International Journal for Modern Trends in Science and Technology, 8(07): 248-254, 2022

ISSN: 2455-3778 online
DOI: https://doi.org/10.46501/IJMTST0807036
Available online at: http://www.ijmtst.com/vol8issue07.html

Real-Time Face Emotion Recognition Using Convolutional Neural Network

Omkar Janardan Joshi | Rohini Nair

Department of Computer Engineering, KJ Somaiya College of Engineering, Mumbai, Maharashtra, India
Email: o.j.joshi@gmail.com,

To Cite this Article
Omkar Janardan Joshi and Rohini Nair. Real-Time Face Emotion Recognition Using Convolutional Neural Network. International Journal for Modern Trends in Science and Technology 2022, 8(07), pp. 248-254. https://doi.org/10.46501/IJMTST0807036

Article Info
Received: 16 June 2022; Accepted: 15 July 2022; Published: 20 July 2022.

ABSTRACT

Faces are the most relevant social catalyst as they communicate information that is needed for the course of interaction and communication. Facial expressions convey information about what sentiment is currently experienced by the person, which in turn affects how the person is perceived and what behavioral tendencies are drawn out from the observer. Face emotion recognition is a process to find the sentiment of the people. The purpose of emotion recognition systems is to detect emotion-related knowledge in such a way that human-computer communication will be enhanced which in turn will make the user's experience more satisfying. This system helps computers sense the user's emotional state and react accordingly. Refining communication with computers is not the only application of emotion recognition; specialized systems can be developed and used for even more serious problems in various medical applications: aggression detection, stress detection, autistic disorder, frustration detection, etc. Face emotion recognition involves categorizing facial emotions into different classes like Anger, Sad, Happy, Fear, Neutral, Surprise, and Disgust. In this thesis titled "Real-Time Face Emotion Recognition Using Convolutional Neural Network", we have implemented the deep learning technique of Convolutional neural network on the Facial Expression Recognition 2013 (FER2013) dataset. These images are first divided into training and testing datasets, normalization is applied and finally, Convolution layers are applied. The evaluation metrics are considered by the percentage accuracy of the training and testing model. Webcam is attached to the trained model and Open CV is used to detect facial expressions in real-time.

KEYWORDS: Convolutional neural network, FER2013 dataset, seven facial expressions, Accuracy

1. INTRODUCTION

Facial expression recognition is a technology that is used by computers to detect sentiments in human faces. More precisely, this technology is a sentiment analysis tool and it can automatically detect the basic universal expressions.

Different cultures have varying expectations for when it's appropriate to use a certain expression and in what manner. In the US, for example, it is common to smile at strangers. However, in other parts of the world, frequent and overeager smiling is not so well received. Some of the studies also say that facial sentiments are a key factor to find someone's feelings. So, considering all these above aspects it becomes tough for computers to classify the true sentiment of the human faces. Although a lot of research has been done to develop the FER system, we find that several problems still exist in the present environment which hamper the development of

the FER system, like the extracted features are delicate to the change in lighting conditions, also the noise in the image influence the recognition accuracy. Also, an inadequate dataset is another problem that degrades the performance of such systems.

2. RELATED WORK

The face detection process is applied to some images having specific expressions, by applying the Viola-Jones algorithm to find face positions on the images and crop them for creating new image files,Then, the CNN model was trained. The maximum number of errors occurred between fear and surprise classes because of the similarities in the features of fear and surprise facial emotions [1]. In [2], a Shallow network with 1 Convolutional layer and a Deep network with 8 layers are compared.It was found that for shallow network, the training accuracy was less compared to deep network.In [3], a lightweight neural network model was created along with a Haar cascade classifier to detect face information from the image.In a lightweight neural network, the concept of depth separable convolution was used to reduce the number of training parameters in convolution operation, reducing the model's complexity. In [4], The process proposed had three stages. The preprocessing stage; consists of preparing the dataset into a form that will work on a generalized algorithm and generate efficient results. In the second face detection stage, the face is detected from the images that are captured in real-time. And finally in the third stage emotion classification is implemented using the CNN algorithm to classify input images into one of seven classes. There are several datasets available for research in the field of Facial Expression Recognition, such as the Japanese Female Facial Expressions (JAFFE), Extended Cohn Kanade dataset (CK+), and the FER2013 dataset. The type and number of images and the method of labeling the images vary in each dataset. There are several challenges with implementing the FER system. Most datasets consist of images of posed people with a certain expression. This is the first challenge, as real-time applications require a model with expressions that are not posed or directed. The second challenge is that the labels in the datasets are broadly classified, which means that in real-time there might be some expressions that the system might be able to classify correctly.

[5] Convolutional Neural Network is a specific brain network for handling information that has an info shape like a 2D grid-like picture. CNNs are commonly utilized for picture recognition and grouping. Pictures are 2D framework of pixels on which we run CNN to either perceive the picture or to arrange the picture. Recognize if a picture is of a person, vehicle, or only digits on a location. Like Neural Networks, CNN additionally draws inspiration from the mind. Max_Pooling_1**:** This layer is used to reduce the input image size. **Kernal size = (2,2)** used here. So input image 48 is reduced to half 24. And model learns nothing from this layer. [6] Early Stopping: Too many epochs can lead to overfitting of the training dataset, whereas too few may result in an underfit model. Early stopping is a method that allows you to specify an arbitrarily large number of training epochs and stop training once the model performance stops improving on a holdout validation dataset.

[7] Learning rate scheduler: The simplest and most used adaptation of learning rate during training are techniques that reduce the learning rate over time. These have the benefit of making large changes at the beginning of the training procedure when larger learning rate values are used and decreasing the learning rate such that a smaller rate and therefore smaller training updates are made to weights later in the training procedure. This has the effect of quickly learning good weights early and fine-tuning them later. Two popular and easy-to-use learning rate schedules are as follows:

- Decrease the learning rate gradually based on the epoch.
- Decrease the learning rate using punctuated large drops at specific epochs.

3. EXPERIMENT

In this project, we have constructed Convolutional Neural Network with 17 layers. Different testing with multiple layers like 10 layers, 13 layers, 15 layers, and finally with 17 layers was done. Along with all the different trials with the layers, we have also tuned their hyperparameters in each test, like 10 layers were tested with batch sizes of 32, 64, 128, and 256 with a kernel size

of 5*5 and 3*3 on each batch size. A total of 8 combination testing was done on 10 layers. Similarly on 13 layers, 15 layers, but for 17 layers we were only able to do testing with 3*3 kernel size a with a batch size of 32,64,128, and 256. i.e., only 4 testing were done due to GPU limitations in Google Colab. All the training is done on Google Colab by enabling GPU resources provided by them. Figure 1 depicts the testing done on multiple layers.

The input dataset taken in our model is the FER2013 dataset. The FER2013 database (Face Emotion Recognition 2013) is a large database of human faces with different expressions like happy, sad, neutral, surprise, disgust, anger, and fear which is commonly used for training in various image processing systems. From FER2013 dataset we have taken 28272 training images and 7068 testing images. The images in the FER2013 dataset are present in form of an array consisting of 48x48 values representing an image along with their labels.

During the training phase of the CNN, each epoch produces up to three plots (time, loss, and accuracy). We have tried different combinations of batch size, convolution layers, and kernel size. So, found out that the training time increases as we increase the Convolution layers. Increasing the epochs, we can get good accuracy but as the number of epochs increases, after some point, there is a chance for overfitting. So, we have implemented up to 100 epochs. But still, we face overfitting; [8] so, we have used early_stopping and lr_scheduler (Reduce learning rate on Plateau) to avoid overfitting and the training will stop early i.e., as soon as the model performance stops improving on the testing dataset. We have used a batch size of 32, 64, 128, and 256.

In the results of training and evaluating the model, we can say that time taken to execute one epoch is nearly 15-95 seconds (with Google Colab GPU) for different batch sizes = 32, 64, 128, 256 number of classes = 7, I.e., it takes nearly 27 minutes – 2 hrs. to train the model with Google Colab GPU. The model trained with 17 Convolution layers with a batch size of 128 and kernel size 3*3 gives maximum training accuracy of 90.24% and training accuracy of 69.36%. So, this model is trained on Google Colab with GPU enabled with 100 epochs and its h5 model is saved. This h5 model saved is then further attached to our Open CV code which is written in Jupiter notebook and the webcam is attached to it. The webcam will be able to classify 7 different expressions like happy, sad, angry, disgust, neutral, surprise, and fear with probabilities of every expression will be shown in another tab.

	Batch Size	No. of epoch	Kernel Size	Training Accuracy	Testing Accuracy	Time taken to train
10 Convolution layers	32	100(87)	3*3	72.07%	68.54%	35 seconds per epoch (55 minutes.)
		100(70)	5*5	74.95%	68.90%	40 seconds per epoch (47 minutes.)
	64	100(89)	3*3	73.24%	68.40%	31 seconds per epoch (46 minutes.)
		100(93)	5*5	78.53%	69.81%	17 seconds per epoch (27 minutes.)
	128	100(74)	3*3	71.74%	67.80%	25 seconds per epoch (33 minutes.)
		100(94)	5*5	79.54%	68.50%	17 seconds per epoch (27 minutes.)
	256	100(93)	3*3	71.47%	68.11%	23 seconds per epoch (37 minutes.)
		100	5*5	76.97%	68.88%	15 seconds per epoch (25 minutes.)
	Batch Size	No. of epoch	Kernel Size	Training Accuracy	Testing Accuracy	Time taken to train
13 Convolution layers	32	100(95)	3*3	74.27%	69.32%	20 seconds per epoch (33 minutes.)
		100(78)	5*5	74.82%	68.61%	26 seconds per epoch (34 minutes.)
	64	100(89)	3*3	74.91%	69.69%	17 seconds per epoch (27 minutes.)
		100	5*5	79.45%	69.55%	20 seconds per epoch (34 minutes.)
	128	100(95)	3*3	76.79%	68.99%	27 seconds per epoch (44 minutes.)
		100(95)	5*5	79.40%	68.91%	35 seconds per epoch (1 hr.)
	256	100(81)	3*3	73.34%	67.86%	15 seconds per epoch (21 minutes.)
		100(72)	5*5	75.26%	68.12%	16 seconds per epoch (20 minutes.)
	Batch Size	No. of epoch	Kernel Size	Training Accuracy	Testing Accuracy	Time taken to train
15 Convolution layers	32	100(95)	3*3	72.90%	68.83%	55 seconds per epoch (1.5 hrs.)
		100(87)	5*5	76.82%	69.45%	40 seconds per epoch (1 hrs.)
	64	100(89)	3*3	74.69%	68.78%	40 seconds per epoch (1 hr.)
		100(62)	5*5	74.56%	68.70%	28 seconds per epoch (30 minutes.)
	128	100(93)	3*3	75.78%	68.74%	30 seconds per epoch (50 minutes.)
		100(94)	5*5	78.54%	68.79%	22 seconds per epoch (35 minutes.)
	256	100	3*3	74.71%	68.19%	16 seconds per epoch (25 minutes.)
		100(92)	5*5	76.75%	68.86%	45 seconds per epoch (1.2 hrs.)
	Batch Size	No. of epoch	Kernel Size	Training Accuracy	Testing Accuracy	Time taken to train
17 Convolution layers	32	100(75)	3*3	83.48%	69.58%	95 seconds per epoch (2 hrs.)
	64	100(95)	3*3	88.98%	69.98%	35 seconds per epoch (1hr.)
	128	100(97)	3*3	90.24%	69.36%	30 seconds per epoch (49 minutes.)
	256	100(88)	3*3	86.64%	69.81%	71 seconds per epoch (2 hrs.)

Fig. 1. Testing done on different layers

4. Architecture Of the Model

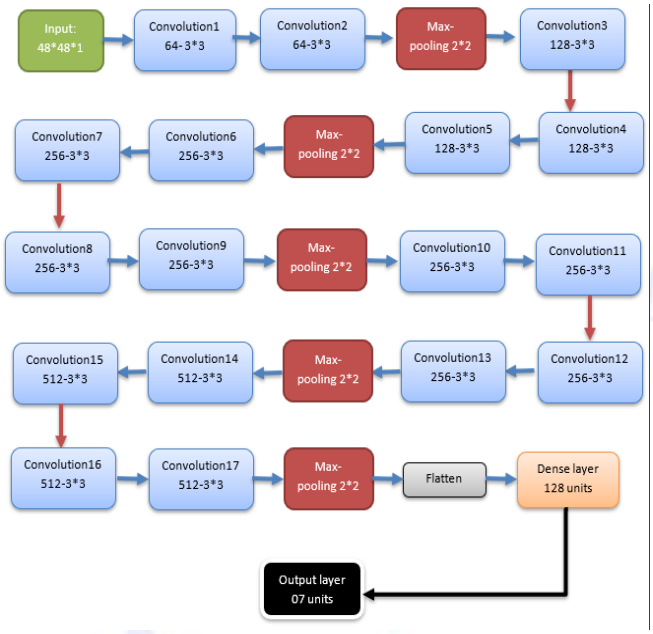

Fig. 2. Architecture of the model

The implementation of this project is based on the architecture shown above in figure 2. [9] The pictures in the FER2013 dataset are available in the type of exhibit comprising 48x48 qualities addressing a picture alongside their names. This input image is then pre-processed by reshaping the matrix to shape (28272,48,48,1) where 28272 is the number of images taken from the dataset, (48,48) is the dimensions of each image, and 1 represents the number of channels. As the images in the dataset are grayscale, it will contain 1 channel, but if a dataset contains images that are in RGB then it will contain 3 channels. Then normalization is applied to it.

A total of 17 convolutional layers are applied with 3*3 kernel size and different numbers of filters in different layers. Max-Pooling of 2*2 is applied in between to reduce the input image size from previous layers. [10] Batch Normalization is applied after every step to normalize the output of the previous layer and it allows every layer of the network to do learning more independently. It also makes learning more efficient and helps to avoid overfitting. A dropout of 0.3 is applied to avoid further overfitting. Then finally, a fully-connected layer is applied which will be helpful for computing the class score which leads us to the final volume of 1*1*7 where 7 represents the categories of the FER2013 dataset.

As we know, by applying too many epochs, our model can go to overfitting and by applying a few numbers of epochs, the model may go underfitting. So, we have used 100 epochs for training and used the early-stopping function which will help us to stop the training once the model performance stops improving. This will help us to reduce the training time and fewer resources will be used.

Table I, shows the calculations done in our model

Table- I: Model Summary Calculations

Layer Names	Shape	Calculation	Parameters
Convolution 1	64 filters, 3*3filter size	((3*3)+1)*64	640
Batch Normalization1	64 filters	64*4	256
Convolution 2	64 filters, 3*3 filter size	((64*3*3*64) +64)	36928
Batch Normalization2	64 filters	64*4	256
Max Pool 1	2*2	-	-
Dropout 1	0.3	-	-
Convolution 3	128 filters, 3*3 filter size	((128*3*3*64) +128)	73856
Batch Normalization3	128 filters	128*4	512
Convolution 4	128 filters, 3*3 filter size	((128*3*3*128) +128)	147584
Batch Normalization4	128 filters	128*4	512
Convolution 5	128 filters, 3*3 filter size	((128*3*3*128) +128)	147584
Batch Normalization5	128 filters	128*4	512
Max Pool 2	2*2	-	-
Dropout 2	0.3	-	-
Convolution 6	256 filters, 3*3 filter size	((256*3*3*128) +256)	295168
Batch Normalization6	256 filters	256*4	1024
Convolution 7	256 filters, 3*3 filter size	((256*3*3*256) +256)	590080
Batch Normalization7	256 filters	256*4	1024
Convolution 8	256 filters, 3*3 filter size	((256*3*3*256) +256)	590080
Batch Normalization8	256 filters	256*4	1024
Convolution 9	256 filters, 3*3 filter size	((256*3*3*256) +256)	590080
Batch Normalization9	256 filters	256*4	1024
Max Pool 3	2*2	-	-
Dropout 3	0.3	-	-
Convolution 10	256 filters, 3*3 filter size	((256*3*3*256) +256)	590080

Batch Normaliation10	256 filters	256*4	1024
Convolution 11	256 filters, 3*3 filter size	((256*3*3*256) +256)	590080
Batch Normaliation11	256 filters	256*4	1024
Convolution 12	256 filters, 3*3 filter size	((256*3*3*256) +256)	590080
Batch Normaliation12	256 filters	256*4	1024
Convolution 13	256 filters, 3*3 filter size	((256*3*3*256) +256)	590080
Batch Normaliation13	256 filters	256*4	1024
Max Pool 4	2*2	-	-
Convolution 14	512 filters, 3*3 filter size	((512*3*3*256) +512)	1180160
Batch Normaliation14	512 filters	512*4	2048
Convolution 15	512 filters, 3*3 filter size	((512*3*3*512) +512)	2359808
Batch Normaliation15	512 filters	512*4	2048
Convolution 16	512 filters, 3*3 filter size	((512*3*3*512) +512)	2359808
Batch Normaliation16	512 filters	512*4	2048
Convolution 17	512 filters, 3*3 filter size	((512*3*3*512) +512)	2359808
Batch Normaliation17	512 filters	512*4	2048
Max Pool 5	2*2	-	-
Dropout 4	0.3	-	-
Dense 1	Flatten-512 Dense-128	(512*128)+128	65664
Batch Normaliation18	128 filters	128*4	512
Dropout 5	0.3	-	-
Output Layer	128 filters 7 classifications	(128*7)+7	903

The above table contains different layer names with shape/size and dimensions of the filters used and parameters with their calculations.

5. Real-Time Implementation on Webcam

We trained our model through Google Colab and saved the model file. This model file is then used along with OpenCV in Jupyter Notebook and implemented on a webcam to differentiate different human emotions in real-time. The model successfully recognized different human expressions. First, it detects the human face making a bounding box around each human face, then labels each face with the relevant expressions like happy, sad, neutral, anger, surprise, fear, and disgust. Simultaneously in a new window, the probabilities of each emotion are detected as shown in figure 3.

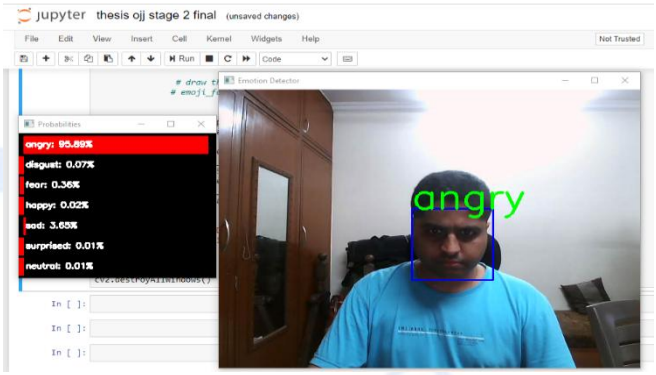

Fig. 3. Angry emotion detected from webcam

In above figure 3, angry expression is detected with probabilities as angry: 95.89%, disgust: 0.07%, fear: 0.36%, happy: 0.02%, sad: 3.65%, surprised and neutral: 0.01%. Similarly, figure 4, figure 5, figure 6, figure7, and figure 8 shows different expressions like fear, happy, neutral, sad, and surprise.

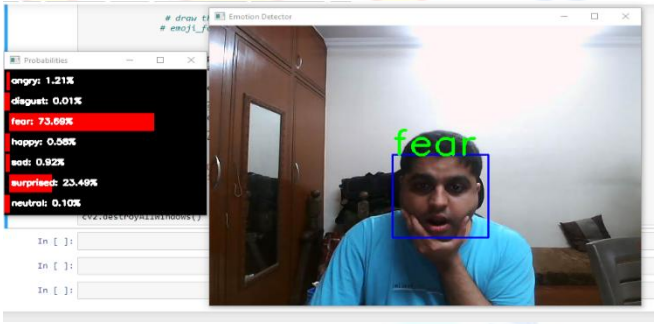

Fig. 4. Fear emotion detected from webcam

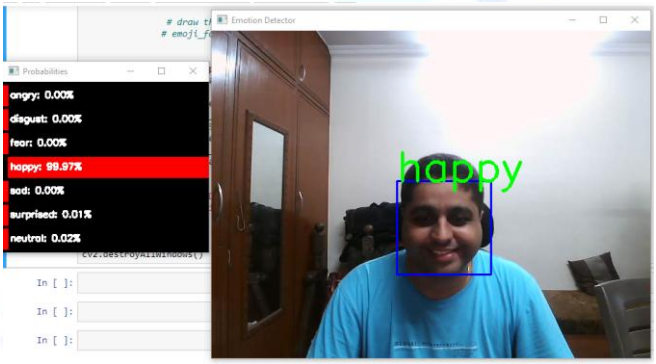

Fig. 5. Happy emotion detected from webcam

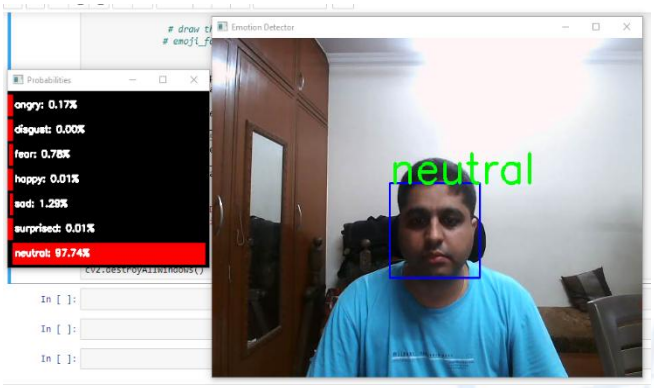

Fig. 6. Neutral emotion detected from webcam

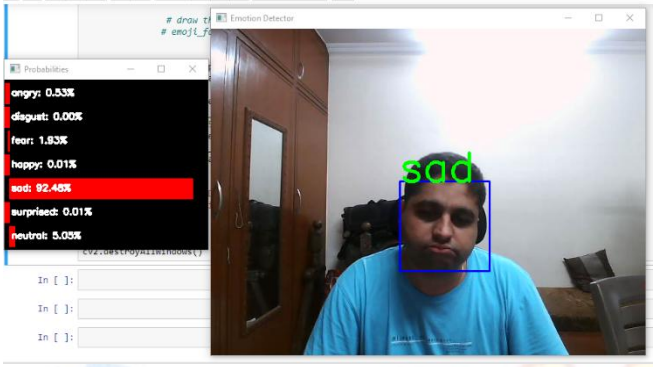

Fig. 7. Sad emotion detected from webcam

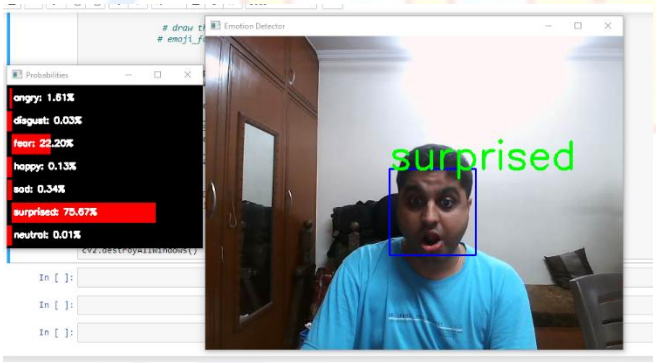

Fig. 8. Surprised emotion detected from webcam

We have also tried to detect the expression from a mobile device and the model is satisfactorily capable to detect the expressions from the too. Figure 9 and figure 10 shows expressions and their probabilities detected from the webcam

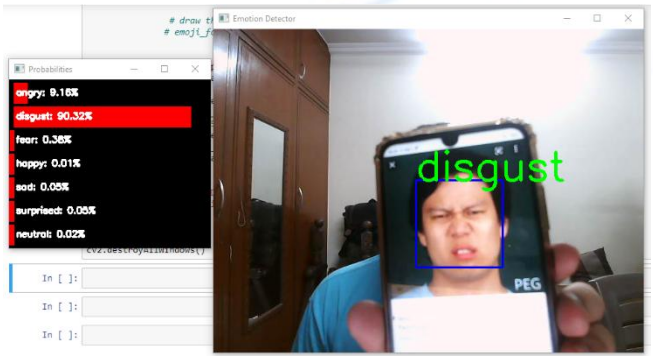

Fig. 9. Disgust emotion detected from mobile device

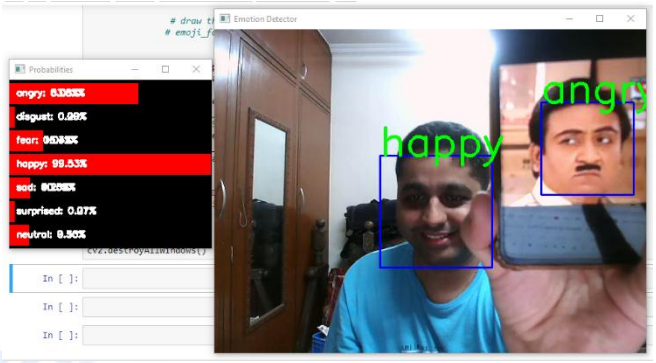

Fig. 10. Multiple emotions detected from webcam

6. Conclusion and Future Work

Here we have demonstrated a model which can classify facial emotions in humans from the FER2013 dataset in real-time. Later, this network could be used as part of the software to detect the emotions of people (in professional meetings) or students (in online classes) in real-time. This capability can be used by machines to improve their interaction with humans by providing more adequate responses. Finally, this is a multidisciplinary project which involves the study and application of affective computing, deep learning, and computer vision. Learning how these different fields are related, and understanding how they can provide solutions to complex problems is another project's goal. As seen from the results of the experiment and from the literature survey done in some paper publications, CNN proves to be far better than other classifiers. Much research effort around the world is being applied to expand the accuracy and capabilities of this biometric domain. But, sometimes due to background, lighting conditions and scale orientation, and Age factor, there is suffering in the recognition system. But by modifying CNN architecture by changing the number of layers, applying computer vision techniques, or by expanding datasets; we can achieve more accuracy.

The results can be made more accurate with more convolution layers and by adding a greater number of hidden neurons. To increase the testing accuracy more, i.e.to avoid misclassification, we can increase the size of the dataset or create our own dataset according to our model. There are many areas for improvement based on our CNN model. Therefore, how to improve the recognition performance of the model remains to be further studied.

Conflict of interest statement

Authors declare that they do not have any conflict of interest.

REFERENCES

[1] Dandıl, E. and Özdemir, R., 2019. Real-time facial emotion classification using deep learning. Data Science and Applications, 2(1), pp.13-17.

[2] Pathar, R., Adivarekar, A., Mishra, A. and Deshmukh, A., 2019, April. Human emotion recognition using convolutional neural network in real time. In 2019 1st International Conference on Innovations in Information and Communication Technology (ICIICT) (pp. 1-7). IEEE.

[3] Li, Q., Liu, Y.Q., Peng, Y.Q., Liu, C., Shi, J., Yan, F. and Zhang, Q., 2021, March. Real-time facial emotion recognition using lightweight convolution neural network. In Journal of Physics: Conference Series (Vol. 1827, No. 1, p. 012130). IOP Publishing.

[4] Talegaonkar, I., Joshi, K., Valunj, S., Kohok, R. and Kulkarni, A., 2019, May. Real time facial expression recognition using deep learning. In Proceedings of International Conference on Communication and Information Processing (ICCIP).

[5] https://en.wikipedia.org/wiki/Convolutional_neural_network

[6] https://machinelearningmastery.com/how-to-stop-training-deep-neural-networks-at-the-right-time-using-early-stopping/#:~:text=Too%20many%20epochs%20can%20lead,a%20hold%20out%20validation%20dataset.

[7] https://machinelearningmastery.com/learning-rate-for-deep-learning-neural-networks/

[8] https://machinelearningmastery.com/how-to-stop-training-deep-neural-networks-at-the-right-time-using-early-stopping/

[9] https://www.kaggle.com/msambare/fer2013

[10] https://www.analyticsvidhya.com/blog/2021/03/introduction-to-batch-normalization/