Facial Recognition Using Deep Learning

Abigail Carbaugh

Elizabethtown College

carbaugha@etown.edu

**ABSTRACT**

One difficulty for people within the autism spectrum disorder is recognizing and understanding facial expressions that others are portraying to them. Facial expressions through conversations are how someone gets their mood across to another, and without this, it is very difficult for someone to understand how another is feeling. Interpreting and understanding facial expression is a daily skill for interacting with people around you, and needed in personal, professional, and academic settings. This project is based on a dataset from a Kaggle challenge, ‘Challenges in facial recognition’ which includes 64x64 greyscale images of faces with different expressions. The goal for this project is to achieve an accuracy core of 80% or higher, ultimately beating the accuracy score of the winner of this competition. In order to achieve this high accuracy score, this project and the models used will be a mix of neural networks with different architectures. Once the best model has been discovered, it will be used in an application that will be able to predict a facial expression on custom photos.

# INTRODUCTION

Understanding facial expressions is a vital human survival tool in today’s world, which is dominated by social interactions in all aspects of life: professional, personal, academic, etc. Facial expressions display one’s personal emotions and can help indicate one’s intentions in a social situation. Studies have previously shown that “…individuals rarely interact with context-less faces [1],” so the act of understanding these expressions is important in these face-to-face interactions. Neurodivergent individuals, however, have a more difficulty recognizing these expressions at higher intensities, making them unable to distinguish what a person is trying to convey. According to the CDC, approximately one in thirty-six children are diagnosed with autism [2]. Because of this statistic I wanted to create a machine learning model for the final project that would help others and be easy to use. This model, along with other technological advances, could possibly be able to be integrated into things like smart glasses therefore helping an individual detect these emotions that cannot do so on their own.

Previous attempts to tackle this problem have occurred in the past, but they only use photos from the dataset being used, and not applying the application on other photos, nor could they be done in real-time scenarios. Previous works have also only used three basic emotions (happy, surprise and neutral) that the model has to recognize. This prevents the model from being used universally. If the model can only classify three expressions, there are limited applications to the project. These works, however, have no application made to show off the model for use.

To overcome these previous works’ shortcomings, I plan to, along with the photos from the dataset, use personal photos of me and others in order to see if this model can be universally used across photos, not just from the dataset. I also plan on incorporating more than three emotions like previous works did. I plan on using seven basic expressions (anger contempt, disgust, fear, happiness, sadness and surprise) in my model in order to classify the photos instead of limited expressions in order to make it more applicable and helpful in different social situations.

In creating this project, neural networks will be used in order to produce a model with a high accuracy rate in recognizing facial expressions. The seven basic human emotions, stated before, will be utilized in classification in order to make the ending application more applicable to real world scenarios. Once a model is selected as performing the best, an application will be made for individuals to use for personal images.

The dataset for this project originates from a Kaggle facial recognition challenge where contestants tried to create a model with high accuracy that could predict a facial expression an image is trying to convey. The dataset consists of 48x48 pixel greyscale images of different faces. This dataset, unlike others previously established, does have the seven categories of facial expressions I would like to use in my model, eliminating the need to create my own new categories. The data is pre-split into training and testing datasets, with training containing the columns pixels and emotion and testing only containing pixels. Each dataset was used for a different testing phase in order to test the model’s accuracy. Because the training dataset was the one those in the competition used, that will be the one utilized and split into further training and testing. The outcome of this challenge was to accurately predict the emotion column for the testing data, the group winning the competition receiving a 71.2 accuracy rate.

In order to create the neural network models needed, the Keras package will be utilized. This package is a high-level API for building neural networks within a Python notebook [3]. It also allows the possibility to create custom layers and models so that one can set up their own transformation and weights to a layer of the model without many restrictions. To evaluate the performance of each model, the created training dataset will be evaluated to see if their confusion matrices show a promising model. A classification report will also be produced for each experiment to find the accuracy score. It is expected that the model chosen for the application development will have at least a 60% accuracy rate, as that is what the top ten contestants from the original Kaggle challenge received that percentage or higher. The long-term goal for this project is to receive a score of over 71.2% score, which is what the winner of the challenge received.

Going beyond the challenge itself, an application was made to make the model itself useful in real world scenarios. This application allowed individuals to upload an image to the application and have it predict the facial expression of the individual. Further advancing the project beyond the initial publishing will hopefully bring more accessible applications to those with cognitive or even mobility issues.

# BACKGROUND

Autism spectrum disorder is defined as a developmental disability caused by differences in the brain [4]. One possible effect of this disability is having problems with social communication and interaction, including understanding and determining facial expressions, one basic human necessity.

There are seven expressions that psychologists have recognized to be universal expressions. These expressions are anger, contempt, disgust, fear, happiness, sadness and surprise. Each of these emotions, signaled by facial expressions, “…turn on different cognitive and physiological reactions in order to prime the body for action [5].” For neurotypical people, one without atypical neurological patterns, recognizing these facial expressions is done by the brain within 200 milliseconds [6]. For neurodivergent people, however, may not be able to recognize an expression that quickly. Within the range of emotions, there are also various different intensities of each emotion. With these different intensities of facial expressions, people with autism spectrum disorder (ASD) may not be able to understand the emotion one is portraying if it is not high intensity. There have been many studies done on people with ASD regarding facial emotion recognition which have found that those with ASD will perform normally by recognizing these expressions with high intensity. With low intensity, however, they were not as easily recognizable [7].

Psychologists have done studies on people with ASD to try to help individuals recognize emotions, but many have fallen short as autism is a spectrum, that presents differently in each individual as well as their ability to identify facial expressions accurately and in a timely manner. Everybody diagnosed with ASD is different as some may exhibit many symptoms while some have very few, so teaching others to interpret and understand expressions may be more challenging or not. Zhenjie Song, author of *Facial Expression Emotion and Recognition Model Integrating Philosophy and Machine Learning Theory* [8], suggests that machine learning models, specifically convoluted neural networks (CNNs) may be the answer to helping people with ASD recognize facial features in social settings better. She found that CNNs can efficiently and accurately recognize facial expressions. Song had determined that finding datasets for this problem was very challenging as many of the accessible datasets were very small. She proposed using image augmentation in order to extend the dataset used. Song had been able to achieve an accuracy of 74%. Song did not say how she classified the images, however, so we do not know how many categories (emotions) the images could be classified as. Song also went into emotion recognition through gestures and speech, but since facial expressions are one of the most important things with face-to-face interactions, I chose to only focus on that key aspect. Song also was not able to achieve the accuracy of 80% or higher, like this project intends to do.

Xiaofeng Lu, author of *Deep Learning Based Emotion Recognition and Visualization of Figural Representation* [9], also tried to solve this problem using CNNs. Lu had been able to achieve a much higher accuracy score or 98.75%, much higher than Song. Lu proposed a convolutional neural network-Bi-directional Long Short-Term Memory algorithm (CNN-BiLSTM) in order to achieve this high accuracy score. Lu found this algorithm not only provides impressive results but provided significant “…experimental reference for research on learners; emotion recognition and graphic visualization of expressions in an intelligent learning environment.” Lu used the Keras package, as this project plans to use, in order to create the architecture for this project. Instead of using an established dataset, Lu used image and audio data of learners in a class as they answered questions. Upon evaluating the model, Lu found that the function loss of the CNN-BiLSTM model decreases to 1.33% after 100 training periods, which reduces the influence of function loss on the model. By using this real-time emotion recognition, Lu was able to create a model having an accuracy score of no less than 90%, and plans on enhancing the model further, as of this paper’s release. Lu had been able to achieve much higher results with their model, but the model also focused on using real-time data and images.

By using these two papers as guides, new CNN models will be made and used with my own hyperparameters in hopes of achieving a higher accuracy score than the winner of the original competition. Multiple Python packages will also be installed in order to achieve the wanted results.

# DESIGN

Figure 3: Image after the pixel’s column was stacked and reshaped.

As previously stated, this project will be using the Keras package through TensorFlow with Python. This project is done mainly on a Google Colab notebook in order to utilize the GPU function for runtime, making it faster and more efficient to run and use when testing and training upon one hundred epochs for each model.

Text, letter

Description automatically generatedWhen loading the dataset, it was decided that it would be downloaded into the computer’s working directory, so a large file was not downloaded to the local device used. In this step of data preprocessing, it was determined that there were three datasets: training, testing, and one labeled ICML. The ICML dataset was used for the final testing for determining the winner of the original competition. As previously mentioned, the training datset will be used within the models created. From here, the get\_file function from keras.util was used which allows a loaded zipped file from a URL and extracts the contents within the notebook eliminating the need to download large files. The one downside, however, is that when working in different sessions, the link will no longer work, and a new link will need to be generated. Once the dataset is loaded in, the listdir() function from os can be utilized to study and use the extracted chosen dataset.

The first thing done to ensure that the data was as uniform as possible. First, it was ensured that there we no labels missing, eliminating the need to delete NAN rows. The next alteration was taking the training dataset at the “pixels” columns (which is our X\_train) and reshape the array to 48x48 and change the pixels column type to float32 because the original type was object. Once this was completed, the array was stacked. This allowed me to have a sequence of arrays along a new axis, allowing the images to be created. A count plot was created in order to find the number of counts per expression in the training dataset.

Figure 1: Train dataset before pixels were stacked and reshaped.

It is shown from the count plot that there are less images labeled disgust than any other one. Because of the possibility of overfitting the model, it was considered being dropped but as to keep the project aligned with the seven basic human emotions, image augmentation will need to be conducted, much like seen in Song’s experiments.

Graphical user interface

Description automatically generatedThe first 10 images of the training dataset were first plotted using the X\_train data, showing the number in the array the image is, and what emotion is being depicted according to the y\_train data. The data taken in for the image has been reshaped to 48x48 with the colormap set to gray to maintain continuity and uniformity throughout the photos.

Figure 3: Image before augmentation with the image number shown above.

Diagram

Description automatically generatedTable

Description automatically generatedTo balance the dataset, images with the label of ‘disgust’ were taken and placed into a separate dataset. From here, the dataset was copied until it reached around the same number of images as the others in the training dataset. The disgust dataset was then added back to the original training dataset.

Figure 5: Basic architecture of a CNN model [10].

A picture containing graphical user interface

Description automatically generatedThe new dataset was then taken through image augmentation using the ImageDataGenerator. The ImageDataGenerator takes the original photo inputs and transforms them randomly according to parameters specified.

Figure 5: Image after augmentation using the Keras imagedatagenerator.

Chart, bar chart

Description automatically generatedAfter the images were augmented, the dataset was split into training and testing, and the y\_train was modified to be categorical using the tensorflow.keras.utils package.

Once all data preprocessing was completed, the next step in the process was to make the first model. For the model that will be considered the base model, a combination of convolutional 2D and Max Pooling 2D layers will be used for the first four layers. The model will then have a dropout layer added, and then the model will be flattened. A dense layer is then planned to be added with the activation function being ‘relu.’ Another dropout layer will be added, and then a final dense layer with activation function ‘softmax’ will be implemented. Other models will also be implemented in order to see which model architecture will perform the best and achieve a high accuracy score. Along with the base model, a Visual Geometry Group with 16 layers depth (VGG16), a Residual Network (ResNet), and a Vision Image Transformer (ViT) models will be used.

Figure 2: Count plot of the number of each emotion in the training dataset.

For each model, the optimizer chosen was Adam and when compiling the models, categorical cross-entropy will be used. If time permits, dimensionality reduction will be further implemented in the design to see if it will deliver a higher accuracy score.

When designing the application itself, a third-party modeling site will be used in order to streamline the development process as this was my first attempt at building something of this scale.

Figure 4: Value\_counts of the emotion column once the dataframe was extended.

# EXPERIMENTS

A picture containing text, screenshot, font, number

Description automatically generatedThe first step for the experiments was to create a sequential CNN model that generates certain model information. The base model will contain a mixture of convolutional 2D and max pooling 2D layers will be used for the first layers. Then, a mix of dropout, flattening and dense layers will be added with the proper activation functions with the optimizer chosen being Adam with categorical cross-entropy as the loss function. Once initial modeling was conducted, it was determined that the base model only produced an accuracy of 21%. Having this low of accuracy was expected as there was no further feature extraction or fine tuning performed.

A picture containing text, screenshot, font, number

Description automatically generated

Figure 8: Classification report of the ResNET model.

Figure 6: Classification report of the base model.

A screenshot of a person's face

Description automatically generated with medium confidenceThe Visual Geometry Group model was the second model experiment conducted. A VGG16 model is a simple convolutional neural network architecture. This model was based on an analysis of how to increase the depths of the networks. A VGG model consists of blocks, and each block is comprised of 2D Convolutional and Max Pooling layers [9]. By using Keras, a VGG16 model can then be imported. By importing the model, all that was needed to do next is add layers that will allow the model to take in the dataset used for this project. The layers added was a flatten layer, three dense and two dropout layers. As done with the base model, the final dense layer had an output shape of (None, 7) as there were seven classification categories. Also, when conducting the VGG model experiment, the input shape was changed to (48, 48, 3) instead of the typical (48, 48, 1) as there are 3 RGB channels within this model.

A picture containing text, screenshot, font, number

Description automatically generatedWhen conducting the VGG16 model experiment, it was found that it could produce an 85% accuracy rate. This was exciting as it was much higher than the winner of the Kaggle competition’s accuracy score, meaning that the initial goal of reaching at least 80% accuracy rate was met. While the goal was met, more experiments were conducted in order to see if other models and their architectures could perform better.

Figure 9: Image before and after the ViT model converts it into patches.

Figure 7: Classification report of the VGG model.

The ResNet model was the next model that the data was fed through. This model was also imported through Keras. The ResNet architecture uses shortcut connections to solve the vanishing gradient problem [10]. Like with the VGG16 model, once imported, more layers were added to allow the dataset to work with the model. The layers added were the same added in the VGG model. The ResNet model was found to have a 55% accuracy rate, not the best but still conducted better than the base model.

The last model chosen was the Vision image Transformer model as it was the most intricate. The ViT model applies the transformer model’s architecture with attention to sequences of image patches and does not use convolutional layers. This model was also imported from the Keras package but works differently from the VGG16 or ResNet models. A ViT model uses a multilayer perceptron that has both a dense and dropout layer. From here, a patch maker converts images into patches and an encoder will perform linear transformation on image patches. Lastly, the program defines a classifier with the vision transformer with the encoded patches acting as the input. No additional layers were needed to be added to this model.

A picture containing text, screenshot, font, number

Description automatically generatedWith this model, however, there are custom layers that do the image augmentation and patch building/encoding within the model instead of doing everything separately. These custom patches prove further complications for the application, however. Once this model was finished running, it was found to have an 89% accuracy rate, the best out of all the previously established models.

Figure 10: Classification report of the ViT model.

Chart, bar chart

Description automatically generatedWhen comparing the four models, it is clear that this accuracy score can possibly be even higher with further testing, making the model even more accurate.

Figure 3: Layout of the application made with Streamlit.

Figure 11: Each model and its corresponding accuracy rate.

To test the validity of the model within the application, it is planned that a dataset of the images used in training will be made to run through the model’s application. This is because due to image sizes and other real-world inconsistencies such as resolution not being the same. By using the training images, we will already have the label as well as the image in the same resolution, making less room for errors, further enhancing my model’s accuracy and reproducibility.

The last goal of the project was to create an application that can accurately predict the emotion expressed on one’s face through an image uploaded. This was the largest and was thought to be the most time-consuming aspect of the project, so the most time was allocated to this aspect.

The application proposed was ultimately made using Streamlit. This is a free and open-source platform for people to demonstrate their machine learning models. By uploading the models saved into the application notebook. We can easily call the model for the application to predict the image uploaded.

With using Streamlit in Google Colab the first thing that had to be done was install Streamlit to utilize its functions. By using a mix of st.write and st.upload, a simple application could be made. In order to have the image be predicted, a button was added that, once pushed, will predict the image’s expression. To have the button work, an if statement was placed around it, making it so if the predict button is pressed, the image is run through the model.

Graphical user interface, text

Description automatically generatedBy using ‘%writefile app.py’ the code was saved into the application. From here, a localtunnel had to be installed in order for Streamlit to produce a link that can be used for testing the application. Lastly, by using a run statement, Streamlit was able to give a URL to the application. Because there was no need to further develop the application, this was the final step needed to run and test the application for the final presentation.

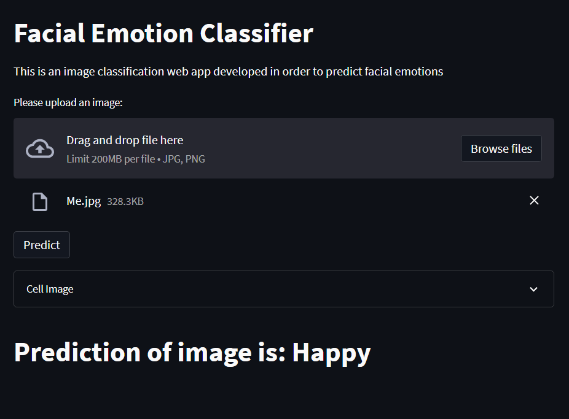
For the final demonstration, the abstract as well as the basic description of the dataset was used in order to allow readers to see the motivation as well as the background of the project before using.

Figure 12: Code ran for Streamlit to generate a URL for the application.

While the ViT model did produce the highest accuracy rate, the VGG model was ultimately used in the application. This is due to the custom layers that make up the ViT’s architecture. When uploading a photo using the ViT model, an error would throw saying that the layers could not be recognized. While attempts were made to fix this issue, there were no solutions found, making the model used in the application the VGG model for the final demonstration.

# SHORTCOMINGS

While the project was able to achieve an accuracy of 80% or higher, there were still some aspects to the project that need further improvement in order to make it more user-friendly as well as to make it more applicable in many different social circumstances and problems.

The VGG model had an 85% accuracy, and it would prove to be the most ideal to have voer the better performing model, the ViT, as the model used within the application utilized custom layers. Within Google Colab, it is very difficult to reference other notebooks when coding as all memory is lost at the end of each session. In order to solve this, the notebook would have to be hosted on a local server, which was not accessible at the time of creating the application. Streamlit was also found to have difficulties with a model having custom layers, so it seemed that the VGG was going to be the end result either way due to the limitations.

When testing the application come presentation day, it was found that the images being taken and uploaded were not being predicted correctly. This could be concluded to be caused by either having someone else be included in the image, or because the image resolution was not the same as in the original images used. By having the same image size and resolution of all the images in the training dataset, there tends to leave room for errors using personal images within the application. In order to fix this for future use of the base application, it is proposed to add personal and other images with varying resolutions to hopefully fix this issue and make the model more accurate. Including various resolutions, number of people in the photos, and different photo qualities in the dataset base, that should also help increase the accuracy of my model in real-time. It is also suggested to add the top five predictions of the expression and their accuracy scores in order to allow the user to see the confidence the model has of the facial expression.

Upon speaking to individuals at the final presentation, it was determined that there could possibly be some issues with having to either take or upload a picture. This is not very user-friendly. It was then suggested by users that the application take on a “real time” aspect where the camera would be constantly on and searching for faces to predict expressions. This could be further tested upon this project.

Personally, I also wanted to publish the application once it reaches its final stage. By publishing it, it will allow users to not only use the application, but hopefully, with it being published on Github, users will be able to use the notebook and possibly tweak the application to further cater to their needs. This also allows other people to potentially use my model as a basis to make an ever more accurate model in predicting facial expressions in real-time, better helping people within the Autism Spectrum Disorder community to determine facial expressions of their peers in social situations. The possibilities for this project keeps growing, making it more exciting to “officially” publish the application, allowing more people to access it, learn from it, and potentially enhance it further to get an even higher accuracy rate in facial expression recognitions. In the end, this project was about me, but me being able to provide something useful to a community in need.

# TIMELINE

With the limited timeline to finish the project, all work has been planned out accordingly in order to ensure the work will be finished in time for the final presentation on April 17, 2023.

1/23/23 – Start proposal

Figure 10: Basic layout of the Streamlit application.

1/30/32 – Data collection and preprocessing

2/6/23 – Exploratory Data Analysis

2/13/23 – Model Design

2/20/23 – Feature Extraction

2/27/23 – Model Implementation

3/13/23 – Model Implementation and Training

3/20/23 – Parameter Tuning

3/27/23 – Application Design and Use Cases

4/3/23 – Model Evaluation and Comparison

4/10/23 – Code Final Revision

4/17/23 – Final Paper Writeup

# REFERENCES

1. Qiang Xu, Yaping Yang, Qun Tan, and Lin Zhang. 2017. Facial expressions in context: Electrophysiological correlates of the emotional congruency of facial expressions and background scenes. (December 2017). Retrieved February 6, 2023 from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5733078/
2. Anon. 2022. Signs and symptoms of autism spectrum disorders. (March 2022). Retrieved May 2, 2023 from https://www.cdc.gov/ncbddd/autism/signs.html
3. Keras Team. Simple. flexible. powerful. Retrieved February 6, 2023 from https://keras.io/
4. Anon. 2022. Signs and symptoms of autism spectrum disorders. (March 2022). Retrieved May 2, 2023 from https://www.cdc.gov/ncbddd/autism/signs.html
5. David Matsumoto. Benefits of Reading Facial Expressions of Emotion. Retrieved February 6, 2023 from https://www.humintell.com/2021/04/benefits-of-reading-facial-expressions-of-emotion/
6. David Matsumoto. Benefits of Reading Facial Expressions of Emotion. Retrieved February 6, 2023 from https://www.humintell.com/2021/04/benefits-of-reading-facial-expressions-of-emotion/
7. Neuroscience News. 2021. Why people with autism read facial expressions differently. (August 2021). Retrieved February 6, 2023 from https://neurosciencenews.com/aans-asd-facial-expression-19065/
8. Zhenjie Song. 2021. Facial expression emotion recognition model integrating philosophy and machine learning theory. (September 2021). Retrieved February 6, 2023 from https://www.frontiersin.org/articles/10.3389/fpsyg.2021.759485/full
9. Xiaofeng Lu. 2021. Deep Learning based emotion recognition and visualization of Figural Representation. (December 2021). Retrieved February 6, 2023 from https://www.frontiersin.org/articles/10.3389/fpsyg.2021.818833/full
10. M.K. Gurucharan. 2022. Basic CNN architecture: Explaining 5 layers of Convolutional Neural Network. (October 2022). Retrieved February 6, 2023 from https://www.upgrad.com/blog/basic-cnn-architecture/
11. Anwar, A. (2022) Difference between alexnet, vggnet, ResNet and inception, Medium. Towards Data Science. Available at: https://towardsdatascience.com/the-w3h-of-alexnet-vggnet-resnet-and-inception-7baaaecccc96 (Accessed: April 24, 2023).
12. [Boesch, G. (2023) VGG very deep convolutional networks (vggnet) - what you need to know, viso.ai. Available at: https://viso.ai/deep-learning/vgg-very-deep-convolutional-networks/#:~:text=VGG%20stands%20for%20Visual%20Geometry%20Group%20and%20consists%20of%20blocks,more%20complex%20functions%20also%20increases. (Accessed: April 24, 2023).