**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 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

**Understanding the dataset**

Each video is a sequence of 30 frames (or images). There are 663 videos provided as training data and 100 videos provided as validation data all images in a video subfolder have the same dimensions different videos may have different dimensions. Specifically, videos have two types of dimensions - either 360x360 or 120x160 There are two csv (one for train, one for validation) files having path of videos.

**Approach**

Our task is to train a model on the 'train' folder which performs well on the 'val' folder as well. Basically, there are two types of architecture commonly used for analyzing videos which we have used here in this case study: - **Convolutions + RNN** and **3D Convolutional Network**, or **Conv3D**.

**Generator** - We have written code for generator function so, that it can provide data in batch while training model. Also, write code for preprocessing image files in generator function.

Preprocessing include cropping the images and making the images in the dataset of same size keeping in mind the crucial pixels. In our case, we have cropped the images to **120\*120 size** making the dataset uniform.

Image normalization is a typical process in image processing that changes the range of pixel intensity values. Its normal purpose is to convert an input image into a range of pixel values that are more familiar or normal to the senses, hence the term normalization.

We have used **min max normalization** here to avoid the effect of outliers and normalize our input dataset

**Model**- We have created models with two architectures

1. Convolution 3D

2.Convolution 2D + RNN

1. RNN with LSTM
2. RNN with GRU

**Output**- Tune model to achieve good accuracy on train as well as validation data

**Plots-** We have also created two plots visualizing training loss with validation loss on one subplot and training accuracy with validation accuracy on the other subplot.

In all we have created 10 models. Among them we have 5 Convolution 3D models, 3 Convolution 2D + RNN with GRU and 2 Convolution 2D + RNN with LSTM.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Experiment Number** | **Model** | **Model Name** | **Number of parameters** | **Result** | **Decision + Explanation** | **Validation Accuracy** |
| **1** | **Conv3D** | conv3D\_model1 | 997,637 | The model was overfitting | Add the drop out layer | 73% |
| **2** | **Conv3D** | conv3D\_model2 | 997,637 | model was not overfitted | Try adding more layers for more accuracy | More than 80%  (~82%) |
| **3** | **Conv3D** | conv3D\_model3 | 822,149 | Not significant difference | Increase the amount of trainabledata/ reduce the filter size | 80% |
| **4** | **Conv3D** | conv3D\_model4 | 616,949 | Giving decent accuracy with lesser number of parameters | Try to change no. of neurons in dense layers | 75% |
| **5** | **Conv3D** | conv3D\_model5 | 1,377,349 | Fairly high accuracy among all models. |  | 87% |
| **6** | **ConvGRU** | model\_gru1 | 3,879,109 | Overfitted model with higher number of parameters | adding a dense layer and decrease no. of neurons in each | 69% |
| **7** | **ConvGRU** | model\_gru2 | 7,545,541 | Again, overfitted model | Add dropouts for regularization and add more layers with different feature maps for better accuracy | 68% |
| **8** | **ConvGRU** | model\_gru3 | 858,085 | Accuracy gone up, overfitting handled, with less number of parameters |  | 80% |
| **9** | **ConvLSTM** | model\_lstm1 | 1,729,317 | model was a case of slight overfitting and accuracy also was not enough. | we have included dropouts to sort this out. Also, we need to work on more number of parameters. | 62% |
| **10** | **ConvLSTM** | model\_lstm2 | 1,657,445 | model gave good results and overfitting was handled with fewernumber of parameters. |  | 82% |
| **Final Model** |  | Conv3D\_model5 | 1,377,349 | Fairly high accuracy among all models. |  | **87%** |