**Deep Learning Course Project - Gesture Recognition**

**Problem Statement**

Imagine you are working as a data scientist 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:

-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

Each video is a sequence of 30 frames (or images).

**Objective**

Our task is to train different models on the 'train' folder to predict the action performed in each sequence or video and which performs well on the 'val' folder as well.

The final model's performance will be tested on the 'test' set.

There are two types of architectures suggested for analysing videos using deep learning:

1. **3D Convolutional Neural Networks (Conv3D)**

3D convolutions are a natural extension to the 2D convolutions. In Conv2D, we move the filter in two directions (x and y). Similarly in Conv3D, we move the filter in three directions (x, y and z).

In this case, the input to a Conv3D is a video (which is a sequence of 30 RGB images). If we assume that the shape of each image is 100 x 100 x 3, for example, the video becomes a 4D tensor of shape 100 x 100 x 3 x 30 which can be written as (100 x 100 x 30) x 3 where 3 is the number of channels.

1. **CNN + RNN architecture**

The Conv2D network will extract a feature vector for each image, and a sequence of these feature vectors is then fed to an RNN-based network. The output of the RNN is a regular softmax (for a classification problem such as this one).

**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.

**Architecture Considerations**

* We have experimented with different model configurations and hyper-parameters and various iterations and combinations of batch sizes, image dimensions, filter sizes, padding and stride length were experimented with.
* We also played around with different learning rates and ReduceLROnPlateau was used to decrease the learning rate if the monitored metrics (val\_loss) remains unchanged in between epochs.
* We experimented with SGD() and Adam() optimizers but went forward with Adam() as it lead to improvement in model’s accuracy by rectifying high variance in the model’s parameters.
* We also made use of Batch Normalization, pooling and dropout layers when our model started to overfit, this could be easily witnessed when our model started giving poor validation accuracy in spite of having good training accuracy ☺.
* Early stopping was used to put a halt at the training process when the val\_loss would start to saturate / model’s performance would stop improving.
* For execution, ran the code on Kaggle with GPU as well as tried Google Colab as the platform for execution.

**Final Summary**

****