GESTURE RECOGNITION – NEURAL NETWORK PROJECT

Adinath Jambulkar
Hemant Sharma

DSC35

Problem Statement:-

Suppose we are working as a data scientist at a home electronics company which manufactures state of the art smart televisions. 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.

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

Dataset Link:- https://drive.google.com/uc?id=1ehyrYBQ5rbQQe6yL4XbLWe3FMvuVUGiL

Understanding the Dataset:-

The As we know training dataset have few hundred videos files which are categorized into 5 classes and every video is two to three second in duration which are then divided into 30 images and all videos are recorded by different peoples and perform one of five gesture in front of webcam as similar to smart TV

Objective

Our main task for this project is training of different models and do prediction on the basis of action which is performed in every sequence of videos as well as on validation dataset. Finally we will do evaluation on test dataset and compare performance.

Types of architectures :-

1. 3D - Convolutional NN i.e. (Conv3D)

3-D convolution is just an natural extension of 2-D which we already familiar. As we know we move filter in 2 directions x and y in case of 2D but in case of 3D we move in 3 directions x,y,z and in this case our input is video of 30 RGB images where there is an assumption of 100x100x30 size of images. Suppose there is a video of 4D tensor 100x100x3x30 which rewrite as (100x100x30)x3 where 3 is No. of channel. Therefore 3D filter is represented as (fxfxf)xc since images have 3 channels so c=3.

2. CNN and RNN Architectures

A conv-2D is used to fetch feature vector of every images and then send the sequence to RNN network and output of RNN is a kind of regular softmax.

Data-Generators:-

This is very important part and in this we will preprocess images because we do have images in two different dimensions i.e 360x360 and 120x160 and create a video frames and generator take this as an input.

Pre-Processing:-

At the last stage to improve accuracy we will do data augmentation, here we will little rotate the preprocess images of the gestures so that more data will come for model training and make more generalizable.

Normalizing is just an RGB value of image and its an good way o rid of distortions coming from shadows and lights in images

Mainly resize and cropping of any image is done because NN recognizes all gestures nicely not on focusing on noise which are there in images.

Observations:-

Initially we built a random model and then we will improve that model with hyper-parameter. We have an input of images in sequence which is video and we consider just half of images which reduces memory consumption. Below table will represent all models along with their results.

Experiment Number	Model	Result	Decision + Explanation
1	**Conv3D**	**Training acc: 97.8% Val acc: 83%**	Little overfitting Initially because of not using any drop-outs and now we will check next model with 128 batch size.
2	**Conv3D**	**Training acc: 82% Val acc: 30 %**	We get lot of overfittings and thebatch size which we took i.e., 128 is not an optiomal one. Now we will reduce batch size and go for batch normalization.
3	**Conv3D**	**Training acc: 99.50% Val acc: 85%**	We do reduce our batch size to 64 and which reduce to overfitting and now we will go for further reducing our batch size and we will see outcome.
4	**Conv3D**	**Training acc: 99.4% Val acc: 75%**	We reduced our batch size again to 32 and rest all same and we found out this batch size is optimal size for both train and validation data.
5	**Conv3D**	**Training acc: 99.7% Val acc: 82%**	We add one hidden layer more to increase accuracy of our model.
6	**Conv2D**	**Training acc:71 % Val acc: 67%**	We converted images in Grey-Scale Format which result in model size shrinkage and due to which huge information reduces which makes model training difficult.
7	**Conv2D**	**Training acc:83 % Val acc: 65 %**	To reduce the overfitting we used dropouts and then increasethe hidden layer to increase accuracy. Now we will try out CNN_RNN stack model.
8	**ResNet50 + GRU**	**Training acc:48 % Val acc: 44%**	Because parameters and depth are less that's why model is not capable to learn itself Model is not able to learn on training dataset itself. Increase hidden layer and will see accuracy.

9	**ResNet50 + GRU**	**Training acc:44 % Val acc: 43%**	Model not fit well even we increase the complexity in model so we reject this model because ResNet50 pre training is not good. In next step we will look into ResNet50V2 pre training model and see improvements.
10	**ResNet50V2 + GRU**	**Training acc:100 % Val acc: 75%**	Now we will increase depth of ResNet50V2 which will give higher accuracy than previous. But we will see little over fitting because of complexity
11	**ResNet50V2 + LSTM**	**Training acc:100 % Val acc: 79%**	Now we will try LSTM and we will check our accuracy of model and see what will be change in performance.
12	**ResNet50V2**	**Training acc:100 % Val acc: 75%**	Now we will try advance ResNet50V2 and we will check our accuracy of model and see what will be change in performance.
13	**ResNet50V2 + GRU**	**Training acc:95 % Val acc: 70%**	Here we can find the problem of overfitting is not not solved

From above all model and their performance we can say that the model no 3 with Training acc: 99.50% & Val acc: 85% is the best model

Thanks ☺