Anil Kumar

C18 Cohort PGDDS

Abstract

Deep learning model built on video data to recognize the gestures and perform necessary tasks

Project- Gesture Recognition

Deep Learning Course – IV

Contents

[Problem Statement: 2](#_Toc63622190)

[Data set Explanation: 2](#_Toc63622191)

[Initial Planning: 2](#_Toc63622192)

[Data Set Exploration: 3](#_Toc63622193)

[Data Generator for Model Training: 4](#_Toc63622194)

[Model Architecture: 6](#_Toc63622195)

[Model Training: 7](#_Toc63622196)

[Conclusion & Observations: 9](#_Toc63622197)

## **Deep Learning Course – IV: Project- Gesture Recognition**

# Problem Statement:

A home electronics company which manufactures state of the art smart televisions wants 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

# Data set Explanation:

The data contains a 'Train' and a 'Val' folder with two CSV files having the links for the two folders. These folders are in turn divided into subfolders where each subfolder represents a video of a particular gesture. Each subfolder, i.e. a video, contains 30 frames (or images). Note that all images in a particular video subfolder have the same dimensions but different videos may have different dimensions. Specifically, videos have two types of dimensions - either 360x360 or 120x160

# Initial Planning:

Using two architectures and get the best possible fit model.

* **CNN using 3D convolutions:** Using only CNN model where we use 3D convolution with 3D filters and 3D pooling etc, here convolution is used by 3D filters to capture all essential features from all the frames of a video and map it to appropriate label by training.
* **CNN 2D convolutions + RNN:** Using this architecture we will first create the feature maps from each image in a video and we will feed to the RNN (sequential architecture) in frame after frame manner by using the time distributed technique so the architecture will have a sequential input and a label to it for training.

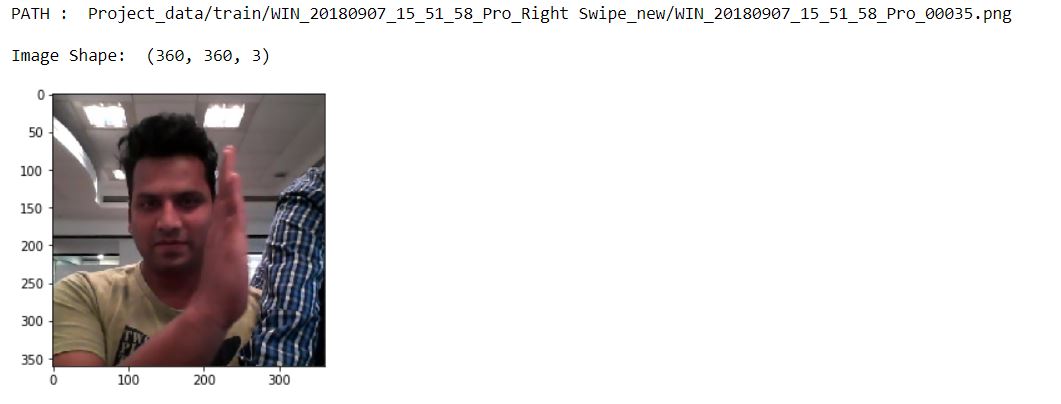
# Data Set Exploration:

Each video has 30 frames, and each video represents a certain gesture which implies certain activity. The gestures video is collected combinations of different backgrounds, different persons and different sizes.

For example:

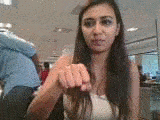
Below image is of size (120,160,3)

Below image is of (360,360,3)



Each video having similar kind of images of 30 each, i.e 30 frames,

Below is the way every gesture is recorded.



# Data Generator for Model Training:

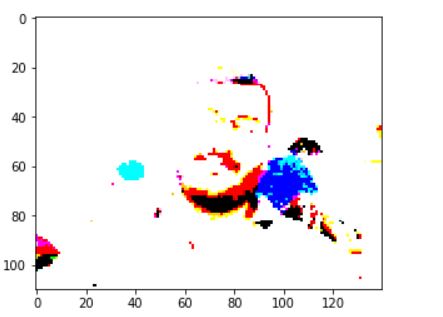
Since we always have the hardware constraints and computing limitations in training the complex neural network architectures. The data generators always come to the rescue. By using the data generators, we feed the data to the network in batches as per the requirement of network which makes the machine to use memory efficiently.

Used two generator function to try different approach:

* Generator1:
  + Total number of frames used is: 30
  + Dimensions considered for each image is: 50,50
  + No cropping used.
  + Normalizing the image by percentile method
    - Before normalizing:



* + - After normalizing by percentile:



* + If the data falls below the batch size written an additional condition to yield the custom batch of data.
* Generator 2:
  + Total number of frames used is: 30
  + Dimensions considered for each image is: 120,160
  + Cropping the image sized (120,160) to remove upper image portion and side backgrounds by 10 and 20 units.
    - Before cropping:



* + - After cropping:



* + Normalizing the image by pixels method: which yielded a good result and hence chosen this approach
    - Before normalizing:



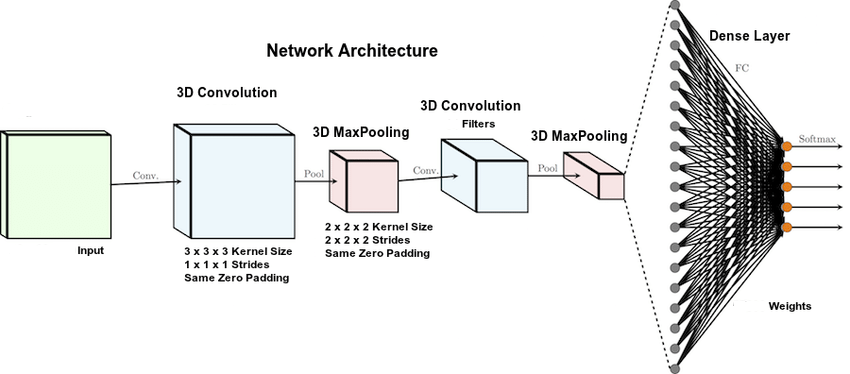
* + - After normalizing by pixels:



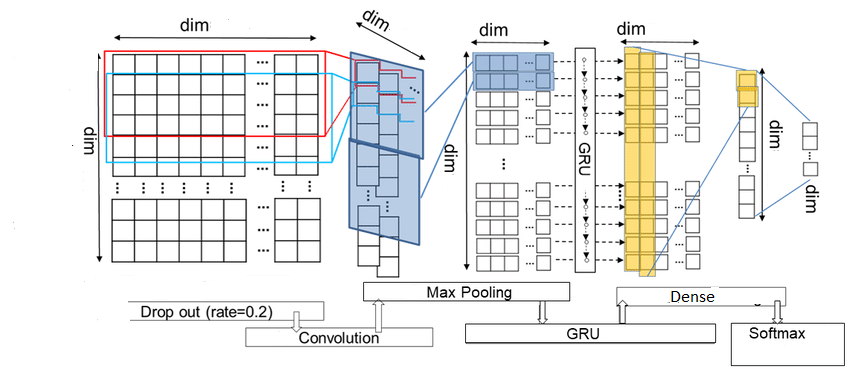
* + If the data falls below the batch size written an additional condition to yield the custom batch of data.

# Model Architecture:

**CNN3D**:



**CNN 2D + GRU**:



# Model Training:

* Tring with CNN 3D convolutions:
  + model no 1
    - 3 convolution layers followed by maxpooling3d
    - Number of cnn filters used 8,16,32,64
    - Size of filters used (5,5,5) , (3,3,3),(3,3,3)
    - 3 dense layers added with (128,64,5) neurons.
    - First two dense layers have a dropout of 25%
    - Last dense layer kept with SoftMax activation for 5 labels classification
    - Adam optimizer used
    - Model checkpoint used to monitor validation loss
    - For Learning rate ReduceOnPlateau is used.
    - Batch size: 64 (Also tried with 90,128,256)
    - Epoch: 50
    - Total number of parameters model has : 462325
    - Validation loss: 1.08
    - Validation accuracy: 40%
  + model no 2
    - 3 convolution layers followed by maxpooling3d
    - Number of cnn filters used 8,16,32,64
    - Size of filters used (3,3,3),(2,2,2)
    - 3 dense layers added with (256,128,5) neurons.
    - First two dense layers have a dropout of 25%
    - Last dense layer kept with SoftMax activation for 5 labels classification
    - Adam optimizer used
    - Model checkpoint used to monitor validation loss
    - For Learning rate ReduceOnPlateau is used.
    - Batch size: 64
    - Epoch: 50
    - Total number of parameters model has: 231781
    - Validation loss: 0.71
    - Validation accuracy: 71%
  + model no 3 (Final Model)
    - 3 convolution layers followed by maxpooling3d
    - Number of cnn filters used 8,16,32,64
    - Size of filters used (5,5,5) , (3,3,3),(3,3,3)
    - 3 dense layers added with (128,64,5) neurons.
    - First two dense layers have a dropout of 25%
    - Last dense layer kept with SoftMax activation for 5 labels classification
    - Adam optimizer used
    - Model checkpoint used to monitor validation loss
    - Batch size: 64
    - Epoch: 50
    - Total number of parameters model has: 1094613
    - Validation loss: 0.58
    - Validation accuracy: 80%
  + model no 4 trained using generator 2:
    - 3 convolution layers followed by maxpooling3d
    - Number of cnn filters used 8,16,32,64
    - Size of filters used (5,5,5) , (3,3,3),(3,3,3)
    - 3 dense layers added with (256,128,5) neurons
    - First two dense layers have a dropout of 35%
    - Last dense layer kept with SoftMax activation for 5 labels classification
    - Adam optimizer used
    - Model checkpoint used to monitor validation loss
    - Batch size: 16
    - Epoch: 30
    - Total number of parameters model has: 1208165
    - Validation loss: 1.08
    - Validation accuracy: 60%
* Training with CNN 2D convolution and RNN:
  + model details using generator 1
    - 3 convolution time distributed layers followed by batch normalization and flatten layers
    - Number of cnn filters used 8,16,32,64
    - Size of filters used (3,3),(2,2),strides used(2,2) , (1,1)
    - 3 dense layers added with (128,64,5) neurons
    - First two dense layers have a dropout of 25%
    - Sequential layer kept with 128 neurons for the sequential data from conv2d
    - Last dense layer kept with SoftMax activation for 5 labels classification
    - Adam optimizer used
    - Model checkpoint used to monitor validation loss
    - Batch size: 64
    - Epoch: 50
    - Total number of parameters model has : 2204045
    - Validation loss: 0.88
    - Validation accuracy: 75%
  + model details using generator 2
    - 3 convolution time distributed layers followed by batch normalization and flatten layers
    - Number of cnn filters used 8,16,32,64
    - Size of filters used (3,3),(2,2),strides used(2,2) , (1,1)
    - 3 dense layers added with (128,64,5) neurons
    - First two dense layers have a dropout of 25%
    - Sequential layer kept with 128 neurons for the sequential data from conv2d
    - Last dense layer kept with SoftMax activation for 5 labels classification
    - Adam optimizer used
    - Model checkpoint used to monitor validation loss
    - For Learning rate ReduceOnPlateau is used.
    - Batch size: 16
    - Epoch: 30
    - Total number of parameters model has: 18,567,501
    - Validation loss: 0.88
    - Validation accuracy: 7

# Conclusion & Observations:

* Increase in batch size pushes the GPU to run faster and we can try to find best batch size the machine can handle.
* Model check point is very useful to keep the check on the model and saving the model every time the model performs better.
* ReduceOnPleatue used the optimized learning rate for training, Some times a constant learning rate also gives good results.
* Conv3D performs equivalently good compared to sequential (Cnn2D + RNN) model with smaller number of parameters
* Trail run gives the complete idea of network and the possibility of fitting on large data.
* Final model is decided on the grounds of validation loss and validation accuracy.
* The final model has 80% accuracy and a loss of 0.58.
* 