PREDICTION OF HUMAN AGES

ASSIGNMENT

18CSE358T- Pattern Recognition Techniques

Submitted by

PRIYANSHU KUMAR JHA (RA2111003010528)

Under the guidance of

Jishnu K S

Assistant Professor, Computing Technologies

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE &

ENGINEERING SCHOOL OF

COMPUTING

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR - 603203

JULY 2024

INTRODUCTION

PREDICTION OF HUMAN AGES

Objective:

The primary objective of this project is to develop a sophisticated gender and age detection system that can estimate the gender and age of individuals based on facial images. This system can work with images from photos or live webcam feeds, providing real-time analysis.

About the Project:

This ambitious Python project leverages advanced deep learning techniques to accurately determine the gender and age of a person from a single image of their face. The project is built upon models trained by Tal Hassner and Gil Levi, which have been specifically designed for facial analysis. These pre-trained models are renowned for their efficacy in identifying human characteristics from facial features.

The gender detection aspect of the project classifies individuals into two categories: 'Male' or 'Female'. This binary classification is achieved through a neural network that has been trained on a diverse dataset, ensuring high accuracy across various facial profiles.

For age detection, the project adopts a classification approach rather than regression. This decision stems from the inherent challenges associated with predicting an exact age from a single image. Factors such as makeup, lighting conditions, obstructions like glasses or hats, and varying facial expressions can significantly influence the perceived age of an individual. By categorizing ages into specific ranges, the model can provide more reliable and interpretable results. The age ranges are: (0–2), (4–6), (8–12), (15–20), (25–32), (38–43), (48–53), and (60–100). These ranges are represented by eight nodes in the final softmax layer of the neural network, each corresponding to a distinct age group.

The project's deep learning model undergoes rigorous training on a large dataset, which includes images of individuals from various age groups and genders. This ensures that the model can generalize well and provide accurate predictions across different demographics. The training process involves fine-tuning the neural network's parameters to optimize its performance in gender and age classification tasks.

One of the key challenges addressed in this project is the variability in facial appearances due to external factors. For instance, makeup can significantly alter the perceived age of a person, while different lighting conditions can cast shadows or highlight certain features, affecting the model's predictions. To mitigate these issues, the project incorporates various data augmentation techniques during the training phase. These techniques help the model become more robust and adaptable to different scenarios.

This gender and age detection system has a wide range of potential applications. In security and surveillance, it can be used to monitor and identify individuals in real-time, enhancing safety measures. In marketing and demographic analysis, businesses can gain insights into their customer base and tailor their services accordingly. Additionally, the system can be integrated into personalized user experiences, such as age-appropriate content delivery or gender-specific product recommendations.

Overall, this project exemplifies the power of deep learning in the field of computer vision, showcasing how artificial intelligence can be harnessed to analyze and interpret human characteristics from facial images. By addressing the challenges associated with facial analysis and providing a practical solution, this gender and age detection system paves the way for future advancements in the interaction between humans and technology.

Dataset Description

For this Python project, I utilized the Adience dataset, a comprehensive and publicly available dataset that serves as a benchmark for face photos. The Adience dataset is inclusive of various real-world imaging conditions such as noise, varying lighting, diverse poses, and different appearances, making it ideal for training and testing robust facial analysis models. The images in this dataset have been collected from Flickr albums and are distributed under the Creative Commons (CC) license, ensuring ethical use and widespread accessibility.

The Adience dataset comprises a total of 26,580 photos featuring 2,284 distinct subjects. These images span eight age ranges: (0–2), (4–6), (8–12), (15–20), (25–32), (38–43), (48–53), and (60–100), which are consistent with the classification nodes in the final softmax layer of the neural network used in this project. With a total size of approximately 1GB, the dataset is substantial enough to train deep learning models effectively while remaining manageable for processing and storage.

The diverse and extensive nature of the Adience dataset, combined with its inclusion of real-world variations, provides a rich resource for developing accurate and generalizable models for gender and age detection. The models I implemented were trained on this dataset, leveraging its variety to enhance the robustness and accuracy of the predictions.

Additional Python Libraries Required:

To run this project, the following Python libraries are necessary:

OpenCV: OpenCV is an open-source computer vision and machine learning software library. It is used for real-time computer vision applications and provides tools for image and video processing. To install OpenCV, use the following command:

```
pip install opencv-python
```

argparse: argparse is a Python library for parsing command-line arguments. It allows the script to accept parameters and options from the command line, enabling flexible configuration and execution. To install argparse, use the following command:

```
pip install argparse
```

These libraries are integral to the project's functionality, providing essential tools for image processing and command-line interface management, thereby facilitating the implementation and execution of the gender and age detection system.

EXPLANATION

This project implements a gender and age detection system using deep learning techniques. The primary goal is to identify the gender and approximate age range of a person from a single image of their face. The system can work with both static images and live webcam feeds.

Project Workflow:

The provided Python code is a comprehensive implementation of a gender and age detection system using OpenCV and pre-trained deep learning models. Below is a detailed explanation of the code:

Importing Libraries

```
import cv2
import math
import argparse
```

The code begins by importing necessary libraries:

- **`cv2` for computer vision tasks using OpenCV.**
- **`math` for mathematical operations.**
- **`argparse` for parsing command-line arguments.**

Function Definition: `highlightFace`:

```
def highlightFace(net, frame, conf_threshold=0.7):

    frameOpencvDnn = frame.copy()
```

```
    frameHeight = frameOpencvDnn.shape[0]
    frameWidth = frameOpencvDnn.shape[1]
    blob = cv2.dnn.blobFromImage(frameOpencvDnn, 1.0, (300, 300), [104, 117, 123], True,
False)

    net.setInput(blob)
    detections = net.forward()
    faceBoxes = []
    for i in range(detections.shape[2]):
        confidence = detections[0, 0, i, 2]
        if confidence > conf_threshold:
            x1 = int(detections[0, 0, i, 3] * frameWidth)
            y1 = int(detections[0, 0, i, 4] * frameHeight)
            x2 = int(detections[0, 0, i, 5] * frameWidth)
            y2 = int(detections[0, 0, i, 6] * frameHeight)
            faceBoxes.append([x1, y1, x2, y2])
            cv2.rectangle(frameOpencvDnn, (x1, y1), (x2, y2), (0, 255, 0), int(round(frameHeight /
150)), 8)
    return frameOpencvDnn, faceBoxes
```

1.
    a. Purpose: This function detects faces in an input frame using a pre-trained neural network model (`net`). It returns the frame with detected faces highlighted and the coordinates of the face bounding boxes.
    b. Parameters:
        i. `net`: The neural network model for face detection.
        ii. `frame`: The input image/frame in which faces need to be detected.
        iii. `conf_threshold`: The confidence threshold to filter weak detections.
    c. Process:
        i. A copy of the frame is created to avoid modifying the original frame.
        ii. The frame's dimensions (height and width) are obtained.
        iii. A blob is created from the image, which is a pre-processing step for deep learning models.
        iv. The blob is set as the input to the face detection model, and forward propagation is performed to get detections.
        v. Each detection is filtered based on the confidence threshold. If a detection passes the threshold, the coordinates of the bounding box are calculated and stored.
        vi. A rectangle is drawn around each detected face on the frame copy.
        vii. The function returns the annotated frame and the list of face bounding boxes.

```
parser = argparse.ArgumentParser()
parser.add_argument('--image')
```

```
args = parser.parse_args()
```

2.
    a. Purpose: To enable the script to accept an optional image file path from the command line. If no image path is provided, the script defaults to using a webcam feed.
    b. `--image`: Optional argument to specify the path to an image file.

Model File Paths:

```
faceProto = "opencv_face_detector.pbtxt"

faceModel = "opencv_face_detector_uint8.pb"
ageProto = "age_deploy.prototxt"
ageModel = "age_net.caffemodel"
genderProto = "gender_deploy.prototxt"
genderModel = "gender_net.caffemodel"
```

3.
    a. Purpose: Define the file paths to the model configuration files and pre-trained weights for face detection, gender classification, and age estimation.

Model Mean Values and Class Labels:

```
MODEL_MEAN_VALUES = (78.4263377603, 87.7689143744, 114.895847746)

ageList = ['(0-2)', '(4-6)', '(8-12)', '(15-20)', '(25-32)', '(38-43)', '(48-53)', '(60-100)']
genderList = ['Male', 'Female']
```

4.
    a. MODEL_MEAN_VALUES: These are the mean values for the channels (R, G, B) used for normalizing the input images before passing them to the deep learning model.
    b. ageList: A list of age ranges used by the age estimation model.
    c. genderList: A list of gender labels used by the gender classification model.

Loading Pre-trained Models:

python

Copy code

```
faceNet = cv2.dnn.readNet(faceModel, faceProto)

ageNet = cv2.dnn.readNet(ageModel, ageProto)
genderNet = cv2.dnn.readNet(genderModel, genderProto)
```

5.
    a. Purpose: Load the pre-trained models for face detection, gender classification, and age estimation using OpenCV's `readNet` function.

Video Capture:

```
video = cv2.VideoCapture(args.image if args.image else 0)
```

6.

a. Purpose: Initialize the video capture object. If an image path is provided via the command line, it reads that image. Otherwise, it captures frames from the default webcam.

Main Loop:

```
padding = 20

while cv2.waitKey(1) < 0:
    hasFrame, frame = video.read()
    if not hasFrame:
        cv2.waitKey()
        break

    resultImg, faceBoxes = highlightFace(faceNet, frame)
    if not faceBoxes:
        print("No face detected")

    for faceBox in faceBoxes:
        face = frame[max(0, faceBox[1] - padding): min(faceBox[3] + padding, frame.shape[0] - 1),
                    max(0, faceBox[0] - padding): min(faceBox[2] + padding, frame.shape[1] - 1)]

        blob = cv2.dnn.blobFromImage(face, 1.0, (227, 227), MODEL_MEAN_VALUES,
swapRB=False)
        genderNet.setInput(blob)
        genderPreds = genderNet.forward()
        gender = genderList[genderPreds[0].argmax()]
        print(f'Gender: {gender}')

        ageNet.setInput(blob)
        agePreds = ageNet.forward()
        age = ageList[agePreds[0].argmax()]

        print(f'Age: {age[1:-1]} years')

        cv2.putText(resultImg, f'{gender}, {age}', (faceBox[0], faceBox[1] - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 255), 2, cv2.LINE_AA)

                cv2.imshow("Detecting age and gender", resultImg)
```

7.
    a. Padding: A padding value of 20 pixels is added around the detected face to ensure the face region is fully captured for gender and age prediction.
    b. Loop: Continuously captures frames from the video source.
        i. Frame Capture: Reads a frame from the video source.
        ii. Face Detection: Calls `highlightFace` to detect faces in the current frame.
        iii. Face Boxes: If no faces are detected, it prints "No face detected".
        iv. Gender and Age Prediction:
            1. For each detected face, a region of interest (ROI) is extracted with added padding.
            2. A blob is created from the ROI.
            3. The blob is passed through the gender detection model to predict the gender.
            4. The blob is then passed through the age detection model to predict the age range.
            5. The predicted gender and age range are printed to the console and annotated on the frame.
        v. Display: The annotated frame is displayed in a window titled "Detecting age and gender".

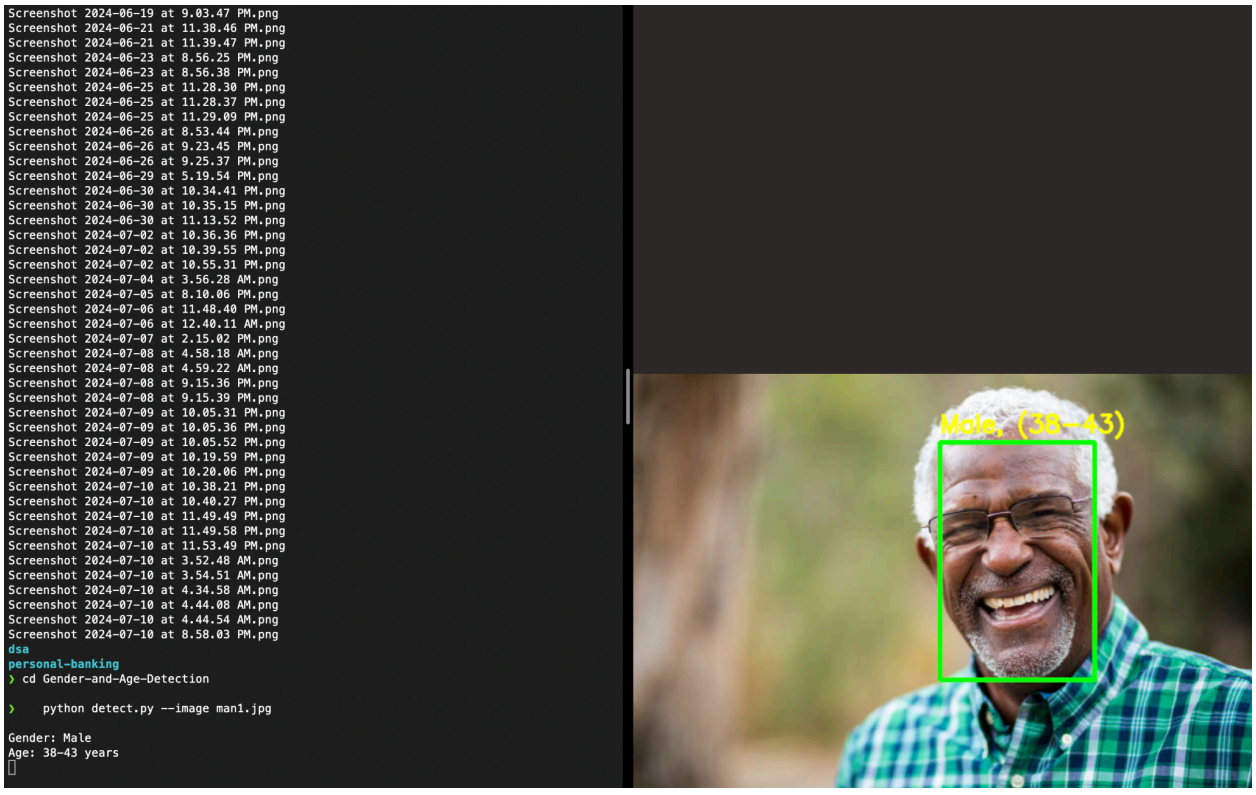
Code:

```
import cv2
import math
import argparse

def highlightFace(net, frame, conf_threshold=0.7):
    frameOpencvDnn = frame.copy()
    frameHeight = frameOpencvDnn.shape[0]
    frameWidth = frameOpencvDnn.shape[1]
    blob = cv2.dnn.blobFromImage(frameOpencvDnn, 1.0, (300, 300), [104, 117, 123], True,
False)

    net.setInput(blob)
    detections = net.forward()
    faceBoxes = []
    for i in range(detections.shape[2]):
        confidence = detections[0, 0, i, 2]
        if confidence > conf_threshold:
            x1 = int(detections[0, 0, i, 3] * frameWidth)
            y1 = int(detections[0, 0, i, 4] * frameHeight)
```

```
            x2 = int(detections[0, 0, i, 5] * frameWidth)
            y2 = int(detections[0, 0, i, 6] * frameHeight)
            faceBoxes.append([x1, y1, x2, y2])
            cv2.rectangle(frameOpencvDnn, (x1, y1), (x2, y2), (0, 255, 0), int(round(frameHeight /
150)), 8)
    return frameOpencvDnn, faceBoxes

parser = argparse.ArgumentParser()
parser.add_argument('--image')

args = parser.parse_args()

faceProto = "opencv_face_detector.pbtxt"
faceModel = "opencv_face_detector_uint8.pb"
ageProto = "age_deploy.prototxt"
ageModel = "age_net.caffemodel"
genderProto = "gender_deploy.prototxt"
genderModel = "gender_net.caffemodel"

MODEL_MEAN_VALUES = (78.4263377603, 87.7689143744, 114.895847746)
ageList = ['(0-2)', '(4-6)', '(8-12)', '(15-20)', '(25-32)', '(38-43)', '(48-53)', '(60-100)']
genderList = ['Male', 'Female']

faceNet = cv2.dnn.readNet(faceModel, faceProto)
ageNet = cv2.dnn.readNet(ageModel, ageProto)
genderNet = cv2.dnn.readNet(genderModel, genderProto)

video = cv2.VideoCapture(args.image if args.image else 0)
padding = 20
while cv2.waitKey(1) < 0:
    hasFrame, frame = video.read()
    if not hasFrame:
        cv2.waitKey()
        break

    resultImg, faceBoxes = highlightFace(faceNet, frame)
    if not faceBoxes:
        print("No face detected")

    for faceBox in faceBoxes:
        face = frame[max(0, faceBox[1] - padding):
                     min(faceBox[3] + padding, frame.shape[0] - 1), max(0, faceBox[0] - padding)
                     :min(faceBox[2] + padding, frame.shape[1] - 1)]
```

```
        blob = cv2.dnn.blobFromImage(face, 1.0, (227, 227), MODEL_MEAN_VALUES,
swapRB=False)
        genderNet.setInput(blob)
        genderPreds = genderNet.forward()
        gender = genderList[genderPreds[0].argmax()]
        print(f'Gender: {gender}')

        ageNet.setInput(blob)
        agePreds = ageNet.forward()
        age = ageList[agePreds[0].argmax()]
        print(f'Age: {age[1:-1]} years')

        cv2.putText(resultImg, f'{gender}, {age}', (faceBox[0], faceBox[1] - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 255), 2, cv2.LINE_AA)
        cv2.imshow("Detecting age and gender", resultImg)
```

OUTPUT




Detecting age and gender
Female, (38–43)