Gesture Recognition Case study IIITB Assignment

Developed by:

1. Priyesh Raj
2. Rupal Garewal

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:

Gesture	Corresponding Action
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.

Each video is a sequence of 30 frames (or images).

Objectives:

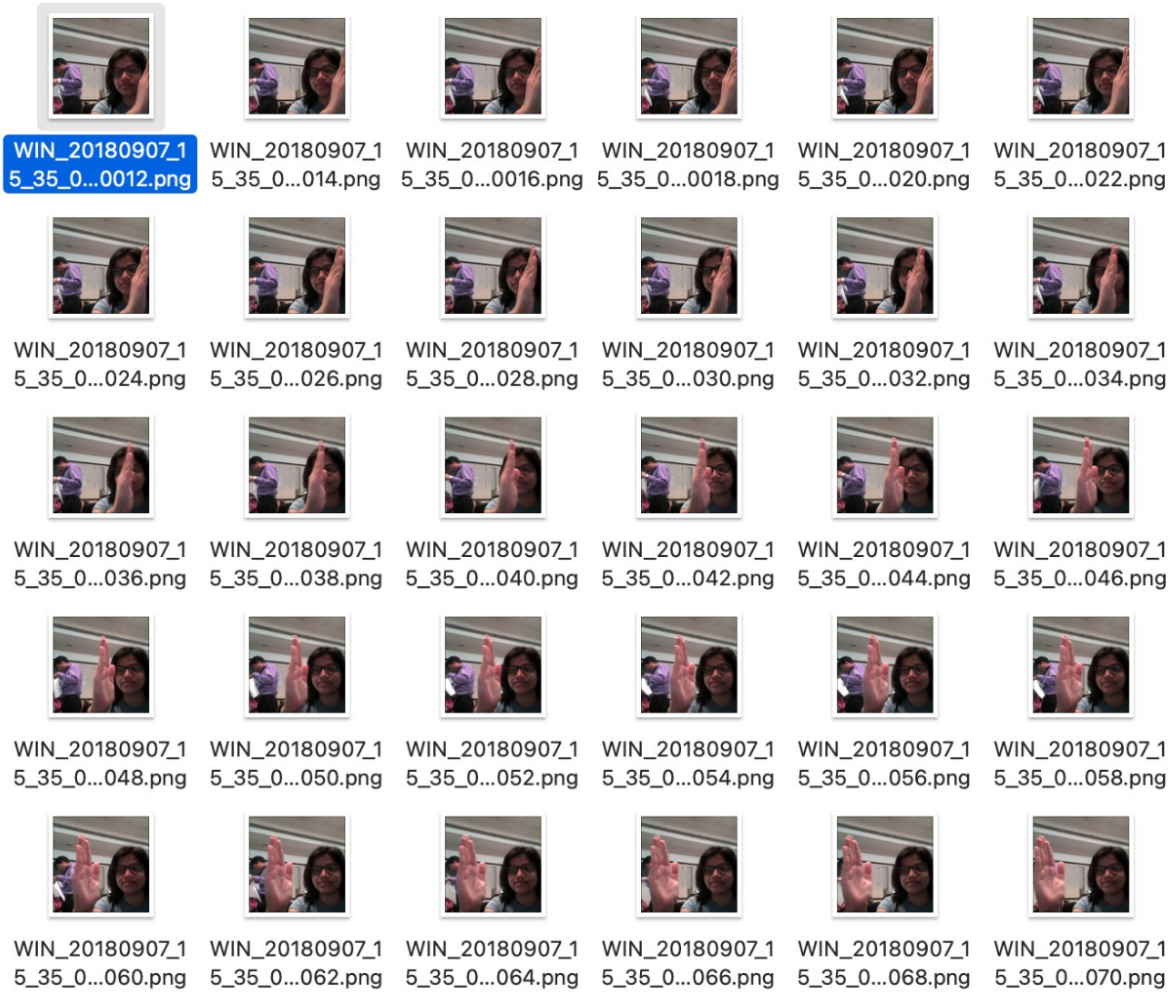
1. **Generator**: The generator should be able to take a batch of videos as input without any error. Steps like cropping, resizing and normalization should be performed successfully.
2. **Model**: Develop a model that is able to train without any errors which will be judged on the total number of parameters (as the inference(prediction) time should be less) and the accuracy achieved. As suggested by Snehansu, start training on a small amount of data and then proceed further.
3. **Write up**: This should contain the detailed procedure followed in choosing the final model. The write up should start with the reason for choosing the base model, then highlight the reasons and metrics taken into consideration to modify and experiment to arrive at the final model.

Installation:

Run *pip install -r requirements.txt* to install all the dependencies.

Dataset:

The training data consists of a few hundred videos categorized 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.It looks like this:



Result :

A lot of models were tried with the provided dataset to generate the right model with the right size. Initially, we struggled with loading the generator. Somehow, the generator was loading all the images without resizing, which resulted in Kernel Dead error multiple times. It took us a lot of time to debug the error and find out the culprit. The kernel was dying on almost all the platforms like Collab, Collab Pro and Kaggle.

Once the issue with the kernel was fixed, we found that our models performed suboptimal. The train accuracy and validation accuracy were quite low. This again was attributed to wrong use of numpy arrays in generator functions.

After we fixed the issue, we were able to run 2 models which gave a good performance. The resultant models of all the experiments are listed below. The final model was selected because it was the only one which gave a proper accuracy on training and validation set.

Experiment No	Model	Batch Size	Image Size	Parameters	Result	
1	Base Conv3D without any thought.	64	150 * 150	Total params: 2,067,621 Trainable params: 2,067,013 Non-trainable params: 608	Kernel Killed	
2	Base Conv3D without any thought.	64	100 * 100	Total params: 2,067,621 Trainable params: 2,067,013 Non-trainable params: 608	Kernel Killed	
3	Base Conv3D without any thought.	32	75 * 75	Total params: 2,067,621 Trainable params: 2,067,013 Non-trainable params: 608	Kernel Killed	
4	Base Conv3D without any thought.	32	50 * 50	Total params: 2,067,621 Trainable params: 2,067,013 Non-trainable params: 608	Kernel Killed	
	Change Introduced:-> Debugged and found that Generator was not working properly and was loading **all images without resize in the main memory**. As a result RAM utilization was off the charts and the kernel was killed on both Collab, Collab Pro and Kaggle. Fixed resizing issue with Kernel.					
5	Conv3D	64	75*75	Total params: 227,370,149 Trainable params: 227,368,869 Non-trainable params: 1,280	Train accuracy: 0.22 Val accuracy: 0.25	
4	Conv3D with reduced params	32	75*75	Total params: 3,357,253 Trainable params: 3,356,645 Non-trainable params: 608	Train accuracy: 0.46 Val accuracy: 0.33	

5	Conv3D with Same pooling layer	32	75 * 75	Total params: 1,121,477 Trainable params: 1,120,613 Non-trainable params: 864	Train accuracy: 0.21 Val accuracy: 0.25	
6	Conv2D with LSTM	32	75 * 75	Total params: 1,773,413 Trainable params: 1,772,421 Non-trainable params: 992	Train accuracy: 0.38 Val accuracy: 0.375	
7	Conv2D with LSTM with Reduced Parameter	32	75 * 75			
	Change Introduced: 1. Batch load logic change in Generator. Use of Numpy was making the image load and processing error prone. 2. Image Size changed to 80 * 80 3. Optimiser changed to SGD from Adam.					
8	Conv3D	32	80 * 80	Total params: 127,493,285 Trainable params: 127,492,005 Non-trainable params: 1,280	Train accuracy: 0.99 Val accuracy: 0.58	
9	*Conv3D with reduced params*	*32*	*80 * 80*	*Total params: 6,203,525* *Trainable params: 6,203,333* *Non-trainable params: 192*	***Train accuracy: 0.82 Val accuracy: 0.67***	**Selected Model**