## **Deep Learning Course Project- Gesture Recognition**

# Problem Statement

As a data scientist at a home electronics company which manufactures state of the art smart televisions. We 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.

# 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 - similar to what the smart TV will use.

**A picture containing photo, many, various, sitting

Description automatically generated**

# Objective

Our task is to train different models on the 'train' folder to predict the action performed in each sequence or video and which performs well on the 'val' folder as well. The final test folder for evaluation is withheld - final model's performance will be tested on the 'test' set.

# 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 *100 x 100 x 3*, for example, the video becomes a 4D tensor of shape *100 x 100 x 3 x 30* which can be written as *(100 x 100 x 30) x 3* where *3* is the number of channels. Hence, deriving the analogy from 2D convolutions where a 2D kernel/filter (a square filter) is represented as *(f x f) x c* where *f* is filter size and *c* is the number of channels, a 3D kernel/filter (a *'cubic'* filter) is represented as *(f x f x f) x c* (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 *(100 x 100 x 30)* tensor

.

**A close up of a box

Description automatically generated**

**30 frames….**

 **Depth**

**Error**

**A picture containing person, woman, holding, sitting

Description automatically generated**

**Conv3D**

**Back**

**Propagation**

**RGB**

***e****.g****.*** *(100 x 100 x 3 x 30)*

**Update**

**Figure 1: A simple representation of working of a 3D-CNN**

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 (for a classification problem such as this one).

A close up of a sign

Description automatically generated

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

# Data Generator

This is one of the most important part of the code. In the generator, we are going to pre-process the images as we have images of 2 different dimensions (*360 x 360* and *120 x 160*) as well as create a batch of video frames. 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.

# 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.

# Observations

* **In most of the model, There is big difference between training and validation accuracy and in transfer learning we are getting 100% accuracy which is not an optimal model hence selecting the less difference and optimal model.**
* **Selecting the Model # 12 as the optimal model with below param**
  + *Model Type - CNN LSTM with GRU ( 20frames + 30 epochs)¶*
  + *Total params: 2,429,28*
  + *Training Accuracy - 0.99*
  + *Validation Accuracy - 0.87*
  + *Inference : Getting descent accuracy with Training and validation*

# Model Details

|  |  |  |  |
| --- | --- | --- | --- |
| Experiment Number | Model | Result | Decision + Explanation |
| Model : 1 | **Conv3D** | **Cropped and resized the images to 100 x100, passed 30 frames, 64 batch size,20 epochs.**  **Total Params :** 2,182,309  **Train accuracy : 0.88**  **Validation accuracy : 0.50** | **Explanation : Descent accuracy on Train but less accuracy on validation data.**  **Decision : As less accuracy on validation to trying different model.** |
| Model : 2 | **Conv3D** | **Frame : 15**  **Epoch : 20**  **Batch size : 32**  **Total Params : 2,067,621**  **Train accuracy : 0.99**  **Validation accuracy : 0.68** | **Explanation : Train accuracy reached to 0.99 which shows that reducing the batch size increased the accuracy little bit.**  **Decision : This is a case of overfitting hence trying different model with more hyperparameter tunning.** |
| Model : 3 | **Conv3D** | **Frame : 20**  **Epoch : 20**  **Batch size : 32**  **Total Params : 2,058,789**  **Train accuracy : 0.99**  **Validation accuracy : 0.81** | **Explanation : Train accuracy reached to 0.99 which shows that reducing the batch size and adding more frames increased the accuracy little bit for validation as well.**  **Decision : This is better model but we can try more hyperparameter tunning.** |
| Model : 4 | **Conv3D** | **Frame : 15**  **Epoch : 20**  **Batch size : 64**  **Total Params :** 3,054,149  **Train accuracy : 0.99**  **Validation accuracy : 0.62** | **Explanation : This looks overfitting by increasing batch size and reducing no of frames.**  **Decision : Here is huge gap between the train and validation accuracy hence not a good optimal model Hence trying more models.** |
| Model : 5 | **Conv3D** | **Frame : 30**  **Epoch : 20**  **Batch size : 20**  **Total Params :** 724,037  **Train accuracy : 0.5**  **Validation accuracy : 0.51** | **Explanation : Very less accuracy here.**  **Decision : discarding this model as very less accuracy for both train and validation data.** |
| Model : 6 | **Conv2D + LSTM** | **Frame : 15**  **Epoch : 20**  **Batch size : 32**  **Total Params :** 3,084,133  **Train accuracy : 0.95**  **Validation accuracy : 0.56** | **Explanation : Trying Conv2D as Conv3D not giving desired result here.**  **Decision : discarding this mode as very less accuracy on train.** |
| Model : 7 | **Conv3D + Adding more dense layers** | **Frame : 30**  **Epoch : 25**  **Batch size : 20**  **Total Params :** 284,805  **Train accuracy : 0.5**  **Validation accuracy : 0.54** | **Explanation : Trying Conv3D again with adding more layers.**  **Decision : discarding this model as very less accuracy for both train and validation data.** |
| Model : 8 | **CNN LSTM with GRU** | **Frame : 30**  **Epoch : 20**  **Batch size : 32**  **Total Params :** 963,941  **Train accuracy : 0.96**  **Validation accuracy : 0.62** | **Explanation : Trying Conv2D as Conv3D not giving desired result here.**  **Decision : discarding this model as very less accuracy for both train and validation data.** |
| Model : 9 | Transfer Learning (MobileNet) with LSTM | **Frame : 20**  **Epoch : 20**  **Batch size : 64**  **Total Params :** 3,857,605  **Train accuracy : 1.0**  **Validation accuracy : 0.87** | **As training gives 1.0 so overfit model** |
| Model : 10 | Transfer Learning (MobileNet) with LSTM | **Frame : 15**  **Epoch : 15**  **Batch size : 64** **Total Params :** 4,611,781 **Train accuracy : 0.94**  **Validation accuracy : 1.0** | **Awesome result on this model.** |
| Model : 11 | Transfer Learning (MobileNet) with GRU | **Frame : 20**  **Epoch : 20**  **Batch size : 64**  **Total Params :** 4,284,613  **Train accuracy : 1.0**  **Validation accuracy : 1.0** | **Awesome result on this model** |
| Model : 12 | **CNN LSTM with GRU** | **Frame : 20**  **Epoch : 30**  **Batch size : 32**  **Total Params :** 2,429,285  **Train accuracy : 0.99**  **Validation accuracy : 0.87** | **Very good model as less difference in training and validation accuracy. We can consider it as final model.** |
| Model : 13 | **CNN LSTM with GRU + more layers + learning rate** | **Frame : 20**  **Epoch : 30**  **Batch size : 64**  **Total Params :** 846,309  **Train accuracy : 0.79**  **Validation accuracy : 0.62** | **Explanation : Very less accuracy here.**  **Decision : discarding this model as very less accuracy for both train and validation data.** |
| Model : 14 | **Conv3D** | **Frame : 20**  **Epoch : 20**  **Batch size : 32**  **Total Params :** 349,765  **Train accuracy : 0.54**  **Validation accuracy : 0.5** | **Explanation : Very less accuracy here.**  **Decision : discarding this model as very less accuracy for both train and validation data.** |
| Model : 15 | **Conv3D + learning rate** | **Frame : 20**  **Epoch : 20**  **Batch size : 32**  **Total Params :** 2,182,309  **Train accuracy : 0.98**  **Validation accuracy : 0.81** | **Good accuracy model.** |
| Final Model | **CNN LSTM with GRU** | **Frame : 20**  **Epoch : 30**  **Batch size : 32**  **Total Params :** 2,429,285  **Train accuracy : 0.99**  **Validation accuracy : 0.87** | **Final Model as it less difference in training and test.** |

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

# Further suggestions for improvement:

* **Using Transfer Learning**: Using a pre-trained *ResNet50/ResNet152/Inception V3* to identify the initial feature vectors and passing them further to a *RNN* for sequence information before finally passing it to a softmax layer for classification of gestures. (This was attempted but other pre-trained models couldn’t be tested due to lack of time and disk space in the nimblebox.ai platform.)
* **Using GRU:** A *GRU* model in place of *LSTM* appears to be a good choice. Trainable Parameters of a *GRU* are far less than that of a *LSTM*. Therefore would have resulted in faster computations. However, its effect on the validation accuracies could be checked to determine if it is actually a good alternative over LSTM.
* **Deeper Understanding of Data:** The video clips where recorded in different backgrounds, lightings, persons and different cameras where used. Further exploration on the available images could give some more information about them and bring more diversity in the dataset. This added information can be exploited in favour inside the generator function adding more stability and accuracy to model.
* **Tuning hyperparameters:** Experimenting with other combinations of hyperparameters like, activation functions (*ReLU, Leaky ReLU, mish, tanh, sigmoid*), other optimizers like *Adagrad()* and *Adadelta()* can further help develop better and more accurate models. Experimenting with other combinations of hyperparameters like the *filter size, paddings, stride\_length, batch\_normalization, dropouts* etc. can further help improve performance.

**\*\* END OF THE DOCUMENT \*\***