IITISoC 2022

Team - AI/ML 7

Face Emotion Detection System

Introduction

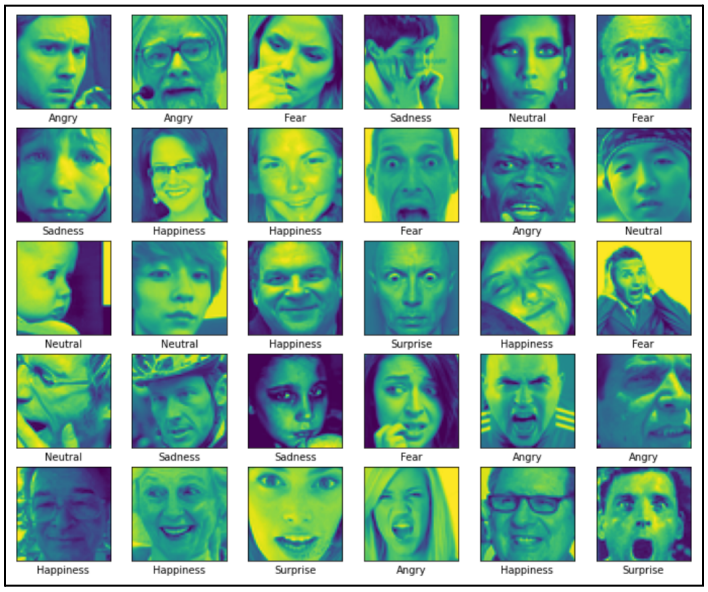
The task was to create a system to predict a person's emotion using CNN. We created and trained a model to classify a person's emotion using Tensorflow. We worked on the FER2013 dataset. Also, we implemented the model in a simple web app to detect a person's emotions using a webcam.

(1) Libraries Used

Tensorflow, Numpy, Pandas, Matplotlib, OpenCV, Flask.

(2) Dataset Evaluation

FER2013 dataset contains approximately 30,000 facial RGB images of different expressions. With the size of 48x48 pixels. Each pixel has a value ranging from 0 and 255. The training set consists of 28,709 examples and the public test set consists of 3,589 samples. The Disgust expression has a minimal number of images – 600, while other labels have nearly 5,000 samples each. There is no repetition of samples. The dataset can be visualized using the pyplot function of the matplotlib library.



(3) Data Preprocessing

The dataset has 3 columns namely emotion(having emotion label), pixel(containing image data), and usage(training and testing). Initially, the images were represented in the form of a 1-D array of size 2304 (48*48). So we reshaped the data into sizes of 48*48 arrays using NumPy.split(), NumPy.reshape() and NumPy.stack(). We converted the data into a NumPy array to feed it to the model. We first tried reshaping a single image data and then used the lambda function to reshape the whole pixels column of the FER2013 dataset into a NumPy array whose each element is a 48*48 pixel image.

Now, for the training and testing part, we used the popular train_test_split function by the sklearn library to split the data into training and testing datasets. We used 80% of the data for training the model and 20% data for testing the model. The images were split into train_images and test_images and the image labels were split into train_labels and test_labels.

(4) Building and Testing the Model

To classify the images, we used TensorFlow's sequential model. We used CNN and Deep layers in our model for image classification. CNN layers consist of convolutional layers which are the core building block of CNN, and Max Pooling layers which help in reducing

the spatial size of the representation, which decreases the required amount of computation and weights. CNN layer performs a dot product between two matrices, where one matrix is the set of learnable parameters otherwise known as a kernel, and the other matrix is the restricted portion of the receptive field. After that, we used a layer to flatten the input data. In the end, we used 2 dense layers of sizes 256 and 7 as there are 7 different outcomes of detected emotions.

The model gave a training accuracy of 94.5% and a testing accuracy of 53.4% on the dataset.

Accuracy of the testing dataset :

```
Epoch 40/50
1010/1010 [==============================] - 5s 5ms/step - loss: 0.2202 - accuracy: 0.9309
Epoch 41/50
1010/1010 [==============================] - 5s 5ms/step - loss: 0.2178 - accuracy: 0.9312
Epoch 42/50
1010/1010 [==============================] - 5s 5ms/step - loss: 0.1837 - accuracy: 0.9424
Epoch 43/50
1010/1010 [==============================] - 5s 5ms/step - loss: 0.1841 - accuracy: 0.9424
Epoch 44/50
1010/1010 [==============================] - 5s 5ms/step - loss: 0.1805 - accuracy: 0.9448
Epoch 45/50
1010/1010 [==============================] - 5s 5ms/step - loss: 0.1954 - accuracy: 0.9408
Epoch 46/50
1010/1010 [==============================] - 5s 5ms/step - loss: 0.1888 - accuracy: 0.9437
Epoch 47/50
1010/1010 [==============================] - 5s 5ms/step - loss: 0.1897 - accuracy: 0.9427
Epoch 48/50
1010/1010 [==============================] - 5s 5ms/step - loss: 0.1724 - accuracy: 0.9481
Epoch 49/50
1010/1010 [==============================] - 5s 5ms/step - loss: 0.2049 - accuracy: 0.9407
Epoch 50/50
1010/1010 [==============================] - 5s 5ms/step - loss: 0.1805 - accuracy: 0.9454
<keras.callbacks.History at 0x7f7be86ea690>
```

Accuracy of the training dataset :

```
score = model1.evaluate(test_images,test_labels, verbose = 0)

print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
/usr/local/lib/python3.7/dist-packages/tensorflow/python/util/dispatch.py:1082: UserWarning:
  return dispatch_target(*args, **kwargs)
Test loss: 4.089258670806885
Test accuracy: 0.5341320633888245
```

(5) Implementation in Web Application

The web app is developed using HTML, Flask, and OpenCV.

The main.py has three functions :

(1) index() - This is the function that is executed first and it renders the index.html. The HTML code consists of code for video display, basically, a dynamic image obtained using the video_feed function.
(2) video_feed() - This function returns a response, calling the function gen_frames function.
(3) gen_frames() - Used to generate frames.

We are using a camera. read() which returns two objects. One, whether the capture was successful or not and the other, if successful, returns the image. We then change the RGB image into gray images using the cvtColor function. Then a faces_detected array is created which uses a face detector and stores all the faces detected. Then the faces are iterated one by one, the image is created to the required dimensions using resize function and the trained model is used to detect the emotion. The maxing stores the class in which the emotion detected belongs and using the emotions array, the class name is displayed using the put text function from cv2.