Write Up - Gesture Recognition Project

- **Rohitash Tulyani**

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.

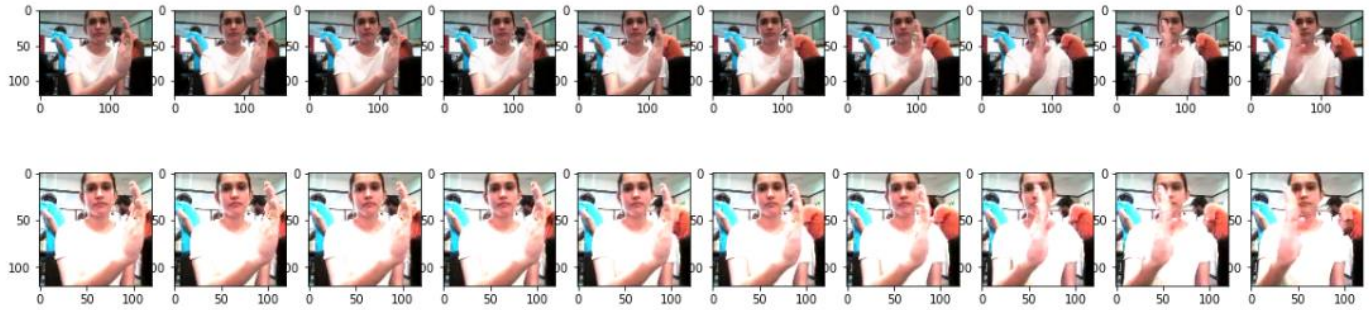
The gestures are continuously monitored by the webcam mounted on the TV. Each gesture corresponds to a specific command:

- 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

Understanding Dataset:

The training data consists of a few hundred videos categorised 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

Sample data:



Task:

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

Following steps has been performed to finalise the model.

Pre-processing steps:

Data Generator
Resizing and cropping of the images
Normalization

Model Training steps:

Changing Model Architecture
Changing batch size
Increasing Epochs
Use of Transfer Learning to use standard models as starting point

Models:

Note: During the project we have used multiple models and changed the same models when needed. In table below, we are presenting details for the models that are present in final python file attached in zip.

Exp #	Model	Hyper parameters	Model Details	Accuracy Results and Decision
1	Conv3D	Epochs = 10 Batch Size = 30 Optimizer = Adam	3 Conv3D layers, Flatten, 3 Dense Layers (last with SoftMax), with batch normalization and max pooling	Train: 0.56 Val: 0.47 # The Accuracy of the model is bad on both Train and Val data. It is underfitting. ## We would need to modify model to get better results.
2	Conv3D	Epochs = 10 Batch Size = 30 Optimizer = Adam	4 Conv3D layers, Flatten, 2 Dense Layers (last with SoftMax), with batch normalization and max pooling	Train: 0.49 Test: 0.32 # The Accuracy of the model is still bad on both Train and Val data. It is still underfitting. ## We would need to modify model to get better results.
3	Conv3D	Epochs = 10 Batch Size = 10 Optimizer = Adam	4 Conv3D layers, Flatten, 2 Dense Layers (last with SoftMax), with batch normalization and max pooling	Train: 0.81 Test: 0.87

			Number of channels/kernels reduced and reduced number of batches	# The Accuracy of the model is good. But Validation accuracy is more than Train. It is overfitting. ## Need to try different params now.
4	Conv3D	Epochs = 20 Batch Size = 10 Optimizer = Adam	Base Architecture: 4 Conv3D layers, Flatten, 2 Dense Layers (last with SoftMax 5 class), with batch normalization and max pooling Number of epochs increased	**Train: 0.97** **Test: 0.90** # The Accuracy of the model is very good. Model is looking stable, but we can try something to increase Validation accuracy. ## Need to try different Model now.
5	CNN + RNN	Epochs = 15 Batch Size = 10 Optimizer = SGD	Transfer Learning + RNN: Time distributed layer with mobilenet input and imagenet weights, time dist flatten layer, GRU layer, 2 Dense layers (last with SoftMax)	**Train: 0.99** **Test: 0.91** # The Accuracy of the model has increase to acceptable level and Model is looking stable also. ## We can be our final model as it is giving acceptable accuracy.

Observations:

- We have also tried hands on layers of LSTM and different batch and epoch size in transfer learning model, however, the presented combination 'Model 5' happens to be the best

- Transfer Learning model provided the stability to the model as well as greater accuracy at the cost of some increase in number of parameters to be trained.