Convolutional Neural Networks for Facial Expression Recognition

Tiasha Mondal
CS826: Deep Learning Theory and Practice
Department of Computer and Information Sciences
University of Strathclyde
Glasgow, UK
tiasha.mondal.2022@uni.strath.ac.uk

***Abstract*—In This electronic document, I have developed a Convolution Neural Network(CNN) for facial expression recognition. The objective is to assign each face image to one of the seven categories of facial emotion considered in this study. Using sample gray-scale photos from the Kaggle platform, I have trained CNN models with varying depths [1]. I have used Tensorflow [2] to create the models, and to speed up the training procedure, the advantage of GPU processing has been used. In addition to this I have used different techniques like dropout, batch normalization, and batch normalization to reduce the overfitting issue of the models. Enhancing the performance of convolutional neural network models is the main objective of this study. At the end, I have compared the performances of all the three CNN Model and shown the visualization of accuracy and loss to get the better clarifications and insights. Also, I have included the discussion and the future work related this experiment.**

Keywords*—FER-2013, CNN, Dropout, batch Normalization, over-fittiing, accuracy, loss function*

1. Introduction

A person's facial expressions of emotion are an intuitive reflection of their mental state and one of the most significant forms of interpersonal communication, even though voice is generally thought of as the primary means of human communication. These expressions contain a wealth of emotional information. Even if no words are spoken, we can understand the messages we send and receive when communicating nonverbally in great detail. Facial expressions are important in interpersonal interactions because they include nonverbal messages [3, 4]. For human-machine interactions, which can also be used in the clinical and behavioral sciences, automatic facial expression recognition is crucial.

Despite the fact that humans can read facial emotions almost instantaneously and precisely, accurate computer expression detection is still difficult to achieve. Even though humans can read facial emotions relatively instantaneously and correctly, accurate expression identification by machines is still difficult. Although numerous studies on face detection have been conducted over the past few years, feature. It is difficult to develop an automated system that accomplishes this task [6] due to extraction techniques and expression classification methods. I provide a method for identifying facial expressions using convolutional neural networks in this study (CNN). CNN is used to predict the facial expression from an image, and the predicted output can be either neutral, fear, anger, happiness, terror, sorrow, or contempt.

2. Methodology

I have used a Sequential function to stack one layer over another to build the CNN Model. The first layer of the shallow model consists of 32 filters and the activation function RELU. After that, I employed a second Convolution Layer with 64 filters using RELU, followed by a MaxPooling of (2,2). The network is guided to fully connected layers after training through number of convolutional layers, where I have implemented the Affine operation and ReLU nonlinearity. Similar to other neural networks, this one is. It has a layer of neurons, all of which are trained throughout the training process. The last layer of the model will be an Affine Layer that is fully connected and this is similar to any other neural network. It has a layer of neurons, all of which are being trained at the same time. A completely connected layer with dimensions equal to the categories in the classification issue will make up the model's top layer, which is called a "affine layer" or “dense Layer”. According to the needs, the user can select the quantity of completely linked and convolutional layers to employ. Models are created and then compiled utilising the Adam Algorithm-based Keras

Optimizer. I parsed the training data in the following step to train the model. Using a subset of the dataset for validation has allowed me to generalise the model.

By including a Dropout Layer into the next CNN Model, I have attempted to somewhat improve the performance of the shallow model. the issue of overfitting at some level. As a result, the overfitting problem has diminished. Last but not least, to test the performance, I created another Deep CNN Model with additional Convolution layers. The experiment and data sets have been thoroughly detailed here.

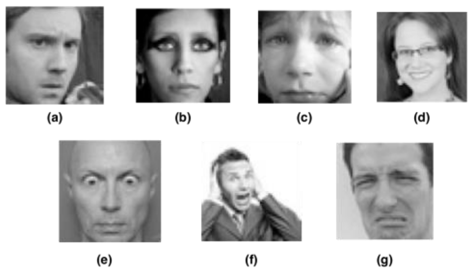


Figure 1: Examples of seven facial emotions from the dataset for this classification problem. (a) angry, (b) neutral, (c) sad, (d) happy, (e) surprise, (f) fear, (g) disgust

In order to speed up the model training process, I have developed the above mentioned model in Tensorflow and used GPU accelerated deep learning features.

2.1. Datasets

In this experiment, I used the FER-2013 dataset from Kaggle for my experiment, which consists of more than 35,000 well structured grayscale 48×48 pixel pictures of faces expressing a range of emotions and .Seven categories have been created based on these facial expressions - Figure 1 illustrates these emotions as happy, disgust, fear, happiness, sadness, neutral and angry. The given dataset are categorized into three different sets called training, validation, and test sets in addition to the image class number (a number between 0 and 6). There are roughly 29,000 training, 4,000 validation, and 4,000 images for testing, all belonging to 7 classes. Data Augmentation has been achieved by the Keras ImageGenerator[5] here. It accepts the original data, randomly transforms it, and returns only the new, transformed data.

2.2. Experiment

In order to accomplish this task, first I constructed a shallow CNN. There were two convolutional layers and one FC layer in this network. I have used 32 3×3 filters in the first convolutional layer and no maxpooling. In the second convolutional layer, I have used 128 3×3 filters and max-pooling with a filter size of 2×2. Did the same for the next layer as well. Then I have used Softmax as the loss function in the hidden layer of the FC layer, which had 1024 neurons. Rectified Linear Unit (ReLU) was used as the activation function in each layer as well. Using all of the training set's images, I have run 10 epochs. During the training of the model, it’s been to my notice that the accuracy of the training data is much higher than that of the validation data. This is a clear sign that the model is overfitted[6]. For this, I have used Dropout and batch normalization. It would be only used at the time of the training. I have trained the model same way expect just added Dropout after each convolution Layer. The dropout layer sets specific inputs to 0 which helps to reduce dependency on any particular set of inputs. Thus the overfitting problem has been resolved. Before adding the dropout, I have obtained an accuracy of 90% on the training data set and 58% on the validation data set which has been changed to 65% on the test data and 61% on the validation data set. Although the training accuracy has been reduced, but the difference between both the training and validation loss has been reduced.

To observe the effect of additional convolution network, the model has been trained with a deeper CNN Network with 4 Convolution and 1 FC Layers. The first layer was built using 64 3×3 filters, second layer was made of 128 3×3 filters, third and the fourth layer consists of 512 3×3 filters. All the layers consist of batch normalization, dropout, max-pooling and ReLU as the activation function. The first FC layer's hidden layer included 512 neurons followed by the dropout layer . ReLU were utilised as the activation function in the FC layer. Additionally, Softmax served as loss function. The architecture of this deep network is depicted in Figure 2. These sanity checks' outcomes demonstrated that the network's implementation was accurate. The network was then trained using all of the images in the training set while utilising 10 epochs. During the compile procedure learning-rate has been changed to 0.0001

This time, the accuracy on the validation set of 59% and accuracy on the test set of 58%. The values for each hyper-parameter in this model that had the maximum accuracy are shown in Table 2.

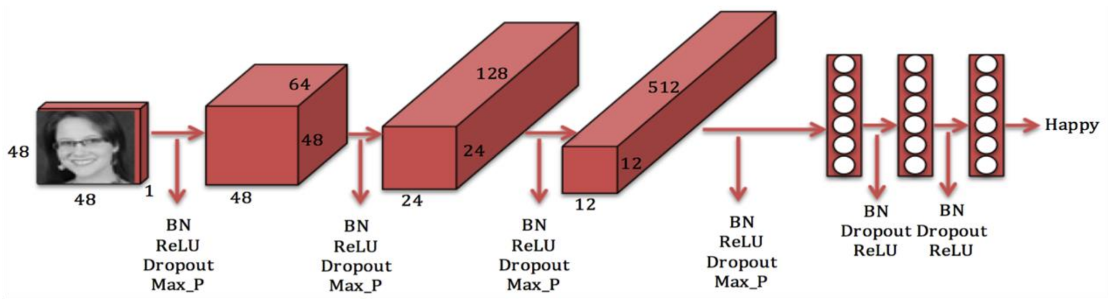


Figure 2: The architecture of the deep network: 4 convolutional layers and 2 fully connected layers

3. Results

To compare the performance of the shallow model the loss history and the accuracy has been plotted. In Figure 3, the overfitting issue is clearly visible for the shallow model while in Figure 4, we can see the both training and validity losses have been reduced with the epochs which clearly proves that the overfitting issue has been resolved. In Figure 5, the accuracy and loss have been plotted for the deep CNN Model. I have also included the model summary of all the three models below which gives a clear overview of the networks. The final accuracy of all the three models has been shown in table 3.

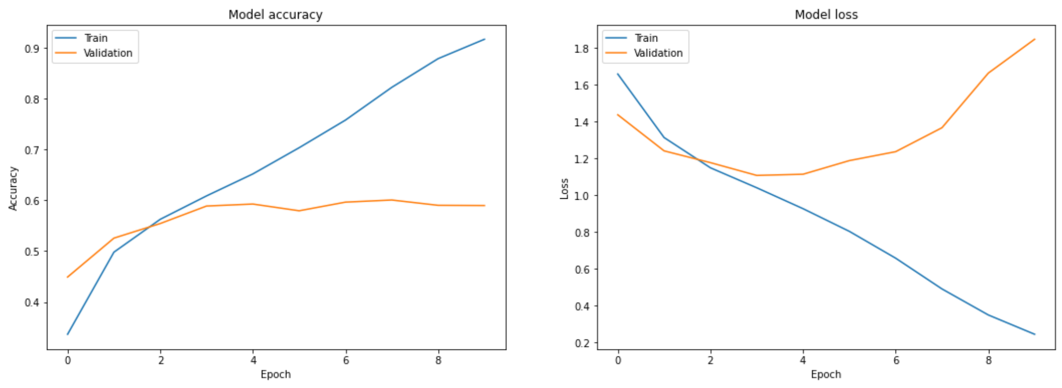


Figure 3: The accuracy and the loss function of train and validation data for different number of iterations of shallow model

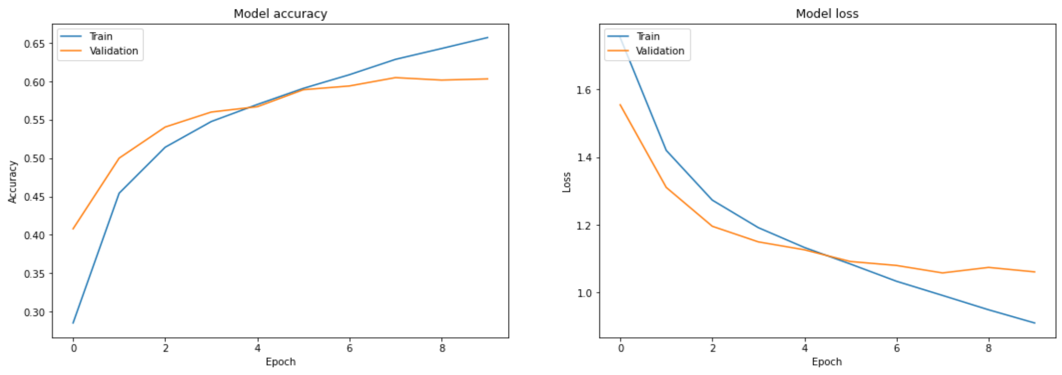


Figure 4: The accuracy and the loss function of train and validation data for different number of iterations of the improved CCN Model

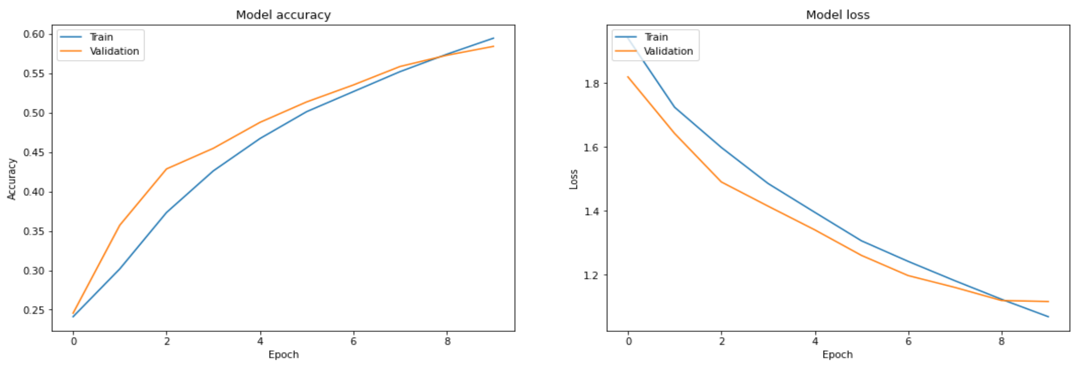
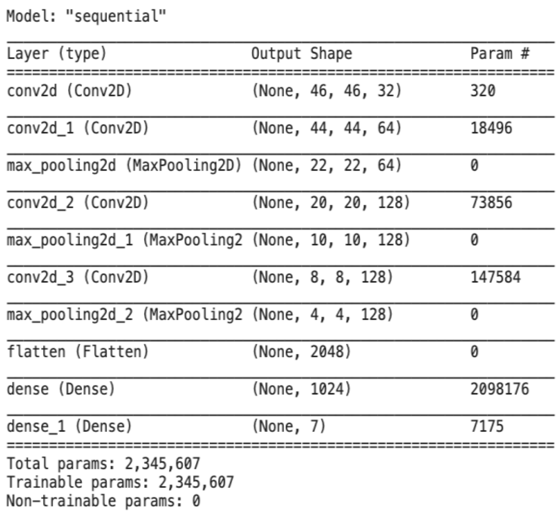
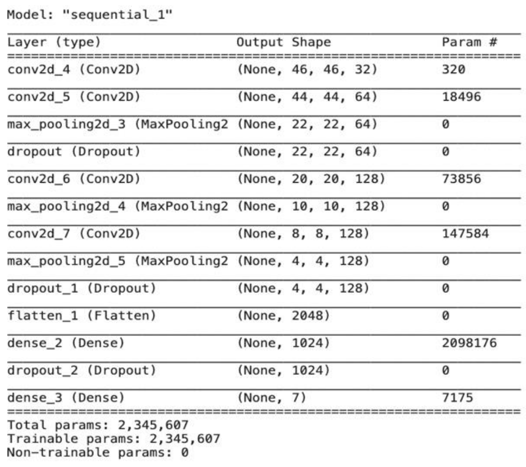


Figure 5: The accuracy and the loss function of train and validation data for different number of iterations of the Deep CCN Model after hyperparameter tuning.



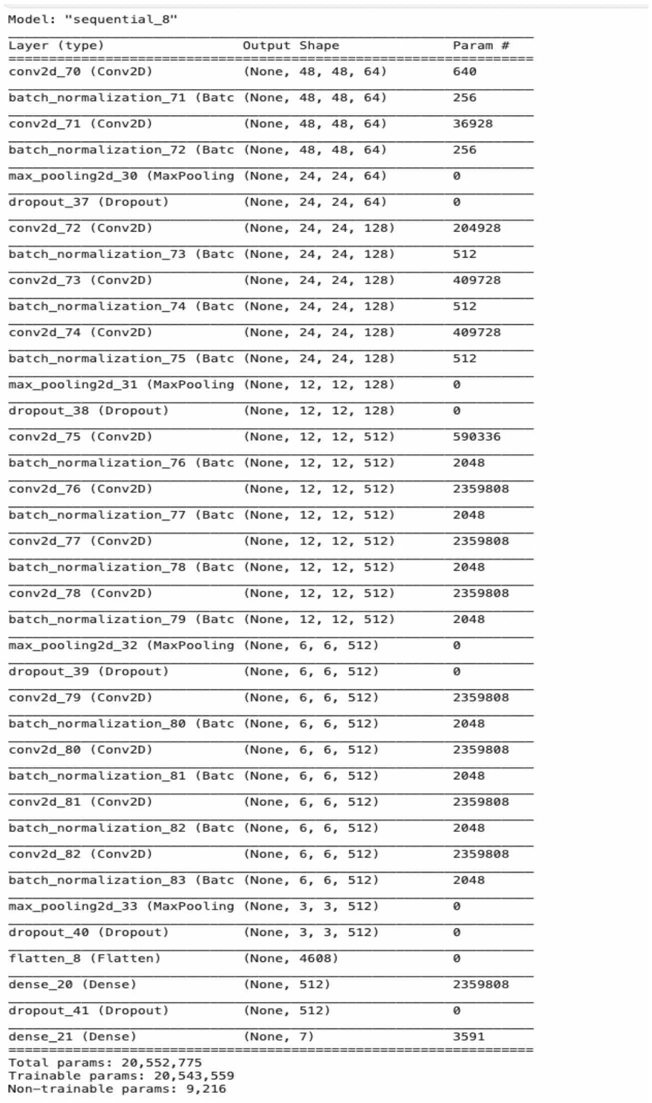
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 46, 46, 32)        320
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 44, 44, 64)        18496
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 22, 22, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 20, 20, 128)       73856
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 10, 10, 128)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 8, 8, 128)         147584
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 4, 4, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 2048)              0
_________________________________________________________________
dense (Dense)                (None, 1024)              2098176
_________________________________________________________________
dense_1 (Dense)              (None, 7)                 7175
=================================================================
Total params: 2,345,607
Trainable params: 2,345,607
Non-trainable params: 0
```

Figure 6: Model Summary for the Shallow Model



```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 46, 46, 32)        320
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 44, 44, 64)        18496
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 22, 22, 64)        0
_________________________________________________________________
dropout (Dropout)            (None, 22, 22, 64)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 20, 20, 128)       73856
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 10, 10, 128)       0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 8, 8, 128)         147584
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 4, 4, 128)         0
_________________________________________________________________
dropout_1 (Dropout)          (None, 4, 4, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2048)              0
_________________________________________________________________
dense_2 (Dense)              (None, 1024)              2098176
_________________________________________________________________
dropout_2 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_3 (Dense)              (None, 7)                 7175
=================================================================
Total params: 2,345,607
Trainable params: 2,345,607
Non-trainable params: 0
```

Figure 7: Model Summary for the improved model



```
Model: "sequential_8"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_70 (Conv2D)           (None, 48, 48, 64)        640
_________________________________________________________________
batch_normalization_71 (Batc (None, 48, 48, 64)        256
_________________________________________________________________
conv2d_71 (Conv2D)           (None, 48, 48, 64)        36928
_________________________________________________________________
batch_normalization_72 (Batc (None, 48, 48, 64)        256
_________________________________________________________________
max_pooling2d_30 (MaxPooling (None, 24, 24, 64)        0
_________________________________________________________________
dropout_37 (Dropout)         (None, 24, 24, 64)        0
_________________________________________________________________
conv2d_72 (Conv2D)           (None, 24, 24, 128)       204928
_________________________________________________________________
batch_normalization_73 (Batc (None, 24, 24, 128)       512
_________________________________________________________________
conv2d_73 (Conv2D)           (None, 24, 24, 128)       409728
_________________________________________________________________
batch_normalization_74 (Batc (None, 24, 24, 128)       512
_________________________________________________________________
conv2d_74 (Conv2D)           (None, 24, 24, 128)       409728
_________________________________________________________________
batch_normalization_75 (Batc (None, 24, 24, 128)       512
_________________________________________________________________
max_pooling2d_31 (MaxPooling (None, 12, 12, 128)       0
_________________________________________________________________
dropout_38 (Dropout)         (None, 12, 12, 128)       0
_________________________________________________________________
conv2d_75 (Conv2D)           (None, 12, 12, 512)       590336
_________________________________________________________________
batch_normalization_76 (Batc (None, 12, 12, 512)       2048
_________________________________________________________________
conv2d_76 (Conv2D)           (None, 12, 12, 512)       2359808
_________________________________________________________________
batch_normalization_77 (Batc (None, 12, 12, 512)       2048
_________________________________________________________________
conv2d_77 (Conv2D)           (None, 12, 12, 512)       2359808
_________________________________________________________________
batch_normalization_78 (Batc (None, 12, 12, 512)       2048
_________________________________________________________________
conv2d_78 (Conv2D)           (None, 12, 12, 512)       2359808
_________________________________________________________________
batch_normalization_79 (Batc (None, 12, 12, 512)       2048
_________________________________________________________________
max_pooling2d_32 (MaxPooling (None, 6, 6, 512)         0
_________________________________________________________________
dropout_39 (Dropout)         (None, 6, 6, 512)         0
_________________________________________________________________
conv2d_79 (Conv2D)           (None, 6, 6, 512)         2359808
_________________________________________________________________
batch_normalization_80 (Batc (None, 6, 6, 512)         2048
_________________________________________________________________
conv2d_80 (Conv2D)           (None, 6, 6, 512)         2359808
_________________________________________________________________
batch_normalization_81 (Batc (None, 6, 6, 512)         2048
_________________________________________________________________
conv2d_81 (Conv2D)           (None, 6, 6, 512)         2359808
_________________________________________________________________
batch_normalization_82 (Batc (None, 6, 6, 512)         2048
_________________________________________________________________
conv2d_82 (Conv2D)           (None, 6, 6, 512)         2359808
_________________________________________________________________
batch_normalization_83 (Batc (None, 6, 6, 512)         2048
_________________________________________________________________
max_pooling2d_33 (MaxPooling (None, 3, 3, 512)         0
_________________________________________________________________
dropout_40 (Dropout)         (None, 3, 3, 512)         0
_________________________________________________________________
flatten_8 (Flatten)          (None, 4608)              0
_________________________________________________________________
dense_20 (Dense)             (None, 512)               2359808
_________________________________________________________________
dropout_41 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_21 (Dense)             (None, 7)                 3591
=================================================================
Total params: 20,552,775
Trainable params: 20,543,559
Non-trainable params: 9,216
```

Figure 8: Model Summary for Deep CNN the Model

4. A Comparison Study

In this section, a comparison study between the shallow model , improved model and the Deep Model has been performed. The learning rate of the shallow model and the improved model has been kept as 0.001. Table 1 shows the hyperparameters used to train this model. In the improved CNN Model, the hyperparameters have not been changed but the dropout has been added to resolve the overfitting issue, whereas in the Deep CNN Model, the hyperparameters have been tuned and added a few more dense layers as shown in the Table 2. Comparison between the performance of the 3 models has been discussed in the table 3. As we can see, in the shallow model although there is a high train accuracy, but the difference between the train loss and the validation loss is also high which is clearly not a sign of a good training model. In the next improved CNN model, this issue has been fixed comparatively. In the Deep CNN model, I have changed the hyperparameters and the batch normalization and as a result the result has been more accurate. Hence, the optimized way is to go with the Deep CNN Model only.

Hyper-parameter	Value
Learning Rate	0.001
Hidden Neurons in Dense Layer	1024

Table 1: The hyper-parameters obtained cross validation for the shallow model

Hyper-Parameter	Value
Learning Rate	0.0001
Hidden Neurons in Dense Layer	512

Table 2: The hyper-parameters obtained cross validation for the Deep model

CNN Model	Train Loss	Validation Loss	Train Accuracy	Validation Accuracy
Shallow Model	0.26	1.8	90%	58%
Improved CNN Model	0.91	1.0	65%	60%
Deep CNN Model	1.0	1.1	59%	58%

Table 3: A comparison study between the performance of the three models

5. Summary

5.1. Conclusion

For a facial expression recognition task, variety of CNNs have been created and assessed their performance using various post-processing and visualisation techniques. The findings showed that deep CNNs are able to learn facial features and enhance facial emotion identification.

5.2. Future Work

In the literature review, I observe that despite of using a number of methodologies like Batch Normalization, dropout , although the overfitting issue is being resolved but the accuracy lies between 59% and 60% .This is because a network with huge network weights may imply an unstable network, where minor changes in the input might lead to significant changes in the output. Also, the model could have been trained for a few more epochs to improve the accuracy to some extent. Therefore, in future this might be the goal of research. Alongside improving the accuracy, I am looking forward to extend my model to colour images instead of greyscale images. This will enable to examine the effectiveness of pre-trained models for face emotion identification, such as AlexNet[7] or VGGNet[8]. Implementing a face identification process and then an emotion prediction process would be another extension. The feature extraction process is comparable to pattern recognition, which is used for identification in forensics[9], the military, and intelligence.
Thus, methods for pattern recognition such as the Capsnet algorithm[10] can be taken into consideration. DL-based techniques are challenging to apply on smartphones and other platforms with low resources since they need a sizable labelled dataset, a lot of memory, and lengthy training and testing cycles. Therefore, less complex solutions should be created that use less storage and memory.

References

[1] https://www.kaggle.com/c/challenges-inrepresentation-learning-facial-expressionrecognition-challenge/data

[2] https://github.com/tensorflow

[3] Bettadapura, Vinay (2012). Face expression recognition and analysis: the state of the art. arXiv preprint arXiv:1203.6722

[4] Lonare, Ashish, and Shweta V. Jain (2013). A Survey on Facial Expression Analysis for Emotion Recognition.

[5] https://stanford.edu/~shervine/blog/keras-how-to-generate-data-on-the-fly

[6] https://www.investopedia.com/terms/o/overfitting.asp#:~:text=Overfitting%20is%20a%20modeling%20error,to%20any%20other%20data%20sets.

[7] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. ”Imagenet classification with deep convolutional neural networks.” Advances in neural information processing systems. 2012.

[8] Simonyan, Karen, and Andrew Zisserman. ”Very deep convolutional networks for large-scale image recognition.” arXiv preprint arXiv:1409.1556 (2014).

[9] https://saudijournals.com/media/articles/SJEAT_512_486-490.pdf

[10] https://www.researchgate.net/publication/341870683_CapsNets_algorithm