**Group Case Study: Hand Gesture Recognition Project**

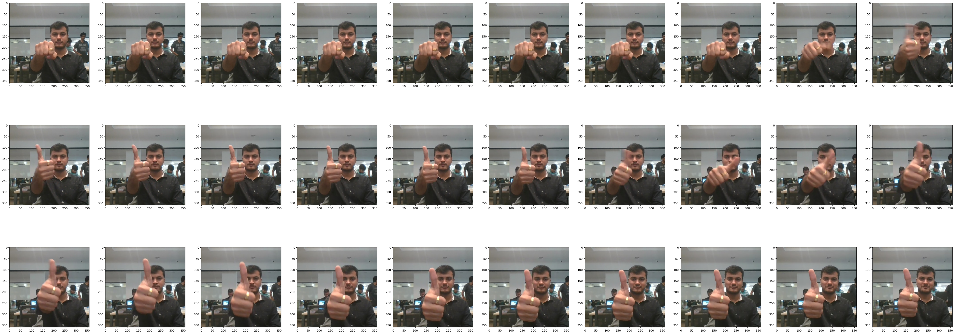
* **Problem Statement**

You are working as a data scientist at a home electronics company which manufactures state of the art smart televisions.

Your goal is to create an innovative feature for the smart TV platform. This feature aims to accurately identify and interpret five distinct gestures executed by users, facilitating television control without the need for a traditional 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
* **Understanding the Dataset**
* The training data consists of a few hundred videos categorized into one of the five classes. Each video (typically 2-3 seconds long) is divided into a sequence of 30 frames(images).
* The videos have two types of dimensions - either 360x360 or 120x160 (depending on the webcam used to record the videos).
* Link to the dataset: <https://drive.google.com/uc?id=1ehyrYBQ5rbQQe6yL4XbLWe3FMvuVUGiL>
* Sample Images from “Thumbs Up” folder:



* **Objective of the Case Study**

Our objective is to train various models using the data from the **'train'** directory to predict the action depicted in each sequence or video, ensuring robust performance on the **'val'** dataset. The 'test' folder, withheld for evaluation, will serve as the final benchmark to assess the performance of the model.

Two recommended architectural approaches proposed for video analysis through deep learning:

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

3D convolutions extend the principles of 2D convolutions you're familiar with. Similar to 2D convolutions operating in the x and y directions, 3D convolutions add the z direction for movement. In our case, 3D convolutions process video inputs, typically sequences of 30 RGB images. If each image is, for instance, 100 x 100 x 3, the video forms a 4D tensor of shape 100 x 100 x 3 x 30, which can be expressed as (100 x 100 x 30) x 3, where 3 signifies the channels. Drawing from 2D convolutions, where a 2D filter is (f x f) x c (f is filter size and c is channels), a 3D filter is (f x f x f) x c, with c as 3 due to the three input channels. This cubic filter convolves across each of the three channels of the (100 x 100 x 30) tensor.

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 case study).

An additional benefit is the potential utilization of **transfer learning** in this context. Given the availability of state-of-the-art networks designed for image classification tasks such as ResNet or VGGNet, we can leverage their pre-trained weights. Subsequently, we can employ these networks to transform the input images, obtaining image representations such as the output of a dense layer.

* **Data Generator**

This section of the code holds significant importance. Within the generator, we undertake preprocessing of images, accommodating two different dimensions (360 x 360 and 120 x 160), while also constructing batches of video frames. The generator must seamlessly handle batches of videos as input, ensuring successful execution of tasks like cropping, resizing, and normalization.

* **Data Pre-Processing**
* **Resizing and cropping of the images:**

The primary purpose behind this was to ensure that the neural network effectively identifies the gestures without being influenced by extraneous background noise within the image.

* **Normalization of the images:**

Sometimes, normalizing the RGB values of an image can be a quick and useful way to minimize the effects/distortions of light and shadow issues in the image.

* **Modelling Results**

Kindly look at the attached spreadsheet containing a detailed list of experiments conducted to determine the optimal model for the specified problem statement.



* **Selected Model**

The final model selected is **Model #11 (mobileNetModelWeight)** which has a training accuracy of 99% and validation accuracy of 99% as well.

**File Name:** model-00013-0.02487-0.99548-0.03481-0.99000.h5