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

***Presented by: Shantnu kumar Singh and Sharan Abhishek***

**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/drive/folders/1kPwCFhnK9e6Lk61CRUhCt0G3ZulmS5dp`

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.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Experiment Number | 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: 0.92 Validation accuracy: 0.76  Indication of overfitting | Our initial gesture recognition model, using a CNN with LSTM architecture, was trained on all 30 frames extracted from each video. While we achieved a high training accuracy of 92%, the validation accuracy was lower at 76%, indicating potential overfitting. We plan to modify the model by decreasing the batch and epoch sizes to mitigate this. By implementing these adjustments, we aim to improve the model's ability to generalise to unseen data and achieve better overall performance. |
| 2 | CNN with LSTM (with reduced hyperparameters) | 1657445 | Number of sample frames = 20  Batch size = 10  Number of epochs = 10 | Training accuracy: 0.89 Validation accuracy: 0.78  Indication of underfitting | Our attempt to reduce overfitting by lowering the number of hyperparameters led to a decrease in both training accuracy (89%) and validation accuracy (78%), suggesting underfitting. To rectify this, we will focus on adjusting only the number of frames used as input to the model while maintaining a batch size and epoch count of 20. This approach aims to find a balance between model complexity and generalization ability. |
| 3 | CNN with LSTM (with reduced frames) | 1657445 | Number of sample frames = 18  Batch size = 20  Number of epochs = 20 | Training accuracy: 0.87 Validation accuracy: 0.74  Indication of overfitting | Despite achieving a high training accuracy of 87.86%, our current CNN-LSTM model still exhibits signs of overfitting, with a validation accuracy plateauing at 74%. To further improve generalization, we will explore replacing the LSTM layer with a GRU (Gated Recurrent Unit) layer while retaining the same number of input frames. This architectural modification may offer a better balance between capturing temporal dependencies and preventing overfitting. |
| 4 | CNN with GRU | 2573925 | Number of sample frames = 18  Batch size = 20  Number of epochs = 20 | Training accuracy: 0.92 Validation accuracy: 0.66  Indication of overfitting | The switch to a GRU-RNN architecture resulted in a slight decrease in training accuracy to 92.3% and a more significant drop in validation accuracy to 66%, suggesting that overfitting persists. To combat this, we will augment our existing data augmentation strategy by adding rotation transformations and re-evaluate the GRU-RNN model. We expect that the increased data diversity from rotation augmentation will improve the model's ability to generalize and reduce overfitting |
| 5 | CNN with LSTM (with modified Augmentation Technique) | 1657445 | Number of sample frames = 18 Batch size = 20 Number of epochs = 20 | Training accuracy: 0.88 Validation accuracy: 0.79  Slightly improved with augmentation | Augmenting our dataset with rotation transformations yielded a slight reduction in training accuracy to 88%, but more importantly, boosted the validation accuracy to 79%. Encouraged by this improvement, we will now investigate whether incorporating transfer learning can further enhance the model's performance, particularly on the validation set. |
| 6 | CNN with LSTM (with Transfer Learning) | 3840453 | Number of sample frames = 16  Batch size = 5  Number of epochs = 20 | Training accuracy: 0.98 Validation accuracy: 0.83  Indication of overfitting | While incorporating LSTM with transfer learning boosted training accuracy to 98%, the validation accuracy only reached 83%, indicating persistent overfitting. We hypothesize that this might be due to the MobileNet weights being frozen during training. To address this, we will re-evaluate the LSTM with transfer learning architecture, but this time allow the MobileNet weights to be trained alongside the rest of the model. This should enable the model to fine-tune the pre-trained features to better suit our specific gesture recognition task and potentially improve generalization. |
| 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: 0.9781 Validation accuracy: 0.97  Improved accuracy but model training was computationally expensive | Fine-tuning the MobileNet weights led to a significant jump in performance, with training accuracy reaching 97.81% and validation accuracy hitting 97%. However, this improvement came with a substantial increase in training time due to the larger number of parameters in the LSTM architecture. To optimize efficiency, we will now explore whether switching to a GRU-based architecture, while still leveraging the trainable MobileNet weights, can maintain or even improve these accuracies with a reduced parameter count and faster training time. |
| 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: 0.9992 Validation accuracy: 0.93  Finally model based on accuracy metric | The combination of a GRU layer and transfer learning, with training of the pre-trained weights, has yielded exceptional results. We achieved a training accuracy of 99.92% and, crucially, a validation accuracy of 93%. This significant improvement in validation accuracy indicates that overfitting has been effectively addressed, and the model generalizes well to unseen data. Therefore, we will select this GRU-based model with transfer learning as our final model for evaluation. |