## **Deep Learning Course Project- Gesture Recognition**

# Problem Statement

As a data scientist at a home electronics company which manufactures state of the art smart televisions. We 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.

* 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 and Objective

The training dataset is a collection of various short videos for the above-mentioned gestures at 30fps.

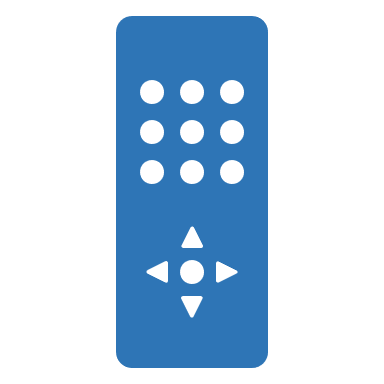
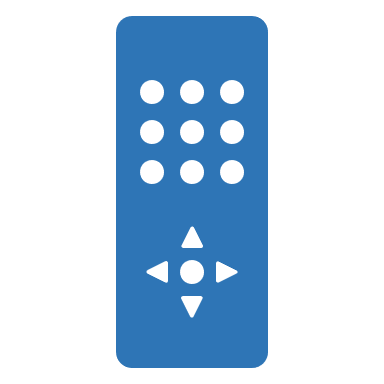
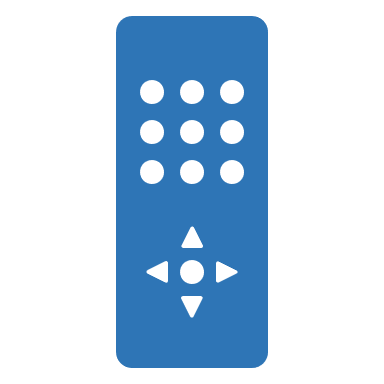
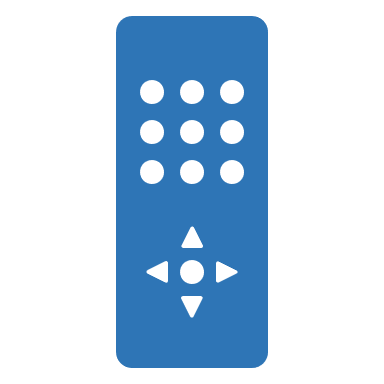
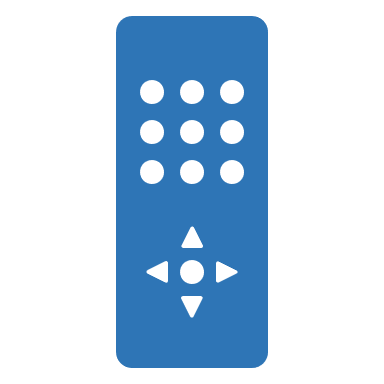
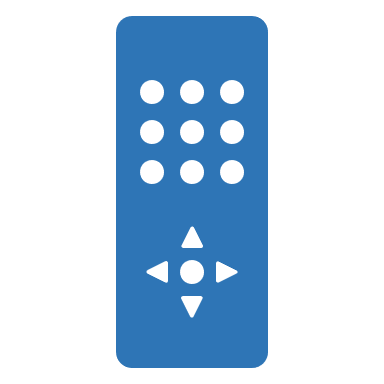
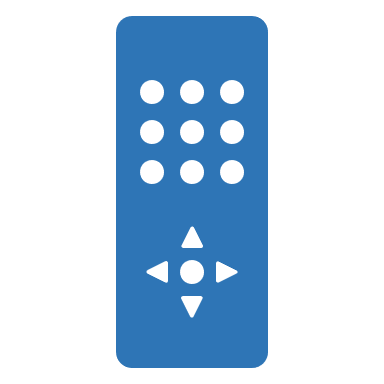
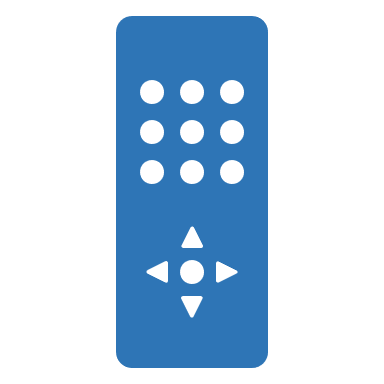
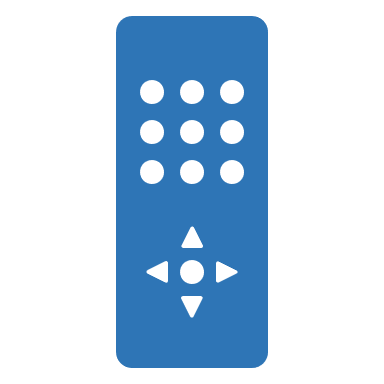
 Our target is to train a model using each video in the train folder that can be used to teach the smart appliance to recognise the basic gesture. Then finally to test the model on test folder for evaluation of the models performance

# For analysis we will be using 3D Convulational structure of deep learning:

1. **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 *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. Hence, deriving the analogy from 2D convolutions where a 2D kernel/filter (a square filter) is represented as *(f x f) x c* where *f* is filter size and *c* is the number of channels, a 3D kernel/filter (a *'cubic'* filter) is represented as *(f x f x f) x c* (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 *(100 x 100 x 30)* tensor

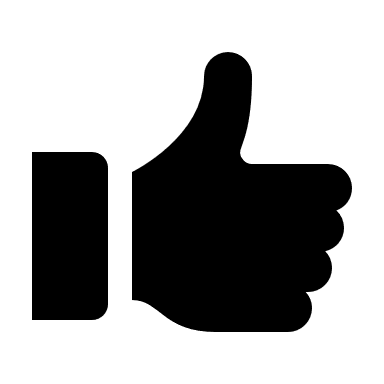
.

****

**30 frames….**

**Depth**

**Error**

****

**Conv3D**

**Back**

**Propagation**

**Gestures**

***e****.g****.*** *(100 x 100 x 3 x 30)*

**Update**

**Figure 1: A simple representation of working of a 3D-CNN**

**Data Generator**

Data generator is used to pre-process the images, Dimensions of images that we have are not same (*360 x 360* and *120 x 160*) then we create the 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 to be performed successfully.

# Neural Network Architecture development and training

* We used 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 used *Adam()* as optimiser 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 inspite of having good training accuracy.
* In the model in notebook it can be seen that the model have used all the above mentioned features to obtain
  + Training Accuracy-74.18%
  + Validation Accuracy- 70%

# Observations

* It was observed that as the Number of trainable parameters increase, the model takes much more time for training.
* Increasing the batch size reduces the training time as it takes max amount of data into consideration but the negative impact this has on model accuracy. There is always a trade-off here between time and accuracy. For faster model and result we use large batch size, else we choose lower batch size if we want our model to be more accurate.

After various hit and trial with models, we selected the final model

**Reason:**

* (Training Accuracy: 83.99%, Validation Accuracy: 80%)
* Number of Parameters (8,64,101) less according to other models’ performance

# Further suggestions for improvement:

* **Deeper Understanding of Data:** The video clips where recorded in different backgrounds, lightings, persons and different cameras where used. Further exploration on the available images could give some more information about them and bring more diversity in the dataset. This added information can be exploited in favour inside the generator function adding more stability and accuracy to model.