**Neural Networks Project - Gesture Recognition for Smart TVs**

***Presented by: Akash Kriplani and Ankit Kr Sharma***

**Problem Statement**

As a data scientist at a home electronics company specializing in advanced smart televisions, the objective is to develop a feature enabling the recognition of five distinct gestures performed by users. This feature aims to facilitate TV control without the need for a remote. The gestures, which include thumbs up, thumbs down, left swipe, right swipe, and stop, correspond to specific commands such as volume adjustment, playback control, and skipping within a video.

The dataset comprises several hundred videos, each categorized into one of the five gesture classes. Each video consists of a sequence of 30 frames captured by a webcam mounted on the TV. The training data is organized into 'train' and 'val' folders, with corresponding CSV files containing information about the videos and their labels. Additionally, the data is stored in subfolders, each representing a video of a particular gesture.

The videos exhibit varying dimensions, either 360x360 or 120x160, depending on the webcam used for recording. Therefore, preprocessing is necessary to standardize the videos before model training. Each row in the CSV files represents a video and includes the subfolder name containing the video frames, the gesture label, and a numeric label (ranging from 0 to 4).

The task entails training a model on the 'train' folder to perform well on the 'val' folder, adhering to standard practices in machine learning projects. The test folder is withheld for evaluation purposes, and the final model's performance will be assessed on this set.

To initiate the model building process, the first step is to obtain the dataset and store it locally. This can be accomplished by following the provided steps:

1. Open the terminal.

2. Go to following link: `https://drive.google.com/uc?id=1ehyrYBQ5rbQQe6yL4XbLWe3FMvuVUGiL`

3. Unzip the downloaded file named 'Project\_data.zip'.

**Solution Overview**

We will be using Convolutional Neural Network (CNN) combined with Recurrent Neural Network (RNN) architecture. In this architecture, the Conv2D network extracts a feature vector for each image, and subsequently, a sequence of these feature vectors is inputted into an RNN-based network. The output of the RNN is a conventional softmax.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Exp No** | **Model** | **Number of parameters** | **Hyperparameters** | **Result** | **Decision + Explanation** |
| 1 | CNN with LSTM | 1657445 | Number of sample frames = 30  Batch size = 20  Number of epochs = 20 | Training accuracy: 0.87  Validation accuracy: 0.71  Indication of overfitting | Utilizing the base CNN with LSTM architecture, we conducted training on all frames (30) extracted from the video. The training accuracy reached 87%, while the validation accuracy attained 71%, revealing potential signs of overfitting. To address this, we aim to explore a modified version of the same model by reducing the number of frames, batch size and number of epochs. |
| 2 | CNN with LSTM (with reduced hyperparameters) | 1657445 | Number of sample frames = 20  Batch size = 10  Number of epochs = 10 | Training accuracy: 0.72  Validation accuracy: 0.61  Indication of underfitting | Upon reducing the number of hyperparameters, the training accuracy drops to 72%, and the validation accuracy decreases to 61%, indicating underfitting. To address this issue, we will attempt to alleviate it by solely reducing the number of frames while keeping both the batch size and the number of epochs at 20. |
| 3 | CNN with LSTM (with reduced frames) | 1657445 | Number of sample frames = 18  Batch size = 20  Number of epochs = 20 | Training accuracy: 0.9291  Validation accuracy: 0.75  Indication of overfitting | The training accuracy demonstrates consistent improvement, achieving 92.91%, while the validation accuracy reaches a maximum of 75%, continuing to indicate potential overfitting. To address this concern, we aim to experiment with an alternative architecture, combining CNN with GRU, while maintaining the same number of frames. |
| 4 | CNN with GRU | 2573925 | Number of sample frames = 18  Batch size = 20  Number of epochs = 20 | Training accuracy: 0.926  Validation accuracy: 0.62  Indication of overfitting | By incorporating GRU-RNN, there is a marginal decline in training accuracy, reaching 92.6%, whereas the validation accuracy decreased to 62%, indicating overfitting. To tackle this issue, we propose introducing additional augmentation techniques such as rotation and reattempting the same architecture to mitigate overfitting. |
| 5 | CNN with LSTM (with modified Augmentation Technique) | 1657445 | Number of sample frames = 18  Batch size = 20  Number of epochs = 20 | Training accuracy: 0.91  Validation accuracy: 0.77  Slightly improved with augmentation | After introducing rotation in the augmentation phase, the training accuracy experienced a slight decrease to 91%, while the validation accuracy rose to 77%. We are now exploring transfer learning to assess whether it can further enhance the validation accuracy. |
| 6 | CNN with LSTM (with Transfer Learning) | 3840453 | Number of sample frames = 16  Batch size = 5  Number of epochs = 20 | Training accuracy: 0.9887  Validation accuracy: 0.74  Indication of overfitting | Utilizing LSTM with transfer learning resulted in an enhancement of the training accuracy to 98.87%. However, despite the validation accuracy increasing to 74%, it still suggests potential overfitting. This may be attributed to the lack of training of MobileNet weights. Let's investigate if this issue is mitigated by employing LSTM and transfer learning, while also training the weights from MobileNet. |
| 7 | CNN with LSTM (with trainable weights of Transfer Learning) | 3840453 | Number of sample frames = 16  Batch size = 5  Number of epochs = 20 | Training accuracy: 0.9465  Validation accuracy: 0.94  Improved accuracy but model training was computationally expensive | After training the MobileNet weights, the training accuracy surged to 94.65%, and the validation accuracy notably increased to 94%. However, this improvement came at the cost of extended training time due to a fourfold increase in the number of parameters used in the LSTM architecture. Next, we aim to explore whether we can enhance the training and validation accuracy by reducing the parameter count using GRU and leveraging the trainable weights of MobileNet. |
| Final Model | CNN with GRU (with trainable weights of Transfer Learning) | 3693253 | Number of sample frames = 16  Batch size = 5  Number of epochs = 20 | Training accuracy: 0.9940  Validation accuracy: 0.97  Finally, model based on accuracy metric | By employing GRU and transfer learning with pre-trained weights, we observe a remarkable increase in the training accuracy to 99.40% and validation accuracy to 97%. This indicates successful resolution of overfitting, and the model performs well on the validation set. Hence, we can regard this as our final model for evaluation. |