Real Time Facial Expression Recognition for Online Lecture

By

Haobang Wu

Supervisor: Prof. Ben Liang

April 2021

B.A.Sc. Thesis

Division of Engineering Science
UNIVERSITY OF TORONTO

Real Time Facial Expression Recognition for Online Lecture

By

Haobang Wu

Supervisor: Prof. Ben Liang

April 2021

B.A.Sc. Thesis

Division of Engineering Science
UNIVERSITY OF TORONTO

Abstracts

Online teaching has become an essential method for teachers to provide lectures during the pandemic. One big problem with online lectures is that it is hard for the lecturers to see the students' facial expressions. According to research, students' facial expressions are essential for the lecturers to understand the students' comprehension of course materials. There are several related research about recognizing human facial expressions. However, most of them did not consider some expressions such as "Enlightened, "Confused," or "Bored," which are common in a lecture. The thesis aims to help improve the quality of teaching by providing lecturers with the students' facial expressions using a program. The program is implemented on students' computers and can capture their faces from cameras, recognize some common and useful facial expressions, and send them to the lecturer periodically during an online lecture. The key methods of the thesis are: 1) Use the face recognition algorithm provided by OpenCV to capture faces. 2) Train a convolutional neural network model to recognize the expression "Happy," "Surprised," "Neutral," "Enlightened," "Confusion," and "Boredom." 3) Use EMQX, a message protocol to transfer the expression information. The program's performance is acceptable because it can successfully capture the face, recognize the expressions at an accuracy of around 80%, and successfully transfer the expression information.

The thesis is innovative in the area of recognizing human facial expressions because it includes some expressions that are seldomly researched by others. The dataset, the deep learning model, and the results can be used for other research teams.

Acknowledgements

I would like to express the deepest appreciation to my supervisor, Professor Ben Liang, who has been spending a whole school year helping me on this thesis paper. Prof. Liang inspired me on choosing the thesis topic, provided his guidance and supervised the whole process. Without his persistent help, I would not have a such a wonderful research experience.

I would also like to thank to Sara Zhalehour and his team from Bahcesehir University, Department of Electrical and Electronics Engineering. The team generously provided their dataset to help me with my thesis work.

Table of Contents

List of Figures

1. Introduction

Online teaching has become an essential method for the university to continue holding semesters for students, researchers, and professors during the pandemic. The online conference platforms are playing a significant role in delivering the online lecturer. These platforms can help restore the traditional lecture model by allowing the lecturers to show the course materials through screen sharing or to write notes on the virtual whiteboard. However, what online teaching can not restore is efficient interaction between the lecturers and their students. Although the online conference platforms can allow attendees to see and talk to each other through the webcams, it is not realistic for the lecture to interact with many students due to network bandwidth and video processing limitations. More specifically, lecturers are having trouble seeing the faces of most of the students and reading their facial expressions. According to research [1], the students' facial expressions are the most used nonverbal communication mode in a lecture and are related to their emotions, which can help the lecturer recognize their comprehension of the lecture. Suppose there is a tool that can provide students' facial expressions to the lecturers periodically during an online lecture. In that case, it can restore the reading facial expression scenarios, and the lecturer can get useful interaction feedbacks from their students and can adjust their ongoing lectures accordingly.

There are several types of research about building programs that can recognize human facial expressions using different methods these days. Among the research, many groups choose to use machine learning methods, such as training a deep learning neural network to achieve the goal. Most of the groups focus on the seven most common human facial emotions, including happy, sad, surprised, angry, disgust, fear, and neutral. However, expressions like boredom, confusion, and enlightened that a student may frequently show during a lecture are always ignored. There is very little research about recognizing these three expressions, and there is hardly any dataset that includes these expressions as well. Therefore, I decided to focus on recognizing the commonplace facial expressions students would have during a lecture. The facial expressions are **boredom, confusion, enlightened, happy, surprised**, **and neutral.**

The goal of my thesis is to design and build a program that can automatically capture students' faces from cameras, can recognize some common and useful facial expressions, including **boredom, confusion, enlightened, happy, surprised**, **and neutral**, and can send the expression information to the lecturer periodically during an online lecture. To accomplish the goal, three objects were achieved step by step:

1. Find a method to capture students' faces on the real-time video with their camera.
2. Design and build a deep learning model that can accurately recognize the target facial expressions
3. Build an output pipeline that can transfer the information to the lecturer’s computer

The purpose of my thesis project is innovative and important. Since there are some emotions that other researchers seldomly considered, I can provide a good supplement in the area of recognizing human facial expressions by sharing my thesis work. The dataset, which contains the rare expressions, and the results in the paper can help other researchers interested in similar topics.

2. Literature Review

Studies in computer vision show that machine learning algorithms are effective in human facial expression recognition. Among all the subfields of machine learning, deep learning is one of the most popular algorithms used by researchers.

The big picture of the deep learning algorithm is to train a deep learning model to predict facial expressions when given a picture of a human face. A deep learning model is a model with a logical structure similar to how a human learns and draws conclusions. The structure is a layered structure inspired by a human's brain's neural network system and is therefore called an artificial neural network. The general way for machine learning researchers to use a deep learning model is to build a proper dataset with enough unbiased data, train and tune the model with training and testing data, and make predictions or classifications given by unseen inputs.
In recognizing human facial expressions, a proper dataset containing enough wanted facial emotions from different people and an accurate deep learning model that can predict the facial expressions given by a picture of the human face are needed.

Paper written by Gory et al. [2] gives an overview of how different machine learning models perform on recognizing facial expressions. The research group applied and tested several machine learning algorithms, including AdaBoost, Logistic Regression, and two deep learning model: Dense Neural Network (DNN), Convolutional Neural Network (CNN), on recognizing one of the seven most common expressions (happy, sad, surprised, angry, disgust, fear and neutral) in their paper. They used 35887 images of faces from the dataset “Challenges in Representation Learning: Facial Expression Recognition Challenge” (FER2013) [3] to train and test on these four different models. The result (Figure 1) claimed that the CNN model had the highest accuracy but also took the longest training time. However, the accuracy of the CNN model was only around 60%, while the accuracy of the other three models was only 40% on average. More specifically, the models got lower accuracy on recognizing emotions like disgust, anger, and fear. The researchers claimed the reason was that the data of those expressions are far less than the others, and those expressions are similar to each other on human faces. Gory

et al. provided a comparison performance test on different machine learning models and proved that CNN might be the most suitable.

Table 1. Accuracy, precision, and recall scores for tested algorithms

Algorithm	Accuracy	Precision	Recall
CNN	**0.64**	0.67	0.64
DNN	0.39	0.46	0.39
Logit-reg	0.36	0.40	0.37
AdaBoost	0.33	0.47	0.33

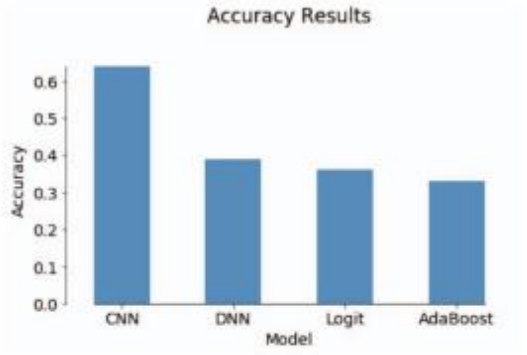


Figure 1 Table and plots of result

As one of the most remarkable developments in the computer vision field, Convolutional neural network (CNN) (figure i) has a significant ability to recognize objects from an image.[4] CNN is a type of deep learning model that contains convolutional layers and dense layers. For each convolutional layer, several convolutional kernels are used to detect a certain feature of the image. The different convolutional layers can detect different features of an image. (Figure 2). With all the convolutional layers stacked together, the image's features will be collected and delivered to the following fully connected layers. The fully connected layers can learn the non-linear combinations of the high-level features extracted by the convolution layers and do the classifications. (Figure 3).

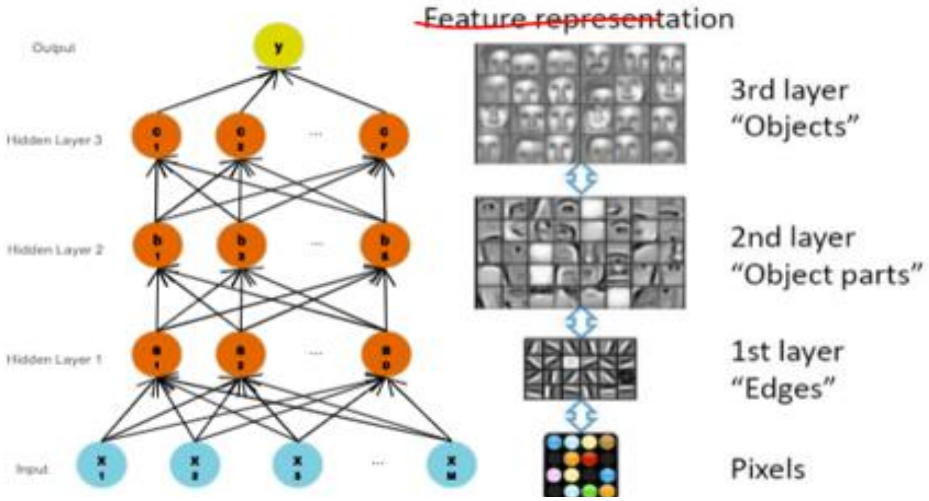


Figure 2illustration of convolutional layers

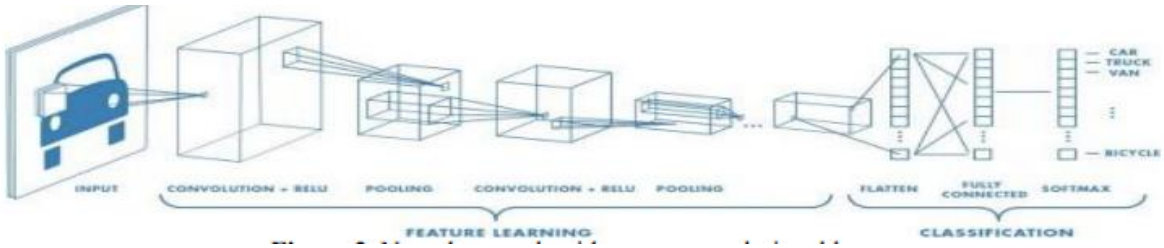


Figure 3 illustration of whole CNN model

To investigate how the CNN model can be used to do the recognition, I read the paper of Dr. Hussain & Dr. Balushi [5]. The authors propose using a deep CNN model to do a real-time face emotion classification and recognition. The paper's core approach is to build a CNN model based on a well-known model called VGG16, as shown in figure 4. VGG16 is a CNN model with a structure containing five different convolutional layers, each with different numbers and sizes of kernels and three fully connected layers. The authors built a CNN model with the same structure and trained it using the KDEF dataset [6]. The KDEF dataset contains 2901 images of human face emotions (the same seven common emotions above). The final accuracy of the trained model described in the paper is 0.88. (Result in Appendix A)

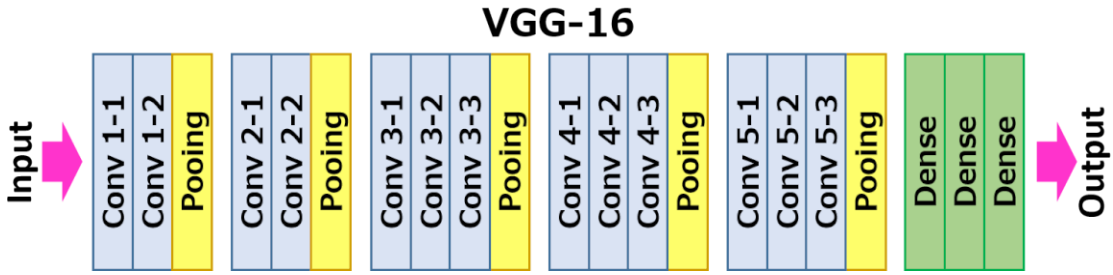


Figure 4VGG16 structure

Besides the two papers mentioned above, Liu et al. [7] and Fathallah et al. [8] also claimed that the convolutional neural network has an excellent performance on human facial expression recognition. Liu et al. designed a CNN model with an accuracy of 65%, making them a fifth of the leaderboard for dataset FER-2013. (Appendix A.) Fathallah et al.'s CNN models can have an accuracy of 90% when testing on datasets CK+, RaFD, and MUG. (Appendix A.) A common fact in the three research is that a dataset with

thousands of data is required to train such complex CNN models. However, the data that can be potentially used for my thesis is not sufficient. Most of the available collected datasets from other researchers do not contain emotions like boredom, confusion, and enlightened. It is hard to build a large dataset in a short time. Therefore, a technique that can do expression recognition on a small dataset is needed.

Heidari & Fouladi-Ghaleh [9] introduced and applied such a technique: transfer learning in their paper. Transfer learning is a machine learning technique that a model trained on one task can be re-purpose on a similar second task. Usually, training a deep learning model requires many data and a long training time. Transfer learning can help to utilize the knowledge of a pre-trained model onto a new model so that only a portion of the parameters in the new model needs to be trained. Therefore, the new model can be trained with a smaller dataset and within a shorter time.

Heidari & Fouladi-Ghaleh used transfer learning for face recognition using small-sample datasets. The authors' purpose is to use a Siamese network, which consists of two similar CNNs to recognize if a pair of human face pictures belong to one person. They used the LFW dataset [10] that contains small samples for every single person. To build a functional mechanism with only a few samples, the authors used the pre-trained VGG16 model. The pre-trained VGG16 is trained with a dataset called ImageNet [11], which contains around 14 million images in 1000 categories. The model is widely used in all kinds of classification tasks. The research team froze the trained parameters of the first four convolutional layers (these parameters are untrained) and only trained the last convolutional layer and the fully connected dense layers (Figure 5). The purpose of using the pre-trained VGG16 model is to extract the features of input human face pictures in the first place. With the face features obtained previously, the authors designed the following layers to find the similarities/differences of the two faces and decide whether they belong to one person. According to the paper, the result accuracy is 95.62%, which is higher than some other common methods. (Appendix A)

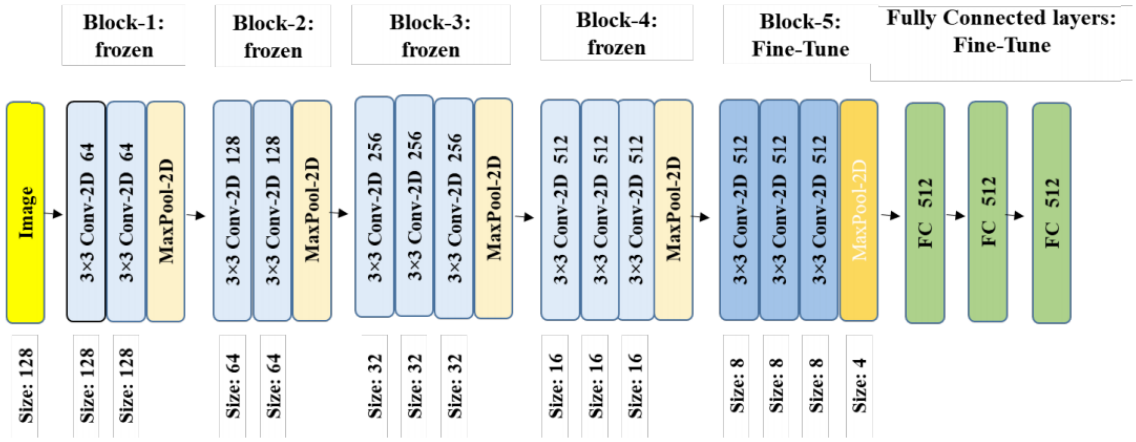


Figure 5 Transfer learning on VGG-16

Although the goal of the paper is different from mine, the idea of using transfer learning with the help of pre-trained models will be potentially helpful in the situation where data is not enough. The key to transfer learning is to choose a proper pre-trained model.

Caroppo et al. [12] provided a comparison of the three widely used pre-trained models for facial expression recognition: VGG16, AlexNet, and GoogLeNet by evaluating them on four different benchmark datasets: FACES, Lifespan, CIFE, and FER2013. The authors kept most layers of the pre-trained models and replaced the output level of the structure that is combined with three different classifiers (i.e., Random Forest, Support Vector Machine, and Linear Regression). With the experiments on the different dataset-model-classifiers combinations, the authors claimed that the overall approach of using a pre-trained CNN model was successful compared to some baseline models. Among the three different models, the VGG16, in combination with the Random Forest classifier, had the best performance on each dataset. (Appendix A) The sufficient experiments and results provide readers a good view of the performance of transferred learning on different models, and it is helpful for me to make decisions in my thesis.

My thesis's big picture is mainly relevant to those research papers, but the difference in the dataset (numbers and emotion types) needs to be considered. A convolutional neural network is used in my work because of the model's good performance in other related papers. Because there is not enough data (especially the pictures for emotions like

boredom, confusion, and enlightened), a transfer learning method using a pre-trained CNN model is a proper way to achieve my goal.

3. Methods

Let's first recap the goal of the thesis: to design and build a program that can automatically capture students' faces from cameras, can recognize the most common and useful facial expressions, including **boredom, confused, enlightened, happy, surprised**, **and neutral**, and can send the expression information to the lecturer periodically during an online lecture.

There are three steps to accomplish the goal: 1) Capture students' faces with their cameras 2) Design and build a deep learning model that can accurately recognize the target facial expressions. 3) Build an output pipeline that can return the information to the lecturer. Different methods are used for each step.

3.1 Capture the face.

For step one, in order to capture students' faces, a popular library in the Computer Vision field called OpenCV [12] is used. OpenCV contains a face detection function named "detect_face" (the specific code implementation is in Appendix B) that can detect a human face from an image by returning the coordinates of the four points. These points are the four corners of a rectangle bounding the face. (Figure 6.)

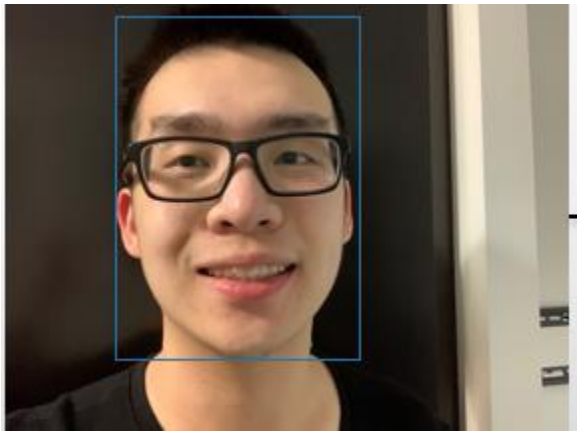


Figure 6 rectangle bounding the face

With the coordinates of the four corners, the faces can be easily cropped from an image. The cropped faces are the data used to train and test the deep learning model in the next step.

The capture method is important because it is used in data processing to build a dataset and implement the complete functional program. In real life, a camera captures not only students' faces but also the background behind the faces, just like in Figure 6. The background is the noise in a training process, and I do not want the model to be confused or misled by the background. By eliminating the noise, the model can do classification by only learning the features of the faces.

The capture method I used is a very common method for researchers who wants to crop out the human face from an image. The method is easy to implement with OpenCV, and it can capture faces with high accuracy but within a short time. One limitation of the method is that the function may sometimes recognize other objects as human faces if low confidence is set.

3.2 Recognize the facial expressions.

The second step is to recognize the facial expressions of the cropped faces from images. The method for this step is a typical machine learning method: build and train a deep learning model with valid data. The method is divided into three small steps: 1) build a proper dataset; 2) design and build a deep learning model; 3) train and test the model.

3.2.1 Build dataset

To build the dataset, I collected 1073 images: 200 images each for **expression** boredom, confusion, happy, surprised**,** and neutral and 73 images for expression enlightened. The images are collected from three different **sources.** One-third of the data are the selfies of my acquaintances (Figure 7). The image providers are asked to make the facial expressions under my instructions. Therefore, all the images are very consistent with similar backgrounds, a similar distance/ angle from the cameras, and similar exaggeration. One limitation of this part of the dataset is that most of the providers are Asians. It is common sense that people of different races have different facial features. For example, Asians are more likely to have black hair, yellow skin color, and relatively small eyes and noises, while white people are more likely to have hair with brighter

color, white skin color, bigger eyes, and taller noises. The difference of the face features may mislead the deep learning model unexpectedly, so it is very crucial to collect images from people other than Asians to build an unbiased dataset.

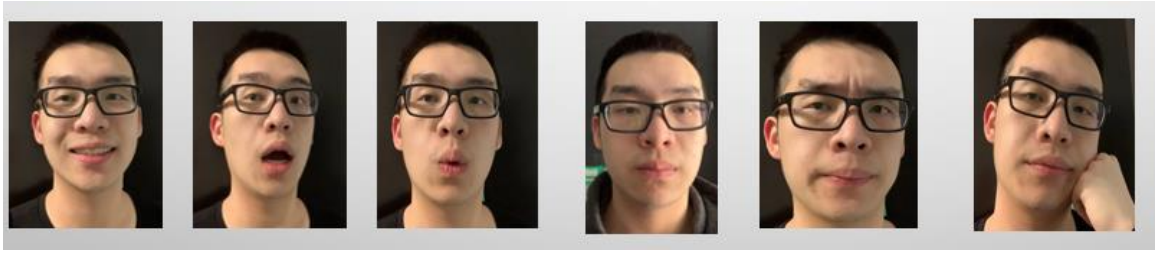


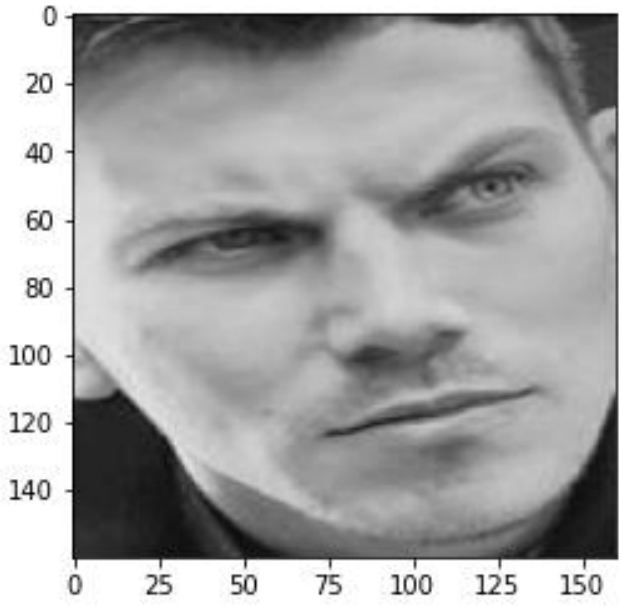
Figure 7Asian dataset

Another one-third of the data comes from open sources, including the dataset provided by Sara Zhalehou and Bahcesehir University ECE department, *"Bahcesehir University Multimodel Face Database of Spontaneous Affective and Mental States"* [13] and dataset FER2013[3]. The data include images of people of different races and gender, and they are very consistent as well. However, the open dataset only contains a few data about the expression of Boredom, enlightened, and Confusion. Most of the data are about the other three popular expressions: Happy, Surprised, and Neutral. In order to build an unbiased dataset with equivalent numbers of data for each expression, I continued to collect more data to fill up the gap between the popular expressions and the less popular ones.

The final one-third of the data is collected from online sources like image webpages and social media. This part of the data mainly contains images of expression Boredom, enlightened, and Confusion. I successfully collected enough images of expressions Boredom and Confusion (to make the number of these images even with the number of the three popular ones.). However, I failed to collect enough images for expression enlightened due to a lack of resources. This part of the data is not consistent because they are coming from different sources. One important note is that all the data is collected and labeled only on my own opinions, so bias in the data can not be avoided.

With the collected raw data, a pre-processing step is needed before the deep learning model train. I first used the face detection algorithm of OpenCV to crop the faces from the images. The cropped faces are then converted into the greyscale and are resized into a resolution of 160*160 pixels (figure 8.) A greyscale face image can reduce the unexpected problems caused by different skin colors, and uniform image sizes are required for the model training.

Besides, in order to increase the number of training data, data augmentation method is used. The new generated images are the original images in the training data after sheared, zoomed, flipped and rotated in certain degrees.



- *Figure 8Example of image after pre-processing*

Figure 9 Data augmentation example

The label of the data is represent using one hot vectors. In this case, the one hot vector is a vector with six values of 1 or 0. There is one value of "1" represent the corresponding category of the data and the rest are "0". (ie. Label for expression Happy: [1,0,0,0,0,0] Label for expression Surprised is [0,1,0,0,0,0] etc.)

The dataset is consistent since most of the images are captured under similar circumstances. The dataset is comprehensive since it contains faces of people of different

races and gender. However, there are also some limitations of the dataset. It is not perfectly unbiased because some of the images are collected and labeled only on my opinions. The dataset is also not sufficient due to the lack of images of the three less popular expressions.

3.2.2 Design and build a model.

The purpose of the model is to recognize the facial expressions after training using a dataset with limited data. Inspired by the research of Caroppo et al. and the paper of Heidari & Fouladi-Ghaleh, I built a convolutional neural network by applying the transfer learning method. The pre-trained model I chose is VGG16, provided by a python library called Keras [14]. As I mentioned in the "Literature Review" section, VGG16 contains five convolutional blocks and three fully connected layers. My CNN model had the same structure but with some addons and modifications. The input layer is modified to accept an input image of size 160*160*3. There are two convolutional layers and one 2D max-pooling layer for each convolutional block. The five convolutional blocks contain 64, 128, 256, 512, and 512 convolutional kernels of a size of 3*3 for each convolutional layer. Three fully connected layers with a number of 512 neural follow the convolutional blocks. Each layer is followed by a "RELU" activation function (Equation 1). Besides, two dropout layers of a rate of 50% are added after the second and the third fully connected layer. The final output layer contains six output neural followed by a "Softmax" activation function (Equation 2). The last layer's output is a vector consisting of six values, and each value represents the prediction probability of each category. The prediction of the model is the expression with the highest prediction probabilities. The complete model structured is showed in figure 10. I chose to freeze the first four convolution layers so that the parameters are not trained. The last convolutional block plus the following linear layers are trainable. Although the training data is not sufficient, the fact that fewer parameters are trainable can guarantee an acceptable result of the model performance. The two dropout layers are helping to fight against potential overfitting problems caused by a lack of data.

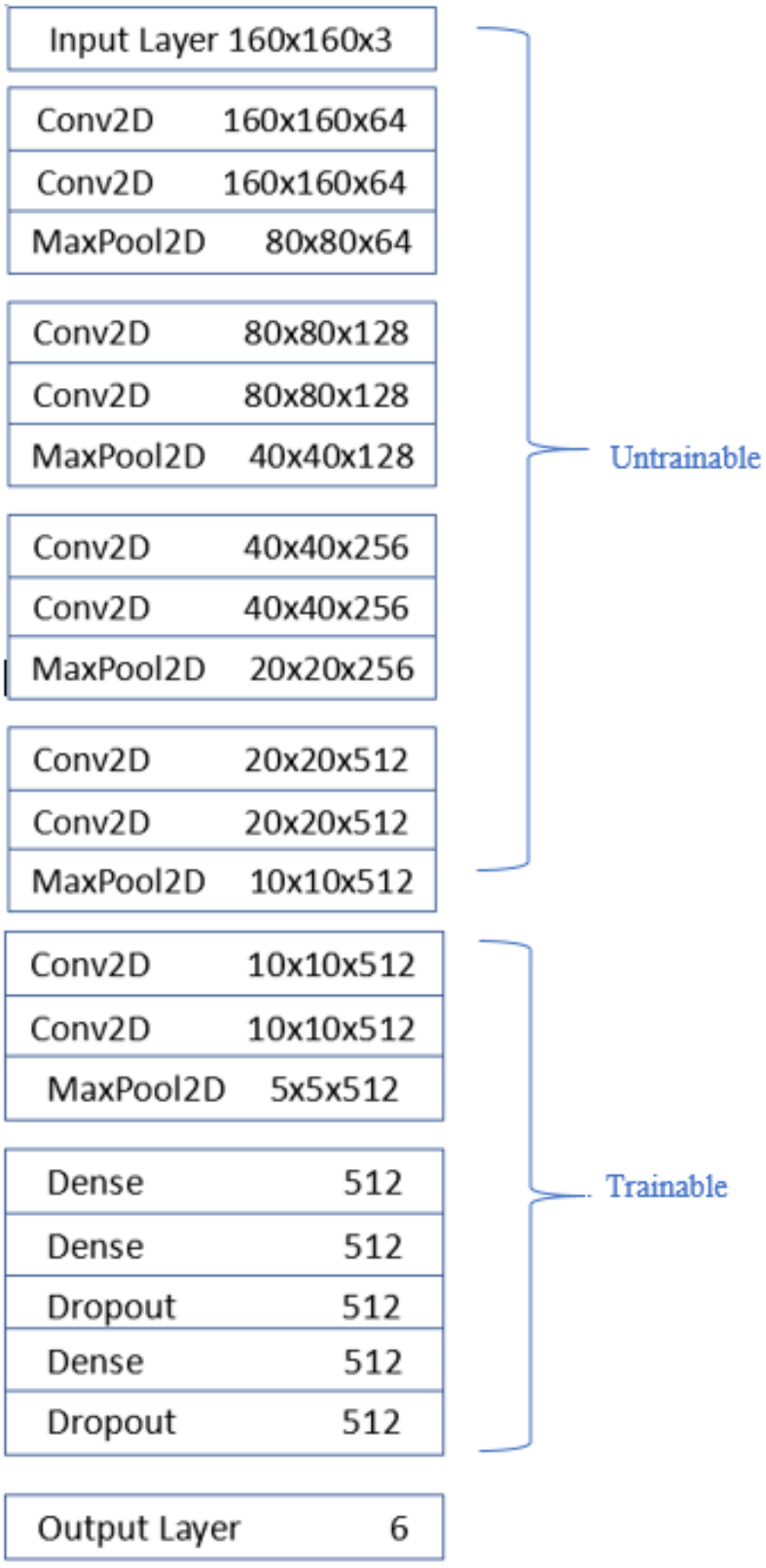


Figure 10Full structure of the CNN model

For model training, the dataset is divided into training dataset and validation dataset with a ratio of 4:1. The input of the CNN model is the 160x160x3 cropped face images from the training dataset. The output of the model is a list of prediction probabilities for the six expressions. The model is trained with a learning rate of 0.001 and is trained for 100 epochs. For each epoch, the loss is calculated with a cross-entropy loss function (Equation 3), and the model is optimized with stochastic gradient descent (Equation 4). The accuracy of the output equals the portion of the number of correct predictions among all predictions (Equation 5).

For testing the performance of the trained model, three-fold cross-validation testing is used (Figure 11). The three-fold cross-validation testing is a three times validation test with three different combinations of training and testing dataset. Each unit validation testing split the dataset randomly and ensured the validation data is different for each validation test. The final testing result is the average of the three unit-testing results. Cross-validation testing can provide a general performance for a trained model and avoid extreme data split situations that lead to an abnormal performance. In the case of my thesis, cross-validation can prevent a situation where the validation data happens only to contain images of Asians or only of women so that the model is evaluated with bias. The model's performance is evaluated by watching the pattern of the training and validation loss/accuracy graph (against the number of epochs) and the accuracy of predictions for each expression. The whole coding for training the model is written using python. The python library Keras is used to create, train and test the model.

$EQ1: RELU: Y = \max(0, X)$
where the input is X, output is Y

$$EQ2: SoftMax: \sigma(z)_i = \frac{e^{z_i}}{\Sigma e^{z_j}}$$
where $z_i$is $\sigma(z)_i$ is the softmax probabilities for the ith category prediction.

$EQ3: Cross\ entropy\ Loss = -\Sigma t_i * \log(p(x))$,
where p(x) is the softmax probabilities and t_i is the truth label.

$$EQ4: SGD: w = w - \eta\Sigma\frac{\partial Loss}{\partial w_j},$$

where Loss is cross-entropy loss and w represents the weights.

$$EQ5: Accuracy = \frac{\#\ of\ correct\ predictions}{\#\ of\ all\ predictions}$$

Figure 11Illustration of cross validation

3.2.3 Implement the methods.

The method of how to build the CNN model for training is clearly stated in the previous section. In my thesis work, the method is implemented in different ways to get different aspects of the results. The first implementation uses the CNN model to train with the data that contains all six facial expressions. (Boredom, Confusion, Neutral, Enlightened, Happy, and Surprised.) Because the number of images about Enlightened is only 73, which is much less than the number of other expressions (200), I have to choose a dataset that contains 73 images for each expression for the training/testing procedure. Although the dataset does not have a bias on any expression since there is the same amount of data for each category, the total data is insufficient to train my complex CNN model. The insufficient data may lead to a result that can be badly overfitting on training data. This is the trade-off I have to make due to the limited size of my dataset. The result of the implementation is shown in the later “Result” section.

The second implementation uses a slightly different CNN model from the first one. Because the result from the first implementation implies that the model's performance in recognizing the expression "Enlightened" is very poor, I chose to evaluate the model without the "Enlightened." Therefore, the new CNN model is almost the same as the

previous one except that its output layer contains five neural instead because the model only needs to classify the other five expressions. This time, the numbers of data for training and testing is larger: 200 images for each expression. With more training data involved and less output category, the result of the second implementation should have a better performance than the first one. The result is also shown in the later "Result" section.

The third implementation is a combination of two CNN models. (Figure 12.) After analyzing the second result, I found out that the CNN model has trouble distinguishing between expressions “Boredom” and “Confusion.” To improve the performance, I chose to use two CNN models. The purpose of the first model is to classify four expression categories: Happy, Surprised, Neutral, and a new category B+C, which is a category that includes both Boredom and Confusion. The first model is trained regarding Bored and Confused faces as one category; it has an output layer with four neural now each representing one category. To train the first model, a dataset with 200 images for each category is used. The second model is a CNN model for binary classification. This model is used to classify between expressions “Boredom” and “Confusion,” so it contains a binary output layer. If the output of the first model is the “B+C” category for an image, the image will go through the second model to be further classified as either “Boredom” or “Confusion.” The second model is trained using 200 Boredom images and 200 Confusion images. After combining the two models, the implementation can output a classification for the five facial expressions. The method is innovative and sounds reasonable, and you can expect a better result than the second implementation.

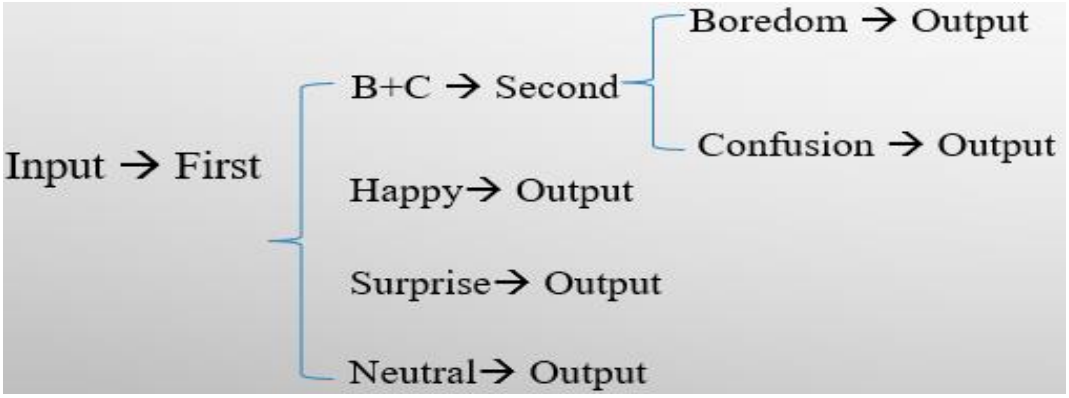


Figure 12illustration of third implementation

3.2.4 Expected results and limitation

The pre-trained VGG16 CNN model already proved that it is a useful technique in various kinds of classification tasks. Since my CNN model has similar structures and contains a lot of untrained parameters, it is expected that the model will have reasonably acceptable results. However, due to the limited size of the dataset and the data's quality, the CNN model is likely to overfit more or less on training data.
According to the results of other research, the performance on recognizing expressions like "Happy" and "Surprised" will be better than recognizing "Neutral," Boredom," and "Confused." The reason can be related to these expressions' features, where "Happy" and "Surprised" involve more apparent motion of eyes and mouth while the other three only involve changes on a small part of the face.

3.3 Inform the facial expression to the lecturer.

After the program captures students' faces and then recognizes their facial expressions, it should inform the information to the lecturer during an online lecture. The method to imform the facial expressions to the lecturer is to build a message transfer protocol between the students' computers and the lecturer's computer. I used EMQX [15], a real-time MQTT [16] message Broker for the Internet of Things (IoT), as the protocol. MQTT is a standard messaging protocol that is designed as an extremely lightweight publish and subscribe messaging transport. The protocol is ideal for connecting remote devices with a small code footprint and minimal network bandwidth. EMQX provides such a MQTT and is easy to implement with python code. The message transfer protocol is composed of two parts: "Publisher" and "Subscriber."
"Publisher" acts as a message sender and should be implemented on students' computers. The "Publisher" can start the program to read and recognize students' facial expressions using their cameras and send them to the "Subscriber." The "Subscriber," acting as a message receiver, should be implemented on the lecturer's computer, and it can constantly receive the facial expression output from students' computers. EMQX can provide 1-N connections (one "Subscriber" and multiple "Publisher") so that one lecturer

can receive a message from multiple students. EMQX can also set the message to deliver intervals so that the message will be delivered periodically. In my thesis, the protocol is only tested between one “publisher” and one “Subscriber.” Further tests and improvement are included in the future step in the following sections. An implementation of the EMQX for python is shown in Appendix B.

.

4. Results and Discussion

This section contains the results and discussion about the three implementations that are mentioned in the “Method” section. Besides, a short demonstration of how the program transfers facial expression information using EMQX protocol is also included.

4.1 Results and discussion about first implementation

The first implementation is to use the CNN model to recognize all six facial expressions. (Boredom, Confusion, Neutral, Enlightened, Happy, and Surprised.) The CNN model is the modified model from pre-trained VGG16 and has an output layer with six output neural. The dataset used for this implementation contains 438 images in total: 73 images for each emotion. The data is divided into a training/validation dataset with a ratio of 4:1. Before the training, the data is enlarged using the default data augmentation method provided by Keras. The model is trained with cross-entropy loss, SGD optimizer, a learning rate of 0.001. The performance of the model is evaluated using three-fold cross-validation. There are three different sets of the training/validation dataset, and the performances of the model on all the sets are counted.

Figures 13 and 14 are the plots for the training/validation loss against the number of epochs and the training/validation accuracy against the number of epochs for the model performed on one of the sets. From the plot of loss vs. epochs, it is clear that the validation loss (orange curves) stopped decreasing after around 30 epochs and finally floated around a loss of 0.8 to 1.0 while the training loss (blue curves) kept decreasing and converged to zero in the end. A similar pattern also showed in the plot of accuracy vs. epochs. The validation accuracy increased very slowly and floated around 72%. Table 1 shows the average accuracy of the cross-validation data. The specific accuracy for each set of validation data is shown in Appendix C. The specific accuracy on recognizing each of the six expressions is shown in the table. The model has accuracies around 80% for Happy, Surprised, and Neutral. Accuracies for Boredom and Confusion are lower and are around 70%. Accuracy for recognizing Enlightened faces is the worst and is only 56%. The total accuracy for the model is 72%.

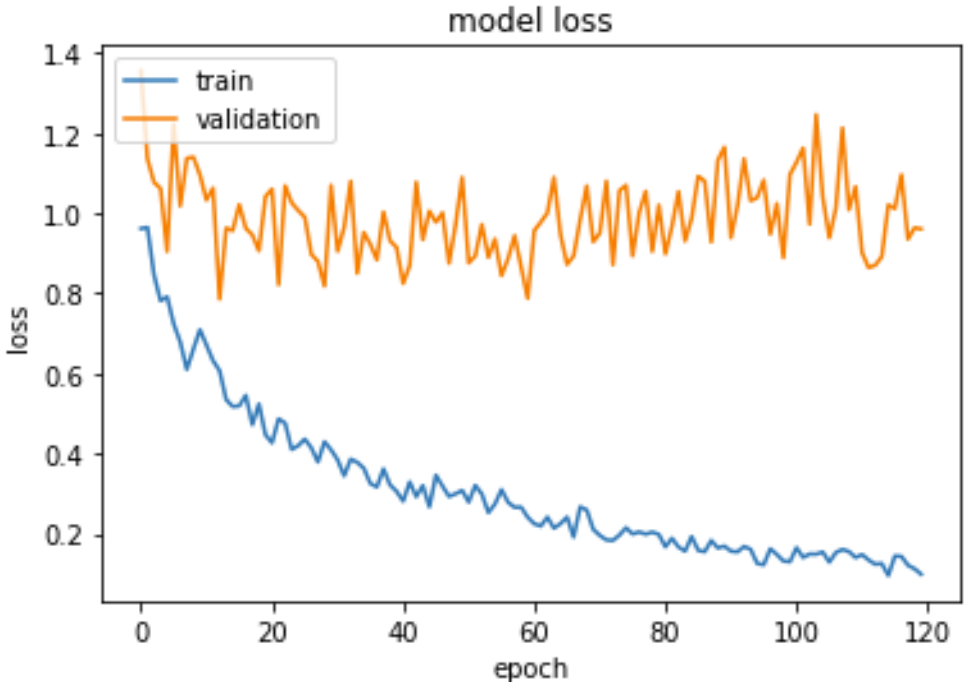


Figure 13 Loss vs Epochs plot for first implementation

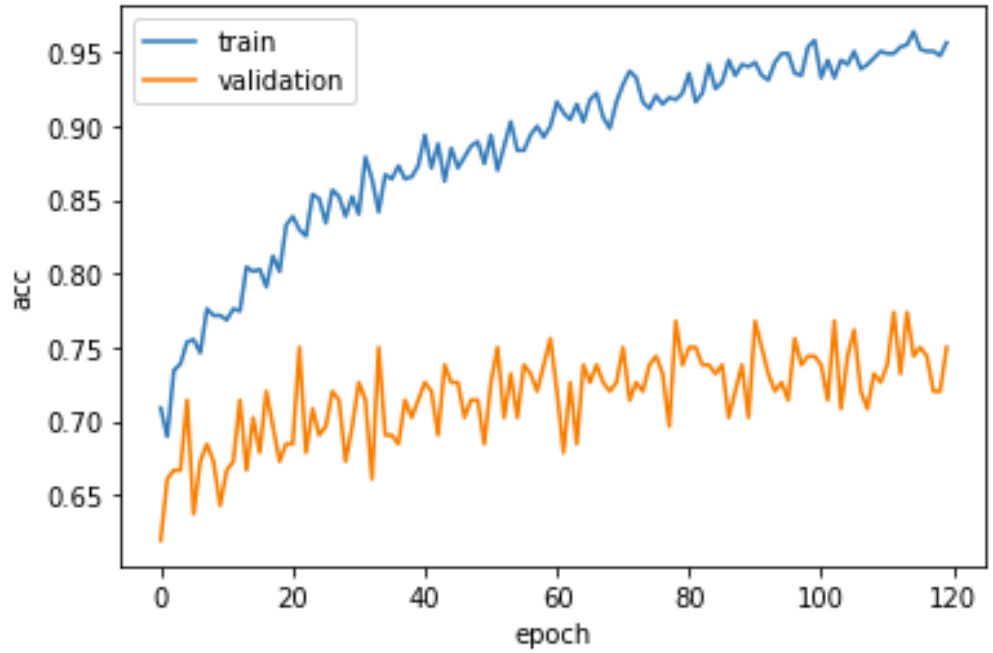


Figure 14Accuracy vs Epochs for first implementation

Table 1 Accuracy for recognizing different six expressions.

Total	Happy	Surprised	Neutral	Boredom	Confusion	Enlightened
0.72	**0.86**	**0.81**	**0.78**	**0.67**	**0.69**	**0.56**

The overall result of the first implementation is poor. The trained model is not an acceptable one for the design.

First of all, from the graph of "loss/accuracy vs. epochs," it is evident that the model is overfitting on the training data. The validation loss failed to decrease and converged to a small value, while training loss converged to zero. The validation accuracy also failed to decrease and reach a high value while the training accuracy is around 95%. Overfitting means that although the model has a good performance in classifying the training data, it has trouble classifying the data that is never seen before. Therefore, the model failed to be generalized for all kinds of data.
Secondly, from the accuracy table, it can be found that the overall accuracy is not high enough, not to mention the accuracy for Boredom, Confusion, and Enlightened is relatively low. The results of other related works that I mentioned in the "Literature Review" section show that the average accuracy on recognizing facial expressions is 80%, and expressions like "Happy" and "Surprised" have higher accuracy. Look back at my result, accuracy for "Happy" and "Surprised" are acceptable since they have accuracy above 80% and are the top two among all the expressions. The accuracy for "Neutral" is nearly 80%, so there is still room for improvement. The model's performance on recognizing Boredom and Confusion is relatively poor because the accuracy is lower than 70%. There are very few results from other research to compare, so it is hard to commend the performance across related works. However, compared with the accuracy of "Neutral," the accuracy for these two expressions should be higher. The performance of the "Enlightened" is the worst. It only has an accuracy of 56%, which is unacceptable.

There are several reasons why the performance of the model is flawed. The biggest problem is that the numbers of data for training the model are insufficient. There are only around 350 training data, around 60 images for each expression. Although the training dataset is enlarged using data augmentation provided by library Keras, the data is still far too less to train such a complicated CNN model. The performance of the expression "Enlightened" is the worst because the training data about "Enlightened" is collected and labeled only based on my own opinion. Since there is no dataset containing images about

the expression “Enlightened” from any accessible related research, all the data is collected by myself. Some of the data is collected from different image websites, and others are collected from social media. The images are found by searching keywords like “Enlightened face,” “Enlightened expressions,” etc. The only criteria to pick suitable images for the dataset is my own opinion on whether the facial expression is about “Enlightened” or not. Naturally, I can make mistakes in recognizing the facial expressions myself; for example, I sometimes can not correctly distinguish Enlightened from Surprised. The mistakes caused by human bias make the data unconvincing, and the dataset is not reasonably valid for research purposes..

During the research, I failed to find more proper images about “Enlightened.” To build a CNN model based on a convincing and valid dataset, I decided to exclude the expression “Enlightened” from my thesis goal and focus on classifying the other five facial expressions, which is my second implementation.

In the future, if I have a chance to get enough data about “Enlightened” by asking enough students to capture their faces voluntarily, I will redo the experiment with this expression.

4.2 Results and discussion about second implementation

The second implementation uses the CNN model to recognize the five facial expressions, excluding "Enlightened." (Boredom, Confusion, Neutral, Happy, and Surprised.) The CNN model is similar to the one used in the first implementation. Again, it is a modified model from pre-trained VGG16 but has an output layer with five output neural this time. The dataset now used for this implementation contains 1000 images in total: 200 images for each emotion. The data is still divided into a training/validation dataset with a ratio of 4:1 and is enlarged with the keras's data augmentation function. The model is trained with cross-entropy loss, SGD optimizer, a learning rate of 0.001, and the model's performance is evaluated using three-fold cross-validation.

Figures 15 and 16 are the plots for the training/validation loss against the number of epochs and the training/validation accuracy against the number of epochs for the model performed on one of the sets. From the loss vs. epochs plot, the validation loss stopped decreasing after around 50 epochs and finally floated around a loss of 1.0 while the training loss kept decreasing. In the accuracy vs epochs plot, the validation accuracy increased along with the training accuracy and finally floated around 75%. Table 2 shows the model's average accuracy in predicting the facial expressions of all three sets of data. The specific accuracy for each set of data is shown in Appendix C. The model has accuracies around 85% for Happy, 80% for Surprised and Neutral. Accuracies for Boredom and Confusion are still lower: they are below 70%, just like the first implementation. The model's total accuracy is 75% because the worst accuracy of Enlightened is excluded and can no longer drag down the average.

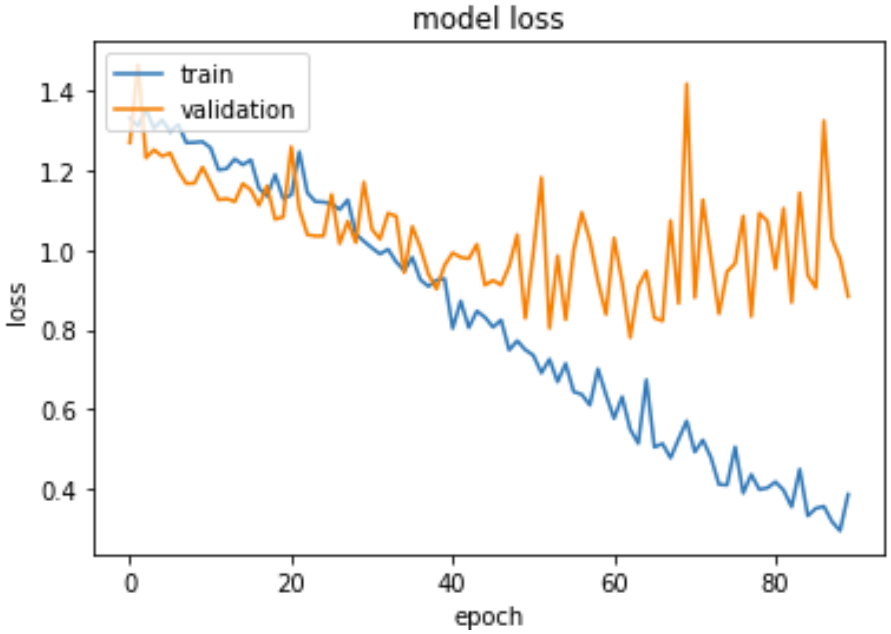


Figure 15 Loss vs Epochs plot for second implementation

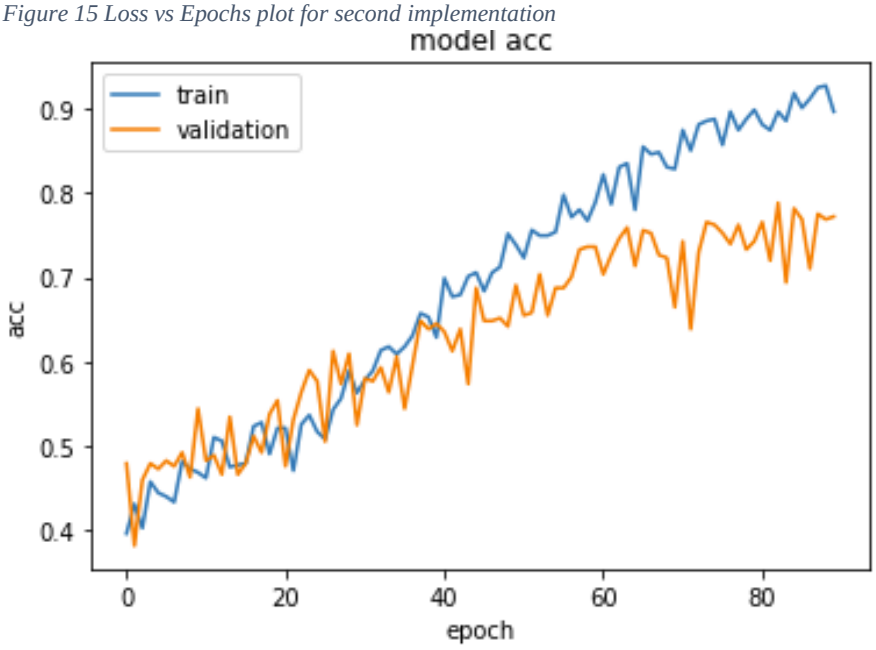


Figure 16 Accuracy vs Epochs plot for second implementation

Table 2 Accuracy of model in the second implementation

Total	Happy	Surprised	Neutral	Boredom	Confusion
0.75	0.85	0.81	0.80	0.65	0.63

The overall result for this implementation is better than the last one, but it is still not very acceptable in my view

From the graph of the “loss/accuracy vs. epochs,” it is clear that the model is still overfitting on training data. The validation loss finally converged to around 0.9, and the validation accuracy finally converges to around 75%. The results from the two plots show the same problem as stated in the first implementation. The model that tends to fit on training data is not good at generalizing for unseen data. From the table of accuracy, the accuracies for "Happy," "Surprised," and "Neutral" are still around 80%, which is similar to the last result. Whereas accuracy for "Boredom" and "Confusion" decreased slightly compared to the first implementation. The model's total accuracy is increased because the low accuracy category "Enlightened" is not involved this time.

It is out of my expectation that the model did not improve a lot after getting rid of “Enlightened” and training with more data. The performance on recognizing “Happy,” “Surprised,” and “Neutral” is acceptable when compared to the result of other related research. However, the model had a poor performance on recognizing “Boredom” and “Confusion.” To find out why, I check every single prediction of the validation data for these two expressions. Surprisingly, I found out that the images of these two expressions are often classified to be the other one incorrectly. To clearly show the behavior of the model, I plotted a confusion matrix for further study. The confusion matrix is shown below (Figure 17). The matrix shows the matching pairs between the ground truth/label of the data and the predictions by the model. If there are more matching pairs between the ground truth and the predictions, the color of the corresponding box is lighter. If there are fewer pairs, the box will have a darker color. A perfect model which can make every correct prediction will have a confusion matrix where the diagonal is white and other boxes are black. From the matrix in figure 17, it can be found that the color for boxes “Happy-Happy,” “Surprised-Surprised,” and “Neutral-Neutral” are lighter than the other boxes in the same rows and columns. This means that most of the data of these three expressions are correctly predicted. When we look at the top left and bottom left box in the “Confusion” label column, we can see that the colors of these two boxes are lighter

than the other three boxes, and their color is very similar to each other. This means that most of the “Confusion” images are predicted to be either “Confusion” or “Boredom.” A similar pattern also shows in the Boredom label column, where the colors for “Boredom-Confusion” and “Boredom-Boredom” boxes are similar to each other. The confusion matrix proved my previous finding: the model has trouble classifying the “Boredom” and “Confusion.”

In order to improve the performance on recognizing “Boredom” and “Confusion.”, I planned first to combine the two expressions with being one category and then using a new model to classify them further. The new model only focuses on learning how to classify between the “Boredom” and “Confusion.” The new implementation is expected to have a better performance.

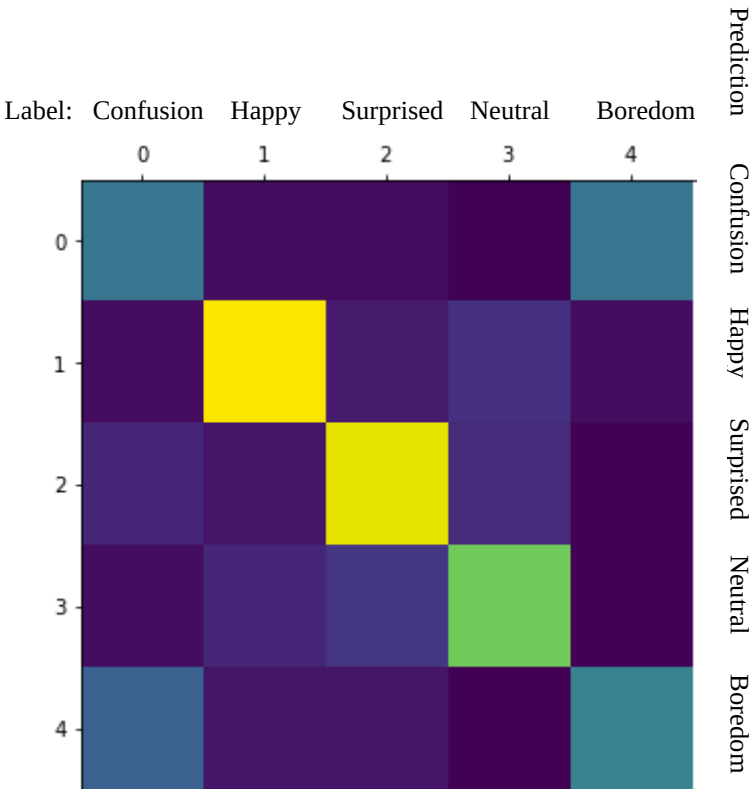


Figure 17 Confusion Matrix

4.3 Results and discussion about third implementation (Combination model)

The third implementation combines two CNN models to recognize the five facial expressions (Boredom, Confusion, Neutral, Happy, and Surprised.). Both two CNN models are similar to those used in the previous implementations but with different output layers. The first model regards “Boredom” and “Confusion” to be one category. Thus it contains an output layer with four neural. The dataset containing 800 images in total: 200 images for each category is used to train the model. If the output of the first model is the “B+C” category for an image, the image will go through the second model to be further classified as either “Boredom” or “Confusion.” (Figure 18.) The second model is trained to classify these two expressions, so it has a binary output layer. The dataset to train this model contains 200 images for each expression. The datasets are still divided into training/validation dataset with a ratio of 4:1 and are enlarged using Keras's default data augmentation method. The models are trained with cross-entropy loss, SGD optimizer, a learning rate of 0.001. The results of the two models will be discussed separately, and the performance for the combination of the models will be evaluated as well.

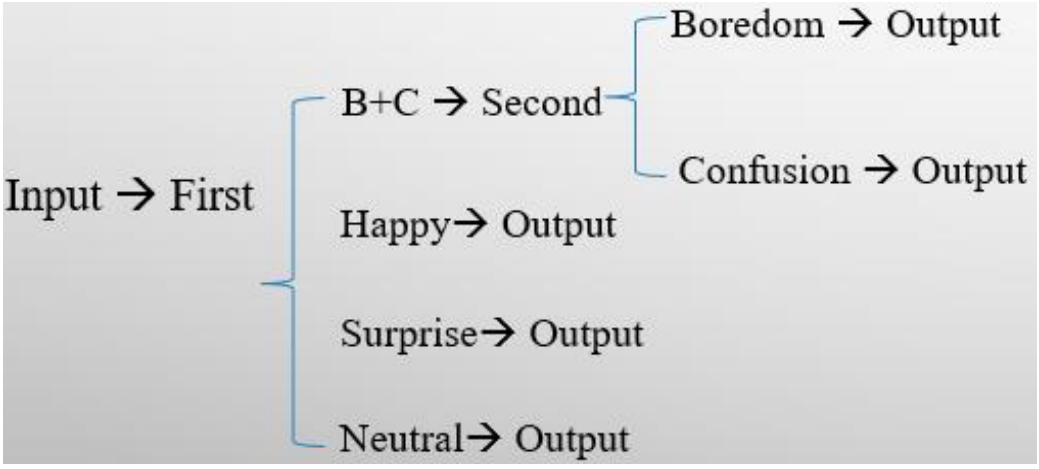


Figure 18 illustration of third implementation

As for the results of the first model: Figures 19 and 20 are the plots for the training/validation loss against the number of epochs and the training/validation accuracy against the number of epochs for the model performed on one of the cross-validation sets. From the plot of loss vs. epochs, we can see that the validation loss converged to a value of around 0.6 after 50 epochs while the training loss kept decreasing towards zero. From the plot of accuracy vs. epochs, we can find that the validation accuracy increased along with the training accuracy and finally converged to a value slightly above 80%. Table 3 shows the accuracy of the four categories. The specific accuracy for the complete sets of data is shown in Appendix C. The model has accuracies around 80% for Happy, Surprised, and Neutral, which is similar to previous results. But the accuracy for the new combined category "B+C" reaches 91%, which is much higher than the accuracy when "Boredom" and "Confusion" are recognized separately. The total accuracy is around 82%.

From the result, we can conclude that the model still has an acceptable performance on recognizing "Happy," "Surprised," and "Neutral," and this time, it can clearly classify expressions "Boredom" and "Confusion" with the other three expressions. If my second binary classification model can also have a good performance, the accuracy for recognizing these two will be higher than before.

Table 3 Accuracy for first model of combination.

Total	**Happy**	**Surprised**	**Neutral**	**B+C**
0.82	**0.83**	**0.79**	**0.81**	**0.91**

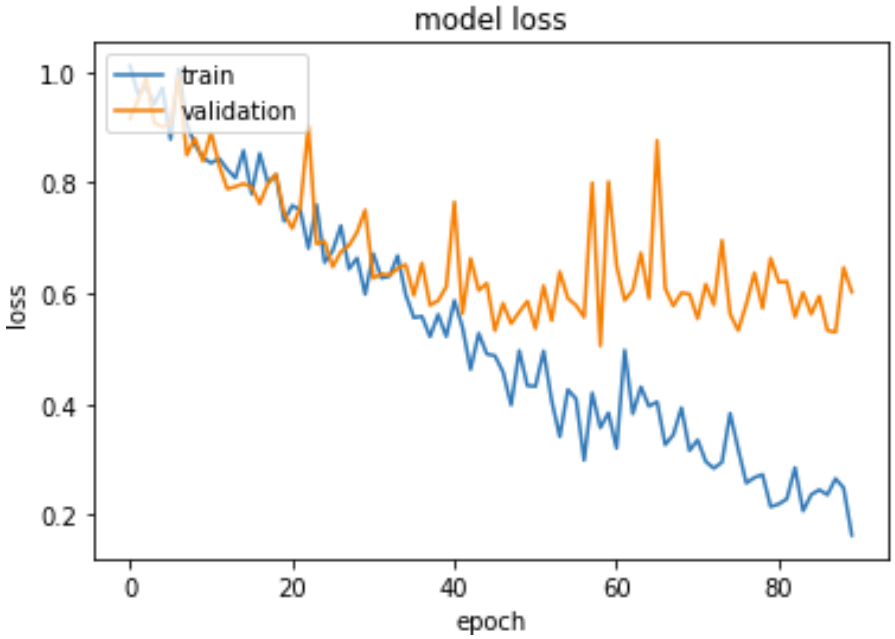


Figure 19 Loss vs Epochs plot for third implementation-first model

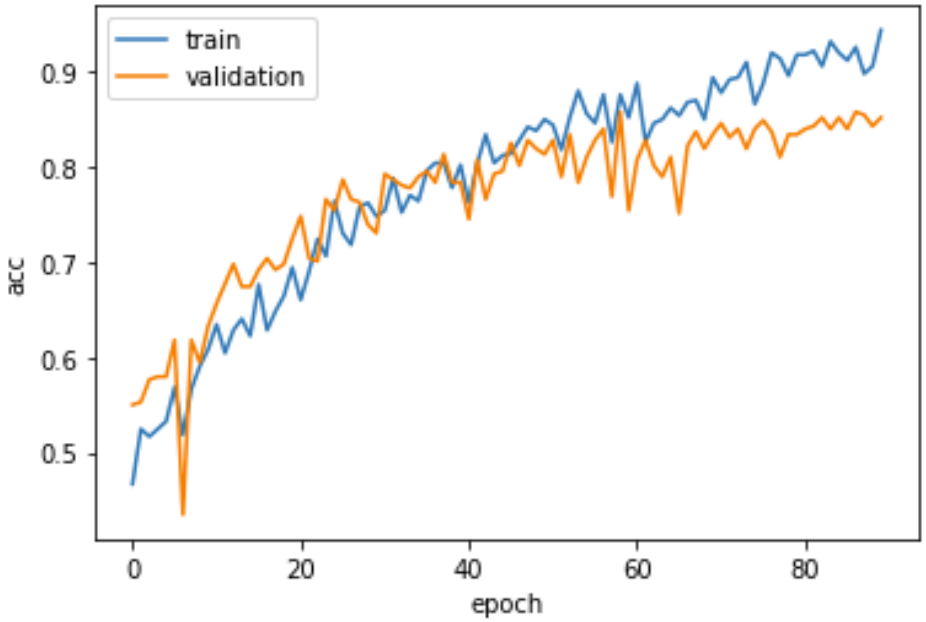


Figure 20 Accuracy vs Epochs plot for third implementation- first model

The results for the second model are shown below. Figures 21 and 22 are the plots of training/validation loss against the number of epochs and the training/validation accuracy against the number of epochs. In the plot of "loss vs. epochs," the validation loss decreased until around 60 epochs and started to be unstable while the training loss keeps decreasing. The validation accuracy increased as the epochs increased and floated around a value of 73% in the end. The training accuracy increased to a point around 80%. When compared to the previous results, the plots are more unstable during the training. Table 4 shows that both two accuracies are above 70%.

The second model of the combination did not have an expected performance since I thought the model would have a better result when the number of classification categories is less. The plots' patterns infer that the model is overfitting and the training process is very unstable because even the training loss and training accuracy plots jump up and down in the process. I think the main reason for the performance is that there are too little data for training in total. Although there are still 200 images for each expression, the same as the previous implementation, the total used data is 400, which is much less when compared to the last one. Another reason is that a CNN model could be too complex for a binary classification task. From the graphs, the model starts to overfit and become unstable after epoch 60 while the accuracy reaches the highest value of 75% at epoch 60. So I pick the model as this point (or pick the parameters that are trained after 60 epochs) for the combination.

The combination model contains the first model after training for 100 epochs and the second model after training for 60 epochs. To evaluate the performance of the third implementation, I used the combination model to predict the validation dataset and check the accuracy. The data first went through the first model and then recognized as one of the four categories (Happy, Surprised, Neutral, or B+C). If the data is one of the first three expressions, it will be output and compared to the ground truth. If the data is B+C, it was further classified by the second model and was compared to the ground truth. The accuracy is showed in table 5.

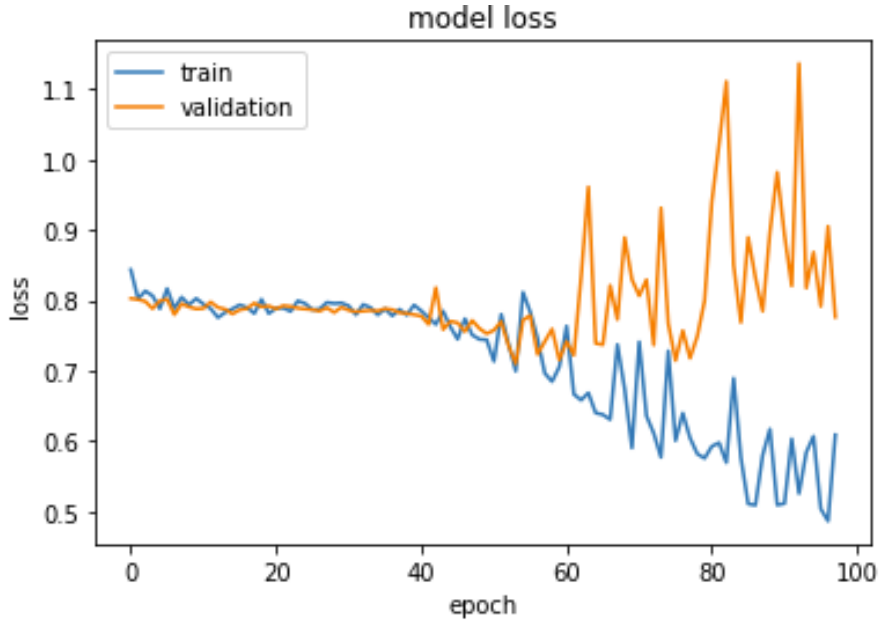


Figure 21 Loss vs Epochs plot for third implementation- second model

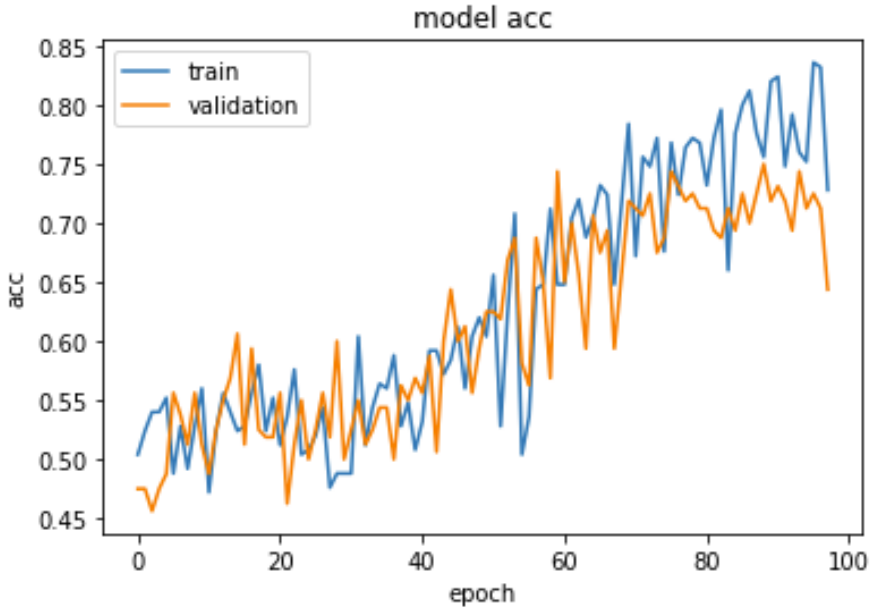


Figure 22Accuracy vs Epochs plot for third implementation- second model

Table 4 Accuracy for second model in combination

Total	**Boredom**	**confusion**
0.75	**0.76**	**0.74**

Table 5 Accuracy for recognizing using the combination model

Total	Happy	Surprised	Neutral	Boredom	Confusion
0.77	**0.83**	**0.79**	**0.81**	**0.71**	**0.69**

Recap: Table 2 Accuracy of model in the second implementation

Total	Happy	Surprised	Neutral	Boredom	Confusion
0.75	**0.85**	**0.81**	**0.80**	**0.65**	**0.63**

The accuracy for Happy Surprised, and Neutral is the same as the one in Table 3 (Accuracy for the first model of the combination) because I am using the same validation sets. When compared to the result of the model in the second implementation (Table 2), the accuracy for Boredom increases from 65% to 71%, and the accuracy for Confusion increases from 64% to 69%. The total accuracy, therefore, increased from 75% to 77%. Although the third implementation indeed improved from the second one, it still did not reach what I have expected: an average accuracy of 80%.

Overall, the three implementations had the same problem: the model is overfitting more or less. The biggest reason is that the data for training is insufficient. I only used a thousand images while the VGG16 is trained with millions of data. Even though more than half of the parameters are not trainable because of the transfer learning mechanism, the data is still not enough to help the model fight against overfitting. Another reason is that the quality of data is not high enough. A portion of the data is collected and labeled only based on my own opinion. What's more, the vagueness of some facial expressions can confuse the model. In real life, human sometimes will misrecognize the facial expressions of others, due to the fact that some facial expressions are too similar and the fact that one expression can represent different emotions, there will be unlikely to have a

perfect machine to recognize each facial expression at a rate of 100%. For me, the designs and the implementations are acceptable because of several limitations during the research. Finally, the combination model is implemented into the final functional program.

4.4 Results/presentations of the functional program

The functional program uses the methods and the deep learning models I introduced earlier to capture student's faces, recognize their facial expressions, and send the information to the lecture's computer. Figure 23 shows how the program captures the faces and recognizes the expressions. Figure 24 and Figure 25 show how the program sends the information from one computer to another. In figure 24, the terminal represents a student's computer screen; the Publisher of the message protocol kept sending the information out. In Figure 25, the new terminal represents a lecturer's computer screen; the Subscriber of the message protocol kept receiving information of facial expressions at the same time.

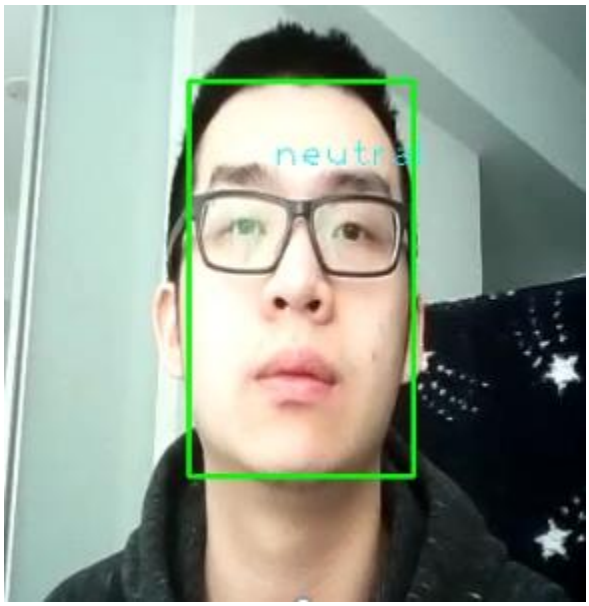


Figure 23 Capture and Recognize

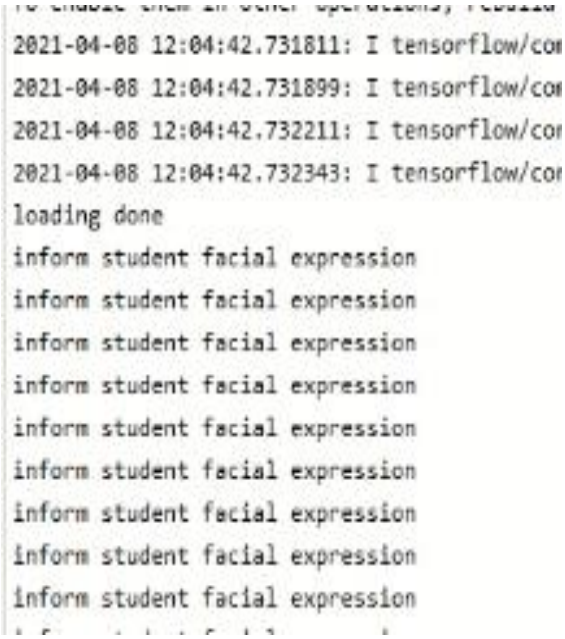


Figure 24 Student screen.

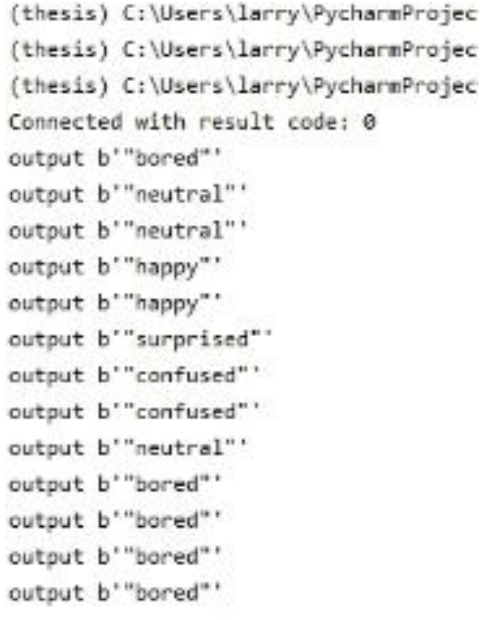


Figure 25 Lecturer's screen

4.5 Overall program performance

The final program design and implementation achieve the goal of the thesis; it can capture the student's face accurately, recognize the expressions that are popular in a lecture, and inform the expressions to the lecturer in real-time. The performance of the program is not perfect, but I am pleased with it. It achieves all the expected functions, and the accuracy of the recognizing part is lower than the results from other research but is acceptable due to limitations.

5 Conclusion

The big goal of the thesis is to improve the interactions between lecturers and students during the online lecture. The program designed and built in the thesis achieves the goal by informing the students' facial expressions to the lecturer. The thesis is innovative in the area of recognizing human facial expressions because it includes some expressions that are seldomly researched by others. The dataset, the deep learning model, and the results can be used for other research teams. The final program is not perfect due to several limitations. One big limitation that prevents me from building a more accurate program is the insufficient dataset. Therefore, one of the future steps is to collect enough and unbiased data. One method can be holding an experiment in the university and asking students to take pictures for the required facial expressions. Another future work can be testing the message transfer performance between two computers. The test can include latency test, message loss rate test, etc. The last possible future step is to create a better display of the facial expression information on lecturers' screens. One idea is to use emoji to represent the expressions for each student on the small windows. A conceptual example is shown in Figure 26. The emoji is consistent with the received facial expressions and is a straightforward way to present.

Figure 26 Illustration of emoji representation

6 References

[1] Effect of facial expressions on student's comprehension recognition in virtual educational environments. By Mohamed Sathik and Sofia G Jonathan· Online[https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3795200/]
[2] S. Gory, M. Al-Khassaweneh and P. Szczurek, "Machine Learning Approach for Facial Expression Recognition," 2020 IEEE International Conference on Electro Information Technology (EIT), Chicago, IL, USA, 2020, pp. 032-039, doi: 10.1109/EIT48999.2020.9208316.

[3] Challenges in Representation Learning: Facial Expression Recognition Challenge, Kaggle.com. 2013. [online] Available: https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge

[4] Ji, Y. Z. a. Q., May 2005. Active and Dynamic Information Fusion for Facial Expression Understanding from Image Sequences. IEEE, 27(5), p. 699–714.
[5] Dr. Shaik Asif Hussain and Ahlam Salim Abdallah Al Balushi 2020 J. Phys.: Conf. Ser. 1432 012087
[6] KDEF | The Kenya Dryland Education Fund (2019) available from [2 September 2019]
[7] K. Liu, M. Zhang and Z. Pan, "Facial Expression Recognition with CNN Ensemble," 2016 International Conference on Cyberworlds (CW), Chongqing, 2016, pp. 163-166, doi: 10.1109/CW.2016.34.
[8] A. Fathallah, L. Abdi and A. Douik, "Facial Expression Recognition via Deep Learning," 2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA), Hammamet, 2017, pp. 745-750, doi: 10.1109/AICCSA.2017.124.

[9] M. Heidari and K. Fouladi-Ghaleh, "Using Siamese Networks with Transfer Learning for Face Recognition on Small-Samples Datasets," 2020 International Conference on

Machine Vision and Image Processing (MVIP), Iran, 2020, pp. 1-4, doi: 10.1109/MVIP49855.2020.9116915.
[10] Labeled Faces in the Wild, [online] http://vis-www.cs.umass.edu/lfw/
[11] ImageNet [http://www.image-net.org/]
[12] Caroppo, A., Leone, A. & Siciliano, P. Comparison Between Deep Learning Models and Traditional Machine Learning Approaches for Facial Expression Recognition in Ageing Adults. J. Comput. Sci. Technol. **35,** 1127–1146 (2020). https://doi.org/10.1007/s11390-020-9665-4
[13] S. Zhalehpour, O.Onder, Z.Akhtar and C.E.Erdem, "BAUM-1: A Spontaneous Audio-Visual Face Database of Affective and Mental States", *IEEE Trans. on Affective Computing*, , Database web site: http://baum1.bahcesehir.edu.tr **DOI: 10.1109/TAFFC.2016.2553038**
[14] keras: the python deep learning API, online[https://keras.io/]

7 Appendices

Appendix A

1.Result of Dr. Hussain & Dr. Balushi

Table 1. shows the validation accuracy with the dataset and VGG 16 CNN model used in testing and classification.

Model	Dataset	Accuracy/F1-score
VGG-16 Face	KDEF	0.88

2.Result of Liu et al

TABLE V: FER-2013 Leaderboard

Rank	Team	Acc. (%)
1	RBM	69.77
2	Unsupervised Team	69.07
3	Maxim Milakov	68.15
4	Radu+Marius+Cristi	67.48
-	**Subnet Ensemble**	**65.03**
5	Lor.Voldy	64.56
:		
8	Xavier Bouthillier	62.78
-	**Subnet3**	**62.44**
9	sayit	61.91
10	Alejandro Dubrovsky	61.38
:		
56	dstarerstor	20.40

3. Result of Fathallah et al.

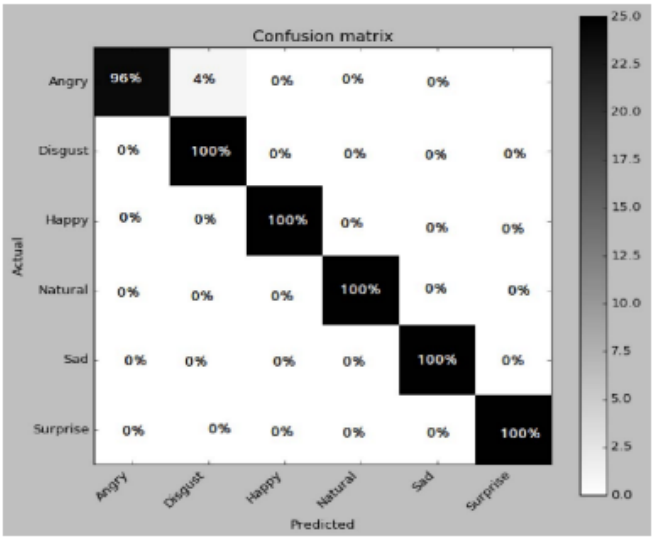


Fig. 3. Confusion matrix of proposed method on CK+ database.

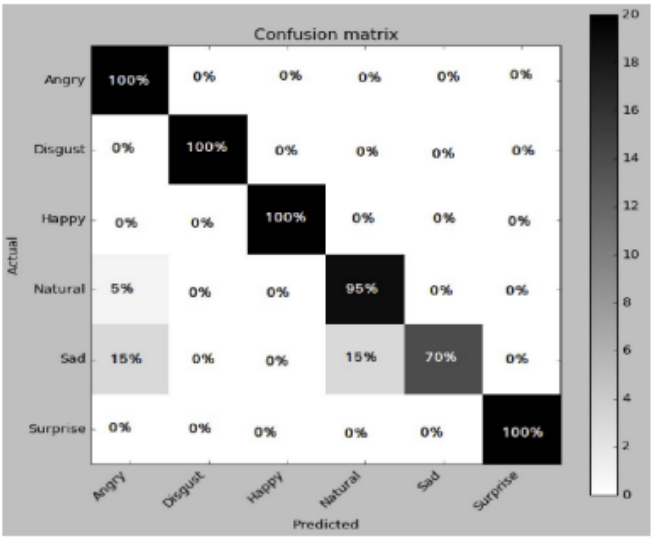


Fig. 5. Confusion matrix of proposed method on RaFD database.

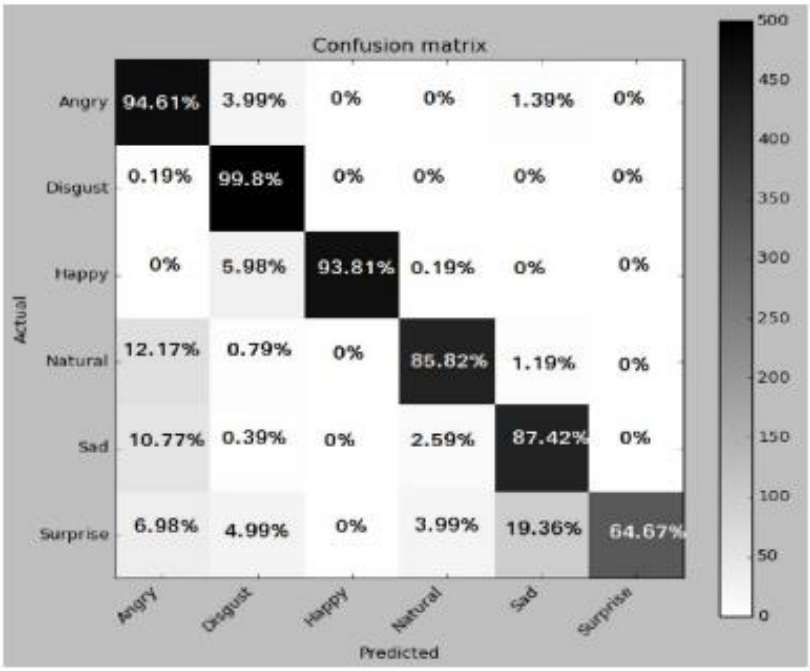


Fig. 4. Confusion matrix of proposed method on MUG database.

4.Result of Heidari & Fouladi-Ghaleh

TABLE I
FACE RECOGNITION RATE IN DIFFERENT METHODS WITH LFW DATASET

Method	Face Recognition Rate (%)
DLB [10]	88.50
CFN+APEM [9]	87.50 $\pm$ 1.57
L-CSSE+KSRC [8]	92.02
SiameseFace1 [11]	94.80
Weighted Pca-Efmnet [7]	95.00 $\pm$ 0.71
Siamese-VGG (Ours)	**95.62 $\pm$ 0.42**
CosFace [15]	99.73

5.Result of Caroppo et al.

Table 8. FER Accuracy on Dataset Lifespan

Age Group	VGG-16+ RF (%)	ASM+ SVM (%)	LBP+ SVM (%)
Young (18–29 years)	99.34	90.16	90.54
Middle-aged (30–49 years)	98.63	89.24	90.01
Old (50–69 years)	97.44	86.12	86.32
Very old (70–93 years)	96.91	85.28	86.01
Overall accuracy	98.08	87.70	88.22

Table 9. FER Accuracy on Dataset CIFE

Age Group	VGG-16+ RF (%)	ASM+ SVM (%)	LBP+ SVM (%)
Young (< 35 years)	86.13	68.05	71.98
Middle-aged (35–55 years)	84.26	66.20	65.97
Old (56–68 years)	83.09	62.64	61.18
Very old (> 68 years)	82.52	58.23	57.35
Overall accuracy	84.00	63.78	64.12

Table 10. FER Accuracy on Dataset FER2013

Age Group	VGG-16+ RF (%)	ASM+ SVM (%)	LBP+ SVM (%)
Young (< 35 years)	75.33	60.68	63.11
Middle-aged (35–55 years)	71.98	57.87	59.51
Old (56–68 years)	70.22	55.99	56.24
Very old (> 68 years)	68.47	51.22	53.76
Overall accuracy	71.50	56.44	58.08

Appendix B

1. Code about Open CV

```
face, confidence = cv.detect_face(img)
#print(face)
num=1
for idx, f in enumerate(face):
    #get corner points of face rectangle
    (startX, startY) = f[0], f[1]
    (endX, endY) = f[2], f[3]
    face_crop = img[startY: endY, startX:endX]
```

2. Code about EMQX Subscriber

```
import paho.mqtt.client as mqtt


def on_connect(client, userdata, flags, rc):
    print("Connected with result code: " + str(rc))


def on_message(client, userdata, msg):
    print(msg.topic + " " + str(msg.payload))


client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message
# client.on_disconnect = on_disconnect
client.connect('127.0.0.1', 1883, 60000) #
client.subscribe('output', qos=0)
client.loop_forever() #
```

3. Code about EMQX Publisher

```
import paho.mqtt.client as mqtt

import json
from time import sleep
import demo_transfer

def on_connect(client, userdata, flags, rc):
    print("Connected with result code: " + str(rc))


def on_message(client, userdata, msg):
    print(msg.topic + " " + str(msg.payload))


def main():
    client = mqtt.Client()
    client.on_connect = on_connect
    client.on_message = on_message
    client.connect('127.0.0.1', 1883, 60000)

    while True:
        #data_pacakge = model.detect()
        data_pacakge =demo_transfer.detect()#use model here
        json_data = json.dumps(data_pacakge)
        client.publish('output', payload=json_data, qos=0)
        sleep(1)  # the interval of data publishing is 500 ms as required by


if __name__ == '__main__':
    main()
```

Appendix C

1. Three sets of validation accuracy for first implementation

Set	Total	Happy	Surprised	Neutral	Boredom	Confusion	Enlightened
1	0.73	0.85	0.79	0.78	0.65	0.69	0.53
2	0.70	0.89	0.80	0.77	0.69	0.70	0.60
3	0.74	0.83	0.83	0.78	0.66	0.66	0.55

2. Three sets of validation accuracy for second implementation

Set	Total	Happy	Surprised	Neutral	Boredom	Confusion
1	0.77	0.88	0.80	0.78	0.65	0.60
2	0.76	0.83	0.79	0.81	0.65	0.64
3	0.74	0.85	0.83	0.80	0.66	0.64

3. Three sets of validation accuracy for third implementation—fist model

Set	Total	Happy	Surprised	Neutral	B+C
1	0.83	0.81	0.79	0.80	0.93
2	0.83	0.84	0.80	0.82	0.90
3	0.81	0.83	0.79	0.79	0.90

4. Three sets of validation accuracy for third implementation—second model

Set	Total	Boredom	Confused
1	0.76	0.77	0.75
2	0.76	0.76	0.77
3	0.73	0.72	0.74

Appendix D Important Coding

1. Use pre-trained VGG16 model

```
VGGmodel = VGG16(include_top=False, input_shape=(160, 160, 3))
for layer in VGGmodel.layers:
  layer.trainable = False
  #layer.trainable = True
VGGmodel.summary()
```

```
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_5 (InputLayer)         [(None, 160, 160, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 160, 160, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 160, 160, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 80, 80, 64)        0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 80, 80, 128)       73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 80, 80, 128)       147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 40, 40, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 40, 40, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 40, 40, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 40, 40, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 20, 20, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 20, 20, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 20, 20, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 20, 20, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 10, 10, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 10, 10, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 10, 10, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 10, 10, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 5, 5, 512)         0
=================================================================
Total params: 14,714,688
Trainable params: 0
```

2. Build own model using transfer learning

```
VGGmodel = VGG16(include_top=False, input_shape=(160, 160, 3))
for layer in VGGmodel.layers[:-12]:
  layer.trainable = False
  #layer.trainable = True
VGGmodel.summary()
```

```
dropout1 = Dropout(0.5)
dropout2 = Dropout(0.5)
dropout3=Dropout(0.5)
flat1 = Flatten()(VGGmodel.layers[-1].output)
x = Dense(512, kernel_regularizer=l2(1e-4), activation='relu')(flat1)
x = Dense(512, activation='relu')(x)
x = dropout1(x)
x = Dense(512, activation='relu')(x)
x = dropout2(x)
x= Dense(6, activation='softmax')(x)

model_1 = Model(inputs=VGGmodel.inputs, outputs=x)
model_1.summary()
```

3. Train the model and plot results

```
from keras.preprocessing.image import ImageDataGenerator
train_datagen=ImageDataGenerator(shear_range=0.15,zoom_range=0.15,rotation_range=20,horizontal_flip=True)
#train_datagen.fit(x_train)
train_datagen.fit(x2_train)
#train_datagen.fit(x_train_grey)
```

```
opt = keras.optimizers.SGD(learning_rate=0.001)
```

```
model_1.compile(optimizer=opt, loss='categorical_crossentropy',metrics=['CategoricalAccuracy'])
```

```
history=model_1.fit_generator(train_datagen.flow(x_train, y_train) , epochs=100, verbose=1, validation_data=(x_test, y_test))
```

```
plt.plot(history.history['loss'][2:])
plt.plot(history.history['val_loss'][2:])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```

```
plt.plot(history.history['categorical_accuracy'][2:])
plt.plot(history.history['val_categorical_accuracy'][2:])
plt.title('model acc')
plt.ylabel('acc')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```

4. Face Cropping use OpenCv

```
import cv2
import numpy as np
import cvlib as cv
import numpy as np
from PIL import Image
import os
from matplotlib import pyplot as plt
file_path="raw_data"
file_dirs = os.listdir(file_path)
#test=None
for folders in file_dirs:
    folder_path="raw_data/"+folders
    folder_dirs = os.listdir(folder_path)
    print(folders)
    for pictures in folder_dirs:
        print(pictures)
        img=Image.open(folder_path+'/'+pictures)
        img= np.array(img)
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        face, confidence = cv.detect_face(img)

        num=1
        for idx, f in enumerate(face):
            #get corner points of face rectangle
            (startX, startY) = f[0], f[1]
            (endX, endY) = f[2], f[3]
            face_crop = img[startY: endY, startX:endX]
            output_string = pictures[:-4]
            cv2.imwrite(folder_path + '/' + '_' + output_string + '_' + str(num)+ 'face.jpg', face_crop)
            num+=1
```

5. Code for the functional program

```
import cv2
import numpy as np
from keras.models import load_model
import cvlib as cv
import h5py
import random
from skimage.color import rgb2gray
print('loading model')
import time
model1 = load_model("model_3.h5")
model2=load_model("model_2.h5")

print('loading done')
camera = cv2.VideoCapture(0)
emotions = ['B+C', 'happy', 'surprised','neutral']
B_C_emotions=['bored','confused']


num_frame=0
while True:

    ret, frame = camera.read()
    num_frame+=1
    if num_frame%5==0:
        if frame is not None:
            outputframe = frame
            output = np.asarray(frame, dtype=np.uint8)
            face, confidence = cv.detect_face(frame)
            if face != []:
                for idx, f in enumerate(face):
                    # get corner points of face rectangle
                    (startX, startY) = f[0], f[1]
                    (endX, endY) = f[2], f[3]
                    width = endX - startX
                    height = endY - startY
                    cv2.rectangle(outputframe, (startX, startY), (endX, endY), (0, 255, 0), 2)
                    face_crop = np.copy(frame[startY:endY, startX:endX])
                    if face_crop!=[]:
                        input = cv2.resize(face_crop, (160, 160))
                        inputarray = []
```

```
        face_crop = np.copy(frame[startY:endY, startX:endX])
        if face_crop!=[]:
            input = cv2.resize(face_crop, (160, 160))
            inputarray = []
            arr = np.array(input)
            inputarray = np.append(inputarray, arr)
            inputarray = np.array(inputarray, dtype='float32')
            inputarray = inputarray.reshape((1, 160, 160, 3))
            inputarray_single = inputarray[:, :, :, 1]
            inputarray_single = inputarray_single.reshape(((1, 160, 160, 1)))
            inputarray_grey = np.array(np.concatenate((inputarray_single , inputarray_single , inputarray_single ), axis=3))
            pred_array1 = model1.predict(inputarray_grey)
            idx1 = np.argmax(pred_array1 [0])
            if idx1!=0:
                cv2.putText(outputframe, text=emotions[idx1], org=(startX + 50, startY + 50), fontFace=1, fontScale=1.5,
                        color=(255, 255, 0))
            else:
                pred_array2=model2.predict(inputarray_grey)
                idx2 = np.argmax(pred_array2[0])
                B_C_emotion = B_C_emotions[idx2]
                cv2.putText(outputframe, text=B_C_emotion, org=(startX + 50, startY + 50), fontFace=1, fontScale=1.5,
                        color=(255, 255, 0))
 k = cv2.waitKey(10)
 if k == 27:
     break
 cv2.imshow('test1', outputframe)
AllWindows()
```