GESTURE RECOGNITION PROJECT

Problem Statement :

Imagine you are working 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

Understanding the Dataset :

The training data consists of a few hundred videos categorised 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. Sample Data is shown below.

WIN_20180907_15_38_17_Pro_00061
WIN_20180907_15_38_17_Pro_00063
WIN_20180907_15_38_17_Pro_00065
WIN_20180907_15_38_17_Pro_00067
WIN_20180907_15_38_17_Pro_00071
WIN_20180907_15_38_17_Pro_00073
WIN_20180907_15_38_17_Pro_00075
WIN_20180907_15_38_17_Pro_00077

Project Objective and Model Architecture:

Train a model that can correctly identify the 5 hand gestures based on the Test Data.

For the same, there are two Architectures suggested.

- **3D Convs and**
- **CNN-RNN Stack**

3D convolutions are a natural extension to the 2D convolutions. 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).

CNN-RNN stack- 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).

Model Architecture and Training:

- Experimentation with different configurations of the model, hyperparameters, number of iterations and combinations of batch sizes with image sizes, choice of kernels(size, padding, stride) for best accuracy.
- Also, tried different Learning Rates and used *ReduceLRonPlateau* to reduce Learning Rate if the metric(*val_loss*) remains unchanged in between epochs.
- For Optimizers, tried *SGD()* and *Adam()* but went ahead with *Adam()* optimizer as it led to improvement in accuracy by rectifying high variance in the model’s parameters.
- Also, we achieved little Regularization by using *BatchNormalization, pooling* and *Dropouts* when we saw that our model was overfitting (giving poor validation accuracy and good train accuracy)
- Also did *Early Stopping*, when the model performance would stop improving.

OBSERVATIONS:

- Training time is directly proportional to number of parameters
- *Batch Size* should be chosen as per the GPU selected. A large Batch size throws Out of Memory error. So, we need to find the optimal Batch size which our GPU can support.
- Increase in Batch size improves on computational cost however it affects accuracy implying a trade-off between both.
  So, we need to find the best Batch size which gives best computational time and highest accuracy.
- We achieved better accuracies when the image size selected was 128*128 in comparison to 64*64.
- *Early Stopping* helped us in overcoming the problem of overfitting to a great extent
- We obtained a better performing model using *Conv3D* than *CNN+LSTM and CNN+GRU based models*
- Detailed Analysis on next page

ARCHITECTURE	MODEL	PARAMETERS	RESULTS	DECISION+EXPLANATION
CONV3D	**1**	**357,541**	**Train Acc: 95%** **Val Acc: 89%**	**Started with simple model, batch size 30, image size 64, gesture frames 30, channels 1, Epochs 25, LR 0.001 and Dense neurons 128. Statistically simpler model and good validation accuracy achieved**
	2	**2,218,501**	**Train Acc: 95%** **Val Acc: 82%**	**Increased Image size to 128, channels to 3 and Dense neurons to 256. Number of parameters drastically increased. However, validation accuracy dropped.**
	3	**645,637**	**Train Acc: 92%** **Val Acc: 86%**	**Reduced gesture frames to 16 and also reduced LR to 0.0005.** **Validation accuracy improved also the number of trainable parameters are lesser**
	4	**357,797**	**Train Acc: 97%** **Val Acc: 91%**	**Same as model 1 but increased number of epochs to 35. Better results** **There is a spike in the loss graph at around 25th epoch. So, choosing model 1 over it**
CNN+LSTM	**5**	**680,357**	**Train Acc: 63%** **Val Acc: 61%**	**Tried CNN 2D(5-layer architecture)with LSTM. Parameters similar as Model 1 with Dropout as 0.25. Simple base model achieved but accuracy was very less.**
	6	**2,572,677**	**Train Acc: 84%** **Val Acc: 73%**	**Image size increased to 128 and number of channels to 3. Better results were seen**
	7	**4,916,869**	**Train Acc: 59%** **Val Acc: 58%**	**Increased Dense neurons to 256 . Reduced Batch Size to 25. Models' performance deteriorated. Also, around 5 million parameters, took much time to train.**
	8	**1,838,501**	**Train Acc: 88%** **Val Acc: 69%**	**Reduced image size to 64 and increased Dense Neurons to 512. Significant reduce in parameters and better accuracy achieved. Still, didn't achieve desired accuracy with CNN+LSTM**

CNN+GRU	**9**	**852,901**	**Train Acc: 97%** **Val Acc: 72%**	**Tried CNN 2D(5-layer architecture)with GRU. Parameters similar as Model 1 with Dropout as 0.25. Model overfits. LR dropped to 0.0002 at 7th epoch**
	10	**2,049,413**	**Train Acc: 99%** **Val Acc: 81%**	**Increased Image size to 128, which helped fixing overfitting a little.**
	11	**1,444,261**	**Train Acc: 76%** **Val Acc: 72%**	**Increased Batch Size to 40. This helped in fixing overfitting. However, accuracy was below 80%**
	12	**1,395,589**	**Train Acc: 82%** **Val Acc: 73%**	**This time tried with Image size 100 and Batch size 40. Better results than previous model. However, model is still overfitting**

FINAL MODEL:

We selected Conv3D model#1 as the best performing model over CNN+LSTM and CNN+GRU for following reasons:

- Training Accuracy : 95%
- Validation Accuracy: 89%
- Number of parameters are low 357,541 which makes the model computationally less expensive
- Model is Statistically simpler.