# GESTURE RECOGNITION CASE STUDY
## - Anmol Mehta

**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 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:
• 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). These videos have been recorded by various people performing one of the five gestures in front of a webcam - like what the smart TV will use. Task is to train a model on the 'train' folder which performs well on the 'val' folder as well (as usually done in ML projects). We have withheld the test folder for evaluation purposes - your final model's performance will be tested on the 'test' set.
**Approach:**
Three architectures used: For analyzing videos using neural networks, 3 types of architectures are used commonly.

1. **3D Convolutional Network, or Conv3D**

3D convolutions are a natural extension to the 2D convolutions. For video data, each data point is IID, but in each datapoint images are in sequence. We perform Conv2D on images, here we have temporal dimension as the third dimension. Just like in 2D conv, we move the filter in two directions (x and y), in 3D conv, we move the filter in three directions (x, y and z). So, for 3D conv cubic or 3-D kernals are used.
Considering a video of 30FPS (Frame per seconds). One second of the video can be represented as 30 sequences of images. If each image or frame is of shape 100x100x3 (3 is no. of channels), then the video becomes a 4-D tensor of shape 100x100x3x30 which can be written as (30x100x100)x3 where 3 is the number of channels and 30 is no. of sequences in each datapoint. This is the input tensor shape of Conv 3D network. Cubic filters are used to convolve and create the feature maps. There are 3D Pooling layers to reduce the dimension. Followed by flattening and then it goes to a Feed Forward network with softmax activation at outer layer for classification scenarios.

2. **Convolutions 2D + RNN**

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).

3. **Conv2D + GRU**

**Data Generator Function**

We have created custom data generator function. It works like a generator object. It takes batch size as input and instead of loading all training datapoints at once, it reads datapoints based on the value of the batch size and then performs required image pre-processing and yields data for training. We have performed required resizing of images. As we have different image sizes in dataset, we resized all images to 100x100 with 3 channels. Performed Normalizations for each channel separately using min-max scaling. Tested the data generator function to check if it yields the data properly.

**Conclusion:**

We have performed experiments with different types of architectures. For Conv2D+ RNN types of models, we have explored different transfer learnings and fine-tuned the pre-trained weights to extract features and then fed it to RNN-based networks like LSTM, RNN. This type of architectures have lot of model parameters and it took good amount of time to train these kinds of models.

We explored multiple models and got best validation accuracy of .70-.85 using Conv3D Model. The model has 864,101 parameters.
We also tried exploring different network architectures of Conv2D + RNN & Conv2D + GRU. Training time required for these models are lower than Conv3D RNN type models. Explored Adam, Nadam, RMSprop as optimizers in different experiments. Adam and Nadam gave faster convergence and better validation accuracy.

Conv3D is considered as our final model.