**Expression Classification from Facial Images**A blue circle with text and a book

Description automatically generated

Name: Nouman Ali

Section:05

Roll No: DSAI\_GIL\_248

Github link: https://github.com/Noumzz

**Summary:**

In this project, we are trying to develop a facial expression recognition system using a deep learning approach. Using the "Expression in the Wild" dataset, we preprocess images and employ a Convolutional Neural Network (CNN) to classify emotions such as anger, disgust, fear, happy, sad, surprise, and neutral. To enhance model performance, we implement data augmentation techniques and experiment with varying input image sizes. The model's architecture is fine-tuned through rigorous validation and testing phases, with a focus on optimizing accuracy. Once the model is developed, we will test its performance using test data.

**Overview:**

Facial expression recognition (FER) is a challenging task in computer vision that involves interpreting human emotions based on facial movements and expressions. The complexity arises from variations in individual facial features, different lighting conditions, occlusions, and cultural differences in expressing emotions. As a result, developing an accurate and robust FER system is essential for many applications.

It has diverse use cases such as human computer interaction, it can be used in health care and also for security purposes.

**Literature review (2 articles from 2022-23):**

In the paper “Theoretical Understanding of Convolutional Neural Network:

Concepts, Architectures, Applications, Future Directions” author Mohammad Mustafa Taye discusses theories used in CNN. He discusses how CNN have dominated the computer vision and pattern recognition tasks in recent years. The author talks about the basics of CNN such as inputs, convolutional layers, pooling layers, activation functions, fully connected layers and output layers. Further, the author discusses some of the popular CNN architectures starting from LeNet (1998) to DenseNet-121 (2017) and also discusses the strengths and gaps of these architectures. He also mentions different types of CNN architectures used for classification, detection and segmentation.

In another article “Convolutional Neural Networks for Image Classification” author Jasmin Bharadiya discusses the importance of CNN in daily life. She mentions the key reasons for the significance of CNN. She discusses in about translation invariance, data efficiency in CNN, transfer learning and scalability. She discusses the benefits of transfer learning and how we can take advantage of using pre-defined models.

**Model architecture:**

The facial expression recognition model is built using a convolutional neural network (CNN) architecture designed to classify images into various emotion categories. It begins with an input layer that accepts images resized to 224x224 pixels with three color channels (RGB). This is followed by multiple convolutional layers that extract spatial features, each using ReLU activation to introduce non-linearity. Max pooling layers are incorporated to reduce spatial dimensions and mitigate overfitting. To further enhance generalization, dropout layers randomly deactivate a portion of neurons during training. The extracted features are then flattened and passed through fully connected layers to learn high-level representations. Finally, an output layer with a softmax activation function provides probabilities for each emotion class, enabling accurate multi-class classification. Transfer learning techniques may also be utilized to leverage pre-trained models, improving overall performance. The full architecture is shown in figure 1.1.

A screen shot of a computer program

Description automatically generated

**Figure:1.1**

**Dataset Details:**

The dataset utilized in this project consists of facial images sourced from the Expression in the Wild (ExpW) dataset, which includes diverse expressions across various subjects. To ensure high-quality training data, we implemented a filtering mechanism based on the confidence score associated with each image. This data set contained a label.lst file which included the name of the image, it’s confidence interval, dimension for face in a image, and the label of the image. We extracted only those images where the confidence interval exceeded 0.51, indicating a reliable detection of facial features. This confidence threshold serves to minimize noise and enhance the quality of the training dataset by ensuring that the model learns from well-annotated examples. By focusing on images with a confidence level greater than 0.51, we aimed to improve the model's ability to generalize and accurately classify facial expressions, thereby enhancing overall performance in emotion recognition tasks. After setting the confidence level to 0.51 we got total of 31122 train images, 6670 test, and 6670 validation images.

**Hyperparameters:**

In this model, various hyperparameters were utilized across different layers of the model architecture to optimize performance:

1. **Convolutional Layer Hyperparameters:** The model includes multiple convolutional layers with parameters such as filter size (3x3), number of filters (e.g., 32, 64, and 128), and activation functions (ReLU).
2. **Pooling Layer:** MaxPooling layers are implemented after convolutional layers, typically with a pool size of 2x2. This reduces the spatial dimensions of the feature maps, decreasing the computational load and helping to prevent overfitting while retaining essential features.
3. **Dense Layer and output layer:** The model features dense layers towards the end of the architecture, with 490, 196, and 49 layers in each of the layer. These layers facilitate the combination of features learned in the convolutional layers to make final predictions. The activation function used in these layers is often ReLU for hidden layers and softmax for the output layer to classify the expressions.
4. **Dropout Layer:** Dropout layers are included to mitigate overfitting by randomly dropping a fraction, in our model we used 0.3 and 0.2 dropout out layer in between the convolutional, dense and fully connected layers.
5. **Optimizer and Learning Rate:** The Adam optimizer is employed for its adaptive learning capabilities, with an initial learning rate set to 0.001. This optimizer adjusts the learning rate dynamically during training, helping the model converge efficiently.
6. **Batch Size:** A batch size of 64 is utilized, determining the number of training samples processed before the model's weights are updated. This choice balances memory usage and training stability, impacting the model's convergence rate.
7. **Number of Epochs**

Number of times our model trains with the training data is called epochs. We set the epoch to 15 for this model.

**Evaluation metrics:**

**Analysis of results:**

The accuracy of the model on the testing data was 64.38%.

The precision of the model on the testing data was 61%.

The recall of the model on the testing data was 64%.

The f1 score of the model on the testing data was 60%.

The classification report and test accuracy of 64.38% suggest that while the model has made some progress, it still has limitations. While it performs well on "Happy" (Precision: 0.75, Recall: 0.81) and "Neutral" (Recall: 0.85), it struggles with emotions like "Disgust" and "Fear," with both classes having 0 precision and recall, indicating complete misclassification.

The validation and training accuracy and loss is show in the figure 2.1 below.

A graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of

Description automatically generated

Fig 2.1

**Possible improvements:**

To improve the results, we can try balancing the dataset to ensure all emotions are represented equally, which can help the model perform better on underrepresented classes. Using a pre-trained model through transfer learning, like VGG16 or ResNet, may improve accuracy by building on learned features. Additionally, tuning the learning rate and adjusting dropout rates can make training more stable and reduce overfitting. Data augmentation, such as rotating or flipping images, can increase the diversity of the training set and improve generalization. If needed, increasing the image size might also help the model capture more detailed features. Finally, running more epochs with early stopping can ensure the model achieves optimal performance without overfitting.

**Github link to the repo:**

<https://github.com/Noumzz/ExpressionInWild/tree/main>

**References:**

Taye, M. M. (2023). Theoretical understanding of convolutional neural network: Concepts, architectures, applications, future directions. Computation, 11(3), 52. <https://doi.org/10.3390/computation11030052>

Bharadiya, J. (2023). Convolutional neural networks for image classification. International Journal of Innovative Research in Science, Engineering and Technology, 8, 673. <https://doi.org/10.5281/zenodo.7952031>