Facial Expression Analysis and Recognition

Maazin Zaidi, Masab bin Nasim, Usama Sohail, Sana Fatimah

January 15, 2024

Abstract

This paper proposes an application of Convolutional Neural Networks (CNNs) in the domain of Emotion Recognition. Leveraging the power of deep learning, a robust model was developed, capable of accurately identifying and classifying human emotions from facial expressions. The CNN architecture is trained on a diverse dataset, encompassing a wide range of emotions and facial variations.

1 Introduction

Emotion Recognition generally refers to the analysis and recognition of facial expressions. This can be done in multiple different ways by the use of various machine learning algorithms. These include, but are not limited to kNNs (k-Nearest Neighbours), Bayesian Classifiers and Single Layer Perceptrons.

However, for the sake of this project, a Convolutional Neural Network (CNN) has been used. It is made up of multiple different layers in order to identify a facial expression in a given image. CNNs include several different types of layers, such as Convolution, Batch Normalization, Max Pooling and Fully Connected Dense Layers. The project makes use of all of them. This is discussed later in greater detail.

Finally, the efficacy of the project has been measured by well-known performance metrics. This is done by generating a Confusion Matrix, and then using that to calculate the Precision, Accuracy, Recall and F1 Score.

2 Methodology

2.1 Data Preparation

The dataset was downloaded from Kaggle.

The first step was to count the total number of images present in both the test and train folders. All of the images (both folders) were then stored in a numpy array called allImages. At the same time, the labels of the corresponding expressions (happy, sad, fearful, neutral, angry, disgusted, and surprised) were collected in a different array. Those labels were then converted into a one hot encoded form in order for it to be recognised by the CNN model.

The images and labels were then split into training, validation and test arrays using the train_test split. Roughly, these ratios were in a 70/20/10 format. Which means that from the original dataset, 70% of images were used for training, 20% for validation and the remainder 10% were unseen images that were used for testing.

2.2 Model Creation

Once the preprocessing was complete, a CNN structure was constructed. The first step in this process was to import the necessary functions from the tensorflow libraries. These included the model type (Sequential), the layers (Conv2D, MaxPooling2D, Dense, Flatten, Dropout, and BatchNormalization), the necessary optimizer (Adam) and a callback (EarlyStopping).

2.2.1 Architecture & Hyperparameters

In order to create the model, the architecture made use of 5 repetitions of the same block. This block started with a convolutional layer. The main aim of this layer would be to apply different filters in order to extract several features from the images. Following the convolutional layer, Batch Normalization was applied to normalize the activations and improve training stability. Subsequently, a MaxPooling2D layer with a pool size of (2,2) was employed to reduce the spatial dimensions. To prevent overfitting, a Dropout layer with a rate of 0.25 was added, randomly setting a fraction of input units to zero during training.

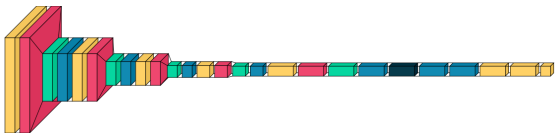

Figure 1: Neural Network Architecture

The way that the five blocks were progressed was by changing the number of filters in each block. The first block had 32 filters, the next had 64, then 128, then another 128 filters, and we ended with 256 filters. Each of these filters had a kernel size of 3x3, used 'same' padding and the ReLU activation function.

Similar to any other CNN, the data was flattened before being passed on to the fully connected layers. However, before passing the data onto the FC layers, two more dropout layers were added in order to cater to any overfitting that the model might experience. Finally, the data was passed onto 3 dense layers. The first two dense layers added 512 neurons each and have a similar ReLU activation. The last layer is also a dense layer. This one had 7 neurons, equal to the number of classes that we had intially. The softmax activation function ensures that the raw outputs are converted into probabilites, making it easier to interpret the model's predictions.

The architecture is shown in Figure 1 [2].

One final hyperparameter to consider is the learning rate of the Adam optimizer. Again, this was extensively tested on, and in the end, a learning rate of 0.0005 was settled upon.

2.2.2 Batch Size and Number of Epochs

After various training runs, and observing the training and validation accuracy and loss graphs, the optimal batch size was decided to be 40. While running the CNN for 20 epochs proved to render the best results.

3 Results

The results of any CNN depends on the hyperparameters that were set. All of these were mentioned and explained in detail in Section 2.2.

Each one of those parameters was varied to make sure that not only does the accuracy increase, but also that the model does not overfit during its training.

The first checkpoint of measuring the accuracy of the model was by observing the training and validation loss and accuracy graphs. After that, the model was evaluated on the unseen test data. Finally, using the sklearn.metrics library, a confusion matrix was created. From this confusion matrix, a classification report, highlighting the precision, recall and F1 score was generated.

3.1 Training & Validation Graphs

While the model trains, a separate set of data called the validation set, continuously ensures that the weights are being properly updated in the process. This also helps in visualizing any sorts of overfitting that might occur during the training phase. The graphs are shown in Figure 2

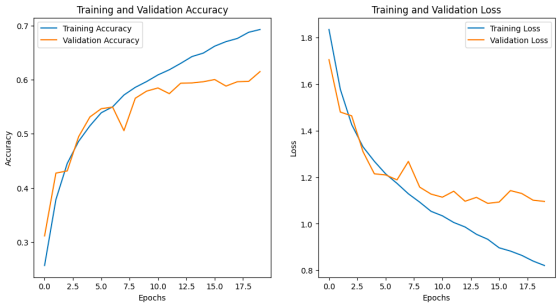

Figure 2: Accuracy & Loss Graphs

3.2 Model Evaluation

This step was rather straightforward, it simply involved calling the evaluate() function available within tensorflow's library. The function outputs the categorical accuracy of the model when run on the test data along with the ground truths. The categorical accuracy measures the percentage of correctly predicted labels out of the total number of samples. This turned out to be 62.66%.

3.3 Confusion Matrix & Classification Report

A confusion matrix is a table used in classification to evaluate the performance of a machine learning model. It provides a detailed breakdown of the model's predictions and actual class labels, allowing for a more comprehensive understanding of the model's behavior. The confusion matrix is shown in

Figure 3

A classification report summarizes various metrics that assess the model's ability to correctly classify instances into different classes, including precision, recall, F1-score, and support for each class, as well as overall metrics like accuracy.

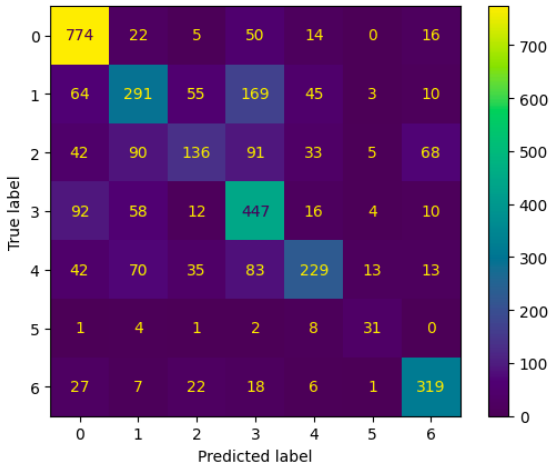

Figure 3: Confusion Matrix

	precision	recall	f1-score	support
0	0.773143	0.862656	0.815451	881.000000
1	0.476252	0.582418	0.524011	637.000000
2	0.486486	0.309677	0.378449	465.000000
3	0.590417	0.597809	0.594090	639.000000
4	0.616402	0.480412	0.539977	485.000000
5	0.622222	0.595745	0.608696	47.000000
6	0.727700	0.775000	0.750605	400.000000
accuracy	0.626899	0.626899	0.626899	0.626899
macro avg	0.613232	0.600531	0.601611	3554.000000
weighted avg	0.621070	0.626899	0.618613	3554.000000

Figure 4: Classification Report

4 Discussion

This project delves into the realm of Emotion Recognition, employing Convolutional Neural Networks (CNNs) to build a potent model for the accurate identification and classification of human emotions from facial expressions. The chosen CNN architecture, detailed in Section 2.2, comprises five blocks with varying filter counts, incorporating essential layers such as Convolutional, Batch Normalization,MaxPooling2D,and Dropout layers. Careful consideration of hyperparameters, including batch size and epochs, was pivotal in achieving optimal model performance.

The model's proficiency was evaluated on a diverse dataset, resulting in a categorical accuracy of 62.66%. The training and validation graphs, depicted in Figure 2, reflect the network's ability to generalize while avoiding overfitting. The confusion matrix (Figure 3) provides a detailed breakdown of correct and misclassifications, offering insights into the model's performance across distinct emotion categories. Further refinements could involve exploring alternative architectures and incorporating larger datasets to enhance real-world applicability.

Although the accuracy of the project seems below par, upon meticulous review of pretrained models, such as MobileNetV3 (Small and Large), VGG-16 and LeNet-15, it was seen that this architecture performed rather well. Their respective accuracies are shown in Table 1.

In addition to that, other attempts at creating a perfect architecture for this problem had an accuracy similar to the model proposed in this paper [1, 3, 4, 5]. The codes to all of these models can be found on Kaggle.

Table 1: Reviewed Pretrained Models

Model	Accuracy
VGG-16	0.590417
LeNet-15	0.616402
MobileNetV3-Small	0.773143
MobileNetV3-Large	0.486486

5 Conclusion

In summary, this project on Facial Expression Analysis and Recognition leverages Convolutional Neural Networks to successfully identify and classify human emotions from facial expressions. With a notable categorical accuracy of 62.66%, the model holds promise for applications in human-computer interaction and mental health monitoring.

While showcasing encouraging results, future refinements may involve exploring alternative approaches and incorporating more extensive datasets to enhance real-world usability.

References

[1] A. Dogra. Resnet50 emotion detection, 2023. Accessed Jan. 5, 2024.

[2] Paul Gavrikov. visualkeras. https://github.com/paulgavrikov/visualkeras, 2020.

[3] R. Janjua. Cnn for emotion detection, 2023. Accessed Jan. 6, 2024.

[4] Anand Kumar. Emotion recognisation transfer learning technique, 2023. Accessed Jan. 5, 2024.

[5] C. Liau. Deep learning project, 2023. Accessed Jan. 6, 2024.