## **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

## **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.

## **Observations**

* It was observed that as the Number of trainable parameters increase, the model takes much more time for training.
* Increasing the batch size greatly reduces the training time but this also has a negative impact on the model accuracy. This made us realise that there is always a trade-off here on basis of priority -> If we want our model to be ready in a shorter time span, choose larger batch size else you should choose lower batch size if you want your model to be more accurate.
* 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.
* Transfer learning boosted the overall accuracy of the model. We made use of the [MobileNet](https://arxiv.org/abs/1704.04861) Architecture due to its light weight design and high-speed performance coupled with low maintenance as compared to other well-known architectures like VGG16, AlexNet, GoogleNet etc.

## **Results**

For detailed information on the results, please refer Table 1.

After doing all the experiments, we finalized **Model 6– CNN +Transfer Learning + GRU**, which performed well.

Reason:

1. Training Accuracy: 99%, Validation Accuracy: 97%
2. Number of trainable parameters (3,669,317)
3. Learning rate gradually decreased after some Epochs

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Model No** | **Model** | **No. of frames** | **Image Size** | **Batch Size** | **Result** | **Decision + Explanation** |
| 1 | Conv3D | 30 | 120 x 120 | 60 | Train Accuracy: 0.91  Validation Accuracy: 0.19 | This model has very low Validation accuracy when compared to the training accuracy. Also, the validation loss did not improve from 1.6 hence let us modify the batch size. |
| 2 | Conv3D | 20 | 120 x 120 | 10 | Train Accuracy: 0.91  Validation Accuracy: 0.84 | This model seems to be overfitting. The validation loss did not improve much from 0.52 |
| 3 | Conv3D | 30 | 120 x 120 | 10 | Train Accuracy: 0.91  Validation Accuracy: 0.85 | The validation accuracy has marginally improved from the previous model. The validation loss did not improve much from 0.27 |
| 4 | Conv3D | 20 | 120 x 120 | 10 | Train Accuracy: 0.73  Validation Accuracy: 0.71 | The validation loss went up in this model. It was 0.69 And also we can observe that the training accuracy was retained as before, but the validation accuracy has dropped. The time is increased to 221s. Let’s try CNN + GRU model, as Conv3D is not giving us desired validation accuracy. |
| 5 | MobileNet  Transfer  Learning  Model + GRU | 20 | 120 x 120 | 10 | Train Accuracy: 0.98  Validation Accuracy: 0.97 | Validation loss has dropped to 0.0174 Let us increase the noof frames to see if val\_accuracy would increase further. |
| 6 | MobileNet  Transfer  Learning  Model + GRU | 30 | 120 x 120 | 10 | Train Accuracy: 0.99  Validation Accuracy: 0.97 | Validation loss improved from 0.11 to 0.055. Also, we can see the improvement in the validation accuracy in this mode. This is by far the best model we have got.  This model is selected as the final model, as we have got 96% Validation accuracy. |
| 7 | MobileNet  Transfer  Learning  Model + LSTM | 20 | 120 x 120 | 10 | Train Accuracy: 0.97  Validation Accuracy: 0.93 | With LSTM model, we can see that the validation loss did not improve from 0.133. But the training time is less when compared to GRU models. |
| 8 | MobileNet  Transfer  Learning  Model + LSTM | 30 | 120 x 120 | 10 | Train Accuracy: 0.97  Validation Accuracy: 0.96 | With LSTM model, we can see that the validation loss did improve from 0.133 to 0.108. But these figures are big when compared to GRU’s validation loss. |
| **Final Model** | MobileNet  Transfer  Learning  Model + GRU | 30 | 120 x 120 | 10 | Train Accuracy: 0.99  Validation Accuracy: 0.97 | Validation loss improved from 1.0734 to 0.0174  This model is selected as the final model, as we have got train accuracy 99% and 96% Validation accuracy. |

**Table 1: Observations and Results**