Gesture Recognition Project

* Shivani Shrivastava
* Rishabh Vimal

# Problem Statement

As a data scientist working at a home electronics company which manufactures state of the art smart televisions. You want to develop a cool feature in the smart-TV that can recognise 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:

1. Thumbs up: Increase the volume
2. Thumbs down: Decrease the volume
3. Left swipe: 'Jump' backwards 10 seconds
4. Right swipe: 'Jump' forward 10 seconds
5. Stop: Pause the movie

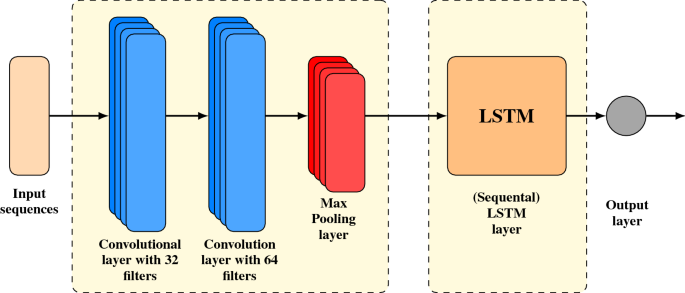
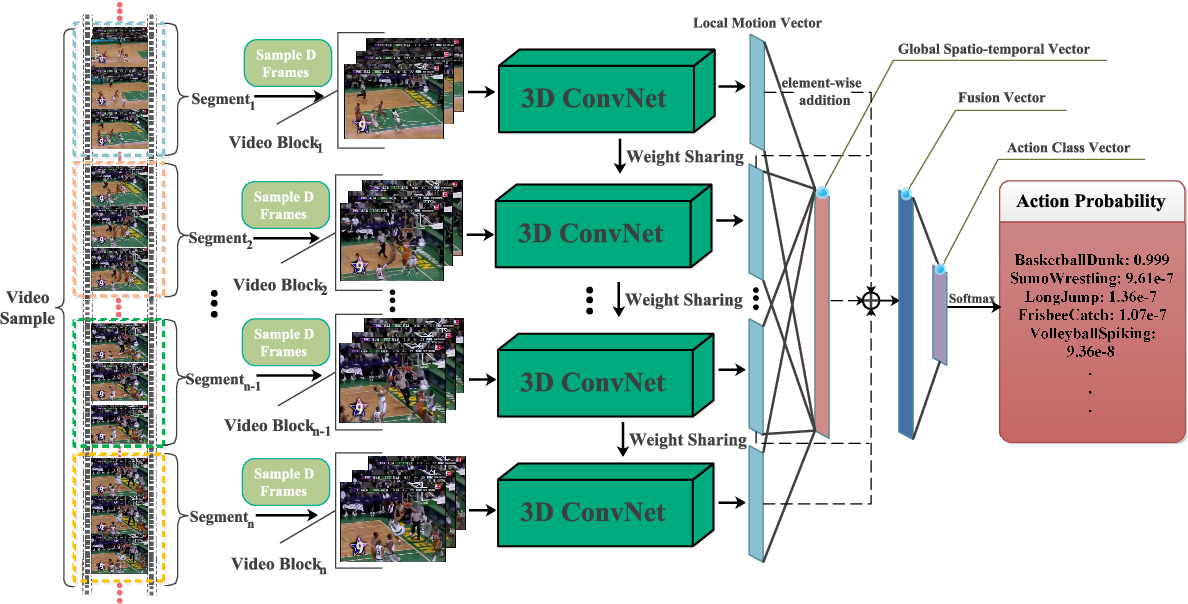
The training data consists of a few hundred videos categorised into one of the five classes. Each video (typically 2-3 seconds long) is divided into a sequence of 30 frames(images). These videos have been recorded by various people performing one of the five gestures in front of a webcam - similar to what the smart TV will use.

# Objective

Our goal is to use the 'train' folder to train multiple models that can anticipate the action taken in each sequence or video and also perform well on the 'val' folder. The final test folder will be reserved for review; the performance of the final model will be assessed on the 'test' set.

# Approach

For analysing videos using neural networks, two types of architectures are used commonly.

1. **CNN + RNN architecture** in which you pass the images of a video through a CNN which extracts a feature vector for each image, and then pass the sequence of these feature vectors through an RNN. ****
2. **3D Convolutional Neural Networks (Conv3D):** 3D convolutions are a natural extension to the 2D convolutions you are already familiar with. Just like in 2D conv, you move the filter in two directions (x and y), in 3D conv, you move the filter in three directions (x, y and z). In this case, the input to a 3D conv is a video (which is a sequence of 30 RGB images). If we assume that the shape of each image is 100x100x3, for example, the video becomes a 4-D tensor of shape 100x100x3x30 which can be written as (100x100x30)x3 where 3 is the number of channels. Hence, deriving the analogy from 2-D convolutions where a 2-D kernel/filter (a square filter) is represented as (fxf)xc where f is filter size and c is the number of channels, a 3-D kernel/filter (a 'cubic' filter) is represented as (fxfxf)xc (here c = 3 since the input images have three channels). This cubic filter will now '3D-convolve' on each of the three channels of the (100x100x30) tensor.****

# Data Generator

This is one of the most important part of the code. In the generator, we are going to pre-process the images as we have images of 2 different dimensions (360 x 360 and 120 x 160) as well as create a batch of video frames. The generator should be able to take a batch of videos as input without any error. Steps like cropping, resizing and normalization should be performed successfully.

# Data Pre-processing

* **Resizing of the images.** This was mainly done to ensure that the NN only recognizes the gestures effectively rather than focusing on the other background noise present in the image.
* **Normalization of the images.** Normalizing the RGB values of an image can at times be a simple and effective way to get rid of distortions caused by lights and shadows in an image.

# MODEL SUMMARY WITH Observations

|  |  |  |  |
| --- | --- | --- | --- |
| **Experiment Number** | **Model** | **Result** | **Decision + Explanation** |
| 1 | Conv3D  Batch size: 10; Epochs: 1 | Train Accuracy: 26%  Validation Accuracy: 16% | As our data is a collection of videos with 30 frames each using Conv3D as base model  First model that we tried was Batch size: 60; Epochs: 10 with which resource was exhausted hence started with lesser values but model didn’t learn much. |
| 2 | Conv3D  Batch size: 20; Epochs: 25 | Categorical accuracy: 90.37%  Val categorical accuracy: 31% | Tried decreasing batch size and increasing epochs.  It was observed that the model was overfitting. |
| 3 | Conv3D  Adding Dropout | Categorical accuracy: 55.11%  Val categorical accuracy: 29% | Added regularization to enhance model performance but it didn’t improve model much it’s still overfitting. |
| 4 | Conv3D+ LSTM | Categorical accuracy: 98.01%  Val categorical accuracy: 39% | Added LSTM as it could help with recurrent gestures. Difference between accuracies reduced but model is still overfitting. |
| 5 | Conv3D  + LSTM + Transfer Learning | Categorical accuracy: 99.33%  Val categorical accuracy: 72% | Added Transfer Learning as already trained model for a slightly different purpose could help and also reduce computational time.  Significant improvement in both accuracy and loss for train and test observed. |
| 6 | Conv3D + GRU +  Transfer Learning | Categorical accuracy: 98.92%  Val categorical accuracy: 93% | It’s known that LSTM and GRU give similar performance but GRU has much lesser computational time. Hence, tried GRU to improve model even further.  It is the best performing model among all of them.  Difference between categorical accuracy and validation accuracy is the least. Also, values for accuracies and losses are good. |

***Note: Please refer to the attached notebook for detailed observations and explanation.***