# Gesture Recognition Project

We proceed experimenting with our network by tuning the input parameters:

1. **Input image size** - a uniform input resolution is maintained based on resource availability , training time and model performance
2. **Batch size** - model is trained in batches so that the available memory is efficiently utilized
3. **Number of images per sequence** - given 30 image per sequence, its down sampled based on resource availability , training time and model performance

## Constraints

### Resource Constraint

When it comes to memory utilization, limits on resource availability is simulated by varying the input parameters. There is a limit on memory resources with image resolution of 160x160 with 30 frames and batch\_size of 40 as observed by the below error,

*ResourceExhaustedError: OOM when allocating tensor with shape[40,16,30,160,160] and type float on /job:localhost/replica:0/task:0/device:GPU:0 by allocator GPU\_0\_bfc(with a filter size of 16 on first layer)*

So trade off needs to be maintained between the input resolution , batch size and number of images per sequence for proper resource utilization

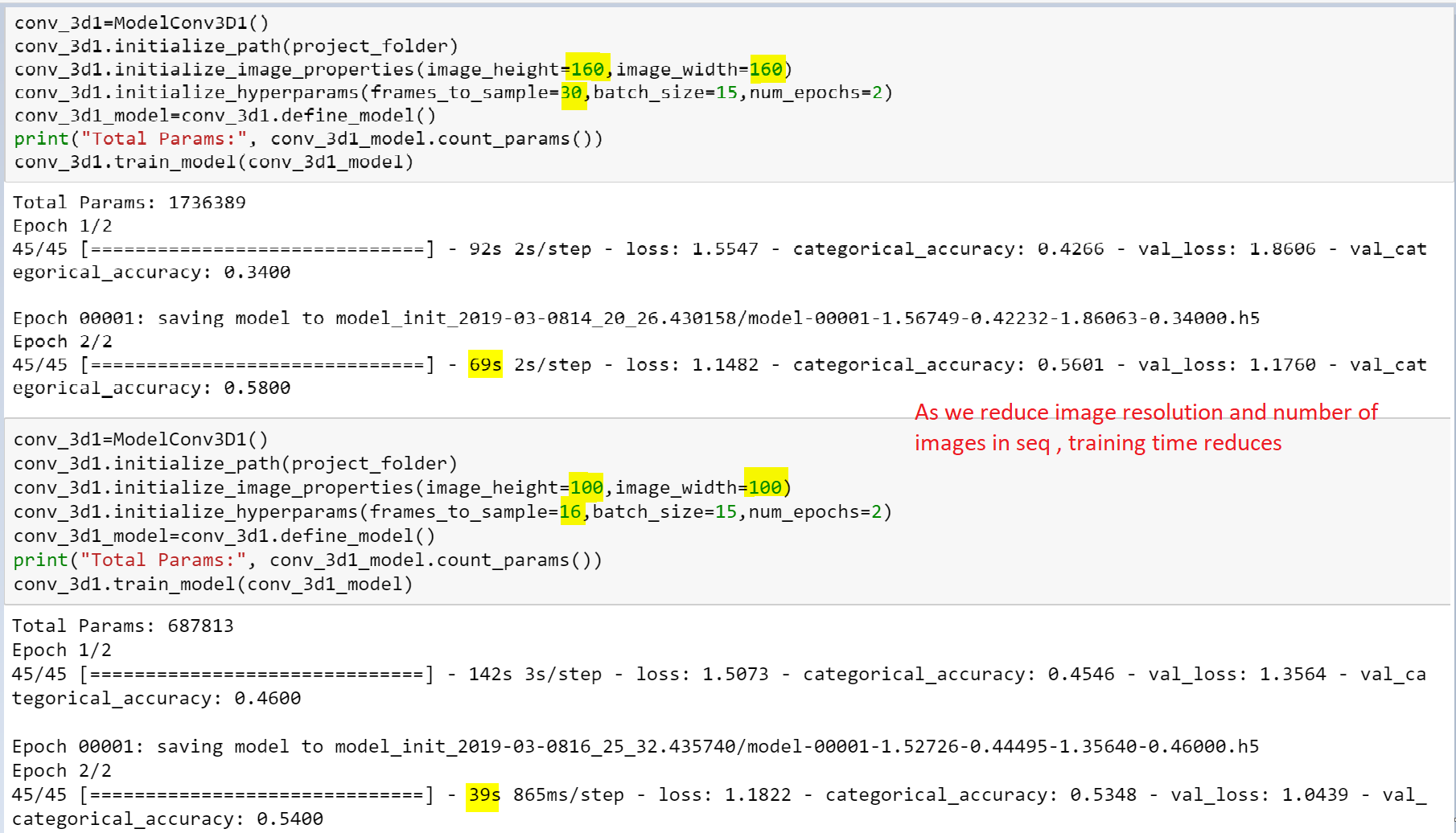
### Device Constraint

With webcam memory constraint in place, models are created in such away that the total number of parameters are fewer in memory footprint i.e targeting the model size typically between 5 to 50 MB in size. 50 MB model corresponds to 4.3 million parameters assuming the datatype size of parameters to be 12 bytes

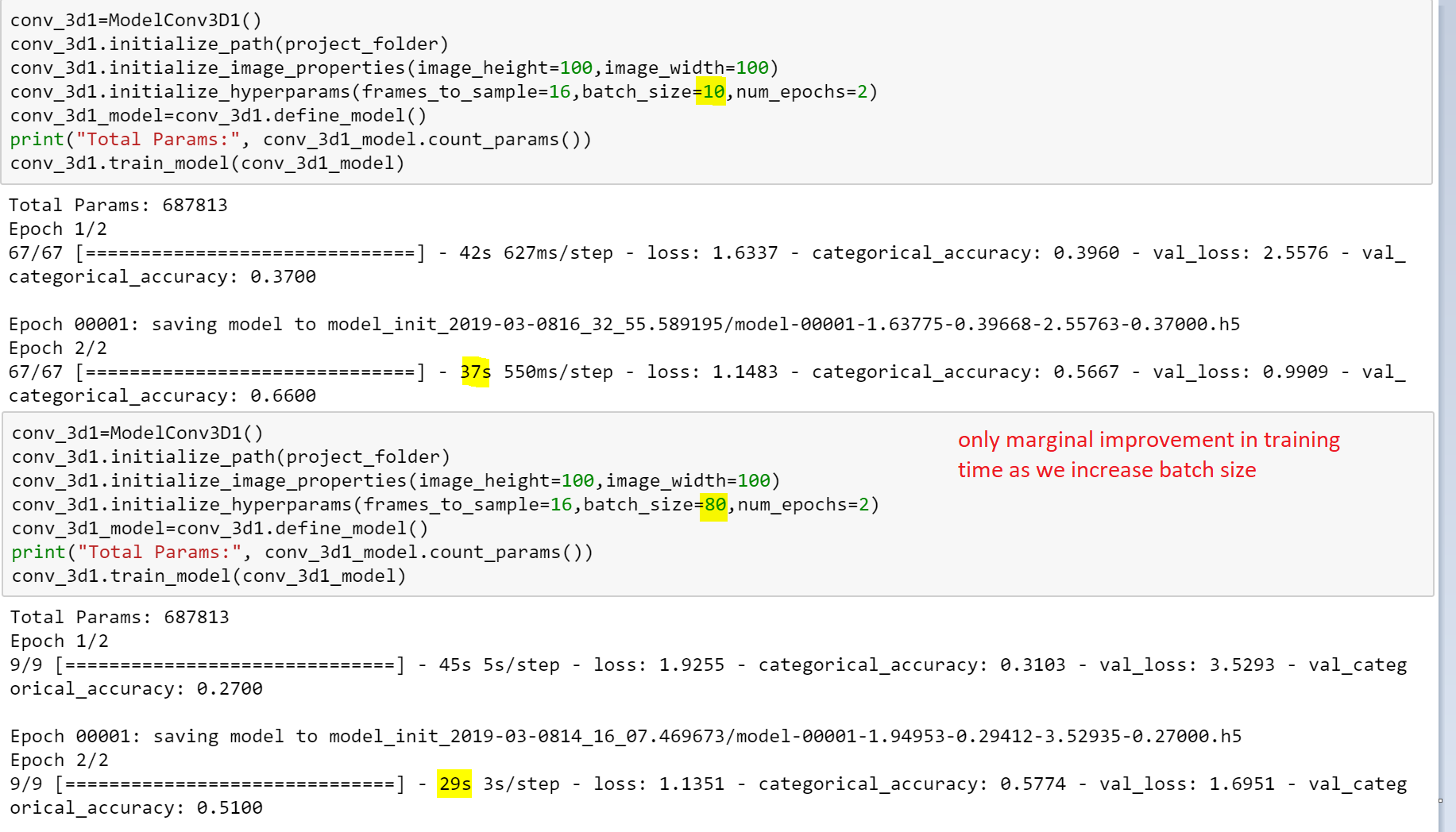
So based on the above 2 constraints we cannot have too shallow network in which case number of parameters in the network will shoot up and we cannot learn more abstract features. At the same time we cannot have too deep networks as we are constrained with less number of images per sequence and low resolution image (because of the pooling operation). Convolutions can be stacked together to increase the width in each layer but that would increase the parameter size as well and might over-fit as well since our training data is not that enormous. Moreover more the convolution layers more the operations needed and hence we will not be able to perform user actions in real-time.

So accordingly we are fixing the conv layers to 4-5.

Based on the initial experiments with the input params its clear that as we decrease the input resolution and number of images, training time reduces.



Also its observed that there is only a marginal improvement in training time with respect to batch size.



So experimentations are carried with batch size fixed around 15-40 and changing the resolution and number of image per sequence based on the device memory constraints

## Model Overview

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Model Name | Model Type | Number of parameters | Augment Data | Model Size(in MB) | Highest Validation accuracy | Corres-ponding 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 |

Let’s augment the data even further with slight rotation as well and see if we can get better results,

## 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% |

Clearly with more data augmentation, over-fitting is reduced .However accuracy hasn’t improved drastically

Let’s see how models perform on transfer learning

## 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. |

## Consolidated Final Models :

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Model Name | Model Type | No of params | Augment Data | Model Size(in MB) | Highest validation accuracy | Corres-ponding Training accuracy | Remarks |
| Model 8 -  conv\_3d8\_model | Conv3D | 230,949 | Yes | 2.87 | 78% | 86% | Low Memory footprint |
| Model 10 -  conv\_3d10\_model | Conv3D | 3,638,981 | Yes | 43.8 | 86% | 86% | - |
| Model 19 -  rnn\_cnn\_tl2\_model | CNN + RNN  Transfer Learning used on CNN | 3,692,869 | Yes | 42.3 | 97% | 99% | Best Model achieved in terms of accuracy |