Predicting the Attention Rate of Patients by Monitoring their Emotions

Result Document

Version 1.0

April 17th, 2021

Akshara Shukla

440692@student.fontys.nl

Semester 04 – ICT & Artificial Intelligence

Fontys University of Applied Sciences,

Eindhoven, Netherlands

Abstract

This report illustrates the methodology and the theorical background of building a CNN model with the aim of classifying emotions of patients in a hospital or in a clinic into four basic emotions (Angry, Happy, Relaxed, and Sad) from facial expressions in real life. Using the FER-2013 dataset an accuracy of 50 % by using VGG16 pretrained model and 63.28% with building our own CNN architecture was achieved. After close evaluation of our model, I formed a general camera usage from my default computer system which when recognizing negative emotions (Sad and Angry) a message was seen on the screen indicating that, that patient needs immediate attention from a specialist.

Data Student:	
Family name, initials	Shukla, Akshara
Student number	440692
Project period (from-until)	01.04.2021 – 17.04.2021
Data Company:	
Name company/institution	FER-2013, Kaggle
University tutor:	
Family name, initials	Veneman, Woody, Avetyan, Rafayel R.
Project Plan:	
Title	Facial Recognition
Date	17.05.2021
Version	**1.0**

Contents

Introduction

Areya is a CNN model which tackles the problem of recognizing the basic emotions expressed by several patients while being admitted in a hospital and/or clinic after undergoing surgery from their facial expressions. Initially, after having understood the working principles of CNN models substantially, I built my own sequential model with storing and training it on the four basic emotions illustrated by general patients: Angry, Happy, Relaxed and Sad. After having, analysed the results gained on a self-made model, the approach changed to implying a widely known pretrained model for image recognition called VGG16 to investigate if the performance of the model can be improved furtherly. Following, we used the haarcasade.xml file available from scikit-learn library for facial detection on a webcam since, Areya is going to be providing notifications on the webcam video feed.

Dataset and Features

The use of dataset FER-2013 was used which consists of 48x48 pixel grayscale images of faces. The faces have been automatically registered so that the face is centred and occupies about the same amount of space in each image. The dataset initially has 7 emotions but for this project and reducing the interval of classification, the emotions were reduced to four. The task was to categorize each face based on the emotion shown in the facial expression into one of four categories (0=Angry, 1 = Happy, 3=Relaxed, 4=Sad). The training set consists of 26,217 examples and the public test set consists of 5212 examples with four classes each.

While conducting the exploratory data analysis, some of the key insights that were found, included the dataset having the least amount of data for Angry emotion in both train and test set and the emotion with the maximum were Happy which can be understood as many people like to take photographs of themselves while smiling. Another insight that was found included, few of the images in our dataset weren't valid i.e., didn't contain any face in them or were completely blank. Furthermore, by using the Min-Max Scaler, the determination of normalized pixel values was done. By plotting the first five images of different emotions, a point that I noted was emotions such as Relaxed and Sad were somewhat similar hence, one inference which could be made here is that classifying these emotions would be difficult.

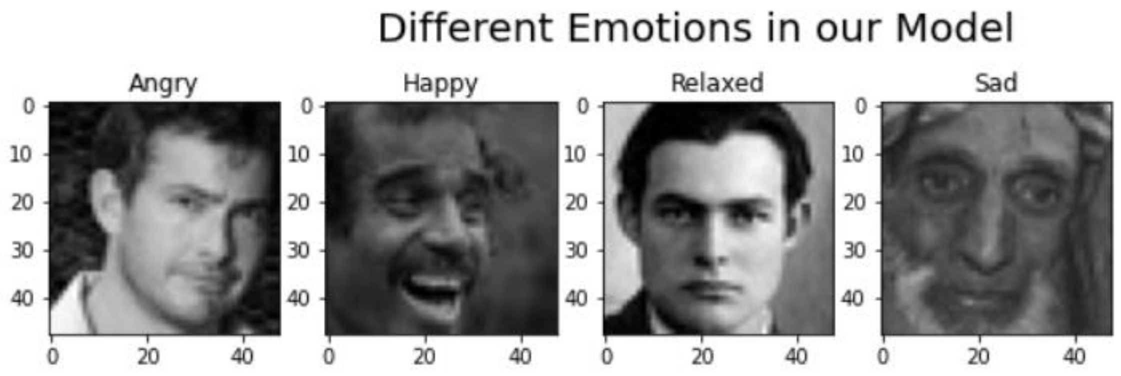


Figure 01: Four Emotions being detected by our model

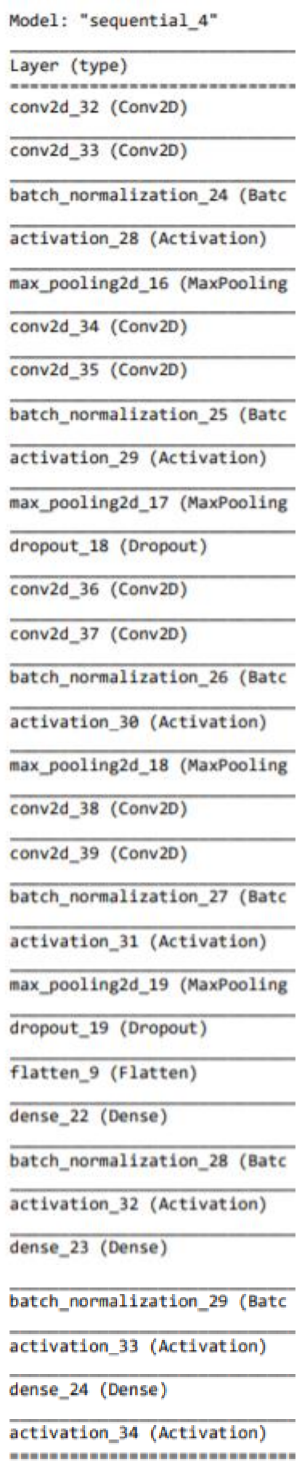
Methodology

I. CNN Model

A custom CNN model was built which essentially required a lot of trial-and-error phases since, getting the accurate value of the hyperparameter expected a lot of experimenting with different values and comprehending which of them improved the performance of our model.

Before training our model, the implementation of image augmentation was carried which included rotating, flipping, cropping, sheering, and shifting of the images already present in our dataset. This helped in generating more images for the model to learn from.

Secondly, the use of callback functions was also used which are unique functions which are being performed while the model is getting trained. From the provided list of callback functions from the official Keras website, the use of EarlyStopping, CSVLogger, ReduceLRPlateau and ModelCheckPoint were used. Since, the priority while building a CNN model is to save its scores of accuracy and loss on every epoch, the use of CSVLogger and ModelCheckPoint was added. While EarlyStopping was used to avoid any overfitting problems with reducing the learning rate if the model is being good at detecting patterns in our images with the help of ReduceLRPlateau.



```
Model: "sequential_4"
____________________________
Layer (type)
============================
conv2d_32 (Conv2D)
____________________________
conv2d_33 (Conv2D)
____________________________
batch_normalization_24 (Batc
____________________________
activation_28 (Activation)
____________________________
max_pooling2d_16 (MaxPooling
____________________________
conv2d_34 (Conv2D)
____________________________
conv2d_35 (Conv2D)
____________________________
batch_normalization_25 (Batc
____________________________
activation_29 (Activation)
____________________________
max_pooling2d_17 (MaxPooling
____________________________
dropout_18 (Dropout)
____________________________
conv2d_36 (Conv2D)
____________________________
conv2d_37 (Conv2D)
____________________________
batch_normalization_26 (Batc
____________________________
activation_30 (Activation)
____________________________
max_pooling2d_18 (MaxPooling
____________________________
conv2d_38 (Conv2D)
____________________________
conv2d_39 (Conv2D)
____________________________
batch_normalization_27 (Batc
____________________________
activation_31 (Activation)
____________________________
max_pooling2d_19 (MaxPooling
____________________________
dropout_19 (Dropout)
____________________________
flatten_9 (Flatten)
____________________________
dense_22 (Dense)
____________________________
batch_normalization_28 (Batc
____________________________
activation_32 (Activation)
____________________________
dense_23 (Dense)
____________________________
batch_normalization_29 (Batc
____________________________
activation_33 (Activation)
____________________________
dense_24 (Dense)
____________________________
activation_34 (Activation)
============================
```

Figure 02: Architeture of custom-made CNN

After investigating enough with the number of layers, dimensions and trying out different hyperparameter values, I came across an architecture which was launched by Google during the ImageNet Recognition Challenge and they implemented a pretrained model Inception V3 with replacing the 3x3 kernel size to 1x3 followed by 3x1 convolutional layer making it to a total of one full dimension convolutional layer. On some tweaking and adjusting the layers such as the dropout rate to 10%, the accuracy received was rather better than the first CNN model made.

It additionally, resulted in better division of the confusion matrix, since when comparing to the confusion matrix of first, the results were all classified with taking one emotion which held potential to be called biased. The confusion matrix for our final CNN model is mentioned in the results section of this document.

II. VGG16

VGG16 stands for Visual Geometry Group from Oxford which is a pretrained model available from the PyTorch library and is know for it's 16 layers deep architecture for classifying and training images. The reason I officially implemented this was to research and make inferences on the functionality of pretrained models. The architecture of VGG16 is mentioned below as follows:

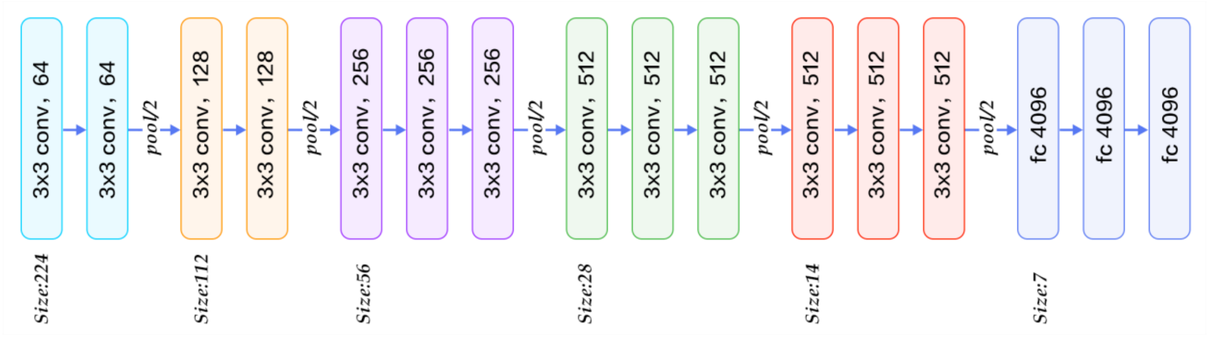


Figure 03: VGG16 Architecture

The implementation of the pretrained model VGG16 on our dataset was carried out by first describing the input shape of our images to (48,48,3) and setting the trainable layers to False because I wanted to customize the last three layers. Here, the powerful optimize Adam was used and the metrics of accuracy was calculated. The model ran on 20 epochs with the callback functions with a batch size of 819 for the training set and produced an accuracy of 50%. The magnitude for this accuracy can be due to the fact that every pretrained models are built to perform on datasets consisting of millions of data values and on large number of epochs taking weeks and months to train on but since our dataset just in thousands, it performed okay. Analyzing errors in deep learning models is infamously difficult but this can be addressed with regards to future implementations.

Results

I. Results on FER-2013 Dataset

Model	Training Accuracy	Training Loss	Testing Accuracy	Testing Loss
Custom CNN	59%	0.95%	63%	0.87%
VGG16	48.24%	0.48%	50%	0.47%

Comparing the testing accuracy of the two models investigated, we can see the second custom-made CNN model performed slightly better than the VGG16 with a total accuracy of 63%. All of the models had same magnitude of learning rate, parameters of callback functions with a batch size of 32 and trained on 20 epochs, respectively. Moreover, the VGG16 model was able to reduce the value of loss attaining the lowest from the rest. This is interesting to notice here the pretrained model didn't provide high accuracy percentage but managed to reduce the value of loss.

After achieving the accuracy, the aim involved researching how and/or what accuracy was achieved from papers applying emotion recognition with this dataset in order to compare mine. One of the papers which was used as a reference was the 'Real-time Emotion Recognition from Facial Expressions' by Stanford University. They managed to get an accuracy of 66.67% when training their own CNN model on 100 epochs, when compared the results, we can conclude that our model performed pretty good on managing to score an accuracy of 63% on just 20 epochs.

Since, the CNN model yielded better results, I decided to evaluate it and continue to understand it more.

II. Confusion Matrix

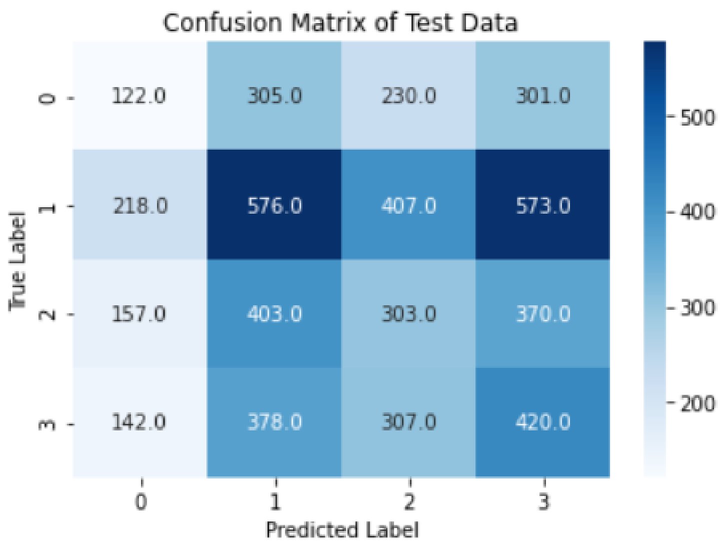


Figure 04: Confusion Matrix of Test Set

The confusion matrix plotted above with the predicted vs true labels of our emotions (0=Angry, 1 = Happy, 3=Relaxed, 4=Sad), we can determine the model classified the emotion Happy with the highest accuracy. Overall, we can see all of the emotions have been predicted with reference as Happy. Some of the emotions: Sad and Relaxed were quite similar having their true label around 300 instances which in reality can also be difficult to detect. In addition, emotion Angry was poorly detected which can be due the smaller number of instances present in the dataset initially.

III. Model Accuracy Over Training Period

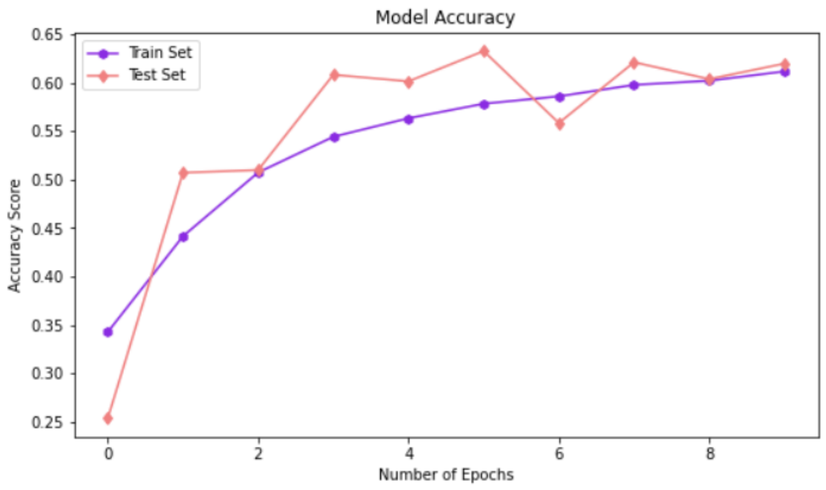


Figure 05: CNN Model Accuracy

The above plot demonstrated the accuracy score of training set and testing set for each epoch. As we can see the maximum number of epochs is 10 which means the callback function, EarlyStopping specified earlier played its role as to when the difference between the value of our validation loss (training set's loss) was less than 0 in a maximum of 4 epochs. It's worth mentioning here that the training sets experiences a gradual increase in accuracy which indicates a perfect learning rate. This means our model was able to learn images accurately. On the other hand, the training set's accuracy seemed to be close to that of training which proves the fact that our model is neither overfitted nor underfitted.

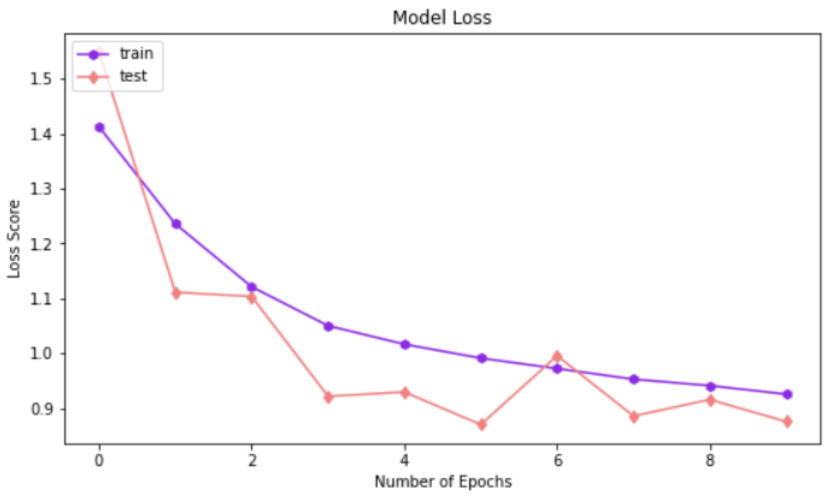


Figure 06: CNN Model Loss

The above plot illustrates the loss score of training and testing set for each epoch. The slight gradual decrease in loss for training implies a good learning rate. While for testing, the loss is reducing but in a non-linear manner. The difference between the loses isn't quite big, therefore in order to improve the performance much better, more data and training for a longer period of time would be necessary.

IV. Emotions Classified from Images

In this section, there are four plots indicating how the CNN model has understood, captured, and predicted the emotion. The image of the emotion is taken from the test set. This helps in understanding what images the model corresponds to emotions. As seen in the confusion matrix above, the model performed slightly better in interpreting emotion Happy and Relaxed which can be seen below as well while it struggled to differentiate between Sad and Angry.

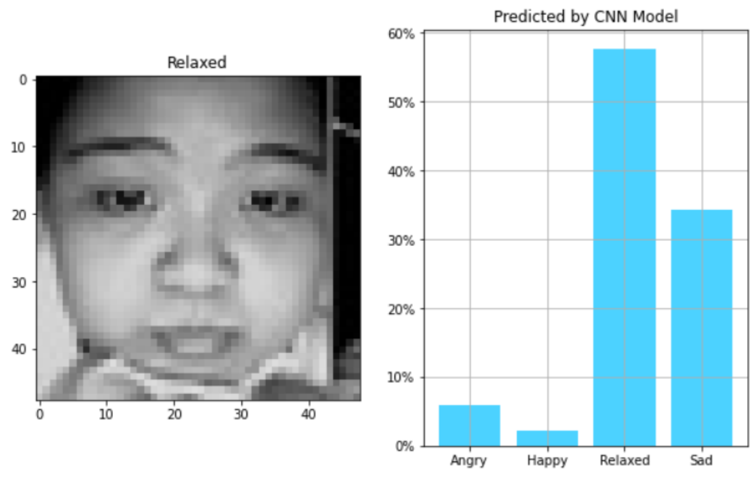


Figure 07: 'Relaxed' Emotion interpreted by the model

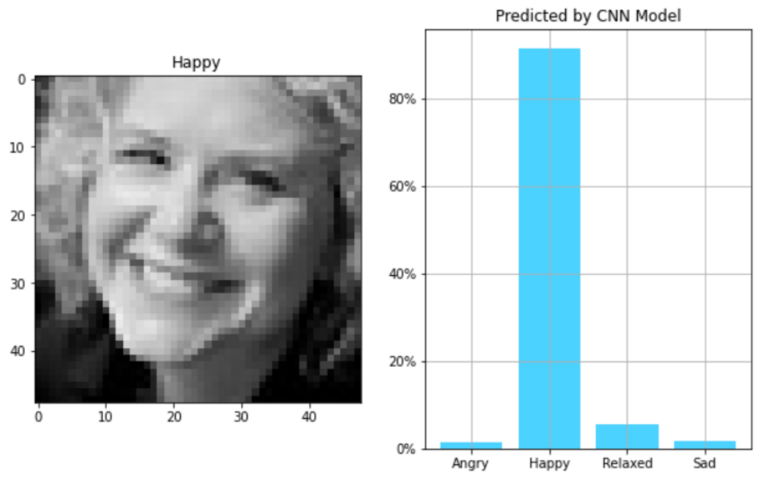


Figure 08: 'Happy' Emotion interpreted by the model

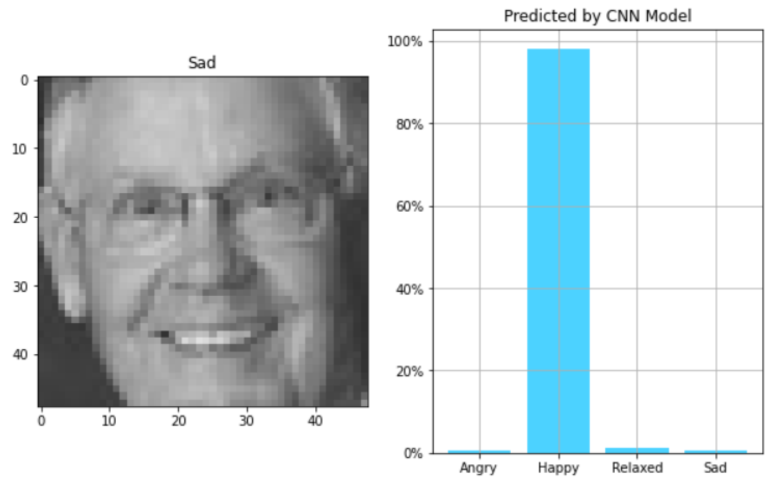


Figure 09: 'Sad' Emotion interpreted by the model

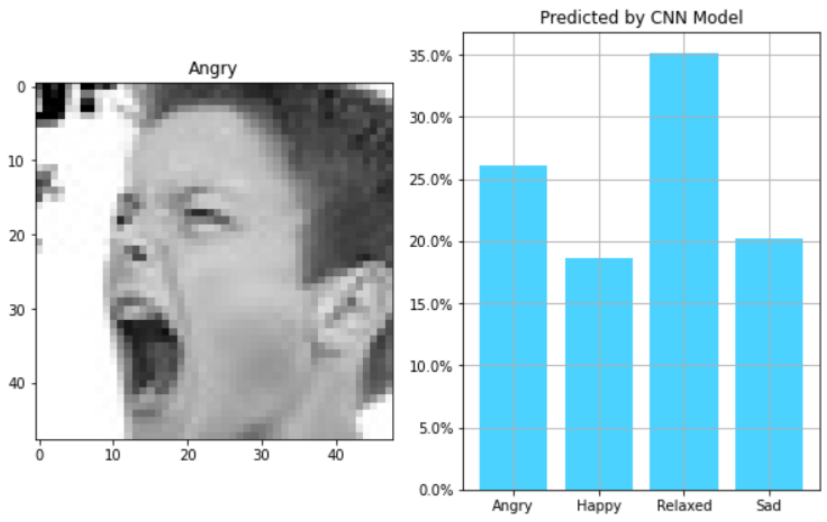


Figure 10: 'Angry' Emotion interpreted by the model

One interesting factor to notice here is that the images from the test set had emotions Sad indicated with a person smiling, this could be one potential task which requires putting more attention to when thinking of improving the performance of our CNN.

V. Real Time Classification

The goal of Areya was to be able to classify and provide immediate notification to the doctor responsible in real life, therefore after having researched, the use of OpenCV's Haar Cascades file which is widely used to detect and extract particular facial details such as eyes, mouth, eyebrows from a webcam video feed. The working process of the default included detecting if a face was seen in the region of interest which was identified by forming a blue square which included the individuals face. Following in time, because the pixel values of our faces being detected in real time from my computers default camera, the resolution and/or the pixel values were extremely high and so to make the face detected by the camera fit into the model, the need for rescaling and normalizing the pixel values was needed. While detecting emotions in real time, when only negative emotions such as Sad or Angry was detected a message on the top saying "Patient Needs Immediate Attention" was visible.

Future Implementations

For future implementations three main areas would require additional researching and working for improving the performance of the model and achieving higher accuracy percentage. Firstly, finding the back story and tuning of measures to make predictions more accurate in real time so that the notification for the negative emotions can be displayed efficiently. This would help is boosting the initial aim of the project. Secondly, adding more images withing our dataset and them being accurate with the emotion displayed and training them for a longer period of time because as seen above, the training accuracy for the second custom CNN was less than the testing. Lastly, achieving training accuracy higher than the testing accuracy. It is essentially, quite a challenge to understand the particular area which could be generating some error but the research on Deep Learning is still active.

Conclusions

To conclude, the main research question of this project is   "How can the AI technology help in augmenting the relationship between the patient and the doctor by cognizing their emotions?" To provide an answer to this question after having understood the behavior of CNN models and few pretrained models, Artificial Intelligence holds great power in creating magic. In this project, I am confident in saying that learning CNNs and experimenting with their functions was a great challenge for me which broadened my horizons of Deep Learning and made me realize the power of being able to create impact with technology. Considering the evaluations of my model Areya, the model was successfully built and performed quite well but possesses the capability to operate much better. Therefore, in near future the goal would be take actions on the future implementations.