**Problem Statement**

To develop a cool feature in the smart-TV that can recognize 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.

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 recognize 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

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

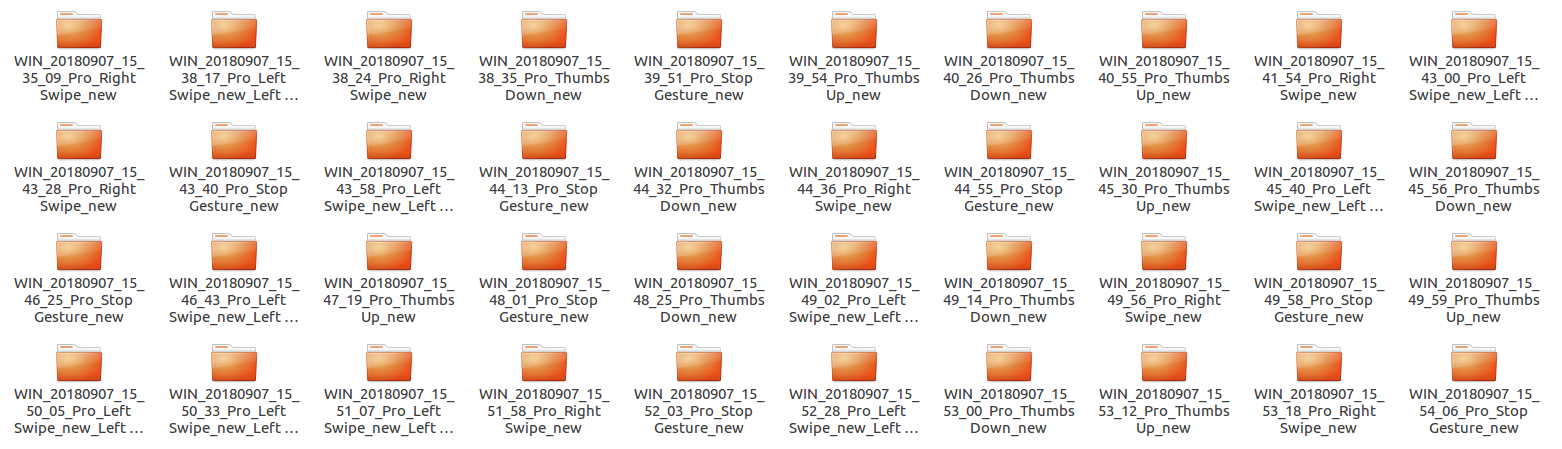
**Understanding the Dataset**

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 - like what the smart TV will use.

The data is in a [zip](https://drive.google.com/uc?id=1ehyrYBQ5rbQQe6yL4XbLWe3FMvuVUGiL) file. The zip file contains a 'train' and a 'val' folder with two CSV files for the two folders.

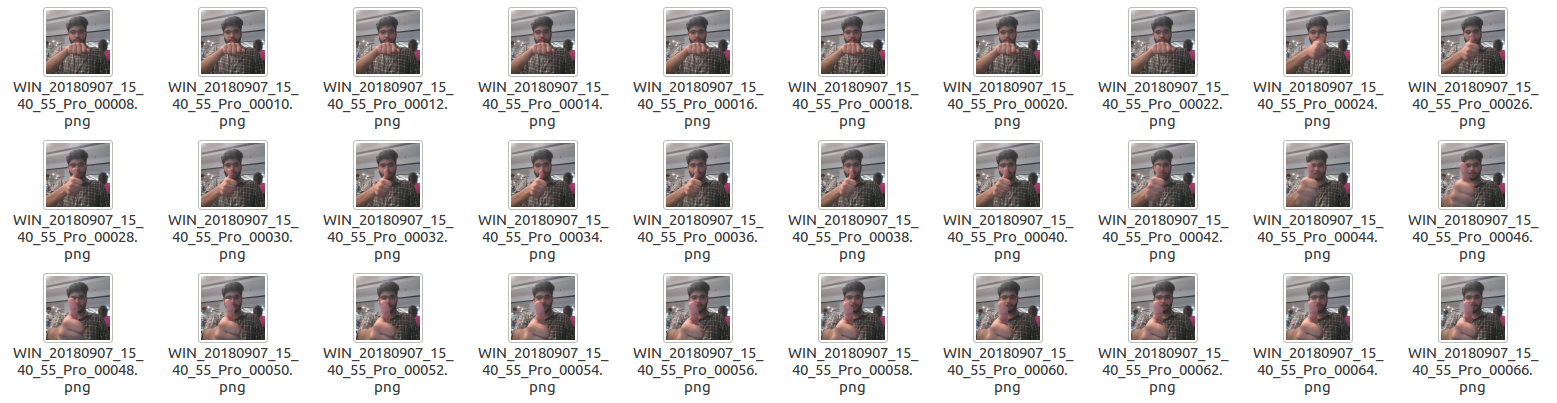
[](https://github.com/santhoshpkumar/hand-gesture-recognition-using-neural-networks/blob/master/images/dataset1.png)

These folders are in turn divided into subfolders where each subfolder represents a video of a particular gesture.

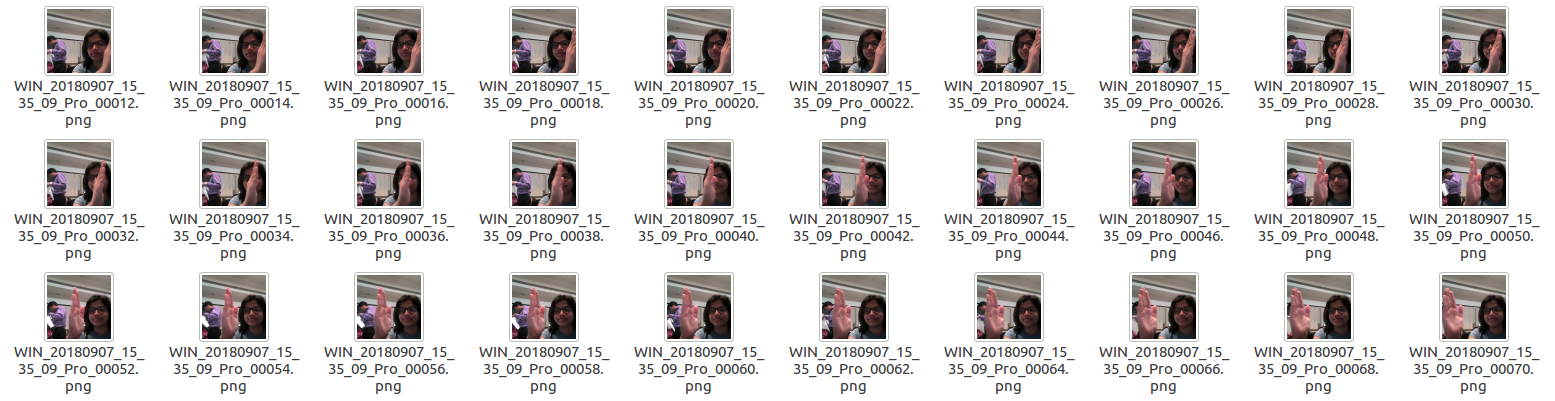
[](https://github.com/santhoshpkumar/hand-gesture-recognition-using-neural-networks/blob/master/images/dataset2.png)

Each subfolder, i.e., a video, contains 30 frames (or images).

* Thumbs Up

[](https://github.com/santhoshpkumar/hand-gesture-recognition-using-neural-networks/blob/master/images/gesture_thumbs_up.png)

* Right Swipe

[](https://github.com/santhoshpkumar/hand-gesture-recognition-using-neural-networks/blob/master/images/gesture_right_swipe.png)

Note that all images in a particular video subfolder have the same dimensions but different videos may have different dimensions. Specifically, videos have two types of dimensions - either 360x360 or 120x160 (depending on the webcam used to record the videos).

**Two Architectures: 3D Convs and CNN-RNN Stack**

After understanding and acquiring the dataset, the next step is to try out different architectures to solve this problem.

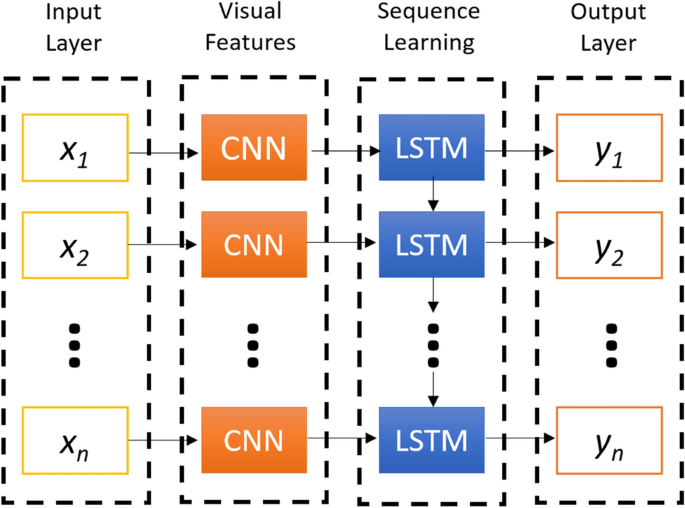
For analyzing videos using neural networks, two types of architectures are used commonly.

One is the standard **CNN + RNN** architecture in which you pass the images of a video through a CNN which extracts a feature vector for each image, and then pass the sequence of these feature vectors through an RNN.

*Note:*

* You can use transfer learning in the 2D CNN layer rather than training your own CNN
* GRU (Gated Recurrent Unit) or LSTM (Long Short-Term Memory) can be used for the RNN. Here we have used GRU to train model

The other popular architecture used to process videos is a natural extension of CNNs - a **3D convolutional network**. In this project, we have tried both these architectures.



**Figure 2: A simple representation of an ensembled CNN+LSTM Architecture**

**Generators**

**Understanding Generators**: In most deep learning projects you need to feed data to the model in batches. This is done using the concept of generators.

Creating data generators is probably the most important part of building a training pipeline. In this project we have implemented our own custom generator, our generator will feed batches of videos, not images.

Let's take an example, assume we have 23 samples, and we pick batch size as 10.

In this case there will be 2 complete batches of ten each

* Batch 1: 10
* Batch 2: 10
* Batch 3: 3

The final run will be for the remaining batch that was not part of the full batch.

Full batches are covered as part of the for loop the remainder are covered post the for loop.

Note: this also covers the case, where in batch size is day 30 and we have only 23 samples. In this case there will be only one single batch with 23 samples.

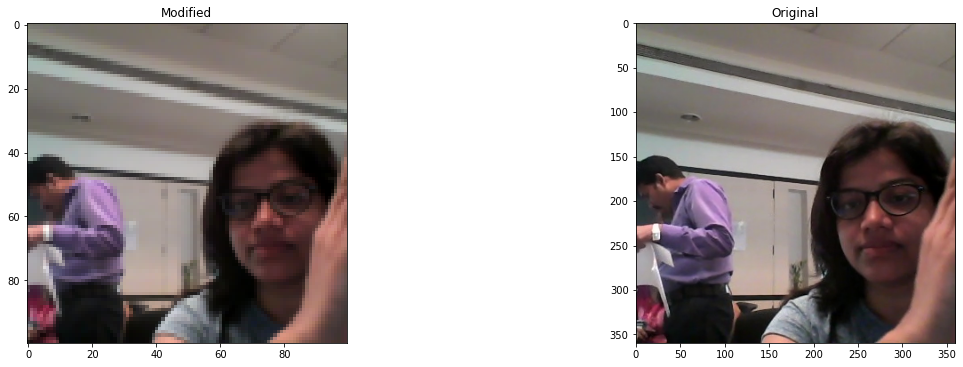
**Reading Video as Frames**

Note that in our project, each gesture is a broken into individual frame. Each gesture consists of 30 individual frames. While loading this data via the generator there is need to sort the frames if we want to maintain the temporal information.

The order of the images loaded might be random and so it is necessary to apply sort on the list of files before reading each frame.

# Data Pre-processing

* ***Resizing* and *cropping* of the images.** This was mainly done to ensure that the NN only recognizes the gestures effectively rather than focusing on the other background noise present in the image.
* ***Normalization* of the images.** Normalizing the RGB values of an image can at times be a simple and effective way to get rid of distortions caused by lights and shadows in an image.
* ***Images are brought to same size***



# Implementation

## 3D Convolutional Network, or Conv3D

Now, let’s implement a 3D convolutional Neural network on this dataset. To use 2D convolutions, we first convert every image into a 3D shape: width, height, channels. Channels represents the slices of red, green, and blue layers. So, it is set as 3. In the similar manner, we will convert the input dataset into 4D shape to use 3D convolution for: length, breadth, height, channel (r/g/b).

While we tried with multiple ***filter size***, bigger filter size is resource intensive, and we have done most experiment with 3\*3 filter

We have used **Adam** optimizer with its default settings. We have additionally used the ReduceLROnPlateau callback to reduce learning rate

## Convolutions + RNN

## The conv2D network will extract a feature vector for each image and a sequence of these feature vectors is then fed to RNN based network(GRU here). The output of RNN is a regular softmax (for classification problem)

## Here GRU variant of RNN is used to train the model and check performance

**Observations -**

* Model has achieved training accuracy of 90% and validation accuracy of 84%
* The trainable parameters is equal to 1,267,685 which is less as compared to other models
* Model is made to overfit. Then overfitting is handled by increasing dropout value in model
* Conv3D model performs better than CNN + GRU model
* ReduceLROnPlateau callback helps to reduce learning rate if no change in loss is observed
* Center cropping is not performed because the information is lost at edges and accuracy obtained was less

**Graphs of training and validation accuracy and loss of final model**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Experiment Number** | **Model** | **Result** | **Decision made** | **Trainable Parameters** |
| **0** | **Conv3D** | **OOM Error** | Used same image resolution and all frames in video | **-** |
| **1** | **Conv3D** | **Training Accuracy:** 0.48%  **Validation Accuracy:** 0.52%  Batch size reduced to 20  The validation accuracy started increasing after epoch 18  The model is underfitting | Reduced batch size, image resolution and number of frames used | 317,557 |
| **2** | **Conv3D** | **Training Accuracy:** 0.69%  **Validation Accuracy:** 0.63%  The model accuracy has increased from previous model  The training accuracy is more than validation accuracy | Decreased batch size to 8 to increase accuracy | 317,557 |
| **3** | **Conv3D** | **Training Accuracy:** 0.65%  **Validation Accuracy:** 0.71%  The validation accuracy has increased  The model is underfitting so increased number of parameters | Changed the optimizer to Adam optimizer | 317,557 |
| **4** | **Conv3D** | **Training Accuracy:** 0.86%  **Validation Accuracy:** 0.75%  The accuracy is significantly increase than previous model  The model seems to overfit so increase dropout value | Increased number of neurons in dense layer | 1,267,685 |
| **5** | **Conv3D** | **Training Accuracy:** 0.67%  **Validation Accuracy:** 0.62%  Overfitting is reduced by increasing dropout value  The accuracy has decreased to some extent | Increased dropout value in model to 0.35 | 1,267,685 |
| **6** | **Conv3D** | **Training Accuracy:** 0.9%  **Validation Accuracy:** 0.84%  This is good model with 90% training and 84% validation accuracy | Decreased batch size to 5 | 1,267,685 |
| **7** | **Conv3D** | **Training Accuracy:** 0.86%  **Validation Accuracy:** 0.77%  The validation accuracy has decreased  Increasing number of frames results in overfitting | Increased the number of frames from 15 to 20 | 2,447,333 |
| **8** | **Conv3D** | **Training Accuracy:** 0.74%  **Validation Accuracy:** 0.73%  By decreasing the image resolution, the accuracy is decreased  There is overfitting or underfitting seen in model | Changed the image resolution to (80,80) | 907,237 |
| **9** | **Conv3D** | **Training Accuracy:** 0.9%  **Validation Accuracy:** 0.78%  By increasing number of epochs, the model started overfitting | Increased number of epochs to 35 | 1,267,685 |
| **10** | **Conv + GRU** | **Training Accuracy:** 0.71%  **Validation Accuracy:** 0.64%  The training accuracy and validation accuracy obtained is 71% and 64%  Add convolution layer to increase accuracy | Used CNN with GRU model to check performance | 3,630,053 |
| **11** | **Conv + GRU** | **Training Accuracy:** 0.83%  **Validation Accuracy:** 0.62%  The model is seen to be overfitting so need to increase dropout value | Added convolution layer to increase accuracy | 3,902,085 |
| **12** | **Conv + GRU** | **Training Accuracy:** 74%  **Validation Accuracy:** 71%  **The overfitting has reduced by increasing dropout value, but the accuracy has reduced** | Increased dropout value to reduce overfitting. | 3,902,085 |
| **13** | **Transfer Learning** | **Training Accuracy:** 91%  **Validation Accuracy:** 85%  The accuracy obtained is good but the trainable parameters are more | Used transfer learning to check model performance with SGD optimizer | 3,422,789 |
| **14** | **Transfer Learning** | **Training Accuracy:** 87%  **Validation Accuracy:** 83%  The accuracy obtained is good but the trainable parameters are more | Used transfer learning to check model performance with Adam optimizer | 3,422,789 |
| **Final Model** | **Conv3D** | **Training Accuracy:** 0.9%  **Validation Accuracy:** 0.84%  This is good model with 90% training and 84% validation accuracy | The parameters used are less and the accuracy obtained is also good | 1,267,685 |