AGE AND GENDER PREDICTION USING DEEP LEARNING

Submitted by

Shivangi

For the partial fulfillment of the degree

B.Tech Computer Science Engineering

Under the supervision of

Supervisior

(Dr Dipanwita Thakur, Professor, Banasthali Vidyapith)

Department of Mathematics and Computing

Banasthali Vidyapith

Banasthali – 304022

Session: 2021-2022

CERTIFICATE

मैथमेटिक्स एण्ड कम्प्यूटिंग संकाय
आपाजी संस्थान
वनस्थली विद्यापीठ
पो. वनस्थली विद्यापीठ ३०४०२२
(राजस्थान)

Faculty of Mathematics & Computing
APAJI INSTITUTE
BANASTHALI VIDYAPITH
P.O. BANASTHALI VIDYAPITH 304022
(RAJASTHAN)

Certificate

Certified that **Shivangi (Institute ID BTBTC18319)** has carried out the project work titled " **Age and gender detection using deep learning"** from 6th July 2021 to 25th December 2021 for the award of the **VII semester internship program** from **Banasthali Vidyapith** under my supervision. The report embodies result of original work and studies carried out by the student herself and the contents of the report do not form the basis for the award of any other degree to the candidate or to anybody else.

15/01/22

Dr. Dipanwita Thakur
Assistant Professor
Department of Computer Science
Banasthali Vidyapith
Rajasthan - 304022

Date: 15/01/2022

ABSTRACT

Automatic age and gender classification has ended up pertinent to an expanding amount of applications, especially since the rise of social stages and social media. In any case, performance of existing methods on real-world pictures is still altogether lacking, especially when compared to the tremendous leaps in performance recently detailed for the related assignment of face recognition. In the event that computers ended up as vigorous as people in face recognition it would offer assistance recognizing faces and complex designs, so as the model which will precisely distinguish faces as well as assess age and gender of the individual using machine learning for age and gender estimation. Application which predicts an age bracket as well as a gender between a male and female from a single image received as input.

ACKNOWLDEGEMT

The satisfaction that accompanies that the successful completion of any task would be incomplete without the mention of people whose ceaseless cooperation made it possible, whose constant guidance and encouragement crown all efforts with success.

I give all honor and praise to GOD ALMIGHTY who gave us wisdom and guided me during the entire course of my project.

I express my heartfelt gratitude towards Prof Ina Shastri, Vice Chancellor, Banasthali Vidyapith for granting me permission to work on this project.

I wish to place on record my sincere thanks to our head of the department Prof. C.K.Jha, project supervisior Dr.Dipanwita Thakur for their thoughtful comments and help.

I also express my gratitude and thanks to our staff tutors and all other faculty members of the department of Computer Science and Engineering, Banasthali Vidyapith, for their quick help and expert opinions for completing this project.

SHIVANGI

TABLE OF CONTENTS

OBJECTIVE

Age and gender, two of the key facial traits, play an awfully foundational part in social interactions, making age and gender estimation from a single confront picture a vital task in intelligent applications, such as access control, human-computer interaction, law enforcement, marketing intelligence and visual reconnaissance, etc.

Similar to the actual world the product which can precisely distinguish faces as well as assess age and gender of an individual. Using machine learning this model can recognize age and gender of an individual with precision. Age and Gender Estimation application has four essential modules that are combined to make it useful Face detection, Face Alignment, Feature Extraction and Face Recognition. Caffe model which is a part of deep learning has been implemented to make predictions.

Requirement Analysis

(SRS) Requirement

Specification

The product (i.e. prototype system) enabling age and gender estimation is expected to be deployed it in a real world as an android app or as a web application. It proposes a system which can be utilized at advanced level in various sectors. It will require following to assess its performance and operational functions:

Hardware and Software Requirements

Hardware Requirements at Client end:

- Optionally, Internet access is helpful
- Android Platform/Internet Browser
- Well functioning mobile/computer camera

Software Requirements at Client end (Development setup):

- Operating system: Android, Windows , Linux , Mac OS
- Storage: Between 850 MB and 1.2 GB, depending on the language version
- Memory required - 16 gb RAM (for developer)

Software Interfaces Developing End:

- Microsoft Windows 7/8/10 (32 or 64 bit)
- 500 MB disk space.
- 1 GB for Android SDK
- Python3.0+
- Anaconda
- OpenCV3
- Browser/Internet
- Specific technologies - Convolutional Neural network
- Development Tools - Anaconda (python idle), Android Studio

FEASIBILIITY STUDY

After doing the project AGE AND GENDER PREDICTION USING DEEP LEARNING, study and analyzing all the existing or required functionalities of the system, the next task is to do the feasibility study for the project. All the projects are feasible- given unlimited resources and infinite time.

Feasibility study includes consideration of all the possible ways to provide a solution to the given problem. The proposed solution should satisfy all the user requirements and should be flexible enough so that future changes can be easily done based on the future upcoming requirements.

- **Product-** The project requires a python ide to be developed that will allow users to predict age and gender accordingly.

- **Technical Feasibility**- We can unequivocally say that it is technically attainable, since we are working on a live project. All the assets and databases required for the advancement of the soft-ware as well as maintenance of the same is accessible effectively on web. Created using python language on jupyter notebook.

- **Economic Feasibility**- The application can be developed within budget. The proposed framework is economically doable since the cost involved in Age and Gender Recognition model is irrelevant. The organization can execute this tool into working without any extra consumption. Consequently it is financially feasible.

PRODUCT FUNCTIONS

To continue with the execution of the framework, we got to distinguish the prerequisites that are required for us to be able to conclude it. We require a model that can extract a face from an image, and after that feed it to another model that's able to anticipate the age and gender based on the facial characteristics of the individual that's passed as input.

We can summarize the main system requirements that got to be taken into consideration all through the system execution, which is the taking after-

- The system ought to be able of identifying faces in pictures with a high accuracy rate (more prominent than 90%), taking after other state-of-the-art results on such models.

- The system ought to be competent of predicting the age class of a individual. The precision ought to be similar (or way better) with other state-of-the-art models – it ought to have an precision rate more noteworthy than 60%.

- The system ought to be able of predicting the gender class of a individual. It ought to have a high accuracy rate (more noteworthy than 90%), taking after comparable results from other models.

- The system ought to permit users to approve the comes about. The results from the predictions ought to be• saved into a particular folder where users can approve them physically in the event that required. The results that are stored in such folder should be the ones that surpass a certain confidence threshold defined by the user.

- Age and Gender Classification – A Proposed System Age classes, for validation purposes, used throughout this document are defined as:
  1. Child - Age 1-9

2. Teen - Age 10-15
3. Young - Age 16-24
4. Early Adulthood - Age 25-40
5. Middle Adulthood– Age 41-59
6. Late Adulthood - Age 60-101.

This app illustrates two functionalities:

1. **Import image and detect**: this function enables user to import images from already existing photos in his/her system and process the image to detect face, estimate its age and predict the gender.

   This feature enables users to import picture from their pre existing images stored in system memory and allows the application to process the picture and provide the result.

   Response sequence

   - Click Button “Import and Detect”.
   - User is redirected to system storage.
   - User need to select image accordingly.
   - Detect faces, if not present display ‘no faces detected’.
   - If faces are present, estimate age and gender.
   - Display the result.

   Functional Requirement

   - If image is not of given dimensions then, error message is generated. It asks the user to choose appropriate photo.
   - If photo is blurred, error message is generated-‘Picture is blurry’.

2. **Capture image and detect**: this function enables user to use the system camera to capture photos and process it.

   This feature enables user to open the camera of the user’s system and asks the user to take a picture which is then forwarded to processing and the results are provided.

Response sequence

- Click the button " Capture and detect "
- User will be redirected to camera screen.
- User should now capture a perfect image.
- Detect faces, if not present, display "No face detected".
- If faces are present, the result containing the estimated age and gender of the persons will be displayed.

Functional Requirement

- If image is not of given dimensions then, error message is generated. It asks the user to choose appropriate photo.
- If photo is blurred, error message is generated-'Picture is blurry'

Product will basically have following functions to perform:

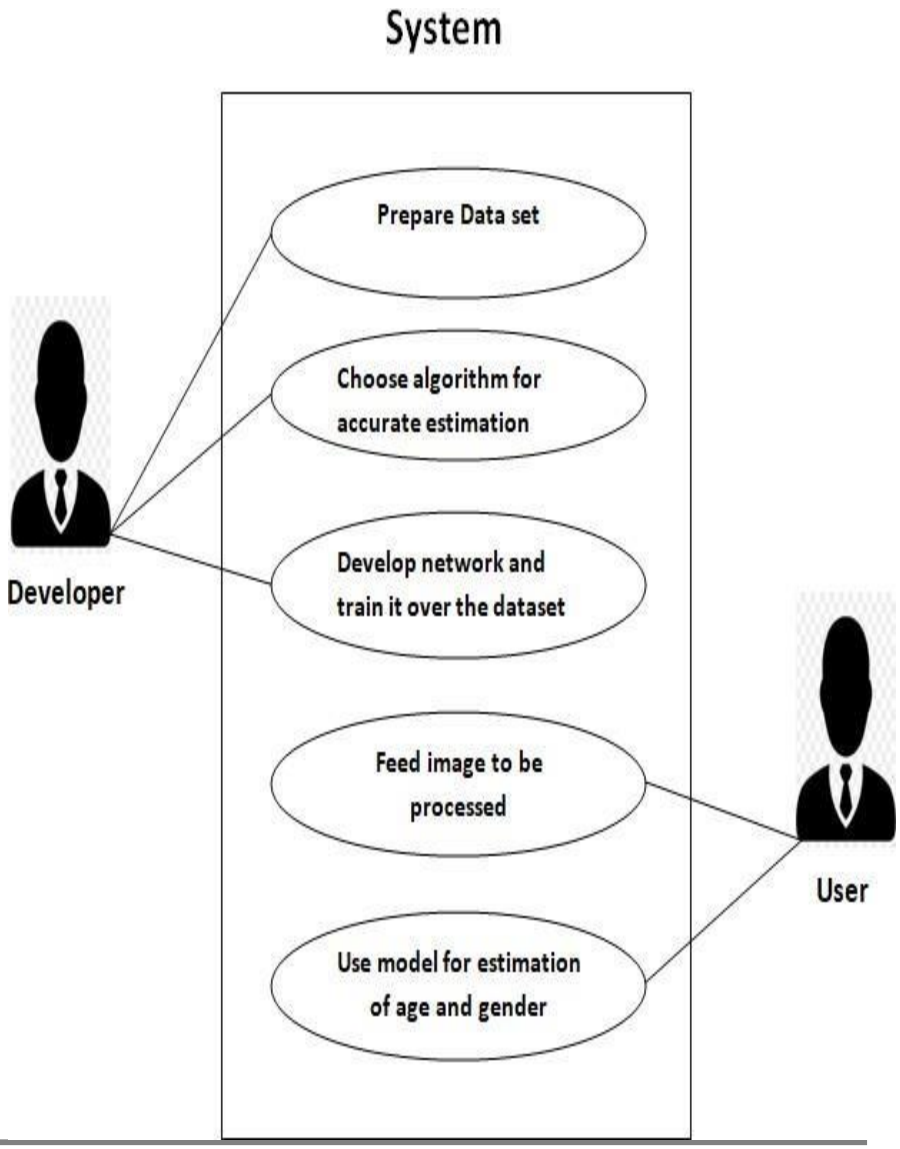
1. **Face detection**: The general structure of face recognition process in this model is made up of three stages. It starts with pre-processing stage: color space conversion and resize of images, continues with extraction of facial features, and afterwards extracted feature set is classified.

2. **Age and Gender estimation**: Deep CNN have additionally been successfully applied to applications including human pose estimation, face parsing, facial key point detection, and speech recognition and action classification. Using a trained CNN as a facial feature extractor is expected to be useful as a keystone for training CNN to estimate the age and gender from the face images. The proposed CNN architecture relies on a very deep face recognition CNN architecture which is capable of extracting facial features distinctively and robustly. As well as, it will be less prone to overfitting.

USE CASE DIAGRAM

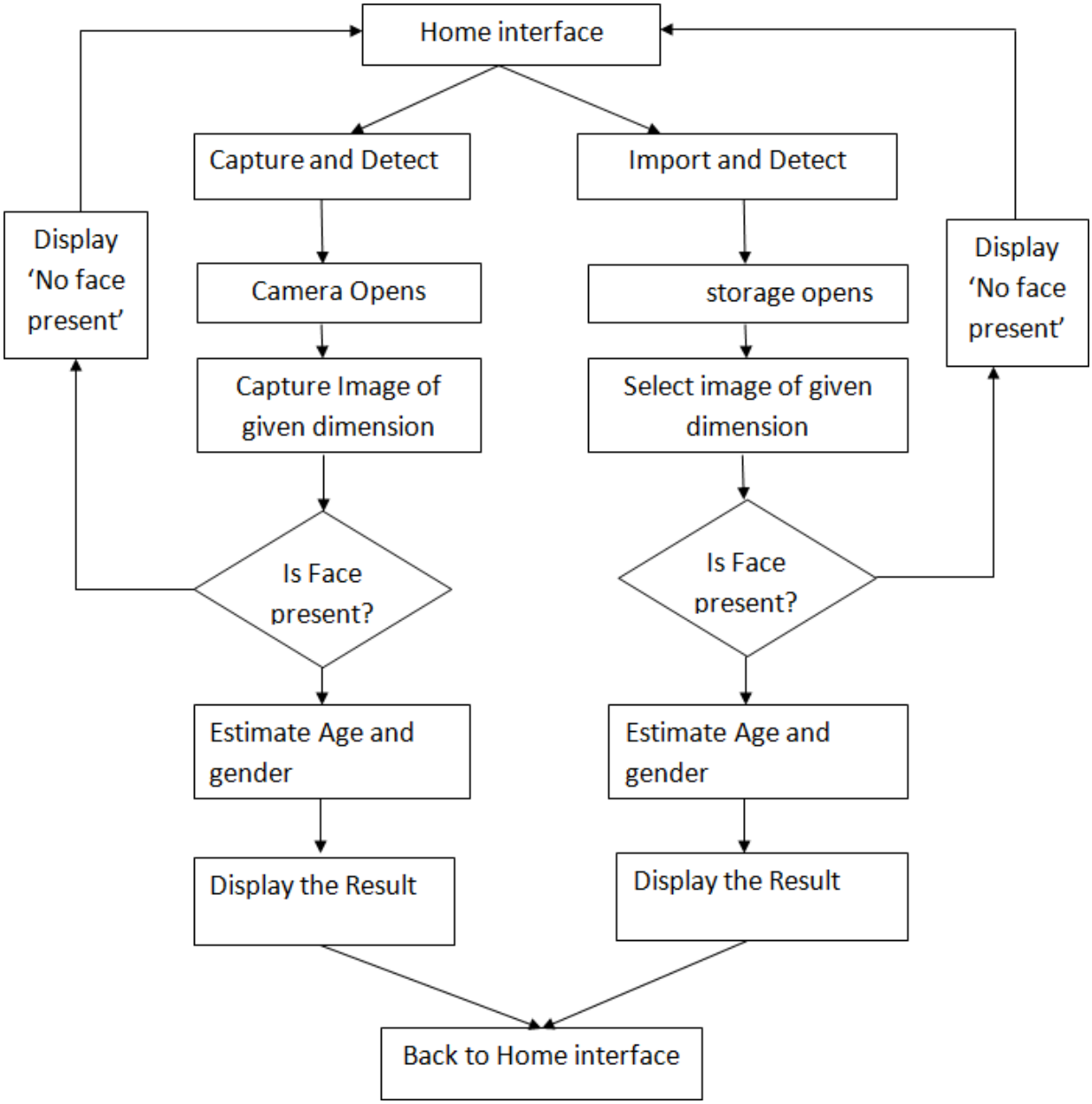
Following processes would be performed while estimation of age and gender:

- Detect faces
- Classify into Male/Female
- Classify into one of the 8 age ranges
- Put the results on the image and display



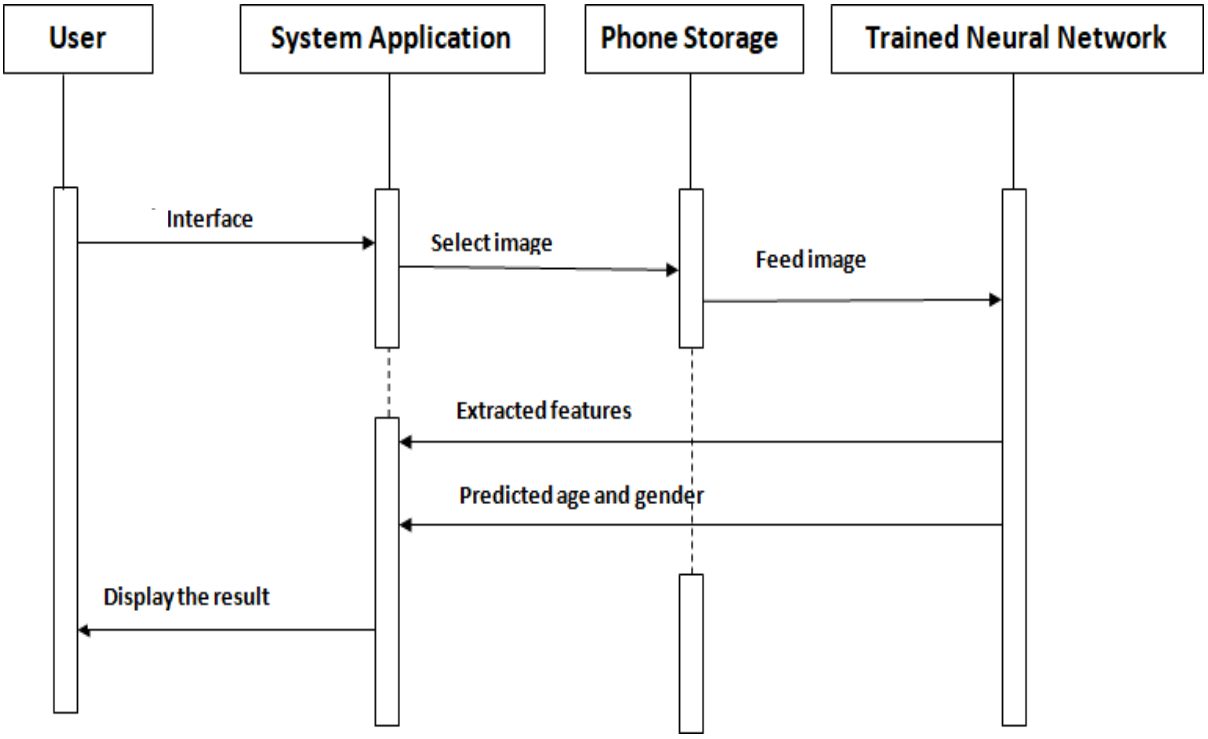
SDS

DATA FLOW DIAGRAM/ACTIVITY DIAGRAM



SEQUENCE DIAGRAM

Sequence diagram for importing an image from the system



In arrange to encourage the study of age and gender recognition, OUI-Adience Face Image give a data set and benchmark of confront photographs. The information included in this collection is expecting to be as genuine as conceivable to the challenges of real-world imaging conditions. In specific, it endeavors to capture all the varieties in appearance, clamor, posture, lighting and more, that can be anticipated of pictures taken without careful preparation or posing.

Total number of photographs: 26,580

Total number of subjects: 2,284

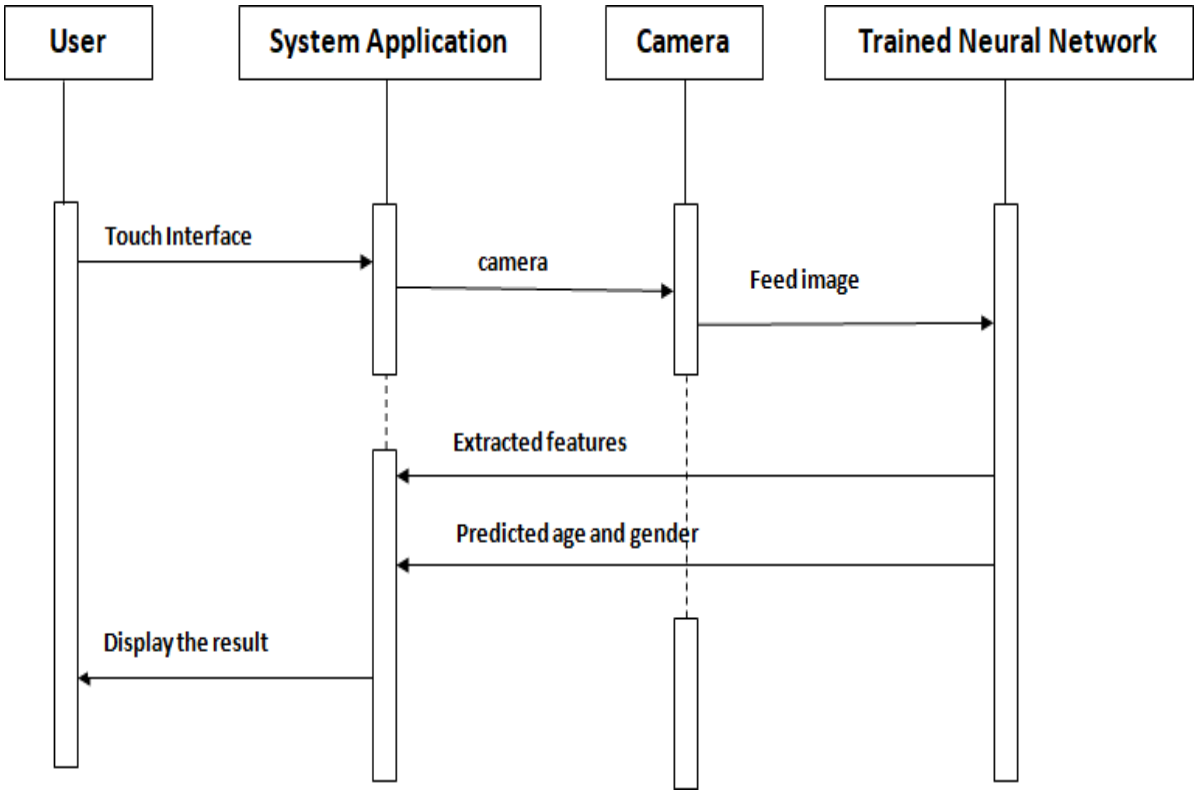
Number of age groups / labels: 8 (0-2, 4-6, 8-13, 15-20, 25-32, 38-43, 48-53, 60-)

Gender labels: Yes

In the wild: Yes

Subject labels: Yes

Sequence diagram for capturing an image from the system's camera



For each of the datasets of IMDb-WIKI and MORPH-II, we part it into two: 90% for training and 10% for validation. On the initial OIU-Adience dataset, training and testing for both age and gender classification are performed utilizing the standard 5-fold cross-validation method .

CODING

MAIN MODULE-

```
!pip install opencv-python --user
```

```
Requirement already satisfied: opencv-python in c:\users\lenovo\anaconda3\lib\site-packages (4.5.4.6
0)
Requirement already satisfied: numpy>=1.17.3 in c:\users\lenovo\appdata\roaming\python\python38\site-
packages (from opencv-python) (1.21.4)
```

```
import cv2 as cv
import math
import time
import argparse
import sys
sys.argv=['']
del sys

#conf_threshold means if the accuracy of recognizing the correct result is greater than 70% then only t

def getFaceBox(net, frame, conf_threshold=0.7):
    frameOpencvDnn = frame.copy() #inputting the frame
    frameHeight = frameOpencvDnn.shape[0]
    frameWidth = frameOpencvDnn.shape[1]
```

```
    frameWidth = frameOpencvDnn.shape[1]
    #creating a 4 dimesional array of image
    blob = cv.dnn.blobFromImage(frameOpencvDnn, 1.0, (300, 300), [104, 117, 123], True, False)
    #by default scale factor is 1.0,resizing the frame to 300X300,mean subtractoin value,swaprb is true

    net.setInput(blob)
    #after preprocessing the image it is passed to neural network

    detections = net.forward()
    #net.forward will gives the output after processing

    bboxes = []
    #Loop for drawing rectangle on the identified face
    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)),
    return frameOpencvDnn, bboxes
```

```
    return frameOpencvDnn, bboxes
parser = argparse.ArgumentParser(description='Use this script to run age and gender recognition using C
parser.add_argument('--input', help='Path to input image or video file. Skip this argument to capture f

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)', '(60-80)']
genderList = ['Male', 'Female']

# Load network
ageNet = cv.dnn.readNet(ageModel, ageProto)
genderNet = cv.dnn.readNet(genderModel, genderProto)
faceNet = cv.dnn.readNet(faceModel, faceProto)
```

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

# Open a video file or an image file or a camera stream
cap = cv.VideoCapture(args.input if args.input else 0)
padding = 20

while cv.waitKey(1) < 0: #while the video is still on/or you haven't exit
    # Read frame
    t = time.time()
    hasFrame, frame = cap.read()
    if not hasFrame: #if there is not frame break the video
        cv.waitKey()
        break

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

    for bbox in bboxes:
        # print(bbox)
        face = frame[max(0,bbox[1]-padding):min(bbox[3]+padding,frame.shape[0]-1),max(0,bbox[0]-padding

        #      for gender preediction
```

```
        #      for gender preediction
        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()]
        # print("Gender Output : {}".format(genderPreds))
        print("Gender : {}, conf = {:.3f}".format(gender, genderPreds[0].max()))

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

        label = "{},{}".format(gender, age)
        cv.putText(frameFace, label, (bbox[0], bbox[1]-10), cv.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 255)
        cv.imshow("Age Gender", frameFace)
```

AGE PREDICTION –

```
name: "CaffeNet"
input: "data"
input_dim: 1
input_dim: 3
input_dim: 227
input_dim: 227
layers {
  name: "conv1"
  type: CONVOLUTION
  bottom: "data"
  top: "conv1"
  convolution_param {
    num_output: 96
    kernel_size: 7
    stride: 4
  }
}
layers {
  name: "relu1"
  type: RELU
  bottom: "conv1"
  top: "conv1"
}
layers {
```

```
layers {
  name: "pool1"
  type: POOLING
  bottom: "conv1"
  top: "pool1"
  pooling_param {
    pool: MAX
    kernel_size: 3
    stride: 2
  }
}
layers {
  name: "norm1"
  type: LRN
  bottom: "pool1"
  top: "norm1"
  lrn_param {
    local_size: 5
    alpha: 0.0001
    beta: 0.75
  }
}
layers {
  name: "conv2"
```

```
}
layers {
  name: "conv2"
  type: CONVOLUTION
  bottom: "norm1"
  top: "conv2"
  convolution_param {
    num_output: 256
    pad: 2
    kernel_size: 5
  }
}
layers {
  name: "relu2"
  type: RELU
  bottom: "conv2"
  top: "conv2"
}
layers {
  name: "pool2"
  type: POOLING
  bottom: "conv2"
  top: "pool2"
  pooling_param {
```

```
  pooling_param {
    pool: MAX
    kernel_size: 3
    stride: 2
  }
}
layers {
  name: "norm2"
  type: LRN
  bottom: "pool2"
  top: "norm2"
  lrn_param {
    local_size: 5
    alpha: 0.0001
    beta: 0.75
  }
}
layers {
  name: "conv3"
  type: CONVOLUTION
  bottom: "norm2"
  top: "conv3"
  convolution_param {
    num_output: 384
    pad: 1
```

```
    kernel_size: 3
  }
}
layers{
  name: "relu3"
  type: RELU
  bottom: "conv3"
  top: "conv3"
}
layers {
  name: "pool5"
  type: POOLING
  bottom: "conv3"
  top: "pool5"
  pooling_param {
    pool: MAX
    kernel_size: 3
    stride: 2
  }
}
layers {
```

```
  name: "fc6"
  type: INNER_PRODUCT
  bottom: "pool5"
  top: "fc6"
  inner_product_param {
    num_output: 512
  }
}
layers {
  name: "relu6"
  type: RELU
  bottom: "fc6"
  top: "fc6"
}
layers {
  name: "drop6"
  type: DROPOUT
  bottom: "fc6"
  top: "fc6"
  dropout_param {
    dropout_ratio: 0.5
  }
}
layers {
```

```
  name: "fc7"
  type: INNER_PRODUCT
  bottom: "fc6"
  top: "fc7"
  inner_product_param {
    num_output: 512
  }
}
layers {
  name: "relu7"
  type: RELU
  bottom: "fc7"
  top: "fc7"
}
layers {
  name: "drop7"
  type: DROPOUT
  bottom: "fc7"
  top: "fc7"
  dropout_param {
    dropout_ratio: 0.5
  }
}
layers {
```

```
}
layers {
  name: "fc8"
  type: INNER_PRODUCT
  bottom: "fc7"
  top: "fc8"
  inner_product_param {
    num_output: 8
  }
}
layers {
  name: "prob"
  type: SOFTMAX
  bottom: "fc8"
  top: "prob"
}
```

GENDER PREDICTION

```
name: "CaffeNet"
input: "data"
input_dim: 10
input_dim: 3
input_dim: 227
input_dim: 227
layers {
  name: "conv1"
  type: CONVOLUTION
  bottom: "data"
  top: "conv1"
  convolution_param {
    num_output: 96
    kernel_size: 7
    stride: 4
  }
}
layers {
  name: "relu1"
  type: RELU
  bottom: "conv1"
  top: "conv1"
}
layers {
```

```
}
layers {
  name: "pool1"
  type: POOLING
  bottom: "conv1"
  top: "pool1"
  pooling_param {
    pool: MAX
    kernel_size: 3
    stride: 2
  }
}
layers {
  name: "norm1"
  type: LRN
  bottom: "pool1"
  top: "norm1"
  lrn_param {
    local_size: 5
    alpha: 0.0001
    beta: 0.75
  }
}
layers {
  name: "conv2"
```

```
  name: "conv2"
  type: CONVOLUTION
  bottom: "norm1"
  top: "conv2"
  convolution_param {
    num_output: 256
    pad: 2
    kernel_size: 5
  }
}
layers {
  name: "relu2"
  type: RELU
  bottom: "conv2"
  top: "conv2"
}
layers {
  name: "pool2"
  type: POOLING
  bottom: "conv2"
  top: "pool2"
  pooling_param {
    pool: MAX
    kernel_size: 3
    stride: 2
  }
}
layers {
  name: "norm2"
  type: LRN
  bottom: "pool2"
  top: "norm2"
  lrn_param {
    local_size: 5
    alpha: 0.0001
    beta: 0.75
  }
}
layers {
  name: "conv3"
  type: CONVOLUTION
  bottom: "norm2"
  top: "conv3"
  convolution_param {
    num_output: 384
    pad: 1
    kernel_size: 3
  }
}
layers{
  name: "relu3"
  type: RELU
  bottom: "conv3"
  top: "conv3"
}
layers {
  name: "pool5"
  type: POOLING
  bottom: "conv3"
  top: "pool5"
  pooling_param {
    pool: MAX
    kernel_size: 3
    stride: 2
  }
}
layers {
  name: "fc6"
  type: INNER_PRODUCT
  bottom: "pool5"
  top: "fc6"
  inner_product_param {
```

```
  inner_product_param {
    num_output: 512
  }
}
layers {
  name: "relu6"
  type: RELU
  bottom: "fc6"
  top: "fc6"
}
layers {
  name: "drop6"
  type: DROPOUT
  bottom: "fc6"
  top: "fc6"
  dropout_param {
    dropout_ratio: 0.5
  }
}
layers {
  name: "fc7"
  type: INNER_PRODUCT
  bottom: "fc6"
  top: "fc7"
```

```
  top: "fc7"
  inner_product_param {
    num_output: 512
  }
}
layers {
  name: "relu7"
  type: RELU
  bottom: "fc7"
  top: "fc7"
}
layers {
  name: "drop7"
  type: DROPOUT
  bottom: "fc7"
  top: "fc7"
  dropout_param {
    dropout_ratio: 0.5
  }
}
layers {
  name: "fc8"
  type: INNER_PRODUCT
  bottom: "fc7"
```

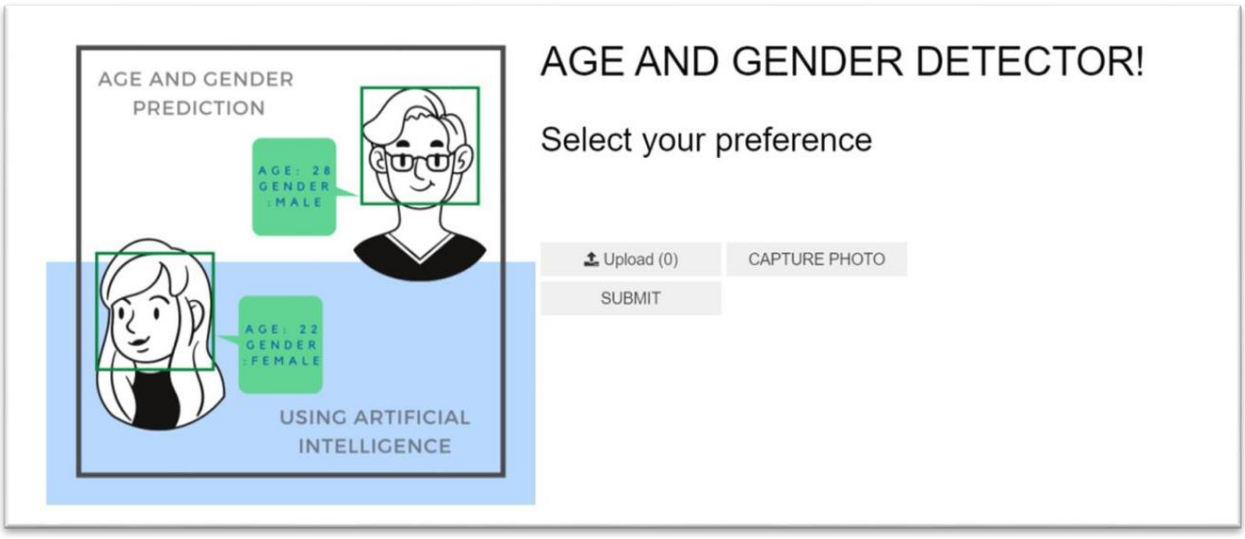
```
  type: INNER_PRODUCT
  bottom: "fc7"
  top: "fc8"
  inner_product_param {
    num_output: 2
  }
}
layers {
  name: "prob"
  type: SOFTMAX
  bottom: "fc8"
  top: "prob"
}

```

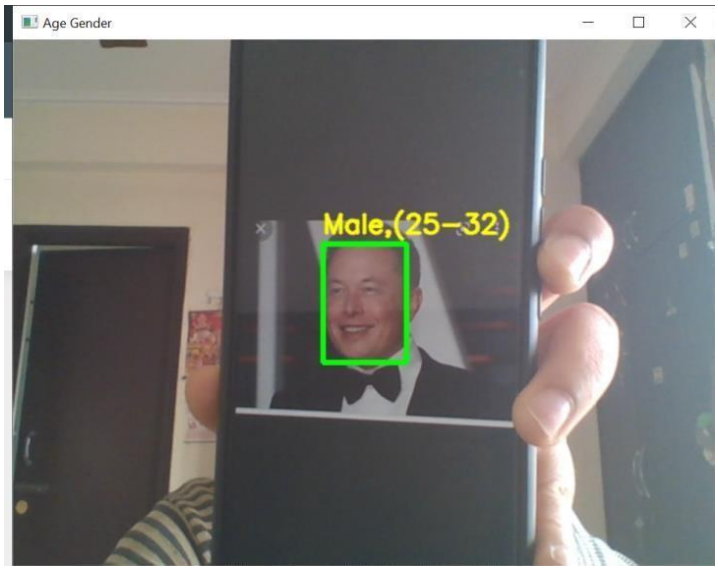
GUI-

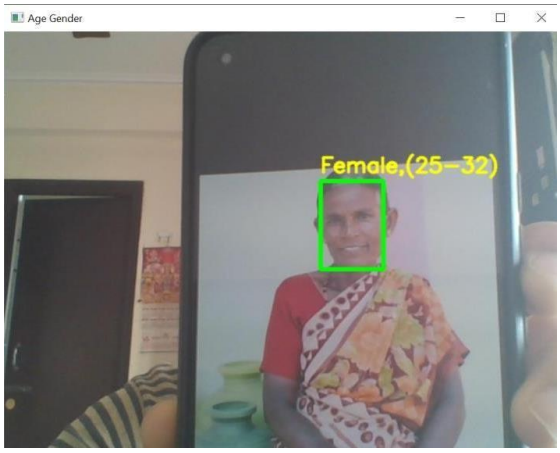
```
import warnings
warnings.filterwarnings('ignore')

import ipywidgets as widgets
from IPython.display import display, clear_output
!jupyter nbextension enable --py widgetsnbextension --sys-prefix
!jupyter serverextension enable voila --sys-prefix
# Image Widget

file = open("proj.png", "rb")
image = file.read()

image_headline = widgets.Image(
                value=image,
                format='png',
                width='400'
            )

vbox_headline = widgets.VBox([image_headline])
#date = widgets.DatePicker(description='Pick a Date')
phone = widgets.FileUpload(
    accept='.jpeg', # Accepted file extension e.g. '.txt', '.pdf', 'image/', 'image/*,.pdf'
    multiple=False, # True to accept multiple files upload else False
    desciption = 'Import and detect'
)
cam = widgets.Button(
            description='CAPTURE PHOTO',
            style={'description_width': 'initial'}
```

```
        )
opt = widgets.HBox([phone,cam])
# button send

button_send = widgets.Button(
            description='SUBMIT',
            tooltip='Submit',
            style={'description_width': 'initial'}
        )

output = widgets.Output()

def on_button_clicked(event):
    with output:
        clear_output()
        print(" ")
        print("PICTURE HAS BEEN SUCCESSFULLY SUBMITTED : ")



button_send.on_click(on_button_clicked)

vbox_result = widgets.VBox([button_send, output])
# stacked right hand side

text_0 = widgets.HTML(value="<h1>AGE AND GENDER DETECTOR!</h1>")
text_1 = widgets.HTML(value="<h2>Select your preference</h2><br><br>")

vbox_text = widgets.VBox([text_0, text_1, opt, vbox_result])
page = widgets.HBox([vbox_headline, vbox_text])
display(page)
```

OUTPUT-


AGE AND GENDER
PREDICTION
AGE: 28
GENDER
:MALE
AGE: 22
GENDER
:FEMALE
USING ARTIFICIAL
INTELLIGENCE
AGE AND GENDER DETECTOR!
Select your preference
Upload (0)
CAPTURE PHOTO
SUBMIT


Age Gender
Male,(25–32)


Age Gender
Female,(25–32)

CONVOLUTIONAL NEURAL NETWORKS (CNNS OR CONVNETS)

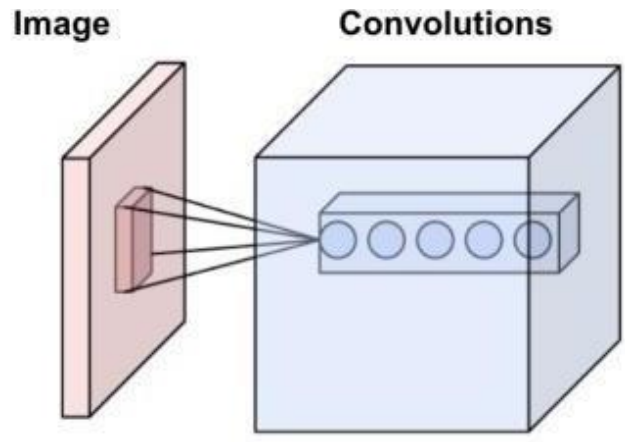
The Convolution Neural Network (ConvNet / CNN) is deep learning algorithm that is utilized as a feature extractor inside the proposed solution.

CNN takes input pictures and allots value to and can recognize between different aspects / objects (learnable weights and predispositions) of the image. ConvNet needs much less pre-processing than other classification algorithms. Whereas the filters are hand-made in primitive methods, ConvNet can learn these filters / features with satisfactory training.

Convolutional neural networks are an uncommon sort of feed-forward networks. These models are laid out to mimic the conduct of a visual cortex. CNNs perform exceptionally well on visual recognition tasks. CNNs have special layers called convolutional layers and pooling layers that permit the network to encode certain images properties.

Convolution Layer

This layer comprises of a set of learnable filters that we slide over the picture spatially, computing dot products between the entries of the filter and the input picture. The channels got to increase to the complete profundity of the input picture. These filters will activate when they see same particular structure within the images.

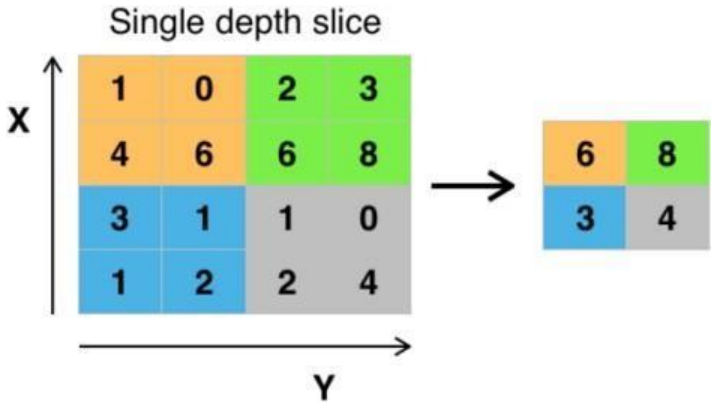


Neurons of a convolutional layer, connected to their receptive field
(source: Wikipedia)

Pooling Layer

Pooling is a form of non-linear down-sampling. The objective of the pooling layer is to continuously decrease the spatial size of the representation to reduce the amount of parameters and computation within the network, and subsequently to also control overfitting.

There are a few functions to execute pooling among which max pooling is the preeminent common one.Pooling is frequently connected with filters of estimate 2x2 connected with a stride of 2 at each



Max pooling with a 2x2 filter and stride = 2

(source: Wikipedia)

depth slice. A pooling layer of estimate 2x2 with stride of 2 shrivels the input picture to a 1/4 of its unique estimate.

RelU (Rectified Linear Unit) Activation Function

Activation functions present non-linearity to the model which permits it to memorize complex useful mappings between the inputs and reaction variables. There are very some different activation functions like sigmoid, tanh, RelU, Defective RelU, etc.

RelU work could be a piecewise linear function that yields the input straightforwardly in case is positive i.e. > 0, otherwise, it'll yield zero.In hone RelU activation function is connected right after a convolution layer and after that that yield is max pooled.

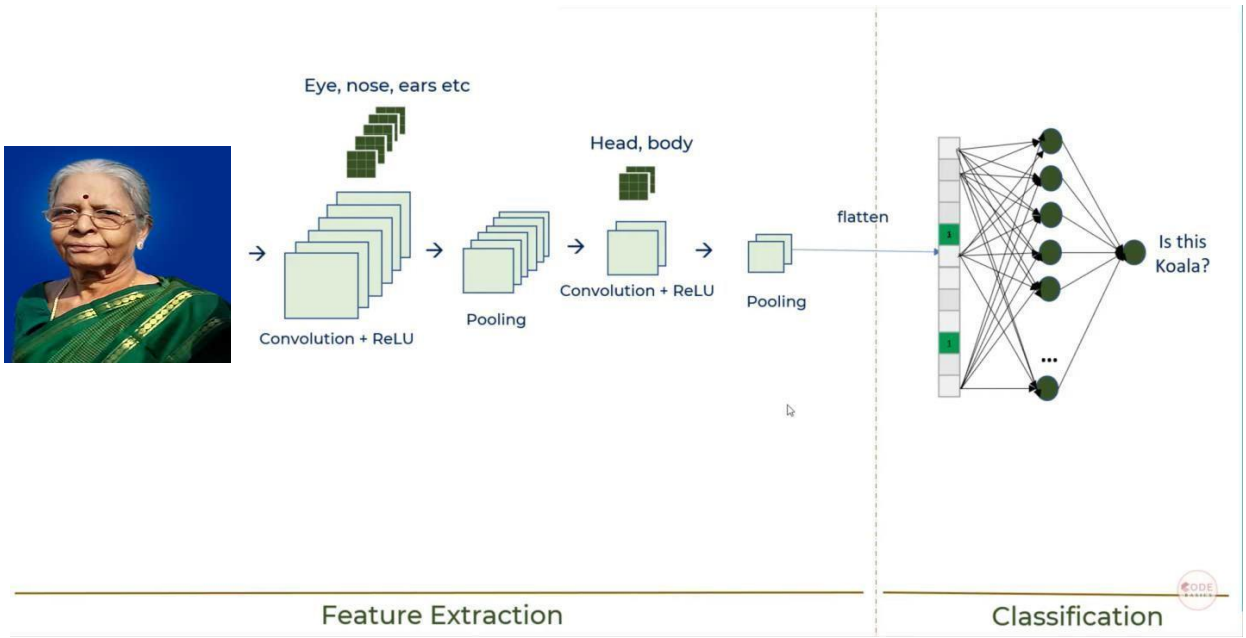
Loss function

Loss is nothing but a forecast blunder of Neural Net. The loss function will yield a higher number in case the expectations are off the actual target values while something else it'll yield a lower number.

Convolution	ReLU	Pooling
• Connections sparsity reduces overfitting.	• Introduces non linearity.	• Reduces dimensions and computation
• Conv and pooling gives location invariant feature detection.	• Speeds up training, faster to compute.	• Reduces overfitting.
• Parameter sharing		• Makes the model tolerant towards small distortion and variations.

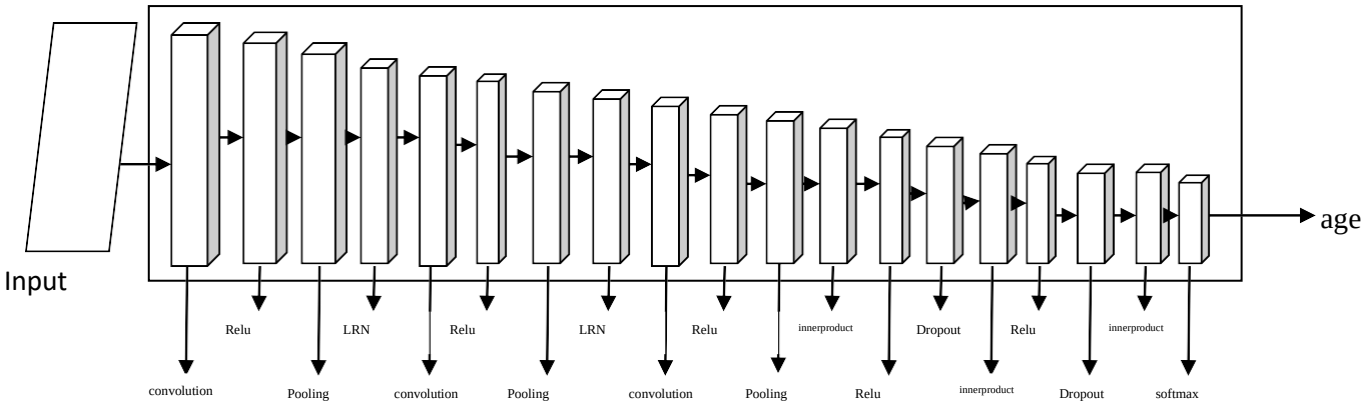
The best engineering of a convolutional neural systems begins with an input layer (pictures) taken after by a arrangement of convolutional layers and pooling layers, and closes with fully-connected layers. The convolutional layers are as a rule taken after by one layer of ReLU actuation functions.

The convolutional, pooling and ReLU layers act as
learnable features extractors, whereas the completely connected layers acts as a machine learning classifier. Besides, the early layers of the network encode generic patterns of the pictures, whereas afterward layers encode the details patterns of the images.

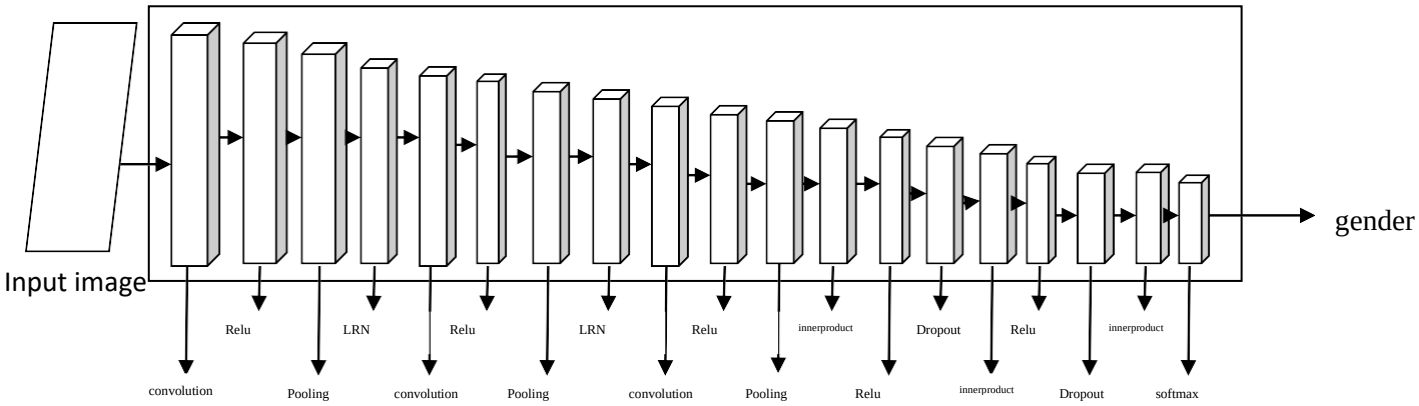


Note that as it were the convolutional layers and fully-connected layers have weights. These weights are learned within the training phase.

CNN ARCHITECTURE OF AGE PREDICTION



CNN ARCHITECTURE OF GENDER PREDICTION



FUNCTIONALITIES OF DIFFERENT LAYERS

Convolution Layer

Layer type: Convolution

Input
n * c_i * h_i * w_i
Output
n * c_o * h_o * w_o, where h_o = (h_i + 2 * pad_h - kernel_h) / stride_h + 1 and w_o likewise.
The Convolution layer convolves the input image with a set of learnable filters, each producing one feature map in the output image.

```
layer {
  name:        "conv1"
  type:  "Convolution"
  bottom: "data"
  top: "conv1"
  # learning rate and decay multipliers for the filters
  param { lr_mult: 1 decay_mult: 1 }
  # learning rate and decay multipliers for the biases
  param    {    lr_mult:    2    decay_mult:    0    }
  convolution_param {
    num_output: 96    #   learn   96   filters
    kernel_size: 11    # each filter is 11x11
    stride: 4         # step 4 pixels between each filter application
}
```

Parameters

Parameters (ConvolutionParameter convolution_param)

- num_output (c_o): the number of filters
- kernel_size (or kernel_h and kernel_w): specifies height and width of each filter
- stride (or stride_h and stride_w) [default 1]: specifies the intervals at which to apply the filters to the input
- pad (or pad_h and pad_w) [default 0]: specifies the number of pixels to (implicitly) add to each side of the input

ReLU / Rectified-Linear and Leaky-ReLU Layer

```
layer {
  name:     "relu1"
  type:     "ReLU"
  bottom: "conv1"
  top: "conv1"
}
```

Given an input value x, The ReLU layer computes the output as x if x > 0 and negative_slope * x if x <= 0. When the negative slope parameter is not set, it is equivalent to the standard ReLU function of taking max(x, 0). It also supports in-place computation, meaning that the bottom and the top blob could be the same to preserve memory consumption.

Parameters (ReLUParameter relu_param)

- negative_slope [default 0]: specifies whether to leak the negative part by multiplying it with the slope value rather than setting it to 0.

Pooling

Layer type: Pooling

Input

n * c * h_i * w_i

Output

n * c * h_o * w_o, where h_o and w_o are computed in the same way as convolution.

```
layer {
  name:     "pool1"
  type:    "Pooling"
  bottom:  "conv1"
  top:        "pool1"
  pooling_param {
  pool: MAX
    kernel_size: 3 # pool over a 3x3 region
    stride: 2      # step two pixels (in the bottom blob) between pooling regions
```

```
  }
}
```

Parameters (PoolingParameter pooling_param)

Required

- kernel_size (or kernel_h and kernel_w): specifies height and width of each filter

Optional

- pool [default MAX]: the pooling method. Currently MAX, AVE, or STOCHASTIC
- pad (or pad_h and pad_w) [default 0]: specifies the number of pixels to (implicitly) add to each side of the input
- stride (or stride_h and stride_w) [default 1]: specifies the intervals at which to apply the filters to the input

Local Response Normalization (LRN)

Layer type: LRN

Parameters (LRNParameter lrn_param)

- local_size [default 5]: the number of channels to sum over (for cross channel LRN) or the side length of the square region to sum over (for within channel LRN)
- alpha [default 1]: the scaling parameter (see below)
- beta [default 5]: the exponent (see below)
- norm_region [default ACROSS_CHANNELS]: whether to sum over adjacent channels (ACROSS_CHANNELS) or nearby spatial locations (WITHIN_CHANNEL)

Inner Product / Fully Connected Layer

Layer type: InnerProduct

Input

n * c_i * h_i * w_i

Output

n * c_o * 1 * 1

Parameters (InnerProductParameter inner_product_param)

- num_output (c_o): the number of filters

Dropout Layer

Layer type: Dropout

Parameters (DropoutParameter dropout_param)

```
message DropoutParameter {
  optional float dropout_ratio = 1 [default = 0.5];
}
```

Softmax Layer

Layer type: Softmax

Parameters

Parameters (SoftmaxParameter softmax_param)

```
message SoftmaxParameter {
  enum Engine {
    DEFAULT = 0;
    CAFFE = 1;
    CUDNN = 2;
  }
```

```
  optional Engine engine = 1 [default = DEFAULT];
  optional int32 axis = 2 [default = 1];
}
```

CAFFE MODEL

Caffe (Convolutional Architecture for Fast Feature Embedding) is a deep learning framework that allows users to create image classification and image segmentation models. Initially, users create and save their models as plain text PROTOTXT files. After a user trains and refines their model using Caffe, the program saves the user's trained model as a CAFFEMODEL file

Deep networks are compositional models that are naturally spoken to as a collection of inter-connected layers that work on chunks of data. The network characterizes the entire model bottom- to-top from input data to loss. As data and derivatives stream through the network inside the forward and backward passes Caffe stores, communicates, and controls the data as blobs.

Blobs

1. A Blob is a wrapper over the genuine data being processed and passed along by Caffe.

2. The blob is the standard cluster of array and bound together memory interface for the framework.

3. The subtle elements of blob portray how data is put away and communicated in and over layers and nets.

4. Blobs give a unified memory interface holding data; e.g., bunches of pictures, model parameters, and derivatives for optimization.

5. The conventional blob measurements for batches of picture data are number N x channel K x stature H x width W. Blob memory is row-major in format, so the final / furthest

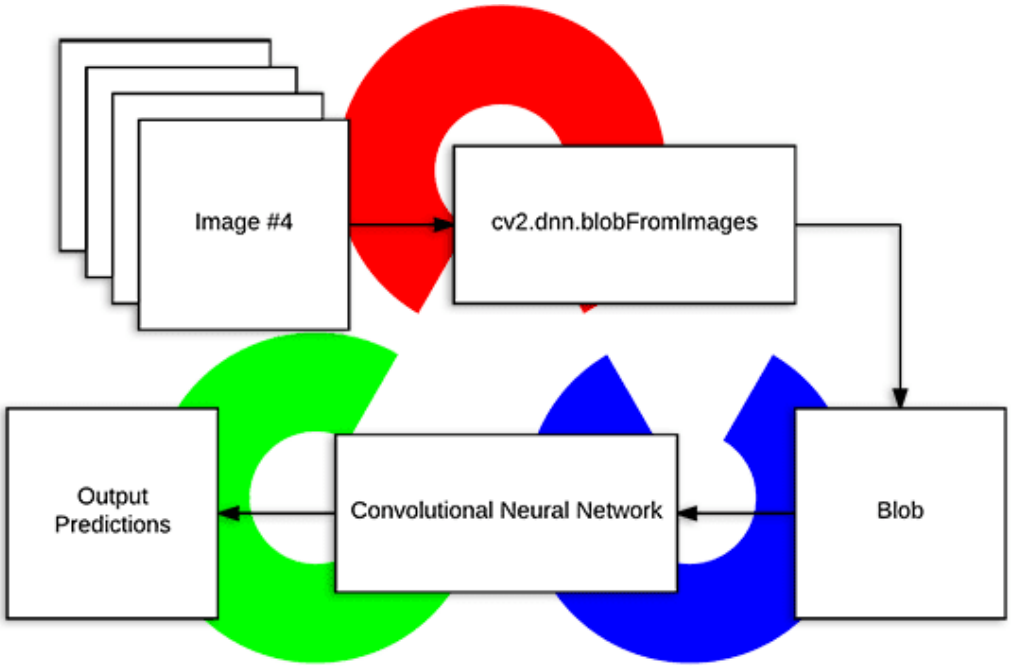
right measurement changes quickest. For case, in a 4D blob, the value at list (n, k, h, w) is physically located at index ((n * K + k) * H + h) * W + w.

As being regularly inquisitive about the values as well as the gradients of the blob, a Blob stores two chunks of memories, data and diff. The past is the normal information that we pass along, and the last mentioned is the gradient computed by the network.

6. The reason for such design is that, a Blob uses a SyncedMem class to synchronize values between the CPU and GPU in order to hide the synchronization details and to play down data transfer.

7. In hone when GPUs are present, one loads information from the disk to a blob in CPU code, calls a device kernel to do GPU computation, and ships the blob off to the taking

after layer, neglecting low-level details while keeping up a high level of performance.



Layer

1. The layer comes taking after as the establishment of both model and computation.

2. The layer is the quintessence of a model and the fundamental unit of computation. Layers convolve channels, pool, take internal products, apply nonlinearities like rectified-linear and sigmoid and other element wise transformations, normalize, stack information, and compute losses like softmax and hinge .

3. A layer takes input through bottom connections and makes output through top connections.

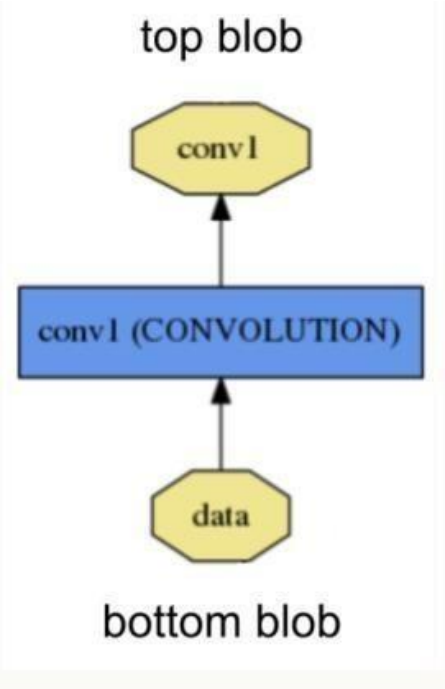
4. Each layer type defines three critical computations: setup, forward, and backward.

Setup: initialize the layer and its connections once at model initialization.

Forward: given input from bottom compute the output and send to the top.

Backward: given the gradient w.r.t. the top output compute the gradient w.r.t. to the input and send to the bottom. A layer with parameters computes the gradient w.r.t. to its parameters and stores it internally.

Layers have two key duties for the operation of the network as a entirety: a **forward pass** that takes the inputs and produces the outputs, and a **backward pass** that takes the gradient with regard to the output, and computes the gradients with respect to the parameters and to the inputs, which are in turn back-propagated to earlier layers. These passes are basically the composition of each layer's forward and backward.



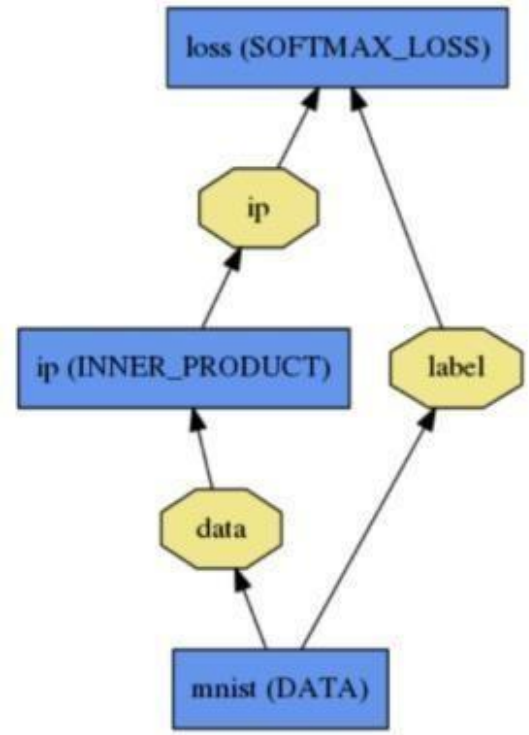
Net

1. The net takes after as the collection and connection of layers.

2. The net mutually characterizes a function and its gradient by composition and auto-differentiation. The composition of each layer's output computes the function to do a given task, and the composition of each layer's backward computes the gradient from the loss to memorize the task. Caffe models are end-to-end machine learning engines.

3. The net could be a set of layers related in a computation chart – a directed acyclic graph (DAG) to be redress. Caffe does all the bookkeeping for any DAG of layers to guarantee rightness of the forward and backward passes. A standard net starts with a data layer that loads from disk and closes with a loss layer that computes the objective for a assignment such as classification or reconstruction.

4. The net is characterized as a set of layers and their affiliations in a plaintext modeling


loss (SOFTMAX_LOSS)
ip
ip (INNER_PRODUCT)
label
data
mnist (DATA)

Model initialization is handled by Net::Init(). The initialization mainly does two things: scaffolding the by and large DAG by making the blobs and layers , and calls the layers'
SetUp() function. It moreover does a set of other bookkeeping things,
such as approving the rightness of the overall network architecture

Blobs and layers cover up execution details from the model definition. After development, the network is run on either CPU or GPU by setting a single switch characterized in Caffe::mode() and set by Caffe::set_mode(). Layers come with comparing CPU and GPU schedules that create indistinguishable results . The CPU / GPU switch is consistent and free of the model definition. For investigate and deployment alike it is best to partition model and implementation.

TESTING

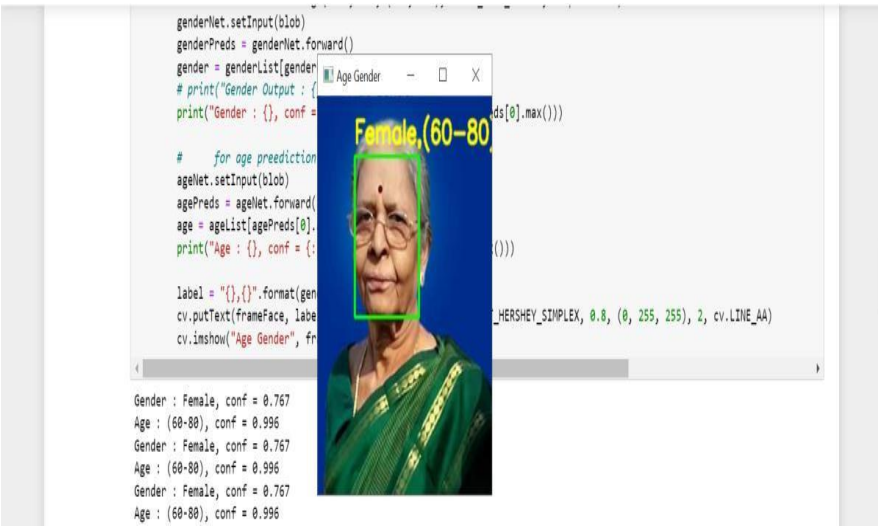
TEST CASES

Component template description

Template used to explain the components given below is as follows-:

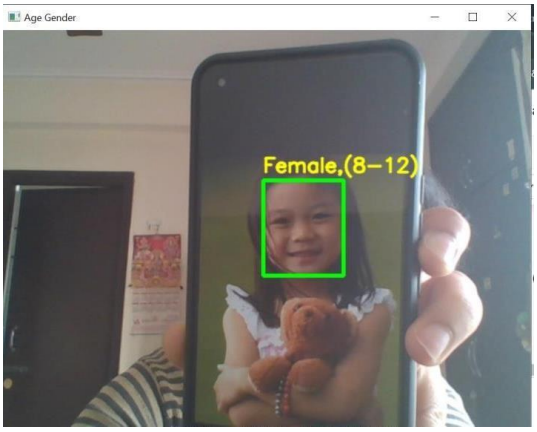
Identification	The unique name for the component and the location of the component in the system.
Type	A module, a subprogram, a data file, a control procedure, a class, etc
Purpose	Function and performance requirements implemented by the design component, including derived requirements.
Function	What the component does, the transformation process, the specific inputs that are processed.
Subordinates	The internal structure of the component, the constituents of the component, and the functional requirements satisfied by each part.
Dependencies	How the component's function and performance relate to other components.
Interfaces	Detailed descriptions of all external and internal interfaces as well as of any mechanisms for communicating through messages, parameters, or common data areas.
Resources	A complete description of all resources (hardware or software) external to the component but required to carry out its functions.
Processing	The full description of the functions presented in the Function subsection.
Accuracy	The percentage of accurate output which matches the pretrained model

Import and detect

Identification	Import and detect It will be present at the home interface of web application
Type	Module
Purpose	To select an already existing picture.
Function	This feature enables users to import picture from their pre existing images stored in memory and allows the web app to process the picture and provide the result.
Subordinates	Redirects to system storage
Dependencies	This module is independent i.e. does not wait for any other task to be completed.
Interfaces	
Resources	Desktop Processing time (4-10 sec)
Processing	• Click Button "Import and Detect". • User is redirected to system storage. • User need to select image accordingly. • Detectfaces, ifnotpresentdisplay 'no facesdetected'. • If faces are present, estimate age and gender. • Display the result

Accuracy	Display the value of age and gender with accuracy greater than 70%.

Capture and detect

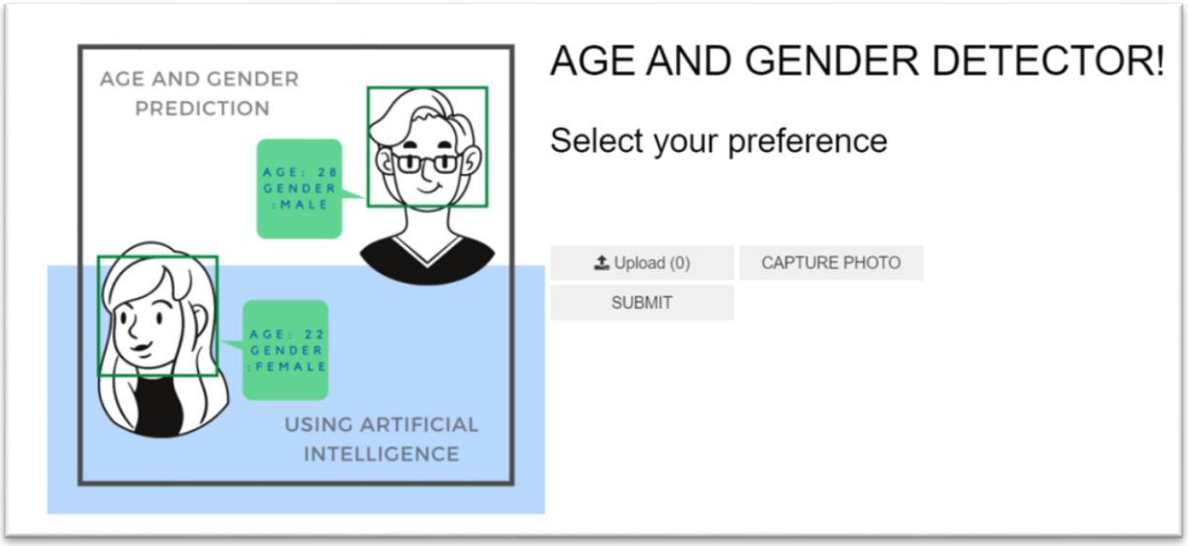
Identification	Capture and detect It will be present at the home interface of web application.
Type	A module
Purpose	To open the video camera and capture to detect age and gender.
Function	This feature enables user to open the camera and click a picture which will be preprocessed with the help of CNN algorithm and provides the result.
Subordinates	Redirects to “camera” of the system.
Dependencies	This component is independent. It doesn’t wait for other component to complete the task .
Interfaces	
Resources	Desktop Processing time (4-10 sec)
Processing	• Click the button " Capture and detect " • User will be redirected to video screen. • User should now capture a perfect image. • Detect faces, if not present, display "No face detected". If faces are present, the result containing the estimated age and gender of the persons will be displayed

Accuracy		Display the value of age and gender with accuracy greater than 70%.

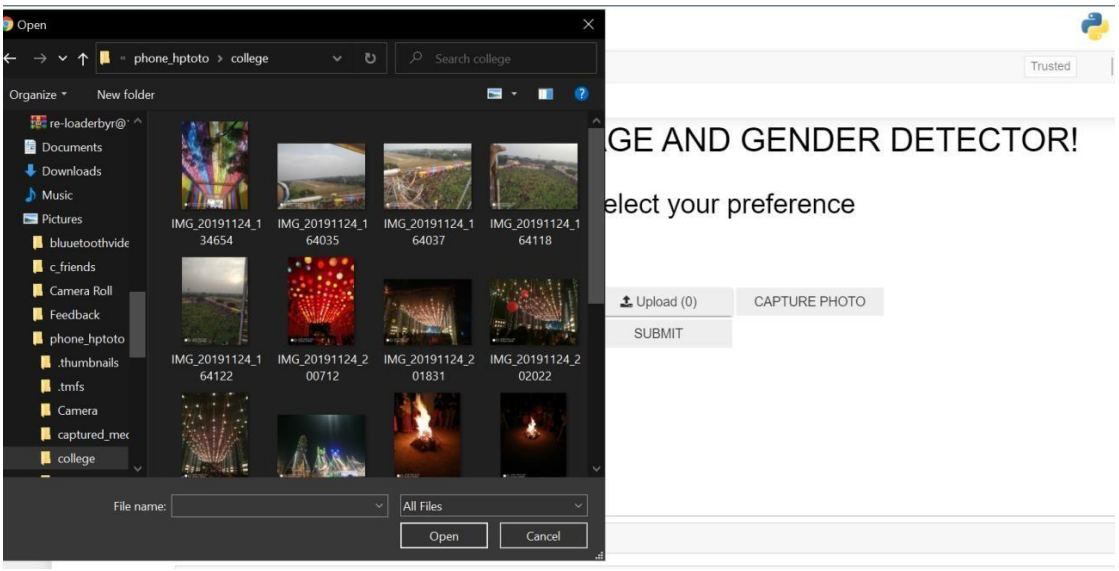
USER INTERFACES

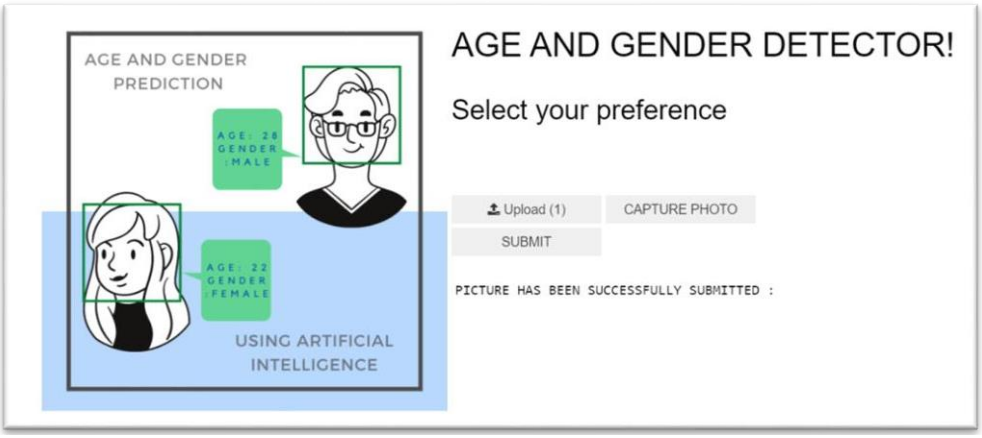
Actual look of the product interfaces are-

- Home interface of web application:

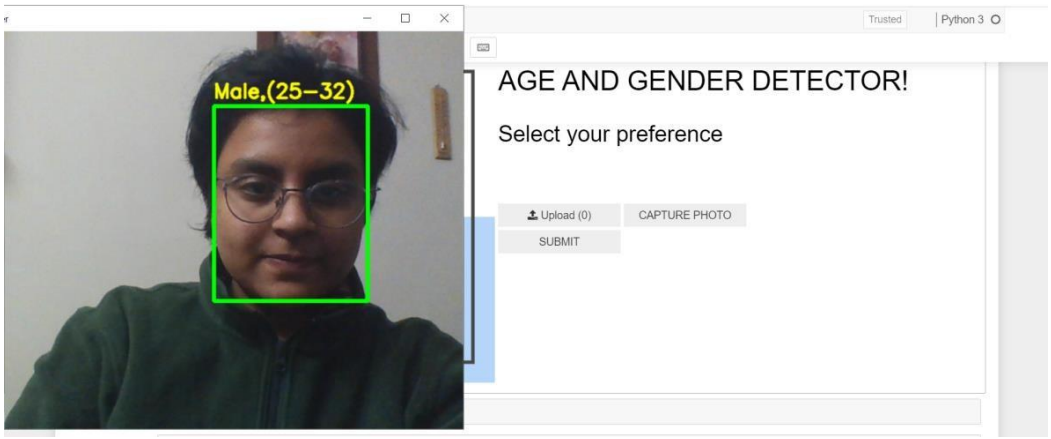


- Import and Detect/Upload:





- Capture and Detect:



ACCURACY

Python layer for the Caffe deep learning framework to compute the accuracy and plot the receiver operating characteristic(ROC) curves.

```
layer {
  type: 'Python'
  name: 'py_accuracy'
  top: 'py_accuracy'
  bottom: 'ip2'
  bottom: 'label'
  python_param {
    module: 'python_roc_curves'
    layer: 'PythonROCCurves'
    param_str: '{"test_iter":100, "show": "yes", "savefig": "result", "figformat", "png"}'
  }
  include {
    phase: TEST
  }
}
```

the module title -- usually the filename -- that has to be in $PYTHONPATH

the layer name -- the class title within the module layer: 'PythonROCCurves'

a set of parameters, incl: # test_iter: number of cycles the ROC plotting is triggered

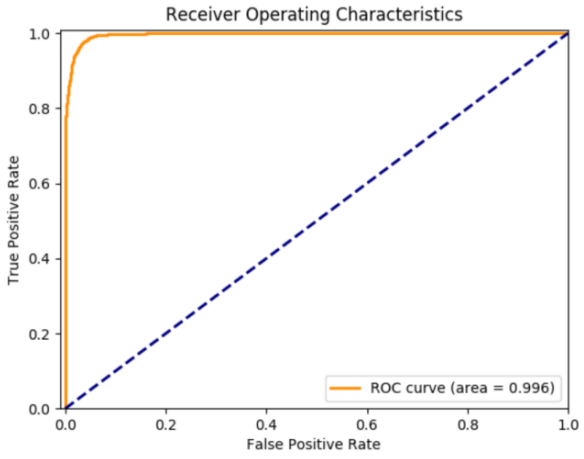
appear: yes to show ROC plot in each test_iter iterations

savefig: the ROC plot can be stored into picture files with this parameter as the file title prefix.

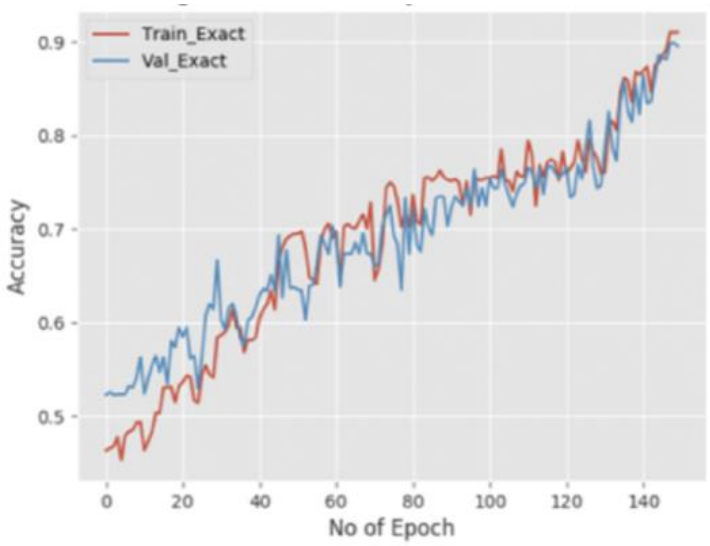
figformat: record name expansion as the format.

This layer will plot the ROC curves of the predictions after the set of each test case have been processed. It'll moreover work as an accuracy layer, giving Caffe with the expectations precision on the set.

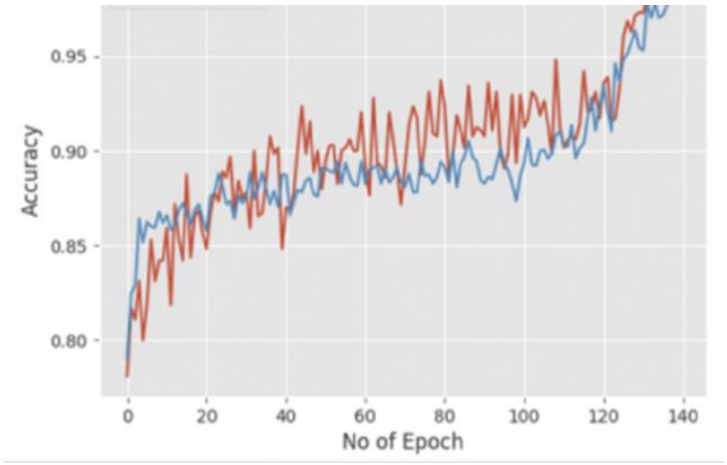
The is utilized as an accuracy layer within the prototxt file like:



Age: accuracy of OUI- Adience Face Image



Gender: accuracy of OUI- Adience Face Image



REFERENCES

- https://erandiganepola.medium.com/machine-learning-based-age-and-gender-predictions-in-image-processing-223031dea847
- https://www.hindawi.com/journals/tswj/2020/1289408/
- https://www.researchgate.net/publication/321479959_Facial_detection_using_deep_learning
- Yangqing Jia (2014) Caffe: Convolutional Architecturefor Fat Feature Embedding