**Chapter 1**

**Introduction**

Human emotion detection is implemented in many areas requiring additional security or information about the person. It can be seen as a second step to face detection where we may be required to set up a second layer of security, where along with the face, the emotion is also detected. This can be useful to verify that the person standing in front of the camera is not just a 2-dimensional representation.

Another important domain where we see the importance of emotion detection is for business promotions. Most of the businesses thrive on customer responses to all their products and offers. If an artificial intelligent system can capture and identify real time emotions based on user image or video, they can make a decision on whether the customer liked or disliked the product or offer.

We have seen that security is the main reason for identifying any person. It can be based on finger-print matching, voice recognition, passwords, retina detection etc. Identifying the intent of the person can also be important to avert threats. This can be helpful in vulnerable areas like airports, concerts and major public gatherings which have seen many breaches in recent years.

Human emotions can be classified as: fear, contempt, disgust, anger, surprise, sad, happy, and neutral. These emotions are very subtle. Facial muscle contortions are very minimal and detecting these differences can be very challenging as even a small difference results in different expressions. Also, expressions of different or even the same people might vary for the same emotion, as emotions are hugely context dependent. While we can focus on only those areas of the face which display a maximum of emotions like around the mouth and eyes, how we 2 extract these gestures and categorize them is still an important question. Neural networks and machine learning have been used for these tasks and have obtained good results.

**1.1 Methodology:**

Machine learning algorithms have proven to be very useful in pattern recognition and classification. The most important aspects for any machine learning algorithm are the features. In this paper we will see how the features are extracted and modified for algorithms. We will compare algorithms and the feature extraction techniques from different papers. The human emotion dataset can be a very good example to study the robustness and nature of classification algorithms and how they perform for different types of data. Usually before extraction of features for emotion detection, face detection algorithms are applied on the image or the captured frame. We can generalize the emotion detection steps as follows:

1) Dataset preprocessing

2) Face detection

3) Feature extraction

4) Classification based on the features

In this work, we focus on the feature extraction technique and emotion detection based on the extracted features.

**1.2 Scope of the Project**

In this project facial expression recognition system is implemented using convolution neural network. Facial images are classified into seven facial expression categories namely

1. Anger
2. Disgust
3. Fear
4. Happy
5. Sad
6. Surprise
7. Neutral

Kaggle dataset is used to train and test the classifier.

**Chapter 2**

**System Analysis**

**2.1 Existing system:**

So far, numerous research projects have been done on recognizing emotion from Face. Many used many methods to implement their systems. Facial expressions provide the building blocks with which to understand emotion. In order to effectively use facial expressions, it is necessary to understand how to interpret expressions, and it is also important to study what others have done in the past. Facial Action Coding System (FACS) is a system to determine human facial expressions, originally developed by Paul Ekman and Wallace V. Friesen, and published in 1978. Ekman, Friesen, and Joseph C. Hager published a significant update to FACS in 2002. Movements of individual facial muscles are encoded by FACS from slightly different instant changes in facial appearance. It is a common standard to systematically categorize the physical expression of emotions. Recently, FACS has been established as a computed automated system that detects faces in videos, extracts the geometrical features of the faces, and then produces 6 | P a g e temporal profiles of each facial movement. FACS is a key system to determine Facial Feature Extraction.

**2.2 Proposed system**:

This project has been divided into two phases. The first phase consisted on the use of a facial emotion labeled data set to train a deep learning network. The chosen data set is the FER2013. More details about the project can be found in next section. Additionally, evaluations were performed on several network topologies to test their prediction accuracy. The use of convolutional neural networks on the topologies was preferred given its great achievements on computer vision tasks. An overview of deep learning concepts, with an emphasis on convolutional networks is presented in Chapter4. In order to perform the implementation 3 of the network and the training process, Google’s library TensorFlow was used. Chapter 4 introduces TensorFlow functions for computer vision, and the reasons it was selected compared to other frameworks. The second phase focused on testing the model against a new data set, Dataset. Similarly, to the previous phase, a detailed explanation is presented in Chapter 3. The idea is to make a comparison on both data sets, and evaluate the generalization property of the network. Also, a focus on some parameters and its effect on the model’s accuracy prediction was performed. These parameters were chosen because their influence over the network’s behavior:

• Convolution operation

• Pooling operation

• Dropout

• Optimizers

More information about parameter’s value selection is displayed in Chapter 4. Experiments results are supplied in Chapter 6. Finally, future work and conclusions are addressed in the Conclusion and Enhancement section

**2.3 System requirements:**

* A dataset testing and training
* **Hardware:**

A system comprising of minimum 8gb of ram and 246gb of storage

* **Software:**
* VSCode
* Python
* Libraries (TensorFlow, Keras, OpenCV etc)

**Chapter 3**

**System Design**

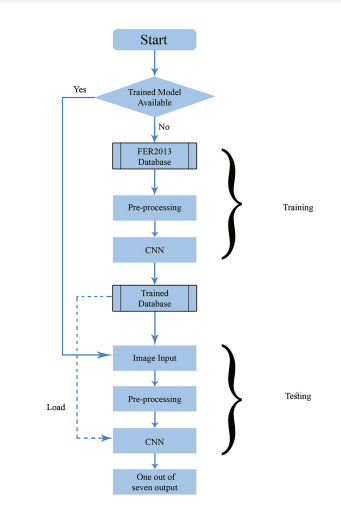
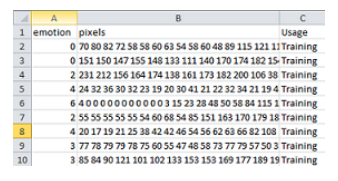


Figure (3.1): Flow chart of whole project

3.1 DATASET DESCRIPTION:

The choice of images used for training is responsible for a big part of the performance of the eventual model. This implies the need for a both qualitative and quantitative dataset. For emotion analysis there are several datasets avail- able for training and testing the model which contain more than thousand of images captured in low or high resolution, at different brightness, etc. The dataset from a Kaggle Facial Expression Recognition Challenge (FER2013) is used for the training and testing. It comprises pre-cropped, 48-by-48-pixel grayscale images of faces each labeled with one of the 7 emotion classes: anger, disgust, fear, happiness, sadness, surprise, and neutral. Dataset has training set of 35,887 facial images with facial expression labels. In which 28,709 are for training and 7,178 are for testing.

 Figure (3.2): FCR2013 Raw dataset

3.2 IMPLEMENTATION

There are two phases of the proposed algorithm – testing and training. The training is done so that network is able to classify the emotions properly of the provided face images. The first step of proposed algorithm is to verify if the trained data/model is already present or not. If it isn’t, then we need to train the system first, to execute the next step which is testing of emotion classification else we would have gone directly for testing.

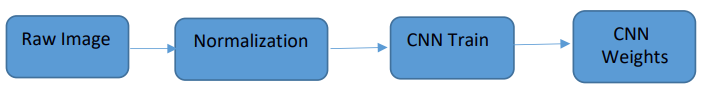
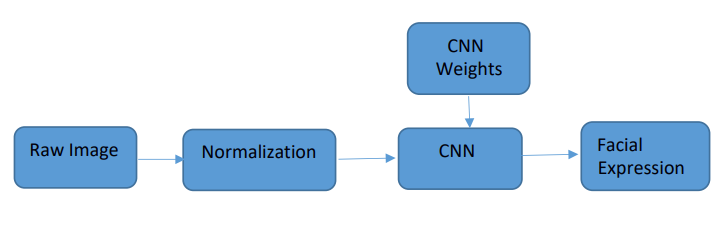


Figure (3.3): Training Phase

Figure (3.4): Testing Phase

Another part is the image pre-processing, in this we make the suitable input to train the model by using certain python library. Last but not the least we use deep convolution network to classify the image according to the trained model and predict the seven basic emotion. The following figure shows such a flow.

Following are the four major steps followed.

3.2.1 Pre-processing:

Usually, the purpose of using preprocessing steps in face detection system is to speed up the detection process and reducing false positives. The dataset is basically rows of individual image features such as label, actual pixel values and the emotions. Hence, this info needs to be converted to serve as an input to our next step, that is, training. Training step will take images as an input.

The Convolutional Neural Network will take image array input of size 48×48. This objective is completed in this pre-processing step.

The pixel values from the dataset are taken as an input and converted to jpegs of size 48×48, also the arrays are created to serve as an input to our convolutional neural network.

3.2.2 Training:

The network is programmed with the use of the TensorFlow library on top of TensorFlow, running on Python. This environment reduces the complexity of the code as neuron layers are created instead of single neurons. The advantage of this set up is that we can get real-time feedback on the training progress and accuracy, which increases the reusability of the trained mode.

Before training, images from FERC2013 dataset are preprocessed, in the pre-processing, 28709 samples are used, after validating we got 11,246 valid samples for training.

3.2.3 Optimizing:

A kind of configuration which is provided to the model, which is external to data and one cannot determine its value from data are hyperparameters. These are often used in processes to help estimate model parameters. Hyperparameters are basically tuned when experimenting with machine learning models. This parameter is tuned such that the prediction accuracy increases.

3.2.4 Prediction:

After the optimizing step, the prediction is done.

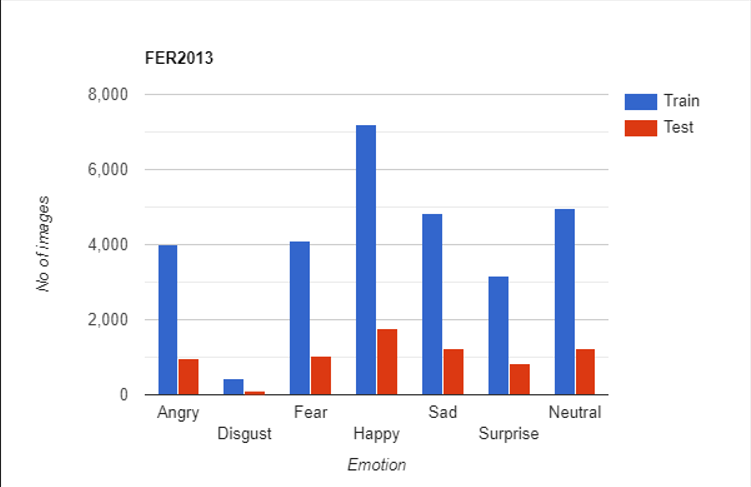


Figure (3.5): Content of Dataset FER2013

**Chapter 4**

**System Implementation**

**4.1 TOOLS AND LIBRARIES USED**

TensorFlow:

TensorFlow is an open-source end-to-end platform for creating Machine Learning applications. It is a symbolic math library that uses dataflow and differentiable programming to perform various tasks focused on training and inference of deep neural networks. It allows developers to create machine learning applications using various tools, libraries, and community resources.

Keras:

**Keras**is an opensource Neural Network library written in Python that runs on top of Theano or TensorFlow. It is designed to be modular, fast and easy to use. It was developed by François Chollet, a Google engineer.

OpenCV:

OpenCV is the library we will be using for image transformation functions such as converting the image to grayscale. It is an opensource library and can be used for many image functions and has a wide variety of algorithm implementations. C++ and Python are the languages supported by OpenCV. It is a complete package which can be used with other libraries to form a pipeline for any image extraction or detection framework. The range of functions it supports is enormous, and it also includes algorithms to extract feature descriptors.

## SciPy:

**SciPy in Python** is an open-source library used for solving mathematical, scientific, engineering, and technical problems. It allows users to manipulate the data and visualize the data using a wide range of high-level Python commands. SciPy is built on the Python NumPy extension. SciPy is also pronounced as “Sigh Pi.”

Threading:

A thread is a unit of execution on concurrent programming. Multithreading is a technique which allows a CPU to execute many tasks of one process at the same time. These threads can execute individually while sharing their process resources.

**4.2 CONVOLUTIONAL** **NEURAL NETWORKS:**

The fundamental building block of a Neural Network is a neuron. Figure 3.1 shows the structure of a neuron. Forward propagation of information through a neuron happens when inputs to are multiplied by their corresponding weights and then added together. This result is passed through a nonlinear activation function along with a bias term which shifts the output. The output is between 0 and 1 Figure (4.1) which makes it suitable for problems with probabilities. The purpose of the activation function is to introduce nonlinearities in the network since most real world, data is nonlinear. The use of a nonlinear function also allows Neural Networks to approximate complex functions.

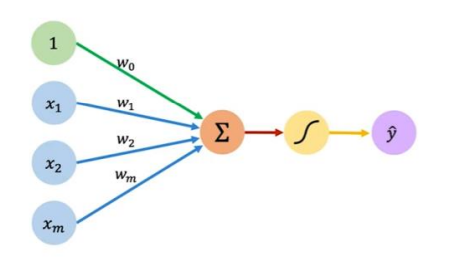
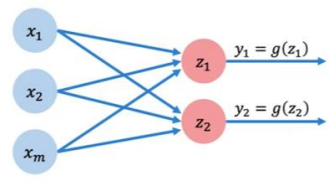


Figure (4.1): The basic structure of a neuron

Neurons can be combined to create a multi output NN. If every input has a connection to every neuron this is called dense or fully connected. Figure 4.2 shows a dense multi output NN with two neurons. A deep NN has multiple hidden layers stacked on top of each other and every neuron in each hidden layer is connected to a neuron in the previous layer. Figure 4.3 shows a fully connected NN with 5 layers

 Figure (4.2): A multi output NN with two neurons

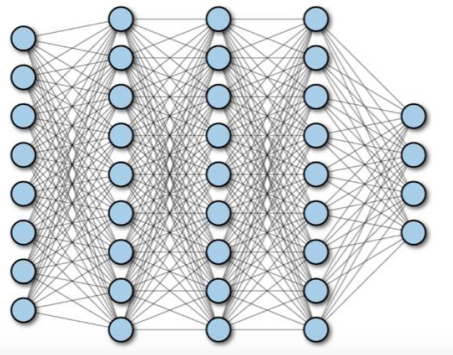


Figure (4.3): A fully connected NN

**4.3 The CNN (Convolutional Neural Network) concept**:

A CNN is a DL algorithm which takes an input image, assigns importance (learnable weights and biases) to various aspects/objects in the image and is able to differentiate between images. The preprocessing required in a CNN is much lower than other classification algorithms. Figure 4.4 shows the CNN operations. The architecture of a CNN is analogous to that of the connectivity pattern of neurons in the human brain and was inspired by the organization of the visual cortex. One role of a CNN is to reduce images into a form which is easier to process without losing features that are critical for good prediction. This is important when designing an architecture which is not only good at learning features but also is scalable to massive datasets.

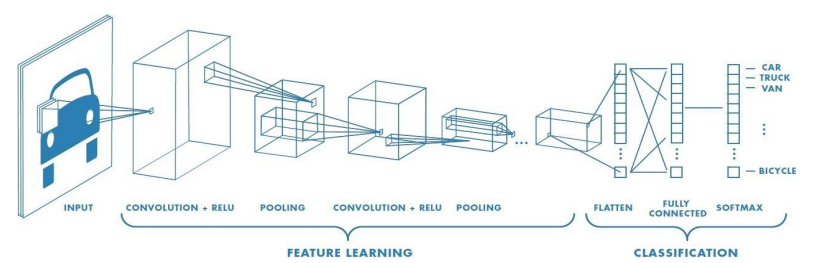


Figure (4.4): The CNN operations

4.3.1 Convolution operation:

The objective of the convolution operation is to extract high level features such as edges from an input image. The convolution layer functions are as follows.

* The first convolutional layer(s) learns features such as edges, color, gradient orientation and simple textures.
* The next convolutional layer(s) learns features that are more complex textures and patterns.
* The last convolutional layer(s) learns features such as objects or parts of objects.

The element involved in carrying out the convolution operation is called the kernel. A kernel filters everything that is not important for the feature map, only focusing on specific information. The filter moves to the right with a certain stride length till it parses the complete width. Then, it goes back to the left of the image with the same stride length and repeats the process until the entire image is traversed.

4.3.2 Pooling operation:

The pooling layer reduces the spatial size of a convolved feature. This is done to decrease the computations required to process the data and extract dominant features which are rotation and position invariant. There are two types of pooling, namely max pooling and average pooling. Max pooling returns the maximum value from the portion of the image covered by the kernel, while average pooling returns the average of the corresponding values. Figure 4.5 shows the outputs obtained by performing max and average pooling on an image.

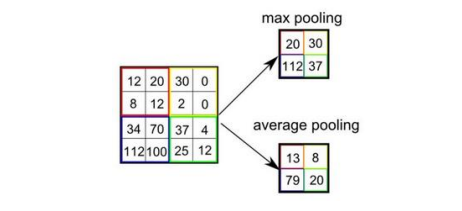


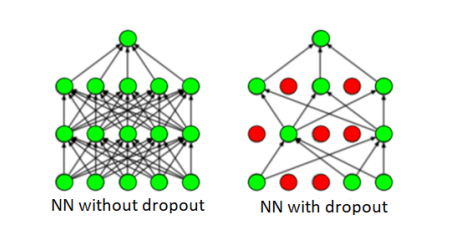
Figure (4.5): Max and average pooling outputs for an image

4.3.3 Fully connected layer:

Neurons in a fully connected layer have connections to all neurons in the previous layer. This layer is found towards the end of a CNN. In this layer, the input from the previous layer is flattened into a one-dimensional vector and an activation function is applied to obtain the output.

4.3.4 Dropout:

Dropout is used to avoid overfitting. Overfitting in a Machine Learning model happens when the training accuracy is much greater than the testing accuracy. Dropout refers to ignoring neurons during training so they are not considered during a particular forward or backward pass leaving a reduced network. These neurons are chosen randomly and an example is shown in Figure 4.6. The dropout rate is the probability of training a given node in a layer, where 1.0 means no dropout and 0.0 means all outputs from the layer are ignored.

Figure (4.6): Dropout in a Neural Network

4.3.5 Batch normalization:

Training a network is more efficient when the distributions of the layer inputs are the same. Variations in these distributions can make a model biased. Batch normalization is used to normalize the inputs to the layers.

4.3.6 Activation function:

An activation function in a neural network defines how the weighted sum of the input is transformed into an output from a node or nodes in a layer of the network.

The rectified linear activation function or ReLU for short is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero.

**4.4 CNN architecture**:

ML models can be built and trained easily using a high-level Application Programming Interface (API) like Keras. In this report, a sequential CNN model is developed using TensorFlow with the Keras API since it allows a model to be built layer by layer. TensorFlow is an end-to-end opensource platform for Machine Learning. It has a flexible collection of tools, libraries and community resources to build and deploy Machine Learning applications. Figure 4.7 shows the structure of a CNN where conv. denotes convolution.

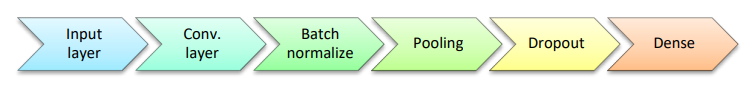


Figure (4.7): Structure of a CNN

.

**Chapter 5**

**System Testing**

**5.1 Confusion matrix**:

A confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix.

The confusion matrix provides values for the four combinations of true and predicted values, True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN). Precision, recall and F-score are calculated using TP, FP, TN, FN. TP is the correct prediction of an emotion, FP is the incorrect prediction of an emotion, TN is the correct prediction of an incorrect emotion and FN is the incorrect prediction of an incorrect emotion. Consider an image from the happy class. The confusion matrix for this example is shown in Figure 4.1. The red section has the TP value as the happy image is predicted to be happy. The blue section has FP values as the image is predicted to be sad, angry, neutral or fear. The yellow 22 section has TN values as the image is not sad, angry, neutral or fear but the model predicted this. The green section has FN values as the image is not happy but was predicted to be happy.

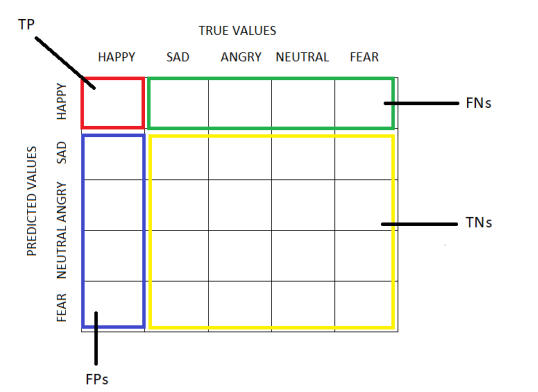


Figure (5.1): Confusion matrix

**5.2 Extensive testing**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Predicted  Actual | Anger | Disgust | Fear | Happy | Sad | Surprise | neutral | All |
| Anger | **8** | 3 | 0 | 0 | 0 | 0 | 0 | 11 |
| Disgust | 3 | **12** | 0 | 0 | 1 | 1 | 0 | 17 |
| Fear | 1 | 0 | **3** | 0 | 0 | 1 | 0 | 5 |
| Happy | 2 | 0 | 0 | **12** | 0 | 1 | 1 | 16 |
| Sad | 2 | 0 | 0 | 0 | **8** | 1 | 0 | 11 |
| Surprise | 0 | 2 | 2 | 0 | 2 | **11** | 1 | 18 |
| Neutral | 1 | 1 | 0 | 0 | 0 | 0 | **2** | 4 |
| All | 16 | 15 | 7 | 15 | 11 | 14 | 4 | 82 |

Figure (5.2): Extensive testing done to verify the system functionality using confusion matrix

From the above table we can find the accuracy of each emotion prediction by the project:

Anger: 8/11=72%

Disgust: 12/17=70%

Fear: 3/5=60%

Happy: 12/16=75%

Sad: 8/11=72%

Surprise: 11/18=61%

Neutral: 2/4=50%

**Chapter 6**

**Results**

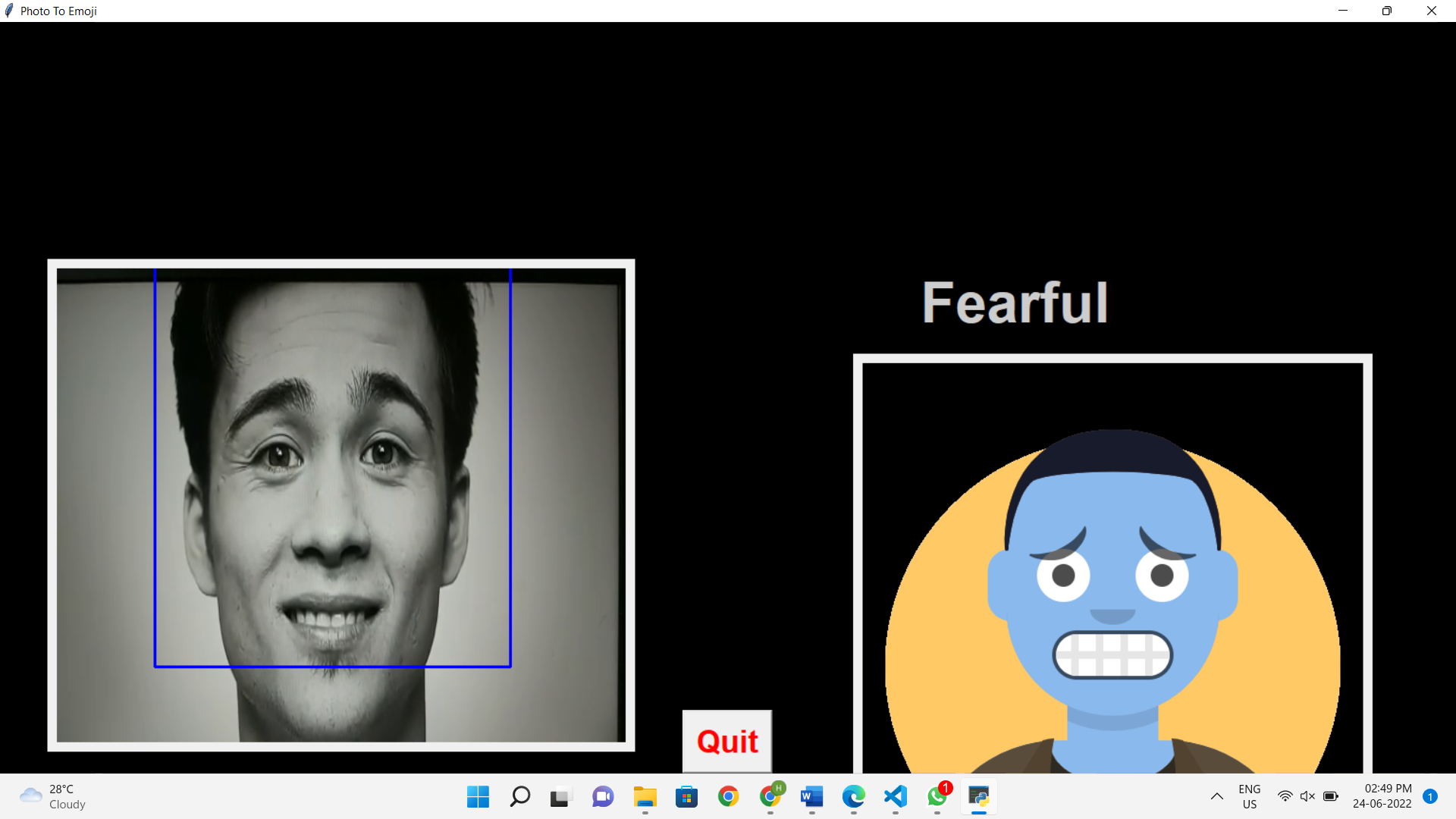


Figure (6.1): The image here of a man is showing a emotion of Fear and the output is the emoji of fear

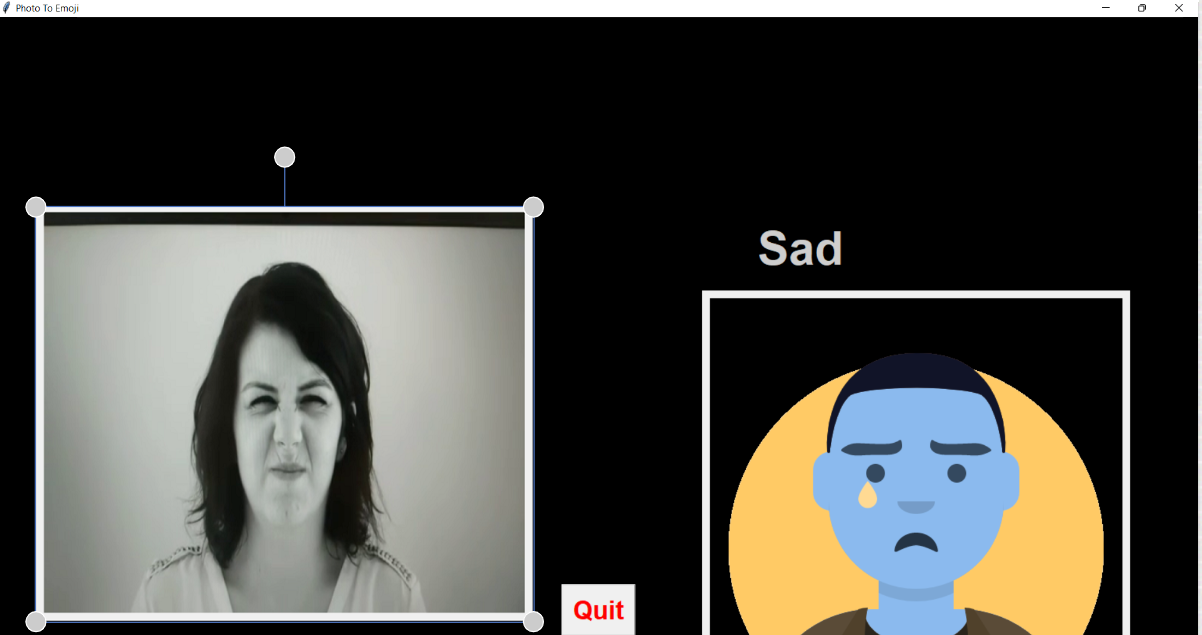


Figure (6.2): The image here of a woman is shown a emotion of Sadness and the output is the emoji of Sad

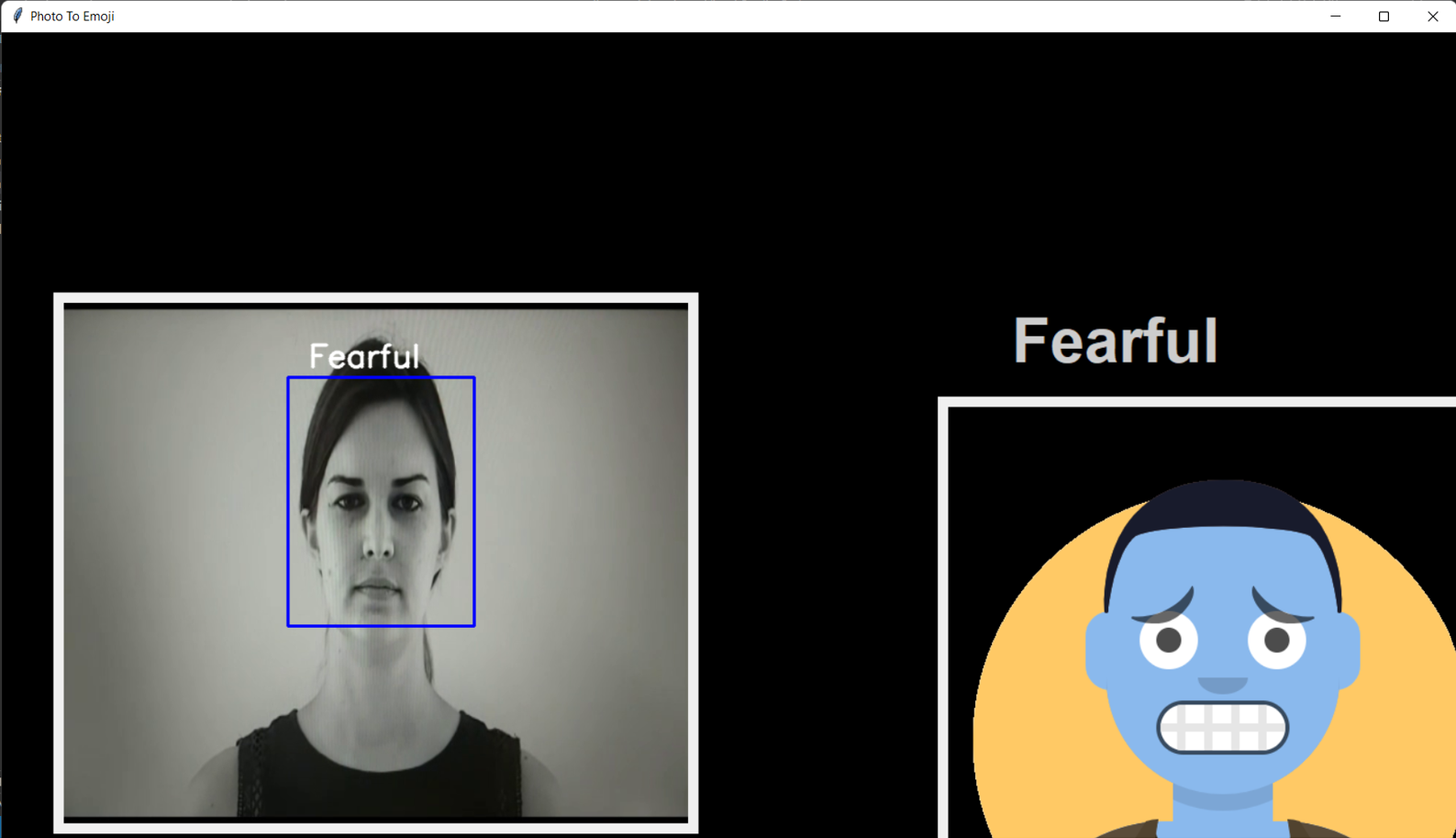


Figure (6.3): The image here of a woman is showing a emotion of Fear and the output is the emoji of fear

**Conclusion and Future Enhancements**

**Conclusion:**

This work presents a convolution neural network architecture that uses feature information from facial parts as input into two separate CNN channels. The output from the two channels converges into a fully connected layer and the result used for classification. This method is aimed to have an advantage over using the whole face as an input by having an increased recognition accuracy and reduced cost. Experimental results based on the FER2013 confirms the effectiveness and robustness of this method. It is shown that our proposed method can achieve the average expression recognition accuracy of 71 percentage. Another interesting aspect of this work that can be explored in the future would be to test this approach on more databases.

**Feature Enhancements:**

Facial emotion recognition is an emerging field so considering other NNs such as Recurrent Neural Networks (RNNs) may improve the accuracy. The feature extraction is similar to pattern recognition which is used in intelligence, military and forensics for identification purposes. Thus, techniques such as the Caps net algorithm for pattern recognition can be considered. DL based approaches require a large labeled dataset, significant memory and long training and testing times which makes them difficult to implement on mobile and other platforms with limited resources. Thus, simple solutions should be developed with lower data and memory requirements

**References**

Shan, C., Gong, S., & McOwan, P. W. (2005, September). Robust facial expression recognition using local binary patterns. In Image Processing, 2005. ICIP 2005. IEEE International Conference on (Vol. 2, pp. II-370). IEEE.

Chibelushi, C. C., & Bourel, F. (2003). Facial expression recognition: A brief tutorial overview. CVonline: On-Line Compendium of Computer Vision, 9.

"Convolutional Neural Networks (LeNet) – DeepLearning 0.1 documentation". DeepLearning 0.1. LISA Lab. Retrieved 31 August 2013.

[www.keras.io](http://www.keras.io)

[www.opoencv.org](http://www.opoencv.org)

[www.youtube.com](http://www.youtube.com)

[www.towardsdatascience.com/](http://www.towardsdatascience.com/)

[www.tensorflow.org](http://www.tensorflow.org)