National Institute of Calicut

Artificial Intelligence Theory and Practice

EC3066D

MINI PROJECT

Facial Emotion Recognition using CNN

M.S.V.M.Meghana B180970EC

M.Reshmasri B180695EC

M.Srichandrika B180637EC

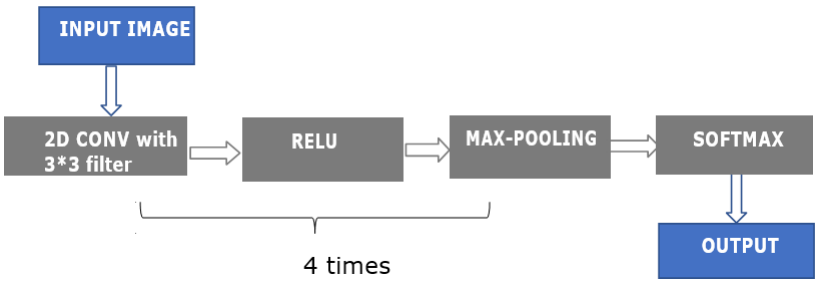
1 Problem statement

To classify each facial image into different facial emotion categories using convolutional neural networks.

2 Dataset

FER-2013 available in kaggle website is used to train and test the model. The data consists of 48x48 pixel grayscale images of faces with 7 different emotion classes.

3 Architecture



The network consists of four convolutional layers with fiter sizes of 32,64,128,256 repectively.Each layer is followed by a activation function relu and max pooling layer and batch normalisation is done for maintaining the range of X zero to one. A dropout of rate 0.2 is applied to each layer to reduce overfitting.In order to convert the output into a single dimensional vector, the output of the previous layers was flattened.The kernel size, that is, the width and height of the 2D convolutional window is set to 3 x 3 for all convolutional layers. Each max pooling layer is two dimensional and uses a pool size of 2 x 2. This halves the size of the output after each pooling layer. The ReLU activation function is used here due to benefits such as sparsity and a reduced likelihood of vanishing gradient. The softmax activation function was used in the final output layer to receive the predicted probability of each emotion.

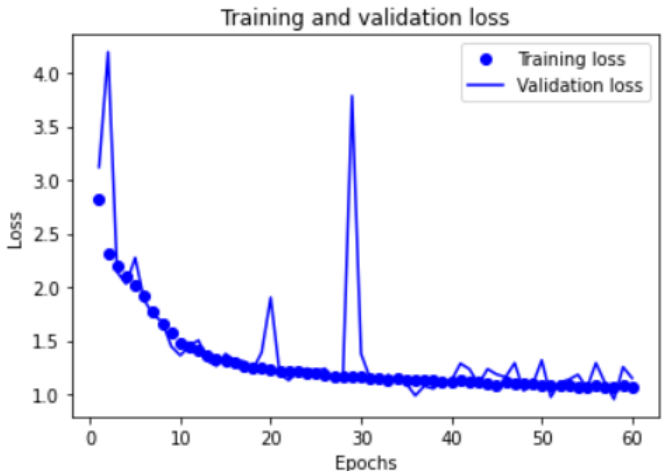
4 Results

4.1 Train and Validation accuracy

```
Epoch 60/60
1436/1436 [==============================] - 42s 30ms/step - loss: 1.0577 - accuracy:
0.5011 - val_loss: 1.1499 - val_accuracy: 0.4958
```

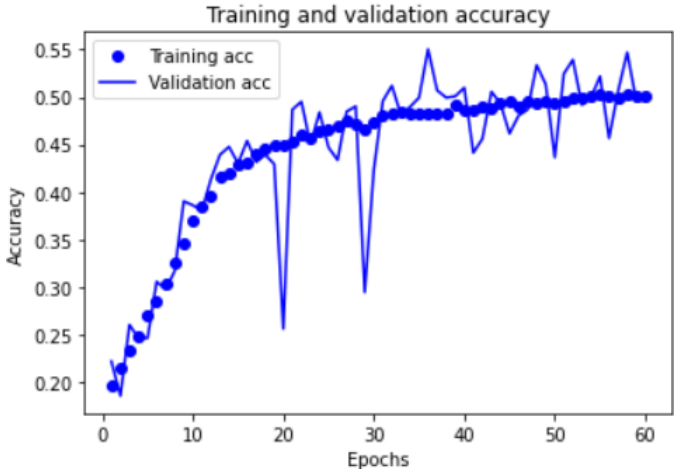
Training accuracy is 50.11%.
Validation accuracy is 49.58%

4.2 Train and Validation loss vs Epochs



As number of epoch increases loss decreases.

4.3 Train and Validation accuracy vs Epochs



As number of epoch increases accuracy increases.

4.4 Test accuracy

```
57/57 [==============================] - 12s 211ms/step - loss: 0.9072 - accuracy: 0.56
05


[0.9072079062461853, 0.5604625344276428]
```

Test accuracy is 56.05%.

5 Experiments

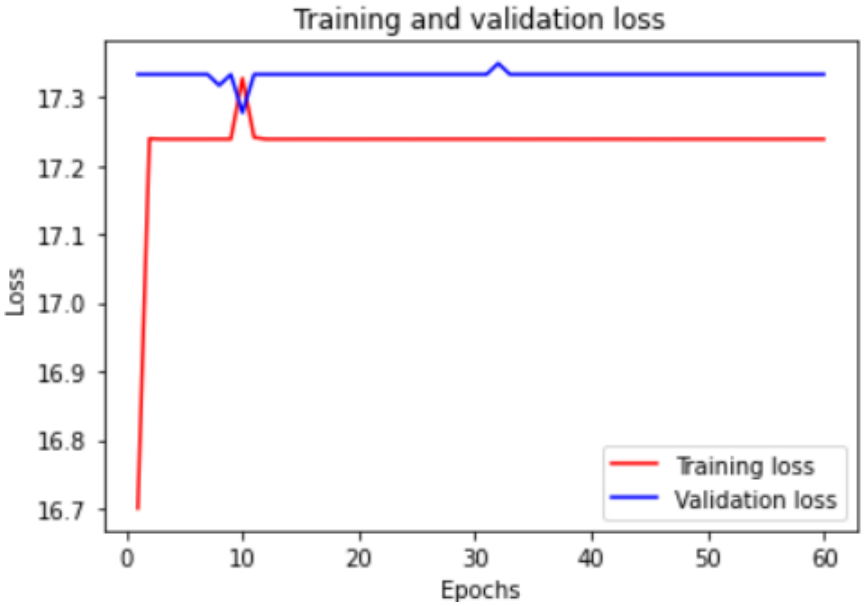
5.1 Changing learning rate from 0.001 to 0.01

5.1.1 Train and Validation accuracy

```
Epoch 60/60
1436/1436 [==============================] - 43s 30ms/step - loss: 17.2841 - accuracy:
0.2493 - val_loss: 17.3330 - val_accuracy: 0.2472
```

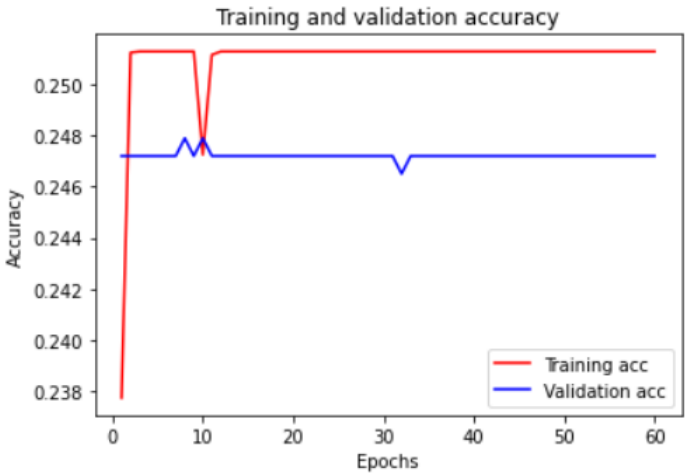
When learning rate increases accuracy is decreased.
Train accuracy is 24.93%.
Test accuracy is 24.72%.

5.1.2 Train and Validation loss vs Epochs



When number of epochs are increasing loss remains constant.

5.1.3 Train and Validation accuracy vs Epochs



When number of epochs are increasing accuracy is very low and remains constant.

5.1.4 Test accuracy

```
57/57 [==============================] - 39s 700ms/step - loss: 17.3345 - accuracy: 0.2
471


[17.334453582763672, 0.24714405834674835]
```

Test accuracy is very low i.e.. 24.71%.

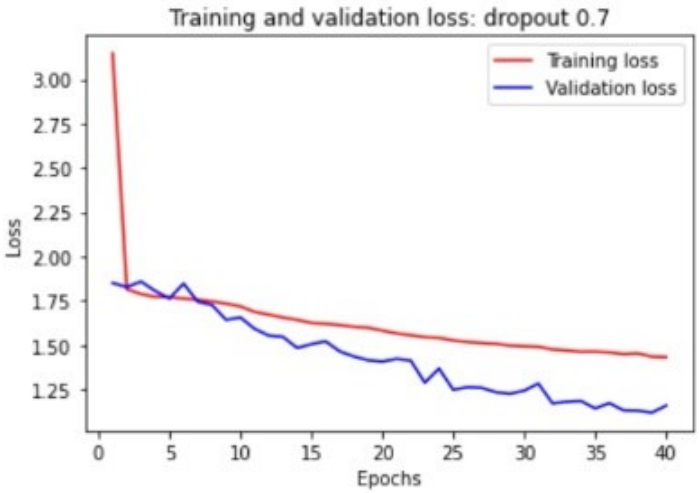
5.2 When dropout is 0.7

5.2.1 Train and Validation accuracy

```
Epoch 40/40
898/898 [==============================] - 188s 209ms/step - loss: 1.4148 - accuracy: 0.4005 - val_
loss: 1.1577 - val_accuracy: 0.4753
```

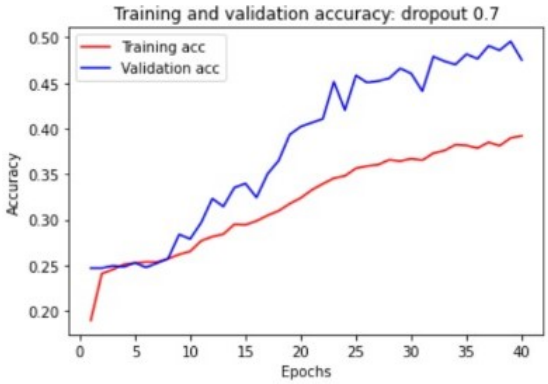
When dropout increases accuracy is decreased.
Train accuracy is 40.05%.
Test accuracy is 47.53%.

5.2.2 Train and Validation loss vs Epochs



When number of epochs are increasing loss decreases.

5.2.3 Train and Validation accuracy vs Epochs



When number of epochs are increasing accuracy increases.

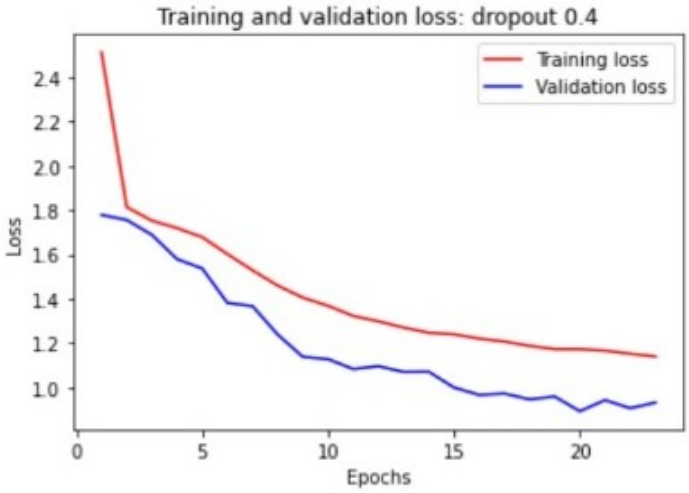
5.3 When dropout is 0.4

5.3.1 Train and Validation accuracy

```
Epoch 23/40
898/898 [==============================] - 190s 211ms/step - loss: 1.1252 - accuracy: 0.4851 - val_lo
ss: 0.9310 - val_accuracy: 0.5422
Restoring model weights from the end of the best epoch.
```

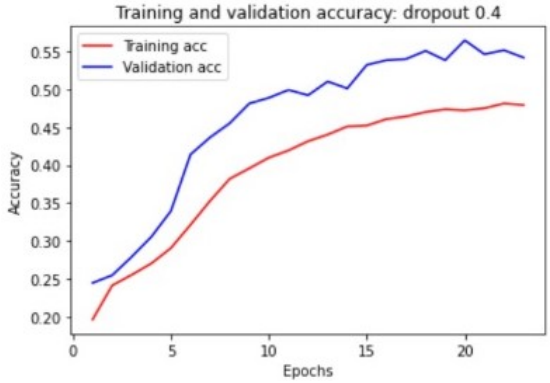
When dropout increases accuracy is decreased.
Train accuracy is 48.51%.
Test accuracy is 54.22%.

5.3.2 Train and Validation loss vs Epochs



When number of epochs are increasing loss decreases.

5.3.3 Train and Validation accuracy vs Epochs



When number of epochs are increasing accuracy increases.

5.4 When dropout is 0.2

5.4.1 Train and Validation accuracy

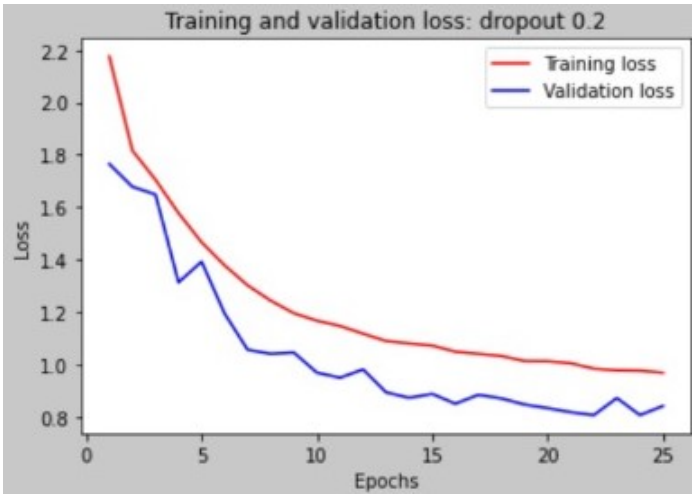
```
Epoch 25/40
898/898 [==============================] - 188s 210ms/step - loss: 0.9656 - accuracy: 0.5447 - val_lo
ss: 0.8410 - val_accuracy: 0.5805
Restoring model weights from the end of the best epoch.

Epoch 00025: ReduceLROnPlateau reducing learning rate to 0.00020000000949949026.
Epoch 00025: early stopping
```

When dropout increases accuracy is decreased.

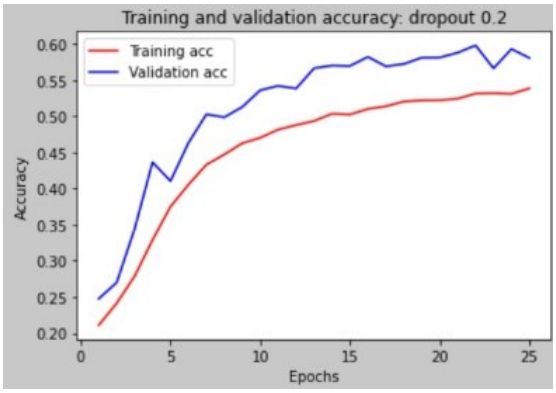
Train accuracy is 54.47%.
Test accuracy is 58.05%.

5.4.2 Train and Validation loss vs Epochs



When number of epochs are increasing loss decreases.

5.4.3 Train and Validation accuracy vs Epochs



When number of epochs are increasing accuracy increases.

Challenges:

- Our data is non-uniform and complex.
- It became very difficult for use to improve our model with the current architecture as it took more time for training and yet gave poor accuracy.
- Real time testing became impossible in colab since there is no live streaming possible in colab and needed another platform to test our model.

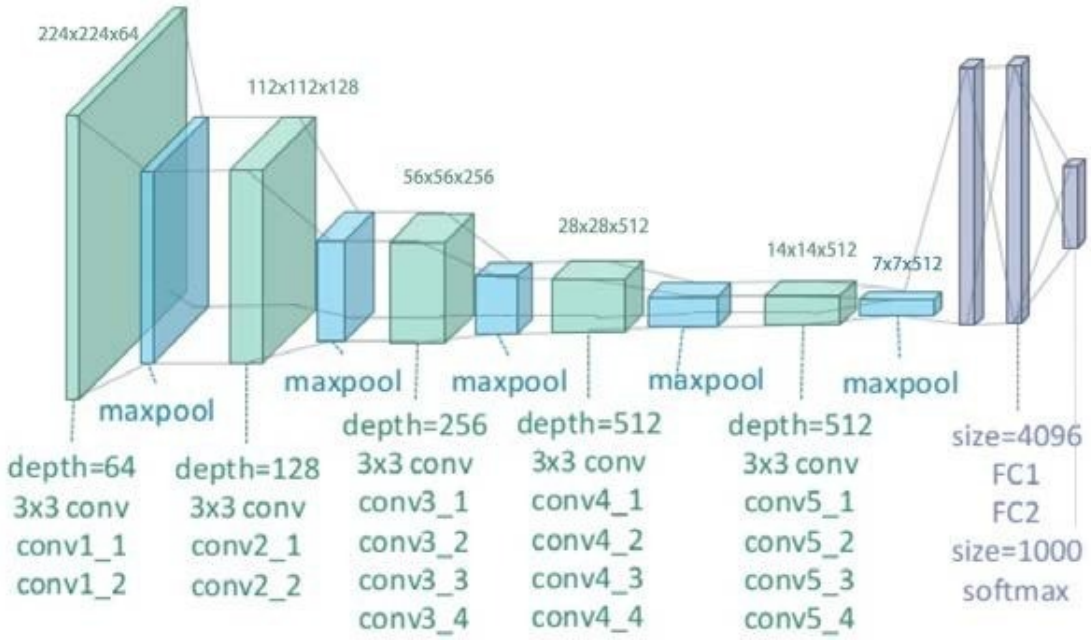
However after trying in many ways to improve the accuracy of our model our architecture wasn't effecient enough to do so, Hence we decided to use vggnet to increase our accuracy and use it to test our model.

6 VGG19

6.1 What is vggnet?

The VGG network architecture was introduced by Simonyan and Zisserman in their 2014 paper, Very Deep Convolutional Networks for Large Scale Image Recognition. This network is characterized by its simplicity, using only 3×3 convolutional layers stacked on top of each other in increasing depth. Reducing volume size is handled by max pooling. Two fully-connected layers, each with 4,096 nodes are then followed by a softmax classifier. vgg16 and vgg19 are 2 network layer architectures of vggnet where "16" and "19" stand for the number of weight layers in the network So it is a pre-trained networks included in the Keras core library represent some of the highest performing Convolutional Neural Networks on the ImageNet challenge over the past few years.

6.2 Architecture



- A fixed size of (224 * 224) RGB image was given as input to this network which means that the matrix was of shape (224,224,3).

- The only preprocessing that was done is that they subtracted the mean RGB value from each pixel, computed over the whole training set.
- Used kernels of (3 * 3) size with a stride size of 1 pixel, this enabled them to cover the whole notion of the image.
- Spatial padding was used to preserve the spatial resolution of the image.
- Max pooling was performed over a 2 * 2 pixel windows with sride 2.
- This was followed by Rectified linear unit(ReLu) to introduce non-linearity to make the model classify better and to improve computational time as the previous models used tanh or sigmoid functions this proved much better than those.
- Implemented three fully connected layers from which first two were of size 4096 and after that a layer with 1000 channels for 1000-way ILSVRC classification and the final layer is a softmax function.

6.3 Model

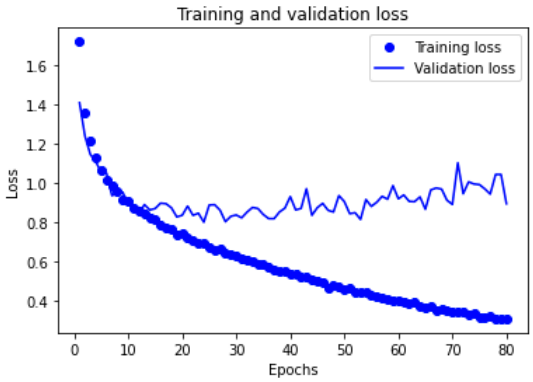
```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_4 (InputLayer)         [(None, 48, 48, 3)]       0
_________________________________________________________________
vgg19 (Functional)           (None, 1, 1, 512)         20024384
_________________________________________________________________
flatten_1 (Flatten)          (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 7)                 3591
=================================================================
Total params: 20,027,975
Trainable params: 20,027,975
Non-trainable params: 0
_________________________________________________________________
```

6.4 Train and Validation accuracy

```
Epoch 80/80
180/180 [==============================] - 45s 248ms/step - loss: 0.3085 - acc: 0.7402
- val_loss: 0.8946 - val_acc: 0.6271
```

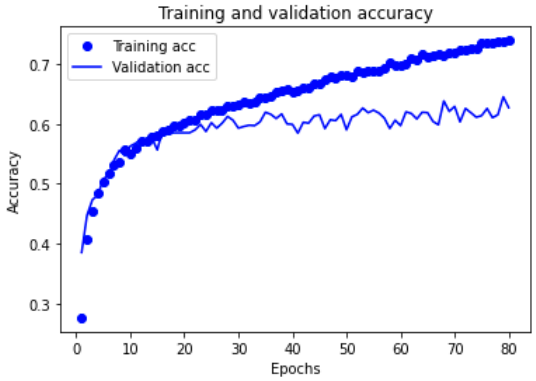
Train accuracy is 74.02%.
Test accuracy is 62.71%.

6.5 Train and Validation loss vs Epochs



Loss decreases as number of epoch increases.

6.6 Train and Validation accuracy vs Epochs



Accuracy increases as number of epoch increases.

6.7 Test accuracy

```
57/57 [==============================] - 6s 103ms/step - loss: 0.9755 - acc: 0.6555


[0.9754959344863892, 0.6554750800132751]
```

Test accuracy is 65.55%.

7 Real time

Since our dataset is a set of gray-scale images with pixel size 48×48 , while real time data is RGB images with different pixel values we have to convert real time

data to fit with our data.
Steps involved:

- From streaming vedio data we have to convert RGB to gray scale.
- From the gray-scale video we used haarcascade_frontalface_classifier to find faces in the frames of video.
- Every pixel inside the ROI(region of interest) indicate data of the face. We have to resize this roi to 48×48 pixel data .
- Pixel values are converted to arrays and normalised as of in our trained data set.

Finally using our model we can classify facial expressions.

7.1 Results