporating features like height, weight, and number of siblings alongside linear regression offers a unique perspective for age prediction. This literature survey aims to explore existing research on age detection models using linear regression with height, weight, and siblings information.

2. Incorporation of Anthropometric Features

Anthropometric features such as height and weight have been widely studied in age estimation literature due to their strong correlation with age. Linear regression models incorporating these features have shown promising results, particularly in applications like pediatric growth monitoring and age estimation in forensic anthropology.

3. Impact of Sibling Information

The number of siblings can provide additional contextual information that influences age perception. Studies have explored the incorporation of sibling-related features, such as birth order or number of siblings, in age detection models. Linear regression, when combined with sibling information, can capture societal norms and cultural influences on age perception.

4. Dataset Construction and Annotation

Building datasets for age detection models incorporating height, weight, and sibling information requires careful curation and annotation. Researchers have developed specialized datasets containing anthropometric measurements, sibling demographics, and corresponding age labels. These datasets serve as valuable resources for training and evaluating linear regression models.

5. Model Architectures and Feature Engineering

Linear regression models can benefit from advanced feature engineering techniques, including feature scaling, normalization, and interaction terms. Additionally, ensemble methods and hybrid architectures combining linear regression with other machine learning algorithms have been explored to improve age prediction accuracy.

6. Evaluation Metrics and Performance Benchmarks

Evaluation of age detection models utilizing linear regression with height, weight, and sibling information often employs metrics such as mean absolute error (MAE), root mean squared error

(RMSE), and coefficient of determination (R-squared). Benchmarking against existing state-of-the-art models provides insights into the efficacy of the proposed approach.

7. Applications and Use Cases

The integration of height, weight, and sibling information into age detection models enhances their applicability across various domains. Potential applications include age estimation in healthcare settings, age verification for online platforms, and age-sensitive marketing strategies.

8. Challenges and Future Directions

Despite the promising results, challenges remain in effectively leveraging height, weight, and sibling information for age detection. These include addressing biases in dataset construction, handling missing or noisy data, and ensuring model interpretability. Future research directions may involve exploring deep learning architectures capable of capturing complex relationships among features.

METHODOLOGY

1. Face Detection

Objective: Identify and locate faces within the input images or video streams.

- **Techniques**:
  - **Haar Cascades**: Utilizes pre-trained classifiers to detect faces based on features like edges and textures.
  - **Deep Learning-Based Detectors**: Advanced techniques such as Single Shot MultiBox Detector (SSD) and Faster R-CNN are employed to locate faces with higher accuracy and robustness.
- **Process**: The face detection algorithm scans the input frame to identify facial regions, generating bounding boxes around detected faces.

2. Region of Interest (ROI) Extraction

Objective: Isolate the face region for further processing.

- **Steps**:
  - Extract the face region from the bounding box generated by the face detection algorithm.
  - This isolated region serves as the input for subsequent preprocessing and feature extraction.

3. Image Preprocessing

Objective: Prepare the face image for feature extraction by standardizing and enhancing it.

- **Steps**:
  - **Normalization**: Adjust pixel values to a standard range, typically [0, 1] or [-1, 1], to ensure uniformity.
  - **Resizing**: Resize the face image to a consistent size (e.g., 224x224 pixels) to match the input dimensions expected by the model.
  - **Transformations**: Apply transformations such as histogram equalization to improve contrast and highlight relevant features, aiding in the detection of age-related cues.

4. Feature Extraction

Objective: Extract critical features from the preprocessed face image that correlate with age.

- **Techniques**:
  - **Facial Landmarks**: Identify key points like eyes, nose, and mouth which are crucial for age estimation.

- **Texture Patterns**: Analyze skin texture to detect age-related features such as wrinkles and skin tone variations.
- **Advanced Cues**: Capture finer details that may indicate age, such as the presence of age spots or facial sagging.

5. Age Classification Model

Objective: Predict the age group using a trained machine learning model.

- **Model Architecture**:
  - **Convolutional Neural Networks (CNNs)**: Utilized for their efficacy in image classification tasks.
  - The model may be trained from scratch or a pre-trained model can be fine-tuned for the specific task.
- **Age Groups**:
  - The model classifies faces into predefined age groups (e.g., 0-18, 19-35, 36-60, 61+) to simplify the problem into a classification task rather than exact age regression.

6. Inference and Prediction

Objective: Generate age predictions from the input face images.

- **Steps**:
  - Pass the preprocessed face image through the trained age classification model.
  - Obtain the predicted age group for each detected face.

7. Post-processing and Visualization

Objective: Map model predictions to a user-friendly format and visualize results.

- **Steps**:
  - Convert the predicted age group to an estimated age range (e.g., “30-40 years old”).
  - Overlay the estimated age on the face image or display it alongside the video stream for real-time feedback.

CODE

CODE FOR "train_age_model.py" –

```
import pandas as pd
import cv2
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error
import joblib
import real_time_age_detection

data = pd.read_csv('data.csv')

def preprocess_image(image_path):
    image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    if image is None:
        print(f"Warning: Unable to load image at {image_path}")
        return None
    image_resized = cv2.resize(image, (64, 64))
    return image_resized.flatten()

image_paths = data['location'].values
X_images = []
valid_ages = []
valid_genders = []

for img_path, age, gender in zip(image_paths, data['age'].values, data['gender'].values):
    img = preprocess_image(img_path)
    if img is not None:
        X_images.append(img)
        valid_ages.append(age)
        valid_genders.append(1 if gender == 'Male' else 0)

X_images = np.array(X_images)
y = np.array(valid_ages)
gender = np.array(valid_genders)

X = np.hstack((X_images, gender.reshape(-1, 1)))

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

y_pred = np.maximum(y_pred, 0)

mae = mean_absolute_error(y_test, y_pred)
print(f'Mean Absolute Error: {mae:.2f}')

joblib.dump(model, 'age_detection_model.pkl')
real_time_age_detection()
```

CODE FOR "real_time_age_detection.py"—

```
import numpy as np
import cv2
import joblib

try:
    model = joblib.load('age_detection_model.pkl')
    print("Model loaded successfully.")
except Exception as e:
    print(f"Error loading model: {e}")
    exit()

cap = cv2.VideoCapture(0)

if not cap.isOpened():
    print("Error: Could not open webcam.")
    exit()

face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')

def preprocess_image(image):
    if image.size == 0:
        print("Warning: Empty image received.")
        return None
    image_resized = cv2.resize(image, (64, 64))
    return image_resized.flatten()

default_gender = 1

while True:
    ret, frame = cap.read()
    if not ret:
        break
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))

    for (x, y, w, h) in faces:
        face_roi = gray[y:y+h, x:x+w]
        face_preprocessed = preprocess_image(face_roi)

        if face_preprocessed is not None:
            X_features = np.hstack((face_preprocessed, default_gender)).reshape(1, -1)
            predicted_age = model.predict(X_features)[0]
            predicted_age = max(predicted_age, 0)
            cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)
            cv2.putText(frame, f'Age: {int(predicted_age)}', (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (255, 0, 0), 2)
    cv2.imshow('Age Detection', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
    if cv2.getWindowProperty('Age Detection', cv2.WND_PROP_VISIBLE) < 1:
        break
cap.release()
cv2.destroyAllWindows()
```

DATA SET – “data.csv”

	A	B	C
1	location	age	gender
2	D:\sml\1.jpg	3	male
3	D:\sml\2.jpg	7	male
4	D:\sml\3.jpg	10	male
5	D:\sml\4.jpg	9	male
6	D:\sml\5.jpg	10	male
7	D:\sml\6.jpg	2	male
8	D:\sml\7.jpg	0	male
9	D:\sml\8.jpg	6	male
10	D:\sml\9.jpg	7	male
11	D:\sml\10.jpg	10	male
12	D:\sml\11.jpg	16	male
13	D:\sml\12.jpg	20	male
14	D:\sml\13.jpg	20	male
15	D:\sml\14.jpg	12	male
16	D:\sml\15.jpg	18	male
17	D:\sml\16.jpg	17	male
18	D:\sml\17.jpg	11	male
19	D:\sml\18.jpg	14	male
20	D:\sml\19.jpg	11	male
21	D:\sml\20.jpg	19	male

RESULT AND ANALYSIS

Certainly! Let's analyze the provided Python program for age detection using a webcam and a pre-trained model:

1. **Model Loading**:
   - The program attempts to load a pre-trained age detection model from a file named 'age_detection_model.pkl'.
   - If successful, it prints a message indicating that the model was loaded. Otherwise, it handles the exception and exits.
2. **Webcam Initialization**:
   - The program initializes the webcam using cv2.VideoCapture(0).
   - If the webcam fails to open, it prints an error message and exits.
3. **Face Detection**:
   - It uses the Haar cascade classifier (haarcascade_frontalface_default.xml) to detect faces in the webcam stream.
   - For each detected face:
     - Extracts the face region of interest (ROI) from the grayscale frame.
     - Preprocesses the face image by resizing it to 64x64 pixels and flattening it.
     - Combines the preprocessed face features with a default gender value (1) to create the input feature vector.
     - Predicts the age using the pre-trained model.
     - Draws a rectangle around the detected face and displays the predicted age on the frame.
4. **Real-time Display**:
   - The program continuously captures frames from the webcam, processes them, and displays the results.

CONCLUSION

The conclusion is that, it displays a real-time age detection system on the user's webcam feed. Here's a summary of what happens when you run the program:

1. **Model Loading:** It attempts to load a pre-trained age detection model named `age_detection_model.pkl`. If successful, it starts the video capture.
2. **Webcam Access:** It opens the webcam (assuming webcam index 0) and checks if it's working. If not, it exits with an error message.
3. **Face Detection Loop:** The program enters a loop that continuously captures frames from the webcam. For each frame:
    - It converts the frame to grayscale for better face detection.
    - It uses a pre-trained Haar cascade classifier to find faces in the grayscale frame.
4. **Age Prediction on Detected Faces:**
    - For each detected face, it extracts the facial region (ROI) from the grayscale frame.
    - It preprocesses the extracted ROI by resizing it and flattening it into a 1D array.
    - It combines the preprocessed face data with a default gender value (set to 1 in this code).
    - This combined data is fed into the loaded model to predict the age.
    - The predicted age is enforced to be a minimum of 0 (to avoid negative values).
    - A bounding box and the predicted age are displayed on the original color frame.
5. **Display and Exit:**
    - The program displays the frame with faces and age predictions.
    - The loop exits when the user presses the 'q' key or closes the window.
6. **Cleanup:**
    - The webcam capture and any OpenCV windows are properly released.

Overall, this code provides a basic real-time age detection application. It's important to remember that the accuracy of the age prediction depends on the quality of the pre-trained model and factors like lighting and pose variations.

FUTURE WORK

While the age detection model using linear regression has shown promising results, there are several avenues forfuture exploration and improvement:

1. **Feature Engineering:** Explore additional features or combinations of features that may enhance the model's predictive power. For example, incorporating additional facial features or demographic variables could improve ageprediction accuracy.

2. **Model Refinement:** Experiment with more sophisticated regression techniques, such as polynomial regressionor regularization methods, to capture nonlinear relationships between features and age more effectively.

3. **Data Augmentation:** Augment the dataset with synthetic data or perform data balancing techniques to addressany class imbalance issues and improve the model's generalization performance.

4. **Cross-Validation Strategies:** Explore different cross-validation strategies to ensure robustness and reliability ofthe model's performance evaluation.

5. **Integration with Other Models:** Consider integrating the linear regression model with other machine learningtechniques, such as ensemble methods or deep learning models, to leverage their complementary strengths and improve overall performance.

6. **Real-world Deployment:** Deploy the trained model in real-world applications, such as age estimation in facialrecognition systems or age-based recommendation systems, and monitor its performance in practical settings.

REFERENCES

- CHAT GPT
- GOOGLE
- YOUTUBE
- STACK OVERFLOW
- GEEKS FOR GEEKS
- HANDS ON SUPERVISED LEARNING WITH PYTHON (BOOK)