**Gesture Recognition Project**

**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

**Understanding the 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 - like what the smart TV will use.

I have done various experiments to build a model to predict the gestures, and here is summary of them:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | Model | Layers | Batch Size | Accuracy | Outcome |
| 1. | Conv3D | Conv3D(8), Maxpool3D, Flatten, Dense(16), Dense(5), | 150 | # | out of Memory Error.  Need to reduce batch size so that all data is not loaded at once |
| 2. | Conv3D | Conv3D(8), Maxpool3D, Flatten, Dense(16), Dense(5), | 50 | Train = 20.34 %  Val = 99.0 % | Generator had issue. Indentation on yield statement was wrong. Method was returning incorrect data. |
| 3. | Conv3D | Conv3D(8), Maxpool3D, Flatten, Dense(16), Dense(5) | 50 | Train = 23.67%  Val = 0.0001 % (almost 0) | Need to add more layers. |
| 4. | Conv3D | 3 sets of (Conv3D, Max Poling and Batch Normalization)  Flatten layer  2 sets of dense layers | 50 | Train = 99%  Val = 60% | Overfitting  Need some dropout layers  Need to drop out some video frames |
| **5.** | **Conv3D** | **Added few dropouts in above #4**  **has 1.8 million params** | **Batch size 50, with alternate video frames (15 out of 30)** | **Train = 99%**  **Val = 86%** | **Possibly still overfitting**  **But this is best accuracy I have achieved.**  **Adding more layers was increasing training time. This model tool hours to run.** |
| 6. | Conv2D + RNN | Time Distributed layers Conv2D, MapPool2D, Flatten, LSTM, Dense | 50 | Train = 23.67%  Val = 0.0001 % (almost 0) | Model not trained enough, need more layers |
| 7. | Conv2D + RNN | 3 sets of (Time Distributed Conv3D, Max Poling and Batch Normalization)  Flatten, LSTM  2 sets of dense layers | 50 | Train = 99%  Val = 45% | Overfitting  Need some dropout layers  Need to drop out some video frames |
| 8. | Conv2D + RNN | Added dropouts in above #7  Has 15 million params | Batch size 50, with alternate video frames (15 out of 30) | Train = 91%  Val = 78% | Still overfitting |

**Learnings:**

1. Generator function yield method is a great way to control data going into the model. The way my generator function works is as below:

e.g., Total folders to read = 663, batch size = 50, So

for 1st batch - it skips 0 records and takes next 50 records

for 2nd batch – it skips 50 records and takes next 50 records.

..

for 13th batch – it skips 600 records and takes next 50 records.

for last 14th batch – it skips 650 records and takes next 13 records.

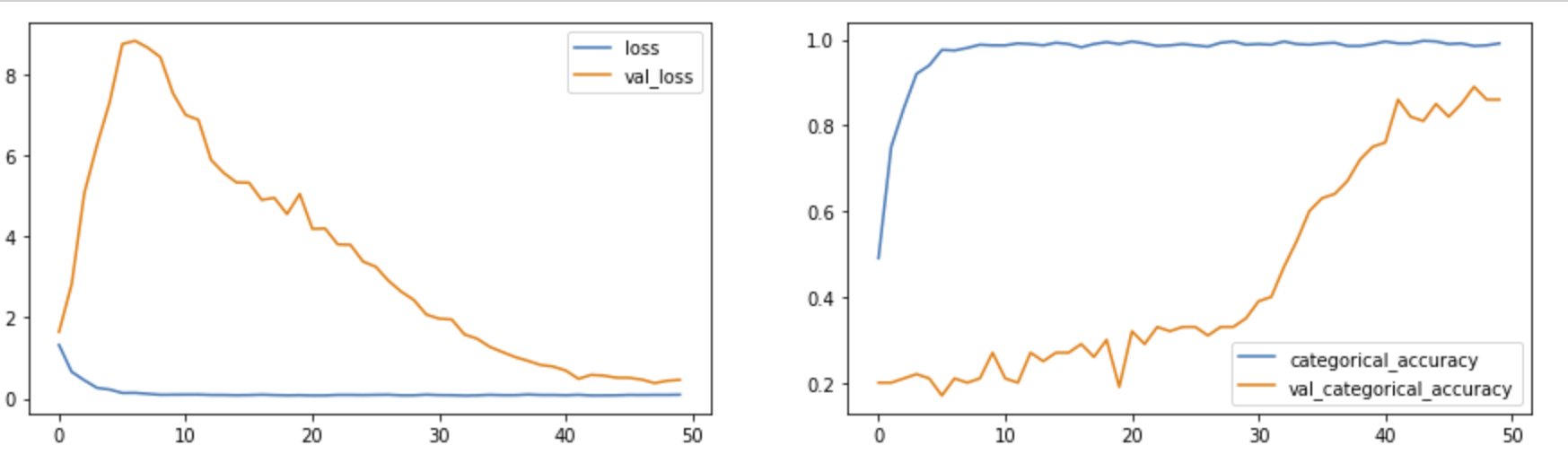
steps per epoch is also calculating this total number of batches to pass on to fit generator method.

1. Number of video frames we pass on to the model has big impact. It reduces the processing time and keeps accuracy almost same. 30 frames in a video have lot of repeated information, so some frames can be skipped.
2. Batch size directly affects memory usage. To work on a system with low memory we can rely on small batch size.
3. Augmentation like flipping/ mirroring not applied to this dataset as images orientation mean everything for this training. Left swipe right swipe are actual features we want to have differentia in outcome.

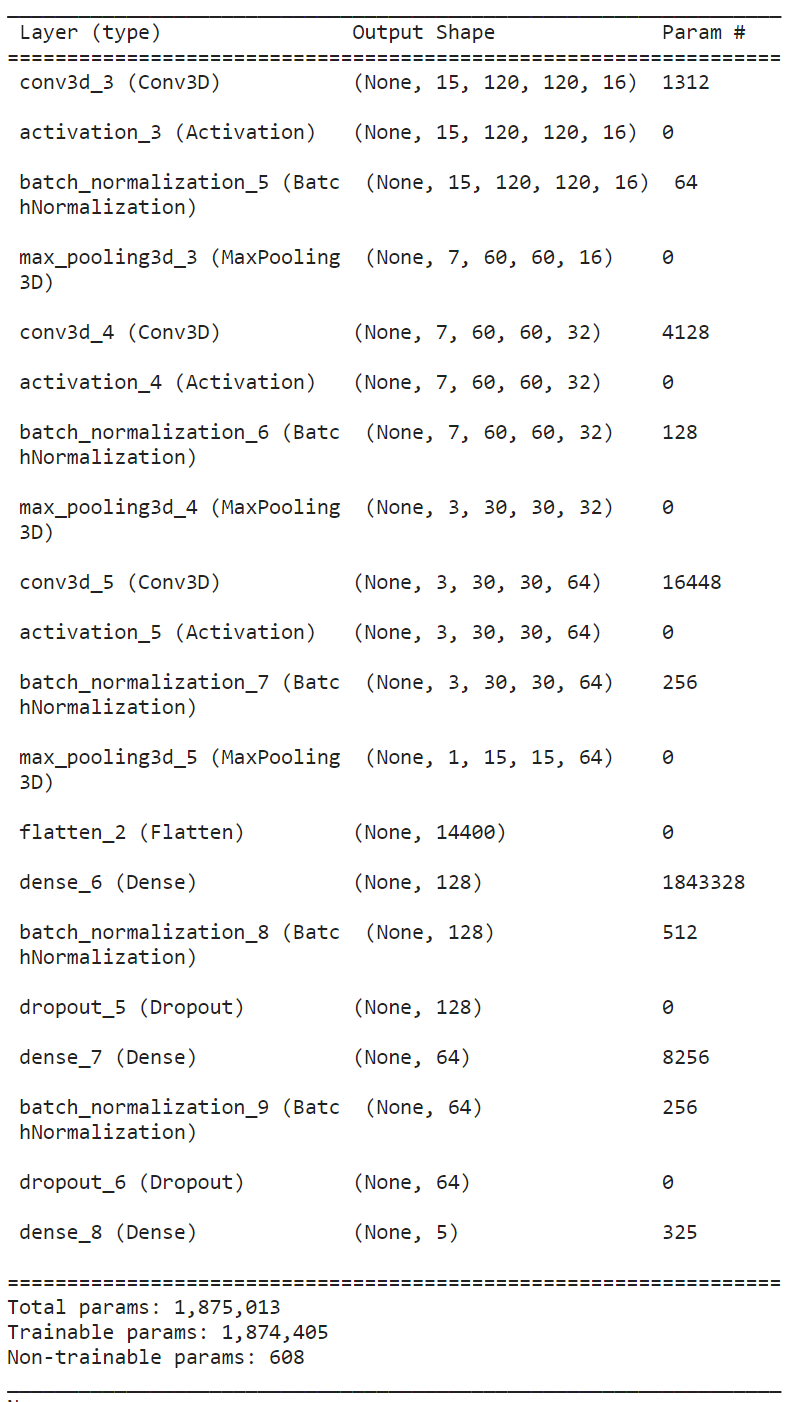
**Submission Model**

**Model #5 has highest accuracy in my experiments and h5 file is submitted for the same model.**

Achieved good accuracy with Conv3D model with 1.8 million params Training = 99%, Validation = 86%. Loss reduced with each epoch. And accuracy also improved with each epoch.



**Model Layers:**



**Last epoch result –**

Possibly will get more accuracy if I have trained this for more epochs as accuracy was improving consistently.

