Gesture Recognition

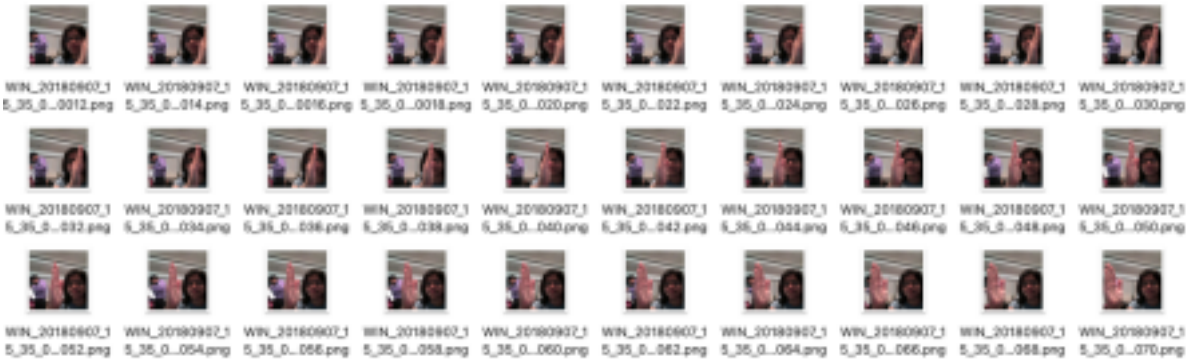
Problem Statement:

As a data scientist working for a leading home electronics manufacturer specializing in cutting-edge smart televisions, your goal is to create an innovative feature for the smart TV. This feature aims to recognize five distinct gestures performed by users, enabling them to control the television without the need for a remote. The gestures and their corresponding actions are as follows:

- ☐ Thumbs up: Increase the volume
- ☐ Thumbs down: Decrease the volume
- ☐ Left Swipe: Rewind the content by 10 seconds
- ☐ Right Swipe: Fast-forward the content by 10 seconds
- ☐ Stop: Pause or stop the current movie or show.

Dataset Overview:

The dataset comprises several hundred videos classified into five distinct classes. Each video consists of a sequence of 30 frames, capturing individuals executing gestures similar to those used for controlling a smart TV.



Here the objective is to build a deep learning model on the train folder and validate it on videos from the var folder. The final test folder is withheld, and final evaluation will be done on that model.

Thus, there are two types of architecture commonly used for analysing videos,

Combining Convolutions with RNN:

The Conv2D network is responsible for extracting a feature vector from each image, and subsequently, a sequence of these feature vectors is inputted into an RNN-based network. The RNN's output is then a conventional softmax,

particularly suitable for classification problems.

3D Convolutional Network, or Conv3D:

Extending the concept of 2D convolutions, 3D convolutions operate in three directions: x, y, and z. While 2D convolutions move filters in two directions (x and y), 3D convolutions move filters in three directions (x, y, and z). In the context of this model, the input to a 3D convolution is a video, which is essentially a sequence of 30 RGB images. Assuming each image has a shape of 100x100x3, the video is represented as a 4-D tensor with a shape of 100x100x3x30, equivalent to (100x100x30)x3, where 3 signifies the number of channels. Analogous to 2D convolutions, a 3D kernel/filter (cubic filter) is denoted as (fxfxf)xc, where f is the filter size, and c is the number of channels. For this model, the cubic filter '3D-convolves' each of the three channels of the (100x100x30) tensor.

Data Generation:

Within the generator, our objective is to preprocess the images. The dataset consists of images with two different dimensions (360 x 360 and 120 x 160). We aim to generate a batch of video frames, ensuring that the generator can seamlessly handle batches of videos without encountering errors. Key preprocessing steps, such as cropping, resizing, and normalization, should be executed successfully.

Data Preprocessing:

Resizing.

Cropping.

Introducing slight rotations to images for data augmentation (ensuring the gesture's intent remains unchanged).

Normalization.

The aforementioned data preprocessing steps have been implemented to enable the neural network to effectively focus on gestures while minimizing the impact of background noise.

Training and Development of the Neural Network:

I conducted experiments with various hyperparameter configurations and batch sizes. Additionally, different learning rates were explored, and the ReduceLROnPlateau technique was employed to reduce the learning rate if the validation loss remained constant across epochs. When the variation in loss became stagnant, indicating a plateau, Early Stopping was implemented to cease the training process.

Furthermore, in instances where overfitting occurred—signified by the model exhibiting lower validation accuracy compared to its optimal training accuracy—I experimented with the inclusion of Batch Normalization, dropout layers, and pooling techniques.

MODEL	EXPERIMENT	RESULT	DECISION + EXPLANATION	PARAMETERS
Conv3D	**1**	**OOM Error**	**Reduce the batch size and Reduce the number of neurons in Dense layer**	**-**
	2	**Training Accuracy : 0.99 Validation Accuracy : 0.81**	**Overfitting Let's add some Dropout Layers**	**1,117,061**
	3	**Training Accuracy : 0.65 Validation Accuracy : 0.52** ***(Best weight Accuracy,Epoch:6/25)***	**Val_loss didn't improve from 1.24219 so early stopping stop the training process. Let's lower the learning rate to 0.0002.**	**3,638,981**
	4	**Training Accuracy : 0.76 Validation Accuracy : 0.72** ***(Best weight Accuracy,Epoch:12/25)***	**Overfitting has reduced but accuracy hasn't improved. Let's try adding more layers**	**1,762,613**
	5	**Training Accuracy : 0.83 Validation Accuracy : 0.76**	**Don't see much performance improvement. Let's try adding dropouts.**	**2,556,533**
	6	**Training Accuracy : 0.84 Validation Accuracy : 0.69**	**Overfitting Increase, adding dropouts has further reduced validation accuracy. Let's try to reduce the parameters**	**2,556,533**
	7	**Training Accuracy : 0.84 Validation Accuracy : 0.74**	**Overfitting reduced, but validation accuracy is still low. Let's try to reduce the parameters. Val Accuracy: 0.49, Train Accuracy: 0.54**	**696,645**
	8	**Training Accuracy : 0.82 Validation Accuracy : 0.73**	**Accuracy remains below same. Let's switch to CNN+LSTM.**	**504,709**

CNN+LSTM	**9 (Model-8 on Notebook)**	**Training Accuracy : 0.93 Validation Accuracy : 0.85**	**CNN - LSTM model - we get a best validation accuracy of 85%.**	**1,657,445**
Conv3D	**Model performance after Data augmentation**			
	10	**Training Accuracy : 0.78 Validation Accuracy : 0.82**	**(3, 3, 3) Filter & 160 x 160 image resolution**	**3,638,981**
	11	**Training Accuracy : 0.72 Validation Accuracy : 0.75**	**(2, 2, 2) Filter & 120 x 120 image resolution. Increase epoch count to 20. Network is generalizing well.**	**1,762,613**
	12	**Training Accuracy : 0.87 Validation Accuracy : 0.78**	**Adding more layers.**	**2,556,533**
	13	**Training Accuracy : 0.65 Validation Accuracy : 0.25**	**Very low performance. Let's reduce the network parameters.**	**2,556,533**
	14	**Training Accuracy : 0.89 Validation Accuracy : 0.78**	**After reducing network parameters, model's performance is quite good.**	**696,645**
	15	**Training Accuracy : 0.88 Validation Accuracy : 0.81**	**Reducing network parameters again.**	**504,709**
CNN LSTM with GRU	**16**	**Training Accuracy : 0.98 Validation Accuracy : 0.77**	**Overfitting is considerably high, not much improvement.**	**2,573,541**
Transfer Learning(Optional)	**17**	**Training Accuracy : 0.85 Validation Accuracy : 0.58**	**We are not training the MobileNet weights that can see, validation accuracy is very poor.**	**3,840,453**
Transfer Learning with GRU &(Optional)	**18**	**Training Accuracy : 0.98 Validation Accuracy : 0.93**	**Overall Training and validation accuracy looks good.**	**3,692,869**