# Problem Statement

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

1. Thumbs up: Increase the volume
2. Thumbs down: Decrease the volume
3. Left swipe: 'Jump' backwards 10 seconds
4. Right swipe: 'Jump' forward 10 seconds
5. Stop: Pause the movie

# Experiments and Observations

* Used Google Colab Pro for training all models. In all the models I chooses 128X128 as image width and height. I have Chosen 30 frames for all the models.
* Experimented with different model configurations and hyper-parameters and various iterations and combinations of batch sizes, image dimensions, filter sizes, padding and stride length.
* Experimented with different kind of Batch Normalization, pooling and dropout values and observed the variations in model performances.
* Experimented with different learning rates and ReduceLROnPlateau was used to decrease the learning rate if the monitored metrics (validation loss) remains unchanged in between epochs.
* I 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.
* Data Augmentation and Early stopping greatly helped in overcoming the problem of overfitting.
* CNN+GRU based model had better performance than Conv3D. As per our understanding, this is something which depends on the kind of data I used, the architecture I developed and the hyper-parameters I chose.
* Transfer learning boosted the overall accuracy of the model. I made use of the InceptionV3 with ImageNet.

The following table consists of the experiments done to build a perfect model to predict the gestures from the given data set.

|  |  |  |  |
| --- | --- | --- | --- |
| **Experiment Number** | **Model** | **Result** | **Decision + Explanation** |
| **1** | **Conv3D** | **Overfitting** | **I haven’t used the image augmentation in generator function.** |
| **2** | **Conv3D** | **Train accuracy:0.89**  **Validation accuracy:0.84** | **Increased dense neurons. Increased the dropout rate from 0.25 to 0.50. Used image Augmentation and reduced batch size. Model produced good results. May be increasing the number of epochs increases the accuracy.** |
| **3** | **Conv3D** | **Overfitting** | **Increased number of filters in all layers and increased dense neurons. Used 0.50 dropout rate. Previous model performed well with a smaller number of dense neurons and filter values.** |
| **4** | **CNN with RNN(LSTM)** | **Overfitting** | **Used LSTM with dropout rate 0.50 and Time distributed layer. This model is not performed well so lets decided to use GRUs.** |
| **5** | **CNN with RNN(GRU)** | **Train accuracy:0.91**  **Validation accuracy:0.89** | **Used GRU layer with dropout rate 0.50 and Time distributed layer. This model is showing good accuracy. Let’s use Transfer learning in next step.** |
| **6** | **CNN with RNN(GRU) and Transfer Learning** | **Train accuracy:0.99**  **Validation accuracy:0.97** | **Used Inception model to train the weights from scratch with GRU. The model was given the best results compared to all other models.** |

# Conclusion

The Model built with InceptionV3, Time Distributed, Conv2D and GRU gave better results compared to all the other models and also the model has very least number of parameters compared few best performing models.