Sure, let's go through the code and explain how it works line by line:

```python

# Import required modules

import cv2 as cv

import math

import time

from google.colab.patches import cv2\_imshow

```

- This section imports necessary modules: `cv2` for OpenCV functions, `math` for mathematical operations, `time` for time-related functions, and `cv2\_imshow` for displaying images in Google Colab.

```python

def getFaceBox(net, frame, conf\_threshold=0.7):

frameOpencvDnn = frame.copy()

frameHeight = frameOpencvDnn.shape[0]

frameWidth = frameOpencvDnn.shape[1]

blob = cv.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])

cv.rectangle(frameOpencvDnn, (x1, y1), (x2, y2), (0, 255, 0), int(round(frameHeight/150)), 8)

return frameOpencvDnn, bboxes

```

- `getFaceBox` function detects faces in a given `frame` using a neural network (`net`). It first creates a copy of the frame (`frameOpencvDnn`) and extracts its dimensions (`frameHeight` and `frameWidth`).

- It preprocesses the frame into a blob suitable for input to the neural network (`cv.dnn.blobFromImage`). This includes resizing to 300x300 pixels, normalizing pixel values, and converting from BGR to RGB.

- The neural network (`net`) is then used to perform forward pass inference on the blob, resulting in `detections` of potential objects.

- For each detection with confidence above `conf\_threshold`, it computes the bounding box coordinates (`x1`, `y1`, `x2`, `y2`) relative to the frame dimensions.

- It stores each bounding box in `bboxes` and draws rectangles around detected faces on `frameOpencvDnn`.

- Finally, it returns the modified frame (`frameOpencvDnn`) with rectangles drawn around faces and a list of bounding boxes (`bboxes`).

```python

faceProto = "modelNweight/opencv\_face\_detector.pbtxt"

faceModel = "modelNweight/opencv\_face\_detector\_uint8.pb"

ageProto = "modelNweight/age\_deploy.prototxt"

ageModel = "modelNweight/age\_net.caffemodel"

genderProto = "modelNweight/gender\_deploy.prototxt"

genderModel = "modelNweight/gender\_net.caffemodel"

```

- These lines define the file paths to the prototxt and model files for face detection (`faceProto` and `faceModel`), age estimation (`ageProto` and `ageModel`), and gender detection (`genderProto` and `genderModel`).

```python

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']

```

- `MODEL\_MEAN\_VALUES` are the mean values used for normalization in the age and gender prediction models.

- `ageList` and `genderList` define the age and gender labels that correspond to the model outputs.

```python

# Load network

ageNet = cv.dnn.readNet(ageModel, ageProto)

genderNet = cv.dnn.readNet(genderModel, genderProto)

faceNet = cv.dnn.readNet(faceModel, faceProto)

```

- These lines load the pre-trained neural networks (`ageNet`, `genderNet`, `faceNet`) using the `cv.dnn.readNet` function, which loads models from the provided prototxt and model files.

```python

padding = 20

```

- Defines the padding value (in pixels) to add around the detected face region for better cropping.

```python

def age\_gender\_detector(frame):

# Read frame

t = time.time()

frameFace, bboxes = getFaceBox(faceNet, frame)

for bbox in bboxes:

face = 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)]

blob = cv.dnn.blobFromImage(face, 1.0, (227, 227), MODEL\_MEAN\_VALUES, swapRB=False)

genderNet.setInput(blob)

genderPreds = genderNet.forward()

gender = genderList[genderPreds[0].argmax()]

ageNet.setInput(blob)

agePreds = ageNet.forward()

age = ageList[agePreds[0].argmax()]

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

cv.putText(frameFace, label, (bbox[0], bbox[1]-10), cv.FONT\_HERSHEY\_SIMPLEX, 0.8, (0, 255, 255), 2, cv.LINE\_AA)

return frameFace

```

- `age\_gender\_detector` is the main function that processes a single frame (`frame`).

- It first calls `getFaceBox` to detect faces (`frameFace`) and retrieve bounding boxes (`bboxes`).

- For each detected face (`bbox`), it crops the face region from the frame with padding (`face`).

- Converts the cropped face into a blob suitable for input to the gender and age detection models.

- Performs inference using `genderNet` and `ageNet` to predict gender (`gender`) and age (`age`) from the face blob.

- Draws the predicted label (`gender, age`) on `frameFace` at the location of each detected face's bounding box.

- Finally, it returns `frameFace` with annotated faces showing predicted gender and age labels.

This code integrates face detection, gender prediction, and age estimation using pre-trained models with OpenCV's deep neural network (dnn) module. It's structured to process each frame, detect faces, and provide real-time annotations for gender and age on each detected face region.