Problem Statement

The project "Facial Recognition and Emotion Analysis in Video Streams" aims to develop a real-time system capable of detecting faces, recognizing individuals, and analysing their emotional expressions within video streams. The goal is to develop a robust and accurate model that can reliably classify facial expressions into predefined categories such as Angry, Disgust, Fear, Happy, Sad, Surprise, and Neutral.Utilizing techniques such as face detection algorithms, facial recognition models, and deep learning-based emotion analysis, the system will identify individuals in the video feed and classify their emotions into predefined categories such as happiness, sadness, or anger.

Introduction

This is a deep learning model to recognize facial expressions from the FER 2013 dataset. The dataset consists of grayscale images of size 48x48 pixels, each labeled with one of seven emotions: Angry, Disgust, Fear, Happy, Sad, Surprise, and Neutral. The task involved data preprocessing, model building, training, and evaluation.

Approach

Dataset Loading and Preprocessing

Loading the Dataset:

The FER 2013 dataset was loaded using the Deeplake library. The dataset contains images and corresponding labels. Emotion of the images are stored in the corresponding labels.

Label Mapping:

A dictionary was created to map emotion labels to integers:
label_map = {'Angry': 0, 'Disgust': 1, 'Fear': 2, 'Happy': 3, 'Sad': 4, 'Surprise': 5, 'Neutral': 6}. This is important to convert the strings to labels to reduce the time and some extra line of code

Data Extraction:

The images and labels were extracted from the dataset, and the labels were converted from string to integer format using the label_map.

Data Normalisation:
Pixel values of the images were normalised to the range [0, 1] by dividing by 255. Additionally, the images were expanded to have a single channel dimension.

Data Splitting: The dataset was split into training, validation, and test sets using an 80-10-10 split ratio. train_test_split function was used from the library sklearn.model_selection.

Data Augmentation: To increase the diversity of the training data, data augmentation techniques such as rotation, width shift, height shift, and horizontal flip were applied using the `ImageDataGenerator` from TensorFlow.

Model Building

A Convolutional Neural Network (CNN) was built using TensorFlow and Keras. The architecture consisted of three convolutional layers followed by max-pooling layers, a flattening layer, and a final output layer with softmax activation. These Conv2D layers and Max Pool layers would train the model by changing the values of weight and bias parameter every epoch.

Model Training

The model was compiled using the Adam optimizer with a learning rate of 0.001 and trained using the sparse categorical cross-entropy loss function. The model was trained for 20 epochs(ran the cell 2 times) with a batch size of 64, using the augmented training data and the validation data for monitoring performance.The training accuracy that I got is around 60%.

Model Evaluation

The trained model was evaluated on the test set to determine its accuracy. Test accuracy is calculated by matching the predicted output(emotion) with the actual output.

Making Predictions

The model was used to make predictions on random test images(I just used random to check the model for different images each time the code is executed. As the accuracy is not so high, the model sometimes predicted emotions not matching with

the true emotion). The predicted labels were converted back to string format using the inverse of the `label_map` dictionary.

Experiments

Several experiments were conducted to fine-tune the model:

1. **Model Architecture:** Different numbers of convolutional layers and dense layers were experimented with to find the best architecture. I tried to add 5-6 layers, but that makes the execution very slow(it was taking hours to run 10 epochs).
2. **Hyperparameters:** Various values for learning rate, batch size, and dropout rate were tested to optimise the model performance.(then I decide to remove the dropout to increase the test accuracy)
3. **Data Augmentation:** Different augmentation techniques and parameters were used to improve the model's generalisation.

Results

The final model achieved a test accuracy of approximately 60%, indicating a moderate performance in recognizing facial expressions.This could have been higher, but due to less dataset for testing, it is not increasing much after 60.

Here is a sample prediction:

- **True Label:** Happy
- **Predicted Label:** Happy

This indicates that the model is able to correctly identify the emotion from the given image. But, for some images model is not predicting correctly. For example, an image with ‘Sad’ emotion was predicted to be ‘Fear’. This was already expected due to low accuracy.

Conclusion

In this project, a CNN Model was successfully built and trained to recognize facial expressions using the FER 2013 dataset. The approach involved thorough data preprocessing, model building, training, and evaluation. The final model showed a moderate level of accuracy, and further improvements could be achieved through more advanced techniques and hyperparameter tuning.

Prabhakar Yadav
23b0653