**GESTURE RECOGNITION**

**We have tried independently with both the 3D CNN model (in CNN3D.ipynb) and the 2D CNN + RNN(GRU/LSTM) (in CNN\_RNN.ipynb)**

**First we tried with 3D CNN model and got following results**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Experiment Number** | **Model** | **Model Parameters** | **Result** | **Total # of Parameters** | **Decision + Explanation** |
| 1 | Model 1 | No. Convolution Layers : 6  Convolution Filter Size : (2,2,2)  - Pooling filter size : (2,2,2)  Batch Size : 15  Epochs : 30  Image size : (120, 120) | loss: 0.6148  categorical\_accuracy: 0.7587  val\_loss: 1.5143  val\_categorical\_accuracy: 0.6100 | 3,370,093 | Clearly, model is overfitting. Lets drop some of the neurons on dense layer in model 2 |
| 2 | Model 2 | No. Convolution Layers : 6  Convolution Filter Size : (2,2,2)  - Pooling filter size : (2,2,2)  Batch Size : 15  Epochs : 25  Image size : (120, 120) | loss: 0.4377  categorical\_accuracy: 0.8282  val\_loss: 1.0525  val\_categorical\_accuracy: 0.6500 | 3,303,021 | Clearly, model is overfitting. Lets further drop some of the neurons on dense layer in model 3 |
| 3 | Model 3 | No. Convolution Layers : 6  Convolution Filter Size : (2,2,2)  - Pooling filter size : (2,2,2)  Batch Size : 15  Epochs : 25  Image size : (120, 120) | loss: 0.7514  categorical\_accuracy: 0.7231    val\_loss: 0.9331  val\_categorical\_accuracy: 0.7600 | 1,464,301 | This is a stable model and it is not overfitting. But it has relatively low accuracy |
| 4 | Model 4 | No. Convolution Layers : 6  Convolution Filter Size : (2,2,2)  - Pooling filter size : (2,2,2)  Batch Size : 15  Epochs : 20  Image size : (120, 120) | loss: 0.4365 - categorical\_accuracy: 0.8552  val\_loss: 0.7503 - val\_categorical\_accuracy: 0.8100 | 3,308,965 | No of features are changed as more features are required then that will be fed to the learning part.  And this trick worked.  Val\_accuracy -> 81% |

H5 file for 3D CNN is attached in the zip.

**Then we tried with 2D CNN + RNN model and got following results:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Experiment Number** | **Model** | **Model Parameters** | **Result** | **Total # of Parameters** | **Decision + Explanation** |
| 1 | Model 1 | No. Convolution TimeDistributed Layers : 1 (64 convolutions)  Convolution Filter Size : (3,3)  - Pooling filter size : (2,2)  GRU : 64  Dense : 100  Batch Size : 50  Epochs : 1  Image size : (70, 70)  Number of images per video : 30 | loss: 1.7145 - categorical\_accuracy: 0.1964 - val\_loss: 1.6298 - val\_categorical\_accuracy: 0.2000 | 14,226,205 | A preliminary model just to check whether things working or not. |
| 2 | Model 2 | No. Convolution TimeDistributed Layers : 4 (64-32-64-64 convolutions)  Dropouts – 0.25  Convolution Filter Size : (3,3)  - Pooling filter size : (2,2)  GRU : 64  Dense : 100  Batch Size : 30  Epochs : 10  Image size : (70, 70)  Number of images per video : 30 | loss: 0.5499 - categorical\_accuracy: 0.8040 - val\_loss: 1.3525 - val\_categorical\_accuracy: 0.5400 | 3,046,013 | Decreased batch size with increased number of layers.  The model is overfitting. |
| 3 | Model 3 | No. Convolution TimeDistributed Layers : 6 (64-32-64-64-128-128 convolutions)  Dropouts – 0.25  Convolution Filter Size : (3,3)  - Pooling filter size : (2,2)  GRU : 64  Dense : 512  Batch Size : 30  Epochs : 10  Image size : (70, 70)  Number of images per video : 30 | loss: 8.5781 - categorical\_accuracy: 0.6247 - val\_loss: 8.6469 - val\_categorical\_accuracy: 0.6100 | 2,181,413 | This model gave better val\_accuracy but it was not fitting even on training data, then we tried with decreasing the learning rate using callbacks(patience =1) |
| 4 | Model 4 | No. Convolution TimeDistributed Layers : 6 (64-32-64-64-128-128 convolutions)  Dropouts – 0.25  Convolution Filter Size : (3,3)  - Pooling filter size : (2,2)  GRU : 64  Dense : 512  Batch Size : 20  Epochs : 20  Image size : (70, 70)  Number of images per video : 30 | loss: 8.1695 - categorical\_accuracy: 0.7564 - val\_loss: 8.6574 - val\_categorical\_accuracy: 0.6300 | 2,181,413 | Decreased learning rate with patience = 1 and also decreased the batch size and tried to run for more epochs to see if model is converging for training data. |
| 5 | Model 5 | Resnet50 with below 160 layers non trainable  GRU : 100  Dense : 100  Batch Size : 20  Epochs : 10  Image size : (198, 198)  Number of images per video : 15 | loss: 0.1180 - categorical\_accuracy: 0.9750 - val\_loss: 0.6621 - val\_categorical\_accuracy: 0.7500 | 6,315,689 | Seemed that feature extraction by our vanilla CNN was not good. Switched to transfer learning using Resnet50, but since it takes size of 197 as min, we had to decrease number of frames in video to 15.  The trick seemed to work well, and we got val\_accuracy of 75% |
| 6 | Model 6 | Resnet50 with below 150 layers non trainable  GRU : 1000  TimedistributedDense : 100  Dense:64  Batch Size : 20  Epochs : 15  Image size : (197, 197)  Number of images per video : 20 | loss: 0.1775 - categorical\_accuracy: 0.9500 - val\_loss: 0.4588 - val\_categorical\_accuracy: 0.8000 | 6,315,689 | It seems that feature extraction using resnet is good, let's make the model learn more  ,now we should fine tune this model giving 20 images and train over 150th layer. Also making GRU layer a little bigger and adding another timedistributed dense layer after it for better learning.  Also learning rate for layers was kept at 0.0004 for good learning  The val\_acc of 80% was gained |
| 7 | Model 7 | Resnet50 with below 140 layers non trainable  LSTM : 1024  TimedistributedDense : 128  Dense:64  Batch Size : 20  Epochs : 25(failed due to nimblebox low memory after 20 epochs)  Image size : (197, 197)  Number of images per video : 20 | categorical\_accuracy: 0.9890 - val\_loss: 0.3093 - val\_categorical\_accuracy: 0.8900 | 27,858,437 | Now we should fine tune this model more, train over 140th layer. Also changing GRU layer to LSTM  We got 89% accuracy on valid set. |
| 8 | Model 7 | Resnet50 with below 140 layers non trainable  LSTM : 1024  TimedistributedDense : 128  Dense:64  Batch Size : 20  Epochs : Rerun for 25 epochs more  Image size : (197, 197)  Number of images per video : 20 | categorical\_accuracy: 0.9971 - val\_loss: 0.2589 - val\_categorical\_accuracy: 0.9100 | 27,858,437 | For 25 more epochs of precious model, we got consistently above 85% val\_acc, and maximum of 91% valid\_acc |

**2D CNN + RNN h5 files as they are quite big:**

Link of h5 file with 89% valid\_accuracy ->   
<https://drive.google.com/open?id=1xhVJrXuFQ8xNO9G-f9hhD-eo3APnF7K7>

Link of h5 file with 91% valid\_accuracy -> <https://drive.google.com/open?id=1iF9zf3sBbZ0ZnQnlCrizfJWWIoV_6UXq>