6-2020

Human Facial Emotion Recognition System in a Real-Time, Mobile Setting

Claire Williamson
Union College - Schenectady, NY

Human Facial Emotion Recognition System in a Real-Time, Mobile Setting

By

Claire Williamson

* * * * * * * * *

Submitted in partial fulfillment

of the requirements for

Honors in the Department of Electrical, Computer and Biomedical Engineering

Union College

March, 2020

Abstract

The purpose of this project was to implement a human facial emotion recognition system in a real-time, mobile setting. There are many aspects of daily life that can be improved with a system like this, like security, technology and general safety.

There were three main design requirements for this project. The first was to get an accuracy rate of 70%, and this rate must remain consistent for people with various distinguishing facial features. The second goal was to have one execution of the system take no longer than half of a second. This keeps it as close to real time as possible. Lastly, the system must maintain user privacy by not saving any of their images for training.

To accomplish the goal within the constraints of the design requirements, a neural network is used. The network has two layers. The first layer has 512 nodes and the second has 7 nodes. The first important step was to run and save a model that contains the weights for the network, which occurred on Google Colaboratory. When this was completed and saved, the network is placed into a local folder that works as the main directory for the project. I used PyCharm to edit my code, and terminal to run it. The system works locally on a laptop by capturing an image with an external camera connected by USB, then manipulating that image to be a grayscale, 48 by 48-pixel image, when it can be sent in through the network. The system provides a best guess as to what the user's emotion is and prints it on the screen.

In the end, the system was successful in recognizing the user's emotions 57.27% of the time. The entire process runs continuously, and a photo is taken roughly once every half second.

Table of Contents

Figures and Tables

Figures

Tables

Introduction

One of the major ways that humans convey emotions is through facial expressions. Likewise, we are capable of reading others' emotions by analyzing these facial expressions. Similar to the process in which we learn factual information, humans continuously refactor their knowledge of the distinct ways that the face expresses various emotions. There are seven basic emotions that humans express, which are anger, sadness, fear, disgust, happiness, surprise and neutrality. When we are angry, our eyebrows often furrow and the lips might purse. Happiness is indicated by smiling and wide eyes. We continuously become better at recognizing these key characteristics that identify emotions by updating our memory of these tendencies, then by using this in the future to recognize a similar emotion or to contrast an unknown.

We want computers to attain human-level performance. When the eye sees an image, it processes and compares it to previous images that we have seen. From here, the image can be analyzed and an emotion is determined. In a similar way, we can have a computer analyze an image and compare it to a database about which it has previously been trained. Then, the computer can determine the emotion of the face in the image within a reasonable amount of precision. There has been a lot of work in this field during recent years developing various systems that are successful. Some systems work well, but some are either not as accurate as desired, not as fast, or both.

The goal of this project is to develop a mobile system capable of continuously predicting user emotions. A system like this could reliably provide things like security and entertainment to the public in many different ways. For example, recognizing emotions of passersby in an airport could flag certain individuals who look suspicious. It also could be used to create an app that mirrors a person's current emotion and responds in a helpful way. There are endless possibilities of what emotion recognition could be used for, but this high level of usability significantly decreases without mobility, or being able to move the system instead of being fixed in one place. If someone were to mass manufacture a system and sell it to airports, mobility would allow the system to still be successful in each distinct airport.

There are factors that increase the difficulty of developing a system like this. It needs to have a stable camera to be able to read clear images, or have a level of tolerance for blur. There could be certain features that block part of the user's face, which decreases performance level. These obstacles and their resolutions, as well as the design of a successful emotion recognition system, are detailed and further explored in this report.

Background

Historical and Societal

In the fourth century, the study of physiognomy first began, which is the study of a person's character based on their outside appearance [1]. Later, in the 17th century, people started looking at head movements and how they relate to facial expression [1]. Charles Darwin studied and concluded that there are basic emotions that people of all cultures and ethnicities show. However, it was not until 1978 when Paul Ekman developed a way to encode the face and its expressions, the Facial Action Coding System (FACS) [1]. Later in 1978, M. Suwa examined expressions using computers and wrote programs that were able to extract distinguishable features from faces [1]. These systems continued to progress throughout the 1990s, focusing mostly on various ways of feature extraction like geometric features in static images, their combination with texture features, and using single global classifiers as opposed to multiple [1].

One way in which systems that are reliably capable of recognizing human facial emotions would be beneficial is for security. Say a young child is lost in a crowded area like a mall. They are going to look fearful and sad. If a system within the mall can track this child's emotion and recognize it as fearful and sad, it can automatically try to correct the problem, like notifying security.

Although applications in this field would be quite useful, current progress is slow and involves artificial occlusion and its effect on system performance. For an exhaustive look at recent work with occlusion, see Table 2 in Zhang's paper, *Facial Expression Analysis under Partial Occlusion: A Survey*

[1]. The table provides a summary of many researchers in the emotion recognition field and the details of their work including the kind of classifier, emotions it is able to detect, and accuracy rates.

Seven Basic Human Emotions

There are seven basic human emotions that humans are capable of expressing, which include anger, sadness, fear, disgust, happiness, and surprise [2]. Neutrality is often added to categorize when a face shows no particular emotion. Each of these emotions is detectable from specific characteristics for that emotion. For example, happiness is indicated with a smile, while surprise has raised eyebrows. Examples of these seven emotions are shown below in Figure 1.

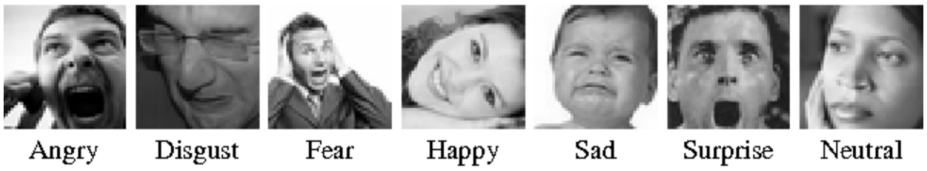

Figure 1. Seven Basic Human Emotions

We use these seven emotions as the output for the system, so it can decide one of these to predict for a user's image. The combination of basic emotions gives us more specific emotions, but those are more difficult to classify, so we remain focused on the basic emotions [1].

Neural networks

Neural networks are used to recognize patterns by classifying the data into numerical values according to similarities. They are loosely modeled after the human brain and the neurons within the brain [3]. Each "neuron," or node, is a mathematical equation that computes a weight for a connected edge [3]. A general model of a single neuron, as well as a node it is analogous to, can be viewed in Figure 2.

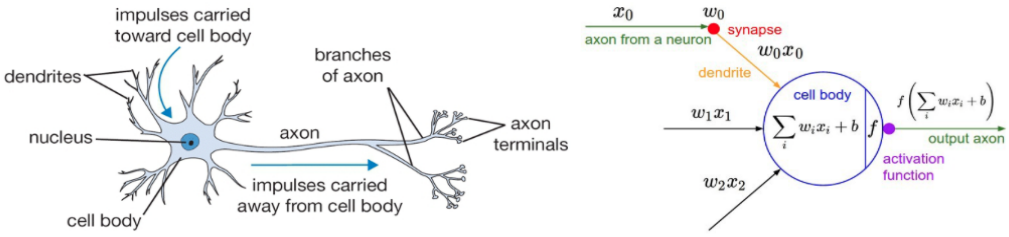

Figure 2. Human neuron and related neural network node [1]

The same way that the axon terminals in the neuron, like on the left in Figure 2, transmit and output its own form of data, the node outputs a mathematical calculation according to its inputs and weight, like on the right in Figure 2 [3]. A sample overall structure of a neural network can be seen in Figure 3.

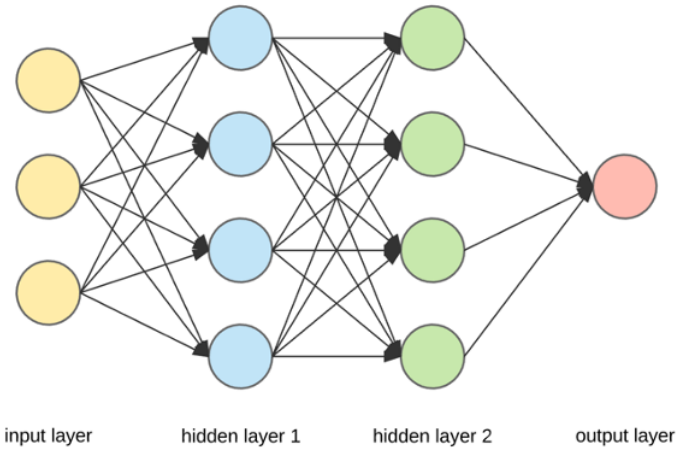

Figure 3. Deep Neural Network Structure [4]

The nodes are grouped into different layers, and each layer is connected to the next layer by connecting each node in one layer to every node in the next layer. In Figure 3, the different layers are differentiated by different color nodes. The structure of Figure 3 is called a deep neural network on which I elaborate in the next subsection [5]. The network uses the weights calculated between nodes for each subsequent layer to decide which node to travel next [3].

Network training, which is the continuous updating of the weights of the network based on observed data, is done to improve the network's ability to match inputs to outputs [3]. The network gives a guess or an approximation to output based on that training within some varying window of confidence [3].

There are two types of learning that can be used in training, supervised and unsupervised. Supervised learning uses labels to map features to certain labels [3]. In this case, that means using the known emotion tied to an image to map that image's distinguishing facial features to its given emotion. These mappings must be analyzed along with the dataset to find a relationship between the label possibilities and the data [3]. The network uses this relationship that it detects to adjust the weights

between layers of the model [3]. Unsupervised learning groups or clusters the data based on similarities detected rather than assigning them to a label [3].

Deep Neural Networks

Deep neural networks, like the structure of the network in Figure 3, are networks that have 3 or more layers, including the input and output layers [3]. Each layer has a certain number of nodes, and each node represents a computation in the network. The nodes combine inputs with coefficients, or weights, which assign significance to inputs [5]. Deep neural networks can be described as a "multistep process of pattern recognition," where more steps and more training allows for more complicated features [3]. This concept of increasing training to detect more specific features is called feature hierarchy [3]. In other words, as the number of steps in the training process increases, the network becomes better at identifying more complex facial features.

Training vs. Testing

When we are training the model, this means that we use a previously chosen database to learn features of facial emotions and how they relate to specific emotions. Testing, on the other hand, is the response of the already trained system on new images that it has never seen before. It is possible to test on images that the system is trained on, which is how we obtain the results in Table 2 of the Preliminary Test Results section.

Another important process is validation. This involves taking a small subsection of data out of the training set and putting it into a validation set. Then, after training is concluded, we can use this validation set to test data that we know is similar to the training set and determine its accuracy.

Design Requirements

Performance

The product should be able to correctly determine human facial emotions 70% of the time. When testing the system on the same images it was trained on, we saw a success rate close to this rate, and we want images that are not previously trained on to have similar success. More information about previous success rates can be seen in Table 2. In real time, we aim to see the success rate decrease no more than 10%.

Platform Independence

The software can be implemented on many different platforms. This is accomplished by using Google Colab for training, which is detailed in the Design section. Moving from one platform to another should be simple and straightforward, or well documented.

Speed

In order to maintain its real time performance, we want the image to be processed and the system to output a prediction in roughly 300-500 milliseconds. Assuming the camera has an average frame rate of 30 frames per second, this means that one frame is taken every 33 milliseconds. We add some time for processing the image and outputting a guess and allow a window of error for faster and slower frame rates and execution times, which gives us the 300-500 millisecond window. Any slower than this will result in a delay that we could no longer classify as real time.

Privacy

Images in real time cannot be saved to the training database in order to protect the privacy of the user. During training, the network uses a set and unchanging database of images. This database must be sufficient enough to detect the seven basic emotions without needing to retrain on user-captured images. This means that we cannot continuously train the model with user images.

Usability

The user interface should be easy to understand by any customer. An average person who has no knowledge of computers or neural networks should be able to easily use and interact with the system.

Design Alternatives

Machine Learning Algorithms

The K-Nearest Neighbor algorithm (KNN) takes a datapoint and finds the closest k points of the other data points that represent various classifications. Whatever the algorithm returns is what it predicts as the output classification [6]. KNN is useful because it does not require any kind of training and is relatively simple. However, the KNN algorithm is rerun entirely with each prediction [6].

Support Vector Machines (SVMs) separate data by putting lines between them [7]. For example, positive and negative x-coordinates on a graph can be separated by the line $x = 0$. For small datasets this algorithm runs very well, but performs more slowly with larger amounts of data [7]. Also, SVMs are a kind of shallow architecture, defined as "one large layer of simple template matchers [that] is followed by a single layer of trainable coefficients." [8] For this project, a shallow architecture is not beneficial.

Neural networks are the most capable for this kind of project due to their deep architecture. A deep architecture is defined as "lower-level features or concepts [that] are progressively combined into more abstract and higher-level representations," which allows for more complex tasks, like emotion recognition, to be more and more successful [8]. Additionally, the code we decided to work off of already using a neural network by Jerin Paul, which secured the decision to use a neural network [9].

Implementation Platform

This project is doable on a Raspberry Pi [10]. However, there are limitations on a Pi that could severely limit functionality. For example, neural networks can get rather large and the power needed to run the network comes close to the maximum power of a Raspberry Pi. This means that even if it does fit onto the Raspberry Pi, it will likely use most of its memory and power. This could cause a significant

delay, thus making it so the system is no longer in real time. If we change the size of the network to limit the power needed, the success rate drops since we originally chose the model with the optimal rate. Due to this, we tabled the Raspberry Pi idea.

Implementing this project within a mobile phone, such as on its own app, adds complications to the project as a whole. Examples of these complications include designing and implementing the interface of the app and transferring the code from Colab to the app. The time we would need to address these complications could be spent on improving the system instead, so we chose to prioritize overall functionality.

The last idea of where to run the system off of is a robot. Luckily, Professor Nick Webb of the Computer Science department graciously allowed us to use his robot V.A.L.E.R.I.E., which has a camera at roughly head height to which we can connect. However, in order to be compatible with V.A.L.E.R.I.E., the system needed to communicate using the local memory on a computer. There were three main ways that this could be achieved:

1. Taking and saving the images locally, then setting up communication with Google Drive to process the image, and sending the predicted result back down to the computer.
2. Loading the model locally, so that it can capture an image and process it entirely locally. In this case, the only use of the cloud is for training.
3. Operate entirely locally, including retraining and moving all data to the local machine.

In the end, the second option was most feasible and easiest to implement.

Camera Choice on V.A.L.E.R.I.E.

We want to try to avoid using the camera lower to the ground since the network is not trained on images where the camera is lower than the human's face. Using a camera from this lower angle, when it's trained on images that are level with the face, we hypothesize will significantly affect our results enough to entirely avoid that camera.

Databases

JApanese Female Facial Expression (JAFFE) [11]

JAFFE contains 213 images of 10 different Japanese women who are roughly the same age. Each woman has 3-4 images for the 7 basic emotions, and every image does not have occlusion. The women's faces are centered in the image. Due to its very small level of variation, JAFFE is not suitable for training a system like ours that needs to be able to recognize a wide variety of people. However, it can be used to test the effectiveness of training on a specific subset of race, gender, and level of occlusion.

Cohn-Kanade (CK)

CK is composed of 486 videos of 97 different people. Subjects range in age from 18 to 30, 35% are male, 15% are African-American and 3% are Asian or Latino. Each video is FACS encoded and involves no occlusion [12]. We have yet to move to video capture, so this database is not practical at the moment.

Facial Expression Recognition 2013 (FER 2013)

FER 2013 consists of 32,298 48x48 images of various genders, races, and types of occlusion. The following table tells the number of images for each emotion that exist in the dataset.

Table 1. Counts for each emotion in FER2013

Emotion	**Number of Images in FER 2013**
Anger	4,953
Disgust	547
Fear	5,121
Happiness	8,989
Sadness	6,077
Surprise	4,002
Neutral	6,198

FER 2013 is useful because of the variation of race, gender and occlusion within the subjects in the images. This allows for training to be able to handle a wide range of these features when it sees a new user image and still predict accurately. However, the database does not have an equal amount of images for each emotion, which can cause problems in performance. If the network is not as trained on images of disgust, like what occurs when using FER 2013 to train, it will have a lower success rate when attempting to predict images of people whose emotion is disgust.

Final Design and Implementation

A functional decomposition of our system's design is outlined below in Figure 4.

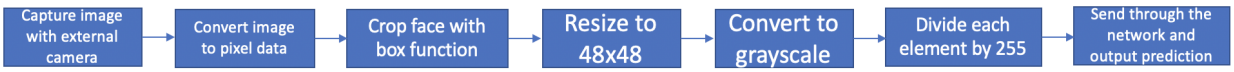

Figure 4. Functional Decomposition of Project Design

First, the system captures an image of the user with a laptop's camera. It takes this raw image data and crops it according to where the face is using the face_locations function from the face_recognition module. It then converts the cropped image to a numpy array and resizes to fit the model's input size of (48, 48, 1). The image is changed to be a grayscale image instead of color, and then run through the trained network. Finally, a predicted emotion is displayed on the screen. These subsystems are detailed in the following subsections.

Image Capturing

The system uses a laptop or computer's built in camera to capture an image, save it locally, and use that as the image to detect an emotion. It is most efficient to take continuous images instead of requiring the user to load in an image and save it in the correct location on the Drive. After the emotion is determined, the code deletes the image using the rm python command.

Image Conversion

The goal is to have an array of images where each image is in grayscale and has dimensions of 48 by 48. These dimensions are desirable because that is what the network is expecting as input. Any other size image will give an error. When an image is taken, its size is 640 by 480 and it needs to be converted to a numpy array and reshaped to fit the system. Thus, the system takes that captured image, crops it to isolate the user's face, then converts it to have dimension 48 by 48 and makes it grayscale. It then compiles all images that are to be predicted and puts them into a list. Thus, the end result of what is inputted is an array where the length of the array is the number of images to test. Each element in that array is a 48x48x1 array with the grayscale pixel values. Instead of the pixel values ranging from 0 to 255, the value is divided by 255 to get a value between 0 and 1.

Neural Network Model

The original network has five layers. The final layer, or the output layer, must have seven nodes for each classification of the basic emotions. The structure of this network can be seen in Figure 5.

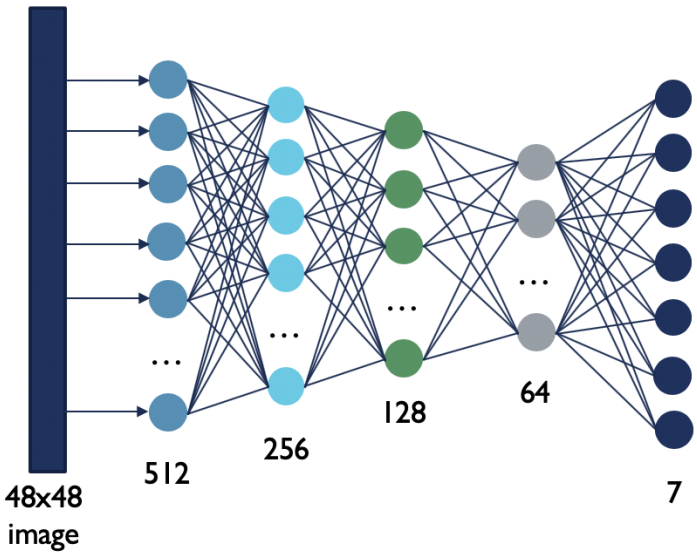

Figure 5. Original Network Structure

Colab

The platform originally used to run the system is Google Colaboratory, further referred to as Colab. Colab is a "free Jupyter notebook that requires no setup and runs entirely on the cloud." [13] Colab makes it very easy to change platforms for this project and allows me to work from anywhere at any time. In addition, if my laptop were to malfunction, I would have no worries regarding if my files were destroyed because they all exist on the cloud. Another major benefit of using Colab is the ability to run on a GPU, which runs much faster than the CPU. Specifications on how much faster can be found in Preliminary Test Results.

Results

Neural Network Layering

The first testing involved optimizing the network structure to give the most successful rate. This involved adding or removing layers from the original model and seeing the effect on performance. A sample of the various layers tested and their corresponding success rates are detailed in the table below.

Table 2. Success rates for different versions of model layers

Layers Included	**Epochs**	**Success Rate**
All	100	63.50%
All	50	58.61%
512, 256, 128, 7	50	64.54%
512, 256, 7	50	65.39%
512, 7	50	65.94%
7	50	63.42%
64, 7	50	63.78%

The "Layers Included" column tells us which of the 5 possible layers of size 512, 256, 128, 64 and 7, are used in that model. The epochs column tells us how many times each image was examined during the training. Finally, the success rate column tells us how successful the system is when testing on images

within the database it was trained on. We chose to proceed with the network using the 512 and 7 layers as it gave the highest success rate. Thus, the rest of the layers that were in the original system after manipulation, which include ones with 256, 128, and 64 output filters, are omitted due to a negative or nonexistent difference in performance success rate.

Capturing an Image, Cropping and Resizing

To test the system's ability to take an image and crop it according to where the detected face is, a user image was taken using the image capturing function and it was displayed how it was cropped. An example is shown below in Figures 8 and 9.

Figure 6. Test of system's ability to capture user photo

Figure 7. Test of system's face-cropping function

As you can see, it is fully capable of detecting where the user's face is and cropping accordingly. Further testing with the user's face in various parts of the window remained successful. Details on how the function detects faces can be found in the face_recognition module documentation [14].

To ensure the image is being properly resized, I simply used the resize function in the Image module, converted it to a numpy array, then printed the size of the array using its "shape" instance variable and verified it against the expected shape.

Importance of Cropping

Experimentation with levels of cropping using the JAFFE database was conducted. The images in JAFFE are not cropped to have only the face in the input image; rather, a sizable amount of the background is in the image that is put through the network. Before cropping, when training with FER 2013 and testing on JAFFE I saw a success rate around 22%, which is extremely low. However, after cropping out the unused background space, the success rate rises to almost 45%. This is still low for what we are hoping to see, but significantly higher than the original 22%. This performance increase emphasizes the importance of doing the same for this system.

Speed

I tested the speed of the system by using the time module in Python to record the time before capturing an image and again once an emotion prediction was outputted. I then found the difference between these times, and found the system to be able to run an execution an average of around every 450 milliseconds. This is slightly higher than what was expected, possibly due to the integration of the photo capturing and image processing actions, but still within the 300-500 millisecond range that was the goal.

Static Testing Results

To test the overall performance of the system, Union students were gathered and images were taken in which they were displaying the seven emotions. Two images for each emotion were taken, leading to 14 images per subject. The environment used for testing was well-lit, and the camera maintained at the same angle and distance away from the subject's face as possible. The testing gave an overall success rate of 57.27%. A breakdown of each emotion's success is detailed in Table 3.

Table 3. Breakdown of Emotion Success

Emotion	Anger	Disgust	Fear	Happiness	Sadness	Surprise	Neutral
Success Rate	63.64%	0%	63.64%	100%	40.91%	54.54%	86.36%

One way to visualize results better is through the use of a confusion matrix, which tells us the amount of images that are predicted both correctly and incorrectly. If an image is predicted incorrectly, the matrix tells us which emotion it predicted instead. A confusion matrix for the above results can be seen in Figure 8.

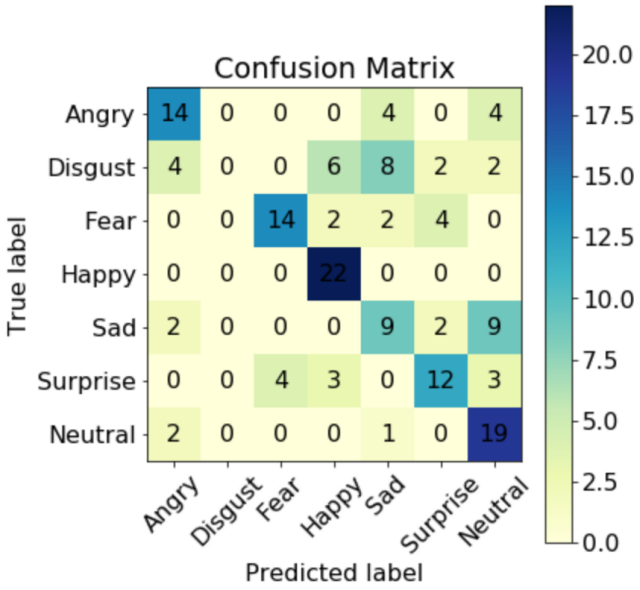

Figure 8. Confusion Matrix for Static Results

Ideally, we would see darker shades of blue and nonzero numbers down the diagonal of the matrix, and zeros elsewhere. This would indicate a 100% success rate. However, since the success rate was 57.27%, there is some variation in the results. For example, it is clear that when images are labeled as sad, they were predicted as neutral 9 times. Similarly, disgust was never predicted correctly, but disgust images were often labeled happy or sad.

V.A.L.E.R.I.E. Testing Results

After static testing was completed, the robot V.A.L.E.R.I.E. was used to test the emotions of subjects walking around Wold center.A student in the Computer Science department was conducting

experiments using V.A.L.E.R.I.E. where she was having the robot approach students walking around the basement of Wold and asking them to help her use the elevator. The system was implemented on the robot at the same time the student was running her experiments. To record their emotions, an external camera was connected to my laptop and the system was run continuously. Only images that detected a face were run through the network, so only 106 images were ever tested. The results from those 106 images are outlined below in Table 4.

Table 4. Breakdown of V.A.L.E.R.I.E. Results

Emotion	Anger	Disgust	Fear	Happiness	Sadness	Surprise	Neutral
Count	11	0	15	11	10	22	37

The emotion the system detected most was neutral, which makes sense since most people walking through Wold will have a neutral expression on their face. The next two highest detected were surprise and then fear. I thought these results made sense, since anyone would guess a human would be surprised or fearful if a robot approaches them randomly.

CPU vs. GPU

When the code is trained on just a CPU, which I accidentally did one time, it predicted to take upwards of 12 hours to complete. However, when using a GPU, that time dropped to about an hour.

Conclusions and Recommendations

Conclusions

There are three main conclusions that I drew over the course of this project.

1. The system is capable of successfully predicting facial emotions, but is not as successful as I was hoping for it to be. From the static testing I found it recognizes emotions accurately 57.27% of the time. This is much lower than my goal rate of 70%, and a bit lower than the overall network accuracy from testing on its own images of 65.94%. Overall, I was pleased with the accuracy that I got and, if time had allowed, believe there could have been tweaks along many steps in the

process that would increase the accuracy, like changing the training slightly and having more consistent and formal testing. This would have included collecting more images from a greater number Union students and being able to use V.A.L.E.R.I.E. in my own experiment instead of having to adapt how another student was using the robot.

2. A neural network is only as good as the training data you feed into it. In FER 2013, the number of images of each emotion is not consistent, leading to a highly differing success rate across the emotions. This can have a drastic effect on the performance of the system and might have been a reason for the lower success in this system.
3. Emotion is very subjective. Throughout the course of this project, I have had to always keep in the back of my mind the fact that not everyone sees emotions in the same way. The way that FER 2013 is classified is by a human going through the images and determining the emotion that they feel best fits that image. For any given image, two people could pick different emotions for what they believe the subject in the image is displaying. This leads to performance discrepancies. Similarly, when telling subjects to display a certain emotion during testing, two subjects might display the same emotion in very different ways. This, too, leads to performance discrepancies.

Recommendations

There are several recommendations I have for those who recreate or work further on this project.

- Be comfortable with neural networks. It might seem obvious, but understanding the internals of the project will help significantly in the long run.
- Also be comfortable with the code. Go through line by line and make sure it is understood why that line is there.
- Test everything. There are many small details that can get switched or confused and cause an unnecessary amount of time to correct. For example, ensure that the emotion encodings are consistent. If your dataset has surprise listed as encoding 5 but you are outputting based on 6 being surprise, you will get a lot of failure. Small things like this can and will happen.

- Check the images that you are running through the system. Are they what you expect them to look like? Compare them to those in the dataset. If the photos in the dataset are vastly different from those that are being tested, performance will drop.
- Expect debugging to take much longer than anticipated. Especially when switching between platforms, there will likely be a seemingly endless stream of errors. Fix them one by one and take a step back if you get too stuck. Consult sources on the internet, but be careful of overcomplicating the error through internet recommendations.
- Work from one location when first getting the system to work. Many errors will come from certain modules or libraries not being installed. Switching where you work from will cause this to continue coming up. I recommend operating entirely off of an external hard drive to keep everything in one location.

References

[1] Ligang Zhang, Brijesh Verma, Dian Tjondronegoro and Vinod Chandran, “Facial Expression Analysis under Partial Occlusion: A Survey.” *ACM Computing Surveys*, Volume 51, Issue 2, June 2018.

[2] The Emotion Machine. *Classification of Emotions*, http://www.theemotionmachine.com/classification-of-emotions/. 2011.

[3] Skymind. *A Beginner’s Guide to Neural Networks and Deep Learning*, https://skymind.ai/wiki/neural-network. 2019.

[4] Dertat, Arden. “Applied Deep Learning - Part 1: Artificial Neural Networks.” *Medium*, Towards Data Science, 9 Oct. 2017, towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-networks-d7834f67a4f6.

[5] Warren S. Sarle, “Neural Networks and Statistical Models,” in *Proceedings of the Nineteenth Annual SAS Users Group International Conference*, 1994, pp. 1

[6] Cover TM, Hart PE, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, 1967, pp. 21–27.

[7] Savan Patel, *Chapter 2: SVM (Support Vector Machine) Theory*, A Medium Corporation. https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72. 3 May 2017.

[8] Yoshua Benico and Yann LeCun, “Scaling Learning Algorithms towards AI,” in *Large-Scale Kernel Machines”, L. Bottou, O. Chapelle, D. DeCoste, J. Weston (eds) MIT Press*, 2007, pp. 1-3.

[9] Paul, Jerin. “How I Developed a C.N.N. That Recognizes Emotions and Broke into the Kaggle Top 10.” *A Medium Corporation*, FreeCodeCamp.org, 12 May 2019.

[10] Adrian Rosebrock, *Raspberry Pi Face Recognition*. https://www.pyimagesearch.com/2018/06/25/raspberry-pi-face-recognition/. 25 June 2018.

[11] Lyons, M., et al.. *Japanese Female Facial Expression (JAFFE) Database*. 2, figshare, 26 July 2017, https://doi.org/10.6084/m9.figshare.5245003.v2.

[12] Jeffrey Cohn, *Cohn-Kanade AU-Coded Facial Expression Database*. Carnegie Mellon University: The Robotics Institute. 1999.

[13] Google Colaboratory - Welcome to Colaboratory. https://colab.research.google.com/notebooks/welcome.ipynb#. 2019.

[14] Geitgey, Adam. “face_recognition Package.” *face_recognition Package - Face Recognition 1.2.3 Documentation*, 6 Mar. 2017, face-recognition.readthedocs.io/en/latest/face_recognition.html.