Emotion Detection through Facial Recognition

Mikel Hermiz
University of Michigan-Ann Arbor
mikelh@umich.edu

Surya Krishnan
University of Michigan-Ann Arbor
suryakr@umich.edu

Ketaki Gaikwad
University of Michigan-Ann Arbor
ketakig@umich.edu

Vishwa Patil
University of Michigan-Ann Arbor
vishwap@umich.edu

1. Introduction

The emotion detector is a tool that could help in a wide range of areas from machines being better able to understand their user to helping the medical field lead to a better illness diagnosis. With so many facial expressions that people are able to make it can be difficult to make out which emotion it is associated with. However a constant that many people have gotten accustomed to and is now commonly used to express emotion are emojis. Easily integrated within our keyboards emojis have a consistent and easy to understand meaning. Our goal with this project is to get an accurate program to help users better identify facial expressions.

We have chosen this project because of the many applicable aspects not only to our life but also those who surround us. When initially talking about this project many of us discussed how difficult it was telling the expression of our mutual friend. Then we got to thinking how hard it must be for computers to do the same thing if our own human brains couldn't do it. There is a widely accepted facial expression for each emotion however not everyone is able to recognize them and fully understand the intended emotion.

On top of the relevance that we saw in our life for this project there was also a more personal connection for one of our members. Alexithymia is the clinical term that is known for inability to recognize emotion. Around ten percent of the world's population is diagnosed with this condition ranging in level of severity [1]. Many people with this condition have a harder time integrating with society and can oftentimes feel secluded from their peers and the rest of the world. Through our project we hope it is easier to diagnose this condition and help people get adjusted quicker.

We attempted to solve this problem through the use of a simple yet effective implementation. Through the use of machine learning and convolutional neural networks we were able to train the network using a database which includes images that are labeled with their respective emotion. To obtain the best accuracy we utilized a variety of databases to see which one obtained the best results. Face detection was not always perfect and if given more time we would have added in more training data with different hyperparameters to improve our validation. With these added improvements the emotion detector would prove to be a useful program that could help many different people.

2. Related Work

There are many examples that have done emotion detection through facial recognition using a multitude of different methods. One of the first ones we looked at when starting this project was a report from the Islamic University Madinah [3]. Their approach utilized histogram-of-oriented gradients, feature extraction, and support vector machines to classify into seven main facial expressions. From our research we saw that this method worked very efficiently but was only accurate for the emotions that are easier to distinguish. This was a very helpful report to help us get started.

Initially planning to implement convolutional neural networks we decided to look and see if there were similarly experiments done using this method. One implementation that matched was the FERC which is a two step convolutional neural network which first removes the background noise from the image [4]. Resulting in a ninety six percent accuracy it is a very effective program but very resource and time intensive. This was a great example to refer to as we went along the way and scaled it down to fit within our scope and time frame of the overall project.

3. Approach

To begin, a flow chart of our approach to solving this problem can be seen in Figure 1 below. In this figure you can see there are two parts that we need to solve before combining them together to create our overall solution. First of all, we had to conduct preliminary research into CNNs and its properties in order to grasp a better understanding of how we want to construct our model. Secondly, we have to refactor the dataset we are using which is the RAF dataset and then design and

implement a convolutional neural network from scratch to detect emotion [5]. Thirdly, is to use an off the shelf CNN for face recognition and use that to determine what part of the image we need to input to the emotion detection CNN. Finally, we use the emotion detection CNN to make a prediction on the emotion from that face and replace the face on the image with an emoji representing the emotion.

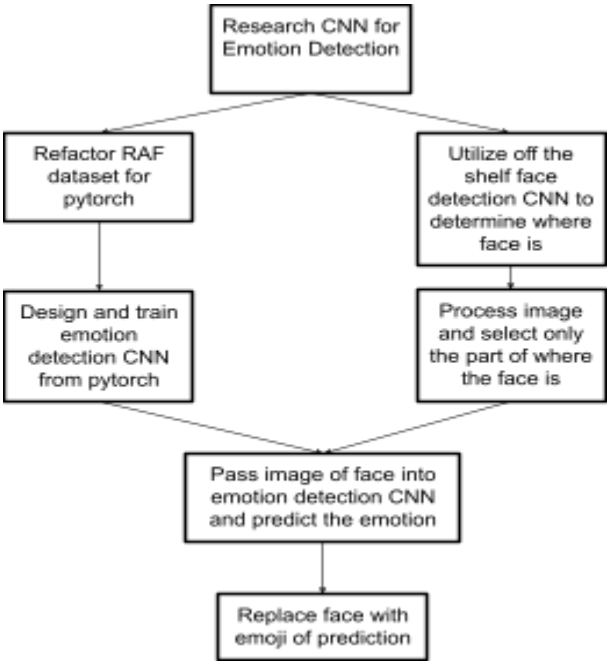

Figure 1.

Before embarking on our CNN implementation, we decided to conduct some initial research on the properties of a convolutional neural network and the impact of constructing an architecture that achieved a high accuracy. Specifically, a CNN utilizes filters to stride over images and extract certain features and compares them to the true label of the image, in this case the true emotion. We understood that the 2 main aspects of creating a successful CNN relied on tuning hyperparameters and choosing ones that yielded the highest test accuracy and creating a robust, sophisticated architecture that didn't overfit.

The hyperparameters consisted of weight decay, which adds a penalty term to loss functions to reduce overfitting and prevent the model from becoming too complex, and learning rate, which controls how much the weights of the neural network update. For weight decay, the greater it is, the less likely it is to overfit, and for learning rate, the greater it is, the more changes and updates the model makes to its weights.

For constructing a sophisticated CNN architecture, we ventured into what makes it up, which included many different types of layers. These layers were regular 2D convolutions, a batch normalization layer, which standardized the input and reduced the number of iterations required to achieve a high accuracy rate, and max pooling layers, which chose the most prominent features in each patch of the image. In addition to these two components of a successful CNN, we also looked at which loss and optimizer functions to use. The most optimal ones we found for the purposes of our application were the Adam optimizer and Cross Entropy Loss due to their high prevalence and performance on CNNs in general.

As a result, we understood that utilizing a convolutional neural network to accomplish the task of classifying images was the best option due to past attempted solutions and their high accuracy rates.

The final architecture is composed of 8 2D convolutional layers, 8 2D batch normalization layers, 2 max pooling layers, 8 ReLU layers, and a fully connected layer. All of these layers had the same kernel size and stride, which was 3x3 and 1, respectively, except the max pooling layer which had a kernel size of 2x2 and a stride of 2. We decided to use a kernel size of 3x3 and stride of 1 because this preserved greater dimensionality and didn't reduce in size much due to other downsampling methods, such as max pooling. The channel path went from 3 -> 32 -> 32 -> 64 -> 64 -> 128 -> 128 -> 256 -> 256. This path between the input and output channels ensured that the model would retain a high accuracy rate. Between each convolution layer we had a batch normalization and relu. Also, the first two convolutions had a max pooling layer after them. Lastly, we input 256 channels of 14 by 14 layers to a fully connected layer which outputs a 7 by 1 vector of predictions for the 7 emotions we had.

For the second part of our method we had to refactor the RAF dataset to be used to train our emotion detection CNN [5]. The dataset was 15,539 images labeled with 7 different emotions which included: Surprise, Fear, Disgust, Happiness, Sadness, Anger, and Neutral. We had to process this data and split it up into pieces where we had training, validation, and testing data. We put about 10,000 images for training, 2,500 for validation, and another 2,500 for testing. After this was split up we put the data into the Pytorch data loader so it can be ready to be inputted into our CNN.

Next, the off the shelf CNN we used for face detection was the MTCNN from Pytorch [2]. This CNN would be able to return the corners of a face in an image. This allowed us to be able to detect where the face is in the images we take and then pass that part of the image to the CNN we made for emotion detection.

After we determine where the face is and pass it into our CNN to make a prediction on what emotion that face is displaying, we replace the area of the image we determined was a face with an emoji representing that emotion.

Additionally, we saw that the emoji image was a box with a black background, however the face doesn't exhibit this structure and shape. So, by using the corners of the face that was detected by the MTCNN we were able to filter out the dark background of the emoji and make the emoji have a good fit on just the face and not impact the rest of the image.

Lastly, to get our images taken in google Colab we utilized some code that was given in Colab to access the camera and take a picture so we can use to process for our purposes.

4. Experiments

We decided to test our model on the RAF dataset because it had substantial information about various emotions that would help capture the complexity and breadth that our robust model needed in order to achieve a high accuracy test score. Additionally, the RAF dataset consisted of 1000s of images, so there was enough images to train our model on, test it on, and also perform k-fold cross validation, which was another metric of understanding the skill set of our model by splitting some of the data into k groups and testing the model on each of there groups to decrease bias towards one patch of the dataset.

We decided to measure our success through the test accuracy rate that was seen by running the model on Google Colab. We utilized the validation accuracy and test accuracy given to us by Pytorch since it displayed the correct amount of images that were classified correctly. Some potential downsides of utilizing solely accuracy are that it could be misclassifying an image as the right emotion when in reality it exhibits characteristics of another emotion. A possible metric to help with this is in the future is to generate a confusion matrix of true positives, false positives, true negatives, and false negatives and derive the F1-Score, which is the mean of precision and recall. This metric gives more information about a dataset when there is an uneven class distribution, such as a high concentration of false positives which produce misclassifications so we could improve it more in the future..

From training our model on hyperparameters, such as learning rate and weight decay, we were able to deduce which set of parameters yielded the highest accuracy. The best set with which we achieved this was having a learning rate of 1e-4 and a weight decay of 1e-6. This corresponded to the general trends understood in our research that low learning rates and weight decays result in a cohesive and complex model that predicts emotions well. We displayed these variations in a table and ran 3 trials with 4 different sets of the weight decay and learning rate to grasp a wide breadth of possible hyperparameters that would result in the model to yield different accuracies as seen in Figure 3 that will be discussed later.

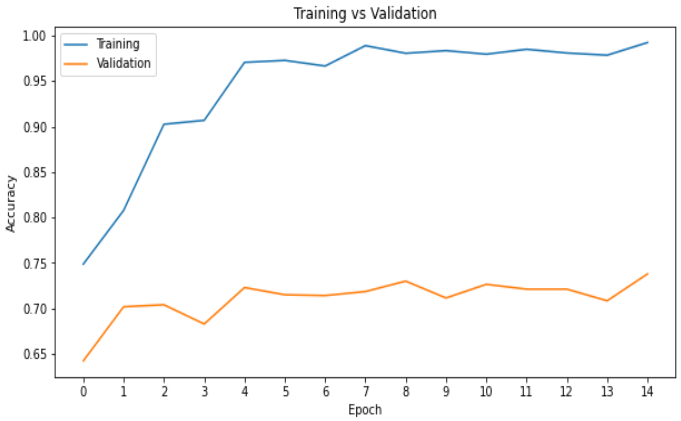

Figure 2.

The graph from Figure 2 describes the relationship between the validation and training accuracy rates. From the figure, it can be deduced that throughout the 15 epochs that the model ran on, the training accuracy was higher than the validation accuracy. This results in the behavior we want since we didn't want the model to exhibit bias towards the training dataset and have it underfit when we performed testing on the validation set. However, even after several implementations this overfitted model was providing us with the best test accuracy compared to the rest of the architectures we designed and tested. Of course it is usually not good to overfit a machine learning model and if we had more time we could have implemented and tested more models to get one that predicts the testing data set even better without overfitting the training data.

Learning Rate	Weight Decay	Accuracy
0.001	0.00001	0.71414
0.001	0.000001	0.71805
0.0001	0.00001	0.70990
0.0001	0.000001	0.72555

Figure 3.

The relationship that can be deduced from the data table in Figure 3 is the learning rate and weight decay had inverse correlations with the accuracy. When the learning rate was 0.001 and the weight decay was

0.00001, the model yielded an accuracy of 71.4%. However, once the learning rate decreased to 0.0001 and the weight decay decreased to 0.000001, the accuracy increased to 72.5% on average. We even had a high accuracy rate of 76.2% on the test data set for one of our runs with the hyperparameters of 0.0001 for learning rate and 0.000001 for weight decay.

From Figures 4 and 5, the results of the model are qualitatively shown. In Figure 4, it can be seen that the human is exhibiting characteristics that can be correlated to a happy emotion. So, as a result, the model predicts "Happiness" based on this. However, there were some mispredictions that occurred which were false positives. In Figure 5, although the human is exhibiting an emotion that can most strongly be correlated with surprise, the model predicts "Neutral". The reasoning for this lies within the data itself since it may contain a distribution of data that contains images which are skewed towards more positive emotions. What we deduced from this is that the model didn't train itself on equal distributions of positive or negative emotions, which led to some overfitting and mispredictions which can be seen.

Overall, we believe that our emotion detection CNN had a great start and was giving us decent accuracy when testing it. However, there definitely needs to be improvements done so we can reduce overfitting, and decrease the skew to more positive emotions when trying to detect emotions. However, with an average accuracy of 0.72 and a high of 0.76 we are really happy with what we have designed and built because it is still doing way better than random chance.

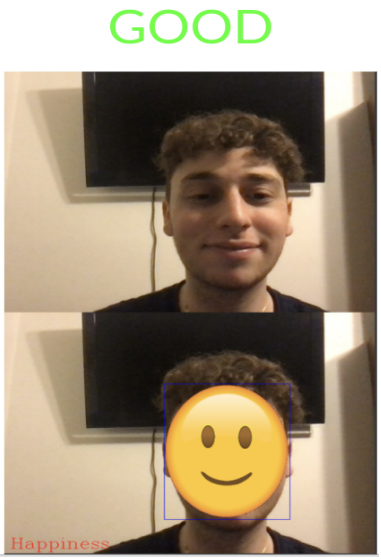

Figure 4.

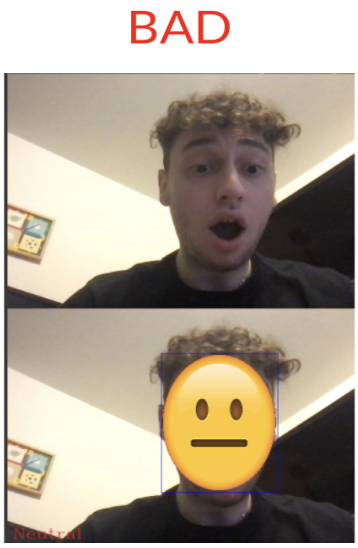

Figure 5.

5. Implementation

As for what we implemented in this project, we designed and built a CNN for emotion detection. Additionally, we processed and formatted the datasets for our CNN which were the RAF datasets [5]. Additionally, we refactored homework 5 code for our training purposes. Now for what we used from others was a face detection CNN called the MTCNN to detect faces [2]. We coded everything else regarding the outputs after where the face was detected and we processed and figured out how to place the emoji where the face was. Lastly, we used some google colab code that was already in google colab as a snippet to be able to access the camera in colab.

6. Conclusion

The emotion detector is a great starting point for us and others to build off of and expand this proof of concept design. Resulting in our highest accuracy of seventy six percent there are some areas of improvements that still need to be implemented. Moving forward we would like to experiment with different architectures within convolutional neural networks while changing the hyperparameters to improve validation and testing accuracy. Also by adding more variety to our training set we could more accurate results that best represent the world we see around us. Overall we are pleased with our results and think this is a great outcome for our initial network. This program is very beneficial and can be used for a multitude of different areas. We hope that with this technology it will lead to better diagnoses through psychological clues and help those who have a difficult time identifying emotions through facial expressions.

7. References

[1] D. Serani, “The emotional blindness of alexithymia,” Scientific American Blog Network, 03-Apr-2014. [Online]. Available: https://blogs.scientificamerican.com/mind-guest-blog/the-emotional-blindness-of-alexithymia/#:~:text=The%20clinical%20term%20for%20this,what%20others%20feel%20and%20think. [Accessed: 26-Apr-2022].

[2] J. Güse, “Face detection using MTCNN - a guide for face extraction with a focus on speed,” Medium, 17-Aug-2020. [Online]. Available: https://towardsdatascience.com/face-detection-using-mtcnn-a-guide-for-face-extraction-with-a-focus-on-speed-c6d59f82d49. [Accessed: 26-Apr-2022].

[3] M. Asad, S. Gilani, and M. Jamil, “Emotion Detection through Facial Feature Recognition,” International Journal of Multimedia and Ubiquitous Engineering, vol. 12, no. 11, 2017.

[4] N. Mehendale, “Facial emotion recognition using convolutional neural networks (FERC) - SN applied sciences,” SpringerLink, 18-Feb-2020. [Online]. Available: https://link.springer.com/article/10.1007/s42452-020-2234-1. [Accessed: 26-Apr-2022].

[5] “Real-world affective faces database,” Real-world Affective Faces (RAF) Database. [Online]. Available: http://www.whdeng.cn/raf/model1.html. [Accessed: 26-Apr-2022].