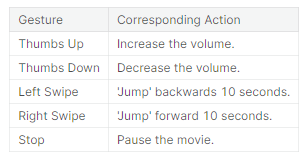
**PG Diploma – IIIT Bangalore – UpGrad**

Gesture Recognition Project

* **Amrita Singh – Group Facilitator**
* **Gowthami Ganga**

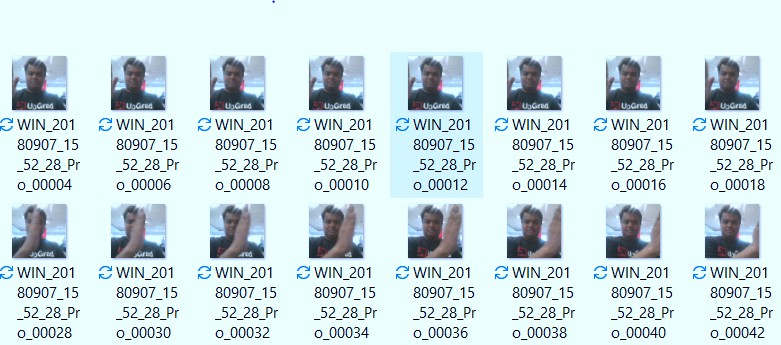
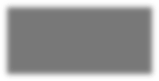
# Problem Statement

1. **This project involves processing videos of 30 frames each for doing Gesture recognition for smart TV.**
2. **Images are from two sources where the dimensions are (160,120,3) and (360,360,3)**
3. **Each Video is tagged with a gesture name amongst the five gesture names.**



# 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.**



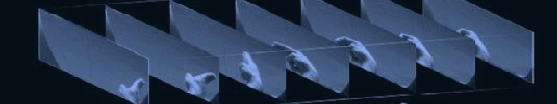
# Objective

**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 analyzing 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 *100x100x3*, for example, the video *becomes a 4D tensor of shape 100x100x3x30 which can be written as (100x100x30)x3 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)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**

****

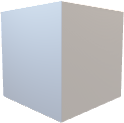
**30 frames….** **Depth**



**RGB**

***e.g.(100x100x3x30)***

# Update



**Conv3D**

**Back Propagation**

**Error**

**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).**

**We are going to use Conv2D + GRU architecture as well as transfer learning of mobile net model + GRU model.**

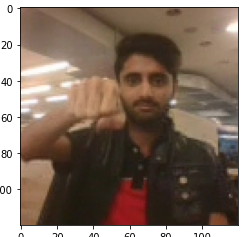
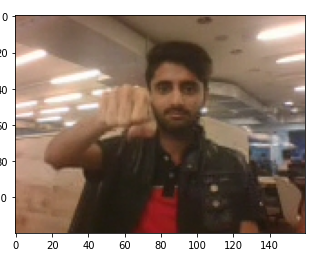
**Mobile net is used because it is lightweight model.**

# 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**

* **It was observed that as the Number of trainable parameters increase, the model takes much more time for training.**
* **Increasing the batch size greatly reduces the training time but this also has a negative impact on the model accuracy. This made us realize that there is always a trade-off here on basis of priority -> If we want our model to be ready in a shorter time span, choose larger batch size else you should choose lower batch size if you want your model to be more accurate.**
* **In our best CONV3D model validation accuracy appears to be 1% more than training accuracy. This happens when the validation set is easier to interpret than the training set. This is NOT a negative sign and is much realistic as you see that the training and validation loss are very close by similar to categorical accuracies.**
* ***CNN+GRU* based model with *GRU* cells had better performance than *Conv3D.*As per our understanding, this is something which depends on the kind of data we used, the architecture we developed and the hyper-parameters we chose.**
* ***Transfer learning* boosted the overall accuracy of the model. We made use of the** [***Mobile Net***](https://arxiv.org/abs/1704.04861)**Architecture due to its light weight design and high speed performance coupled with low maintenance as compared to other well-known architectures like VGG16, Alex Net, Google Net etc.**
* **For detailed information on the Observations and Inference, please refer below table.**

**Why did we not choose data Augmentations?**

* + **In many cases, obtaining a large, diverse dataset can be challenging or expensive. Data augmentation allows you to artificially increase the effective size of your dataset by generating new, slightly modified examples from your existing data. This, in turn, can lead to more robust models.**
  + **We might give some wrong information to the labels if we rotate or flip the image as the order of images is crucial.**
  + **Though we could apply few other methods, we have achieved a descent model without augmentation.**
  + **Also there was no class imbalance problem.**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MODEL | EXPERIMENTS | RESULT | DECISION + EXPLANATION | PARAMETERS |
| CONV3D | **1** | **Training Accuracy : 0.99**  **Validation Accuracy : 0.29** | **Over fitting**  **Reduce the batch size from 32**  **to 20 and increase the epochs from 20 to 30** | 1,155,397  - |
| CONV3D | **2** | **Training Accuracy : 0.97 Validation Accuracy : 0.73** | **Over fitting**  **Let’s add some Dropout Layers and decrease batch size more** | 1,155,397 |
| CONV3D | **3** | **Training Accuracy : 0.78 Validation Accuracy : 0.84 *(Best weight Accuracy,Epoch:26/30)*** | **Val\_ loss didn’t improve from 0.5910. Best model so far for CONV 3D**  **Increase filter size to 3X3X3 and check out results** | 1,155,397 |
| CONV3D | **4** | **Training Accuracy : 0.98 Validation Accuracy : 0.33** | **Over fitting**  **Let’s reduce batch size from 32 to 20** | 1,359,685 |
| CONV3D | **5** | **Training Accuracy : 0.87 Validation Accuracy : 0.34** | **We can see the model is still over fitting, we can reduce the over fitting by a large extent.(By increasing the Epochs)** | 1,359,685 |
| CONV3D | **6** | **Training Accuracy : 0.92 Validation Accuracy : 0.28** | **Let’s add more convolutional layers.** | 1,359,685 |
| CONV3D | **7** | **Training Accuracy : 0.95 Validation Accuracy : 0.15** | **Over fitting**  **Let’s reduce batch size from 32 to 20 and check performance** | 1,618,629 |
| CONV3D | **8** | **Training Accuracy : 0.90 Validation Accuracy : 0.73** | **Validation accuracy increase. More scope for increase. Let’s increase more epochs and check** | 1,618,629 |
| CONV3D | **9** | **Training Accuracy : 0.84 Validation Accuracy : 0.67** | **Validation accuracy doesn’t seem to increase after 30th epoch** | 1,618,629 |
| CONV2D+GRU | **10** | **Training Accuracy : 0.95 Validation Accuracy : 0.69** | **Over fitting**  **Let’s increase dropout and increase batch size**  **to see if it reduces over fitting.** | 1,346,405 |
| CONV2D+GRU | **11** | **Training Accuracy : 0.72 Validation Accuracy : 0.39** | **val\_loss did not improve from 1.7004 Let’s use transfer learning** | 1,346,405 |
| MOBILENET + GRU | **12** | **Training Accuracy : 0.87 Validation Accuracy : 0.76** | **Let’s push the accuracy by decreasing batch size & dropout and increasing frames to sample** | 3,693,253 |
| MOBILENET + GRU | **13** | **Training Accuracy : 0.99 Validation Accuracy : 0.83** | **Let's increase the batch size to 64 and decrease the frames to sample from 25 to 20 and see if it helps.** | 3,693,253 |
| MOBILENET + GRU | **14** | **Training Accuracy : 0.93 Validation Accuracy : 0.81** | **Let’s reduce the number of parameters by reducing the gru cells and dense neurons to 64 from 128 each** | 3,693,253 |
| MOBILENET + GRU | **15** | **Training Accuracy : 0.89 Validation Accuracy : 0.74** | **let’s reduce the batch size to 8 and below configuration with previous increased dense**  **and GRU units of 128 , reduce the frames to 15** | 3,446,725 |
| MOBILENET + GRU | **16** | **Training Accuracy : 0.99 Validation Accuracy : 0.70** | **There is a slight decrease in the validation loss from 0.74 to 0.70**  **Let’s try training on whole mobile net model with the same parameters** | 3,693,253 |
| MOBILENET + GRU | 17 | Training Accuracy : 0.99 Validation Accuracy : 0.98 *(Best weight Accuracy,Epoch:21/25)* | This is the best model till now with the highest validation accuracy of 98% and least validation loss of just 0.0980 | 3,693,253 |

PG Dipl

ma – IIIT Ban

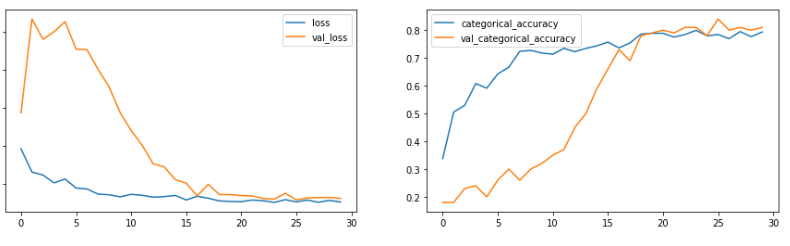
galore – Upgrad

**With all of these experiments these are the best two models along with the plots.**

**Model - CONV3D Best Model**

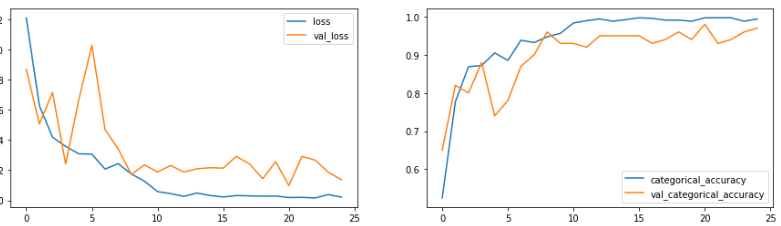
**Weight size ~14 MB Training Accuracy 0.78 Validation Accuracy 0.84**

**Total Parameters 1,155,397**

****

**Model – Mobile net + GRU Best Model**

**Model weights size ~44 MB Training Accuracy 0.99 Validation Accuracy 0.98 Total Parameters 3,693,253**



**Observations and Thoughts:**

**1. We see loss: 0.0191 - categorical\_accuracy: 0.9970 - val\_loss: 0.0980 - val\_categorical\_accuracy: 0.9800 # Epoch - 21**

**2. This is the best model till now with the highest validation accuracy of 98% and least validation loss of just 0.0980**

**Which Model to Choose?? We got a good accuracy of two models:**

**Conv3D**

**- Model3 -> Conv3D -> Epoch26th Model -> Validation accuracy 84% -> validation loss 0.5910 - Total params: 1,155,397**

**- Trainable params: 1,154,405**

**- Non-trainable params: 992**

**Mobile Net (with complete training over all the layers) + GRU**

**- Model17 -> Mobile Net (with complete training over all the layers) + GRU -> epoch 21st Model -> Validation accuracy 98% -> validation loss 0.0980**

**- Total params: 3,693,253**

**- Trainable params: 3,669,317**

**- Non-trainable params: 23,936**

**Conclusion**

**We select the second one which is the Mobile Net + GRU for the following reasons:**

**- The validation loss is way lesser ~0.0980 than the conv3D Model with 0.5910**

**- The accuracy is 98% which is remarkable than Conv3D model with 84%**

**- Trainable params are almost thrice but a 44MB model is not a huge model when compared to CNNs which end up in GBs.**

**- Models in MBs are easily deployable on smart devices.**

**Testing the model on a random batch in validation set**

**Finally we tested the model using a random batch in validation dataset.**



