**CHAPTER-1**

**INTRODUCTION**

* 1. **MOTIVATION**

The technique of generating captions for an image is known as Image Captioning. We must first comprehend the significance of this challenge in real-world scenarios. Let us consider few scenarios in which a solution to this problem could be extremely beneficial.

Automatic captioning could help Image Search become as good as Web Search, because every image could be transformed into a caption first, and then searches could be conducted based on the caption.

CCTV cameras are meant for monitoring, but if we can provide useful captions in addition to watching the world, we can trigger warnings as soon as dangerous conduct is detected. This is likely to help minimise crime or accidents.

We can design a product for the blind that will assist them in navigating the roadways without the assistance of others. This may be accomplished by first translating the scene to text, then the text to speech. Both are now well-known Deep Learning applications.

**1.2 PROBLEM DEFINITION**

Every day, we are exposed to a significant number of images from a variety of sources, including the internet, news articles, schematics in documents, and advertisements. These resources provide visuals that visitors must interpret for themselves. Majority of photos do not contain a description, yet humans can make sense of them without them. However, if people want automated image captions from the machine, the system must be able to understand and interpret them.

In this Project, we are going to implement a encoder-decoder architecture where the encoder is a pre-trained models like VGG16, ResNet50, InceptionV3 and MobileNet. It may be a single model or the combination of vectors consisting of high-level features from two or more models works exceptional in most of the cases. The decoder part is a slightly modified LSTM network and contains the Time Distributed Layer which are helpful for time series data as we post the entire sentence or sequence as a time series problem and also in case of videos as well which works fabulously for storing sequence information and text generation.

**1.3 OBJECTIVE OF THE PROJECT**

* The main objective of this project is to provide a technology oriented, low computational power, easily scalable, and a robust model.
* The secondary thing is to set a color composition to images and represent the high-level features in an effective way using several pre-trained models.
* Treating the input sentence/video as a time series data and maintaining the sequence information using time distributed LSTM layers.
  1. **LIMITATIONS OF PROJECT**
* Captioning is now available for image formats, but that will be accessible for video formats eventually.
* Our Deep Learning Model itself is based on objects present in the Image Captioning datasets. Majority of work in image captioning has been limited due to image captioning datasets.

**1.5 ORGANISATION OF DOCUMENTATION**

1.5.1 Feasibility Study

Preliminary investigation examines project feasibility; the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All systems are feasible if they are given unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

* Technical Feasibility
* Operation Feasibility
* Economic Feasibility

**Technical Feasibility**

The technical issue usually raised during the feasibility stage of the investigation includes the following:

* Does the necessary technology exist to do what is suggested?
* Do the proposed equipment‘s have the technical capacity to hold the data required to use the new system?
* Will the proposed system provide adequate response to inquiries, regardless of the number or location of users?
* Are there technical guarantees of accuracy, reliability, ease of access and data security?

**Operation Feasibility**

The operational feasibility includes User friendly, reliability, security, portability, availability, and maintainability of the software used in the project.

**Economic Feasibility**

Analysis of a project costs and revenue in an effort to determine whether or not it is logical and possible to complete.

**CHAPTER-2**

**LITERATURE SURVEY**

**2.1 LITERATURE REVIEW**

Several works have been done so far for real time face detection, facial emotion recognition and gender-age classification . For this project, we reviewed the current literature on convolutional face detection and gender and classification and facial emotion recognition. We found that convolutional face detection and gender/emotion classification is still evolving as a technology, Dr.Hlaing Htake Khaung Tin

Proposed that Gender classification is important visual tasks for human beings, such as many social interactions critically depend on the correct gender perception. Age prediction is the determination of a person’s age based on biometric features. It is an important technique to predict age form facial pictures automatically in computer vision.

**2.2 EXISTING SYSTEM**

1. There are two different models for facial emotion recognition and age, gender prediction, which won't be user friendly to detect all the facial attribute at a time.
2. All the three attributes requires three different model, one each to analyze and predict also it makes use more space and requires more time to obtain three different models using CNN.

.

**2.3 DISADVANTAGE OF EXISTING SYSTEM**

* Age and gender prediction model requires more memory space to load all the images for classification. That leads to unable to allocate the .GB memory for the image in pixels

**2.4 PROPOSED SYSTEM**

1. The new system will predict emotion,gender and age of the faces camera at a time.It is more efficient compared to existing system
2. The developed method uses pre trained age and gender prediction model known as caffe model which limits the memory area and reduces time

**2.5 ADVANTAGES OF OUR SYSTEM**

* CNN is used only once to recognize facial emotions and only single GUI application for all the face attribute analysis
* Better Results

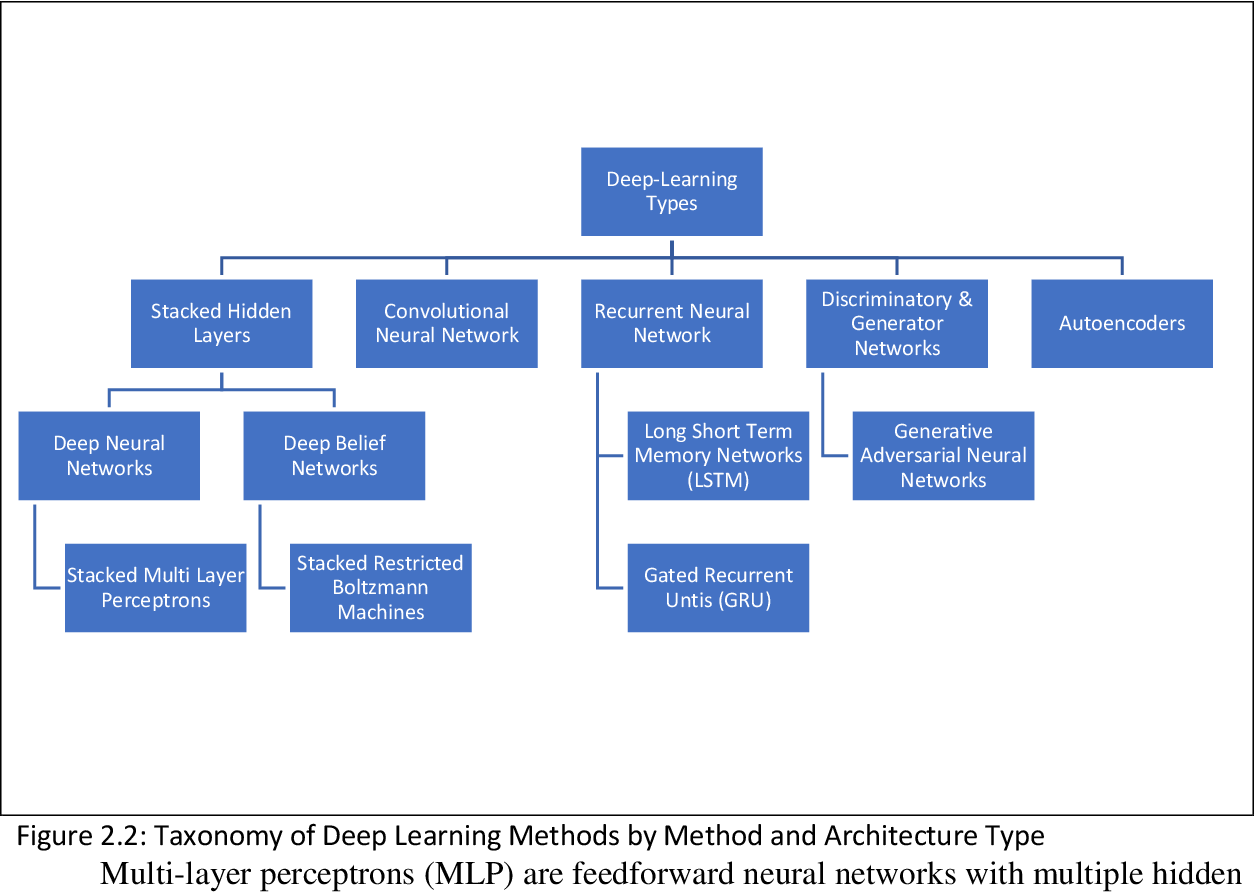
**CHAPTER-3**

**ANALYSIS**

**3.1 INTRODUCTION**

Our project title**, " Real-Time Face Recognition and Facial Attribute (Age,Gender, Emotion) Prediction”** is meant to become more reliant on virtual interactions such as Zoom meetings / Teams chat. These livestream webcam videos have become a rich data source to explore. Analyzing expressions on the person’s face plays a very vital role in identifying emotions and behavior of a person. Recognizing these expressions automatically results in a crucial component of natural human-machine interfaces. Therefore research in this field has a wide range of applications in biometric authentication, surveillance systems, emotions to emoticons in various social media platforms.

The most accelerated technologies of this era are deep learning and machinery learning. Artificial intelligence is now compared to the human mind and it does great work than people in some fields. New research in this area occurs every day and this field is growing very quickly because we now have enough computational power to do this. Deep learning is a machine learning branch that uses many-layered neural networks. Deep learning networks are often enhanced by increasing the amount of data used to train them. The comprehension of an image depends largely on image features. The types of learning techniques is shown in the fig. 3.1.



**Fig 3.1 Types of Learning**

**3.2 HARDWARE AND SOFTWARE DESCRIPTION**

**HARDWARE DESCRIPTION**

Processor : Intel i5

Ram : 8GB

Hard Disk Space : 50GB

**SOFTWARE DESCRIPTION**

Operating System : Windows7/8/10 or Ubuntu

Back-end Design : Python3

Tool : Anaconda

**ANACONDA:**

Anaconda Enterprise is a business-ready, safe, and scalable data science platform that enables teams to manage, collaborate, and implement data science projects.

* Deploy your projects easily into interactive data applications, live notebooks, and API learning machines.
* Share applications with colleagues and collaborators.
* Manage your data in a comprehensive data science experience: notebooks, packages, environments, and project in an integrated data science experience.
* The Anaconda.org command line interface (CLI), anaconda-client, allows you to manage your account - including authentication, tokens, upload, download, remove, and search.

**Installation:**

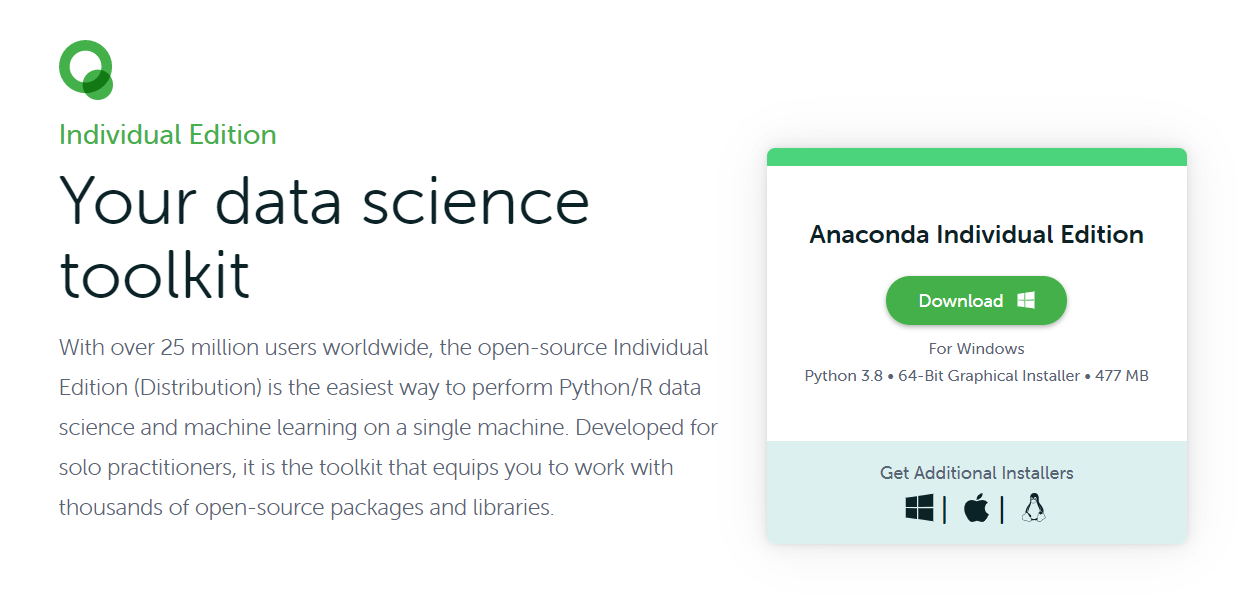


Fig 3.2 **Anaconda Installation**

1. Download the Anaconda installer from the official anaconda source.
2. Verify the data integrity of the Anaconda installer files by running a local program to generate their SHA-256 cryptographic hashes and checking the output to be sure it matches the hashes below.
3. To start the installer, double-click.
4. Click Next and read the terms and conditions of the licence and click “I Agree”.
5. Select an install for “Just Me” unless you are installing for all users (which requires Windows Administrator privileges) and click Next.
6. To install Anaconda, select the destination folder, click Next.

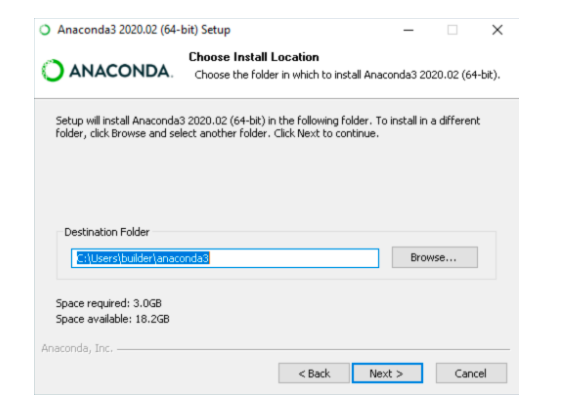
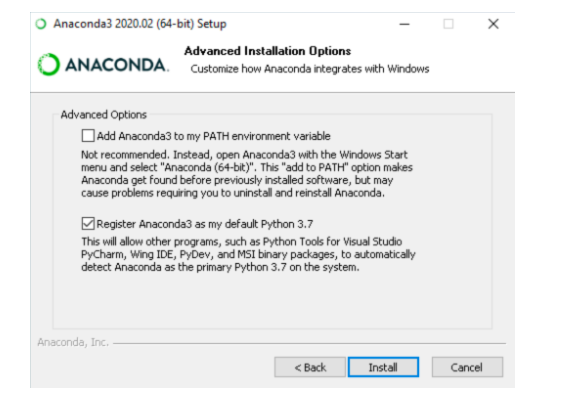


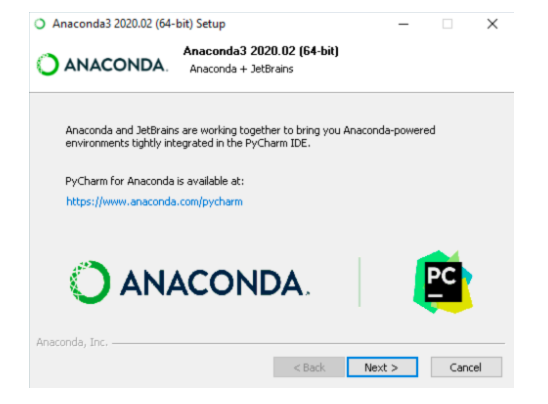
Fig 3.3 **Anaconda Installation**

1. Choose whether to add Anaconda to your PATH environment variable.



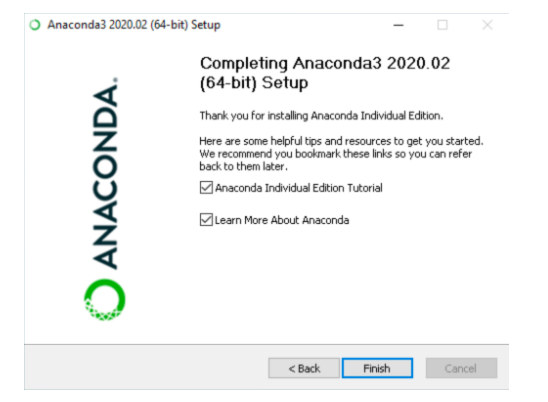
**Anaconda Installation (3.2.3)**

1. Choose whether to register Anaconda as your default Python.
2. Click the Install button



**Anaconda Installation (3.2.4)**

1. Click the Next button and it is optional to install PyCharm for Anaconda or to install Anaconda without PyCharm, click the Next button.
2. After a successful installation you will see the “Thanks for installing Anaconda” dialog box.



**Anaconda Installation (3.2.5)**

Once the installation is done you can launch the Jupyter notebook through anaconda command prompt running simple commands and install all the necessary packages that we use in the project.

The modules that should be installed are listed below:

1.TensorFlow

2. Keras

3. Numpy

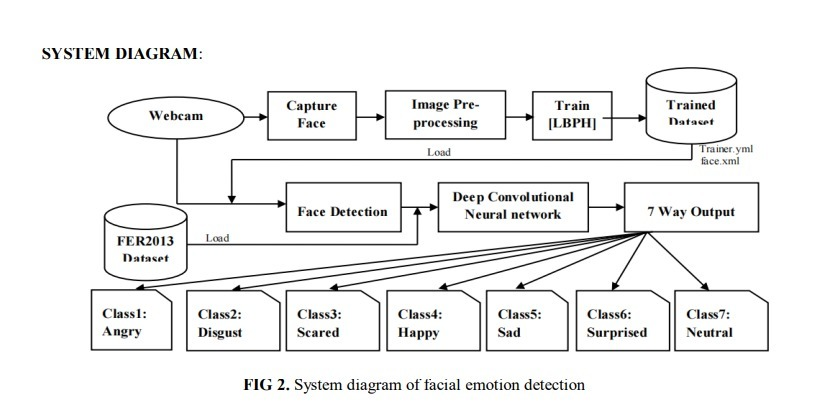
4. NLTK

5. Pandas

6. Opencv

**3.3 CONTENT DIAGRAM OF PROJECT**

**SYSTEM DIAGRAM (3.3.1)**



**Fig:3.3.1**

**WORKING OF THE PROJECT:**

Real time face detection and gender & emotion recognition is a robust complex problem in computer vision because of real time image frames. First, we have to take a real time video frame then convert it as an image and extract the face from the image to detect the human face. After extracting face, we consider each face part of the image as a full image for further processing. Each extracting face image is then providing as input to pre pre-process step of classification model and each pre-processing step takes some operation on it input to resize as model input and data augmentation as input to our proposed convolutional neural network (CNN) model for classification of the emotion, age and gender. The resulting label that is output of the CNN is then used for making descriptions of gender {“man” or “women”} and facial emotion classification {“angry”, “disgust”, “fear”, “happy”, “sad”, “neutral”}and their age in number .

It mainly follow these four steps:

1. Receive input frame from the webcam
2. Identify faces in the webcam and prepare these images for the 3 deep learning models, i.e. age, gender and emotion models
3. Send processed faces to the models and receive prediction outcomes
4. Render prediction outcomes with bounding boxes to screen

**CHAPTER-4**

**DESIGN**

**4.1 INTRODUCTION**

This model is the implementation of an ensemble CNN for building a real-time system that can detect emotion, age and gender of the person. The experimental results shows accuracy of 68% for emotion classification into 7 classes (angry, fear, sad, happy, surprise, neutral, disgust) on FER-2013 dataset. Automatic age and gender prediction from face images has lately attracted much attention due to its wide range of applications in numerous facial analyses. We show in this study that utilizing the caffe model architecture of deep learning framework.

Also when input is given through webcam our complete pipeline of this real-time system can take less than 0.5 seconds to generate results These images are divided as follows:

* Training Set — 6000 images
* Dev Set — 1000 images
* Test Set — 1000 images

**4.2 UML DIAGRAMS**

**Diagram

Description automatically generated**

**Fig 4.1 Class Diagram**

Diagram

Description automatically generated

**Sequence Diagram (4.2.2)**

**Use case Diagram (4.2.3)**

Diagram

Description automatically generated

**Flow Chart(4.2.5)**

**4.3 MODULE DESIGN AND ORGANIZATION**

**4.3.1 LIST OF MODULES**

* Face capturing
* Data Pre-processing
* Training
* Face recognition
* Emotion, age and gender recognition

Diagram

Description automatically generated

**MODULES (4.3.1.1)**

**Face capturing module:**

During this phase, we are taking pictures of people's faces for further processing. We are utilising a webcam or an external web camera for this purpose. There is no way to complete the procedure without first taking the image, and there is no way to identify the emotions without first capturing the image.

**Data Preprocessing module:**

1. Following the capture of photos, we will do image processing on the captured images. The grey scale photos will be created by converting the colour photographs to grey scale.
2. In this stage, we cleaned the photos and saved them in a caffe-compatible format. A python script to handle picture pre-processing, as well as storage, is developed.

**Training**

One Caffe command from the terminal is used to train the model. The trained model in a file with the extension .caffemodel is then obtained after we have trained the model. We used a .pb file for face recognition, which is a protobuf file (protocol buffer) that carries the graph description and training weights of the model.

Diagram

Description automatically generated

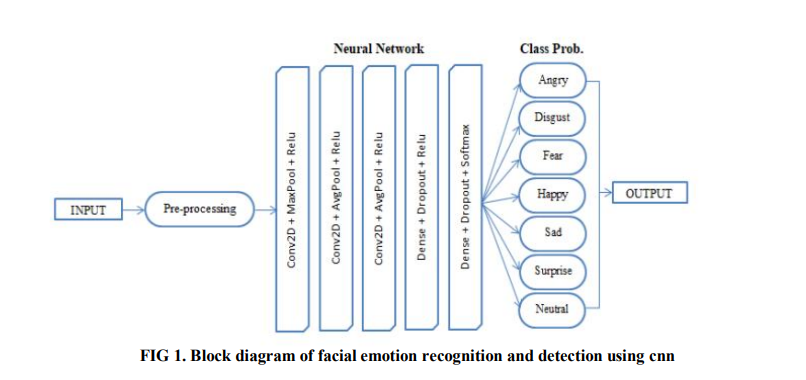
**Face Emotion,Age and Gender Prediction:**

Facial expression recognition software is a system that detects emotions in human faces.

The gender of the individual in the photograph is determined by the higher of the two numbers. After that, the process is repeated again to determine ages. Gender and age texts will then be added to the final image and use imshow to display it.

Diagram

Description automatically generated



**4.4 CONCLUSION**

Based on all the diagrams we are able to design the required functionalities and the flow of data that is to be maintained between each of them. By doing all this we are able to maintain the application without any bugs and errors. All the diagrams that are developed show us the functionalities of the model and the application developed on top of it.

**CHAPTER-5**

**IMPLEMENTATION AND RESULTS**

**5.1 INTRODUCTION**

Face recognition is a deep learning-based model which contains complex architectures and corresponding weights that are to be trained for better performance and predictions. Integrating different functions is one of the crucial things in training the model because each layer should be compatible with the next corresponding layers. Choosing a right Optimizer and loss function makes the model performance better in short period of time. The computational power to train the deep learning models is high and we are balancing with simple model architectures. The respective pre-trained models should be downloaded, and all the packages should be installed.

**5.2 IMPLEMENTATION OF KEY FUNCTIONS**

The Key functions that are to be noticed, processed, and implemented are:

1. Accessing and reading Image (FER-2013)
2. Preparing the Image Data and processing it.
3. Extracting features from images.
4. Evaluating the model using pre-trained caffe models.
5. Developing a deep learning-based CNN Model.
6. Training the model with CNN and existing caffe models.
7. Evaluate the trained model with test dataset.
8. Creating a web-app and good User Interface.

**5.1 INTRODUCTION**

Face recognition is a deep learning-based model which contains complex architectures and corresponding weights that are to be trained for better performance and predictions. Integrating different functions is one of the crucial things in training the model because each layer should be compatible with the next corresponding layers. Choosing a right Optimizer and loss function makes the model performance better in short period of time. The computational power to train the deep learning models is high and we are balancing with simple model architectures. The respective pre-trained models should be downloaded, and all the packages should be installed.

**5.2 IMPLEMENTATION OF KEY FUNCTIONS**

The Key functions that are to be noticed, processed, and implemented are:

1. Accessing and reading Image (FER-2013)
2. Preparing the Image Data and processing it.
3. Extracting features from images.
4. Evaluating the model using pre-trained caffe models.
5. Developing a deep learning-based CNN Model.
6. Training the model with CNN and existing caffe models.
7. Evaluate the trained model with test dataset.
8. Creating a web-app and good User Interface.

**5.3 METHOD OF IMPLEMENTATION**

**Model Code:**

from keras.preprocessing.image import ImageDataGenerator

from keras.models import Sequential

from keras.layers import Dense,Dropout,Flatten

from keras.layers import Conv2D,MaxPooling2D

import os

from matplotlib import pyplot as plt

import numpy as np

IMG\_HEIGHT=48

IMG\_WIDTH = 48

batch\_size=32

train\_data\_dir='C:/Users/THANMAY\_PC/Desktop/fine/Final/Dataset/train/'

validation\_data\_dir='C:/Users/THANMAY\_PC/Desktop/fine/Final/Dataset/test/'

train\_datagen = ImageDataGenerator(

rescale=1./255,

rotation\_range=30,

shear\_range=0.3,

zoom\_range=0.3,

horizontal\_flip=True,

fill\_mode='nearest')

validation\_datagen = ImageDataGenerator(rescale=1./255)

train\_generator = train\_datagen.flow\_from\_directory(

train\_data\_dir,

color\_mode='grayscale',

target\_size=(IMG\_HEIGHT, IMG\_WIDTH),

batch\_size=batch\_size,

class\_mode='categorical',

shuffle=True)

validation\_generator = validation\_datagen.flow\_from\_directory(

validation\_data\_dir,

color\_mode='grayscale',

target\_size=(IMG\_HEIGHT, IMG\_WIDTH),

batch\_size=batch\_size,

class\_mode='categorical',

shuffle=True)

#Verify our generator by plotting a few faces and printing corresponding labels

class\_labels=['Angry','Disgust', 'Fear', 'Happy','Neutral','Sad','Surprise']

img, label = train\_generator.\_\_next\_\_()

import random

i=random.randint(0, (img.shape[0])-1)

image = img[i]

labl = class\_labels[label[i].argmax()]

plt.imshow(image[:,:,0], cmap='gray')

plt.title(labl)

plt.show()

##########################################################

###########################################################

# Create the model

model = Sequential()

model.add(Conv2D(32, kernel\_size=(3, 3), activation='relu', input\_shape=(48,48,1)))

model.add(Conv2D(64, kernel\_size=(3, 3), activation='relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.1))

model.add(Conv2D(128, kernel\_size=(3, 3), activation='relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.1))

model.add(Conv2D(256, kernel\_size=(3, 3), activation='relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.1))

model.add(Flatten())

model.add(Dense(512, activation='relu'))

model.add(Dropout(0.2))

model.add(Dense(7, activation='softmax'))

model.compile(optimizer = 'adam', loss='categorical\_crossentropy', metrics=['accuracy'])

print(model.summary())

train\_path = "C:/Users/THANMAY\_PC/Desktop/fine/Final/Dataset/train"

test\_path = "C:/Users/THANMAY\_PC/Desktop/fine/Final/Dataset/test"

num\_train\_imgs = 0

for root, dirs, files in os.walk(train\_path):

num\_train\_imgs += len(files)

num\_test\_imgs = 0

for root, dirs, files in os.walk(test\_path):

num\_test\_imgs += len(files)

epochs=50

history=model.fit(train\_generator,

steps\_per\_epoch=num\_train\_imgs//batch\_size,

epochs=epochs,

validation\_data=validation\_generator,

validation\_steps=num\_test\_imgs//batch\_size)

model.save('emotion\_detection\_model\_100epochs.h5')

#plot the training and validation accuracy and loss at each epoch

loss = history.history['loss']

val\_loss = history.history['val\_loss']

epochs = range(1, len(loss) + 1)

plt.plot(epochs, loss, 'y', label='Training loss')

plt.plot(epochs, val\_loss, 'r', label='Validation loss')

plt.title('Training and validation loss')

plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.legend()

plt.show()

acc = history.history['accuracy']

#acc = history.history['accuracy']

val\_acc = history.history['val\_accuracy']

#val\_acc = history.history['val\_accuracy']

plt.plot(epochs, acc, 'y', label='Training acc')

plt.plot(epochs, val\_acc, 'r', label='Validation acc')

plt.title('Training and validation accuracy')

plt.xlabel('Epochs')

plt.ylabel('Accuracy')

plt.legend()

plt.show()

####################################################################

from keras.models import load\_model

#Test the model

my\_model = load\_model('emotion\_detection\_model\_100epochs.h5', compile=False)

#Generate a batch of images

test\_img, test\_lbl = validation\_generator.\_\_next\_\_()

predictions=my\_model.predict(test\_img)

predictions = np.argmax(predictions, axis=1)

test\_labels = np.argmax(test\_lbl, axis=1)

from sklearn import metrics

print ("Accuracy = ", metrics.accuracy\_score(test\_labels, predictions))

#Confusion Matrix - verify accuracy of each class

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(test\_labels, predictions)

#print(cm)

import seaborn as sns

sns.heatmap(cm, annot=True)

class\_labels=['Angry','Disgust', 'Fear', 'Happy','Neutral','Sad','Surprise']

#Check results on a few select images

n=random.randint(0, test\_img.shape[0] - 1)

image = test\_img[n]

orig\_labl = class\_labels[test\_labels[n]]

pred\_labl = class\_labels[predictions[n]]

plt.imshow(image[:,:,0], cmap='gray')

plt.title("Original label is:"+orig\_labl+" Predicted is: "+ pred\_labl)

plt.show()

**Front end:**

!pip install opencv-python

import numpy as np

from keras.models import load\_model

import cv2

from keras.preprocessing import image

from tensorflow.keras.utils import img\_to\_array

from time import sleep

def faceBox(faceNet,frame):

frameHeight=frame.shape[0]

frameWidth=frame.shape[1]

blob=cv2.dnn.blobFromImage(frame, 1.0, (300,300), [104,117,123], swapRB=False)

faceNet.setInput(blob)

detection=faceNet.forward()

bboxs=[]

for i in range(detection.shape[2]):

confidence=detection[0,0,i,2]

if confidence>0.7:

x1=int(detection[0,0,i,3]\*frameWidth)

y1=int(detection[0,0,i,4]\*frameHeight)

x2=int(detection[0,0,i,5]\*frameWidth)

y2=int(detection[0,0,i,6]\*frameHeight)

bboxs.append([x1,y1,x2,y2])

cv2.rectangle(frame, (x1,y1),(x2,y2),(0,255,0), 1)

return frame, bboxs

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"

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

ageNet=cv2.dnn.readNet(ageModel,ageProto)

genderNet=cv2.dnn.readNet(genderModel,genderProto)

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']

padding=20

face\_classifier = cv2.CascadeClassifier(r'haarcascade\_frontalface\_default.xml')

emotion\_model =load\_model(r'model.h5')

class\_labels=['Angry','Disgust', 'Fear', 'Happy','Neutral','Sad','Surprise']

cap=cv2.VideoCapture(0)

while True:

ret,frame=cap.read()

labels=[]

gray=cv2.cvtColor(frame,cv2.COLOR\_BGR2GRAY)

faces=face\_classifier.detectMultiScale(gray,1.3,5)

for (x,y,w,h) in faces:

roi\_gray=gray[y:y+h,x:x+w]

roi\_gray=cv2.resize(roi\_gray,(48,48),interpolation=cv2.INTER\_AREA)

#Get image ready for prediction

roi=roi\_gray.astype('float')/255.0 #Scale

roi=img\_to\_array(roi)

roi=np.expand\_dims(roi,axis=0) #Expand dims to get it ready for prediction (1, 48, 48, 1)

preds=emotion\_model.predict(roi)[0] #Yields one hot encoded result for 7 classes

label=class\_labels[preds.argmax()] #Find the label

label\_position=(x+10,y-70)

cv2.putText(frame,label,label\_position,cv2.FONT\_HERSHEY\_SIMPLEX,1,(0,255,0),2)

frame,bboxs=faceBox(faceNet,frame)

for bbox in bboxs:

# face=frame[bbox[1]:bbox[3], bbox[0]:bbox[2]]

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=cv2.dnn.blobFromImage(face, 1.0, (227,227), MODEL\_MEAN\_VALUES, swapRB=False)

genderNet.setInput(blob)

genderPred=genderNet.forward()

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

ageNet.setInput(blob)

agePred=ageNet.forward()

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

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

cv2.rectangle(frame,(bbox[0], bbox[1]-30), (bbox[2], bbox[1]), (0,255,0),-1)

cv2.putText(frame, label, (bbox[0], bbox[1]-10), cv2.FONT\_HERSHEY\_SIMPLEX, 0.8, (255,255,255), 2,cv2.LINE\_AA)

cv2.imshow("Emotion-Age-Gender",frame)

k=cv2.waitKey(1)

if k==ord('q'):

break

cap.release()

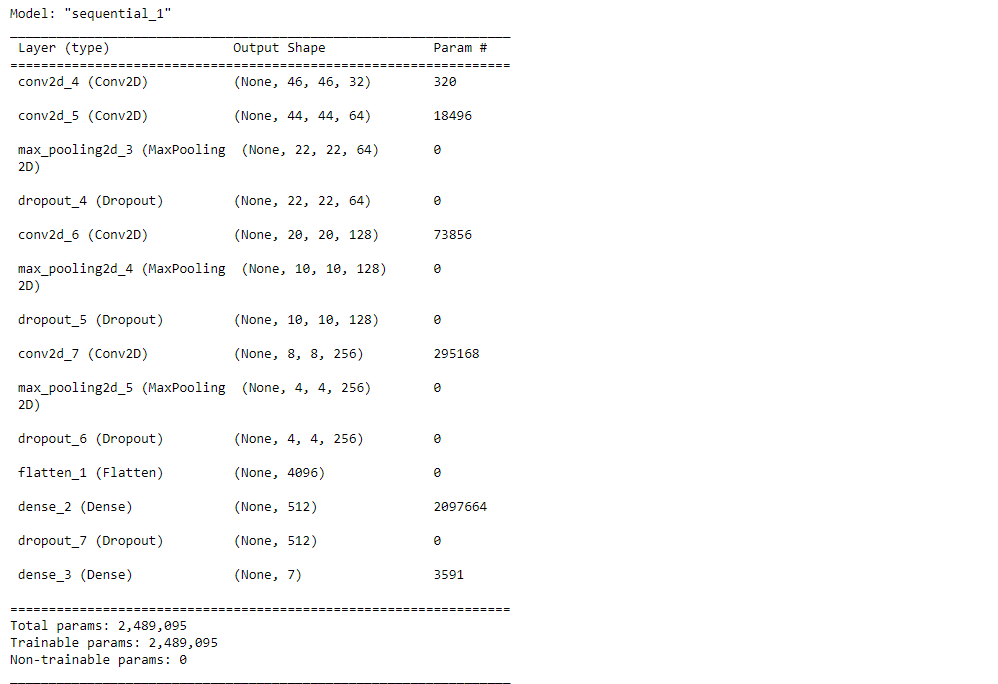
cv2.destroyAllWindows()

**5.4 OUTPUT SCREENS AND RESULT ANALYSIS**

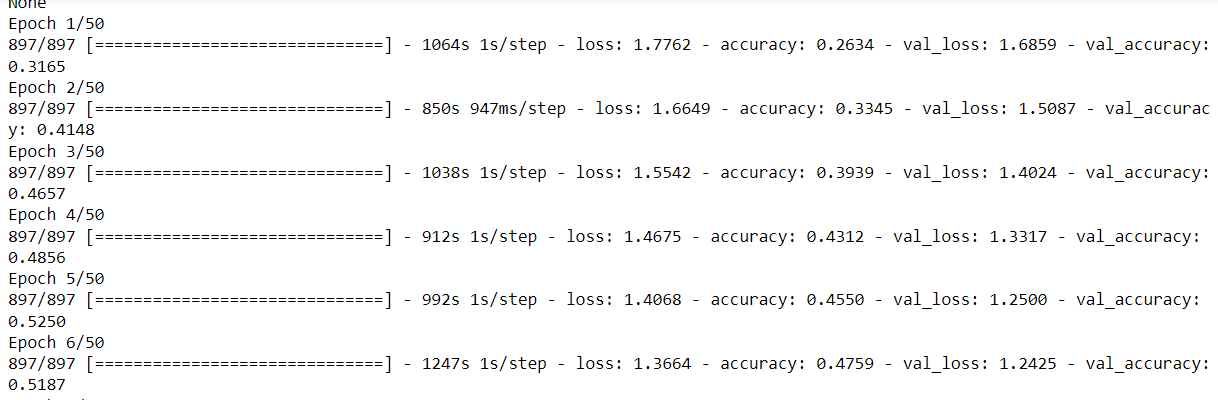
**A picture containing background pattern

Description automatically generated**

**Model Design: (5.4.1)**

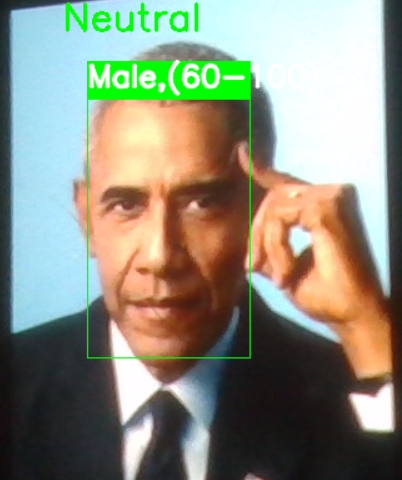


**Model Design: (5.4.2)**



**Model Design: (5.4.3)**

**OUTPUTS:**

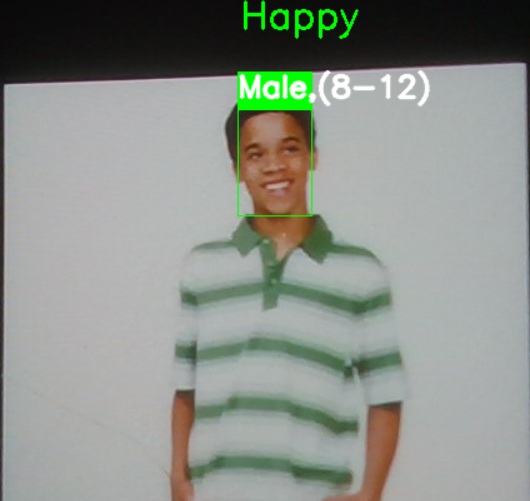
****

**OUTPUT: (5.4.4)**

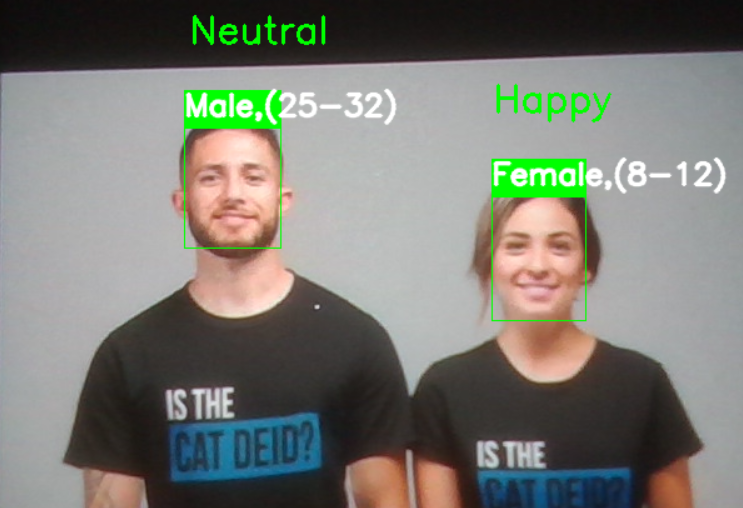
**A picture containing text, person, person, smiling

Description automatically generated**

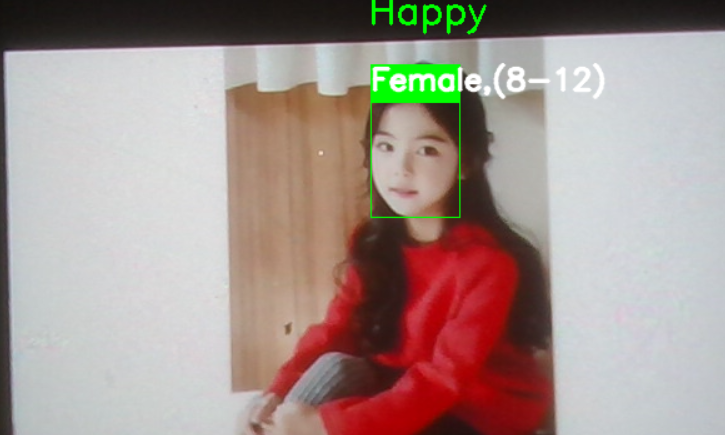
**OUTPUT: (5.4.5)**

****

**OUTPUT: (5.4.6)**

****

**OUTPUT: (5.4.7)**

****

**OUTPUT: (5.4.8)**

**CONCLUSION:**

We have proposed and tested a general building designs for creating real-time CNNs. Our proposed architectures have been systematically built in order to reduce the number of parameters as much as possible. We have shown that our proposed models can be stacked for multiclass classification while maintaining real-time inferences. In conclusion, we’ve successfully constructed a working CNN model to recognize the Facial Expressions, Age and Gender of Human Beings.

**CHAPTER-6**

**TESTING AND VALIDATION**

**6.1 INTRODUCTION**

**INTRODUCTION TO TESTING**

Testing is a process, which reveals the errors in the program. It is the major quality measure employed during software development process. During testing, the program is executed with a set of test cases and the output of the program for the test cases is evaluated to determine if the program is performing as it is expected to perform or not.

In order to make sure that the system does not have errors, the different levels of testing strategies that are applied at differing phases of software development.

**UNIT TESTING**

The purpose is to validate that each unit of the software performs as designed.

A unit is the smallest testable part of software. Unit Testing is a level of software testing where individual units/components of a software are tested. Typically, the unit test will establish some sort of artificial environment and then invoke methods in the unit being tested. It then checks the results returned against some known value. When the units are assembled, we can use the same tests to test the system as a hole. It usually has one or a few inputs and usually a single output.

**FUNCTIONAL TESTING**

Functional Testing is a testing technique that is used to test the features /functionality of the system or Software, should cover all the scenarios including failure paths and boundary cases.

**INTEGRATION TESTING**

Upon completion of unit testing, the units or modules are to be integrated which gives raise to integration testing. The purpose of integration testing is to verify the functional, performance, and reliability between the modules that are integrated. In the system signup, login, updating values in database are all integrated and tested once so that.

They don’t tend to make errors while integrated.

**SYSTEM TESTING**

System testing of software or hardware is the testing conducted on a complete, integrated system to evaluate the system’s compliance with its specified requirements. The hardware and the software units are tested separately and then tested together to check if the desired results are obtained.

**PERFORMANCE TESTING**

Performance testing, a non-functional testing technique performed to determine the system parameters in terms of responsiveness and stability under various workload. Performance testing measures the quality attributes of the system, such as scalability, reliability, and resource usage for running the application in the system.

**6.3 VALIDATION**

The following test case scenarios were used in the integrated system testing to prove the working of the developed system.

* Encoder Output vectors and their corresponding Shapes.
* Word embeddings and the vector dimensions of the LSTM input.
* Model prediction on Validation and Test data.
* Semantic meaning in the generated captions of the Image.
* Validating the model with irrelevant images of the Train data.
* Checking all the compatibilities of the vectors and input shapes.
* Display of web-based Application and the GUI. All test cases were successfully tested. The system developed is user friendly and no special training is required for caption generation.

**6.4 CONCLUSION**

The proposed model is robust, with low-computational power and does not require any special training. This architecture uses of the existing developments of the pre-trained models and various types of deep learning-based and Natural language techniques. As the whole system is simplified, the scalability of major applications with this model is tremendous.

**CHAPTER-7**

**CONCLUSION**

**7.1 CONCLUSION**

In this work we propose a multi-task learning framework, to simultaneously predict age and gender from face images. Our framework is based on an ensemble of ResNet-based model and an attention-based model. We trained and tested the proposed model on the UTKFace dataset consisting a large variety of faces from different ages, genders and ethnicities. Through experimental studies, we show that the prediction accuracy of the ensemble model (for both age and gender prediction tasks) surpasses in those of the separated models. We also showed that providing the prediction of the gender model as one of the input signal for the age-prediction branch, can improve the accuracy of predicted age values. Through visualization of the attention maps of the trained model, we show that the model learned to focus on the most salient part of the face, useful for predicting age and gender.

**CHAPTER-8**

**REFERENCES**

**REFERENCES**

1. P. Aiswarya, Manish and P. Mangalraj, "Emotion recognition by inclusion of age and gender parameters with a novel hierarchical approach using deep learning," 2020 Advanced Communication Technologies and Signal Processing (ACTS), 2020, pp. 1-6, doi: 10.1109/ACTS49415.2020.9350479.
2. Dehghan, Afshin & G. Ortiz, Enrique & Shu, Guang & Zain Masood, Syed. (2017). DAGER: Deep Age, Gender and Emotion Recognition Using Convolutional Neural Network
3. Bailenson, Jeremy N., et al. "Real-time classification of evoked emotions using facial feature tracking and physiological responses." International journal of human-computer studies 66.5 (2008): 303-317.
4. Rothe, Rasmus, Radu Timofte, and Luc Van Gool. "Dex: Deep expectation of apparent age from a single image." In Proceedings of the IEEE International Conference on Computer Vision Workshops, pp. 10-15. 2015.
5. Levi, G., Hassner., T.: Age and gender classification using convolutional neural networks. In: CVPRW. (2015)