Facial Expression Recognition

Team Members:

Omar Ismail(Leader)

Vineeth Reddy Sheri

Piyush Narhire

Story

A Human being can easily sense another person's emotion, but it is a very difficult task for the machines to understand. Person emotions consist of Fear, Anger, Happy, Surprise, Sad, Neutral and Disgust. If a machine was able to detect these emotions and take necessary actions it would be of huge success like preventing people from making some rash decisions, enlighten the mood. For instance, in a classroom environment, an emotion detection model will help in generating a report of students' emotions which will help the lecturer to improve the upcoming lecturers to attract more students.

An web-based application where the user can upload an image or take a picture through a webcam. So how do we detect emotions in an image? We are going to make use of an open-source data set — Face Emotion Recognition (FER) from Kaggle. The model consists of two modules one is Face detection and the other is Emotion detector. Face detection will detect the faces in the image and pass these identified faces to the Emotion detector module to identify the persons emotion.

Workflow

The data we utilized for the project is the FER2013. The dataset contains many facial RBG images of size 48*48 over different Expressions such as Fear, Anger, Happy, Surprised and Neutral. We are making use of Google Colab as the IDE as it provides a free GPU. Setting up the IDE to load the data FER2013 from the Kaggle. Performing preprocessing tasks on the dataset like normalizing the data and splitting the train dataset in 4:1 ratio for training and validation phase.

Defined the custom model which contains 11 convolutional layers and 5 fully connected layers. Introduced Batch Normalization layer to deal with covariate shift and dropout layer to avoid overfitting of data. Used Softmax at the output layer since dealing with more than 2 categorical data. The model contains 9,019,719 parameters out of which 9,012,295 trainable parameters. Trained the model for 100 epochs with learning rate as 0.001, Adam optimizer and loss function as categorical cross entropy. Making use of ReduceLROnPlateau callback function to reduce the learning rate by a factor of 0.9 when there is no improvement in the validation loss for 3 epochs, EarlyStopping callback function to stop the training process when the validation loss doesn't decrease for the last 6 epochs and a ModelCheckpoint callback for Storing the models trained parameters when there is a improvement in the validation loss. After training for 100 epochs,the training accuracy of the model is 92%, validation accuracy is 97% and testing accuracy is 67.5%.

Using a pre-trained MobileNetV2 model for transfer learning with FER2013. MobileNetV2 model is pretrained on the ImageNet dataset, these pre-trained weights act as starting to train our model which enables the model to train faster and reach the desired point quickly. Stripping the output layer of the MobileNetV2 and using custom Softmax layers since dealing with only 7 categorical data. The model contains 3,538,984 parameters out of which 3,504,872 trainable parameters. Trained the model for 50 epochs with learning rate as 0.001, Adam optimizer and loss function as categorical cross entropy. Making use of ReduceLROnPlateau callback function to reduce the learning rate by a factor of 0.9 when there is no improvement in the validation loss for 3 epochs, EarlyStopping callback function to stop the training process when the validation loss doesn't decrease for the last 6 epochs and a ModelCheckpoint callback for Storing the models trained parameters when there is a improvement in the validation loss. After training for 60 epochs, the training accuracy of the model is 95%, validation accuracy is 90.2% and testing accuracy is 66.4%.

MTCNN(Multi-task Cascaded Convolutional Neural Networks) to detect the human faces in the image and passing the identified face to the trained model for detecting the emotion. Drawing a box around the face of a person and displaying the person's emotion. Testing the model on random images and the model is performing pretty good.

The next step will be to use the model to classify expressions in real time using a webcam.

Model Architecture

Custom Model

Layer (type)	Output Shape	Param #
conv2d_22 (Conv2D)	(None, 46, 46, 64)	1792
conv2d_23 (Conv2D)	(None, 46, 46, 64)	36928
batch_normalization_28 (Batc	(None, 46, 46, 64)	256
max_pooling2d_8 (MaxPooling2	(None, 23, 23, 64)	0
dropout_14 (Dropout)	(None, 23, 23, 64)	0
conv2d_24 (Conv2D)	(None, 23, 23, 128)	73856
batch_normalization_29 (Batc	(None, 23, 23, 128)	512
conv2d_25 (Conv2D)	(None, 23, 23, 128)	147584
batch_normalization_30 (Batc	(None, 23, 23, 128)	512
conv2d_26 (Conv2D)	(None, 23, 23, 128)	147584
batch_normalization_31 (Batc	(None, 23, 23, 128)	512
max_pooling2d_9 (MaxPooling2	(None, 11, 11, 128)	0
dropout_15 (Dropout)	(None, 11, 11, 128)	0
conv2d_27 (Conv2D)	(None, 11, 11, 256)	295168
batch_normalization_32 (Batc	(None, 11, 11, 256)	1024
conv2d_28 (Conv2D)	(None, 11, 11, 256)	590080
batch_normalization_33 (Batc	(None, 11, 11, 256)	1024
conv2d_29 (Conv2D)	(None, 11, 11, 256)	590080
batch_normalization_34 (Batc	(None, 11, 11, 256)	1024
max_pooling2d_10 (MaxPooling	(None, 5, 5, 256)	0
dropout_16 (Dropout)	(None, 5, 5, 256)	0
conv2d_30 (Conv2D)	(None, 5, 5, 512)	1180160
batch_normalization_35 (Batc	(None, 5, 5, 512)	2048
conv2d_31 (Conv2D)	(None, 5, 5, 512)	2359808
batch_normalization_36 (Batc	(None, 5, 5, 512)	2048

```
conv2d_32 (Conv2D)           (None, 5, 5, 512)         2359808
_________________________________________________________________
batch_normalization_37 (Batc (None, 5, 5, 512)         2048
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 2, 2, 512)         0
_________________________________________________________________
dropout_17 (Dropout)         (None, 2, 2, 512)         0
_________________________________________________________________
flatten_2 (Flatten)          (None, 2048)              0
_________________________________________________________________
dense_13 (Dense)             (None, 512)               1049088
_________________________________________________________________
batch_normalization_38 (Batc (None, 512)               2048
_________________________________________________________________
dropout_18 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_14 (Dense)             (None, 256)               131328
_________________________________________________________________
batch_normalization_39 (Batc (None, 256)               1024
_________________________________________________________________
dense_15 (Dense)             (None, 128)               32896
_________________________________________________________________
batch_normalization_40 (Batc (None, 128)               512
_________________________________________________________________
dropout_19 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_16 (Dense)             (None, 64)                8256
_________________________________________________________________
batch_normalization_41 (Batc (None, 64)                256
_________________________________________________________________
dense_17 (Dense)             (None, 7)                 455
=================================================================
Total params: 9,019,719
Trainable params: 9,012,295
Non-trainable params: 7,424
_________________________________________________________________
```

Work Sharing

Omar Ismail - Used transfer learning to train model

Vineeth Reddy Sheri - Training and testing custom model, configuring colab to load FER2013 dataset from kaggle

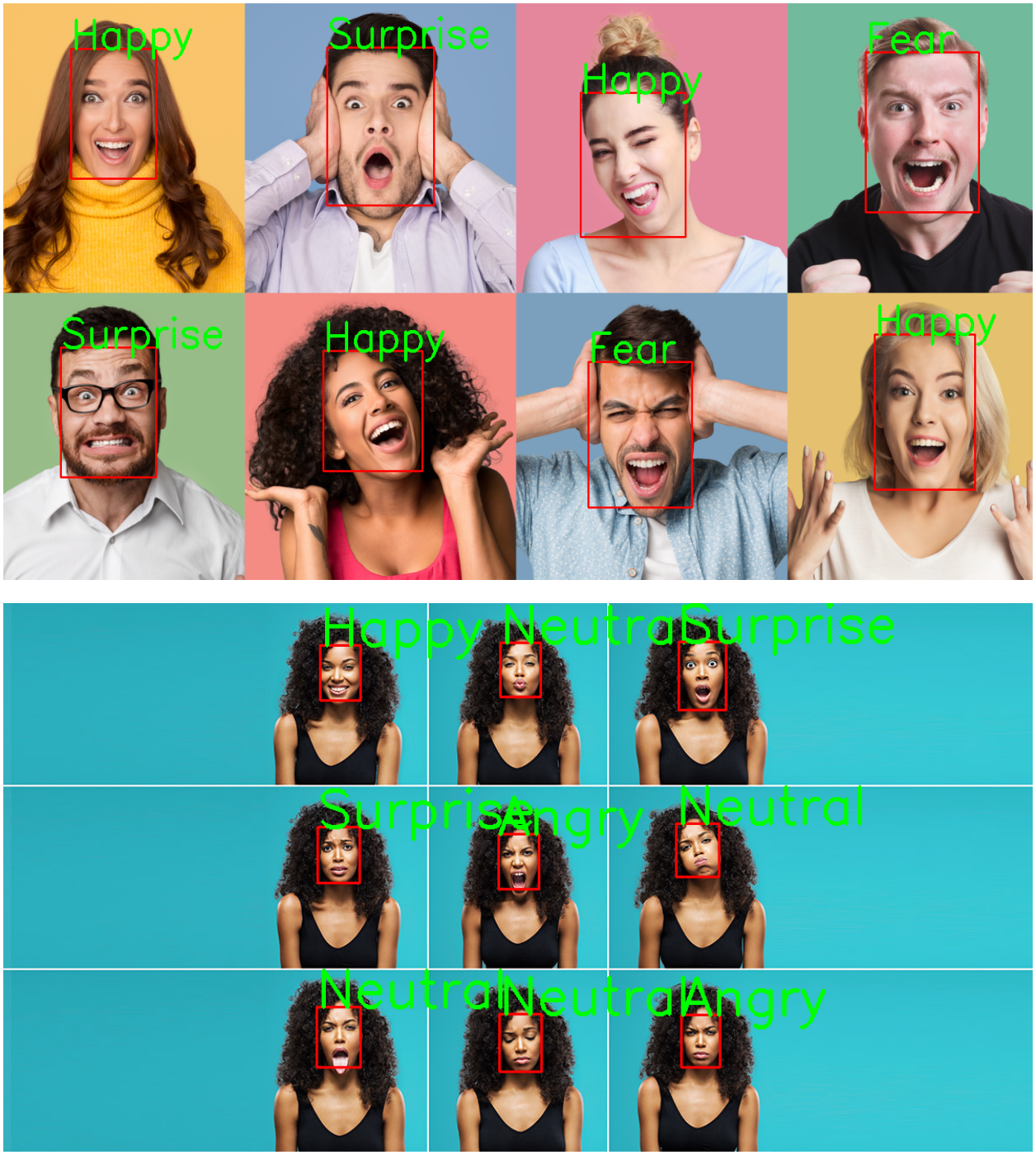
Piyush Narhire - Face detection module

Working screens from the project

Testing Model performance on random data

Custom Model

MobileNetV2 Model



The performance of the custom model is better than the MobileNetV2 model.

Issues or blockage with the project

The model is giving incorrect accuracy value when using the accuracy_score function. The value is alway around 0.17. Even when tested through the Training dataset, the accuracy is 0.17.

Github link: https://github.com/OmarIsmail7980/Facial-Expression-Recognition

References

Research paper on Facial Emotion Recognition:

https://ieeexplore.ieee.org/document/8734924

https://ieeexplore.ieee.org/document/8549735

https://ieeexplore.ieee.org/document/9299256

https://ieeexplore.ieee.org/document/9239720

Google Colab configuration for using Kaggle datasets:

https://towardsdatascience.com/setting-up-kaggle-in-google-colab-ebb281b61463