CASE STUDY DEEP LEARNING

Project Title:

Gender and Age Detection on UTKFace Dataset

Student Name:

Bhakti Shinde

College Name:

Anantrao Pawar College of Engineering and Research , Pune

Project Title: Gender and Age Detection on UTKFace Dataset

Objective: To develop a deep learning model capable of identifying the gender and estimating the age of individuals from their facial images using the UTKFace dataset.

Hardware Requirements:

Multi-core CPU or a CPU with hyper-threading capability.

Optional but recommended GPU for accelerated training.

Adequate RAM to support the dataset and model operations.

Software Requirements:

Python programming language.

Keras for constructing and training deep learning models.

Jupyter Notebook or any Python IDE for coding and testing.

Theory:

In the context of gender and age detection, deep learning theory forms the foundation for developing accurate and efficient models that can automatically learn meaningful representations from raw data (such as facial images) without the need for manual feature engineering. Here's a breakdown of key concepts in deep learning theory relevant to this project:

1. Neural Networks: Deep learning relies on artificial neural networks (ANNs), which are computational models inspired by the biological neural networks in the human brain. ANNs consist of interconnected layers of artificial neurons (nodes), each performing simple mathematical operations on input data and passing the result to the next layer.
2. Convolutional Neural Networks (CNNs): CNNs are a type of neural network particularly well-suited for processing grid-like data, such as images. They are composed of convolutional layers, pooling layers, and fully connected layers. Convolutional layers apply convolution operations to input images, extracting spatial hierarchies of features through learned filters. Pooling layers downsample feature maps, reducing computational complexity and promoting translation invariance. Fully connected layers combine extracted features to make predictions.
3. Transfer Learning: Transfer learning is a technique where pre-trained models developed for one task are reused as the starting point for a new task. In gender and age detection, transfer learning allows leveraging pre-trained CNN models (trained on large datasets

like ImageNet) to extract generic facial features, which can then be fine-tuned for the specific tasks of gender classification and age estimation.

4. Activation Functions: Activation functions introduce non-linearity into neural networks, enabling them to learn complex mappings between inputs and outputs. Common activation functions include ReLU (Rectified Linear Unit), sigmoid, and softmax. ReLU is typically used in hidden layers of CNNs for its computational efficiency and ability to mitigate the vanishing gradient problem.
5. Backpropagation: Backpropagation is a key algorithm used to train neural networks by iteratively adjusting model parameters to minimize a specified loss function. It involves computing gradients of the loss function with respect to each parameter using the chain rule of calculus and updating parameters in the direction that reduces the loss.
6. Optimization Algorithms: Optimization algorithms determine how model parameters are updated during training to minimize the loss function. Common optimization algorithms include stochastic gradient descent (SGD), Adam, RMSprop, and AdaGrad.
7. Loss Functions: Loss functions quantify the difference between predicted outputs and ground truth labels, providing feedback signals used during training. In classification tasks like gender detection, categorical cross-entropy loss is often used. In regression tasks like age estimation, mean squared error (MSE) or mean absolute error (MAE) are commonly employed.

Methodology:

1. Dataset Collection and Preparation: Utilize the UTKFace dataset, which includes facial images accompanied by gender and age labels. Preprocess the images to a uniform size, shuffle the dataset, and cap ages at 100 years for consistency.
2. Dataset Splitting: Divide the dataset into training, validation, and testing sets with a typical ratio of 80%, 10%, and 10%, respectively.
3. Data Generators: Implement data generators for the training, validation, and testing sets using Keras's ImageDataGenerator class. Employ data augmentation techniques like rotation, zoom, and horizontal flipping to enhance model robustness.
4. Model Architecture: Design a convolutional neural network (CNN) that inputs face images and outputs two values: the probability of the subject being male and their predicted age. Incorporate multiple convolutional and pooling layers, followed by dense layers.
5. Model Compilation: Compile the model with binary cross-entropy loss for the gender classification and mean squared error (MSE) for age prediction. Select appropriate metrics such as accuracy for gender and mean absolute error (MAE) for age.
6. Model Training: Train the model using the fit method, providing the data generators for training and validation sets, specifying the number of epochs and batch size.
7. Model Evaluation: Assess the model on the testing set using the evaluate method to determine accuracy and MAE.

8. Prediction: To predict gender and age for a new sample image, use the OpenCV library (cv2) to load and preprocess the image (resize and normalize). Employ the model's predict method to obtain predictions.

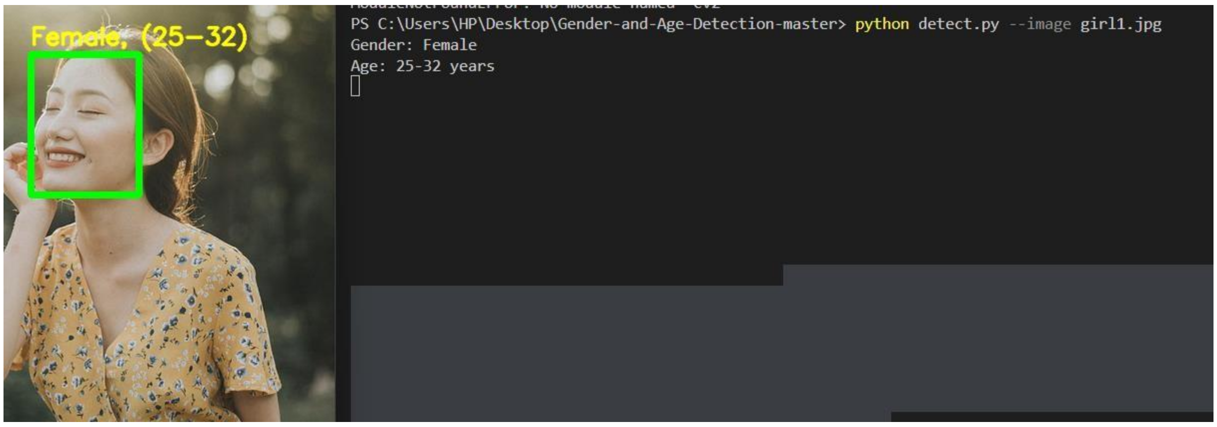
Code:

Detect.py

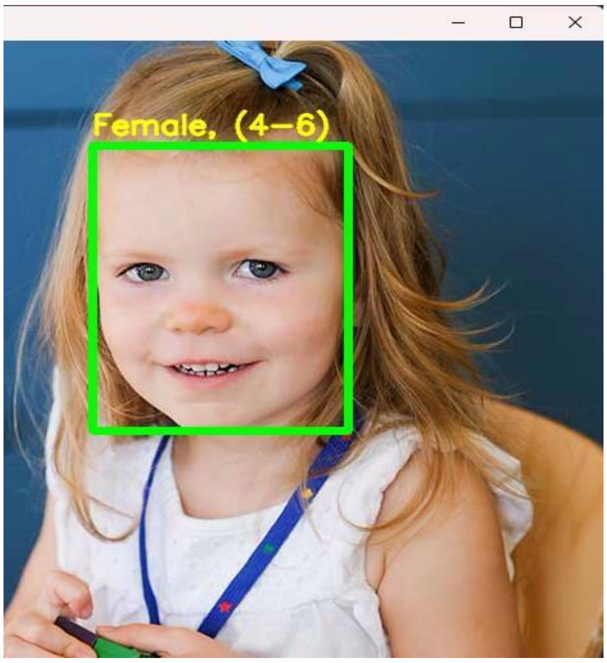
```
#A Gender and Age Detection program by Mahesh Sawant

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"
```

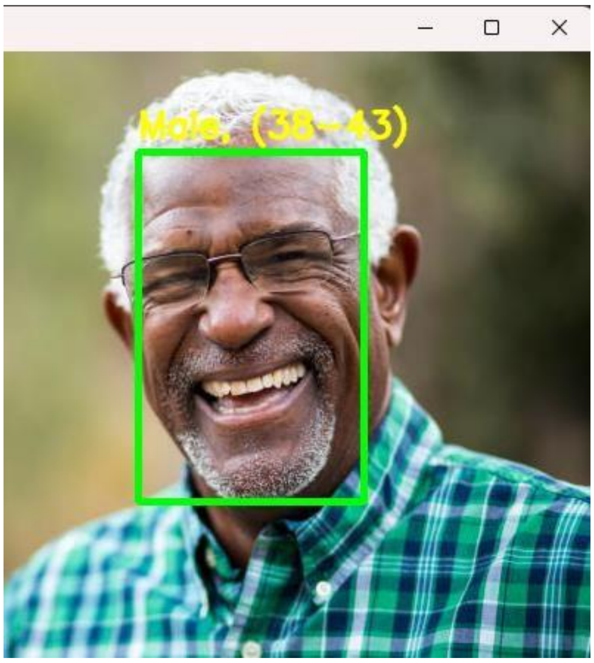
```
MODEL_MEAN_VALUES=(78.4263377603, 87.7689143744, 114.895847746)
ageList=['(0-2)', '(4-6)', '(8-12)', '(15-20)', '(25-32)', '(38-43)', '(48-
53)', '(60-100)']
genderList=['Male','Female']

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:



```
PS C:\Users\HP\Desktop\Gender-and-Age-Detection-master> python detect.py --image girl1.jpg
Gender: Female
Age: 25-32 years
```





Conclusion:

In this way we have implemented model capable of identifying the gender and estimating the age of individuals from their facial images using the UTKFace dataset.