ML Challange (YouBotics)

Emotion Detection

PROJECT REPORT

Nilesh Sonawane
August 28, 2020

Problem Statement

Process an image of a person using an ML model served through an API, to predict the emotions of a given person.

Approach Overview

Build a two-stage emotion predictor:

1. **Face Detector**: Used OpenCV implementation of object detection using Haar feature-based cascade classifiers to detect single or multiple faces in a given image.
2. **Emotion Detector**: Tried various techniques of transfer learning, but to gain the best results have built a CNN based model to classify images based on the emotion of the person in the given image.

All experiments are done in google colab, ipynb notebook with the name “Colab_Emotion.ipynb” is added in the repo.

Dataset

Used a ready dataset containing grayscale ROIs of faces with different emotions available on Kaggle here. All images are stored in 7 folders. Emotion folders present are as follows, angry, disgust, fear, happy, neutral, sad, and surprise. The task is to predict how much is the person happy, sad, confident, or neutral. I was unable to find data for confident class. As observed, there are many incorrectly labeled images present in the given dataset. To achieve the results based on the given task, the following folders are discarded/deleted (not included while fitting model): angry, disgust, and fear.
Image distribution among the folders:

- **Train**
  - Happy: 7164 images
  - Neutral: 4982 images
  - Sad: 4982 Images
  - Surprise: 3205 Images
- **Validation**
  - Happy: 1825 Images
  - Surprise: 797 Images
  - Neutral: 1216 Images
  - Sad: 1139 Images

All images are grayscale in nature and are of size (48,48) each.

Face Detector:

I have used OpenCV's Haar Cascade Classifier to detect human faces. This OpenCV's implementation is trained on many images with positive points(with face) and negative points(without face) labels. The detectMultiscale method returns a list of BBox(Bounding box) coordinates values for all detected faces in the same image.

Emotion Detector:

Tried to use transfer learning. Approach:

- ResNet50: Used resnet architecture which has 50 layers with residual blocks for alternate layers for entire architecture. Resnets help in gradient flow and hence is one of the best techniques for image classification. Due to the reason that data has many incorrectly labeled images present in it, resnet tends to overfit the model by achieving high training accuracy and low validation accuracy. Hence discarded this method. Validation accuracy after 100 epochs was 0.71
- Similarly tried few other networks too, like a deeper resnet, ResNet101, InceptionV3, MobileNet, etc. All these models had same issues of overfitting the data.
- To avoid overfitting, I have build a 7-layer CNN based network. Chosen less number of layers to avoid the overfitting problem and to regularize the weights properly.

Data Loader:

Prepared the training data by dividing into batches of batch size 32 using TensorFlow's ImageDataGenerator. (batch_size as 64 and 128 were tried as well, 32 performs the best as per the observation. Model optimized quickly in the case of 32 as batch_size). Performed Data Augmentation such as scaling, rotation, shear, zoom, and horizontal flip using TensorFlow's ImageDataGenerator.

Network Architechture:

CNN layers:

- Conv2D_1: 64 (3,3) filters, BatchNorm layer, relu activation, MaxPooling with (2,2) filters, and 25% dropout.
- Conv2D_2: 128 (3,3) filters, BatchNorm layer, relu activation, MaxPooling with (2,2) filters, and 25% dropout.
- Conv2D_3: 256 (5,5) filters, BatchNorm layer, relu activation, MaxPooling with (2,2) filters, and 25% dropout.
- Conv2D_4: 256 (3,3) filters, BatchNorm layer, relu activation, MaxPooling with (2,2) filters, and 25% dropout.

Single Image size after this layer is: (3,3,256)

Flattening layer: size here is a 2304 sized vector.

Fully Connected Layer:

- Dense_5: 1024 neurons with batchnorm, relu activation, and 30% dropout
- Dense_6: 256 neurons with batchnorm, relu activation, and 30% dropout
- Dense_7(Softmax layer): 4 neurons(number of classes) with softmax activation, to return log probabilities representing each output emotion.

Training

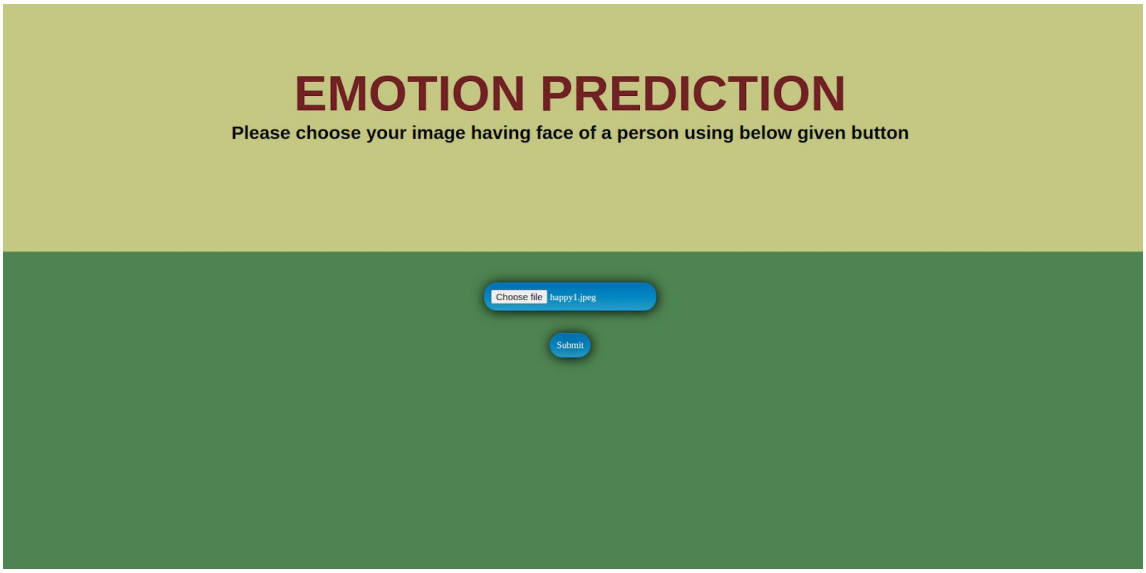
- Compiled the model using Adam optimizer and categorical_crossentropy as loss function, and ‘accuracy’ as metrics to evaluate the model after each epoch.
- Trained the model on training data present in train folder with 4 classes up to 100 epochs, achieved results after 100 epochs:
    - Training acc: 0.88
    - Training loss: 0.32
    - Validation acc: 0.8048
    - Validation loss: 0.61

API - Flask REST API:

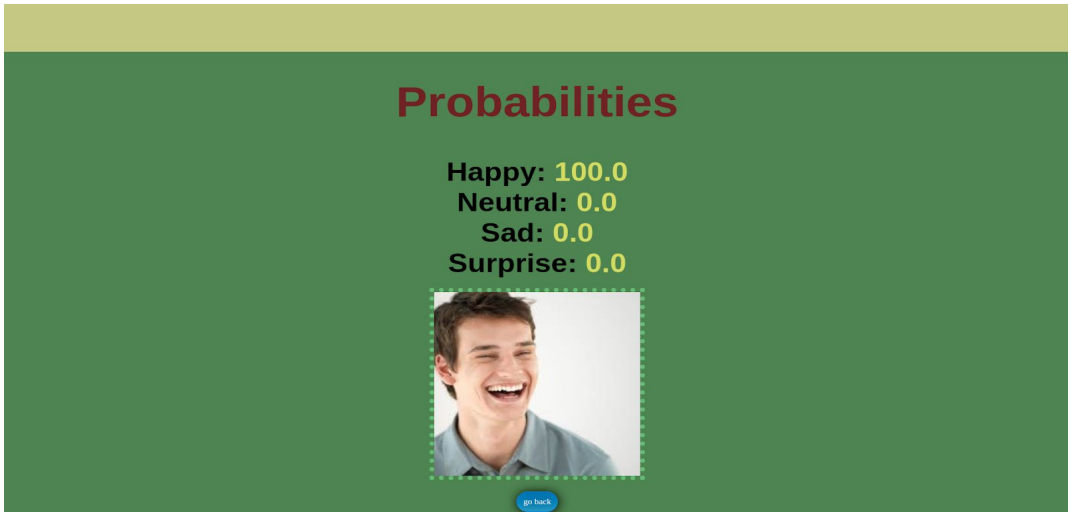
A simple web app is made to serve the model through the API to fulfill the motive of providing the model with a random image of a person and output the emotion percentage.

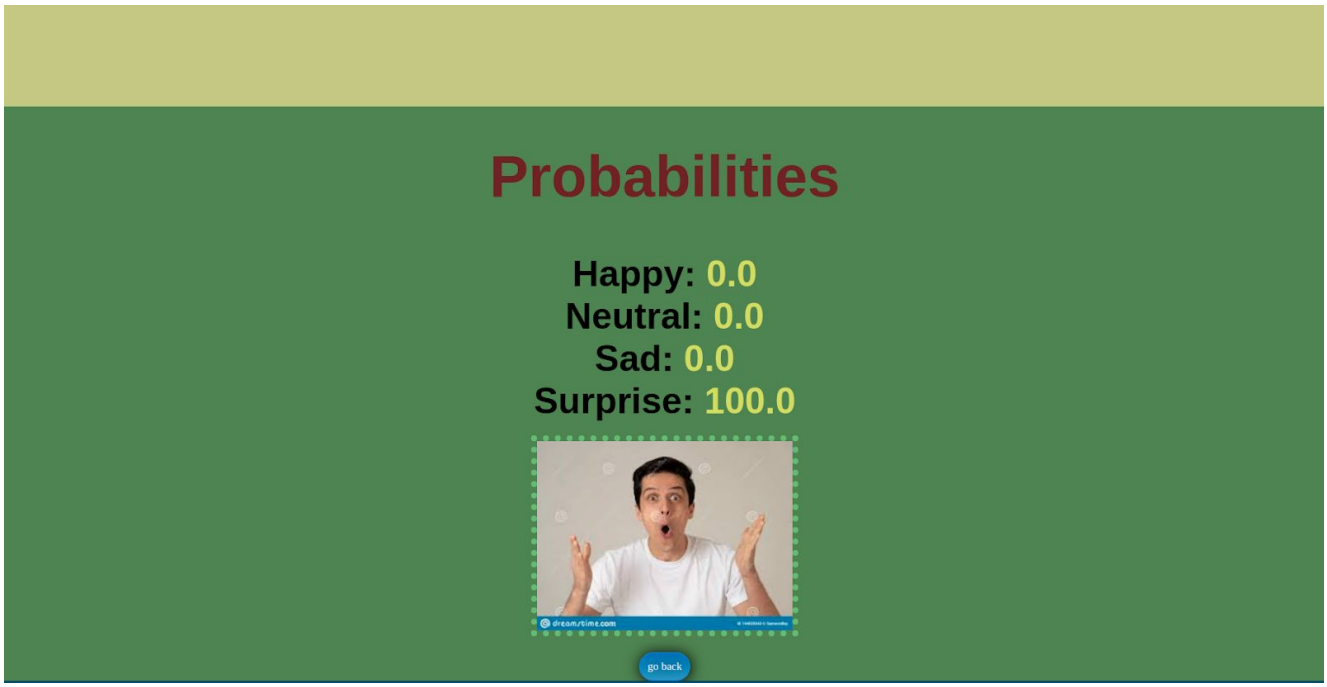
Results:

- Web app interface to inject image in the model served through API



- Sample Predictions.





Conclusion:

A very very low level and simple emotion predictor is made. A lot can be done, some of the excellent papers can be implemented(have searced for some papers, will surely try to implement), improved dataset will be a huge plus point. Few other points such as training on the better dataset, training for a longer time, and deeper networks with properly chosen hyperparameters and few other techniques will surely help in better models and their predictions.