## **Gesture Recognition Case Study**

* **Adarsh Kumar Ray**
* **Abhishek Kumar Ray**

# Problem Statement

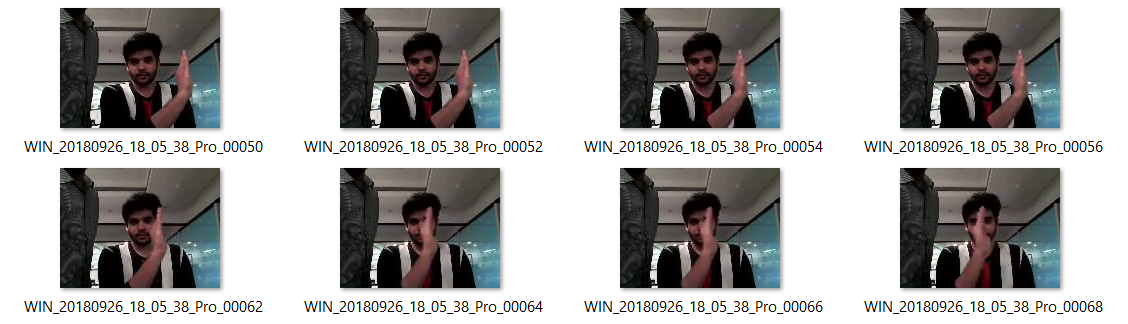
As a data scientist at a home electronics company which manufactures state of the art smart televisions. We want to develop a cool feature in the smart-TV that can recognize five different gestures performed by the user which will help users control the TV without using a remote.

* 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 white background with black text

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.

# Suggested Architectures for analysing videos using deep learning:

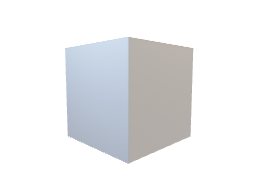
1. **3D Convolutional Neural Networks (Conv3D)**

*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**

A close up of a sign

Description automatically generated

**Figure : 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.

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

# Observations

* It was observed that as the Number of trainable parameters increase, the model takes much more time for training.
* **Batch size ∝ GPU memory / available compute.** A large batch size can throw *GPU Out of memory error,* and thus here we had to play around with the batch size till we were able to arrive at an optimal value of the batch size which our GPU could support.
* Increasing the batch size greatly reduces the training time but this also has a negative impact on the model accuracy.
* **If we want our model to be ready in a shorter time span we can choose larger batch size else, we should choose lower batch size if you want our model to be more accurate.**
* *Data Augmentation* and *Early stopping* helped in overcoming the problem of overfitting.
* Please refer the Below Table which illustrates the Observations and Inference.

|  |  |  |  |
| --- | --- | --- | --- |
| **Experiment Number** | **Model** | **Result** | **Decision + Explanation** |
| **Experiment Run A** | **Conv3D** | **Accuracy:** 0.96  **Validation accuracy:** 0.27  loss: 0.139  Validation loss:2.935  **The model is overfitting. Also, the validation loss is increasing significantly indicating that the model is not generalizing well to unseen data.** | 100 \* 100, Batch Size : 51 and using 18 images from each video frame. |
| **Experiment Run B** | **Conv3D** | **Loss & Validation loss both have Increased.**  **Accuracy:** 0.7385 **Validation accuracy:** 0.100 | **Change batch size to 30 and introduce early stopping and add regularization to the model (like L2 or dropout).** |
| **Experiment Run C** | **Conv3D** | **Total params: 697,541**  The validation accuracy is improved here if needed we can experiment by training the further. | **Decrease parameters and train with batch size 15.** |
|  | **Now we have got an understanding of the impacts. We can now Experiment for accurate results** | | |
| **1** | **Conv3D** | **Total Params: 892101**  **Accuracy:** 0.61  **Validation accuracy:** 0.52  loss: 0.9303  Validation loss:1.1517 | **Dimension : 100 \* 100, Batch Size : 30 and using 18 images from each video frames** |
| **2** | **Conv3D** | **Accuracy:** 0.67  **Validation accuracy:** 0.66  loss: 0.8621  Validation loss:0.8732 | **Decrease the batch size and make Epoch to 30.**  **new\_batch\_size = 20** |
| **3** | **Conv3D** | **Accuracy:** 0.9238  **Validation accuracy:** 0.7100  loss: 0.2715  Validation loss:0.5714.  Increasing Batch size also increased total training time as seen in Experiment Model A as well, Also comparison from previous model after increasing batch size is not satisfactory. | **Increase the batch size and make Epoch to 25**  **new\_batch\_size = 50** |
| **4** | **Conv3D** | **Accuracy:** 0.8130  **Validation accuracy:** 0.8667  loss: 0.4551  Validation loss:0.4838 | **Change the optimizer from SGD to Adam and batch size to 10** |
| **5** | **Conv3D** | **Accuracy:** 0.9249  **Validation accuracy:** 0.9091  loss: 0.2453  Validation loss:0.3109 | **Change batch size to 15 keeping optimizer as Adam** |
| **6** | **Conv3D** | **Total params:** 1,105,093  **Accuracy:** 0.9336  **Validation accuracy:** 0.8000  loss: 0.2071  Validation loss:0.5670  The validation accuracy is dropping and not constant compared to previous results | **Use 120 \* 120 image resolution and we will use 15 batch size.** |
| **7** | **Conv3D** | **Total params:** 1,728,565  **Accuracy:** 0.9442  **Validation accuracy:** 0.9110  loss: 0.1641  Validation loss:0.2925 | **Reduce filter size to (2,2,2) and Batch Size = 15 and adjust dense\_neurons and No. of Epochs = 25** |
| **8** | **Conv3D** | **Total params:** 910,533  **Accuracy:** 0.9819  **Validation accuracy:** 0.9300  loss: 0.0891  Validation loss:0.2660 **Best Results Throughout until now.** | **Adjust filter size further and Reduce dense\_neurons & dropout** |
| **9** | **CNN- LSTM Model** | **Total params:** 1,260,389  **Accuracy:** 0.9382  **Validation accuracy:** 0.8500  loss: 0.2125  Validation loss:0.5647 | **Change Model using LSTM and batch size to 15. Dimension is 100 \* 100** |
| **10** | **CNN- LSTM Model** | **Total params:** 802,917  **Accuracy:** 0.9291  **Validation accuracy:** 0.9100  loss: 0.2652 & Validation loss:0.2975 | **Try to reduce the parameters. Changed dropouts, included Conv2D(256)** |
| **11** | **CNN- LSTM Model** | **Total params:** 799,141  **Accuracy:** 0.9849  **Validation accuracy:** 0.8300  loss: 0.3292  Validation loss:0.9315 | **Try same again with dimensions changed to 120 \* 120** |
| **12** | **CNN- GRU Model** | **Total params:** 639,077  **Accuracy:** 0.9925  **Validation accuracy:** 0.9400  loss: 0.3485  Validation loss:0.5045 | **Try with GRU. Dimension is 120 \* 120** |
| **13** | **CNN- GRU Model** | **Total params:** 639,077  **Result revolves around the previous result itself.  i.e: Accuracy:** 0.99to 1.0000 | **Try further by training more. Epoch = 20** |
| **14** | **CNN- GRU Model** | **Total params:** 246,837  **Accuracy:** 0.9985  **Validation accuracy:** 0.8300  loss: 0.3130  Validation loss:1.0088  In the dimension 100\*100 validation accuracy is less as compared to 120\*120 in experiment 12,13. | **Change dimension to 100 \* 100 and try same again on GRU and also parameters is decreased here.** |
| **15** | **CNN- GRU Model** | **Validation accuracy is not improving and did not convert to 90% so we can discard this.** | **Try Training more on dimension 100 \* 100** |
|  |  |  |  |
|  |  |  |  |
| **Final Model: Experiment 12** | **CNN- GRU Model** | **Total params:** 639,077  **Accuracy:** 0.9925  **Validation accuracy:** 0.9400 | **Dimension: 120 \* 120** |

# Conclusion

CNN- GRU Model can be considered the Final Model.

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