## **Gesture Recognition Case Study**

**Problem Statement**

As a data scientist at a home electronics company which manufactures state of the art smart televisions. We 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 - similar to what the smart TV will use.

**Objective**

Is to train a model on the 'train' folder which performs well on the 'val' folder as well. The final test folder is withheld for evaluation purposes - final model's performance will be tested on the 'test' set.

Below architectures are suggested for analyzing videos using deep learning: -

1. **3D Convolutional Neural Networks (Conv3D)**
2. **CNN + RNN architecture**

**Data Generator**

This is one of the most important part of the code. In the generator, we are going to pre-process the images as we have images of 2 different dimensions (*360 x 360* and *120 x 160*) as well as create a batch of video frames. The generator should be able to take a batch of videos as input without any error. Steps like cropping, resizing and normalization should be performed successfully.

**Data Pre-processing**

* ***Resizing* and *cropping* of the images.** To ensure that the network recognizes the gestures effectively we have used a custom code to crop and resize the image
* ***Normalization* of the images.** Normalized the RGB values of an image to overcome the distortions caused by lights and shadows in an image and also to ensure that they are no issues with gradient propagation

| **Experiment Number** | **Model** | **Result** | **Decision + Explanation** | **Parameters** |
| --- | --- | --- | --- | --- |
| **Conv3D** | | | | |
| 1(Initial Conv3D Model in the notebook) | Conv3D  Images used = 10 , 120x120  Batch size = 128  Layers = Conv3D,BN,Max Pooling,Conv3D,BN,Max Pooling,Conv3D,BN,Max Pooling,GlobalAveragePooling3D,Dropout, Dense,BN,Dense,BN,Dense  Activation = Relu and softmax  Epoch = 15  Optimiser=SGD | Resource Exhaustion Error | Reduced the batch size to 32 from 128 | Total: 539,973 Trainable: 537,925 Non-trainable: 2048 |
| 2(Initial Conv3D Model in the notebook) | Conv3D  Images used = 10 , 120x120  Batch size = 32  Layers = Conv3D,BN,Max Pooling,Conv3D,BN,Max Pooling,Conv3D,BN,Max Pooling,GlobalAveragePooling3D,Dropout, Dense,BN,Dense,BN,Dense  Activation = Relu and softmax  Epoch = 15  Optimiser=SGD | Model was running too slow so, runtime was interrupted to change the batch size and run again as the 3rd model. | Further reduction of batch size will be done. | Total: 539,973 Trainable: 537,925 Non-trainable: 2048 |
| 3(Initial Conv3D Model in the notebook) | Conv3D  Images used = 10 , 120x120  Batch size = 25  Layers = Conv3D,BN,Max Pooling,Conv3D,BN,Max Pooling,Conv3D,BN,Max Pooling,GlobalAveragePooling3D,Dropout, Dense,BN,Dense,BN,Dense  Activation = Relu and softmax  Epoch = 15  Optimiser=SGD | Training Accuracy: 0.64 Validation Accuracy: 0.22 Best\_Epoch/Total\_no\_of\_epochs: 15/15 | A Basic Sequential Model was created to understand the effect of each hyperparameter choice. 3 stacks each consisting of Conv3D, Batch Normalisation, Max Pooling and Dropout were used, then GlobalAveragePooling3D, Dropout,Denselayer, Batch Normalisation,Denselayer, Batch Normalisation and finally Dense with softmax for output. In the input 10 images of 120x120x3 were used. Overfitting was observed. | Total: 539,973 Trainable: 537,925 Non-trainable: 2048 |
| 4(Second Model in the notebook) | Conv3D  Images used = 18 , 84x84  Batch size = 15  Layers = Conv3D,BN,Max Pooling,Conv3D,BN,Max Pooling,Conv3D,BN,Max Pooling,GlobalAveragePooling3D,Dropout, Dense,BN,Dense,BN,Dense  Activation = Relu and softmax  Epoch = 30  Optimiser=SGD | Training Accuracy:0.65 Validation Accuracy:0.79 Best\_Epoch/Total\_no\_of\_epochs:30/30 | Reduced image sizes to 84x84x3 and increased the epochs to observe the effect. Seeing the accuracy scores, architecture change is decided. | Total: 539,973 Trainable: 537,925 Non-trainable: 2048 |
| **5**  (Final Model of Conv3D) and (Final Submitted Model .h5) | Conv3D  Images used = 15 , 120x120  Batch size = 20  Layers = Conv3D,LN,Activation,Max Pooling,Conv3D,LN,Activation,Max Pooling,Conv3D,LN,Activation,Max Pooling,Conv3D,LN,Activation,Max Pooling,GlobalAveragePooling3D, Dense,LN,Activation,Dense  Activation = Relu and softmax  Epoch = 30  Optimiser=Adam | Training Accuracy:0.94 Validation Accuracy:0.89 Best\_Epoch/Total\_no\_of\_epochs:30/30 | Changed the architecture, Added Layer Normalization in place of BatchNormalization since it is more suitable to inputs which are sequential in nature and batch independent as was mentioned in its paper. Removed Dropout as no requirement with LayerNormalization. Image sizes were restored to 120x120x3 and were fed in a batch size of 20. | Total: 1,299,941 Trainable: 1,299,941 Non-trainable: 0 |
| **CNN+RNN** | | | | |
| 1 | Conv2D+GRU  Pre-trained model = VGG16  Images used = 15 , 120x120  Batch size = 32  Layers = 6 (Timedist(Cnn2d),GRU, GRU, GRU  Dense, Dense)  Activation = Relu and softmax  Epoch = 30  Optimiser=SGD | Training Accuracy: 0.76 Validation Accuracy: 0.38 | The model is clearly overfitting, will try with another architecture with CNN LSTM with batch normalization and dropouts and 'ADAM' optimizer and less batch size and less epochs to overcome the overfitting issue | Total params: 15,046,613 Trainable params: 331,925 Non-trainable params: 14,714,688 |
| 2 | Conv2D+LSTM with batch normalization and dropouts  Images used = 15 , 120x120  Batch size = 8  Layers = 19 comprising of (Timedist(Cnn2d-16 layers),LSTM, Dense, Dense)  Activation = Relu and softmax  Epoch = 15 Optimiser=ADAM | Training Accuracy: 0.62 Validation Accuracy: 0.58 | The overfitting issue was resolved in this architecture, however the accuracy is not very good, let us use CNN with GRU with 'ADAM' optimizer to check if the accuracy can be improved | Total params: 1,657,445 Trainable params: 1,656,453 Non-trainable params: 992 |
| 3 | Conv2D+GRU with batch normalization and dropouts  Images used = 16 , 120x120 Batch size = 20  Layers = 15 comprising of (Timedist(Cnn2d-12 layers,GRU, Dense, Dense) Activation = Relu and softmax Epoch = 20  Optimiser=ADAM | Training Accuracy: 0.68 Validation Accuracy: 0.69 | The increase in the batch size from 8 to 16 and increase in epoch from 15 to 20 with number of frame 16, had a very little increase in the accuracy, will use the lightweight mobile net architecture to see if the accuracy can be improved | Total params: 2,573,925 Trainable params: 2,573,445 Non-trainable params: 480 |
| 4 | Transfer Learning using Mobilenet on RNN-CNN with GRU  Images used = 16 , 120x120 Batch size = 5  Layers = 4 (Timedist, GRU, Dense, Dense)  Activation = Relu and softmax Epoch = 15  Optimiser=ADAM | Training Accuracy: 0.93 Validation Accuracy: 0.87 | The transfer learning using mobilnet, with a small batch size of 5, drastically increased the accuracy of both training and validation and the problem of overfitting was resolved. | Total params: 3,693,253 Trainable params: 3,669,317 Non-trainable params: 23,936 |

Based on the above experiments, the conclusion is that final model is Conv3D of experiment 5 :-

Reason:

* Training Accuracy : 94%, Validation Accuracy : 89%
* Number of Parameters is 1,299,941 which is considerably less than all other models and it has achieved this with very less tuning.
* Loss decreases smoothly with the run of each epoch that too with less model parameters.