## **- Gesture Recognition Assignment**

* **Rahul Kumar**

**Batch: MLC21**

# Problem Statement

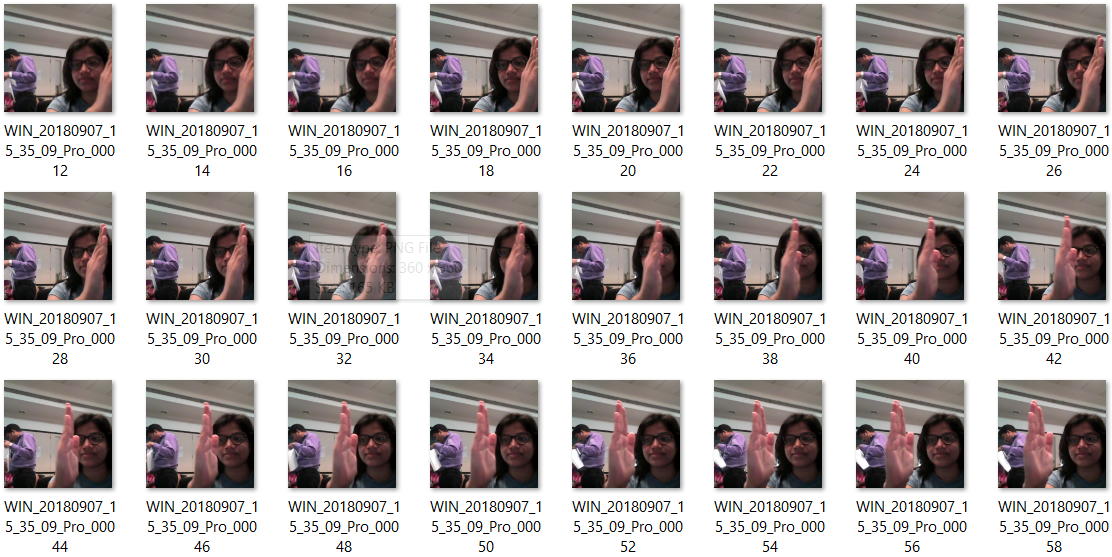
Imagine you are working as a data scientist at a home electronics company which manufactures state of the art smart televisions. You want to develop a cool feature in the smart-TV that can recognise five different gestures performed by the user which will help users control the TV without using a remote.

* 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.

**Here’s the data:** <https://drive.google.com/uc?id=1ehyrYBQ5rbQQe6yL4XbLWe3FMvuVUGiL>

# Understanding the Dataset

Training data contains videos categorized five classes. Each video is of 30 frames. 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.



# Objective

Objective is to build a model to recognise 5 hand gestures by building model on given dataset with minimum parameters so that prediction time is reduced and accuracy is increased.

# Two types of architectures suggested for analysing videos using deep learning:

1. **3D Convolutional Neural Networks (Conv3D)**

3D convolutions are a natural extension to the 2D convolutions you are already familiar with. Just like in 2D conv, you move the filter in two directions (x and y), in 3D conv, you 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.

As an example, let's calculate the output shape and the number of parameters in a Conv3D with an example of a video having 7 frames. Each image is an RGB image of dimension 100x100x3. Here, the number of channels is 3.

The input (video) is then 7 images stacked on top of each other, so the shape of the input is (100x100x7)x3, i.e (length x width x number of images) x number of channels. Now, let's use a 3-D filter of size 2x2x2. This is represented as (2x2x2)x3 since the filter has the same number of channels as the input (exactly like in 2D convs).

1. **CNN + RNN architecture**

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.

# Data 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.

# NN Architecture development and training

* Different learning rates and configuration was tested to find the optimal configuration of model . *ReduceLROnPlateau* was used to decrease the learning rate if the monitored metrics (*val\_loss*) remains unchanged in between epochs.
* We used *SGD()* and *Adam()* optimizers and finally decided to use *Adam()* as it lead to improvement in model’s accuracy by rectifying high variance in the model’s parameters.
* We used *Batch Normalization*, *pooling* and *dropout* *layers* to remove overfit and achieve better accuracy.
* *Early stopping* was used to stop training process when the *val\_loss* start saturating or model’s permormance would stop improving.

# Observations

* Time taken to tran a model is directly proportional to number of parameters.
* Increasing the batch size improves the training time but has a negative impact on the model accuracy. This shows there is trade off between timing and accuracy.
* *CNN+LSTM* based model with *GRU* cells had better performance than *Conv3D.* As per our understanding, this is something which depends on the kind of data we used, the architecture we developed and the hyper-parameters we chose.
* *Transfer learning* **boosted** the overall accuracy of the model. [*MobileNet*](https://arxiv.org/abs/1704.04861) Architecture was used due to it’s light weight design and high speed performance .
* For detailed information on the Observations and Inference, please refer Table 1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **MODEL** | **EXPERIMENT** | **RESULT** | **DECISION + EXPLANATION** | **PARAMETERS** |
|  | **1** | **Training Accuracy : 0.94**  **Validation Accuracy : 0.82** | **Overfitting** | **1,736,389** |
| **2** | **Training Accuracy : 0.80**  **Validation Accuracy : 0.66**  ***(Best weight Accuracy,Epoch:6/25)*** | **Validation accuracy has decrease**  **Val loss didn’t improve beyond 1** | **3,638,981** |
| **3** | **Training Accuracy : 0.71**  **Validation Accuracy : 0.19**  ***(Best weight Accuracy,Epoch:12/25)*** | **Underfitting, accuracy didn’t improve beyond 0.23** | **1,762,613** |
| **4** | **Training Accuracy : 0.80**  **Validation Accuracy : 0.23** | ***Don’t see accuracy improvement. Let's try adding dropouts, best val accuracy is near to 0.3. Lets try with drop outs*** | **2,556,533** |
| **5** | **Training Accuracy : 0.76**  **Validation Accuracy : 0.30** | **Overfitting Increase, adding dropouts has further reduced validation accuracy. Let's try to reduce the parameters** | **398,213** |
| **CNN+LSTM** | **6** | **Training Accuracy : 0.97 Validation Accuracy : 0.88** | **CNN - LSTM model - we get a best validation accuracy of 88%.** | **1,657,445** |
| **Transfer Learning with GRU &(Optional)** | **7** | **Training Accuracy : 0.98 Validation Accuracy : 0.93** | **Best result but with large number of training params** | **3,692,869** |

**Table 1: Observations and Results for numerous tested NN architectures**

# Final Model chosen was 6 (CNN + LSTM )

**Reason for choosing Model 6 i.e.. CNN + LSTM as final model:**

* Training accuracy 97% and validation accuracy is 88%.
* Less number of training params i.e.. 1,657,445.
* Learning rate gradually decreasing after some Epochs