**Write-up for Gesture Recognition case study.**

Piyush Kadve and Ajith Rao.

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.

* 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.

**Here’s the data:** <https://drive.google.com/uc?id=1ehyrYBQ5rbQQe6yL4XbLWe3FMvuVUGiL>

# 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.

A number of experiments were run and after a lot of trial and error we finalized 4 models. As discussed in starter videos by **Snehansu** we will try to over fit our model with too much data and

|  |  |  |  |
| --- | --- | --- | --- |
| **Experiment Number** | **Model** | **Result** | **Decision + Explanation** |
| **1** | **Conv3D -** [Creating a model with 160x160 image size,epochs=10 and batch\_size=32] | Too slow as dimensions seem to be too high.  Maximum Accuracy we got is **50%**  Trainable params: 7,708,109 | Reduce the batch size to 20 and image dimensions to 60x60 with 30 frames. Our model might not be able to train properly on huge number of parameters. |
| **2** | **Conv3D** | Reducing parameter lowered the training speed but our accuracy has tanked below **31%**  Trainable params: 1,283,685 | Let us take it up a notch and add more layers and epochs to train new model. Also currently trainable parameters are very low. |
| **3** | **Conv3D** | Increasing the layers and epochs led to increased accuracy but still not upto mark. **40%**  Trainable params: 1,429,173 | Let us try to increase the dimensions of our images. To convey more data change the image resolution to 80x80 ,with frames=30 and by keeping batchsize=10 and epochs=20 |
| **4** | **Conv3D** | This gave us much better and desirable accuracy. **79% & 73%**  Trainable params: 2,177,589 | From exp-2 and exp-3 there is very small change in training and validation accuracy even though we increase the no of layers and no of epochs.We can see the model is more impacted by image resolution,batch size and no of frames.  Lets us go with Conv2D & RNN to build next models. |
| **5** | **Conv2D + RNN** | As mentioned above lets start with simple module then continue enhancing it later. **55%**  Trainable params: 999,813 | Now that initial model is built lets try and enhance it by adding dense neurons and GRU layers.  We will try to over fit this model now. |
| **6** | **Conv2D + RNN (GRU)** | The accuracy has been improved. **68% & 63%**  Trainable params: 1,934,469 | As we have observed improvement in accuracy. Let us check by adding more layers in GRU. Also based on |
| **7** | **Conv2D + RNN (GRU)** | The accuracy has been improved. **79% & 69%**  Trainable params: 1,492,869 | Finally let us add more dropouts and adjust learning rate 0.0001 to optimize our accuracy even further. |
| **Final Model** | **Conv2D + RNN (GRU)** | The accuracy has been further improved. **80% & 79%**  Trainable params: 2,033,541 | This is final optimized model with highest accuracy so far. |

Conclusion on what we have observed in our experiments.

* Dimensions of our inputs have more impact in our accuracy than layers and neurons.
* Huge number of parameters will over fit our data, we need to figure out trade-off in this case.
* We get best possible validation accuracy when epochs are between 15-20.