Rafik Hariri University
جامعة رفيق الحريري

Course:

Special Topics in Computer Science

Project Title:

Real Time Gender and Age Detection

Presented by:

Maya Bsat

Hamzah Jomaa

Daniel Al Hassanieh

Presented to:

Dr. Roaa SOLOH

Date: April 18, 2022

Table of Contents

List of Acronyms

AI – Artificial Intelligence

BLOB - Binary Large Object

CNN - Convolutional Neural Networks

DNN - Deep Neural Network

OpenCV - Open Source Computer Vision’s

ReLU - Rectified Linear Units

FPS – Frames per Second

I. Introduction

Human Classification is an age-old and gender procedure are being done in various fields such as biometrics, forensics sciences, Image processing, Identification system, etc. With the development of Artificial Intelligence and techniques such as Neural Network and Deep Learning, it has become increasingly easier to classify human into certain categories. These new technologies help identification, classification of Individuals without the need of another professional or Individual records. Age and gender detection can be useful for several use cases such as: Identification of the target audience in marketing organization, in Recruitment procedure, to verify legitimacy of the applicants, verification of authentic person applying for government IDs, and Classification of human resources in bulk.

There have been multiple approaches that focus their work on achieving a high accuracy rate for these kinds of predictions and classifications. Latest approaches commonly adapt Convolutional Neural Networks (CNN), although some limitations to this kind of networks reduce the progress of such models. One of the limitations of such systems is the large amount of data needed when training the model. The model's accuracy rates depend not only on the implemented network but also on the data that is used to train it. Since the training data is what allows the system to learn to identify and predict outcomes correctly, we need more data to feed the network to get better results. One of the problems faced when it comes to face detection or age/gender classification are partial occlusions and low-quality images. Those influence directly the outcome results as the model has less information to work on, which makes it harder to predict accurate results. The same goes when human makes a prediction. If the image has low quality, it is harder for a human to understand what is being seen, and therefore, to make a prediction.

With this in mind, this project has the intention of creating a system which can efficiently detect people in images, identifying what their gender and age are. One way to attain this is to use age and gender classification methods. This can be achieved using neural networks, provided that these are able to produce high confidence predictions, i.e., relevant and trustworthy. Therefore, this project falls under the area of artificial intelligence (AI) since it uses Deep Learning models in order to extract information from images and make accurate predictions using those.

II. Research and Methodology

The Project uses Python Deep Learning to identify the gender and age of given face data accurately. Deep learning is a subset of machine learning, which is essentially a neural network with three or more layers. These networks mimic the functionality of human cognitive thinking and acts as an Artificial Intelligence system. It is able to recognize and detect objects, faces, speeches, characters from unstructured data sets.

The Algorithm designed for this project is divided into four main parts:

1) Input
2) Face Detection
3) Face Processing (Age and Gender classification)
4) Output

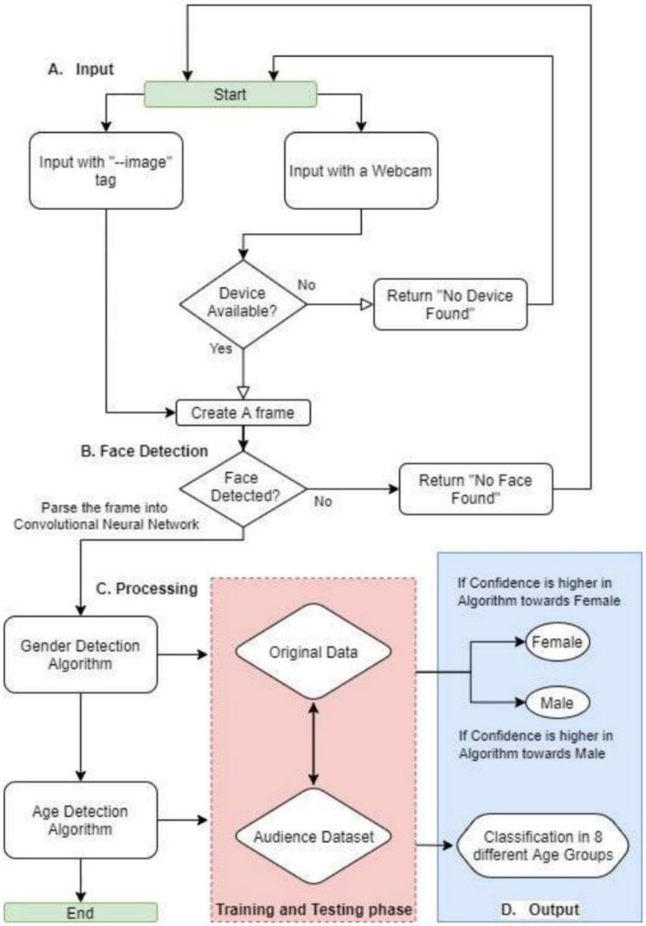


Fig. 1 Flowchart of the Algorithm.

Required Module

OpenCV: It is a tool that specializes in the areas of image processing, video analysis, or computer vision. It can be used to solve lots of problems analyzing images and videos through sophisticated digital algorithms. After taking images or videos as input data, the module runs filters that convert them into Boolean values so features can be recognized through comparison functions given that they share similar traits.

Why choose OpenCv Module?

One of the OpenCV DNN module’s best features is that it is highly optimized for Intel processors. We can get good FPS when running inference on real-time videos for object detection and image segmentation applications. We often get higher FPS (frames per second**)** with the DNN module when using a model pre-trained. For instance, we can see in the below image that OpenCV is a lot faster than Tensor Flow’s original implementations while falling behind PyTorch by a little. In fact, Tensor Flow’s inference time is close to 1 seconds whereas OpenCV takes less than 200 milliseconds

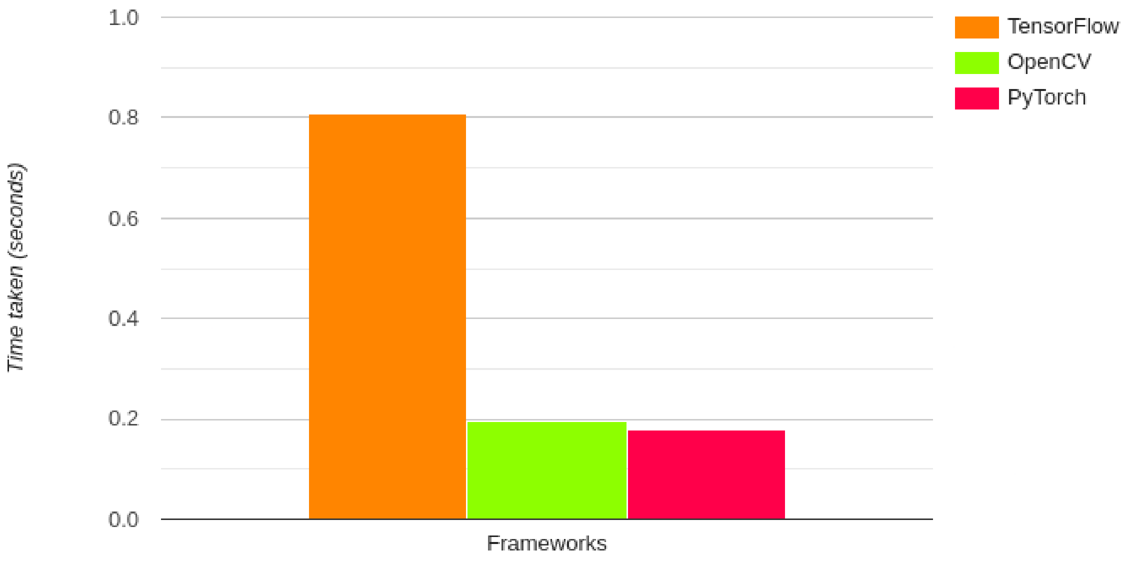


Image Classification Time Comparisons (Average of three images)

Dataset:

For this python project, we’ll use the Adience dataset.This dataset serves as a benchmark for face photos and is inclusive of various real-world imaging conditions like noise, lighting, pose, and appearance. It has a total of 26,580 photos of 2,284 subjects in eight age ranges (as mentioned below).The models we will use have been trained on this dataset.

To detect the age and gender of a face using OpenCV, there is a need of a pre-trained model or set of images containing a large number of male and female faces of different ages. In this project, we will use some simple models.

1. Age_deploy.prototxt
2. Age_net.caffemodel
3. Gender_deploy.prototxt
4. Gender_net.caffemodel
5. Opencv_face_detector.pbtxt
6. Opencv_face_detector_uint8.pb

For face detection, we have a .pb file- this is a protobuf file (protocol buffer); it holds the graph definition and the trained weights of the model. We can use this to run the trained model. And while a .pb file holds the protobuf in **binary format**, one with the .pbtxt extension holds it in **text format**. These are TensorFlow files. For age and gender, the .prototxt files describe the network configuration and the .caffemodel file defines the internal states of the parameters of the layers.

In this project, we proposed a simplistic architecture that learns a total of eight age brackets:

1) 0-2
2) 4-6
3) 8-12
4) 15-20
5) 25-32
6) 38-43
7) 48-53
8) 60-100

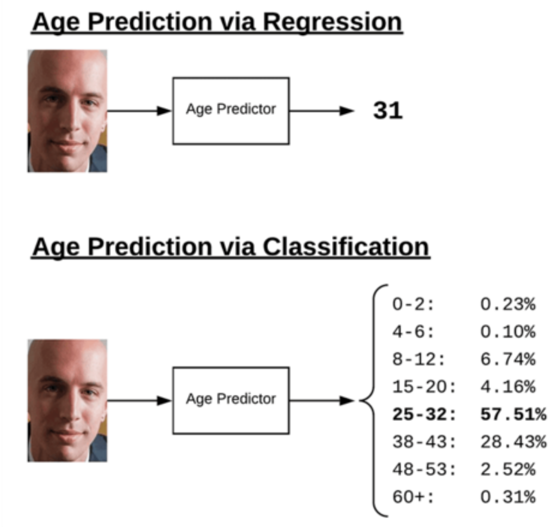


Figure 1. Age prediction with deep learning can be framed as a regression or classification problem

Age prediction framed as classification problem

The reason for choosing to work on age prediction as a classification model is that age prediction is inherently subjective and based solely on appearance. A person in their mid-50s who has never smoked in their life, works out, always wore sunscreen, and took care of their skin daily will look younger than someone in their late-30s who smokes a carton a day, works manual labor without sun protection, has bad eating habits and doesn't have a proper skin care routine. Another important driving factor in aging we should take into consideration is *genetics* — some people simply age better than others.

III. Design of the Model

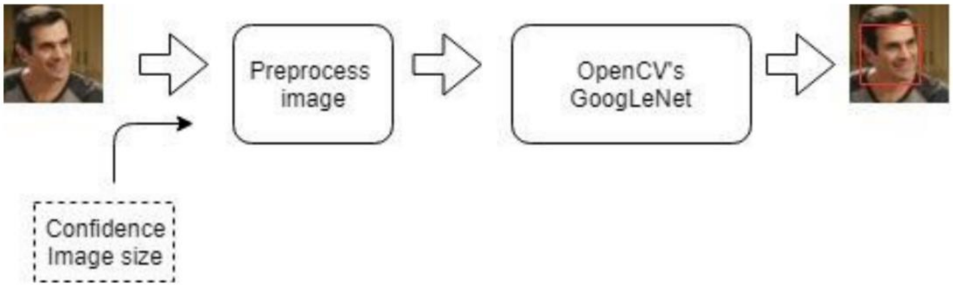


Figure 2. Diagram of the face model detection

As a first step, the image is preprocessed and resized to specific image size and converted to a specific object (a large binary object (**BLOB**)) in order to pass it to the **DNN**. This image size is configurable by the user but has a default value of 200x200. The confidence with which the network detects the faces has a default threshold value of 0.5. This means that only detections with a probability higher than 0.5 are considered real faces; however, we can set our own threshold value. Therefore, in our model we chose the threshold = 0.75. Finally, after the image is preprocessed, it is passed to the network which outputs all detectable faces with coordinates that indicate 4 points that are used to crop the face, and another output which is one value between 0and 1 which shows the confidence that such detection is, in fact, a face.

IV. Algorithm: Deep convolutional neural networks

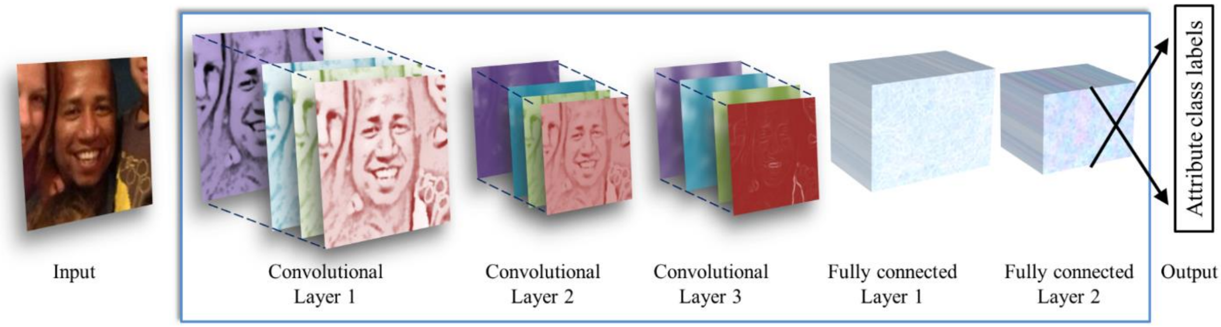


Figure 3. Illustration of the CNN architecture

The network contains 3 convolutional layers, each followed by a rectified linear operation and pooling layer. The first two layers also follow normalization using local response normalization. The first Convolutional Layer contains 96 filters of 7×7 pixels, the second Convolutional Layer contains 256 filters of 5×5 pixels, the third and final Convolutional Layer contains 384 filters of 3 × 3 pixels. Finally, 2 fully-connected layers are added, each containing 512 neurons

Full schema of network architecture

Our proposed network architecture is used throughout our experiments for both age and gender classification. The network comprises of only three convolutional layers and two fully-connected layers with a small number of neurons. Our choice of a smaller network design is motivated both from our desire to reduce the risk of overfitting.

All three color channels (BGR)are processed directly by the network. Images are first rescaled to 256 × 256 and a crop of 227 × 227 is fed to the network. The three subsequent convolutional layers are then defined as follows:

1) 96 filters of size 3×7×7 pixels are applied to the input in the first convolutional layer, followed by a rectified linear operator (ReLU), a max pooling layer taking the maximal value of 3 × 3 regions with two-pixel strides and a local response normalization layer.
2) The 96 × 28 × 28 output of the previous layer is then processed by the 2nd convolutional layer, containing 256 filters of size 96 × 5 × 5 pixels. Again, this is followed by ReLU, a max pooling layer and a local response normalization layer with the same hyper parameters as before.
3) Finally, the 3rd and last convolutional layer operates on the 256 × 14 × 14 blob by applying a set of 384 filters of size 256 × 3 × 3 pixels, followed by ReLU and a max pooling layer.

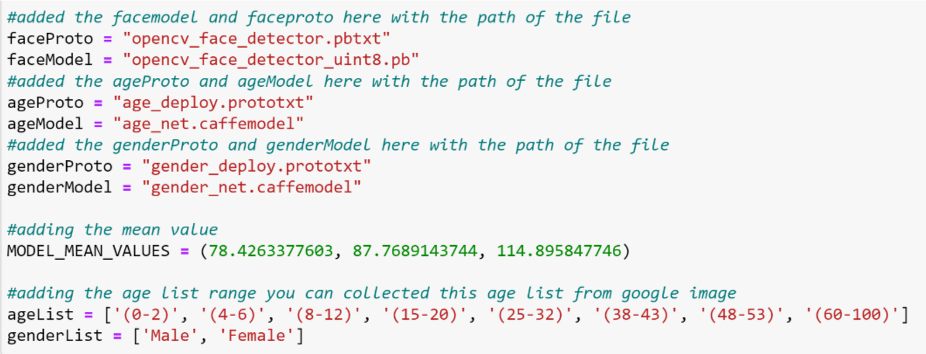
The following fully connected layers are then defined by:

4) A first fully connected layer that receives the output of the 3rd convolutional layer and contains 512 neurons, followed by a ReLU and a dropout layer.
5) A second fully connected layer that receives the 512- dimensional output of the first fully connected layer and again contains 512 neurons, followed by a ReLU and a dropout layer.
6) A third, fully connected layer which maps to the final classes for age or gender.

Finally, the output of the last fully connected layer is fed to a soft-max layer that assigns a probability for each class. The prediction itself is made by taking the class with the maximal probability for the given test image.

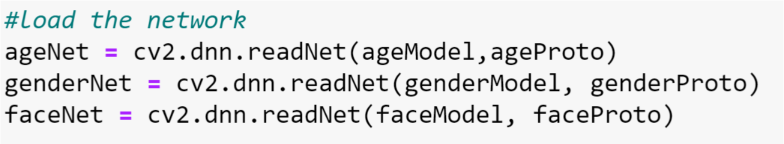
V. Code Documentation

1) Import Libraries
2) For face, age, and gender, initialize protocol buffer and model.
3) Initialize the mean values for the model and the lists of age ranges and genders to classify from.



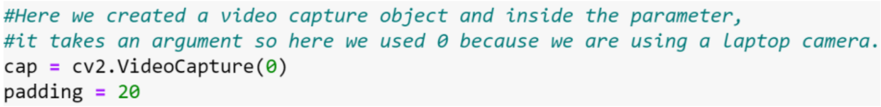
```
#added the facemodel and faceproto here with the path of the file
faceProto = "opencv_face_detector.pbtxt"
faceModel = "opencv_face_detector_uint8.pb"
#added the ageProto and ageModel here with the path of the file
ageProto = "age_deploy.prototxt"
ageModel = "age_net.caffemodel"
#added the genderProto and genderModel here with the path of the file
genderProto = "gender_deploy.prototxt"
genderModel = "gender_net.caffemodel"

#adding the mean value
MODEL_MEAN_VALUES = (78.4263377603, 87.7689143744, 114.895847746)

#adding the age list range you can collected this age list from google image
ageList = ['(0-2)', '(4-6)', '(8-12)', '(15-20)', '(25-32)', '(38-43)', '(48-53)', '(60-100)']
genderList = ['Male', 'Female']
```

4) Now, use the readNet() method to load the networks. The first parameter holds trained weights and the second carries network configuration.



```
#load the network
ageNet = cv2.dnn.readNet(ageModel,ageProto)
genderNet = cv2.dnn.readNet(genderModel, genderProto)
faceNet = cv2.dnn.readNet(faceModel, faceProto)
```

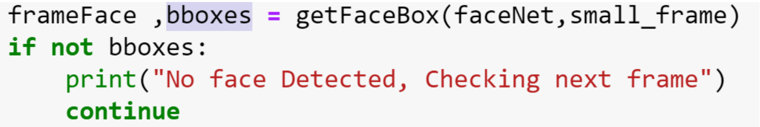
5) Let’s capture video stream in case you’d like to classify on a webcam’s stream. Set padding to 20.



```
#Here we created a video capture object and inside the parameter,
#it takes an argument so here we used 0 because we are using a laptop camera.
cap = cv2.VideoCapture(0)
padding = 20
```

6) Now until any key is pressed, we read the stream and store the content into the names hasFrame and frame. If it isn’t a video, it must wait, and so we call up waitKey() from cv2, then break.

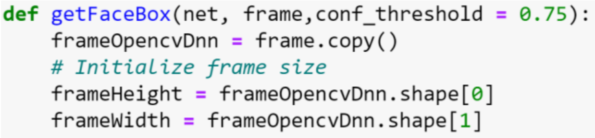
7) Let’s make a call to the getFaceBox () function with the *faceNe*t and frame parameters, and what this returns, we will store in the names *frameFace* and *bboxes*. And if we got 0 *bboxes (face boxes)*, it means no face detected. Here, net is faceNet- this model is the DNN Face Detector and holds only about 2.7MB on disk.



```
frameFace ,bboxes = getFaceBox(faceNet,small_frame)
if not bboxes:
    print("No face Detected, Checking next frame")
    continue
```

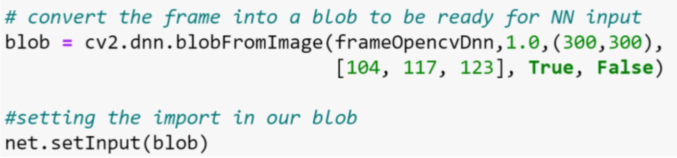
8) getFaceBox function

- Create a shallow copy of frame and get its height and width.



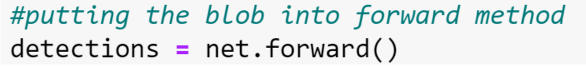
```python
def getFaceBox(net, frame,conf_threshold = 0.75):
    frameOpencvDnn = frame.copy()
    # Initialize frame size
    frameHeight = frameOpencvDnn.shape[0]
    frameWidth = frameOpencvDnn.shape[1]
```

- Create a blob from the shallow copy.



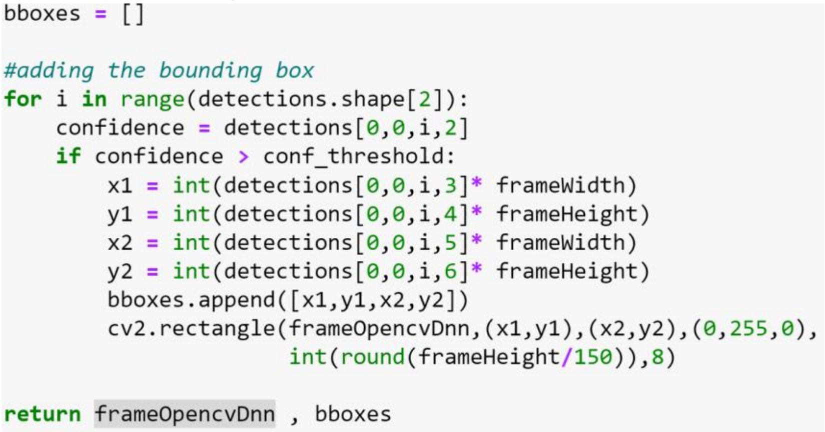
```python
# convert the frame into a blob to be ready for NN input
blob = cv2.dnn.blobFromImage(frameOpencvDnn,1.0,(300,300),
                             [104, 117, 123], True, False)

#setting the import in our blob
net.setInput(blob)
```

- Set the input and make a forward pass to the network.



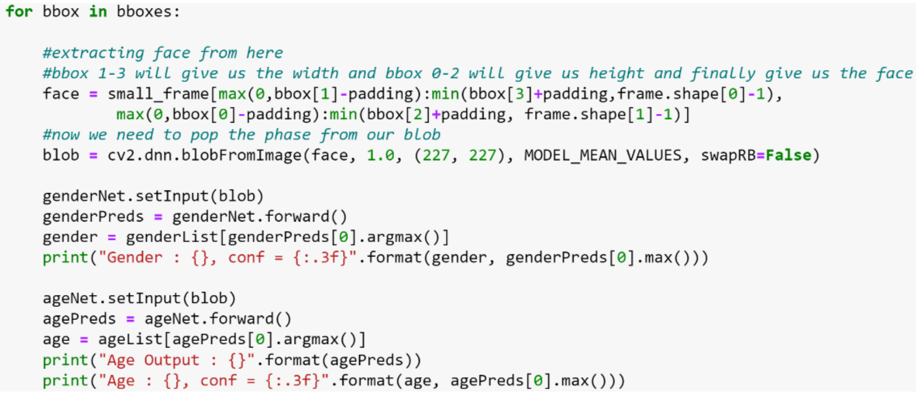
```python
#putting the blob into forward method
detections = net.forward()
```

- bboxes is an empty list now. for each value in 0 to 127, define the confidence (between 0 and 1). Wherever we find the confidence greater than the confidence threshold, which is 0.75, we get the x1, y1, x2, and y2 coordinates and append a list of those to bboxes.
- Then, we put up rectangles on the image for each such list of coordinates and return two things: the shallow copy and the list of bboxes.



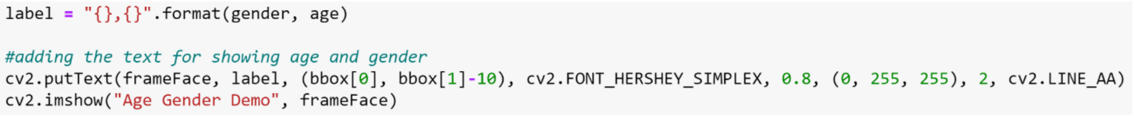
```python
bboxes = []

#adding the bounding box
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)
        bboxes.append([x1,y1,x2,y2])
        cv2.rectangle(frameOpencvDnn,(x1,y1),(x2,y2),(0,255,0),
                      int(round(frameHeight/150)),8)

return frameOpencvDnn , bboxes
```

- But if there are indeed bboxes, for each of those, we detect the face, create a 4-dimensional blob from the image. In doing this, we scale it, resize it, and pass in the mean values.
- We feed the input and give the network a forward pass to get the confidence of the two class. Whichever is higher, that is the gender of the person in the picture.



```
for bbox in bboxes:

    #extracting face from here
    #bbox 1-3 will give us the width and bbox 0-2 will give us height and finally give us the face
    face = small_frame[max(0,bbox[1]-padding):min(bbox[3]+padding,frame.shape[0]-1),
            max(0,bbox[0]-padding):min(bbox[2]+padding, frame.shape[1]-1)]
    #now we need to pop the phase from our blob
    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("Gender : {}, conf = {:.3f}".format(gender, genderPreds[0].max()))

    ageNet.setInput(blob)
    agePreds = ageNet.forward()
    age = ageList[agePreds[0].argmax()]
    print("Age Output : {}".format(agePreds))
    print("Age : {}, conf = {:.3f}".format(age, agePreds[0].max()))
```

- We'll add the gender and age texts to the resulting image and display it with imshow().

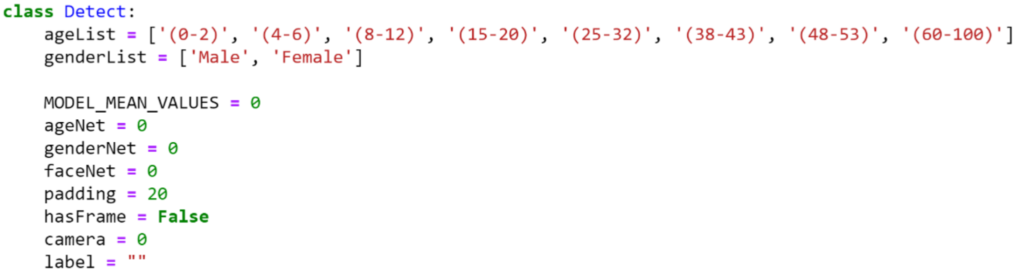


```
label = "{},{}".format(gender, age)

#adding the text for showing age and gender
cv2.putText(frameFace, label, (bbox[0], bbox[1]-10), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 255), 2, cv2.LINE_AA)
cv2.imshow("Age Gender Demo", frameFace)
```

Optimization of the Code

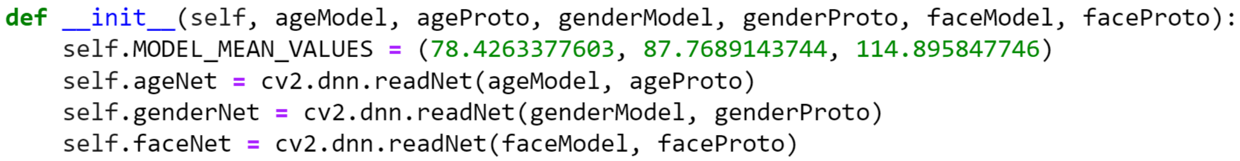
In order for the code to be more structured and optimized, we used the object oriented programming methods and used classes and objects as follows:

1) Declare a class "Detect" where we initialize the variables



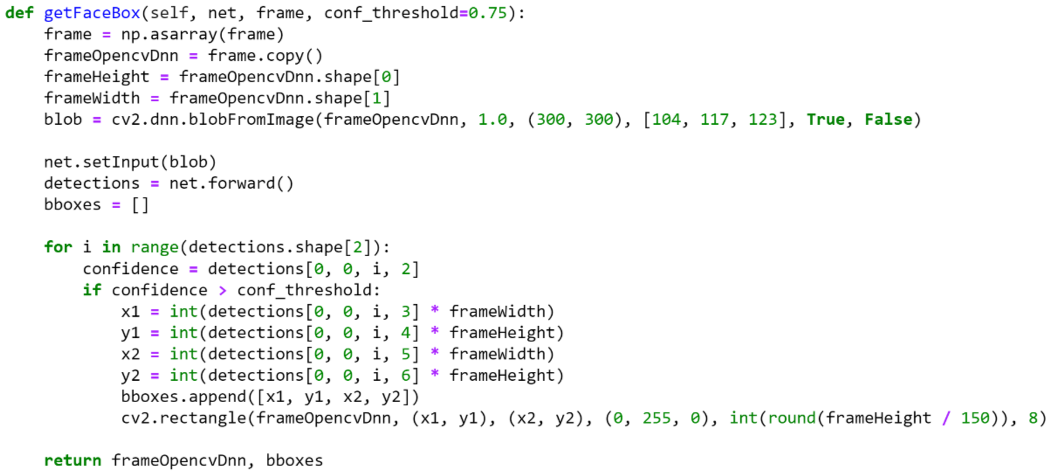
```
class Detect:
    ageList = ['(0-2)', '(4-6)', '(8-12)', '(15-20)', '(25-32)', '(38-43)', '(48-53)', '(60-100)']
    genderList = ['Male', 'Female']

    MODEL_MEAN_VALUES = 0
    ageNet = 0
    genderNet = 0
    faceNet = 0
    padding = 20
    hasFrame = False
    camera = 0
    label = ""
```

2) Created a Constructor for the class in order to initialize the neural network, use the readNet() method to load the networks. The first parameter holds trained weights and the second carries network configuration. And also initialize the mean values MODEL_MEAN_VALUES. Write them in a function in class “Detect”.



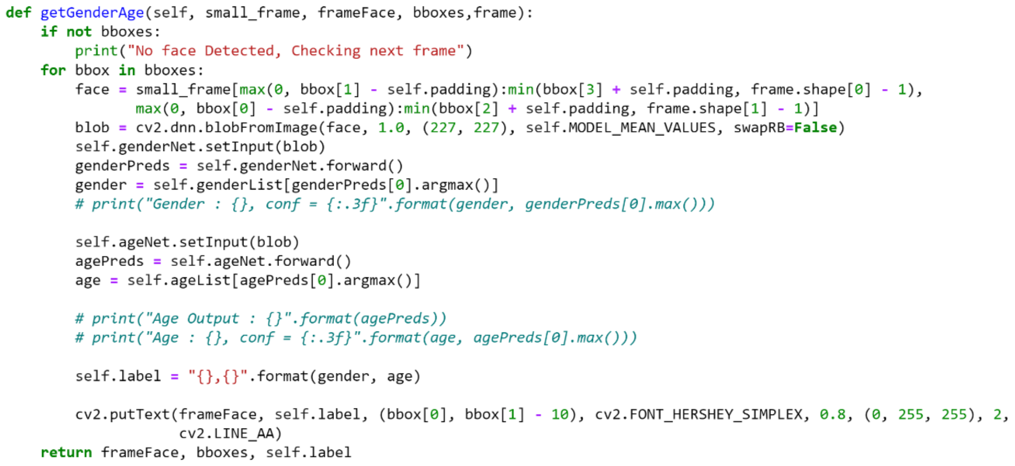
```
def __init__(self, ageModel, ageProto, genderModel, genderProto, faceModel, faceProto):
    self.MODEL_MEAN_VALUES = (78.4263377603, 87.7689143744, 114.895847746)
    self.ageNet = cv2.dnn.readNet(ageModel, ageProto)
    self.genderNet = cv2.dnn.readNet(genderModel, genderProto)
    self.faceNet = cv2.dnn.readNet(faceModel, faceProto)
```

3) Declare a function *getFaceBox* and implement it as it was done and explained above.



```
def getFaceBox(self, net, frame, conf_threshold=0.75):
    frame = np.asarray(frame)
    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()
    bboxes = []

    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)
            bboxes.append([x1, y1, x2, y2])
            cv2.rectangle(frameOpencvDnn, (x1, y1), (x2, y2), (0, 255, 0), int(round(frameHeight / 150)), 8)

    return frameOpencvDnn, bboxes
```

4) Declare another function in class “detect” named *GetGenderAge* to predict gender and age as well as display it in front of us.



```
def getGenderAge(self, small_frame, frameFace, bboxes,frame):
    if not bboxes:
        print("No face Detected, Checking next frame")
    for bbox in bboxes:
        face = small_frame[max(0, bbox[1] - self.padding):min(bbox[3] + self.padding, frame.shape[0] - 1),
               max(0, bbox[0] - self.padding):min(bbox[2] + self.padding, frame.shape[1] - 1)]
        blob = cv2.dnn.blobFromImage(face, 1.0, (227, 227), self.MODEL_MEAN_VALUES, swapRB=False)
        self.genderNet.setInput(blob)
        genderPreds = self.genderNet.forward()
        gender = self.genderList[genderPreds[0].argmax()]
        # print("Gender : {}, conf = {:.3f}".format(gender, genderPreds[0].max()))

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

        # print("Age Output : {}".format(agePreds))
        # print("Age : {}, conf = {:.3f}".format(age, agePreds[0].max()))

        self.label = "{},{}".format(gender, age)

        cv2.putText(frameFace, self.label, (bbox[0], bbox[1] - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 255), 2,
                    cv2.LINE_AA)
    return frameFace, bboxes, self.label
```

VI. GUI

a. Libraries Used

- Tkinter: is a library used for rendering components on a graphical user interface
- Pillow: is another library used in python for image processing, it is used here since tkinter is uses pillow in order to display the images.
- Matplotlib.figure: is responsible for plotting images on the figure
- Matplotlib.backends.backend_tkagg: is responsible for integration between Figure from matplotlib and tkinter

b. Components Used

- LabelFrame
- Label
- Button
- ImageTk
- FigureCanvasTkAgg

c. Functions Used

- Tk(): is used to initialize the window from python
- Geometry(): is used to set the dimension of a window
- Pack(): is used to render the image, it takes parameters to know where to place the component.

VII. Definitions

1) **Relu**: The **rectified linear activation function** or **ReLU** for short is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. It has become the default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance.

2) **Max pooling**: Maximum pooling, or max pooling, is a pooling operation that calculates the maximum, or largest, value in each patch of each feature map.The results are down sampled or pooled feature maps that highlight the most present feature in the patch, not the

average presence of the feature in the case of average pooling. This has been found to work better in practice than average pooling for computer vision tasks like image classification.

3) **Dropout Layer**: The Dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting. Inputs not set to 0 are scaled up by 1/(1 - rate) such that the sum over all inputs is unchanged.

4) **Soft max Layer**: The soft max function is used as the activation function in the output layer of neural network models that **predict a multinomial probability distribution**.

VIII. References

Pereira da Silva, D. (2019, October). *Age and gender classification a proposed system*. ISCTE – IUL. https://repositorio.iscte-iul.pt/bitstream/10071/20194/4/master_david_pereira_silva.pdf

Age and Gender Detection using Python. (n.d.). https://www.irjmets.com/uploadedfiles/paper//issue_3_march_2022/19566/final/fin_irjmets1646664351.pdf

Kunchala, P. (2021, February 7). *Real-time age gender detection using opencv*. Medium. https://medium.com/analytics-vidhya/real-time-age-gender-detection-using-opencv-fa705fe0e1fa

Levi, G., & Hassner, T. (n.d.). *Age and Gender Classification using Convolutional Neural Networks*. https://talhassner.github.io/home/projects/cnn_agegender/CVPR2015_CNN_AgeGenderEstimation.pdf

OpenCV's DNN module and Deep Learning (a definitive guide). LearnOpenCV. (2021, October 18). https://learnopencv.com/deep-learning-with-opencvs-dnn-module-a-definitive-guide/#what-is-opencvv-dnn-module

Raghvendra, Rosebrock, A., yourfanfromKR, Vadim, HongYing, Gianfranco, & Prachi. (2021, April 17). *OpenCV age detection with Deep Learning*. PyImageSearch https://pyimagesearch.com/2020/04/13/opencv-age-detection-with-deep-learning/