## **Deep Learning Course Project- Gesture Recognition**

* **S. Anil**

# 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 recognise five different gestures performed by the user which will help users control the TV without using a remote.

* 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

Our task is to train different models on the 'train' folder to predict the action performed in each sequence or video and which performs well on the 'val' folder as well. The final test folder for evaluation is withheld - final model's performance will be tested on the 'test' set.

# Architecture development and training

* Experimented with different model configurations and hyper-parameters and various iterations and combinations of batch sizes, image dimensions, filter sizes, padding and stride length were experimented with. I also played around with different learning rates and *ReduceLROnPlateau* was used to decrease the learning rate if the monitored metrics (*val\_loss*) remains unchanged in between epochs.
* Experimented with *SGD()* and *Adam()* optimizers but went forward with *Adam()* as it lead to improvement in model’s accuracy by rectifying high variance in the model’s parameters. I was unsupportive of experimenting with *Adagrad()* and *Adadelta()* due to the limited computational capacity as these take a lot of time to converge because of their dynamic learning rate functionalities.
* I also made use of *Batch Normalization*, *pooling* and *dropout* *layers* when the model started to overfit, this could be easily witnessed when the model started giving poor validation accuracy in spite of having good training accuracy.
* *Early stopping* was used to put a halt at the training process when the *val\_loss* would start to saturate / model’s performance would stop improving.

# Observations

* It was observed that as the Number of trainable parameters increase, the model takes much more time for training.
* **Batch size ∝ GPU memory / available compute.** A large batch size can throw *GPU Out of memory error,* and thus here I had to play around with the batch size till I was able to arrive at an optimal value of the batch size which GPU could support.
* Increasing the batch size greatly reduces the training time but this also has a negative impact on the model accuracy. This made me realise that there is always a trade-off here on basis of priority.
* *Data Augmentation* and *Early stopping* greatly helped in overcoming the problem of overfitting which our initial version of model was facing.
* *CNN+LSTM* based model with *GRU* cells had better performance than *Conv3D.* As per our understanding, this is something which depends on the kind of data we used, the architecture we developed and the hyper-parameters we chose.
* *Couldn’t execute the Transfer learning model due to computational issues.*
* For detailed information on the Observations and Inference, please refer Table 1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No | Experiment | Result | Explanation | Parameters |
| 1 | Model 1 – Conv 3D -  Base Model - Batch Size = 40 and No. of Epochs = 15 | Training Accuracy - 0.97  Validation Accuracy - 0.23 | Very poor validation accuracy, so augmented the model for better performance | Total params: 1,117,061  Trainable params: 1,116,325  Non-trainable params: 736 |
| 2 | Model 2 – Conv 3D - Augment Data , (3,3,3) filter & 160x160 image resolution | Training Accuracy - 0.81  Validation Accuracy - 0.29 | Data augmentation increased the validation accuracy but not drastically, so reduced the filter size to check the model performance | Total params: 3,638,981  Trainable params: 3,637,477  Non-trainable params: 1,504 |
| 3 | Model 3 – Conv 3D -Reduce filter size to (2,2,2) and image res to 120 x 120 | Training Accuracy - 0.68  Validation Accuracy - 0.14 | Reduction in filter size does not increase the model performance but made it worst, so added more layers in the network to increase model performance | Total params: 1,762,613  Trainable params: 1,761,109  Non-trainable params: 1,504 |
| 4 | Model 4 - Conv 3D- Adding more layers | Training Accuracy - 0.79  Validation Accuracy - 0.73 | Both validation and training accuracies are good in this model, added dropout to check the model performance | Total params: 2,556,533  Trainable params: 2,554,549  Non-trainable params: 1,984 |
| 5 | Model 5 – Conv 3D -Adding dropout at convolution layers | Training Accuracy - 0.84  Validation Accuracy - 0.28 | Poor validation accuracy, so reduced the number of parameters to check the model performance | Total params: 2,556,533  Trainable params: 2,554,549  Non-trainable params: 1,984 |
| 6 | Model 6 -Conv 3D - reducing the number of parameters | Training Accuracy - 0.77  Validation Accuracy - 0.26 | Poor validation accuracy further reduced the number of parameters in the next model | Total params: 696,645  Trainable params: 695,653  Non-trainable params: 992 |
| 7 | Model 7 – Conv 3D - reducing the number of parameters | Training Accuracy - 0.74  Validation Accuracy - 0.32 | Slight increase in the validation accuracy but not drastically | Total params: 504,709  Trainable params: 503,973  Non-trainable params: 736 |
| 8 | Model 8 - CNN- LSTM Model | Training Accuracy - 0.90  Validation Accuracy - 0.45 | CNN & LSTM model increased the training and validation accuracies, but validation accuracy is only 45% | Total params: 1,657,445  Trainable params: 1,656,453  Non-trainable params: 992 |
| 9 | Model 9- CNN LSTM with GRU | Training Accuracy - 0.90  Validation Accuracy - 0.69 | CNN & LSTM model added with GRU increased the model performance drastically, but the validation accuracy is only 69% | Total params: 2,573,925  Trainable params: 2,573,445  Non-trainable params: 480 |

**Selected Model 9 of CNN + LSTM with GRU which had training accuracy as 90% and validation accuracy as approximately 70%**

**\*\* END OF THE DOCUMENT \*\***