## **Gesture Recognition Project**

* **Anubhav Mishra**
* **Bitragnuta Venkata Sai Vijaya Aditya**

**Cohort: ML C43**

# 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.

**Here’s the data:** <https://drive.google.com/uc?id=1ehyrYBQ5rbQQe6yL4XbLWe3FMvuVUGiL>

# 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.

# 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* 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.

# NN Architecture development and training

* Experimented with different model configurations and hyper-parameters and various iterations and combinations of batch sizes, image dimensions, filter sizes, padding and stride length were experimented with. We also played around with different learning rates and *ReduceLROnPlateau* was used to decrease the learning rate if the monitored metrics (*val\_loss*) remains unchanged in between epochs.
* We experimented with *SGD()* and *Adam()* optimizers but went forward with *Adam()* as it lead to improvement in model’s accuracy by rectifying high variance in the model’s parameters. We were unsupportive of experimenting with *Adagrad()* and *Adadelta()* due to the limited computational capacity as these take a lot of time to converge because of their dynamic learning rate functionalities.
* We also made use of *Batch Normalization*, *pooling* and *dropout* *layers* when our model started to overfit, this could be easily witnessed when our model started giving poor validation accuracy inspite of having good training accuracy J.
* *Early stopping* was used to put a halt at the training process when the *val\_loss* would start to saturate / model’s performance would stop improving.

# 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 realise 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.
* Data Augmentation and Early stopping greatly helped in overcoming the problem of overfitting which our initial version of model was facing.
* CNN+LSTM based model with GRU cells had better performance than Conv3D*.*
* Transfer learning boosted the overall accuracy of the model.
* For detailed information on the Observations and Inference, please refer Table 1.

|  |  |  |  |
| --- | --- | --- | --- |
| **Experiment Number** | **Model** | **Result** | **Decision + Explanation** |
| **1** | **Conv3D**  **Base Model:**  **Batch Size: 40**  **Number of Epochs:15**  **Number of Frames:30** | **Training Accuracy: 42%**  **Validation Accuracy: 24%** | * **Underfitting** * **Remove cropping of image** * **Increase number of epochs** * **Reducing number of frames** * **Reducing batch size** |
| **2** | **Conv3D**  **Batch Size: 20**  **Number of Epochs:25**  **Number of Frames:18** | **Training Accuracy: 85%**  **Validation Accuracy: 30%** | * **Overfitting** * **Implemented Regularization** |
| **3** | **Conv3D with Dropout** | **Training Accuracy: 71%**  **Validation Accuracy: 23%** | * **Overfitting** * **Implemented LSTM or GRU.** |
| **4** | **Conv3D + LSTM** | **Training Accuracy: 86%**  **Validation Accuracy: 35%** | * **Overfitting** * **Implemented Transfer Learning** |
| **5** | **Transfer Learning + LSTM** | **Training Accuracy: 99%**  **Validation Accuracy: 73%** | * **Improve in Validation Accuracy** * **Implemented GRU** |
| **6** | **Transfer Learning + GRU** | **Training Accuracy: 99%**  **Validation Accuracy: 96%** | * **Better Training & Validation Accuracy** |

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

After doing all the experiments, we finalized **Model 6– Transfer Learning + GRU** which performed well.

**Reason:**

* (Training Accuracy: 99%, Validation Accuracy: 96%)

# Further suggestions for improvement:

* **Using Transfer Learning**: Using a pre-trained algorithms 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.
* **Deeper Understanding of Data:** The video clips were 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, other optimizers which can further help develop better and more accurate models. Experimenting with other combinations of hyperparameters can further help improve performance.

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