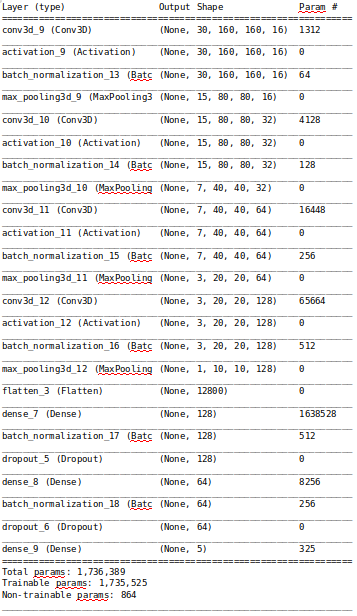
**Gesture Recognition Write-Up**

**Yash Sarawgi**

**Prathmesh Palande**

**Experimenting with Image resolution and batch\_size**

**We used a sample model for the same and the model summary is as below:**



We had issues with hitting memory limit on resources with image resolution of 160x160 with 30 frames and batch\_size of 40, getting 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*

So lets trade-off between these parameters  
Below are the experiments to see how training time is affected by image resolution, number of images in sequence and batch size

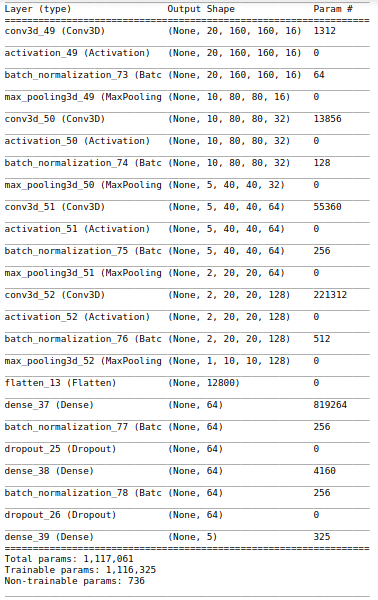
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Experiment Number** | **Model** | **Image\_size** | **Batch\_size** | **Time taken** |
| **1** | **Conv3DSample** | **160 x 160** | **30** | 4 min 20 s |
| **2** | **Conv3DSample** | **100 x 100** | **30** | 2 min 5 s |
| **3** | **Conv3DSample** | **100 x 100** | **60** | 3 min 3 s |
| **4** | **Conv3DSample** | **100 x 100** | **80** | **3 min 7s** |
| **5** | **Conv3DSample** | **160 x 160** | **15** | **4 min 15s** |
| **6** | **Conv3DSample** | **100 x 100** | **15** | 1 min 50s |
| **7** | **Conv3DSample** | **100 x 100** | **10** | 2 min 4s |
| **8** | **Conv3DSample** | **160 x 160** | **10** | **3 min 51s** |

##### **Clearly, the image resolution in sequence have greater impact on training time than batch\_size**

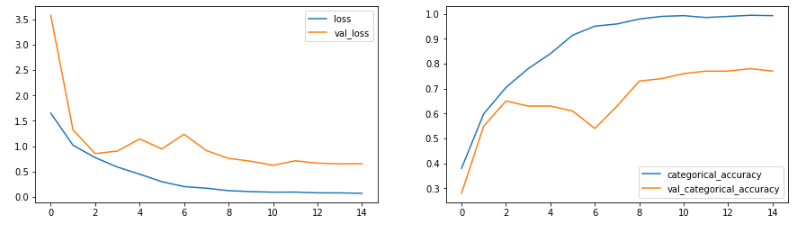
##### **We carried out experimentation with a fixed batch size of around 15-40, changing the resolution, and number of image per sequence taking into consideration our device memory constraints. We designed the models to have their memory foot print less than 50 MB corresponding to 4.3 million parameters assuming an industry standard parameter datatype size as 12 bytes.**

**Attempting 3D convolution Models**

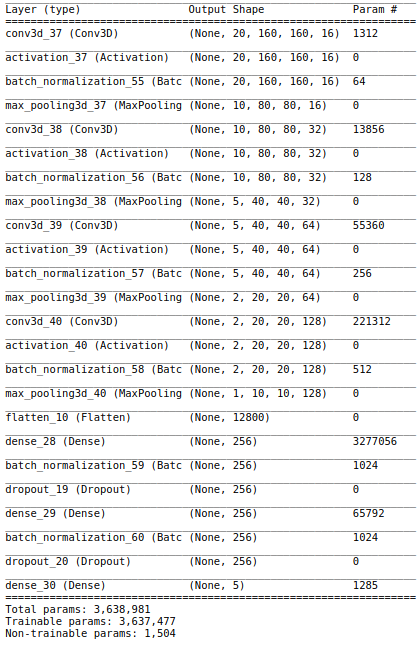
**Model 1 : Batch Size - 40, Epochs - 15 and without augmentation**



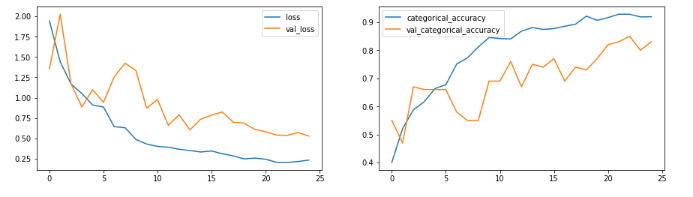
**Accuracy and Loss Curve:** Model is clearly **over-fitting**. So we need to do data augmentation

****

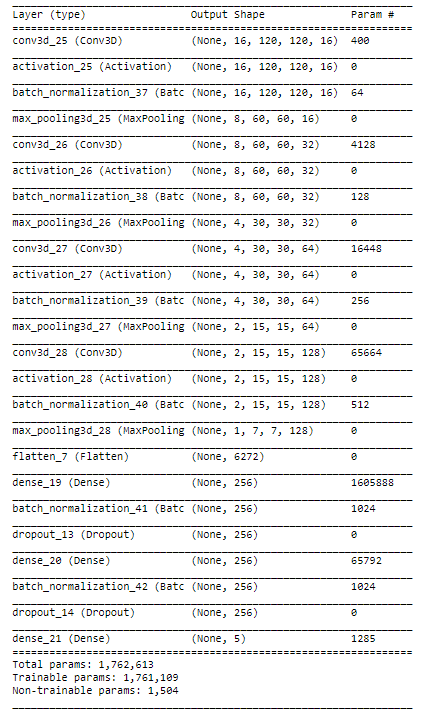
**Model 2 : (3,3,3) kernel, 160x160 res., with augmentation**



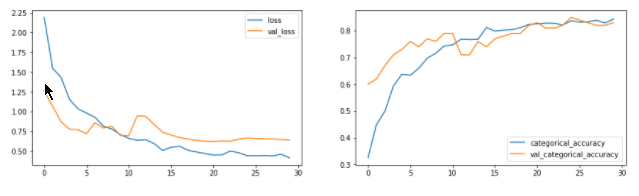
**Accuracy and Loss Curve**: We observe that the model is not over-fitting now, and we have a validation accuracy of 85% with training of 91%. To take care of the ridges in the graph, we may want to turn down the learning rate to 0.0002



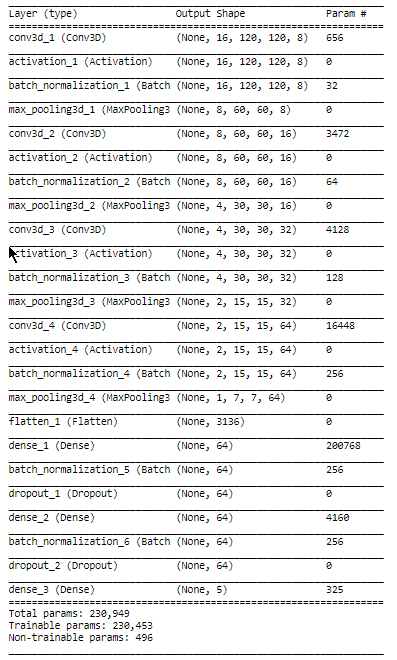
# **Model 3 : Kernel to be (2,2,2) and res to 120 x 120**



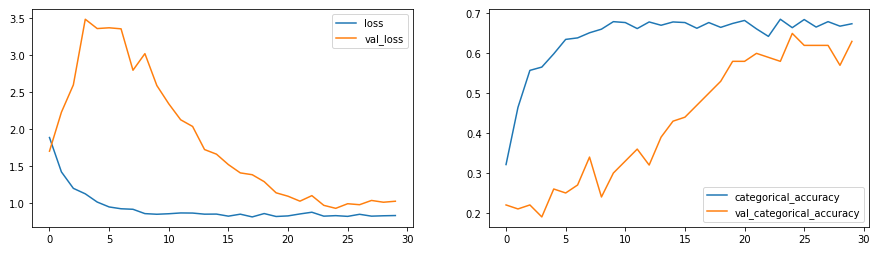
##### **Accuracy and Loss curve:** Now that we have a validation accuracy of 84%, and training of the same. We may try to reduce the number of parameters to further reduce our memory footprint.



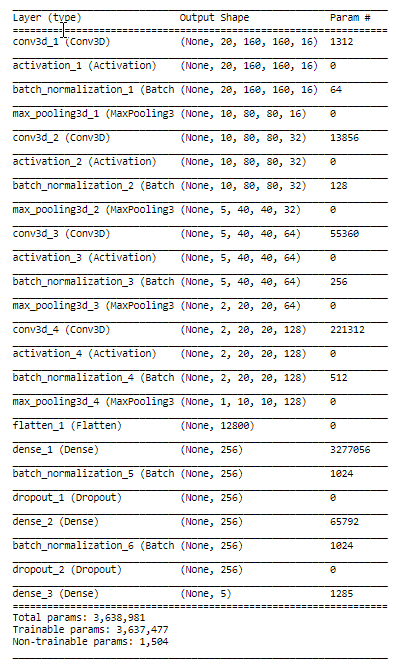
# **Model 4 - with reduced number of parameters**



**Accuracy and Loss curve:** Taking into consideration the model with reduced parameters to achieve low memory capability, the best validation accuracy of 65% was achieved. Though this may be fine to use in low memory devices, let’s try and augment the data more by rotating and changing colors. We would also want to go back to the kernel size of 3x3x3, as in an attempt to lower the memory foot print we had a significant hit to the validation accuracy.



# **Model 5 - (3,3,3) kernel & 160x160 res with added augmentation**

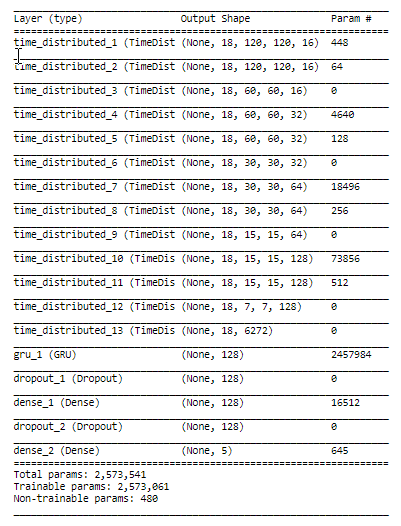


# **Accuracy and Loss curve:** Following up with the above model, we achieved maximum accuracy of 82% on validation.

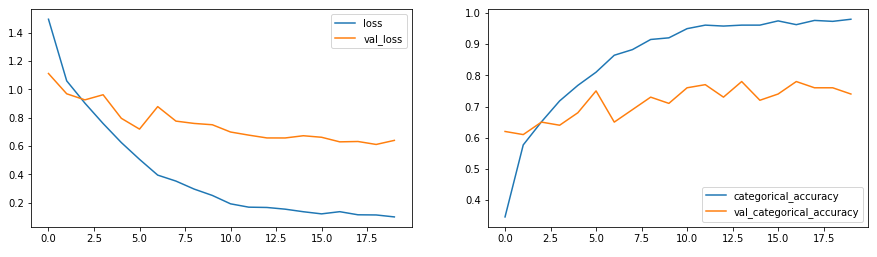
# 

# **Attempting CNN-LSTM models and transfer learning**

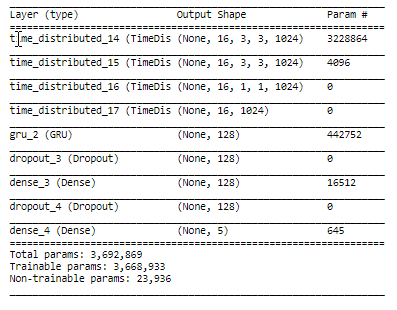
# **Model 6 - CNN LSTM with GRU**



**Accuracy and Loss curves:**  We can clearly see the model is over-fitting here. Just out of curiosity to try something different, instead of solely relying on data augmentation, we leveraged transfer learning and obtained fascinating results in the next model. As we already know, a kick-starter model trained on a vastly bigger dataset could be of significant use.



# **Model 7 - Transfer Learning on RNN-CNN model with GRU**



**Accuracy and Loss Curves:** The best model obtained was with Transfer learning (**mobilenet**) on RNN-CNN model with GRU. The training accuracy obtained was of 99.23% and the validation accuracy was 98%.

