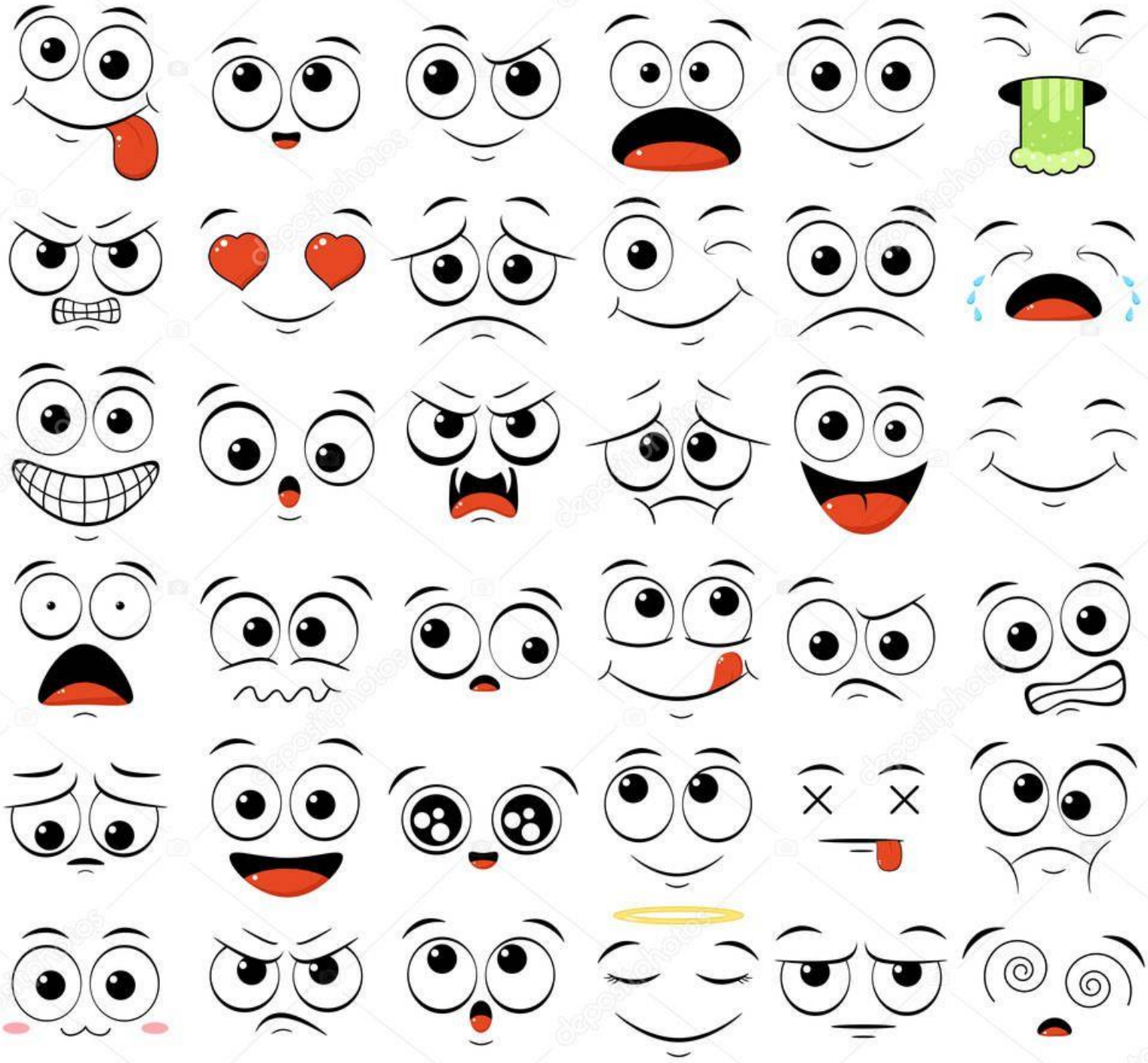
Mood Detector



-By Shivam Kumar Giri

AIM

. The objective of this study is to classify mood of the person from facial expressions Images are categorized in three classes namely sadness, fear and happiness based on the emotion shown in the facial expressions

ABSTRACT

A **mood** is a feeling or a person's specific state of mind at any particular time. A **mood** is also the prevailing emotion found not only in people but also in literature, music, and other expressive arts.

Here I designed a Machine Learning model using python which can detect the mood of the person at a particular time, based on the specs of facial expression. It takes the image/ pixels of image as input and output the mood based on image specifications. There are 3 moods: Fear, sad and happy which are detected by this application

DATASET

The dataset consists of 2045 columns with 1 column consist of emotions and other consist of image data. The image data consists of 48x48 pixel grayscale images of faces. The pixel values are stored in 2304 (48*48) columns. These column names start with pixel. Along with pixel values, there is emotion column that say about mood of the image.

There are 10817 rows of data, i.e. 10817 distinct data present, each with the emotions and pixel values of image. We will use this data for training, validation and testing with the ratio 8:1:1 of the data present in the dataset.

LIBRARIES USED

Here is developed the application using python. I used the following Libraries for implementation of the mood detector:

Numpy: for Numerical computing

Pandas: for data manipulation and analysis.

Tensorflow: for dataflow and differentiable programming

Keras: for neural-network

OpenCV: for real-time Computer Vision

os: for interacting with the operating system.

Pillow: for Image Processing.

Matplotlib: for data visualization

Seaborn: for data visualization and heat maps.

Sklearn: for classification algorithms and other Machine Learning metrics.

TRAINING DATA

Here I used Keras CNN for image classification.

Conv2D is the layer to convolve the image into multiple images
Activation is the activation function.
MaxPooling2D is used to max pool the value from the given size matrix and same is used for the next 2 layers. then, **Flatten** is used to flatten the dimensions of the image obtained after convolving it.
Dense is used to make this a fully connected model and is the hidden layer.
Dropout is used to avoid overfitting on the dataset.
Dense is the output layer contains only one neuron which decide to which category image belongsImageDataGenerator that rescales the image, applies shear in some range, zooms the image and does horizontal flipping with the image. This ImageDataGenerator includes all possible orientation of the image.

train_datagen.flow_from_directory is the function that is used to prepare data from the train_dataset directory Target_size specifies the target size of the image.

test_datagen.flow_from_directory is used to prepare test data for the model and all is similar as above.

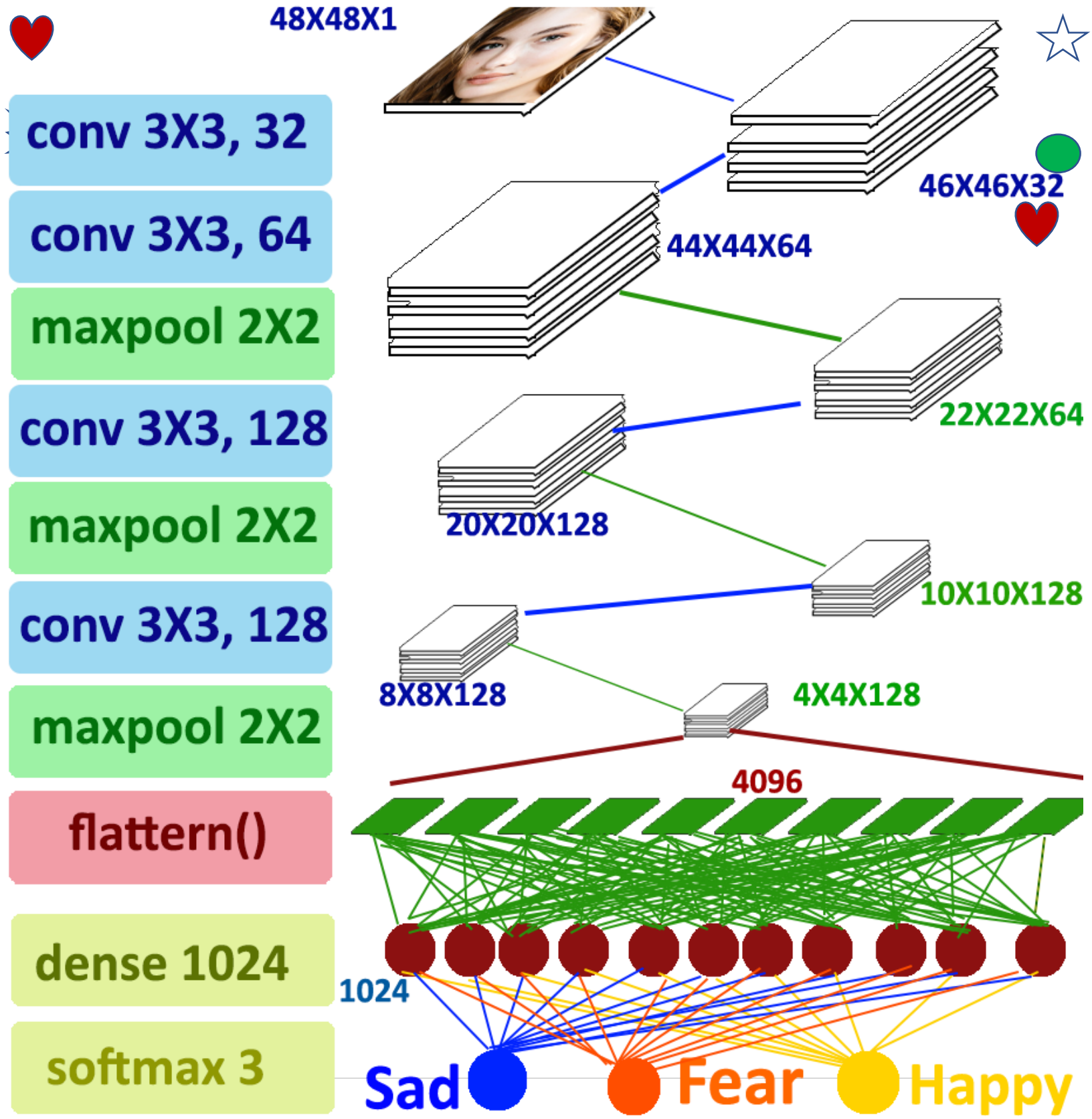
fit_generator is used to fit the data into the model made above, other factors used are **steps_per_epoch**s tell us about the number of times the model will execute for the training data.

epochs tell us the number of times model will be trained in forward and backward pass.

validation_data is used to feed the validation/test data into the model.

validation_steps denotes the number of validation/test samples.

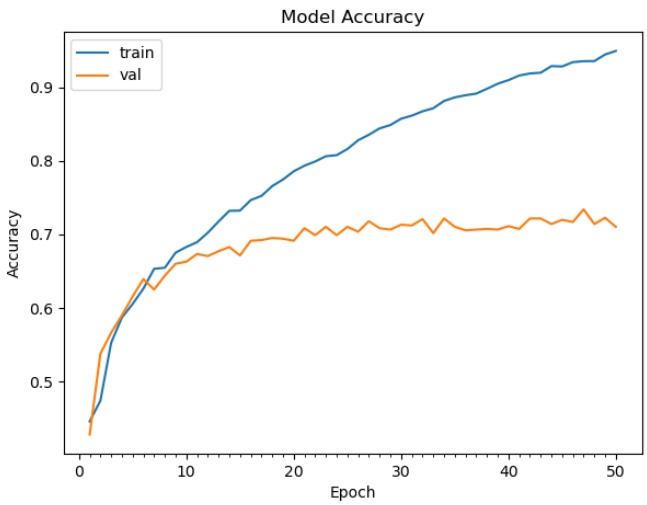
I used following CNN model in my application:

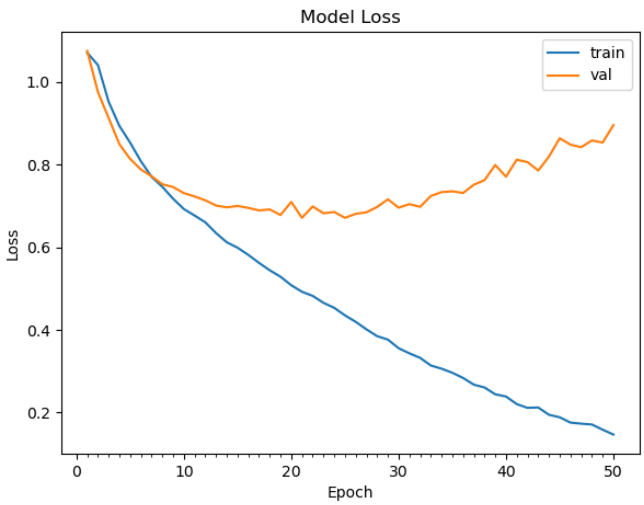


VALIDATION & TESTING

10% of dataset is used for validation and 10% for testing.

For testing the following accuracy and losses at each step of validation is recorded below:





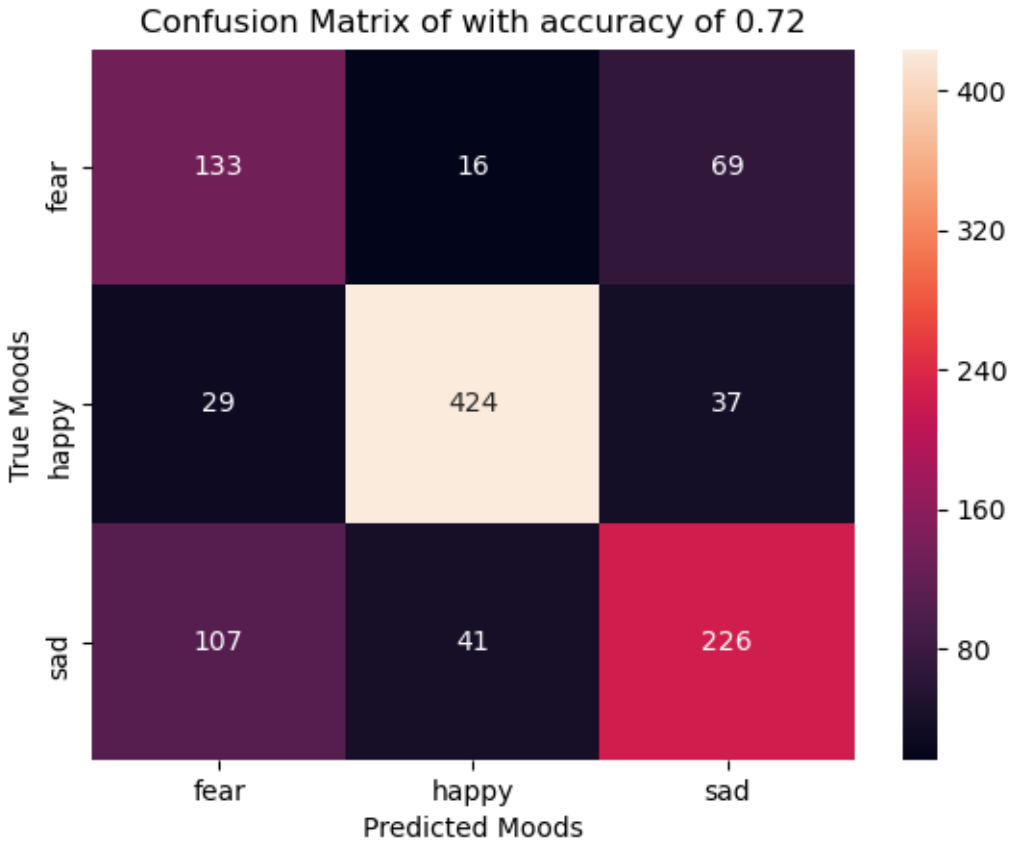
Since our epoch is 50, it measured validation in 50 steps, and plotted the graph.

We can observe that the accuracy of training is reaching upto 95% and the validation reaching around 70%.

Also we noted loss graph is decreasing down to 0 in training but down to 80% in case of validation.

We saved the weights of train model in file model.h5. For testing we imported the weight and used.

We found the accuracy of the model as 72% and the confusion matrix as printed as below.

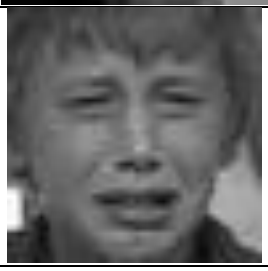


Hence **our model work with accuracy of 72%** (approx as it may change due to randomness of training). And detect most of the expression correctly.

Also we found overlapping of sadness and fear are close enough.

CONCLUSION.

Our application of face detection is ready with accuracy of 72%. Some of the results are plotted below:

Sl. No.	Actual Image	Predicted Image	Emotion
1.			Fear
2.			Fear
3.			Happy
4.			Happy
5.			Sad
6.			Sad