**Deep Learning Case Study**

Table of Contents

[Problem Statement :- 1](#_Toc171110391)

[Understanding the Dataset :- 1](#_Toc171110392)

[Objective :- 1](#_Toc171110393)

[Model Description :- 2](#_Toc171110394)

[CNN + RNN Architecture – 2](#_Toc171110395)

[3D Convolutional Neural Networks (Conv3D) – 2](#_Toc171110396)

[Observations :- 2](#_Toc171110397)

# Problem Statement :-

**Develop a New feature in Smart TV to recognize gestures**

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 :-

**Training Data** – Around 660 videos categorized into one of the 5 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. Along with that ‘train.csv’ file which has the sub folder names, along with gesture name & the class numeric label.

**Validation Data** – Around 100 videos of similar category and ‘val.csv’ file which has the sub folder names, along with gesture name & the class numeric label.

# Objective :-

Our objective is to train a model on the 'train' folder which performs well on the 'val' folder as well (as usually done in ML projects). The data in which the final model performance to be tested is withheld in another ‘test’ folder for evaluation purposes.

# Model Description :-

## CNN + RNN Architecture –

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).

## 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.

# Observations :-

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Model Type** | **Model Name** | **No of Parameters** | **Frames/ Batch Size** | **Epochs** | **Results** | **Observations** |
| Conv3D | conv\_model1 | 105647621 | 10/30 | 15 | Training Accuracy : 0.49 Validation Accuracy: 0.33 | Base Model, with Conv3D, BN layers & Drop Out after each layer. |
| Conv3D | conv\_model2 | 6620005 | 14/30 | 15 | Training Accuracy : 0.2 Validation Accuracy: 0.18 | Reduced the Feature maps, Increased the frames in the same network but Model did not learn anything. |
| Conv3D | conv\_model3 | 13518469 | 14/50 | 15 | OOM Error | Increased the Batch Size to see if learning happens but got OOM Error |
| Conv3D | conv\_model3 | 13518469 | 14/20 | 15 | Training Accuracy : 0.68 Validation Accuracy: 0.41 | Used the same previous model & only updated the batch size |
| Conv3D | conv\_model5 | 694853 | 20/20 | 15 | Training Accuracy : 0.97 Validation Accuracy: 0.7 | Added one more Conv3D layer, but looks like Clearly Overfitting |
| Conv3D | conv\_model6 | 1137861 | 16/20 | 15 | Training Accuracy : 0.97 Validation Accuracy: 0.59 | Added one more Conv3D layer, but looks like Still Overfitting |
| Conv3D | conv\_model7 | 2579653 | 15/20 | 15 | Training Accuracy : 0.96 Validation Accuracy: 0.27 | Added Drop Out Layers after Conv Layers but model became worse, looks like even more Overfitting. So means adding Drop Outs increases the Overfitting. |
| Conv3D | conv\_model8 | 1539909 | 16/20 | 15 | Training Accuracy : 0.96 Validation Accuracy: 0.7 | Included L2 Regulariser. Looks better than the previous model, but almost similar to conv\_model5 (which is so far the best model) but with more parameters when compared to it. |
| **Conv3D** | **conv\_model9** | **1539909** | **16/20** | **25** | **Training Accuracy : 0.97 Validation Accuracy: 0.81** | **Added Learning rate in Optimisers, Increased the Number of epochs to see if the model is learning after 15 epochs. So far the best model.** |
| Conv2D + LSTM | conv\_model10 | 13996645 | 16/20 | 15 | Training Accuracy : 0.98 Validation Accuracy: 0.52 | Still more Overfitting |
| Conv2D + GRU | conv\_model11 | 10524261 | 6/20 | 20 | Training Accuracy : 0.98 Validation Accuracy: 0.76 | Still more Overfitting. GRU Taken less training time due to less parameters for same results as LSTM. Common observation is when the number of epochs increase, the validation accuracy increases. |
| Conv2D + ResNet50 + GRU | conv\_model12 | 26792197 | 10/20 | 15 | Training Accuracy : 0.97 Validation Accuracy: 0.66 | Transfer learning with ResNet50 model seems to Overfit. |
| Conv2D + MobileNet + GRU | conv\_model13 | 3676741 | 10/20 | 20 | Training Accuracy : 0.98 Validation Accuracy: 0.76 | Transfer learning with MobileNet model gives good results comparatively with less number of parameters. |