Gesture Recognition Project – Vaibhav Jain

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 - similar to what the smart TV will use.

I have done various experiments to build a model to predict the gestures, and here is summry 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	TimeDistributed 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 (TimeDistributed 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:

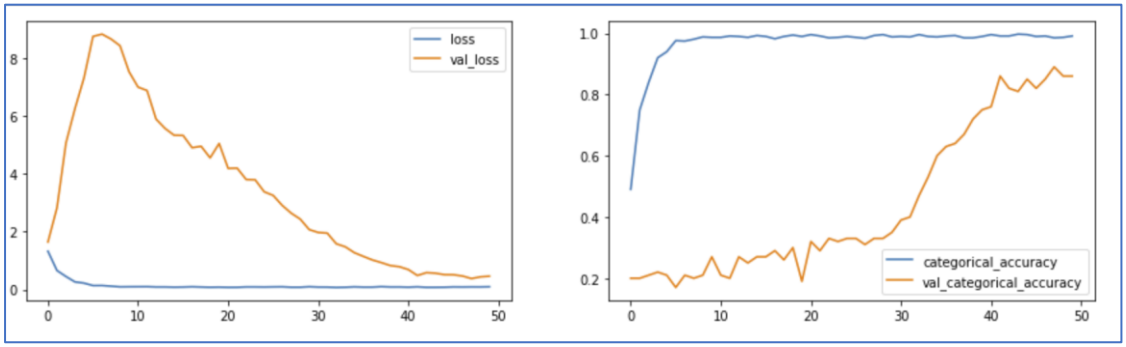
1. Generator function yeild method is a great way to control data going into the model. The way my generator function works is as below:
   e.g. Total folders to read = 663, batch size = 50, So
   for 1st batch - it skips 0 records and takes next 50 records
   for 2nd batch – it skips 50 records and 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.
2. 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 has lot of repeated information, so some frames can be skipped.
3. Batch size directly affects memory usage. To work on a system with low memory we can rely on small batch size.
4. Augumentaion 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 differention 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:

```
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 conv3d_3 (Conv3D)            (None, 15, 120, 120, 16)  1312

 activation_3 (Activation)    (None, 15, 120, 120, 16)  0

 batch_normalization_5 (Batc  (None, 15, 120, 120, 16)  64
 hNormalization)

 max_pooling3d_3 (MaxPooling  (None, 7, 60, 60, 16)     0
 3D)

 conv3d_4 (Conv3D)            (None, 7, 60, 60, 32)     4128

 activation_4 (Activation)    (None, 7, 60, 60, 32)     0

 batch_normalization_6 (Batc  (None, 7, 60, 60, 32)     128
 hNormalization)

 max_pooling3d_4 (MaxPooling  (None, 3, 30, 30, 32)     0
 3D)

 conv3d_5 (Conv3D)            (None, 3, 30, 30, 64)     16448

 activation_5 (Activation)    (None, 3, 30, 30, 64)     0

 batch_normalization_7 (Batc  (None, 3, 30, 30, 64)     256
 hNormalization)

 max_pooling3d_5 (MaxPooling  (None, 1, 15, 15, 64)     0
 3D)

 flatten_2 (Flatten)          (None, 14400)             0

 dense_6 (Dense)              (None, 128)               1843328

 batch_normalization_8 (Batc  (None, 128)               512
 hNormalization)

 dropout_5 (Dropout)          (None, 128)               0

 dense_7 (Dense)              (None, 64)                8256

 batch_normalization_9 (Batc  (None, 64)                256
 hNormalization)

 dropout_6 (Dropout)          (None, 64)                0

 dense_8 (Dense)              (None, 5)                 325

=================================================================
Total params: 1,875,013
Trainable params: 1,874,405
Non-trainable params: 608
_________________________________________________________________
```

Last epoch result –

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

```
Epoch 50/50
14/14 [==============================] - ETA: 0s - loss: 0.0956 - categorical_accuracy: 0.9910
Epoch 00050: saving model to Conv3D_1_init_2022-11-1516_59_51.635879/model-00050-0.09562-0.99095-0.46140-0.86000.h5
14/14 [==============================] - 79s 6s/step - loss: 0.0956 - categorical_accuracy: 0.9910 - val_loss: 0.4614 - val_categorical_accuracy: 0.8600 - lr: 2.5600e-09
```