**Gesture Recognition Project**

# Problem Statement

Need 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

# About Dataset

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.

The data is in a zip file. The zip file contains a 'train' and a 'val' folder with two CSV files for the two folders. These folders are in turn divided into subfolders where each subfolder represents a video of a particular gesture. Each subfolder, i.e. a video, contains 30 frames (or images). Note that all images in a particular video subfolder have the same dimensions but different videos may have different dimensions. Specifically, videos have two types of dimensions - either 360x360 or 120x160 (depending on the webcam used to record the videos).

Each row of the CSV file represents one video and contains three main pieces of information - the name of the subfolder containing the 30 images of the video, the name of the gesture and the numeric label (between 0-4) of the video.

# Objective

Task is to train a model on the 'train' folder which performs well on the 'val' folder as well.

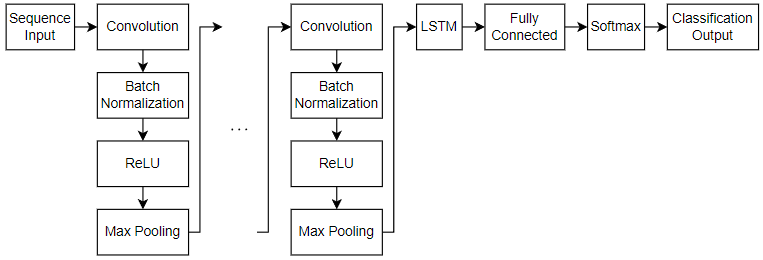
# Approach

Two types of architectures are used commonly in these type of usecases

1. Standard 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. Other popular architecture used to process videos is a natural extension of CNNs - a 3D convolutional network.

# Standard CNN + RNNT

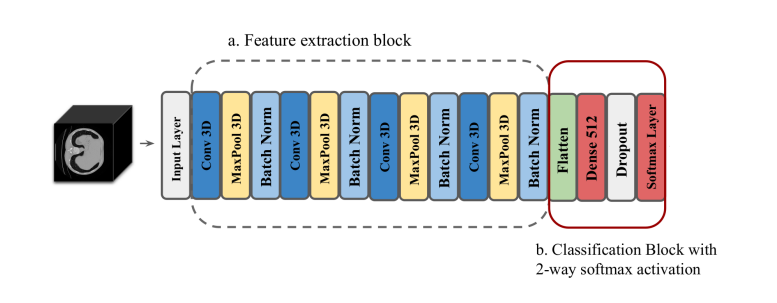
Typical architecture for Standard CNN +RNN looks like below ::



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

# CNN – 3D Conv Network

Typical architecture for 3D Conv network looks like below ::



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)

# Data ingestion pipeline

In most deep learning projects you need to feed data to the model in batches. This is done using the concept of generators

# Data Preperation

In this stage, image data will go through the below activities before getting as a feed to network.

1. cropping the images
2. resizing the images
3. normalizing the images

# Observations

In this activity, different models got evaluated using two approaches. First approach was using Conv3D and in second approach it was using CNN +RNN where first few models where tried using LSTM or GRU. Later used transfer learning concept with LSTM or GRU.

Here is the summary of different experiment’s carried out in this exercise.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Architecture** | **Model Name** | **Epochs** | **Batch Size** | **Sample Frame** | **Model Parameters** | **Training Accuracy** | **Validation Accuracy** | **Notes** |
| Conv3D | model\_arch1\_conv3D\_1 | 20 | 64 | 30 | Total params: 1,933,765 (7.38 MB)  Trainable params: 1,932,517 (7.37 MB)  Non-trainable params: 1,248 (4.88 KB) | 100 | 19 |  |
| Conv3D | model\_arch1\_conv3D\_2 | 20 | 32 | 20 | Total params: 1,933,765 (7.38 MB)  Trainable params: 1,932,517 (7.37 MB)  Non-trainable params: 1,248 (4.88 KB) | 99 | 33 |  |
| Conv3D | model\_arch1\_conv3D\_3 | 20 | 8 | 20 | Total params: 1,933,765 (7.38 MB)  Trainable params: 1,932,517 (7.37 MB)  Non-trainable params: 1,248 (4.88 KB) | 93 | 84 |  |
| Conv3D | model\_arch1\_conv3D\_4 | 20 | 16 | 20 | Total params: 4,446,853 (16.96 MB)  Trainable params: 4,444,869 (16.96 MB)  Non-trainable params: 1,984 (7.75 KB) | 91 | 88 |  |
| CNN + RNN with LTSM | model\_arch2\_cnn\_rnn\_1 | 20 | 64 | 30 | Total params: 1,052,773 (4.02 MB)  Trainable params: 1,051,013 (4.01 MB)  Non-trainable params: 1,760 (6.88 KB) |  |  | ##### OOM Error got triggered as GPU is running out of memory. |
| CNN + RNN with LTSM | model\_arch2\_cnn\_rnn\_2 | 20 | 16 | 30 | Total params: 1,052,773 (4.02 MB)  Trainable params: 1,051,013 (4.01 MB)  Non-trainable params: 1,760 (6.88 KB) | 91 | 71 |  |
| CNN + RNN with LTSM | model\_arch2\_cnn\_rnn\_3 | 20 | 16 | 20 | Total params: 1,052,773 (4.02 MB)  Trainable params: 1,051,013 (4.01 MB)  Non-trainable params: 1,760 (6.88 KB) | 66 | 60 |  |
| CNN + RNN with GRU | model\_arch2\_cnn\_rnn\_4 | 20 | 16 | 20 | Total params: 901,349 (3.44 MB)  Trainable params: 899,589 (3.43 MB)  Non-trainable params: 1,760 (6.88 KB) | 88 | 77 |  |
| Transfer Learning using MobileNet + LTSM + Training of Internal layers set to False | model\_arch2\_cnn\_rnn\_trslrn\_1 | 20 | 16 | 20 | Total params: 3,529,669 (13.46 MB)  Trainable params: 298,757 (1.14 MB)  Non-trainable params: 3,230,912 (12.32 MB) | 99 | 79 |  |
| Transfer Learning using MobileNet + LTSM + Training of Internal layers set to True | model\_arch2\_cnn\_rnn\_trslrn\_2 | 20 | 16 | 20 | Total params: 3,529,669 (13.46 MB)  Trainable params: 3,505,733 (13.37 MB)  Non-trainable params: 23,936 (93.50 KB) | 97 | 94 |  |
| Transfer Learning using MobileNet + GRU + Training of Internal layers set to False | model\_arch2\_cnn\_rnn\_trslrn\_3 | 20 | 16 | 20 | Total params: 3,460,165 (13.20 MB)  Trainable params: 229,253 (895.52 KB)  Non-trainable params: 3,230,912 (12.32 MB) | 99 | 71 |  |
| Transfer Learning using MobileNet + GRU + Training of Internal layers set to True | model\_arch2\_cnn\_rnn\_trslrn\_4 | 20 | 16 | 20 | Total params: 3,460,165 (13.20 MB)  Trainable params: 3,436,229 (13.11 MB)  Non-trainable params: 23,936 (93.50 KB) | 99 | 93 | Based on the scores choosen this model |