Gesture Recognition Case Study

Presented by : Nihar Ranjan Nayak

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'.

|  |  |  |  |
| --- | --- | --- | --- |
| **Experiment Number** | **Model** | **Result** | **Decision + Explanation** |
| **1** | **Conv3D**  Simple model with CONV3D, MaxPooling3D, BatchNormalization and Dense Layers | **Train Accuracy: 42%**  **Validation Accuracy: 24%** | **Model is pretty inconsistent- the accuracy is fluctuating** |
| **2** | **TimeDistributed Conv2D** | **Train Accuracy: 24%**  **Validation Accuracy: 28%** | **The accuracy is low and decreasing- so model is underfitting** |
| **3** | **LSTM**  TimeDistributed with Conv2D, BatchNormalization, Dropout, Flatten | **Train Accuracy: 39%**  **Validation Accuracy: 24%** | **Only 5 epochs used for computational power limitations. looking at gap between validation and train accuracy- may be model is overfitting** |
| **4** | **GRU:**  TimeDistributed + Conv2D, BatchNormalization, BatchNormalization,Flatten, Dropout with GRU | **Train Accuracy: 49%**  **Validation Accuracy: 47%** | **Used only 5 epochs due to computational limitations**  **The model Seems fitting well- the training validation is close to 50%, whereas validation accuracy is around 30%- we try couple of more options to see if accuracy increases further** |
| **5** | **LSTM : more layers to model 3**  TimeDistributed + Conv2D, BatchNormalization, BatchNormalization,Flatten, Dropout with LSTM | **Train Accuracy: 56%**  **Validation Accuracy: 35%** | **Used only 5 epochs due to computational limitations**  **The training accuracy increased to 56%, but validation accuracy is only 35%. So model may be overfitting** |
| **6** | **Transfer Learning with GRU** | **Train Accuracy: 41%**  **Validation Accuracy: 47%** | **This model accuracy is not on par with the other GRU model- so my final model would be model 4** |