Gesture Recognition Case Study

Dwaraganathan DK, Suganya Thirumurthy

In this case study, you are going to build a conv2D+ RNN model and 3D Conv model that will be able to predict the 5 gestures correctly.

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.

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

Goals of this Project:

In this project, you will build a model to recognize 5 hand gestures. The starter code has been shared with you.

You need to accomplish the following in the project:

Generator: The generator should be able to take a batch of videos as input without any error. Steps like cropping, resizing and normalization should be performed successfully.

Model: Develop a model that is able to train without any errors which will be judged on the total number of parameters (as the inference(prediction) time should be less) and the accuracy achieved.

Write up: This should contain the detailed procedure followed in choosing the final model. The write up should start with the reason for choosing the base model, then highlight the reasons and metrics taken into consideration to modify and experiment to arrive at the final model.

Generator:

First step is to write a generator function.

Steps performed in the generators:

- **Crop Image**

  - First image is translated using OpenCV function
  - Then it is converted into gray-scale image.
  - Centre cropping is performed with width and height calculation.

- **Resized image**
  - Cropped Image is resized using the width and height hyper-parameter.
- **Normalizing Image**
  - The cropped image is normalized by dividing the image by 255.

With the given batch_size parameter, number of batches are calculated. For the given batches, image processing is done. Remaining batch size is calculated, and image processing is done for the left-out image batch.

Generator function is re-written such that all these processing happens for one chunk of batch data which contains images.

Model:

As per requirement we used both Conv3D model and Conv2D + RNN model.

Conv3D Model:

For this model, we used various layers in the model. It has Conv3D layer, 'relu' activation layer, batch normalization layer and MaxPooling3D layer with filter size of (2,2,2). It also has Flatten, Dropout, Dense and 'softmax' layer.

We selected 'Adam' optimizer which gave better performance than SGD. For loss, we used categorical_crossentropy and categorical_accuracy as metrics.

Hyper-parameter Tuning:
Some of the hyper-parameters tuning followed in this model are:
Image size (height, width)
Batch Size
Frames to sample

SI.No	Model + Hyper-Parameters	Categorical Accuracy	Details
1	16,32,64,128 layers, 160X160, BS=40, FTS=30	ResourceExhaustedError	We hit the limit on memory resources. So the image size selected is not correct.

2	16,32,64,128 layers, 160X160, BS=40, FTS=16	Training accuracy – 59% and validation accuracy – 15%	Model didn't perform well
3	16,32,64,128 layers, 160X160, BS=30, FTS=30	Training accuracy – 56% and validation accuracy – 18%	Model didn't perform well
4	16,32,64,128 layers, 100X100, BS=60, FTS=30	Training accuracy – 56% and validation accuracy – 20%	Model didn't perform well
5	16,32,64,128 layers, 100X100, BS=60, FTS=16	Training accuracy – 52% and validation accuracy – 17%	Model didn't perform well
6	16,32,64,128 layers, 100X100, BS=60, FTS=16	Training accuracy – 52% and validation accuracy – 17%	Model didn't perform well
7	16,32,64,128 layers, 100X100, BS=80, FTS=16	Training accuracy – 55% and validation accuracy – 14%	Model didn't perform well
8	16,32,64,128 layers, 160X160, BS=15, FTS=30	Training accuracy – 53% and validation accuracy – 15%	Model didn't perform well
9	16,32,64,128 layers, 160X160, BS=15, FTS=16	Training accuracy – 61% and validation accuracy – 23%	It has improved little bit
10	16,32,64,128 layers, 100X100, BS=15, FTS=16	Training accuracy – 54% and validation accuracy – 23%	Model didn't perform well
11	16,32,64,128 layers, 100X100, BS=10, FTS=16	Training accuracy – 55% and validation accuracy – 21%	Model didn't perform well
12	16,32,64,128 layers, 100X100, BS=10, FTS=30	Training accuracy – 50% and validation accuracy – 24%	Model didn't perform well
13	16,32,64,128 layers, 160X160, BS=10, FTS=30	Training accuracy – 56% and validation accuracy – 15%	Model didn't perform well
14	16,32,64,128 layers, 160X160, BS=10, FTS=16	Training accuracy – 54% and validation accuracy – 22%	Model didn't perform well
15	16,32,64,128 layers, 160X160, BS=40, FTS=16	Training accuracy – 62% and validation accuracy – 20%	Training accuracy has improved but validation accuracy has reduced.

As we see from the above experiments image resolution and number of frames in sequence have more impact on training time than batch_size

So experimentations are carried with batch size fixed around 15-40 and changing the resolution and number of image per sequence. Models are designed such that their memory consumption is less.

When the image resolution is greater, then we get ResourceError.

When the Data is augmented, then the performance was better. So data augmentation was done to the model to get better performance.

SI.No	Model + Hyper-Parameters	Categorical Accuracy	Details
1	Conv3D Model with no data augmentation (Batch Size=40) Image Size = (160,160) FTS = 20	Accuracy – 97% and validation accuracy is 26%	Model is overfitting
2	Conv3D Model with data augmentation (Batch Size=20) Image Size = (160,160) FTS = 20	Accuracy – 79% and validation accuracy is 79%	Accuracy value has dropped but
3	Conv3D Model with more layers and different filter size (Batch Size=30) Image Size = (120,120) FTS = 16	Accuracy – 80% and validation accuracy is 62%	There is some improvement in accuracy
4	Conv3D Model with more layers (Batch Size=20) Image Size = (120,120) FTS = 16	Accuracy – 76% and validation accuracy is 75%	There is decrease in training accuracy but validation accuracy have increased.
5	Conv3D Model with more layers and data augmentation(Batch Size=20) Image Size = (160,160) FTS = 20	Accuracy – 78% and validation accuracy is 83%	Model is best till now with better performance for the validation accuracy.

Model : CNN + RNN

For this model, we have used various layers in the model. It has Conv2D layer with Time distributed, 'relu' activation layer, batch normalization layer and MaxPooling2D layer with filter size of (3,3). It also has Flatten, Dropout, Dense and 'softmax' layer. I tried with LSTM

We selected 'Adam' optimizer which gave better performance. For loss, we used categorical_crossentropy and categorical_accuracy as metrics.

SI.No	Model + Hyper-Parameters	Categorical Accuracy	Details
1	Conv2D model + BS=40, FTS=30 IS=(120,120) + LSTM	Accuracy – 64% and validation accuracy is 55%	Model isn't performing well
2	Conv2D model + BS=40, FTS=30 IS=(120,120) + GRU	Accuracy – 78% and validation accuracy is 64%	Results are better.

With CNN + RNN, the performance was not better for my models with various tuning of hyperparameters.

So I tried transfer learning as mentioned in the Project description. I used mobileNet for my Model

SI.No	Model + Hyper-Parameters	Categorical Accuracy	Details
1	Transfer learning Model (MobileNet)	Accuracy – 98% and validation accuracy is 72%	Model is overfitting.
2	Transfer learning Model (MobileNet) with GRU	Accuracy – 99% and validation accuracy is 98%	Best model so far.