A Project on Deep Learning - Gesture Recognition

Document by – 1) Shashank Karthik D Joshi

2) Rhea Eaphen

Problem Statement

- Considering ourselves working as a data scientist at a multinational company which produces Smart Televisions. We want to introduce a smart feature into the upcoming TV series. The feature includes 5 hand gestures which can control the process on the TV, like Volume increase, decrease and so on.
- As a initial step, we have chosen 5 different hand gestures, which can be expanded further. These gestures are as follows: -

● 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 dataset for the project in: https://drive.google.com/uc?id=1ehyrYBQ5rbQQe6yL4XbLWe3FMvuVUGiL

Understanding the Dataset

- The training data consists of a hundreds of videos categorized into one of the five above mentioned 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.

1. 3D Convolutional Neural network (Conv3D)

- Conv3D is just an extension of Conv2D. The difference here is that, with X, Y and 3 channels, we all have 4th dimension called frames. This frame is an hyperparameter which is tweaked in the model building to get the best Model possible.

2. Conv2D + RNN
   - Here the video folders are initially sent across Conv2D frame to get the initial features from the set of frames. When which is then sent across the RNN network (LSTM and GRUs) to track the feature fetched from CNN. Finally the RNN layers, the dataset goes across dense layers.

Data Generator

This is 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 increases, so do the time of executing/ training the model which increases exponentially
- As we increase the batch-size, the training time decreases. But this doesn't mean the accuracy has increased. Hence, we can say batch size is inversely proportional to Accuracy.
- We could see that, *CNN+LSTM* based model with *GRU* cells had better performance than *Conv3D.*
- Using *CNN+GRU* based model increased the accuracy even better
- Transfer learning did **boost** the overall accuracy of the model. We used the ***ModelNet*** Architecture due to its light weight design and high-speed performance coupled with low maintenance as compared to other well-known architectures like VGG16, AlexNet, GoogleNet etc.
- For detailed information on the Observations and Inference, please refer Table.

Exp. No	Model	Result	Decision + Explanation	Parameters	Hyperparameter
1	Conv3D	Training Accuracy : 0.87 Validation Accuracy : 0.62	Model is too much overfitting. The reason could be the frame size is too less	Total params: 2,067,621 Trainable params : 2,067,013	num_frames = 15 num_epochs = 20 batch_size = 64
2	Conv3D	Training Accuracy : 0.91 Validation Accuracy : 0.875	Model is doing great, as the number of epochs which increased from 20 to 30, which helped the model to learn more on the pattern	Total params: 4,050,085 Trainable params : 4,049,477	num_frames = 20 num_epochs = 30 batch_size = 64
3	Conv3D	Training Accuracy : 0.89 Validation Accuracy : 0.375	Now we tried a new variant by decreasing the batch size from 64 to 32, which affected the model a lot. This could be because the trainable parameter went down which could have affected the pattern training	Total params: 5,618,245 Trainable params : 5,617,637	num_frames = 30 num_epochs = 20 batch_size = 64
4	Conv3D	Training Accuracy : 0.93 Validation Accuracy : 0.75	In this experiment, even though accuracy increased, this model is still overfit.	Total params: 1,907,909 Trainable params : 1,907,045	num_frames = 20 num_epochs = 20 batch_size = 32
5	Conv3D	Training Accuracy : 0.98 Validation Accuracy : 0.375	Next, we tried batch Normalizing before Maxpooling. Same as above the parameter for this is reduced. Hence the Accuracy is very less	Total params: 1,301,045 Trainable params : 1,300,565	num_frames = 25 num_epochs = 25 batch_size = 64
6	Conv2D + LSTM	Training Accuracy : 0.96 Validation Accuracy : 0.69	We switched the model arch. To RNN based. The accuracy here came to be average.	Total params: 3,084,133 Trainable params : 3,083,141	num_frames = 25 num_epochs = 30 batch_size = 32
7	Conv2D + GRU	Training Accuracy : 0.99 Validation Accuracy : 0.875	We then tried RNN based model with GRU, we did this because GRU models take less time to train than LSTM.	Total params: 1,301,045 Trainable params : 1,300,565	n_frames = 20 num_epochs = 30 batch_size = 32

8 (Final Model)	**Transfer Learning (MobileNet) with LSTM**	**Training Accuracy : 1.0** **Validation Accuracy : 1.0**	**Lastly we took the MobileNet model(pre trained model) and use transfer learning to train the few last layers. It the model was trained and was found to be most efficient model created.**	**Total params: 1,301,045** **Trainable params : 1,300,565**	**num_frames = 25** **num_epochs = 30** **batch_size = 64**

After doing all the experiments, we finalized **Model 8– MobileNet + LSTM**, which performed well.

Reason:

- (Training Accuracy: 100%, Validation Accuracy: 100%)
- Number of Parameters (**1,301,045**) less according to other models' performance
- Learning rate gradually decreasing after some Epochs