**Gesture Recognition - Deep Learning Project**

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.

**Observations**:

1. Data Preparation

* The script reads training and validation data from CSV files (train.csv and val.csv).
* Images are organized in folders, each representing a gesture category.
* The script randomly selects images from each folder for training and validation.

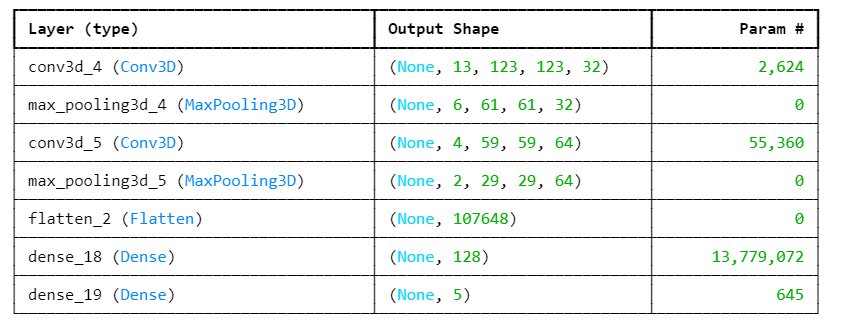
2. Image Preprocessing

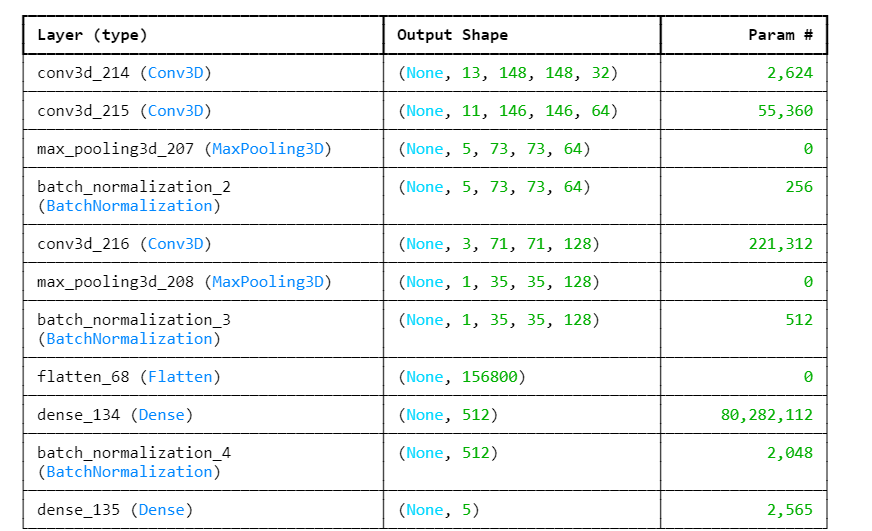
* Images are resized to a consistent size (150x150 pixels) to ensure uniform input dimensions.
* Tried with size 125 x 125 as well to reduce the processing time
* Aspect ratio is maintained by cropping
* Normalisation also applied to images

3. Experiments

* Each model is compiled with the ‘adam’ optimizer and ‘categorical\_crossentropy’ loss function.
* The script experiments with different model architectures:
  + **Conv3D model** – Training Accuracy is 84% and Validation accuracy is 62%. when image size was 150, 150. The model showing that its overfitted

Training Accuracy is 79% and Validation accuracy is 67%. when image size was 125, 125. The model showing that its overfitted

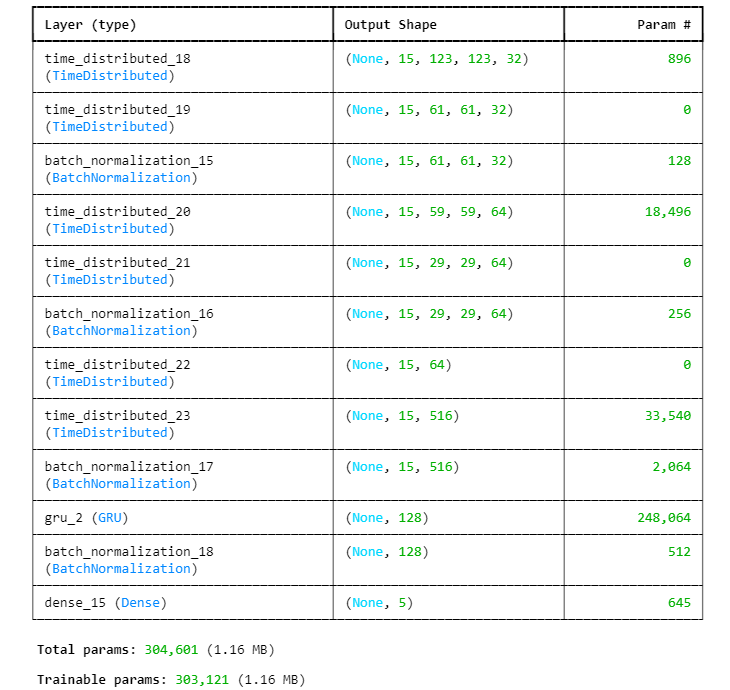


**Conv3D with increased number of neurons** - Increasing the layers, we could achieve the Training Accuracry to 98% but Validation accuracy dropped to 35%. However the model is heavily overfitted

Conclusions – Adding more number of neurons the training accuracy is increased however model is overfitted

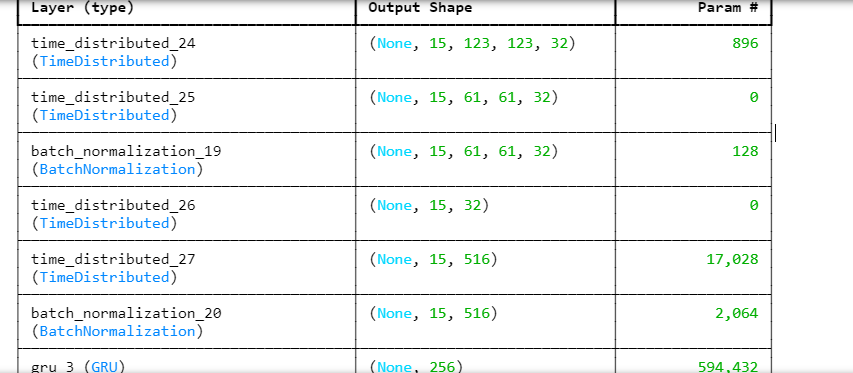
* + **Time Distributed Conv2D + GRU model** - Training Accuracy is 73% and Validation accuracy is 53% for when image size was 150, 150. The model showing that its overfitted

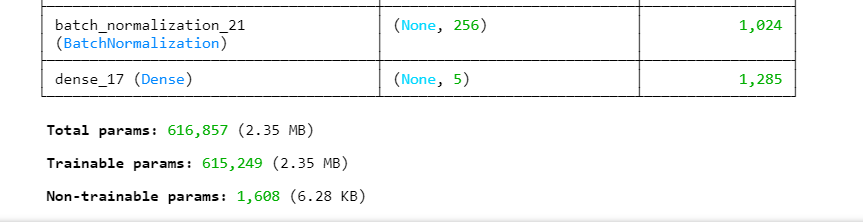
Training Accuracy is 73% and Validation accuracy is 59% for when image size was 125, 125. The model showing that its overfitted



* + **Time Distributed Conv2D + GRU model with greater number of neurons/deep layers -** Training Accuracy is 63% and Validation accuracy is 35% for when image size was 150, 150. The model showing that its overfitted

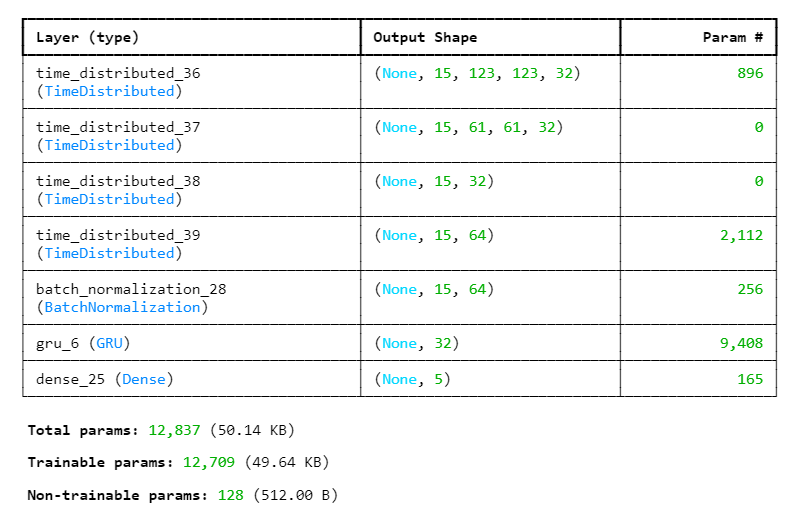
Training Accuracy is 64% and Validation accuracy is 44% for when image size was 125, 125. The model showing that its overfitted

****

****

Conclusions – Adding more number of neurons the training accuracy is increased however model is overfitted

* + **Time Distributed Conv2D + GRU model with lesser number of neurons/shallow layers** – Training accuracy is 57% and validation 32%. However this model does not seems to be a good model

****

4. Challenges Faced/ Lessons Learnt

1. The keras version which we used was 3.1.1. However the given starter code was not working and giving errors for each compilation. For instance, Keras 3.1.1 do not support .h5 file instead it will give as dot keras extension
2. Fit\_generator method was depricated by keras 3.1.1 hence used fit function
3. Memory error or Resource error while running the model – Reduced the batch size , epochs resolved the issue
4. GRU was comparatively faster
5. The greater number of neurons or the more deep the model is the more overfit the model becomes. This statement might not be valid for all other cases. It depends on the input data and the domain

5. Training

* The training data is generated using custom data generators (generator).
* The script trains the chosen model using the training data.
* Learning rate reduction and model checkpointing are implemented during training.