Hand-gesture-recognition-deep-learning

Project to recognize hand gesture using state of the art neural networks.

Team

Archana Shastri

Subbarao M

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

Each video is a sequence of 30 frames (or images)

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.

The file contains a 'train' and a 'val' folder with two CSV files for the two folders.

Model Overview

Model Name	Model Type	Number of parameters	Augment Data	Model Size(in MB)	Highest Validation accuracy	Corresponding Training accuracy	Observations
conv_3d1_model	Conv3D	1,117,061	No	NA	78%	99%	Model is over-fitting. Augment data using cropping
conv_3d2_model	Conv3D	3,638,981	Yes	43.8	85%	91%	Model is not over-fitting. Next we will try to reduce the parameter size. Moreover since we see minor oscillations in loss, let's try lowering the learning rate to 0.0002
conv_3d3_model	Conv3D	1,762,613	Yes	21.2	85%	83%	Model has stable results .Also we were able to reduce the parameter size by half. Let's trying adding more layers at the same level of abstractions
conv_3d4_model	Conv3D	2,556,533	Yes	30.8	76%	89%	With more layers added model is over-fitting. Let's try adding dropouts at

							the convolution layers
conv_3d5_model	Conv3D	2,556,533	Yes	30.8	70%	89%	Adding dropouts has further reduced validation accuracy as its not to learn generalizable features and its further over-fitting
conv_3d6_model	Conv3D	696,645	Yes	8.46	77%	92%	Reducing the number of network parameters by reducing image resolution/ filter size and dense layer neurons. Comparably good validation accuracy
conv_3d7_model	Conv3D	504,709	Yes	6.15	77%	85%	
conv_3d8_model	Conv3D	230,949	Yes	2.87	78%	86%	
rnn_cnn1_model	CNN-LSTM	1,657,445	Yes	20	75%	92%	Model is over-fitting. Let's try reducing the number of layers in next iteration

Models with More Data Augmentation

Model Name	Model Type	Number of parameters	Augment Data	Model Size(in MB)	Highest validation accuracy	Corresponding Training accuracy
conv_3d10_model	Conv3D	3,638,981	Yes	43.8	86%	86%
conv_3d11_model	Conv3D	1,762,613	Yes	21.2	78 %	79 %
conv_3d12_model	Conv3D	2,556,533	Yes	30.8	81%	84%
conv_3d13_model	Conv3D	2,556,533	Yes	30.8	31%	78%
conv_3d14_model	Conv3D	696,645	Yes	8.46	77%	87%
conv_3d15_model	Conv3D	504,709	Yes	6.15	75%	82%
conv_3d16_model	Conv3D	230,949	Yes	2.87	76%	77%
rnn_cnn2_model	CNN-LSTM	1,346,021	Yes	31	78%	96%

Transfer Learning Models (CNN + RNN)

Mobilenet model is considered as its parameter size is less compared to Inception and Resnet models

Model Name	Number of parameters	Augment Data	Model Size(in MB)	Highest validation accuracy	Corres-ponding Training accuracy	Observations
rnn_cnn_tl_model	3,840,453	Yes	20.4	56%	85%	For this experiment, Mobilenet layer weights are not trained. Validation accuracy is very poor. So let's train mobilenet layer's weights as well
rnn_cnn_tl2_model	3,692,869	Yes	42.3	97%	99%	We get a better accuracy on training mobilenet layer's weights as well.