Neural Networks Project - Gesture Recognition

- Ajit John – Group Facilitator
- Ashutosh Kumar Pathak

Objective

Build a state-of-the-art model which can recognize five different gestures. List the of gestures to be recognized is mentioned below:

- 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

Business

This state-of-the-art model can be used in smart television. We want to develop a cool feature in the smart-TV that can recognize five different gestures performed by the user which will help users control the TV without using a remote.

Approach

1. **Understanding data:**

   Data consists of video frames (images) of 2-3 seconds, 30 frames per video. Each data point (video) is placed inside a folder. Further there are two folder containing train and validation sets. Size of train is 663 and size of validation set is 100. Data is balanced. Along with the data csv files are provided with all the datapoint and corresponding labels.

2. **Data Generator:**

   Data Generator is important part for training a model. Here we are going to preprocess the images and create a batch of video frames. Data Generator will perform tasks below:

   - Image Scaling and normalization: Making all frames of same size, as we have different resolutions of video frames in the dataset. After scaling images were normalized.
   - Selecting Video Frames: Generator will select video from the given input indices.
   - Augmentation: Generator will also perform some data augmentation during training. These include affine transform and cropping. This is optional.
   - Batch of Video: Finally, Generator will generator a batch of video as per the requirement.

3. **Model-Hyperparameter:**
   - Selecting Batch Size: Selecting batch size is based some constraint and system capacity. I have used batch size of 2^n in order get best performance out of GPU’s. A batch size of 32 was good for most of the 3D-CNN and a batch size of 16 for CNN-RNN model. During this experiment I have got “out of memory error” also.
   - Input resolution: Input resolution in this dataset is 60x360 and 160x120. However, while processing I tried different-different resolutions like 160x160, 100x100 & 120x120. Finally, I

used 120x120. 160x160 was too big and requires some images to scale up, while 100x100 model performance was not good. Finally, I used 120x120.

- Epochs: Deciding number of epochs is again dependent on some factors like system configuration, model performance and accuracy, time constraint. Number of epochs equal 10 seems to less, as most of the model were under performing, higher number like 20,30 was consuming more time to train. 15 seems to be good number, but I have trained few models with 20 epochs as well.
- Image Indices: Indices played a great role while training a model. Indices played a role while deciding the length of video series. Though one video consists of 30 frames, but all was not required for training. I have experimented with 30 frames, 16 frames (all alternate + last frame) and 15 frames (all alternate frames).

4. **<u>Experiments: Details of experiments pre final model training steps:</u>**

#	Model	Result	Remark
1	Conv3D	Use Batch size 32	Got Out-of-Memory multiple times
2	Conv3D Conv2D+RNN	Use image size 120x120	160x160: high computation 100x100: Accuracy not good even after 20 epochs 120x120: seems to be ideal
3	Conv3D (Data Generator)	Generator is capable of handling any number of data-points and batch size.	Testing Generator for some possiblebatch-size and data-point length
4	Conv3D (Data Generator)	Generator can handleany input indices	Tried with some possible combination of input indices, and it was able to handle.
5	Conv3D (Data Generator)	Generator produced Augmented output	Tried generating augmented data,which was handled successfully by generator.
6	Conv2D+RNN	Use Batch size of 16	Got Out-of-Memory multiple times
7	Both	Overfit	128 data points
8	Both	Improved from 128 datapoints	663 (all) data points
9	Both	Better results than 663 data points	Added more data points using Augmentation

5. **<u>Model-Building:</u>**

For analyzing, the videos sequence two types of architectures are used commonly. One is the standard CNN + RNN architecture in which you pass the images of a video through a CNN which extracts a feature vector for each image, and then pass the sequence of these feature vectors through an RNN. The other popular architecture used to process videos is a natural extension of CNNs - a 3D convolutional network. Though I have experimented with many models, I have recorded only 13 model for this assignment to keep jupyter-notebook lighter and easy to understand.

We start training models with lesser number of inputs. Initially we have used 128 datapoint for train, which in the process we increased to full 663. However, for some of the models we have used augmentation to generate and use more data points. Let's see in these experiments in details:

a. 3D-CNN:

Started with few layers (3-layers) of Conv3D, and few fully connected layers at output. Model unperformed, added few more layers followed by Dropouts and batch normalization to reduce overfitting. Model seems to be improving; however, overfitting was always an issue. I also used data augmentation to overcome this issue. Later I experimented with different sizes of filter like 3x3x3 and 2x2x2, it helped in improving the model performance. I finally used a model with filter size of 3x3x3 and 2x2x2.

Once We got a good model, I tried to reduce the number of input frames. Instead of using all 30 frames, we used 16 frames and was able to achieve similar performance. With a smaller number of input sequence, overall computation was also reduced.

Below I have listed down some of the observation in a tabular format.

#	Model	Result	Remark
1	CNN-3D	Train:98%, Test 77% Overfitted model	Use some techniques to reduce overfitting like Dropouts.
2	CNN-3D	Overfitted model	Dropouts didn't help much, though gap was narrower
3	CNN-3D	Overfitted model	Added batch-normalization, not much improvement
4	CNN-3D	Poor accuracy, no overfit	Data Augmentation used, but model accuracy was not good
5	CNN-3D	Good, ~78% accuracy no overfitting (Best outcome)	Added more layers, experimented with filter size to reduce the parameters.
6	CNN-3D	Good, ~87% accuracy no overfitting	Training with a smaller number of input sequence, used 16 frames instead of 30 frames

b. 2D-CNN & RNN:

After completing all the CNN-3D experiments, next I moved on to use CNN+RNN architecture, which is widely used for training a sequence. I stared with few Conv2D layers and LSTM layers. Model was underperforming, while we went on adding layers. Then we observed the model is overfitting and used techniques like Dropout, Batch-Normalization and Augmentation to reduce the Overfitting. On improved model we replaced LSTM with GRU, and we were able to achieve similar performance.

Next, we tried out transfer learning. We tried with VGG-16 model, but it was taking very high time to train. We tried out some others model also, but finally decided to use "mobilenet". Mobilenet was lighter and takes less time to train. We tried Mobilenet with LSTM and GRU, but finally continued with GRU and build few more models.

I have placed some the experiment in the tabular format below.

#	Model	Result	`Remark
7	CNN+LSTM	Highly overfit (87% train vs 52% validation accuracy)	Started with few cnn layers followed by LSTM. It resulted in overfit.
8	CNN+GRU	Highly overfit (95% train vs 63% validation accuracy)	Here we used GRU instead of LSTM. Resulted in overfit.
9	mobilenet + LSTM (Transfer Learning)	Overfit (99% train vs 80% validation accuracy)	We tried adding layers more layers, later we used transfer learning to improve on accuracy. It gave better result
10	mobilenet + GRU (Transfer Learning)	Overfit (98% train vs 81% validation accuracy)	Here we replaced LSTM with GRU. Got similar results.
11	mobilenet + GRU (data augmentation)	Overfit (99% train vs 78% validation accuracy)	Trained with data augmentation, model did not improve much.
12	mobilenet + GRU (Full training)	Good (98% train vs 93% validation accuracy)	Trained full network, got very good result.
13	mobilenet + GRU (Reduced sequence 15-Frames)	Good (96% train vs 95% validation accuracy)	Trained with 15 video sequences, got similar results.

Result

Based upon the model architecture type i.e., CNN-3D and CNN-RNN architecture We have selected 2 model based upon performance. Let's see the details of these models and compare these two.

Model-1: "model-00019-0.45443-0.83032-0.56978-0.80000.h5".
Model-2: "model-00020-0.05898-0.98039-0.07431-0.97000.h5".

#	Model Architecture	Number of parameters	Input	Final Accuracy	Time taken for one step
Model-1	CNN-3D	574,693	16 Frames	79%	2.11 sec
Model-2	Mobilenet+GRU	3,684,677	15 Frames	96%	1.87 sec

Based upon speed and accuracy we have selected "Model-2" as our final model. Though Model-2 has lot many parameters than Model-1, but computationally Model-2 is faster. In terms of accuracy Model-2 outperforms model-1. Model-2 usages 15-frames as input, where as Model-1 usages 16 frames as input.

Final model: Model-2: “model-00020-0.05898-0.98039-0.07431-0.97000.h5”.