Facial Expressions Identification

ECS 171 Machine Learning

Group 8 Project Report

Group members:

Joshua Baylasy, Julio Flores, Yiyang Huo, Anthony Huynh, Kenneth Lieu, Jorge Menjivar, Jong Chan Park, Ricardo Sun, Jing Xie, Michael Yang, Harry Yong

Github repository:
https://github.com/michael-yang2/ECS171-Facial-Recognition

Introduction and background

Facial expression is one of the most significant non-verbal means of communication and social interaction between individuals. As humans are social species, we can understand one's thoughts or emotions by observing facial expressions. In situations where an individual is presenting a particular facial expression, it is possible to understand how they're feeling through analyzing their reaction to something or their emotional status or wellbeing at a given time. The ability to identify one's emotion through facial expression proves to be an important ability since it can be more informative than verbal communication in certain situations, especially when verbal communication is unavailable or unrevealing of much information.

Facial expression identification through machine learning is a field subject to inherent error due to variable lighting, positioning, and other factors of facial images. Additionally, the problem is difficult even by human standards- many have trouble discerning emotion from the face alone. As a result, models will struggle with achieving high accuracy.

If a concrete and reliable classification model is successfully built, it can be utilized in many different fields. Here are a few examples:

- Clinical and psychological purposes
  - Assisting individuals with developmental or mental disorders such as autism spectrum disorder and schizophrenia, who often struggle with recognizing emotion from facial expressions. A reliable model can be used to offer training to said individuals to help them build the skills needed to recognize nuances in facial expressions. In turn, a model can also teach proper responsive emotions, thus helping individuals to understand certain social situations better.
- Criminology and forensic psychology
  - Assisting investigations by identifying facial expressions during an interrogation of suspects, comparable to the outcomes of a lie detector.
- Human-Computer Interaction (HCI)
  - Assisting the consumer and market research by measuring advertisement efficacy and customer satisfaction. With a reliable model, their findings can be further advanced and contribute to better products or recommendations.
- Political Science
  - Detecting emotions of politicians during their public appearances or speech.

Literature review

Machine learning has several applications to the problem of emotion recognition and the training of models in research adds to the ongoing efforts in emotion classification and better understanding of various features in visual and audio data. Presented in this section are examples of related work done by researchers.

Han et al. investigated the problem of speech emotion recognition by utilizing deep neural networks in order to extract high level features from data which are revealing of their importance for emotion recognition. They achieved an unweighted accuracy of 48.2% and a weighted accuracy of 54.3% using their DNN. Additionally, more similar to the work presented in this paper is the work done by Ng et al. They used the FER2013 dataset to pretrain their convolutional neural network model

as the model they used to compete in the 2015 Emotion Recognition in the Wild contest. Ng et al. used their model to classify the emotions of faces in a much smaller dataset in order to evaluate the performance on different sized datasets. They achieved an overall accuracy of 55.6% in the test set through fine-tuning their CNN models using the auxiliary dataset and an accuracy of 48.5% in the validation set. The accuracies of the models in these works, though not very high, are promising. These accuracies serve as a reference point for our experimental results and are able to indicate how well our models do in comparison to past works.

Dataset Description and exploratory data analysis of the dataset

The dataset being used in this project is the fer2013 dataset in the form of a CSV file. There are 35887 rows and three columns in the dataset: the emotion, the pixels, and the usage. The emotion is encoded as a number from zero to six which represents the emotions anger, disgust, fear, happiness, sadness, surprise, and neutral respectively. The pixels are arrays of numbers representing the images through their pixel values in the format of a string. The size of the arrays is 2304, suggesting that the images are 48 by 48 pixels. The images are centered on faces and are in greyscale already, thus each item of the array represents how black that specific pixel is. The usage specifies the purpose of the entries being either for training (80%), public testing (10%), or private testing (10%). Some sample images from the dataset as well as the distribution of the images based on their corresponding emotions are shown below.

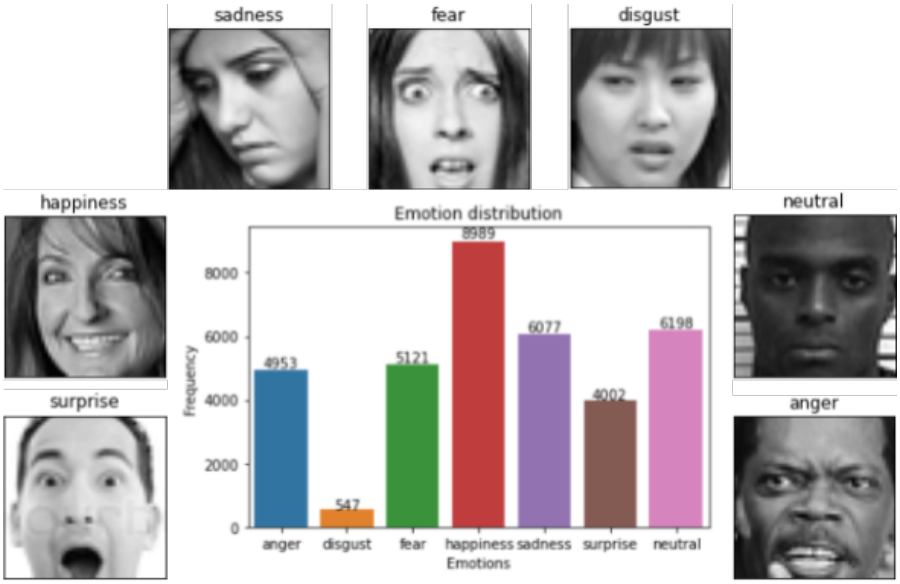

Figure 1. Distribution of emotion and sample images

Happiness is the most abundant emotion in the dataset with 8,989 instances, while disgust is the least with 547 instances. The other emotions are relatively evenly distributed between 4,000 and 6,200 entries. It is important to note that the amount of data points for disgust is drastically lower than the rest of the emotions, which may affect our results.

There are a few reasons for choosing this dataset for our project. Since our goal is to predict emotions

based on images of facial expressions, a dataset of human faces is best suited to our project. Fer2013 is a famous dataset that many of the related papers used when testing out their models. Notably, it "has more variation in the images, including facial occlusion (mostly with a hand), partial faces, low-contrast images, and eyeglasses" (Minaee, Minaei & Abdolrashidi, 2021, p. 5). Also, because the images contain only the faces and are already in greyscale and are in the perfect size, they do not require much preprocessing.

Proposed Methodology

This problem is a multi-classification problem. However, since we are dealing with images, the normal Deep Neural Network will not suffice. From the papers that we studied, we concluded that Convolutional Neural Network (CNN) and Support Vector Machine (SVM) are the most popular models for image-related problems. In the end, we came up with three different models. Aside from the CNN and SVM, we also developed a K-Nearest Neighbor (KNN) for an extra solution.

While we were figuring out which model to use, we also started to develop our applications for the project. In order to use our model, we needed an interface for user input. Additionally, our model is trained on grayscale images cropped to focus on the face, but our user input will likely be full color images, and may not be focused in the same way. We needed another layer of processing to format the input images into a form that our model is comfortable with.

OpenCV provides both the utility to crop the face from the original image with CascadeClassifier and cvtColor to turn the image into grayscale. These two functions allow us to process user input appropriately. With that, we can actually start to develop the applications.

Our initial plan was to create a web application that allows the user to upload an image that contains a person's face and get a prediction of the emotion back. After that, we expanded our scope and developed an application that could recognize facial expressions and classify them into emotions in real time, using an interface that uses a camera to accept live video. This was surprisingly simple. Every few frames, we capture an image of the feed, process the image like normal, and allow our model to predict the emotion. With this interface, users are able to get the emotion prediction outputs faster and the result will update constantly as long as the video is still playing.

Experimental results

CNN:

	precision	recall	f1-score	support
0	0.58	0.63	0.60	491
1	0.64	0.51	0.57	55
2	0.54	0.41	0.47	528
3	0.92	0.81	0.86	879
4	0.54	0.56	0.55	594
5	0.72	0.80	0.76	416
6	0.59	0.71	0.64	626
accuracy			0.66	3589
macro avg	0.65	0.63	0.64	3589
weighted avg	0.67	0.66	0.66	3589

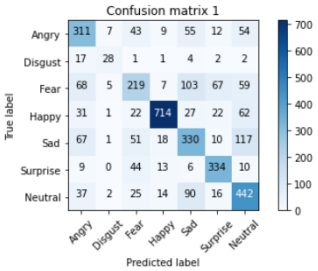

Figure 2. Testing result of CNN

For CNN, we built the mode based on the sequential model from Keras. The CNN is generally a very complicated model with all kinds of possible combinations of layers. We started with three modules, each with two Conv2D layers followed by a Batch Normalization layer and a MaxPooling2D layer. The kernel size of Conv2D was set to (3,3) with ReLU activation function, and the number of filters started from 256 and was halved with each module, ending at 64 filters. The MaxPooling2D used pool size of (2,2) and strides of (2,2). After the Flatten layer, we had three modules of Dense layer each followed by a Batch Normalization layer. The Dense layers also used the activation function of ReLU, and started from 512 filters, which was halved with each module, ending at 128 filters. The final output Dense layer had 7 filters, and used softmax as the activation function.

The Conv2D layer is the center of this model that applies filters to the input. The Batch Normalization layer is meant to accelerate the training, while the MaxPooling2D layer is meant to reduce overfitting and computational load. The Flatten layer is used to reduce the dimensionality of the output from preceding layers down to 1 to match the input shape requirement of the Dense layers. The Dense layer plays the role of accepting all of the outputs from previous neurons and applying a vector-matrix multiplication with its filters.

For hyper-parameter tuning, it was impossible to do a fully organized Grid Search since our computational power for us was very limited. The model's average training time is above 14 hours, so we were only able to train a few combinations of the parameters. We tested various modifications of the parameters individually. We reduced the number of filters in Conv2D and Dense layers, used a swish activation function, added regularizations of L1 and L2 in the Cov2D layers, and added Dropout Layers to reduce overfitting and outliers. Unfortunately, each of these changes resulted in a worse performing model.

Looking at the classification report and the confusion matrix, the best model has a testing accuracy of 66%. We can see that the model has the tendency to confuse sadness with fear and neutral with sadness. This may be caused by the kinds of images in the dataset - many images have faces covered by hands to show sadness or fear. Visually, the neutral faces are also quite similar to some of the sad faces. Although the accuracy on predicting the disgust emotion is not too bad, it may affect the accuracy of the model when it deals with other images of disgust because of the lack of data.

SVM:

Classification report SVM with RBF

	precision	recall	f1-score	support
0	0.45	0.23	0.31	496
1	1.00	0.06	0.11	52
2	0.40	0.23	0.29	533
3	0.48	0.72	0.57	909
4	0.35	0.40	0.37	607
5	0.61	0.51	0.56	404
6	0.40	0.42	0.41	588
accuracy			0.44	3589
macro avg	0.53	0.37	0.37	3589
weighted avg	0.45	0.44	0.43	3589

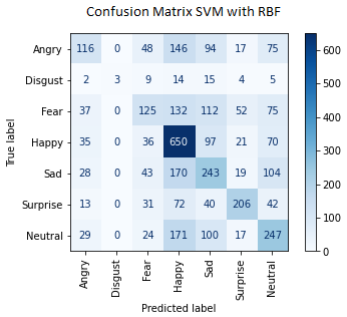

Figure 3. Testing result of SVM

For SVM, we used sklearn's SVC function to create all of the SVM models. We started by creating a single model for each Kernel option, and set the rest of the parameters to the default values of the SVC function. RBF had the best accuracy at 44%, and the polynomial kernel had 43% accuracy. The rest of the kernels had accuracies below 40%, with the linear kernel having the lowest accuracy, at

35%. Since the difference between the RBF kernel and polynomial kernel was only 1%, we decided to fine tune the other parameters for both of the kernels. We started with the Polynomial first because it has the degree of the function as an extra parameter. The original Polynomial model we trained used a degree of 3, so we tested new models with higher degrees.However, increasing the degree did not improve the accuracy. Our initial RBF and Polynomial models were the best two models we would create as changing any of the other parameters only decreased the accuracy of the model. Interestingly, most SVM models were more likely to predict happiness than any other emotion, and would avoid predicting disgust. SVM models take relatively little time to train compared to the other models- the fastest SVM took about 30 minutes to train, and the slowest SVM took about 2 hours to train.

KNN:

	precision	recall	f1-score	support
0	0.37	0.31	0.34	467
1	0.36	0.50	0.42	54
2	0.41	0.40	0.40	522
3	0.46	0.41	0.43	930
4	0.38	0.35	0.36	617
5	0.62	0.58	0.60	401
6	0.31	0.44	0.36	598
accuracy			0.41	3589
macro avg	0.42	0.43	0.42	3589
weighted avg	0.42	0.41	0.41	3589

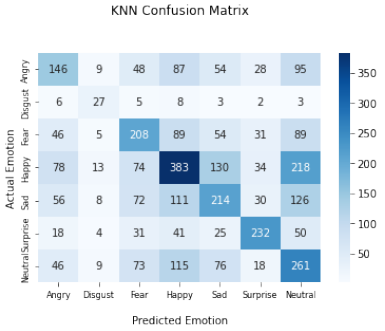

Figure 4. Testing result of KNN

Multiple KNN models were tested for this experiment. Hyperparameter tuning was done using gridsearch. The parameters used in gridsearch were leaf_size, n_neighbors, and p. The highest KNN accuracy was 41% and it was achieved when n_neighbors = 1 and p = 1. However, this model frequently confused Happy and Neutral faces.

Software Implementation:

After training three models, CNN, SVM, KNN, with a relatively high accuracy, we implemented an app that can capture the video stream of the webcam and detect the emotion per frame. As mentioned before, our models can only classify the emotion of a face picture, and relies on other software to provide face detection. In order to complete our app, we implement such structure in our software:

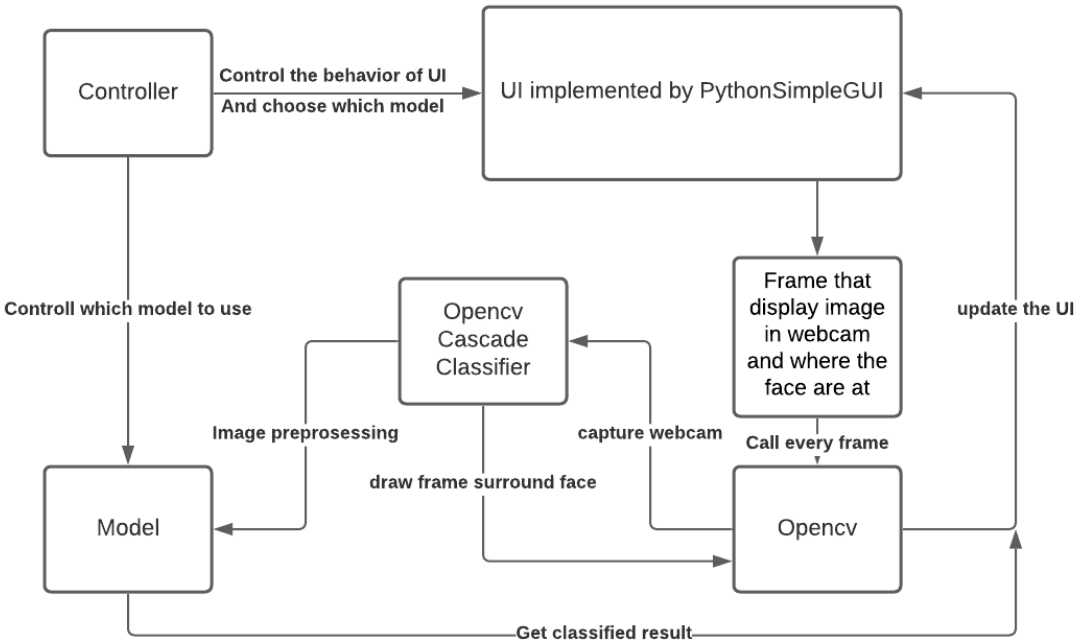

Figure 5. Implementation of the software

We use the built in opencv cascade classifier to capture the face and draw the frame to surround the face in the opencv captured picture. Then, we crop the image to 48*48 with a normalized gray scale. Then, we do the prediction and update the UI implemented by PythonSimpleGUI library.

For future implementation of the software, someone asked about a post training process that adjusts the pretrained model with the face of a specific person in order to improve the accuracy of detecting the emotion of that person. We are thinking about adding the post training process to the software by asking the specific person to express some emotions in order to improve the post-training dataset. A more complicated but feasible approach is to recognize the face as nodal points in the pre-training process and identify the specific person's facial nodal points in the software. In that case, the accuracy for detecting a specific person's emotion can also be improved.

Conclusion and discussion

Through using the dataset 'fer2013', which contains images of 6 different facial expressions and their emotion labels, and 3 different models (CNN, SVM, KNN), we were able to build a model with 66% accuracy with using Convolution Neural Network (CNN) with three sections.

Since 'disgust' data had relatively small amounts of data compared to other emotions, we could have improved the accuracy of the model by oversampling the data of disgusted faces, or removing the 'disgust' data from the model. However, it would limit the scope of our model in real world situations, so we decided to keep the label. Retaining the label allows us to add more data into our model and gradually improve its accuracy.

For software implementation, we added the face detection to our model using the opencv cascade classifier, which captures the face from the picture. The opencv cascade classifier process the image into a normalized grayscale with the size of 48*48, which then our model predicts its emotion based on the image given. We can further improve the model by post-training a dataset, in which we can ask subjects to make facial expressions with specific emotions.

References

Dataset:

https://www.kaggle.com/msambare/fer2013

Face Recognition:

https://realpython.com/face-recognition-with-python/

Literature Review:

Han, K., Yu, D., & Tashev, I. (2014, September). Speech emotion recognition using deep neural network and extreme learningmachine. Interspeech 2014.

Ng, H. W., Nguyen, V. D., Vonikakis, V., & Winkler, S. (2015, November). Deep learning for emotion recognition on small datasets using transfer learning. Proceedings of the 2015 ACM on international conference on multimodal interaction (pp. 443-449).

Minaee, S., Minaei, M., & Abdolrashidi, A. (2021 April). Deep-Emotion: Facial Expression Recognition Using Attentional Convolutional Network. Sensors 2021, 21, 3046. https://doi.org/10.3390/s21093046