**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

**Data Provided:**

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). Hence, you will need to do some pre-processing to standardise 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.

**Task:**

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.

**Procedure:**

1. Create a generator function to read the train and test csv files.
2. Create randomly selected feeds from the dataset. We will shuffle the records in the files to avoid any overfitting and make it more generic. Train and Test csv files has own labelled videos defines in the files.
3. As stated in the Data provided section, we can see that images are there in two dimensions : 360x360 0r 120x160. We have to make all the images in same size. To make it we will reduce the size of 360x360 to 120x120 and crop the 120x160 to 120x120 to get the middle portion of the videos.
4. Images are normalized for each channel dividing by 255
5. There is a generator function which will create the batches for the assigned batch size and a smaller batch will be created for the remaining videos.
6. **Model Creation :**

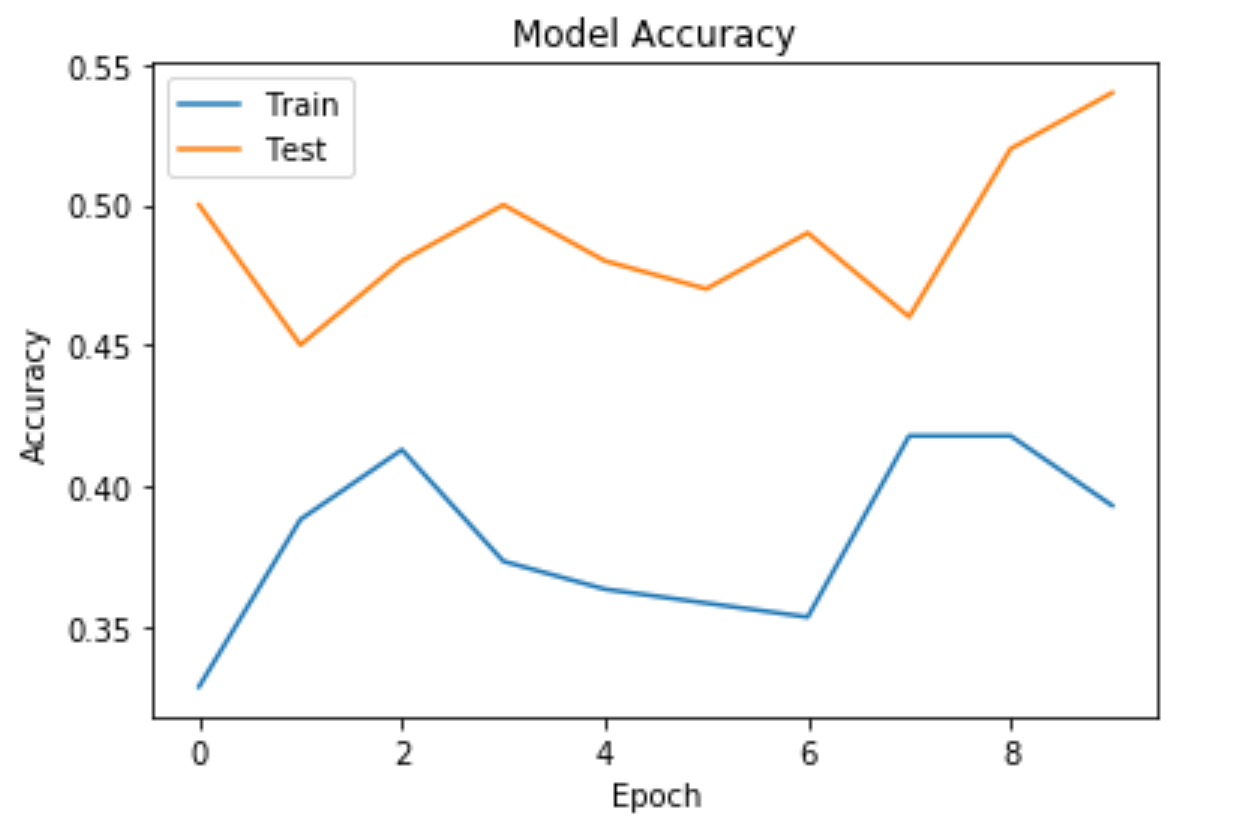
Various models are created and experimented with different batch, epoch and different image indexes. These tare described below in details woth their outouts:

1. **Model 1**
2. batch\_size = 10
3. num\_epochs = 10
4. img\_idx = [1,3,5,7,9,11,13,15,17,19]

With the model code as written below :

|  |
| --- |
| input\_shape=(nb\_frames,nb\_rows,nb\_cols,nb\_channel)  model=Sequential()  model.add(Conv3D(8,  kernel\_size=(3,3,3),  input\_shape=input\_shape,  padding='same'))  model.add(BatchNormalization())  model.add(Activation('relu'))  model.add(MaxPooling3D(pool\_size=(2,2,2)))  model.add(Conv3D(16,  kernel\_size=(3,3,3),  padding='same'))  model.add(BatchNormalization())  model.add(Activation('relu'))  model.add(MaxPooling3D(pool\_size=(2,2,2)))  model.add(Conv3D(32,  kernel\_size=(3,3,3),  padding='same'))  model.add(BatchNormalization())  model.add(Activation('relu'))  model.add(Dropout(0.25))  model.add(MaxPooling3D(pool\_size=(2,2,2)))  #Flatten Layers  model.add(Flatten())  model.add(Dense(128, activation='relu'))  model.add(Dropout(0.25))  model.add(Dense(64, activation='relu'))  model.add(Dropout(0.25))  #softmax layer  model.add(Dense(5, activation='softmax')) |

The history curve of this model training we got as below :

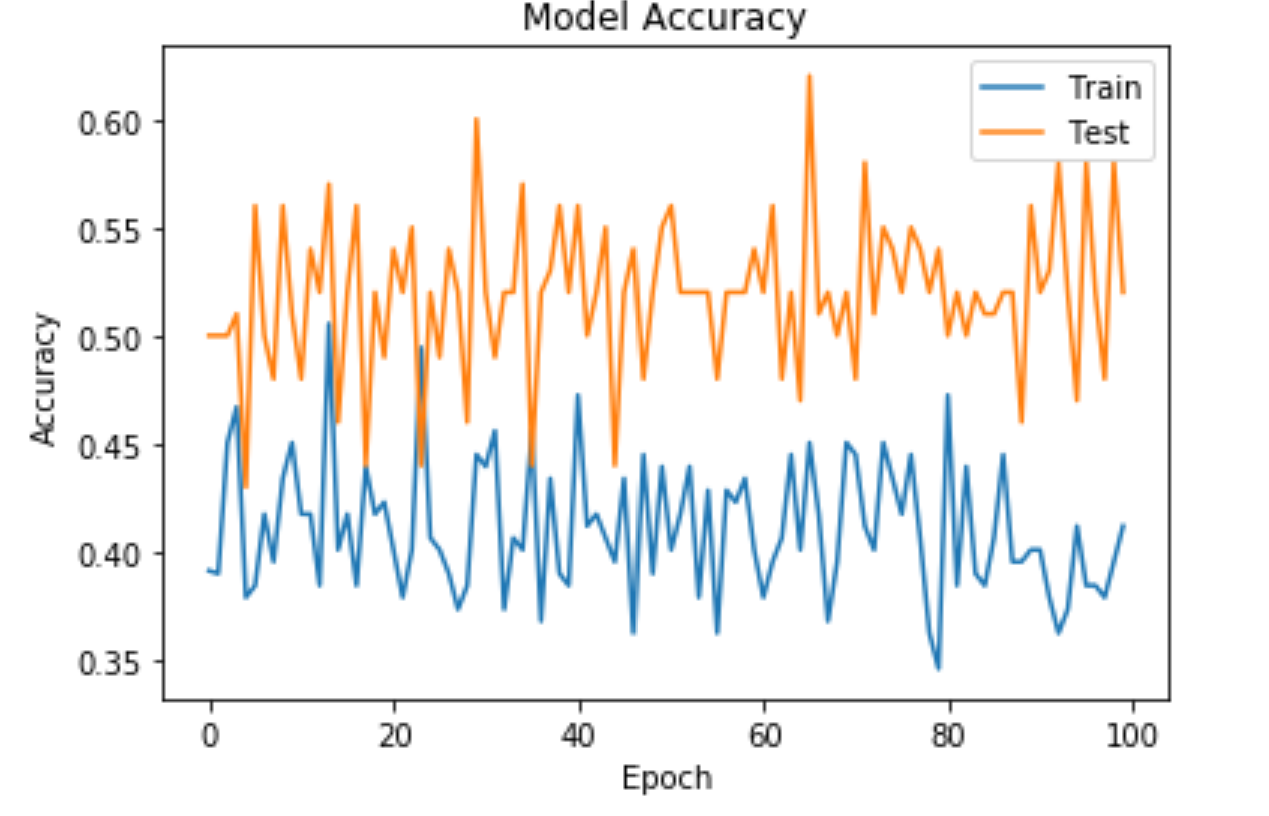


1. **Model 2**

Now same model we are executing with below parameters :

1. batch\_size = 50
2. num\_epochs = 100
3. img\_idx = [0, 3, 6, 9, 12, 15, 18, 21, 24, 27]

And we got the below history curve:



1. **Model 3**

Now same model we are executing with below parameters :

1. batch\_size = 30
2. num\_epochs = 100
3. img\_idx = [0, 3, 6, 9, 12, 15, 18, 21, 24, 27]

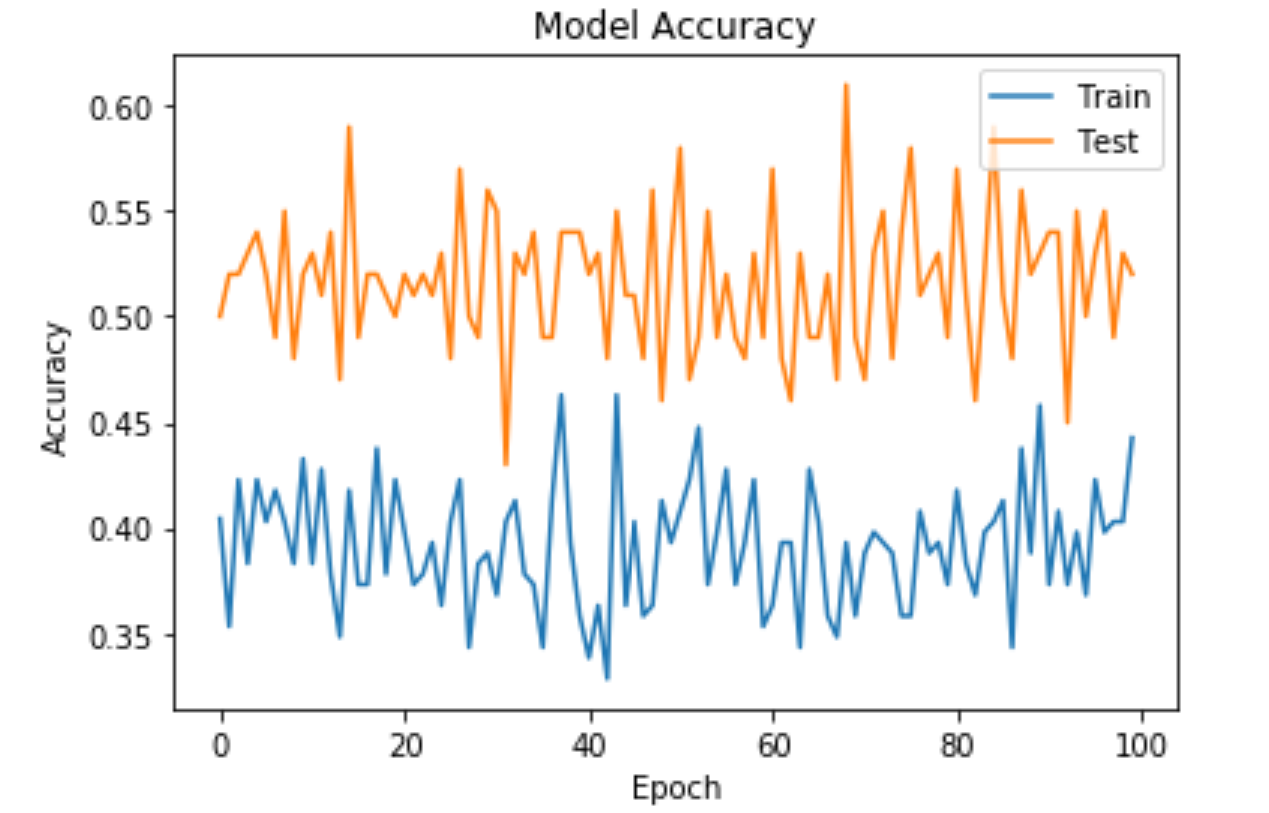
And we got the below history curve:

1. **Model 4**

Now same model we are executing with below parameters :

1. batch\_size = 10
2. num\_epochs = 100
3. img\_idx = [0-30]

And we got the below history curve:

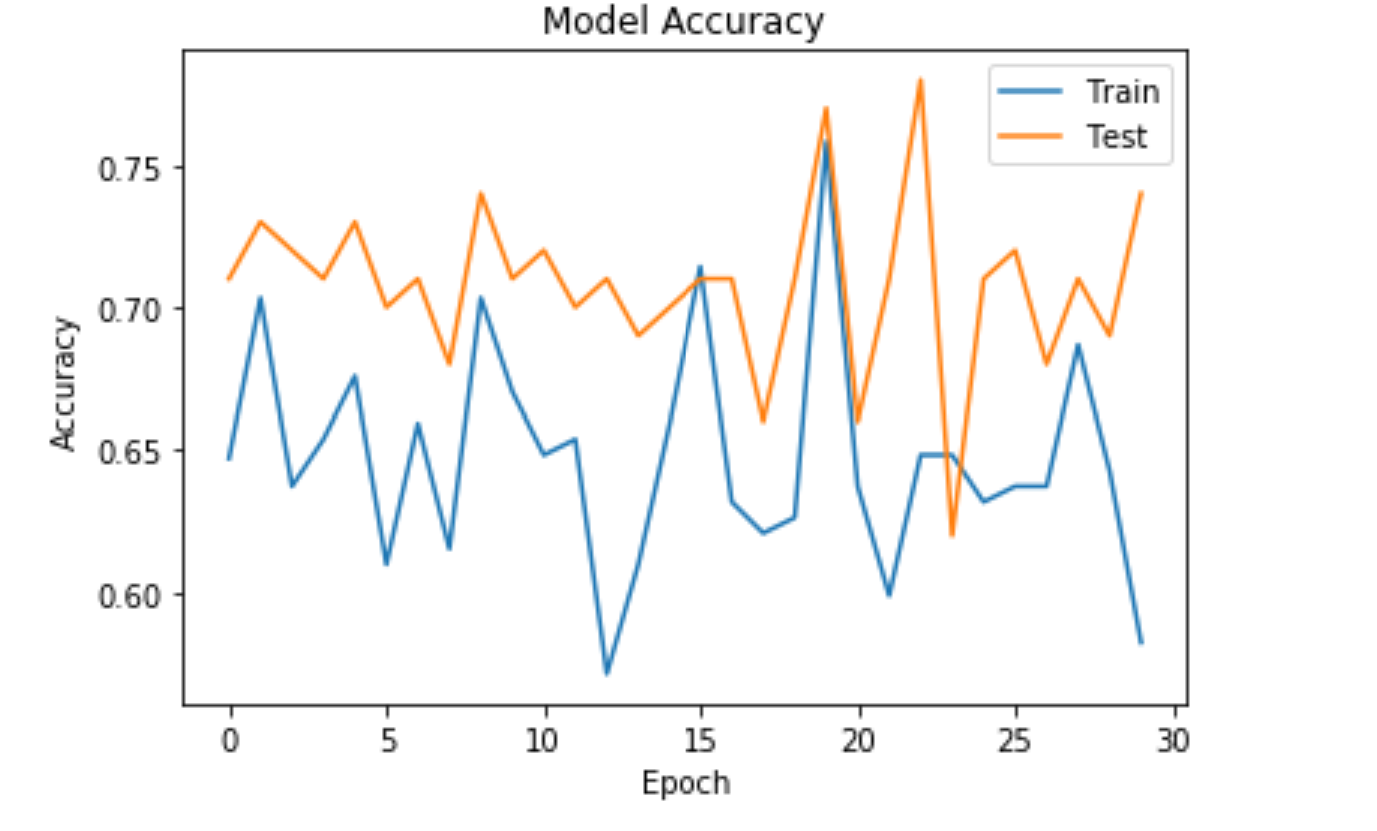


1. **Model 5**

Now same model we are executing with below parameters :

1. batch\_size = 50
2. num\_epochs = 30
3. img\_idx = [0-30]

And we got the below history curve:



1. **Model 6**

Now same model we are executing with below parameters :

1. batch\_size = 50
2. num\_epochs = 100
3. img\_idx = [0-30]

And we got the below history curve:

