Emotion Detection Using Facial Expression on Real-time Webcam Data

Tina Gholami

Electrical and Computer Engineering Department, University of Western Ontario
London, Canada
tgholam2@uwo.ca

***Abstract*— In this paper, emotion detection is studied on real-time webcam streaming data, using both facial expression recognition and image processing technics. Then, two models are proposed and later compared for their performance evaluation. Finally, a demo of the best model's performance is shown.**

***Keywords*— Feeling detection, real-time data, Facial Expression recognition, Image processing**

I. Introduction

Detecting emotions through sensory data has recently gained popularity and is being used in marketing and health sectors. This paper aims to detect people's feelings through their facial expression. More accurately, it tries to analyse webcam video data, predict, and label an emotion tag based on the person's facial expression at any given time. Detecting emotions with computers is a pretty challenging task, yet machine learning algorithms have shown great promises. Using facial emotion recognition and customer sentiments, businesses can access their customer's feedback more accurately within each use of the product. These results can then help companies develop higher quality products or even improve their customer services by directly contacting the unsatisfied customers for their suggestions or even refunds. Such customer support can significantly boost a business. Furthermore, health centres can keep track of a patient's state of mind and check if abnormalities happen with their psychotic behaviour. Moreover, legislative sectors can also take advantage of this concept in lie detectors to find how a criminal feels and if he is trying to deceive the police. Thus, it is undeniable how valuable emotion detection can be in helping businesses and the society [1]. This paper uses convolutional neural networks (CNN) in conjunction with image processing technics to predict emotions from streaming data using a model that was trained on the Kaggle contest dataset [2] for facial expression recognition. First, the CNN model is trained to predict emotions using grey scale images. Then with the help of image processing and face recognition, webcam data are used as input for the model to detect the user's feelings based on their facial expressions. This paper is will discuss some background information, then it will inspect related works in the past. Next, it delves into the methodology and model training. After that, results are presented. Finally, impediment and suggestions are highlighted before final conclusions as well as the references.

II. Background

In this section, some background information on the methods used, is provided:

A. Convolutional Neural Networks (CNNs)

Convolutional neural networks or CNNs are a class of deep neural networks commonly applied to image data. The architecture of these networks consists of an input and an output layer. In between, there are convolutional layers that convolve the input images with specific kernels (filters) to abstract the data into a feature map. There are also dropout layers to help combat overfitting by randomly subsampling the data at the corresponding layers. Then, there are max pooling layers to further abstract the data into smaller matrices in order to recognize more vital features in each input matrix data. After repeating a desired number of such layers in order, finally the data should be flattened in a 1D vector to be capable of being fed into a fully connected feed forward neural network (FFNN).

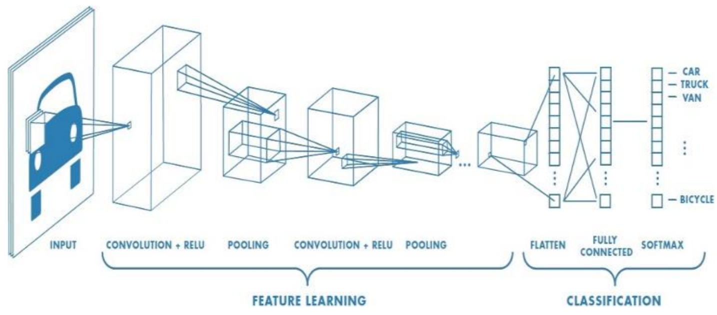


Fig. 1 CNN model architecture [1]

B. Fully Connected Feed Forward Neural Network (FFNN)

As can be seen in the picture bellow, the FFNN model is basically an artificial neural network with all the neurons being fully connected to the neurons in the previous and the next layers. The reason for such intensive connections is to

[1] https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148

best show and employ the subtle features found in the CNN model to be able to predict the output for a set of new test data, even if they are totally distinctive from the train data. In fact, the goal is to avoid wasting the vital features found by the CNN model.

C. Accuracy Score Metric:

In any machine learning application, after the model is trained, its performance should be evaluated on the test data, using specific metrics. The metrics used in this paper include accuracy, precision, recall, and f1-score. Accuracy is calculated as:

$$Accuracy = \frac{True\ positive\ (TP) + True\ negative\ (TN)}{Total\ samples}$$

Where True positive (TP) refers to the number of predictions where the classifier correctly predicts the positive class as positive. And True negative (TN) refers to the number of predictions where the classifier correctly predicts the negative class as negative. Accuracy helps in understanding how well the model's performance is. But one has to beware of a common issue with the accuracy score, which happens when the data is imbalance. In such cases, one ought to employ other metrics

D. Precision Metric:

Precision is about how precise our model is from the predicted positives, meaning how many of the predicted positives are actual positives i.e. the percentage of the results which are relevant. Precision is a good measure to determine if the costs of false positives are high and it is calculated as the following relation:

$$Precision = \frac{True\ positive\ (TP)}{True\ positive\ (TP) + False\ positive\ (FP)}$$

Where false positive (FP) refers to the number of predictions where the classifier wrongly predicts the negative class as positive.

E. Recall Metric:

Recall refers to the percentage of total relevant results correctly classified by the algorithm and it is calculated as the following relation:

$$Recall = \frac{True\ positive\ (TP)}{True\ positive\ (TP) + False\ negative\ (FN)}$$

Where false negative (FN) refers to the number of predictions where the classifier wrongly predicts the positive class as negative.

F. F1 Metric:

F1 score takes into account both precision and recall, and therefore, it can get maximized to improve the model. In fact, it is simply the harmonic mean of precision and recall and is calculated as the following relation:

$$ecall = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

III. RELATED WORKS

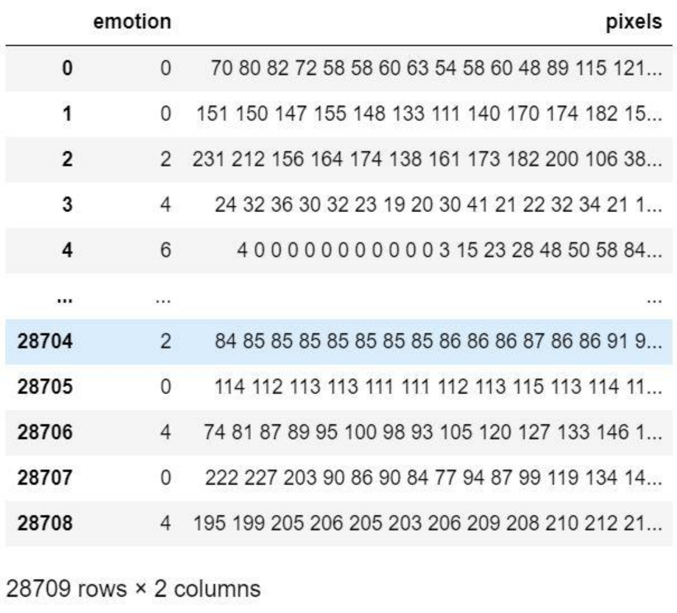
There are numerous works and literature on extracting information solely from image data. For instance, [3] tries to detect the emotions evoked by paintings using CNN, VGGnet-16, and ResNet-50. Interestingly, the CNN model's accuracy is around 40%, whereas the other two algorithms yield 56% and 53% respectively. There is a similar research in [4] as well. In another work [5] sentiment analysis was conducted on image data on the web using deep learning, namely CNN, Region-based CNN (R-CNN), and Fast RNN. This study also discusses the challenges and perspectives of the rising field of sentimental analysis using image data. In another paper [6] image emotions are detected using deep neural network (DNN) as building the so called "emotional machine". Moreover, in another student project report paper [7] from Stanford university on a similar problem for steaming data, CNN model was implemented and the maximum accuracy on the train set was around 60%. As compared, the CNN model alone at best, will yield an accuracy around 50% on the test data. But in order to improve this accuracy, we either have to have more complex models such as those used in computer vision, or we can try to pre-process the data more efficiently. In the current paper, however, stream data (webcam videos) are used as test data (as appose to images) and then fed into a CNN model which was trained by grey scale images. This approach is different from the other works since the testing phase is not only conducted on images again, but also on webcam frames. The CNN model in this paper has two versions, model version I is similar to that of [7], whereas model version II is designed to be more complex and to have different filter sizes on consecutive layers in order to detect main features differently at each layer. In fact, the CNN in model version II is more complex than the CNN in model version I. However, the FFNN in model version II is simpler and has less neurons compared to the FFNN in model version I.

IV. METHODOLOGY

In general, the methodology that was followed in this paper was to first train the CNN model (both model version I and II) on the input grey scale images from Kaggle contest dataset [2] for facial expression recognition. And then, use this model to predict the emotion in webcam data frames as being streamed with the help of image processing technics. Furthermore, before feeding in the webcam data into the model, the facial part of the streaming data must be the test input for the model, not the whole frame. Hence, the face of the user should first be detected, and then the model will assign the appropriate emotion tag to it. In what follows, first, a description of the model preparation and training will be discussed. Two different architectures for the CNN model are proposed. Then, the model is employed on the webcam data stream using image processing technics. Next, the performance of these two models (CNN version I and CNN version II) are compared using metrics accuracy, precision, recall and f1 score.

A. Dataset Characteristics:

The data consists of 28,709 images, each being 48x48 pixel grey scale images of faces. The faces have been automatically registered so that the face is more or less centred and occupies about the same amount of space in each image. All the pixels' grey level values for a specific picture are within a single string. This is one of the attributes in the data set. The corresponding feeling to that picture is another attribute and is indicated by an integer number ranging from 0 to 6 (0 = Angry, 1 = Disgust, 2 = Fear, 3 = Happy, 4 = Sad, 5 = Surprise, 6 = Neutral).



	emotion	pixels
0	0	70 80 82 72 58 58 60 63 54 58 60 48 89 115 121...
1	0	151 150 147 155 148 133 111 140 170 174 182 15...
2	2	231 212 156 164 174 138 161 173 182 200 106 38...
3	4	24 32 36 30 32 23 19 20 30 41 21 22 32 34 21 1...
4	6	4 0 0 0 0 0 0 0 0 0 0 0 3 15 23 28 48 50 58 84...
...	...	...
28704	2	84 85 85 85 85 85 85 85 86 86 86 87 86 86 91 9...
28705	0	114 112 113 113 111 111 112 113 115 113 114 11...
28706	4	74 81 87 89 95 100 98 93 105 120 127 133 146 1...
28707	0	222 227 203 90 86 90 84 77 94 87 99 119 134 14...
28708	4	195 199 205 206 205 203 206 209 208 210 212 21...

28709 rows × 2 columns

Fig. 2 Input Dataset

B. Data Pre-processing:

For data pre-processing, the data attribute "pixels" was first converted to a list of integers from a single string, because otherwise, the model cannot have access to this attribute values. Next, the data is split into train set and test set by 80% and 20% accordingly. Moreover, there is another test set file on Kaggle [2] that does not have any "emotion" attributes, and only consists of pixel values. This is the validation set. Next, the train set should be scaled using the standardization method according to the formula:

$$x_{scaled} = \frac{x - \mu}{\sigma}$$

This process should be implemented on both the train set and the test set. Furthermore, the dataset didn't have any missing values. After scaling the data, the target attribute should be encoded using one-hot encoding method to avoid information leakage and wrong-pattern recognition.

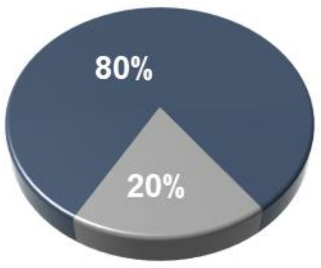


Fig. 3 Splitting data into train set (dark blue) and test set (grey)

C. **CNN Model Version I**

The CNN model version I is designed as the following:

- Convolution layer: 32
- Convolution layer: 64
- Max pooling: (2, 2)
- Convolution layer: 128
- Drop out: 0.1
- Max pooling: (2, 2)
- Flatten
- Dense: 2048
- Drop out: 0.5
- Dense: 1024
- Drop out: 0.5
- Dense: 512
- Drop out: 0.5
- Softmax: 7

For the output layer, Softmax activation function is used since there are 7 target classes, aka. Categorical classes. Then the model is trained using Adam optimizer and Categorical cross entropy loss function through 25 epochs. Two noteworthy points is that in this CNN model (and also in the CNN version II model), the kernel sizes for the layers were chosen differently to help increase nonlinearity and kernel diversity. Moreover, early stopping method was also employed to avoid overfitting. Also, since it takes a very long time (2+ hours) to train the model on a conventional CPU, the code was run on GPU. Finally, the model weights and parameters are all saved in Json format to be used later on. Finally, after the model is trained, its performance is evaluated on the test set (20% of the total data that was split, in the previous section) using the specified metrics in the (IV) Methodology section.

D. CNN Model Version II

The CNN model version II is designed as the following:

- Convolution layer: 128
- Max pooling: (2, 2)
- Drop out: 0.1
- Convolution layer: 128
- Max pooling: (2, 2)
- Drop out: 0.1
- Convolution layer: 256
- Max pooling: (2, 2)
- Drop out: 0.1
- Flatten
- Dense: 500
- Drop out: 0.5
- Dense: 200
- Drop out: 0.4
- Dense: 100
- Drop out: 0.3
- Softmax: 7

As can be seen, the architecture of the CNN models is quite different, with the CNN version II model having a more complex CNN but a less complex FFNN.

E. Image Processing and Employing the CNN Model:

In order to use the CNN model for webcam streaming data in a faster way, the model and the network weights should better be saved in a separate Json file (as was explained in the previous part) and then be uploaded later on for the webcam data input. To do so, image processing technics are implemented. First, the webcam frames are caught. Then, facial recognition is employed to detect the face of the individual(s) in front of the camera. Next, the model is called upon the frames and then feeling detection will take place on the frames. Thus based on the individual's facial expression at any given time, the model will assign an according emotion tag to the frames. The results are mentioned in section (V) Results.

F. Parameter Tuning:

Parameter tuning is also employed to find the best possible architecture to yield the best result. Since the CNN version I model has better accuracy on both the test set and the train set, parameter tuning was based on the CNN version I model. Different parameters are tested as follow:

- Neuron: 32, 64
- Activation function: Rectified linear unit (relu), Tangent hyperbolic (tanh)
- Optimizer: Adam, Stochastic gradient descent (SGD)
- Epoch: 25, 35

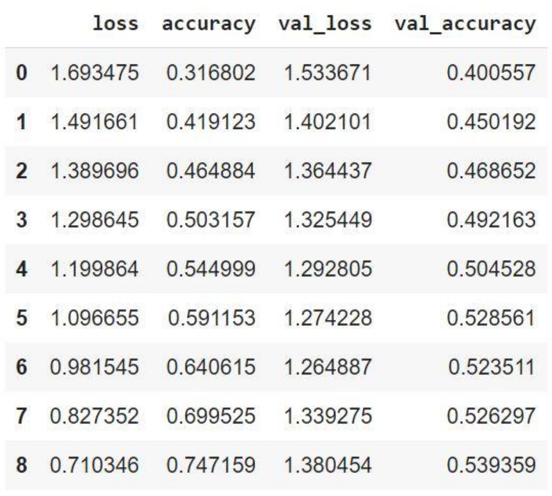
After running tuning the model using the above parameters, the best model yields:

TABLE I
PERFORMANCE METRICS OF THE BEST MODEL, AFTER PARAMETER TUNNING

Accuracy	Precision	Epochs	Neurons	Optimizer	Activation function
56%	59%	25	32	Adam	Relu

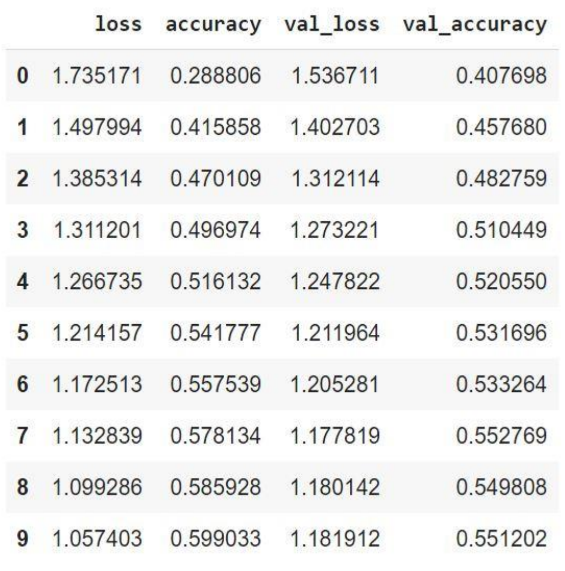
V. RESULTS

As was mentioned in the methodology section, different metrics were used for performance evaluation. Here are the results for both of the CNN models:



	loss	accuracy	val_loss	val_accuracy
0	1.693475	0.316802	1.533671	0.400557
1	1.491661	0.419123	1.402101	0.450192
2	1.389696	0.464884	1.364437	0.468652
3	1.298645	0.503157	1.325449	0.492163
4	1.199864	0.544999	1.292805	0.504528
5	1.096655	0.591153	1.274228	0.528561
6	0.981545	0.640615	1.264887	0.523511
7	0.827352	0.699525	1.339275	0.526297
8	0.710346	0.747159	1.380454	0.539359

Fig. 4 Model history for CNN version I



	loss	accuracy	val_loss	val_accuracy
0	1.735171	0.288806	1.536711	0.407698
1	1.497994	0.415858	1.402703	0.457680
2	1.385314	0.470109	1.312114	0.482759
3	1.311201	0.496974	1.273221	0.510449
4	1.266735	0.516132	1.247822	0.520550
5	1.214157	0.541777	1.211964	0.531696
6	1.172513	0.557539	1.205281	0.533264
7	1.132839	0.578134	1.177819	0.552769
8	1.099286	0.585928	1.180142	0.549808
9	1.057403	0.599033	1.181912	0.551202

Fig. 5 Model history for CNN version II

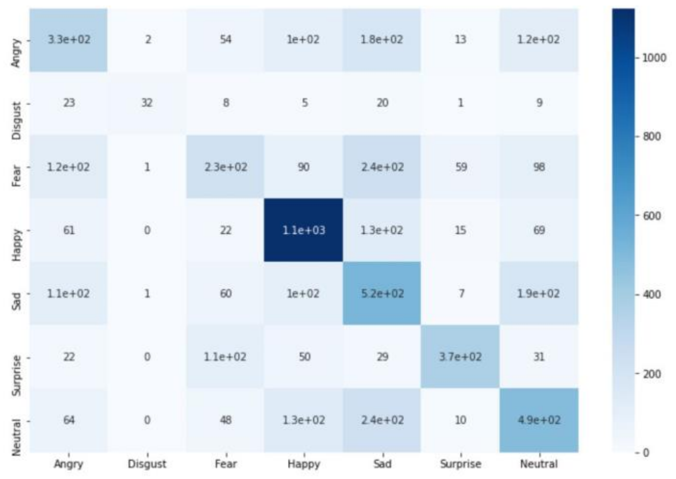


Fig. 6 Heat map for CNN version I

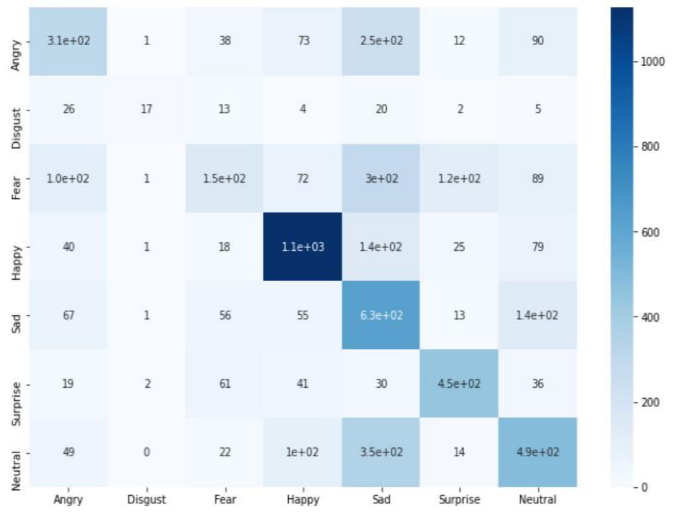


Fig.7 Heat map for CNN version II

TABLE 2
PERFORMANCE EVALUATION OF BOTH CNN VERSION I AND CNN VERSION II

Model	Accuracy	Precision	Recall	F1 score
Model I	54%	61%	48%	50%
Model II	56%	59%	48%	50%

Hence, according to the model history results, for CNN version I, the accuracy on the train set is about 75% and the accuracy on the test set is about 54%. It also has a less complex CNN and a more complex FFNN. In contrast, CNN version II has nearly 60% accuracy on the trainset and about 55% accuracy on the test set. It also has a more complex CNN and a less complex FFNN. Therefore, the CNN architecture is more important than that of the FFNN's, especially for future improvements. As can be seen in table 2, Model I has worse accuracy but a better precision compared to model II. But since the input data is highly imbalanced (Fig.13), it is better not to only rely on the accuracy score, but also to consider precision as well.

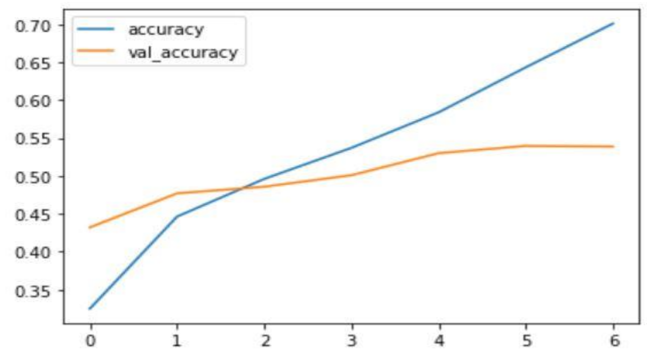


Fig. 8 Accuracy on the train set (blue), and the accuracy on the test set (orange) for CNN version I

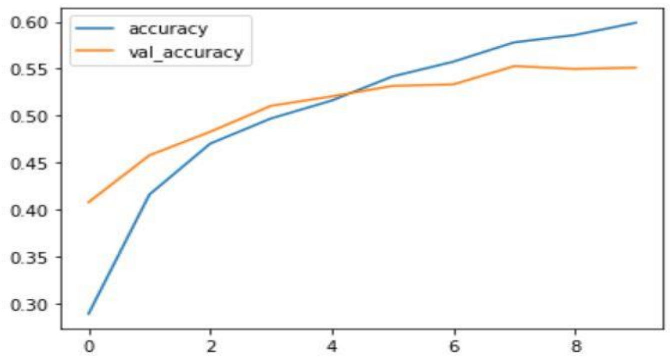


Fig. 9 Accuracy on the train set (blue), and the accuracy on the test set (orange) for CNN version II

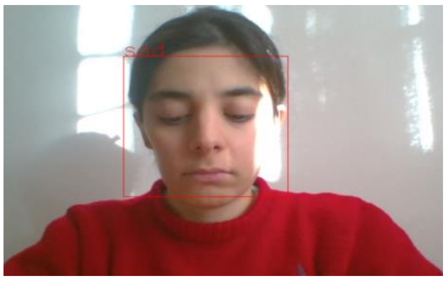


Fig. 10 Testing the model on webcam data, labelled as "sad"

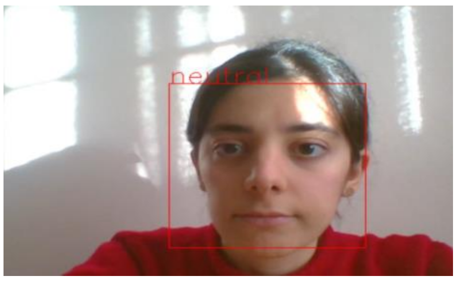


Fig. 11 Testing the model on webcam data, labelled as "neutral"

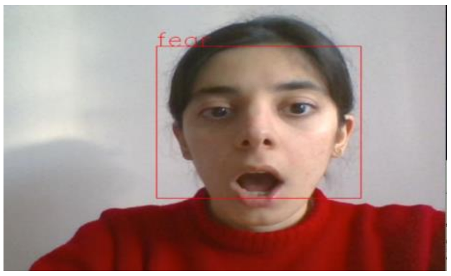


Fig. 12 Testing the model on webcam data, labelled as “fear”

VI. IMPEDIMENTS

As was mentioned in section (3) Related works, most of the feeling detection works has been done on images, not streaming videos in real-time. For real-time input, the volume of data is larger and therefore the error will probably increase as the accuracy decreases. Another issue is that the data is highly imbalanced. For example, the instances having “disgust” labels are much less than instances having “happy” labels.

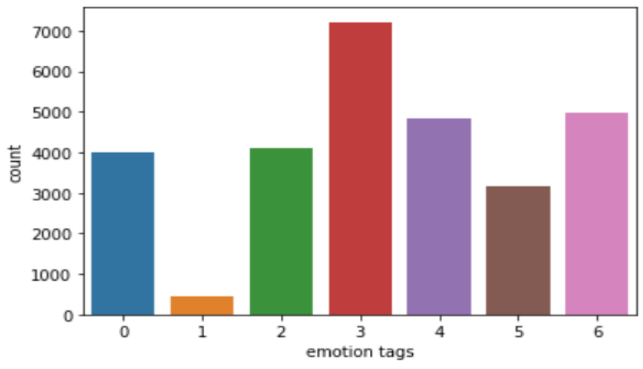


Fig. 13 Imbalanced input dataset

Another hot-topic issue in the modern world is with the AI ethics (artificial intelligence ethics). Is it fair to use a model with the current medium accuracy of nearly 60% to detect people’s emotions? Is it okay to use such methods to label people’s inner side with just facial expression analysis and probably without their knowing? What about people’s privacy? All these questions arise when it is tried to delve into the emotion detection domain.

VII. SUGGESTIONS

There are several suggestions to help improve the model’s performance. First, there are better and faster methods to use in emotion detection, such as AlexNet, ResNet, Residual learning, XGBoost, CatBoost, and even Ensemble algorithms. Furthermore, more parameters can be tuned to help find the best possible architecture for the model (which clearly is more time consuming). Second, sharper processing units can be taken advantage of, such as GPUs or even supercomputers for real-life applications. Third, the Kaggle’s input data [2] mostly had low-quality images. But if we can use better imaging devices to save higher quality pictures, then the model’s accuracy will increase as well. Towards that end, one may try saving coloured images (RGB), rather than grey scale in this paper. The reason is that coloured images are better able to show facial expressions which sometimes tend to be quite hideous in grey scale. Forth, it is beneficial to add more diversity to the input data. Tilting, rotation, and transportation can be added to the input images to help increase non-linearity and diversity for later predictions. Moreover, the data can include images of different age groups, so that the model won’t get biased. Hopefully in the future, more extensive work is conducted to analyse distinct bio-signals (not only facial expression but also voice, eye movement, brain’s EEG[2] signals, etc.) to detect feelings or even criminal intentions much more accurately using typical sensory devices.

VIII. CONCLUSIONS

In this paper, feeling detection was conducted on streaming webcam data. Two different models were proposed (CNN version I and CNN version II). The architecture of convolution and FFNN parts in both models were different, but as it was seen in the results, the convolution part is a more defining factor. Later, the model was employed on the streaming webcam data to label an individual’s emotion based on the person’s facial expression. And although the best model architecture (found using parameter tuning) showed only 60% accuracy on the test set, it was still able to detect the person’s feeling in front of the camera (although for classes were there were less instances such as the “disgust” class, the person’s facial expression had to get a little more exaggerative so that the model could detect it correctly as “disgust”).

ACKNOWLEDGMENT

Hereby, I want to give many thanks to Professor Katarina Grolinger, at University of Western Ontario, Software Engineering Program, who was the instructor of the course *Data Analytics Foundation* (ECE 9603, Fall 2020) for her insightful lectures, her patient guidance, her open and flexible character.

REFERENCES

[1] https://www.researchgate.net/publication/287867911_Emotion_Recognition_and_Its_Applications

[2] https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/data

[3] Wanliang Tan, Jiahui Wang, Yu Wang, “CNN Models for Classifying Emotions Evoked by Paintings,” *Stanford University*, CS231N, 2018

[2] Electroencephalography

[4] Xin Lu, Neela Sawant, Michelle G. Newman, Reginald B. Adams, Jr., James Z. Wang, Jia Li, “Identifying Emotions Aroused from Paintings,” Adobe Systems Inc., Amazon Inc., The Pennsylvania State University

[5] Namita Mittal, Divya Sharma, Manju Lata Joshi, “Image Sentiment Analysis Using Deep Learning,” IEEE/WIC/ACM International Conference on Web Intellignce (WI), IEEE, Santiago, Chile, 2018

[6] Hye-Rin Kim, Yeong-Seok Kim, Seon Joo Kim, In-Kwon Lee, “Building Emotional Machines: Recognizing Image Emotions through Deep Neural Networks,” arXiv:1705.07543v2 [cs.CV] 3 Jul 2017

[7] Minh-An Quinn, Grant Sivesind, Guilherme Reis, “Real-time Emotion Recognition from Facial Expressions,” CS 229, Stanford University