Using Convolutional Neural Networks to Classify Facial Expressions

Group 5

Jenna Cogswell - B00829098

Brijesh Hota - B00844617

2022-12-04

CSCI 4155 Machine Learning

Dr. Sukhchandan

Abstract

Being able to see the expression on someone's face and have an idea of how they might be feeling is an ability people often take for granted. There are various reasons for someone not having the ability to read facial expressions, such as impaired vision. Through a supervised learning approach, a machine can read in images and learn to classify them according to emotion. For this project a dataset of 70% training images, 15% validation image, and 15% testing images, each with 7 categories of emotions, was used. A convolutional neural network was trained and validated then used to predict the emotions in the testing set with an accuracy of up to 62.09% for the third model.

Introduction

Machine learning technology has been rapidly growing and improving. Its impact can be visualized in many different fields of society. Deep learning has been largely reported to be capable of achieving automated screening and diagnosis of common vision-threatening diseases, such as diabetic retinopathy (DR), glaucoma, age-related macular degeneration (AMD), and retinopathy of prematurity (ROP). Today machine learning is widely used as an accessibility assistant to help people with disabilities.

Almost 3% of Canadians above the age of 15 are reported to have vision disability that limits their daily activities (NCBI - WWW Error Blocked Diagnostic, n.d.). These vision disabilities forbid them from properly learning, understanding and doing any tasks independently. Machine Learning models have been extensively used by different pharmaceutical research companies to create different products that can help individuals with partial and complete blindness to ease their tasks.

Blindness also restricts an individual from understanding the facial expressions of people in his/her surroundings independently. Machine learning models can be used in these scenarios to identify the expressions of people in visually impaired person’s surroundings. Hence, this project deals with emotion recognition of an individual through facial image processing.

Literature Survey

Facial Emotional expressions elicit rapid responses, which often imitate the emotion in the observed face. These effects can even occur for faces presented in such a way that the observer is not aware of them [10]. Humans usually employ different cues to express their emotions, such as facial expressions, hand gestures and voice. Facial expressions represent up to 55% of human communication while other ways such as oral language are allocated a mere 7% of emotional

expression [11]. This calls for a need for Facial Expression Recognition (FER) systems that can identify the basic human emotions.

The majority of FER systems attempt to recognize six basic emotional expressions including fear, disgust, anger, surprise, happiness, and sadness introduced by Ekman [12]. The dataset that we've selected in the project is based on the kaggle's FER-2013 dataset that contains the 7 basic human emotions (which includes Neutral emotion) as it fits the description stated by Ekman. Our model is built with the purpose of classifying emotions found in images into different categories like sad, happy, neutral, fear, anger, disgust and surprise.

Artificial neural networks have shown a performance breakthrough in the area of object detection and scene classification, especially convolutional neural networks commonly known as CNN [14]. For emotion recognition we use the CNN algorithm, as the CNN architecture is frequently utilized in a variety of domains for learning features and classification [13]. The Convolutional Neural Network (CNN or ConvNet) is a subtype of Neural Networks that is mainly used for applications in image and speech recognition. Its built-in convolutional layer reduces the high dimensionality of images without losing its information.

CNN is made up of numerous layers, each of which performs a different transformation function. Convolution is a mathematical procedure that combines 2 functions to generate a third function that expresses the function's modified shape. convolution, pooling layers, and fully connected networks are part of CNN. According to Goud and Hussain's 'Estimation of Emotion using CNN' research paper an average training accuracy of 96% and test accuracy of 59.19% can be achieved when a dataset of around 30000 grayscale images and 200 epochs is used for training [15]. Upon research and observation, we found that most of the models that utilized the FER-2013 dataset from kaggle (which is used in this project) resulted with a train accuracy of around 70% and test accuracy of around 60%.

Problem Statement

For people who may not have any disabilities, there are a lot of simple day to day interactions that may be taken for granted. When someone is making a joke or using sarcasm their tone of voice could be very neutral, making it difficult to be aware of the true meaning behind their words unless you can see their expression. For example, if someone were to say "wow that sounds fun" this could have two very different meanings, saying it with a neutral or sad face indicates the expression was sarcastic, versus saying it with a smile and raised eyebrows can indicate that the person is excited, so the expression matches their emotion.

The problem that this project will address is the inability of visually impaired persons to read the facial expressions of others. For many people, picking up on visual and other non-verbal cues while interacting with someone might seem like second nature, for those who struggle with reading visual cues, this requires much more effort and confusion. There may be various reasons why someone cannot visually detect emotions, the person could be visually impaired, they may be autistic and struggle with connecting expressions to certain emotions, etc. Without an accessibility tool to aid in recognizing emotions, confusion and frustrations from all people involved in the conversation or interaction, may arise. The person demonstrating facial cues through emotions may get continually frustrated that their cues are not being picked up on. Having an accessibility tool available that can recognize the emotions through facial image processing and communicate them to the user would be greatly beneficial.

Unfortunately, emotions and how each person displays them are both very subjective. Everybody experiences emotion differently and therefore has different ways of showing it as well. As X. Xu, Z. Ruan and L. Yang express in their paper on facial expressions of emotion, even though there are similarities in the ways we express emotion, these similarities are not always as reliable as we may think [9]. This unpredictability is something that was kept in mind throughout this project. It is to be expected that the test accuracy for any model classifying facial expressions may not get very high, since even other people cannot be 100% sure of the emotion someone is feeling.

The models used in this project could be implemented as a software tool that can take in images and relay the emotion it predicts back to the user. It should be noted that this tool should be used more as an aid, as opposed to a solution.

Proposed Technique

Choosing a Dataset

Our solution for the proposed problem is to create an accessibility tool in the form of an image processing model to recognize certain emotions. A supervised learning approach will be used in the form of classified images. We will use these images to train and validate a model to properly classify test images and use the true labels to test the accuracy of our model.

In the beginning stages of the project once the premise was chosen, the next step was to decide on a dataset to use. Finding a dataset with images that are specifically classified by emotion was proven to be difficult. Taking the images ourselves, as opposed to finding a dataset online, was an option that we concluded not to go with because of various issues such as the number of images needed, finding enough people to consent to having their picture taken, and the time

constraint of the project. Knowing this, we were able to find a dataset of emotion classified images on the Website Kaggle. This is a dataset from the Fer-2013 project [7], a classification model with a similar aim to this one.

In the dataset chosen, there are 23 298, 48x48 greyscale images, the faces are already centered and arranged into folders based on the 7 emotions used. Examples of the images in each category are shown below in Figure 1.

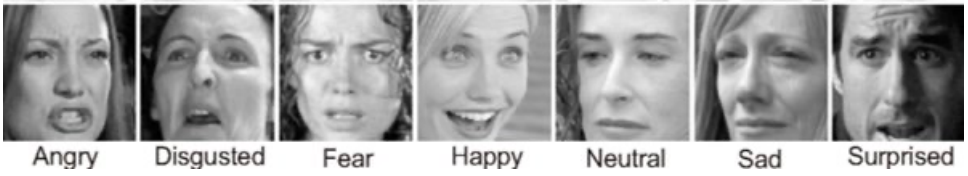

Fig. 1. Example images in each emotion class [7].

The original dataset also has the images arranged into a training and a test set, for this project a validation set was going to be used as well, so when saving the data, the training and testing images were unseparated.

Data Preprocessing

The first step in the data preprocessing stage was to manually separate the images into training, validation, and testing sets, with 70/15/15 split. The code that creates these folders will only run once the dataset is uploaded, any rerunning of the code will skip this part since the folders are already made and saved.

Once the images were sorted into proper train/validation/test sets, we were able to analyze the data more to do some further preprocessing. The number of images in each emotion directory for the training set was analyzed. In a proper training set each class should have close to an equal number of images to minimize any bias towards classifying images of a certain class, as further expressed by E. P. Watkins in her article on balancing classes for image data [2]. It was found that the 'happy' class had a much greater number of images compared to the other classes, and the disgust class had a much lower number of images. To fix this an approach of combining under sampling and oversampling was used. For the 'happy' folder under sampling was used, random images were deleted to get an amount closer to the other classes (4399 images). For the 'disgust' folder oversampling was used through a form of data augmentation using the image generator function to shift the images and save copies of them. This was done about 10 times to increase the size of the 'disgust' folder to 3999 images.

Directory iterators are then made to go through the folders and generate each image for the model input, along with the emotion directory it was in saved as an array where each value in the array corresponds to the emotions, 0=anger, 1=disgust, 2=fear, 3=happy, 4=sad, 5=surprised,

6=neutral. We then had the idea that turning the images into pixel arrays would be easier to work with and train our model with. After trying it with the images made into arrays, we concluded it did not have any positive changes on our models and was more difficult to work with than the directory iterators, so we continued working directly with the images in the directories through the iterators.

Now that the dataset is properly split and sized, an example image from the training set was then printed along with its class label to give an idea as to what the images look like as seen in figure 2.

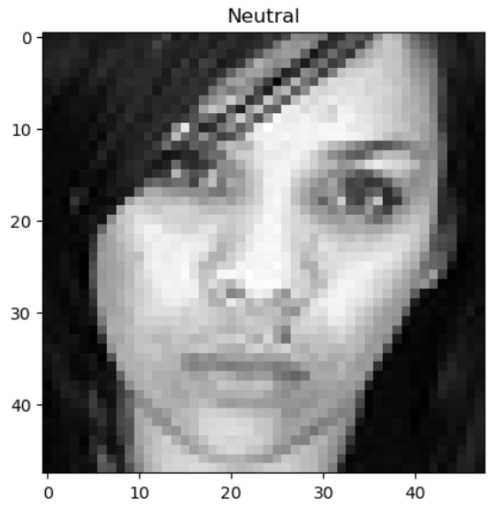

Fig 2. Example of a neutral image.

Building the First Model

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 46, 46, 32)        320

 max_pooling2d (MaxPooling2D  (None, 23, 23, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 21, 21, 32)        9248

 max_pooling2d_1 (MaxPooling  (None, 10, 10, 32)       0
 2D)

 flatten (Flatten)           (None, 3200)              0

 dense (Dense)               (None, 128)               409728

 dense_1 (Dense)             (None, 7)                 903

=================================================================
Total params: 420,199
Trainable params: 420,199
Non-trainable params: 0
```

The first model is the simplest model which was built to check the dataset complexity and the performance requirements of the dataset. The purpose of this simple model is to identify the number of Convolution, MaxPooling and Dense Layers required.

The first model consists of a Convolution layer with 32 filters, a kernel of 3x3, ‘ReLu’ as the activation function, input shape of 48x48. It is followed by a MaxPooling layer of pool size 2x2 and stride size 2. It followed a Convolution layer with 32 filters, a kernel of 3x3, ‘ReLu’ as the activation function and no specific input shapes. It is followed by a MaxPooling layer of pool size 2x2 and stride size 2.

Finally, the Flatten layer takes all the pixel s along with all channels a creates a 1D vector with 10*10*32 = 3200 values. The Dense layer first has around 128 units and then 7 units (depicting the 7 emotions).

Training and Testing the First Model

For training purposes, we use the use the adam optimizer and accuracy as our evaluation metrics. We fit the train data and the validation data, we specified a batch size of 32 and 25 epochs. As the training is complete, we observe that each step took an average of 36 seconds to complete and the training accuracy of an average of 67.84% while the validation accuracy of around 54.71%.

Even though the training accuracy of the first model is high, the validation accuracy is low. When the first model is tested using the cnn.evaludate feature, with the test data it gave a test accuracy of around 53.74%.

Building the Second Model

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 conv2d_2 (Conv2D)            (None, 46, 46, 64)        640

 max_pooling2d_2 (MaxPooling  (None, 23, 23, 64)       0
 2D)

 dropout (Dropout)            (None, 23, 23, 64)        0

 conv2d_3 (Conv2D)            (None, 21, 21, 128)       73856

 max_pooling2d_3 (MaxPooling  (None, 10, 10, 128)      0
 2D)

 dropout_1 (Dropout)          (None, 10, 10, 128)       0

 flatten_1 (Flatten)          (None, 12800)             0

 dense_2 (Dense)              (None, 128)               1638528

 dropout_2 (Dropout)          (None, 128)               0

 dense_3 (Dense)              (None, 7)                 903

=================================================================
Total params: 1,713,927
Trainable params: 1,713,927
Non-trainable params: 0
```

The second model is the improved model which has additional Dropout Layers and L2 regularizers as we needed to reduce the overfitting for the test data.

The second model consists of a Convolution layer with 64 filters, a kernel of 3x3, ‘ReLu’ as the activation function, input shape of 48x48 and L2 kernel regularizers of 0.01. It is followed by a MaxPooling layer of pool size 2x2. In this model we add a Dropout layer of alpha 0.4. Like the 1st model it is followed by a Convolution layer with 64 filters, a kernel of 3x3, ‘ReLu’ as the activation function and no specific input shapes and L2 kernel regularizers of 0.01. It is again followed by a MaxPooling layer of pool size 2x2. Additionally, we add a Dropout layer of alpha 0.2.

Finally, the Flatten layer takes all the pixel s along with all channels a creates a 1D vector with 10*10*128= 12800 values. The Dense layer first has around 128 units with Dense layer with

‘ReLu’ activation function and then 7 units (depicting the 7 emotions) Dense layer with ‘SoftMax’ activation function. However, there is an additional Dropout Layer of 0.1 between both the Dense layers to avoid overfitting.

Training and Testing the Second Model

For training purposes, we use the use the ‘adam’ optimizer and accuracy as our evaluation metrics. We fit the train data and the validation data, we specified a batch size of 32 and 25 epochs. As the training is complete, we observe that each step took an average of 65 seconds to complete and the training accuracy of an average of 48.00% while the validation accuracy of around 47.71%.

When the second model is tested using the cnn2.evaludate feature, with the test data it gave a test accuracy of around 47.74% (which is the lowest amongst the 3 models).

Building the Third Model

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (None, 46, 46, 64)        640

 batch_normalization (BatchN  (None, 46, 46, 64)       256
 ormalization)

 max_pooling2d_4 (MaxPooling  (None, 23, 23, 64)       0
 2D)

 conv2d_5 (Conv2D)           (None, 21, 21, 128)       73856

 batch_normalization_1 (Batc  (None, 21, 21, 128)      512
 hNormalization)

 max_pooling2d_5 (MaxPooling  (None, 10, 10, 128)      0
 2D)

 dropout_3 (Dropout)         (None, 10, 10, 128)       0

 conv2d_6 (Conv2D)           (None, 8, 8, 512)         590336

 batch_normalization_2 (Batc  (None, 8, 8, 512)        2048
 hNormalization)

 max_pooling2d_6 (MaxPooling  (None, 4, 4, 512)        0
 2D)

 conv2d_7 (Conv2D)           (None, 2, 2, 512)         2359808

 batch_normalization_3 (Batc  (None, 2, 2, 512)        2048
 hNormalization)

 max_pooling2d_7 (MaxPooling  (None, 1, 1, 512)        0
 2D)

 dropout_4 (Dropout)         (None, 1, 1, 512)         0


 flatten_2 (Flatten)         (None, 512)               0

 dense_4 (Dense)             (None, 512)               262656

 batch_normalization_4 (Batc  (None, 512)              2048
 hNormalization)

 dropout_5 (Dropout)         (None, 512)               0

 dense_5 (Dense)             (None, 512)               262656

 batch_normalization_5 (Batc  (None, 512)              2048
 hNormalization)

 dropout_6 (Dropout)         (None, 512)               0

 dense_6 (Dense)             (None, 7)                 3591

=================================================================
Total params: 3,562,503
Trainable params: 3,558,023
Non-trainable params: 4,480
```

The third (and final) model is the most imporoved model which has additional Con2D layers before the Dropout Layers and it is added with Batch Normalization layers as it helps us in speeding the training using higher learning rates.

The third model consists of a Convolution layer with 64 filters, a kernel of 3x3, 'ReLu' as the activation function, input shape of 48x48 and L2 kernel regularizers of 0.01. It is followed by a Batch Normalization layer of pool size 2x2. It is followed by a MaxPooling layer. In this model we add a Dropout layer of alpha 0.25. Like the 1st model it is followed by a Convolution layer with 64 filters, a kernel of 3x3, 'ReLu' as the activation function and no specific input shapes and L2 kernel regularizers of 0.01. It is followed by a MaxPooling layer. It is again followed by a MaxPooling layer of pool size 2x2. Additionally, we add a Dropout layer of alpha 0.25.

Finally, the Flatten layer takes all the pixel s along with all channels a creates a 1D vector with 10*10*512= 51200 values. In the third model, we have added 2 Dense layers instead of 1 compared to second model. Both the Dense layers have around 512 units and 'ReLu' activation function. It is followed by a Batch Normalization layer and Dropout layer of 0.4. However, there is an additional Dropout Layer of 0.4 between both the Dense layers to avoid overfitting. Lastly, there is a 7 units (depicting the 7 emotions) Dense layer with 'SoftMax' activation function.

Training and Testing the Third Model

For training purposes, we use the use the 'adam' optimizer with learning rate 0.0001 and accuracy as our evaluation metrics. We fit the train data and the validation data, we specified a batch size of 64 and 35epochs. As the training is complete, we observe that each step took an average of 190 seconds to complete and the training accuracy of an average of 63.99% while the validation accuracy of around 61.43%.

When the second model is tested using the cnn3.evaludate feature, with the test data it gave a test accuracy of around 61.90% (which is the highest amongst the 3 models).

Evaluation Metrics

Based on our dataset and the proposed problem we will be using Accuracy, Recall, Precision and F1-Score as the Evaluation Metrics for our project. Accuracy is the ratio of correctly classified points (prediction) to the total number of predictions. Recall is used to identify the ratio of all the correctly classified emotions with the positively classified emotions. Precision helps us to identify the quality of the model with the ratio between all the correctly classified emotions with the total of truly positive and false positives in the model. The F1-Score can be calculated with Recall and Precision and is a better metric than Accuracy as it allows us to evaluate our model even if the classes are imbalanced (as the dataset is bigger and the model is using realistic dataset).

Even though our classes were imbalanced before, we populated the less values datasets by oversampling it using a data augmentation technique. Hence, in our model we can calculate the training, validation and testing rate by calculating their Accuracy.

Performance Evaluation and Results

Results of First Model

The first model being the simplest, with the least number of convolutional layers, it is not surprising that it is overfitting the data. It can be seen in figures 3 and 4 that the accuracy and loss of the validation set fails to improve as the model is trained. This indicates that the model is greatly overfitting, memorizing the data given as opposed to finding patterns to predict new data.

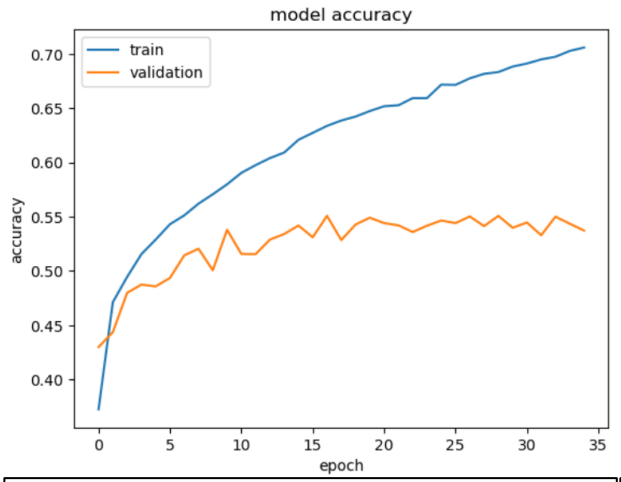

Fig. 3. Model 1 accuracy graph.

Fig. 4. Model 1 loss graph.

After fitting the model, it was then evaluated on the test set, this gave us a test accuracy of 54.19% and a test loss of 1.3576. A confusion matrix was then made to display the number of correct and incorrect predictions per each class. It can be seen in figure 5 that class one, the disgust class which originally had significantly less number of images, was predicted incorrect almost every time. Unfortunately, this misclassification was something we were unable to improve much. It can also be seen that class 3, the happy class which had slightly more images than the other classes, was predicted correctly almost every time.

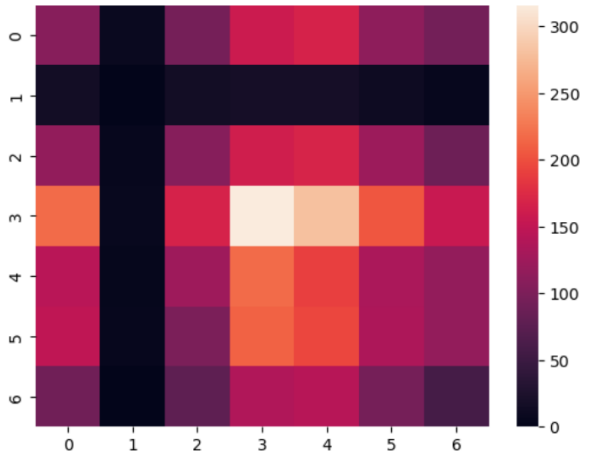

Fig. 5. Model 1 confusion matrix.

The test accuracy of this model being on the low side indicated that we may need to increase the depth of our convolutional neural network and change some parameters to see if that would improve our results. The major overfitting indicated that we more particularly needed to add some dropout layers.

Results of Second Model

For the second model, we kept the number of convolutional layers at 2 and held off adding more at this point and instead focused on correcting the overfitting. To do this, 3 dropout layers were added between the convolutional layers and before the dense layers. It can be seen in figures 6 and 7 that these dropout layers greatly improved the accuracy and loss values from the previous model.

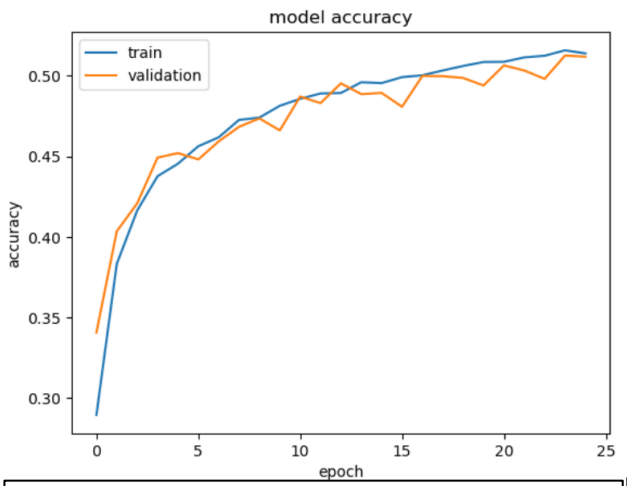

Fig. 6. Model 2 accuracy graph.

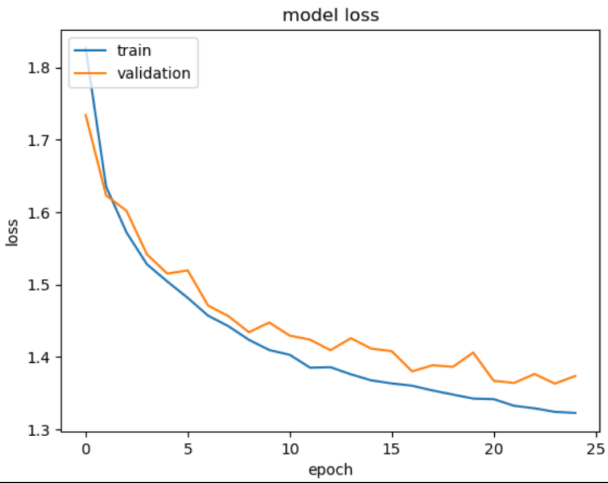

Fig. 7. Model 2 loss chart.

Despite the correction of the overfitting, after performing the model evaluation on the test set, the accuracy dropped to 50.25%, and the loss increased to 1.3628. These values are not

drastically worse than the previous model, but they can still be improved since the main additions to the model so far were the dropout layers. It can be seen in figure 8 that the fear (class 2) class has significantly lost the amount of correctly predicted images.

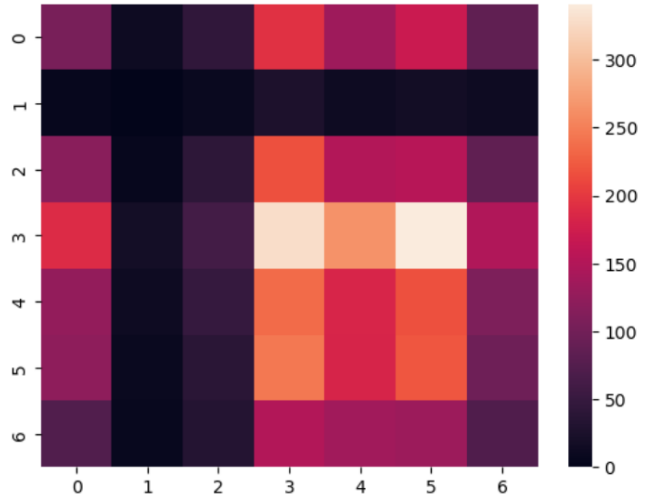

Fig. 8. Model 2 confusion matrix.

This second model was able to improve the overfitting but was not able to improve the test accuracy. This indicates that the next step could be to implement more convolutional layers and possibly more dense layers to improve the accuracy of the model.

Results of Third Model

For the third and final model, we were able to add an additional two convolutional layers and an additional dense layer. The filter size of the layers was increased as more layers were added. A batch normalization was applied for each layer as well. It can be seen in figure 9 that the validation accuracy got slightly off trach with the training accuracy once it got up to around 61%, although this number is higher than previously achieved. For the loss, it started at an unfortunately high number but was able to decrease significantly as the model was trained.

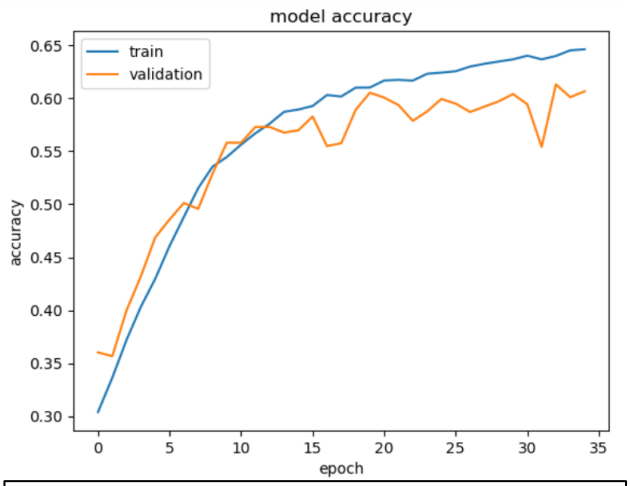

Fig. 9. Model 3 accuracy graph.

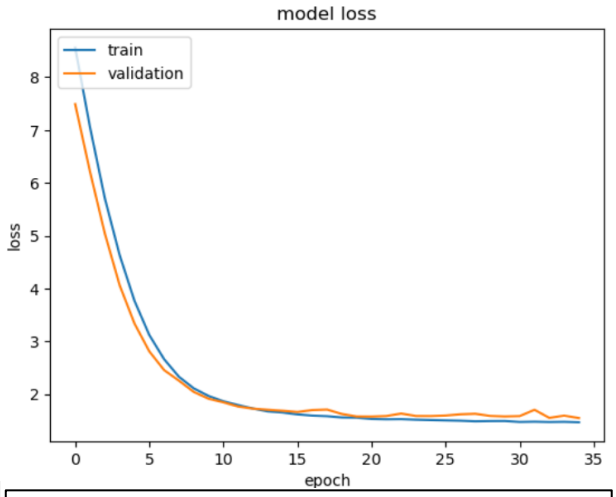

Fig. 10. Model 3 loss graph.

When performing the test evaluation on this model it was found that the accuracy is 62.09% and the loss is 1.5281. Despite the loss increasing, this accuracy value is the highest that we were able to achieve with the additions made to the model. It can be seen in figure 11 that unfortunately the disgust class has not made any improvement but some of the other classes have some more correct predictions.

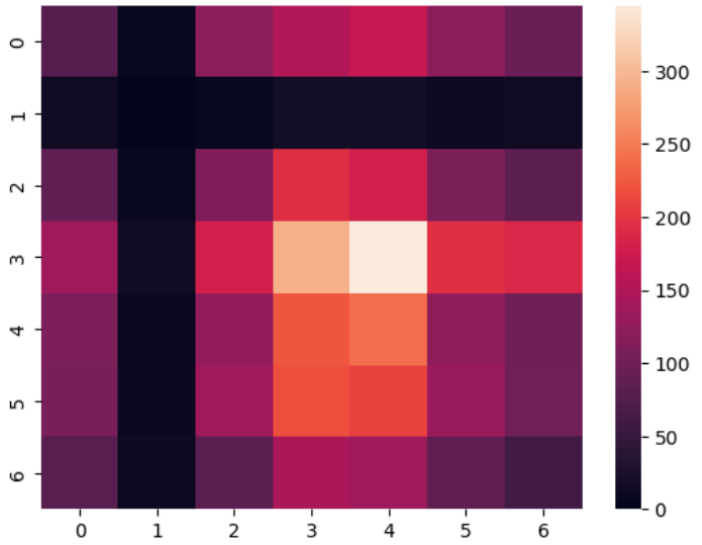

Fig. 11. Model 3 confusion matrix.

This model was able to improve our testing accuracy to the best we have seen yet. Despite this there are still areas that could use improvement, mainly the overall loss and the specific accuracy per class.

Overall Results

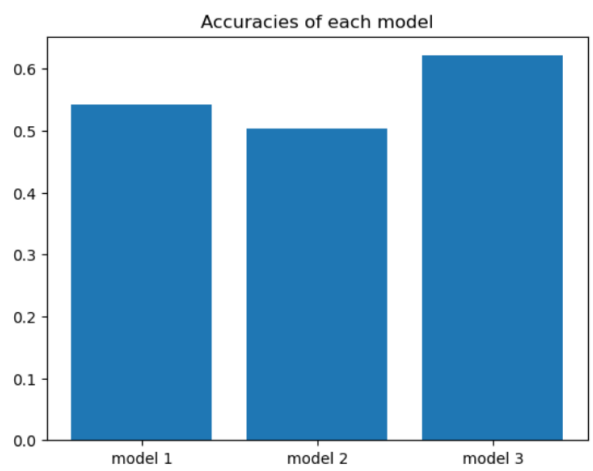

Fig. 12. Bar graph comparing the accuracies of each model.

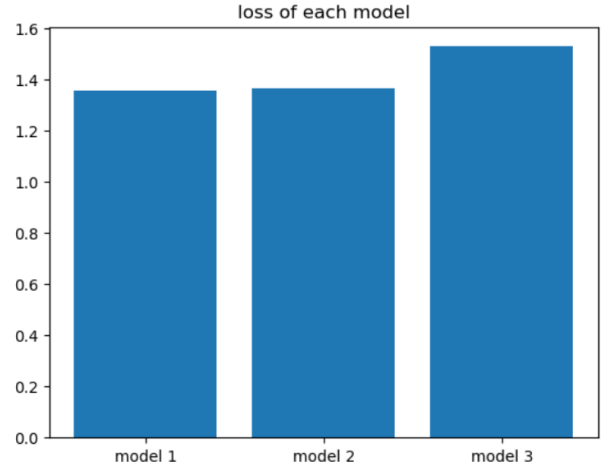

Fig. 13. Bar graph comparing the loss of each model.

The goal for our models was to increase the accuracy as best as possible. It is shown in figure 12 that our latest model does show improvement in accuracy, despite this model having the most loss, as seen in figure 13.

Conclusion and Future Scope

We present an Emotion Classification system in this report that can be utilized to identify the emotion of a person through an image. The thorough analysis of our 3rd model demonstrates that a test accuracy of 61%. From our experiments we understood that more layers without proper parameter tuning is not helpful to increase the accuracy of a model. As observed in the case of 2nd and 1st models, where we increased the number of different layers, but the training and test accuracy decreased. However, we also observed that greater number of convolutional layers resulted in better accuracy of the training set (and testing set) and greater training time (as each step took around 190 seconds to be trained).

The results from the current project have shown that the proposed CNN model can be tuned even further into developing a real-time system for automated detection and recognition of human emotion. Hence, we intend to utilize this project with different visualization software's that can be further integrated with assistive technologies for visually impaired or individuals who cannot properly identify other's emotions. As the current project only utilizes grayscale images, we plan to expand and understand CNN (or CNN like) models, working with other datasets like RGB images, audio, speech, text, and videos.

References

[1] Bhattacharyya, A., Chatterjee, S., Sen, S., Sinitca, A., Kaplun, D., & Sarkar, R. (2021, October 19). A deep learning model for classifying human facial expressions from infrared thermal images. Nature News. Retrieved October 9, 2022, from https://www.nature.com/articles/s41598-021-99998-z

[2] E. P. (Watkins), "4 ways to improve class imbalance for Image Data," *Medium*, 18-Mar-2021. [Online]. Available: https://towardsdatascience.com/4-ways-to-improve-class-imbalance-for-image-data-9adec8f390f1. [Accessed: 03-Dec-2022].

[3] Khanal, S. R., Barroso, J., Lopes, N., Sampaio, J., & Filipe, V. (2018). Performance analysis of Microsoft's and Google's Emotion Recognition API using pose-invariant faces. Proceedings of the 8th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-Exclusion. https://doi.org/10.1145/3218585.3224223

[4] Menon, K. (2021, December 14). How to build powerful Keras Image Classification Models: Simplilearn. Simplilearn.com. Retrieved October 9, 2022, from https://www.simplilearn.com/tutorials/deep-learning-tutorial/guide-to-building-powerful-keras-image-classification-models

[5] NCBI - WWW Error Blocked Diagnostic. (n.d.). Retrieved October 9, 2022, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8437147/

[6] R. R. Hassin, H. Aviezer, and S. Bentin, "Inherently ambiguous: Facial expressions of emotions, in context," *Emotion Review*, vol. 5, no. 1, pp. 60–65, 2013.

[7] Sambare, M. (2020, July 19). Fer-2013. Kaggle. Retrieved October 9, 2022, from https://www.kaggle.com/datasets/msambare/fer2013?select=train

[8] Vemou, K., & Horvath, A. (2021). Facial Emotion Recognition. Tech Dispatch, (1). https://doi.org/10.2804/014217.

[9] X. Xu, Z. Ruan and L. Yang, "Facial Expression Recognition Based on Graph Neural Network," 2020 IEEE 5th International Conference on Image, Vision and Computing (ICIVC), 2020, pp. 211-214, doi: 10.1109/ICIVC50857.2020.9177430.

[10] Frith C. Role of facial expressions in social interactions. Philos Trans R Soc Lond B Biol Sci. 2009 Dec 12;364(1535):3453-8. doi: 10.1098/rstb.2009.0142. PMID: 19884140; PMCID: PMC2781887.

[11] Mehrabian A. Communication without words. *IOJT*. 2008:193–200. https://www.taylorfrancis.com/chapters/edit/10.4324/9781315080918-15/communication-without-words-albert-mehrabian

[12] Ekman P., Friesen W.V., Ellsworth P. *Emotion in the Human Face: Guide-Lines for Research and an Integration of Findings: Guidelines for Research and an Integration of Findings.* Pergamon; Berlin, Germany: 1972 https://books.google.ca/books?hl=en&lr=&id=dOFFBQAAQBAJ&oi=fnd&pg=PP1&ots=

ziFCqMZyQb&sig=wwlWcaSlKH1s5AAPMeLPfGO8_yc&redir_esc=y#v=onepage&q&f=false

[13] M. Lee, Y.K. Lee, M.-T. Lim, T.-K. Kang. Emotion recognition using convolutional neural network with selected statistical photoplethysmogram features Appl. Sci., 10 (10) (2020), p.3501

[14] Srinivas, S., Sarvadevabhatla, R. K., Mopuri, K. R., Prabhu, N., Kruthiventi, S. S., &Babu, R. V. (2016) “A taxonomy of deep convolutional neural nets for computer vision

[15] K. M. Goud and S. J. Hussain, "Estimation of Emotion using CNN," 2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC), 2021, pp. 1561-1565, doi: 10.1109/ICESC51422.2021.9532763.

Appendix A: Team Contribution

Code:	**Who worked on it:**
Data preprocessing – splitting data	Jenna
Data preprocessing – evening out classes	Jenna
Data preprocessing – example image	Jenna
Building models x3	Brijesh (Conversations on what to add/change in the models were had between both of us, suggestions were made by both of us on what to do, Brijesh ultimately implemented the build of the models.)
Evaluating models x3– loss and accuracy graphs + example prediction	Brijesh
Evaluating models x3 – confusion matrix and classification reports	Jenna
Report:	
Abstract	Jenna
Introduction	Brijesh
Literature Survey	Brijesh
Problem statement	Jenna
Proposed techniques - dataset	Jenna
Proposed techniques - preprocessing	Jenna
Proposed techniques - models	Brijesh
Evaluation Metrics	Brijesh
Performance evaluation and results	Jenna
Conclusion and future scope	Brijesh
PowerPoint:	
Title page up to conclusion	Jenna
Conclusion and future scope	Brijesh

Appendix B: Contribution Percentage

Brijesh:	**43**
Jenna:	**57**