**Team Members: Ram Krishn Mishra, SaiRaz Kota**

**Problem Statement:**

Need to develop a cool feature in the smart-TV that can recognize five different gestures performed by the user which will help users control the TV without using a remote.

The gestures are continuously monitored by the webcam mounted on the TV. Each gesture corresponds to a specific command as below,

* Thumbs up:  Increase the volume
* Thumbs down: Decrease the volume
* Left swipe: 'Jump' backwards 10 seconds
* Right swipe: 'Jump' forward 10 seconds
* Stop: Pause the movie

**Solution Approach:**

After understanding and acquiring the dataset, the next step is to try out different architectures to solve this problem

1. Conv3D
2. CNN + RNN

**Implementation Steps:**

Step 1: Added required code in the starter file provide for image re-sizing, cropping, normalization and the generator function

Step 2: Implemented models based on Conv3D and CNN + RNN

Step 3: Experimented with the models by providing different batch sizes and epoch’s and observed the results

Step 4: Finalized the models based on the observed results from the previous step

**Observations**

**Model 1:**

**Based on Conv3D**

**Convolutions :**

**Apply 5 layer groups each with Convolutional layer with increasing order of filter size (standard size : 16, 32,64, 128, 256) and fixed kernel size = (3, 3, 3)**

**Apply Max Pooling layers, in each layer group after convolution. 5th layer add a zero padding**

**Flatten and Dense layers with 3 layers followed by 2 dropout after 1st two dense layers to avoid overfitting**

**Batch normalization on convolutional architecture after every convolution**

**Dense layers with 2 layers of sizes :256, 128 followed by 0.5 dropout )to avoid overfitting.**

**CNN-RNN**

**Based on CNN-RNN**

**Apply 10 Convolutional layer with increasing order of filter size (standard size : 32, 64, 128, 256, 512) and fixed kernel size = (3, 3)**

**Apply Max Pooling layer, after every 2nd convolutional layer (total 5)**

**Flatten and dropout of 0.5 followed by LSTM .**

**Please note : each model has a final softmax layer.**

**Results:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model Number** | **Model Description** | **Result** | **Batch Size / Number of Epoch** | **Decision + Explanation** |
| **Model 1** | **Conv3D** | **Accuracy:** 0.6600 | 10/10 | Resize to 120\*120, Raw image input, No cropping, No normalization, No augmentation, No flipped images, No edge detection |
| **Model 1** | **Conv3D** | **Accuracy:** 0.80000 | 20 / 20 | Resize to 120\*120, augmentation, No flipped images, No cropping, No normalisation, No edge detection |
| **Model 1** | **Conv3D** | **Accuracy:** 0.82500 | 30 / 30 | Resize to 120\*120, augmentation, flipped images, No normalization, No cropping, No edge detection |
| **CNN-RNN** | **Conv2D + LSTM** | **Accuracy:** 0.20 | 10/10 | Tried Conv2D + LSTM model as an alternate approach and observedLimited to only one model since the execution time was too high and frequent issues with nimble box |
| **Final Model** | **Conv3D** | **Accuracy: ~0.66 (10/10)**  **~0.80(20/20)**  **~0.82.5(50/30)** |  | Final model with batch size of 50 and epoch 30 gave 82.5 % accuracy |