**Gesture Recognition Case Study Report**

**Introduction**

The case study focuses on making the model learn various gestures from various videos in form of frames. These frames when played in order give a video.

In this Case Study various commands like volume up, volume down, rewind, fast forward and stop commands are given in form of gestures. These gestures are continuously monitored with the help of a camera. The purpose of this case study is to create a model, with enough video samples, to recognize any of the mentioned gesture and pass the required command to the device. These features can be seen in high end TV, smartphones, vehicles etc.

**Model Building**

For this Case Study several models were tried for better accuracies by changing both model architectures and hyperparameters. They are summarized below. Our aim was to keep the model size small so as to fit in a raspberry pi or a jetson nano to run a webcam. We have tried 7 models and finally came up with the best model.

Initial Testing Model: Testing of different combinations of batch-size, number of image frames and resolution to reach max GPU performance before OOM is done.

Results:

- We see that that GPU gives OOM error beyond 20 frames and batch size of 20 for a resolution of 160\*160. We will reduce image resolution

- For RTX-2060 with 8gb of v-ram the max params are: batch size=40, 25 frames and res of 100\*100. For a resolution of 160x160 we might have to reduce batch size.

We find the following by testing few epochs on a CONV 3D model:

- More training time: increase frames, increase resolution

- No effect on training time: batch size change

- OOM error of VRAM: increase in batch size

We were also overfitting the model so data augmentation was necessary.

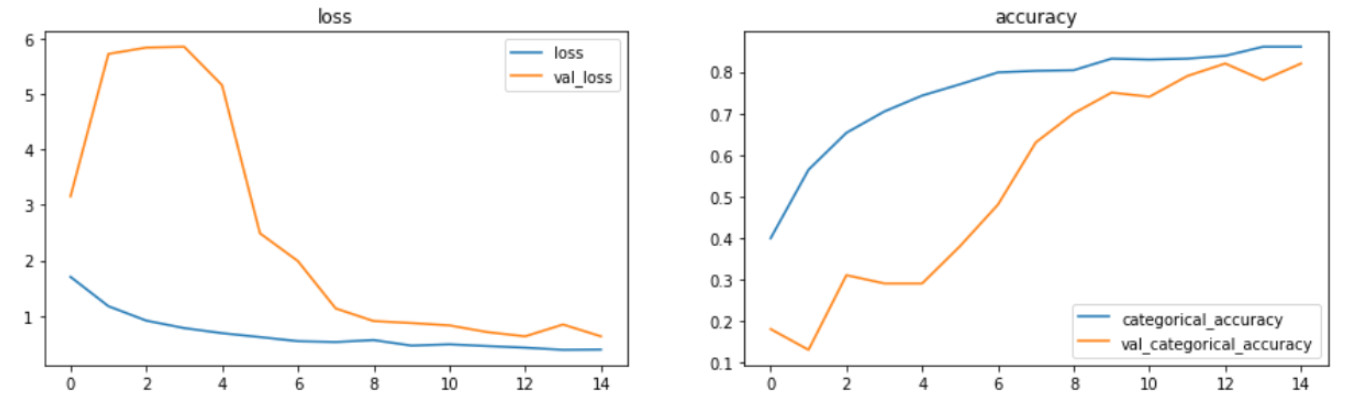
We have made a model summary table with model architecture details at the end.

Model 1: This is the initial model to test the network behaviour. We have taken 160x160 resolution for the images. But as the resolution has increased so we have to compensate with lower batch size and frames. We have used 3D convolutional layers for this model.

* **Conclusion**: The accuracies are low but the overfitting in the data is not very high. The training accuracy is low as compared to validation accuracy. This indicates that the model is not able to pick much details from the videos. This may be due to the high dropout rate of 0.5.
* **Decision**: Reduce dropout rate and increase number of units in the dense layers

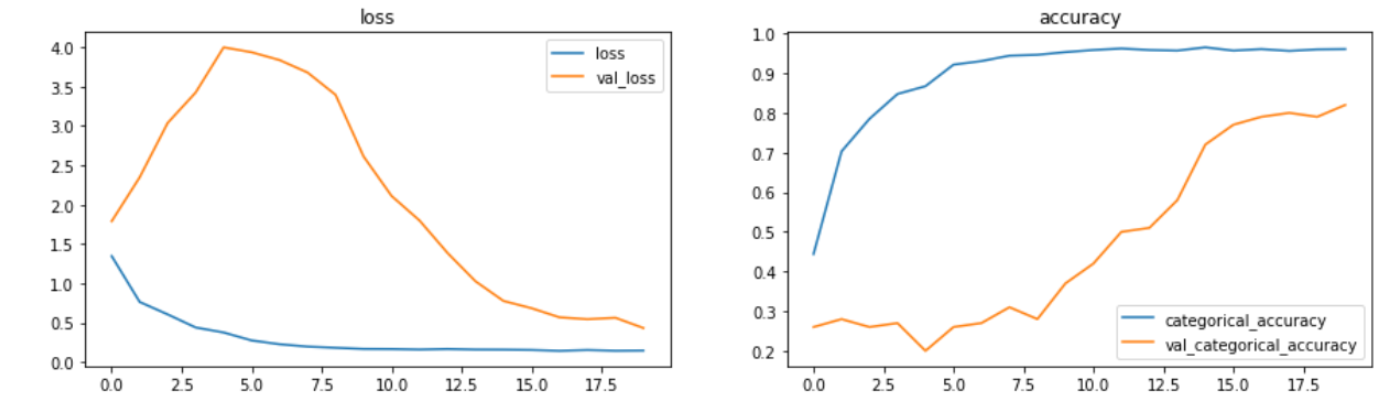
Model 2: In the second model, we have increased the number of dense units from 64 to 256 and decreased the dropout percent to 25%.

* **Conclusion**: The model plateaus to a validation accuracy of close to 86% and the training accuracy is 82% after only 15 epochs. From the plot we see the validation accuracy still has a scope of improvement. An increase in the number of dense units to 256 and reducing dropouts have helped increase the training accuracy.
* **Decision**: We will explore time-distributed CNN- RNN model architectures to check if higher accuracies can be obtained.



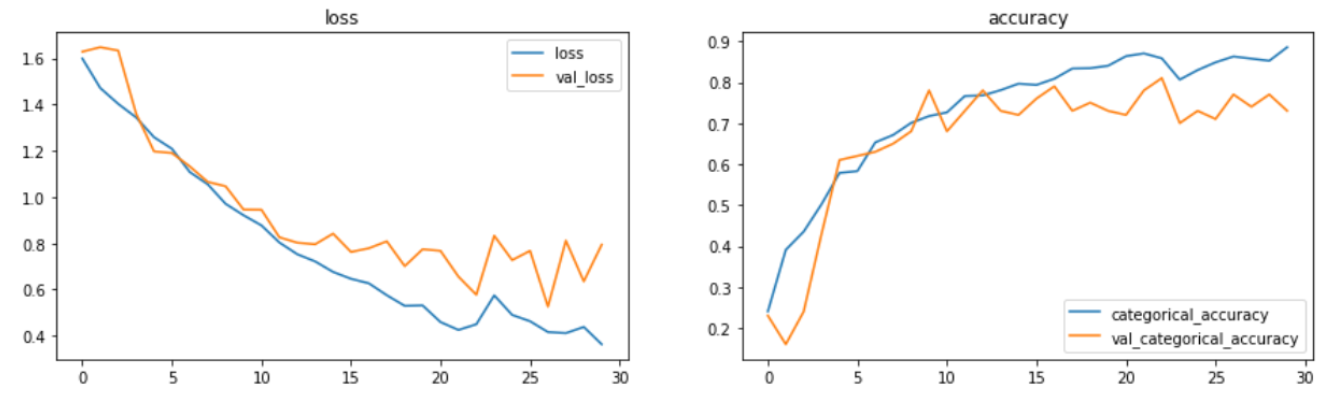
Model 3: From the 3rd model, we have not used Conv3D models because even though the memory occupied by Conv3D model is low, its accuracy has also been low. So in this model we have Time Distributed Conv2d + GRU model.

* **Conclusion:** The model plateaus to a training accuracy of close to 97% and the validation accuracy is 85% after 30 epochs. Time distributed layer of Conv2D is giving a higher training accuracy than Conv3D layers.
* **Decision:** Try using LSTM architecture if it can increase training accuracy by learning more features.



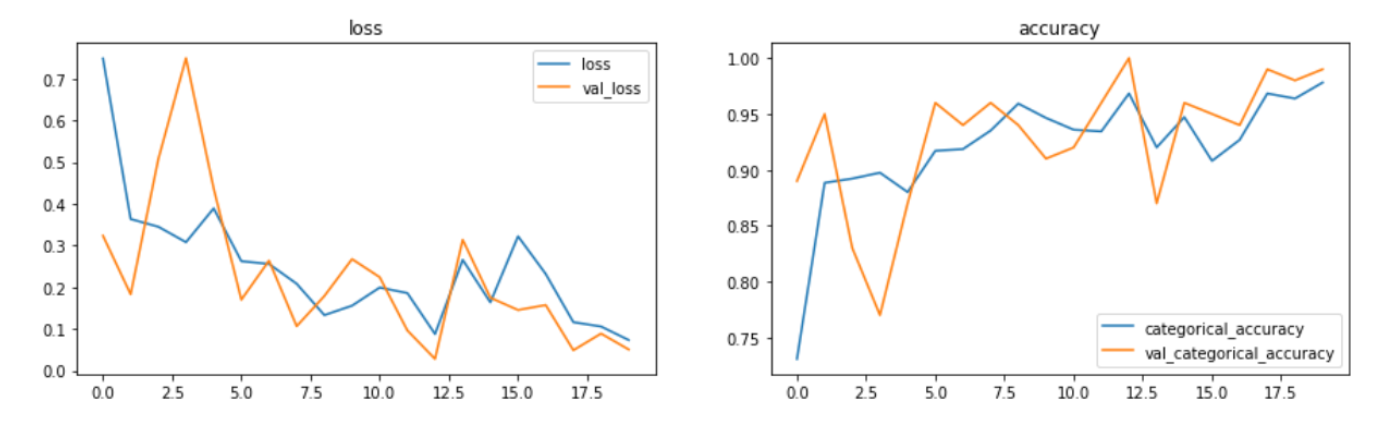
Model 4: For this model we have used Time Distributed Conv2D + LSTM cell model. Except from the presence of GRU cell, all other parameters are kept the same.

* **Conclusion:**  THE Conv2D+LSTM model gives slightly more overfitting results compared to the GRU for an increase in the model size. We would prefer GRU over LSTM.
* **Decision:** Prefer GRU RNN architecture to keep model size small



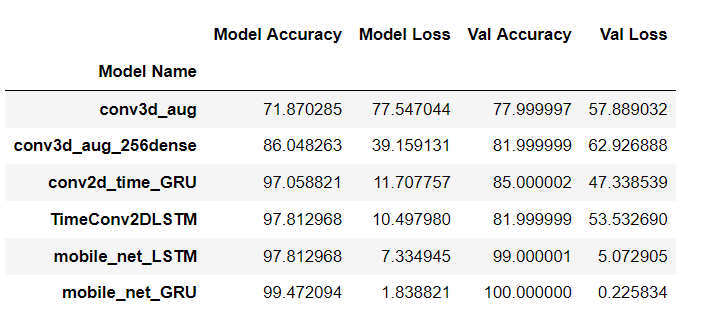
Model 5: In this a transfer learning model is used i.e. MobileNet. It is a model with pre-trained weights. It is combined model of Time Distributed MobileNet and LSTM. The performance of this model has improved drastically as compared to the previous models.

* **Conclusions**: The time distributed mobile net has a large file size but a remarkably high validation accuracy of 99% . The model has generalized pretty well and should perform well on unseen data as well. The batch size had to be reduced to 5 in order to prevent OOM error due to complex architecture of the CNN mobile-net.
* **Decision**: We will try to use GRU in place of LSTM to reduce model size and check accuracy



Model 6: In this we are creating a model of Time Distributed MobileNet + GRU cells. This helped in decreasing the memory used and improved performance too.

* **Conclusion**: The time distributed mobile net with GRU has a relatively smaller file size than mobilenet-LSTM and it is giving a very high accuracy model with no overfitting. The high validation score is due to the small validation set; but it should perform reasonably well with unseen data.
* **Decision**: We will check if the model size can be reduced even further to get comparable results by reducing GRU and Dense layer units.



Model 7: this is our final model which has given the best results so far. It is the same model with Time Distributed MobileNet +GRU. But this time in the GRU layer and FC layers we decreased units from 256 to 128.

* **Conclusions**: The final model uses the same architecture as before but with a significantly lower model size of 57 MB; getting a very good accuracy with lesser model parameters.
* **Decision**: We use it as our final model and save the model as .h5 file.

Model summary table: next page **Model Summaries**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model number | Model | Parameters | Hyper - parameter | Result |
| 1 | Conv3D-64 | * + 4 X Conv3D + FC layers.   + Filters in 4 layers- 8,16,32,64. They have equal kernel size (3x3x3).   + Relu activation.   + Batch Normalization layer   + Maxpooling3d layer of size (2x2x2)   + Flatten Layer   + 2X Fully connected Dense layer (64 units) with relu activation + Dropout(0.5) + Batch Normalization   + Softmax output layer   + Adam optimizer | * + Resolution: 160x160   + frames: 16   + batch\_size: 10   + epochs: 30 | * + Training Loss: 0.775   + Training accuracy: 71.8%   + Validation Loss: 0.579   + Validation accuracy: 77.9%   + Model Size: 5.9 MB |
| 2 | Conv3D-256 | * + 4 X Conv3D + FC layers.   + Filters in 4 layers- 8,16,32,64. They have equal kernel size (3x3x3).   + Relu activation.   + Batch Normalization layer   + Maxpooling3d layer of size (2x2x2)   + Flatten Layer   + 2X Fully connected Dense layer (256 units) with relu activation + Dropout(0.25) + Batch Normalization   + Softmax output layer   + Adam optimizer | * Resolution:160x160 * frames:16 * batch\_size:10 * epochs: 15 | * Training Loss: 0.392 * Training accuracy: 86.05% * Validation Loss: 0.63 * Validation accuracy: 81.9% * Model Size: 21 MB |
| 3 | Conv2D + GRU | * 4 X TimeDistributed CONV2D + GRU + FC layers. * Number of filters- 8,16,32,64. 1st and 2nd layer kernel size (3x3). 3rd and 4th have kernel size (2x2) * Relu activation function. * Batch Normalization layer * Maxpooling2d layer of size (2x2) * Time distributed Flatten Layer * GRU layer (256 units) with dropout (0.25) * 1 Fully connected Dense layer (256 units) with relu activation +Dropout(0.25) * Softmax output layer * Adam optimizer | * Resolution:160x160 * frames:16 * batch\_size:20 * epochs: 30 | * Training Loss: 0.117 * Training accuracy: 97.06% * Validation Loss: 0.47 * Validation accuracy: 85% * Model Size: 61 MB |
| 4 | Conv2D +LSTM | * 4 X TimeDistributed CONV2D + LSTM + FC layers. * Filter Sizes in 4 layers- 8,16,32,64. 1st and 2nd layer kernel size (3x3). 3rd and 4th have kernel size (2x2) * Relu activation. * Batch Normalization layer * Maxpooling2d layer of size (2x2) * Time distributed Flatten Layer * LSTM layer (256 units) with dropout (0.25) * 1 Fully connected Dense layer (256 units) with relu activation +Dropout(0.25) * Softmax output layer * adam optimizer | * Resolution:160x160 * frames:16 * batch\_size:20 * epochs: 30 | * Training Loss: 0.105 * Training accuracy: 97.81% * Validation Loss: 0.53 * Validation accuracy: 82% * Model Size: 81 MB |
| 5 | Time Distributed Mobile Net + LSTM | * 1 X TimeDistributed MobileNet (without output layer) + LSTM + FC layers. * Batch Normalization layer * Maxpooling2d layer of size (2x2) * Time distributed Flatten Layer to convert into 1d for input into lstm layer * LSTM layer (256 units) with dropout (0.25) * 1 Fully connected Dense layer (256 units) with relu activation +Dropout(0.25) * Softmax output layer * adam optimizer | * Resolution:160x160 * frames:16 * batch\_size:5 * epochs: 20 | * Training Loss: 0.0733 * Training accuracy: 97.81% * Validation Loss: 0.0507 * Validation accuracy: 99% * Model Size: 91 MB |
| 6 | TimeDistributed MobileNet (without output layer) + GRU + FC layers | * 1 X TimeDistributed MobileNet (without output layer) + GRU + FC layers. * Batch Normalization layer * Maxpooling2d layer of size (2x2) * Time distributed Flatten Layer to convert into 1d for input into lstm layer * GRU layer (256 units) with dropout (0.25) * 1 Fully connected Dense layer (256 units) with rel activation +Dropout(0.25) * Softmax output layer * adam optimizer | * Resolution:160x160 * frames:16 * batch\_size:5 * epochs: 20 | * Training Loss: 0.018 * Training accuracy: 99.47% * Validation Loss: 0.002 * Validation accuracy: 100% * Model Size: 78 MB |
| 7 | TimeDistributed MobileNet + GRU (lesser units)  FINAL MODEL | * 1 X TimeDistributed MobileNet (without output layer) + GRU + FC layers. * Batch Normalization layer * Maxpooling2d layer of size (2x2) * Time distributed Flatten Layer to convert into 1d for input into lstm layer * GRU layer (128 units) with dropout (0.25) * 1 Fully connected Dense layer (128 units) with rel activation +Dropout(0.25) * Softmax output layer * adam optimizer | * Resolution:160x160 * frames:20 * batch\_size:5 * epochs: 20 | * Training Loss: 0.036 * Training accuracy: 98.94% * Validation Loss: 0.026 * Validation accuracy: 99.0% * Model Size: 57 MB |