**Project - Gesture Recognition**

**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 recognize five different gestures performed by the user which will help users control the TV without using a remote. 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 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.  Videos have two types of dimensions - either 360x360 or 120x160

**Data Pre-processing:**

A function named Generator was created to pre-process the data. In the generator, we are going to preprocess the images as we have images of 2 different dimensions as well as create a batch of video frames. Images were pre-processed by normalization, standard scaling, cropping to extract the best features and reduce size of images. Batch of images are fed into different CONO3D and CNN-RNN models to identify correct gesture.

**Model Building:**

We worked on multiple models with different batch size, epochs, hyper parameters and we have mentioned details below in provided table. The model which gave better performance in terms of validation accuracy was convolutional 3D with 4 convolutional layers.  Each convolutional layer was followed by batch-normalization, max-pooling layers, dropout of .25 was added after 3rd convolutional layer to reduce over fitting. At last output from these layers was flattened, drop-out and fed into soft-max layer with 5 outputs.

**Different Model Variants Used:**

1. Conv3D and MaxPooling3D
2. Time Distributed while building a Conv2D + RNN model

**Model Performance:**

As per summary of model, it has total 9,439,365 trainable parameters. We have designed the network in such a way that the model is able to give good accuracy on the least number of parameters so that it can fit in the memory of the webcam. Training Accuracy (78.9%) and **Validation accuracy is 77%** with 40 epochs, 32 batch size and we have noticed that with increase in number of epochs accuracy goes up but with limited resources we have stopped it on 40 epochs.

**Below are the details of different models tried before reaching to final CONV3D model mentioned at end:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Experiment Number** | **Model** | **Result** | **Decision + Explanation** |
| **1** | **Conv3D** | **Throws Resource exhausted error** | When we tried to train the model with 3 convo 3D layers with 26 frames of size 120\*120 it throws resource exhausted due to lot of parameters and not enough memory. Batch size 50 and epochs 40. |
| **2** | **Conv3D** | **Unstable model.**  **High ups and downs in accuracy in subsequent epochs and between training and validation accuracy also.** | Adjusted the cropping size of image which was varying from 80 to 120. Difference in final accuracy didn’t change much after correction.  Training accuracy reaching 90s but validation accuracy in 50s. |
| **3** | **Conv3D** | **No apparent difference in accuracies** | Reduced the frames from 30 to 25 to see if validation accuracy come closer to train accuracy. |
| **4** | **Conv3D** | **Training accuracy: 85**  **Validation accuracy: 68** | Filter size increased from 1, 3, 3 to 3, 3, 3. Model was better than the last model. |
| **5** | **Conv3D** | **Training loss seemed to increase with every epoch. Training accuracy dropped down to 60s.** | Tried to reduce number of layers but the loss increased with every epoch. Disregarded the decision. |
| **6** | **Conv3D** | **Training accuracy 86%**  **validation accuracy 53%** | Increased batch size to 30 and changed the SGD optimizer to Adam. Training time reduced. But validation accuracy wasn’t impacted much. |
| **7** | **Conv3D** | **Training accuracy 85%**  **Validation accuracy 75%** | Padding was changed to “Same”. Still looks over fit. |
| **8** | **Conv3D** | **Decreased model accuracy** | To generalize the model, increased dropouts to 0.5 in all convolutional layer but it worsened the model accuracy. |
| **9** | **Conv3D** | **Training accuracy:75%**  **Validation accuracy: 68%** | Kept the dropout in dense layers of 0.5 and one dropout of 0.25 after last convolutional layer. Added the batch normalization layers. |
| **10** | **Conv3D** | **Training accuracy: 84%**  **Validation accuracy: 63%** | To stabilize the model, modified the learning rate parameters in Reduce LR on plateau. min\_delta=0.001, patience=2 |
| **11** | **Conv3D** | **Validation Accuracy: 62%** | Crop the images correctly, try to over fit on less amount of frames.  Reduce the size of the image/Reduce the number of layers. Tried to over fit with 3 convo 3D layers and worked with 10 epochs was able to get 62% accuracy. It has 3 convolutional layers followed by batch normalization and max pooling. Later it was fed into dense layer with 5 outputs after passing through flattened and dropout.  Increase the amount of trainable data and let’s try to add more frames and reduce the size of image to extract only features which are of our importance. |
| **12** | **ConvGRU** | **Train Accuracy : 60%**  **Validation Accuracy: 50%** | With 40 epochs and batch size 32. Total  2.8 million Trainable parameters. 6 convolutional 2D layer with time distributed batch normalization followed by max pooling 2D. We have updated filter size at convolutional first layer to 7, 7 but there was no much improvement in accuracy. And model seem to over fit. |
| **13** | **ConvGRU** | **Accuracy : 58%**  **Train accuracy : 85%** | With 4 convolutional layer followed by time distributional batch normalization and max pooling it has given accuracy for validation as 58% only with 35 epochs. Trainable parameters were 6 million but model seems to be over fitting. We will try other variants as well to see how it works. |
| **14** | **ConvLSTM** | **Validation Accuracy : 61%**  **Train Accuracy: 80%** | After few models of conv3D we started with convLSTM to see how it works. We started with 3 layers of conv2d followed by time distributed batch normalization and max pooling 2D. Later it sends output to LSTM thru dropout. Model was executed on 30 epochs with batch size 40. It has 26 million trainable parameter. Model was over fitted with train accuracy of 80 and validation accuracy of 61. Train loss was getting reduced but validation loss after some point of time did not improve but started increasing. |
| **15** | **Conv3D** | **Train Accuracy : 73.05%**  **Validation Accuracy: 73%** | With 40 epochs and batch size 32.  9.5 million Trainable parameters and filter size for first layers was updated to 7\*7, we got good accuracy without over-fit. There are 4 convolutional layer followed by batch normalization and max pooling. Dropout has been added after last convolutional layer to reduce over fit. Let’s experiment with more different parameters and different filters. |
| **16** | **Conv3D** | **Validation Accuracy: 73%**  **Train Accuracy : 76%** | After multiple attempts on con3D now we are getting good model with good accuracy which doesn’t seems to be over fitting much. It has given us **73% accuracy** with 30 epochs and trainable parameters are close to 9.4 million which seem to good one even for web cam hosting. Images were cropped as well before feeding into model to reduce size and improve performance. There are 4 convolution 3D layers used followed by batch normalization and max pooling 3D. Later it was fed into soft max with 5 outputs after being flattened and dropout. Batch normalization after each convolutional layer has improved training and validation accuracy significantly. |
| **17** | **ConvLSTM** | **Validation Accuracy: 46%** | We tried convLSTM model with 3 layers of convolutional layers 2D without batch normalization. Accuracy that it gave for 10 epochs was **46%** with 27 million trainable parameters and it also didn’t show much improvements from one epoch to another and seemed stuck at one point.  Let’s add batch normalization as well after each con2D layer and add one additional conv2D in same model to see if it improved performance. |
| **18** | **ConvLSTM** | **Validation Accuracy: 66%** | We tried convLSTM another model to see if accuracy improves further. Added time distributed batch normalization after each four convolutional layer followed by max pooling 2D. It did improved **accuracy to 66%** with 35 epochs. Trainable parameters were reduced to 5 million. So actually it did improve from last model where we had used only 3 convolutional layers. Let’s try some more variants. |
| **19** | **Conv3D** | **Validation Accuracy: 76%**  **Train Accuracy: 81.5%**  **(Seems bit over fit)** | After failed attempts of multiple CONV3D models finally we were able to get good model with **76% accuracy** with 35 epochs.  Model has 4 layer of conv3d and each convolutional layer is followed by batch normalization, max pooling and fed into soft max layer with 5 output. It has total 9,439,365 trainable parameters |
| **Final Model** | **Conv3D** | **Train Accuracy: 78.97%**  **Validation Accuracy: 77.00%** | * So finally we have got very good model using CONVO 3D with below parameters: * 40 epochs and 32 batch size. * We have used 25 frames of size 80\*80 with 3 channels. * Filter that we used were 3\*3 with each convolutional layer. * Frame size was reduced/cropped to 80\*80 to extract fine features only. * As we can see we got ***categorical\_accuracy: 0.7897*** and ***val\_categorical\_accuracy: 0.7700*** * This accuracy we got after trying multiple models with different parameters. * Total trainable parameters ***9,439,365*** which is even good for webcam hosting. * This model has 4 convolutional 3D layer followed by batch normalization, max pooling. * 0.25 dropout has been added after third convolution layer to reduce over fitting. * Later it is flattened followed by dropout and fed into dense layer with 5 classification outputs. |