### **Team Members**

### **Phaneesh N**

* **Praveen Thota**
* **Rama Shankar Sah**

### **Problem Statement**

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 recognise 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.

Please find the link to the dataset: [Project\_data.zip](https://drive.google.com/uc?id=1ehyrYBQ5rbQQe6yL4XbLWe3FMvuVUGiL)

**Understanding the Dataset:**

The Training Dataset consists of a few videos which are categorized into one of the five classes. Each video is divided into a sequence of 30 frames per second. These videos have been taken via webcam by various people.

#### **Objectives:**

1. **Generator:** 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.
2. **Model:** Develop a model that is able to train without any errors which will be judged on the total number of parameters (as the inference(prediction) time should be less) and the accuracy achieved. As suggested by Snehansu, start training on a small amount of data and then proceed further.
3. **Write up:** This should contain the detailed procedure followed in choosing the final model. The write up should start with the reason for choosing the base model, then highlight the reasons and metrics taken into consideration to modify and experiment to arrive at the final model.

**Data Generator**

This is one of the most important parts 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 and not the 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.
* **Data Augmentation**. This is the process to slightly rotate the image so that training can get the data with different angles. This will bring in more data for the model to train on and to make it more generalizable in nature as sometimes the positioning.

**Development & Training of NN Architecture**

We experimented with different model configurations and hyper-parameters and various iterations and combinations of batch sizes, image dimensions(160\*160, 320\*320), filter sizes, padding and same stride length. We used ReduceLROnPlateau to apply different learning rates when model performance i.e. val\_loss is not getting improved. ReduceLROnPlateau was used to decrease the learning rate

We used Adam() optimizers and tried to explore the sgd() but we thought to use Adam() as it would improve the model’s accuracy by rectifying high variance in the model’s parameters.

We also used Batch Normalization, pooling and dropout layers when our model started to overfit, this tried to make model overfitting and could not receive the better validation accuracy instead of having good training accuracy.

Early stopping was used to put a halt at the training process when the val\_loss not getting model performance improvement so it would stop the training process

**Observations:**

1. The initial model was conv3D which had a **Batch Size = 64, 24, 20 and No. of Epochs = 25, 20 No. Of Frames =15, 30 & Image Size = 160\*160, 320\*320** this model was getting overfitted.
2. The second model was CNN-LSTM Model had a **Batch Size = 64, 24, 20 and No. of Epochs = 25, 20 No. Of Frames =15, 30 , Image Size = 160\*160 and dropout of 0.25/0.5** again we saw overfitting.
3. The third model was a conv3D with data augmentation with **Batch Size = 24 and No. of Epochs = 25, 20 , No. Of Frames =15 & Image Size = 160\*160 dropout=0.5,0.25 and change filter = (2,2,2).** the val\_loss was improved in this model and the validation accuracy of 82% and training accuracy of 81% at last stage of epoch
4. The fourth model was CNN-LSTM with GRU Model which had the **Batch Size = 24 and No. of Epochs = 25, No. Of Frames =15 & Image Size = 160\*160 dropout=0.25/.5** model performed very well with the training accuracy of 80% and validation accuracy of 79%
5. The final model was a Transfer Learning with GRU and Augmentation which had a **Batch Size = 5 and No. of Epochs = 20, No. Of Frames =15, Image Size = 120\*120 & Dropout = 0.25** model performed very well with the training accuracy of 99% and validation accuracy of 98%

| **SNo** | **Model** | **Result** | **Decision/Outcome** | **Total Parameters** |
| --- | --- | --- | --- | --- |
| **1** | **Conv3D** | **Training Accuracy: 0.87** | **Model are getting overfitting** | **1,117,061** |
| **Validation Accuracy: 0.31** |
| **2** | **Conv3D** | **Training Accuracy: 0.87** | **Model are getting overfitting** | **1,936,261** |
| **Validation Accuracy: 0.28** |
| **3** | **Conv3D** | **Training Accuracy: 0.76** | **Model are getting overfitting** | **3,574,661** |
| **Validation Accuracy: 0.31** |
| **4** | **Conv3D** | **Training Accuracy: 0.76** | **Model are getting overfitting** | **3,638,981** |
| **Validation Accuracy: 0.31** |
| **5** | **Conv3D - Change the filter size** | **Training Accuracy: 0.85** | **Model are getting overfitting** | **3,433,781** |
| **Validation Accuracy: 0.25** |
| **6** | **Conv3D - Adding more layers** | **Training Accuracy: 0.39** | **Model are getting overfitting** | **4,227,701** |
| **Validation Accuracy: 0.21** |
| **7** | **Conv3D - Adding Dropout at Convolution Layers** | **Training Accuracy: 0.39** | **Model are getting overfitting** | **4,227,701** |
| **Validation Accuracy: 0.32** |
| **8** | **Conv3D - Reduce the parameters** | **Training Accuracy: 0.62** | **Model are getting overfitting** | **922,501** |
| **Validation Accuracy: 0.3** |
| **9** | **NN-LSTM Model** | **Training Accuracy: 0.39** | **Model are getting overfitting** | **3,754,597** |
| **Validation Accuracy: 0.33** |
| **Data Augmentation** | | | | |
|
| **10** | **Conv3D** | **Training Accuracy: 0.76** | **Model perform well at the last stage of epoch** | **3,638,981** |
| **Validation Accuracy: 0.76** |
| **11** | **Conv3D - Change the filter size** | **Training Accuracy: 0.76** | **Model perform well at the last stage of epoch** | **3,433,781** |
| **Validation Accuracy: 0.75** |
| **12** | **Conv3D - Reduce the parameters** | **Training Accuracy: 0.82** | **Model perform well at the last stage of epoch** | **922,501** |
| **Validation Accuracy: 0.81** |
| **13** | **CNN-LSTM with GRU Model** | **Training Accuracy: 0.80** | **Overall model is good and need more number of epoch to get the good outcome** | **5,080,677** |
| **Validation Accuracy: 0.79** |
| **14** | **Transfer Learning with GRU** | **Training Accuracy: 0.99** | **This is one of best model provided very high accuracy** | **3,693,253** |
| **Validation Accuracy: 0.98** |

**Suggestions & Improvement for Further models:**

1. Need to explore the Adagrad(), Adadelta() and other optimizers and check the model performance. Due to time limitations, we are not able to explore the performance of other optimizers.
2. Other Transfer Learning: Need to explore transfer learning methods like VGG16, VGG19, ResNet50, ResNet152, Inception V3 etc..
3. Activation Function: Need to test the different activation functions to check the performance of the model like Leaky ReLU, tanh etc..
4. Hyperparameters: Experimenting with other combinations of hyperparameters like the filter size, paddings, stride\_length, batch\_normalization, dropouts etc. can further help improve performance.