CMP304 AI Part
Project Report (50%)

Gwydion Saxelby Mood Recognition Application

by: Gwydion Saxelby 1701267

1. Introduction (5%)

End Goal and Users

The goal of this project was to create a desktop application that could assist people with Autism in being able to more easily identify people's moods in photos. The completed project would work by analyzing photos of peoples faces and presenting the user with a predicted mood and a percentage of certainty about the prediction. This prediction would enable them to more confidently respond to people's pictures online where they otherwise might have had difficulty interpreting the mood of the photo.

Identifiable Moods:

The completed program will be able to recognize the 6 base emotions that can be displayed by the human face: Sadness, Happiness, Disgust, Anger, Fear and Surprise.
These moods are primarily distinguishable by subtle changes in the subjects' mouth, lips combined with eyes and eyebrow position.
These changes are easily identified by the human eye but would prove the crux of the project in the context of Computer Vision where an image is boiled down to a series of binary values.

2. Methodology (15%)

Powering the Project:

The desktop application was written using Python and Tkinter (a python library for building GUI applications). Python is a powerful general-purpose programming language often used to power websites and to program machine learning applications.
I chose it for this project because it is well documented, open source with an active community of developers and has a strong support for machine learning and computer vision libraries such as OpenCV and TensorFlow.

Vanilla Python doesn't have the functionality necessary to build Computer Vision applications such as the proposed mood recognizer. For this I have chosen to use the OpenCV library. OpenCV was chosen because it is Opensource, well documented and has an active community of developers and has been successfully used in this type of project countless times over. The alternative option to OpenCV was TensorFlow. TensorFlow is another opensource library and in many ways is more powerful than OpenCV as it provides support for more in-depth machine learning and neural networks. However, for the purposes of this project this extra functionality wasn't required and the additional complexity that TensorFlow required would have slowed development.

The Datasets:

The program will use supervised learning to identify the moods in its photos. For supervised learning to be effective the system needs a dataset of labeled images that it can use to train itself on what facial characteristics make up the different moods. The dataset I chose for this task was the Cohn-Kanade + Dataset (CK+).

I chose the CK+ dataset as my training dataset because it is a large dataset containing quality images that represent a wide range of facial emotions: Anger, Contempt, disgust, fear, happiness, neutral, sadness and surprise. This application doesn't require us to identify Neutral moods or Contempt as contempt is a more specialized form of disgust, so I removed those collections from the dataset. The CK+ dataset was supplemented with a selection of mood images found using google images to provide it with a more varied training model that would hopefully help it identify real world images more effectively.

For testing the program, I decided to use a random selection of sample images from the Multimedia Understanding Group Database (MUG) and google images. The MUG dataset is another freely available dataset. However, it is smaller than the CK+ dataset and the moods it captures aren't as well defined as those in the CK+. I decided to use it because it was important that model testing used images that were not included in the training set as this would provide the program with an un-realistic test as it would be analyzing images it was already trained on.

Image Pre-Processing:

It is important that the images in the training dataset and the images about to be processed by the program are processed prior to the mood prediction to reduce the amount on un-helpful and possibly distracting 'noise' in the images. It is important that as many non-mood features of an image are removed prior to training and evaluation to ensure that the program doesn't get confused and start to make mood connections to elements that have nothing to do with the mood such as a person's freckles or the colour of the car behind them or colour of their clothing etc.

Image pre-processing for my applications images underwent 3 phases:

Phase 1: Apply Bilateral Filter:

The first stage of pre-processing was to smooth the images. This involved applying a bilateral filter to the images. A bilateral filter takes a large amount of the finer detail in a photo and removes it. The end result of this filter technique is a flatter looking photo with more clearly defined edges to its features such as eyes and lips. Enhancing these features and removing any non-mood related details such as freckles or texture from the image helps the training algorithm focus on only the intended features.

Phase 2: Convert Image to Grayscale:

The next stage was to convert the colour image into grayscale format. Most normal images are taken in RGB or BRG format (Colour) these images are nicer to look at and provide more detail through colour, but they present an extra dimension of complexity. The training technique used in this application required an input image to be in grayscale format, so this step was essential. However, even if the trainer could take in RGB or BRG colours I would still have processed them into grayscale because it has the benefit of removing further non-mood related details from the images. This will could prevent a bias in the application where it starts to associate colours with specific moods.

Phase 3: Face Detection:
The 3rd face in the applications image pre-processing has been to run the images through several face Classifiers called "HAAR Cascade Filters" (HAAR). It would be possible to train a custom face Detection classifier. However, the pre-trained ones provided as part of OpenCV work well enough for this application.
Each image in the application is ran through 4 pre-trained HAAR filters twice using different parameters to maximize the chances of finding a human face in the image. Once the face is detected. The image is cropped and then saved again to remove as much of the image as possible that is not a human face displaying an emotion. This further reduces the amount of 'noise' in the images and improves the chances of an accurate mood prediction.

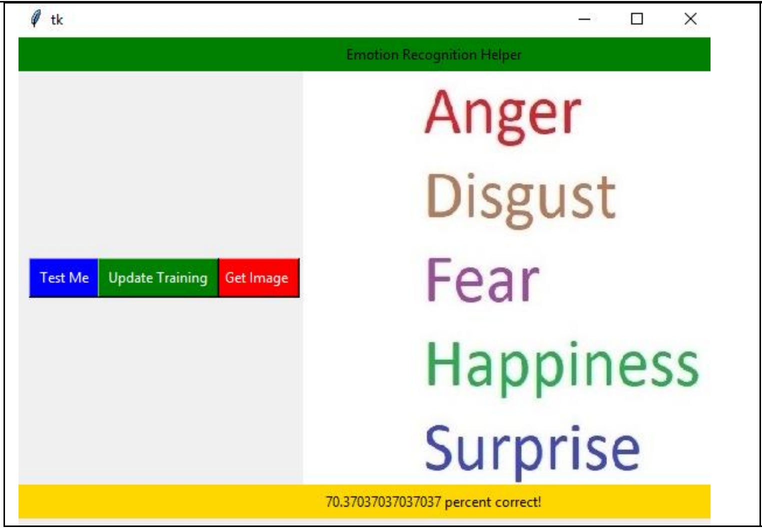
Training and Recognizing:

To perform the applications Image Training and predictions OpenCV provides several "Face Recognizer" algorithms that can be trained to identify items in an image: FisherFace, Eigenfaces, and LocalBinaryPatternHistograms (LBPH). I tested the application using all 3 provided algorithms and found FisherFaceRecognizer performed the best. The Fishers algorithm was invented by Sr R. A. Fisher in 1936 where he trained it to Identify different types of Iris flowers however can be trained to classify any image such as moods in this project.
Once the categorization training had been completed the trained model was programmed to be saved so that future recognition tasks could be performed quickly as the algorithm does not need to re-train itself every run through. This massively improves the speed and usability of the application.

3. Results (10%)

Automated Testing of Dataset Accuracy:

The applications accuracy was tested using a random selection of images from the MUG image Database. These images were moved into a separate "Training Dataset folder" and were labeled before being ran through the application and the percentage of accurate predictions returned:

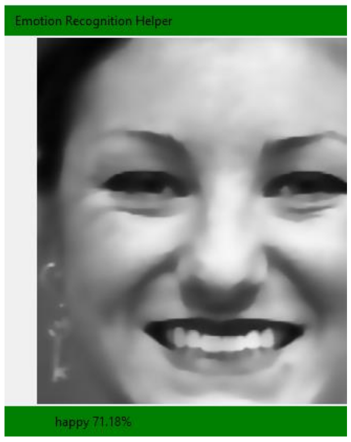
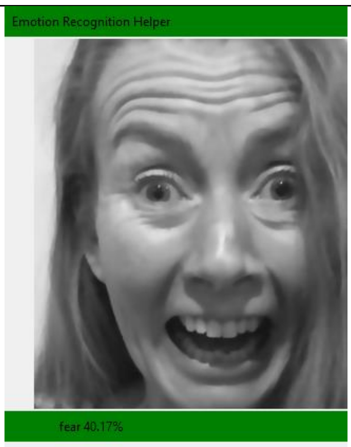
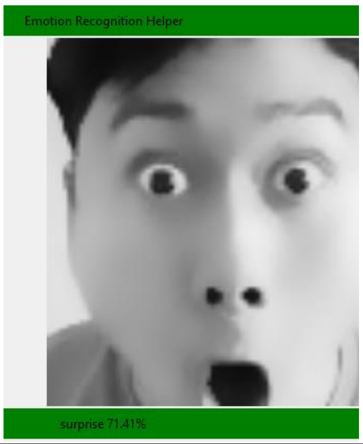
 tk Emotion Recognition Helper Test Me Update Training Get Image Anger Disgust Fear Happiness Surprise 70.37037037037037 percent correct!	The application performs acceptably well during automated testing.

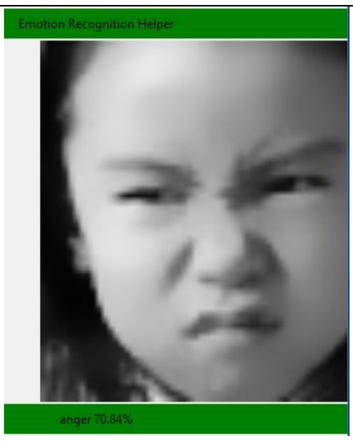
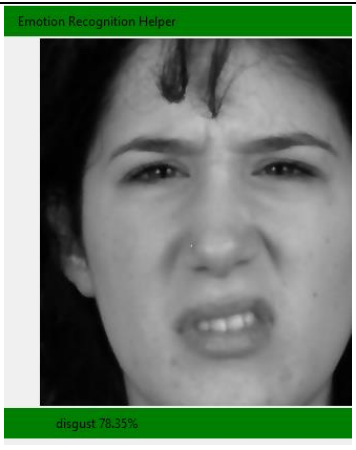
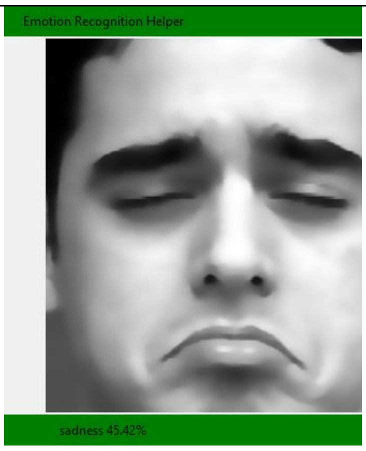
Testing the application like this gives us a fast means of analyzing how effectively the application works as the programming and pre-processing can be modified, and the different percentage of correct predictions quickly analyzed without any changes to the training or testing dataset. Changes would reduce the credibility of the test as I wouldn't be sure if it was the images or processing that was affecting the results.

<u>Manual Testing Results</u>:

Returning a percentage of accuracy to the user for manually selected images (such as those selected during normal running of the application) presented a problem because none of OpenCVs computer vision recognition algorithms returned a percentage of probability. Instead they returned a "Distance" along with the prediction. This distance is representative of how far away from what the model considers to be a "definite mood" the image is.

To turn this 'Distance' into a percentage probability for the user I had to calculate what the threshold distance for the current dataset was.
This was achieved by modifying the automated tests to return the average distance as well as the average score. From there I was able to calculate what distance represented 1%, 100% and anything in between.

 happy 71.18%	 fear 40.17%	 surprise 71.41%

4. Conclusion (10%)

Future Improvements:

The project meets its brief as a desktop application capable of predicting people's moods from photos. There are however several improvements that could be made to its practical usability.

The program is written in python and is limited to desktop environments. It would be more helpful to its users if it was re-written in a more popular format such as a mobile application. This way its users could take it on the go and get mood predictions on people they interact with while out.
A second shortcoming of the program in its current state is that it is only capable of interpreting still images. It could be improved by adding support for live video processing so that the user could see someone's current mood as it changes throughout their conversation.

Final Notes:

Although something we take for granted in our daily lives emotion recognition is a complex task. This is especially true for computers and machine learning where all there is to interoperate are ones and zeros.

The project can be considered a success because the application completes its core task to a reasonably high level of accuracy considering the complex nature of the task and the lack of context or additional information it has access to in performing its predictions.
The application only has access to a relatively small number of images to teach it to recognize moods and these images have no context or sound to accompany them. Comparing this to the amount of audio-visual interactions your average adult has had over the cause of their lifetimes the applications training dataset bairly scratches the surface of possible ways to identify

people’s moods.
Even with that limitation If the applications training was totally ineffective and instead it guessed the moods at random we would see a success rate of only 1/6 ~ 15%. So, for it to achieve scores as high as it does with this relatively very small amount of training data is a huge success.

5. References (5%)

Reference	**Source**
OpenCV	https://docs.opencv.org
Tensorflow	https://www.tensorflow.org/
PaulVangent Recognition Example	http://www.paulvangent.com/2016/04/01/emotion-recognition-with-python-opencv-and-a-face-dataset/
Python	https://docs.python.org/3/library/tk.html
MUG Dataset	https://mug.ee.auth.gr/fed/
CK+ Dataset	https://ieeexplore.ieee.org/document/5543262
Iris Dataset	https://archive.ics.uci.edu/ml/datasets/Iris