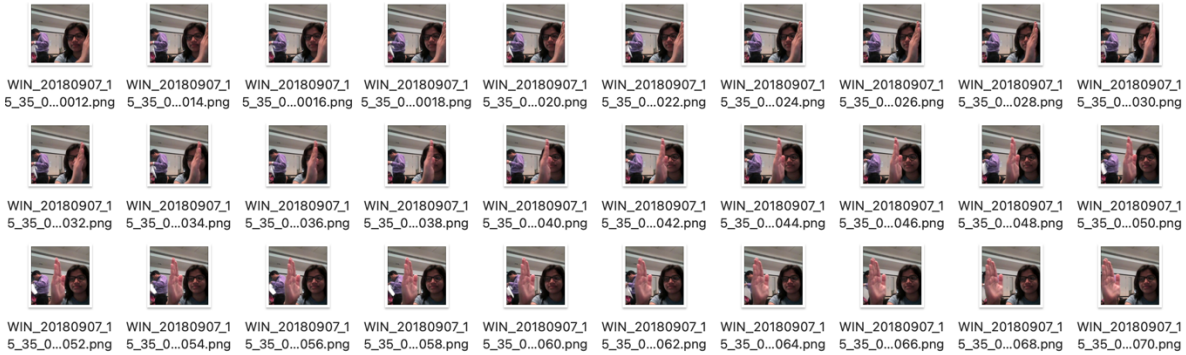
Gesture Recognition

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 recognise 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 backward by 10 sec
4. Right Swipe: Jump forward by 10 sec
5. Stop: Stop the movie

Understanding of dataset:
Dataset contains few hundreds of videos categorized into 5 classes. Each video is a sequence of 30 frames. These videos are recording of various people performing the gestures like what can be performed on smart TV.



Here objective is to build a deep learning model on train folder and validate it on videos from 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,

1. **Convolutions + RNN**

   The conv2D network will extract a feature vector for each image, and a sequence of these feature vectors is then fed to an RNN-based network. The output of the RNN is a regular softmax (for a classification problem such as this one).

2. **3D Convolutional Network, or Conv3D**

   3D convolutions are a natural extension to the 2D convolutions.
   Just like in 2D conv, you move the filter in two directions (x and y), in 3D conv, we move the filter in three directions (x, y and z). In this case, the input to a 3D conv is a video (which is a sequence of 30 RGB images). If we assume that the shape of each image is 100x100x3, for example, the video becomes a 4-D tensor of shape 100x100x3x30 which can be written as (100x100x30)x3 where 3 is the number of channels. Hence, deriving the analogy from 2-D convolutions where a 2-D kernel/filter (a square filter) is represented as (fxf)xc where f is filter size and c is the number of channels, a 3-D kernel/filter (a 'cubic' filter) is represented as (fxfxf)xc (here c = 3 since the input images have three channels). This cubic filter will now '3D-convolve' on each of the three channels of the (100x100x30) tensor.

Data Generation:
In the generator, we are going to pre-process the images. In the dataset images are of 2 different dimensions (*360 x 360* and *120 x 160*). We create a batch of video frames. The generator should be able to take a batch of videos as input without any error. Steps like cropping, resizing and normalization should be performed successfully.

Data Pre-processing:

1. Resizing.
2. Cropping.
3. Slight rotation of images for data augmentation. (Taken care not to change the intent of the gesture)
4. Normalisation

Above data preprocessing steps have been performed, so that neural network focusses on the gestures effectively rather than focusing on the background noises.

Neural Network Training and development:

Experimented with different hyperparameters and batch size combinations. Also tried around different learning rates and ReduceLROnPlateau used to decrease learning rate incase of var_loss remains same between different epochs.

Whenever var_loss started saturating and there is no significant difference in model performance, Early Stopping was used to put a halt on training process.

Also tried out Batch normalization, dropout layers and pooling, when models started overfitting (meaning model giving less validation accuracy instead of best training accuracy).

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