Gesture Recognition

Jikky John & MayurKumar Nanda

# Introduction

**The goal of this project is to classify 5 hand gestures to control the functions of a TV. The gestures and their interpretations are as follows**

Thumbs up - Increase Volume

Thumbs down - Decrease Volume

Right Swipe - Jump forward 10 seconds in the content being played

Left Swipe - Jump backwards 10 seconds in the content being played

Stop - Pause the content

**DataSet: For each gesture, videos have been split into frames of images. The frames are all in RGB and have different shapes.**

**Requirements**

1. **Since the dataset is large, generators are to be used so that memory consumption is within the limits of the system we are using to train**
2. **Build two models**
   1. **Using Conv3D**
   2. **CCN + RNN using the custom-built model and also using transfer learning**
3. **Models should have good accuracy and low inference time indicating that the models should be small**

# Plan of action

Since 2 models are to be built, the development will be split between the two project team members. One will work on the Conv3D model while the other works on the CCN + RNN model. Multiple experiments need to be conducted hence, these will be done in parallel in two different notebooks.

**For submission, two notebooks will be submitted with all the experiments carried out along with the h5 file for the best models.**

# CNN + RNN Model Experiments

## Resnet50 - Transfer Learning

We chose Resnet50 since it’s already a trained network with very high accuracy and small size. The model is basically Resnet + Conv layer + GRU + Dense + Softmax. The Dense layer was added in later experiments. Since this is transfer learning, we experimented with the layers of Resnet that should be trainable.

In the experiment below, the input shape is 120x120x3 for each image.

|  |  |  |  |
| --- | --- | --- | --- |
| Sr. No. | Model | Result | Decision + Explanation |
| 1 | Model Resnet50-CNN-RNN-GRU  Resnet50 - No layers trainable  CNN - 64 Conv + 16 Conv  GRU - 16 units + 8 units  FC - softmax  Trainable params: 535,885  Non-trainable params: 23,587,712  **Params:**  Frames\_used=8  img\_idx = [0, 4, 8, 12, 16, 20, 24, 28]  batch\_size=32  epoch=30  min\_lr=0.00001 | loss: 0.9945  categorical\_accuracy: 0.6124  val\_loss: 1.1370  val\_categorical\_accuracy: 0.5500  lr: 0.0010 at epoch 27 | Train- 61 %, Val-55%  Using a Resnet with all the layers as “Not Trainable” gives a low val accuracy of 55%. Also, we can clearly see that the model is overfitting. Let’s try to see if we can increase the val accuracy by making some of the Resnet layers trainable.  Only 8 frames are used and a batch size of 32 is used to start with. |
| **2** | Model Resnet50-CNN-RNN-GRU  Resnet50 – Last 5 Layers trainable  CNN - 64 Conv + 16 Conv  GRU - 16 units + 8 units  FC - softmax  Trainable params: 1,590,605  Non-trainable params: 22,532,992  **Params:**  Frames\_used=8  img\_idx = [0, 4, 8, 12, 16, 20, 24, 28]  batch\_size=32  epoch=50  min\_lr=0.00001 | loss 1.610034  categorical\_accuracy 0.185520  val\_loss 1.598412  val\_categorical\_accuracy 0.250000  lr 0.001000 | Train - 51%, Val-53%  Making the last 5 layers of Resnet has caused a lower train accuracy. But val accuracy is more than train accuracy.  Training accuracy was increasing very slowly while val accuracy was increasing very quickly by the 30th epoc.  **This indicates that the network needs further tweaking. Also, let’s tweak the network by keeping the Reset layers non-trainable** |
| **3** | Model - Resnet50-CNN-RNN-GRU32,16  Resnet50 - No layers trainable  CNN - 64 Conv + 16 Conv  GRU - 32 units + 16 units  FC - softmax  Trainable params: 540,869  Non-trainable params: 23,587,712  Params:  Frames\_used=8  img\_idx = [0, 4, 8, 12, 16, 20, 24, 28]  batch\_size=32  epoch=50  min\_lr=0.00001 | loss 0.679805  categorical\_accuracy 0.754148  val\_loss 1.130542  val\_categorical\_accuracy 0.600000  lr 0.001000 at epoch 27 | Train - 75%, Val-60%  Increasing epochs from 30 in exp 1 to 50 has resulted in further training and improvement in val accuracy.  Let’s see the impact if providing a further set of frames per video to the net improves results  Also learning rate is peaked out. So let’s reduce min\_lr further. |
| **4** | Model - Resnet50-CNN-RNN-GRU32,16  Resnet50 - No layers trainable  CNN - 64 Conv + 16 Conv  GRU - 32 units + 16 units  FC - softmax  Trainable params: 540,869  Non-trainable params: 23,587,712  Params:  Frames\_used=12  img\_idx = [0, 3, 6, 9, 12, 15, 17, 19, 21, 24, 27, 29]  batch\_size=32  epoch=50  min\_lr=0.00000001 | loss 1.081768  categorical\_accuracy 0.580694  val\_loss 1.225056  val\_categorical\_accuracy 0.510000  lr 0.000008 | Train - 58%, Val-51%  Adding more frames, i.e. total frames to 12 has resulted in reducing overfitting. But val accuracy has not improved.  Lets check if adding furthermore frames per video improves the train and val accuracy. |
| **5** | Model - Resnet50-CNN-RNN-GRU32,16  Resnet50 -No layers trainable  CNN - 64 Conv + 16 Conv  GRU - 32 units + 16 units  FC - softmax  Trainable params: 540,869  Non-trainable params: 23,587,712  Params:  Frames\_used=15  img\_idx = 15  batch\_size=32  epoch=50  min\_lr=0.00000001 | loss 0.894891  categorical\_accuracy 0.675716  val\_loss 1.218430  val\_categorical\_accuracy 0.540000  lr 0.000008 | Train - 67%, Val-54%  Increasing frames per video has increased the train and val accuracy.  **Lets experiment with different batch\_sizes** |
| **6** | Model - Resnet50-CNN-RNN-GRU32,16  Resnet50 - No layers trainable  CNN - 64 Conv + 16 Conv  GRU - 32 units + 16 units  FC - softmax  Params:  Frames\_used=12  img\_idx = [0, 3, 6, 9, 12, 15, 17, 19, 21, 24, 27, 29]  batch\_size=64  epoch=50  min\_lr=0.00000001 | loss 0.897857  categorical\_accuracy 0.669683  val\_loss 1.172591  val\_categorical\_accuracy 0.510000  lr 0.001000 | Train - 67%, Val-51%  With larger batch size, the train accuracy between epochs swings widely.  **Results are not better than when batch size was 32.**  Lets see if we make the 5 layers of Resnet trainable and batch\_size at 64, if the accuracy increases. |
| **7** | <Build on Model 4>  Resnet50 - Last 5 layers trainable  CNN - 64 Conv + 16 Conv  GRU - 32 units + 16 units  FC - softmax  Trainable params: 1,595,589  Non-trainable params: 22,532,992  Params:  Frames\_used=12  img\_idx = [0, 3, 6, 9, 12, 15, 17, 19, 21, 24, 27, 29]  batch\_size=64  epoch=30  min\_lr=0.00000001 | loss 1.113020  categorical\_accuracy 0.589744  val\_loss 1.237370  val\_categorical\_accuracy 0.510000  lr 0.000040  Run for another 30 epoch  loss 0.577146  categorical\_accuracy 0.803922  val\_loss 1.114633  val\_categorical\_accuracy 0.550000  lr 0.000010 | Train - 58%, Val - 51%  With a larger batch size, the accuracy between epochs swings widely. Val accuracy has remained the same in spite of adding a training layer.  Since the val accuracy is not increasing, the CNN layer following resnet should be tweaked. Let’s increase the layers to 32, 64, 128 with batch normalization and max-pooling layers.  Also reduce the batch\_size to 32. |
| **8** | Resnet50 - Last 5 layers trainable  CNN - 32, 64, 128  GRU - 32 units + 16 units  FC - softmax  Trainable params: 1,755,413  Non-trainable params: 22,533,440  Params:  Frames\_used=12  img\_idx = [0, 3, 6, 9, 12, 15, 17, 19, 21, 24, 27, 29]  batch\_size=32  epoch=30  min\_lr=0.00000001 | loss 0.208598  categorical\_accuracy 0.962293  val\_loss 1.365964  val\_categorical\_accuracy 0.580000  lr 0.000040 | **Train - 99, Val - 58**  **Changing the CNN network to 32, 64, 128 has actually shown some better training. The network has overfitted while val accuracy is 58.**  **Lets try to add dropouts to reduce overfitting** |
| **9 - Rerun** | Resnet50 - Last 5 layers trainable  CNN - 32, 64, 128  GRU - 32 units (return\_sequences=True) + 16 units  Dropout (0.5)  FC - softmax  Trainable params: 1,755,413  Non-trainable params: 22,533,440  Params:  Frames\_used=12  img\_idx = [0, 3, 6, 9, 12, 15, 17, 19, 21, 24, 27, 29]  batch\_size=32  epoch=30  min\_lr=0.00000001 | loss 0.497638  categorical\_accuracy 0.886878  val\_loss 0.940125  val\_categorical\_accuracy 0.620000  lr 0.000008 | **Train - 88, Val - 62**  **Adding dropouts has reduced overfitting and improved val accuracy**  Overall the val accuracy is not improving so we need to tweak the network further. |
| **10** | Resnet50 - Last 5 layers trainable  CNN - 32, 64, 128  GRU - 128 (Single GRU, return\_sequences=False)  Dropout (0.25)  Dense - 128  Dropout (0.25)  FC 5 - softmax  Trainable params: 1,853,413  Non-trainable params: 22,533,248  Params:  Frames\_used=12  img\_idx = [0, 3, 6, 9, 12, 15, 17, 19, 21, 24, 27, 29]  batch\_size=32  epoch=30  min\_lr=0.00000001 | loss 0.199097  categorical\_accuracy 0.942685  val\_loss 1.539697  val\_categorical\_accuracy 0.570000  lr 0.000040 | **Train - 94, Val - 57**  **After changing the network to just 1 GRU from 2 GRU’s, and adding a dense layer between GRU and softmax, the time to train has increased and the val accuracy has increased a little bit. The model is overfitting.**  **Lets try to increase dropout.** |
| **11** | Resnet50 - Last 5 layers trainable  CNN - 32, 64, 128  GRU - 128 (Single GRU, return\_sequences=False)  Dropout (0.5)  Dense - 128  Dropout (0.5)  FC 5 - softmax  Trainable params: 1,853,413  Non-trainable params: 22,533,248  Params:  Frames\_used=12  img\_idx = [0, 3, 6, 9, 12, 15, 17, 19, 21, 24, 27, 29]  batch\_size=32  epoch=30  min\_lr=0.00000001 | loss 0.244673  categorical\_accuracy 0.930618  val\_loss 1.359962  val\_categorical\_accuracy 0.590000  lr 0.000002 | **Train - 93, Val - 59**  **Increasing dropout has reduced overfitting but not by much.**  **Possibly 128 GRU is too high.**  **Also val accuracy is not increasing. Lets try by increasing frames per video and reduce GRU.** |
| **12** | Resnet50 - Last 5 layers trainable  CNN - 32, 64, 128  GRU - 64 (Single GRU, return\_sequences=False)  Dropout (0.5)  Dense - 64  Dropout (0.5)  FC 5 - softmax  Trainable params: 1,778,917  Non-trainable params: 22,533,248  Params:  Frames\_used=18 (randomly selected)  batch\_size=32  epoch=30  min\_lr=0.00001 | loss 0.425030  categorical\_accuracy 0.861237  val\_loss 1.107370  val\_categorical\_accuracy 0.600000  lr 0.000002 | **Train - 86, Val - 60**  **GRU reduced to 64 and frames increased to 18. Train and Val accuracy are similar to previous run which means that reducing GRU is not a good option.** |
| **13** | Resnet50 - Last 5 layers trainable  CNN - 32, 64, 128  GRU - 64(Single GRU, return\_sequences=False)  Dropout (0.25)  Dense - 64  Dropout (0.25)  FC 5 - softmax  Trainable params: 1,778,917  Non-trainable params: 22,533,248  Params:  Frames\_used=18 (randomly selected)  batch\_size=32  epoch=30  min\_lr=0.00000001 | loss 0.152146  categorical\_accuracy 0.956259  val\_loss 1.488340  val\_categorical\_accuracy 0.590000  lr 0.000040 | **Train - 96, Val - 59**  **Reducing dropout increased overfitting but val is still at 59. GRU 64 does not seem to be sufficient.** |
| **14** | Resnet50 - Last 5 layers trainable  CNN - 32, 64, 128  GRU - 128 (Single GRU, return\_sequences=False)  Dropout (0.25)  Dense - 128  Dropout (0.25)  FC 5 - softmax  Trainable params: 1,853,413  Non-trainable params: 22,533,248  Params:  Frames\_used=18 (randomly selected)  batch\_size=32  epoch=30  min\_lr=0.00000001 | loss 0.100028  categorical\_accuracy 0.972851  val\_loss 1.823814  val\_categorical\_accuracy 0.480000  lr 0.000200 | **Train - 97, Val - 48**  **GRU back to 128 with low dropouts leads to overfitting.**  **Lets try with batch\_size 25 and dropouts after conv layers** |
| **15** | Resnet50 - Last 10 layers trainable  CNN - 128, Dropout (0.4)  CNN - 256, Dropout (0.4)  GRU - 128 (Single GRU, return\_sequences=False)  Dropout (0.2)  Dense - 128  Dropout (0.4)  FC 5 - softmax  Trainable params: 8,024,837  Non-trainable params: 19,123,584  Params:  Frames\_used=18 (randomly selected)  batch\_size=32  epoch=50  min\_lr=0.00000001 | loss 0.621170  categorical\_accuracy 0.736048  val\_loss 1.274802  val\_categorical\_accuracy 0.550000  lr 0.000002 | **Train - 74, Val - 55**  **Changing CNN layer and adding dropouts has surely reduced overfitting. Val accuracy is at 55**  **But lets try by reducing dropouts at CNN layers and also adding an additional dense layer of 256 after dense 128** |
| **16** | Resnet50 - Last 10 layers trainable  CNN - 128, Dropout (0.2)  CNN - 256, Dropout (0.2)  GRU - 128 (Single GRU, return\_sequences=False)  Dropout (0.15)  Dense - 128  Dropout (0.2)  Dense - 256  Dropout (0.4)  FC 5 - softmax  Trainable params: 8,059,269  Non-trainable params: 19,124,352  Params:  Frames\_used=18 (randomly selected)  batch\_size=32  epoch=50  min\_lr=0.00000001 | loss 1.120279  categorical\_accuracy 0.506787  val\_loss 1.393608  val\_categorical\_accuracy 0.500000  lr 0.000002 | **Train - 50, Val - 50**  **Changing dropouts has surely reduced overfitting but val accuracy also has dropped**  **Lets go back to CNN 32, 64, 128, 256 and try with dropouts.** |
| **17** | Resnet50 - Last 10 layers trainable  CNN - 32, Dropout (0.2)  CNN - 64, Dropout (0.2)  CNN - 128, Dropout (0.2)  CNN - 256, Dropout (0.2)  GRU - 128 (Single GRU, return\_sequences=False)  Dropout (0.15)  Dense - 128  Dropout (0.2)  Dense - 256  Dropout (0.4)  FC 5 - softmax  Trainable params: 5,643,813  Non-trainable params: 19,123,776  Params:  Frames\_used=18 (randomly selected)  batch\_size=32  epoch=50  min\_lr=0.00000001 | loss 0.929674  categorical\_accuracy 0.636501  val\_loss 1.135600  val\_categorical\_accuracy 0.590000  lr 0.000008 | **Train - 63, Val - 59**  **Train and Val are pretty close but val accuracy is still below 60.**  **Try adding one more CCN 16 layer before CNN 32** |
| **18** | Resnet50 - Last 10 layers trainable  CNN - 16, Dropout (0.2)  CNN - 32, Dropout (0.2)  CNN - 64, Dropout (0.2)  CNN - 128, Dropout (0.2)  CNN - 256, Dropout (0.2)  GRU - 128 (Single GRU, return\_sequences=False)  Dropout (0.15)  Dense - 128  Dropout (0.2)  Dense - 256  Dropout (0.35)  FC 5 - softmax  Trainable params: 5,353,557  Non-trainable params: 19,123,808  Params:  Frames\_used=18 (randomly selected)  batch\_size=32  epoch=50  min\_lr=0.00000001 | loss 0.540016  categorical\_accuracy 0.787330  val\_loss 1.343717  val\_categorical\_accuracy 0.620000  lr 0.000008 | **Train - 79, Val - 62**  **Train and Val are pretty close but val accuracy is still below 60.** |

### Resnet50 Results

After conducting many experiments with different combinations of batch\_size, image\_frames, min\_lr and also changing the number of layers of resnet that will be trained (limited to upto last 10 only), resnet has not provided satisfactory val accuracy. We limited the number of trainable layers in resnet to last 5, 10 layers so that we do not have to train an already-trained network. It seems that resnet is not a suitable candidate for our problem, because in our opinion, the only change in the images is a small hand gesture change. Resnet has been trained to classify large objects (animals, humans from other objects). To enable resnet to learn our specific feature, we will need to train many more layers, possibly last 25 or 30 layers of resnet to get high val accuracy. But if we do that, then the purpose of using an already trained network and use transfer learning is lost.

## 

## Custom CNN + RNN Model

Resnet50 did not give the necessary accuracy so we went about building a custom CCN + RNN network to see if we can get better accuracy. The main idea behind constructing the model is to have a TimeDistributed CCN layer that feeds into GRU and then a Dense and finally the softmax layer for classification.

In the experiment below, the input shape is 120x120x3 for each image.

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No.** | Model | **Result** | **Decision + Explanation** |
| **50** | CNN - 16, 32, 64, 128  GRU - 128 (Single GRU, return\_sequences=False)  Dropout (0.25)  Dense - 128  Dropout (0.25)  FC 5 - softmax  Trainable params: 2,573,445  Non-trainable params: 480  Params:  Frames\_used=18 (randomly selected)  batch\_size=32  epoch=30  min\_lr=0.00000001 | loss 0.075908  categorical\_accuracy 0.977376  val\_loss 0.918698  val\_categorical\_accuracy 0.710000  lr 0.000008 | **A simple CNN network of 16, 32, 64, 128 Conv2D layers are added to the model, with batch normalization and max pooling. These are ofcourse TimeDistributed. The output is then fed to a GRU 128 and Dense 128. Dropouts are used since we know that overfitting is expected.**  **Train - 98, Val - 76**  **Val accuracy is way better than resnet50 already**  **Model is overfitting clearly. Lets try with increased dropouts** |
| **51** | CNN - 16, 32, 64, 128  GRU - 128 (Single GRU, return\_sequences=False)  Dropout (0.4)  Dense - 128  Dropout (0.4)  FC 5 - softmax  Trainable params: 2,573,445  Non-trainable params: 480  Params:  Frames\_used=18 (randomly selected)  batch\_size=32  epoch=30  min\_lr=0.00000001 | loss 0.225382  categorical\_accuracy 0.930618  val\_loss 0.786345  val\_categorical\_accuracy 0.720000  lr 0.000040 | **Train - 93, Val - 72**  **Increasing the dropout has reduced the overfitting. Lets increase it a bit more and check.**  **Also lets increase the epoch’s to 40 since till epoch 25, the val\_accuracy was 54% and after that it increased to 73% in 5 epochs** |
| **52** | CNN - 16, 32, 64, 128  GRU - 128 (Single GRU, return\_sequences=False)  Dropout (0.5)  Dense - 128  Dropout (0.5)  FC 5 - softmax  Trainable params: 2,573,445  Non-trainable params: 480  Params:  Frames\_used=18 (randomly selected)  batch\_size=32  epoch=50  min\_lr=0.00000001 | loss 0.628093  categorical\_accuracy 0.791855  val\_loss 0.747681  val\_categorical\_accuracy 0.770000  lr 0.000002 | **Train - 79, Val - 77**  Increasing the dropout has caused the train accuracy to fall further so much so that the train accuracy is now almost the same as val accuracy and in one case the val accuracy is more than train accuracy. This is caused because during training dropout is used but during val, dropouts are not considered. Or the other possibility is that the network may be slightly overfitting.  Overall the results are the best till now and a simple network seems to be performing better than Resnet50.  Lets see if reducing the dropout after the Dense128 layer makes any difference. |
| **53** | CNN - 16, 32, 64, 128, 256  GRU - 128 (Single GRU, return\_sequences=False)  Dropout (0.5)  Dense - 128  Dropout (0.4)  FC 5 - softmax  Trainable params: 1,345,413  Non-trainable params: 992  Params:  Frames\_used=18 (randomly selected)  batch\_size=32  epoch=50  min\_lr=0.00001 | loss 0.306677  categorical\_accuracy 0.906486  val\_loss 0.623556  val\_categorical\_accuracy 0.830000  lr 0.000008 | **Adding of a CCN 256 layer and reduced dropout after Dense128, leads to train of 90% and val of 83%. Thus adding a CNN 256 has improved val accuracy.** |
| **54** | CNN - 16, 32, 64, 128, 256  GRU - 128 (Single GRU, return\_sequences=False)  Dropout (0.5)  Dense - 128  Dropout (0.4)  FC 5 - softmax  Trainable params: 1,345,413  Non-trainable params: 992  Params:  Frames\_used=18 (randomly selected)  batch\_size=20  epoch=50  min\_lr=0.00000001 | loss 0.393817  categorical\_accuracy 0.856712  val\_loss 0.462857  val\_categorical\_accuracy 0.850000  lr 0.000040 | **After reducing batch size to 20, Train - 86, Val = 85**  **Try with adding droput in the conv layers** |
| **55** | CNN - 16,  CNN - 32,Dropout (0.3)  CNN - 64,Dropout (0.3)  CNN - 128,Dropout (0.3)  CNN - 256, Dropout (0.3)  GRU - 128 (Single GRU, return\_sequences=False)  Dropout (0.5)  Dense - 128  Dropout (0.4)  FC 5 - softmax  Trainable params: 1,345,413  Non-trainable params: 992  Params:  Frames\_used=18 (randomly selected)  batch\_size=20  epoch=50  min\_lr=0.00000001 | loss 8.157827e-01  categorical\_accuracy 6.636501e-01  val\_loss 1.542846e+00  val\_categorical\_accuracy 3.900000e-01  lr 1.000000e-08 | **Train - 66, Val - 39**  **Adding dropouts in the CNN layer has caused the train and val accuracy both to fall.** |
| **Final Model** | **Experiment 54**  CNN - 16, 32, 64, 128, 256  GRU - 128 (Single GRU, return\_sequences=False)  Dropout (0.5)  Dense - 128  Dropout (0.4)  FC 5 - softmax  Trainable params: 1,345,413  Non-trainable params: 992  Params:  Frames\_used=18 (randomly selected)  batch\_size=20  epoch=50  min\_lr=0.00000001 | loss 0.393722  categorical\_accuracy 0.856712  val\_loss 0.613356  val\_categorical\_accuracy 0.810000  lr 0.000040 | **The combination of CNN layers of 16, 32, 64, 128, 256 connected to a GRU 128, Dense 128 and finally softmax leads to a good model with 81% accuracy. Batch size of 20 and frames used 18 also helped achieve good accuracy.** |

### Custom CNN+RNN Results

Custom network using Conv2d layers, GRU and Dense layer has yielded a much better result than resnet50 within 6 experiments. Additional experiments where image augmentation, adding some more CNN layers and fine-tuning dropouts can lead to even better results.

The best model has a val accuracy of 85% with 1.34 million parameters to be trained.

Hyper params used are as follows

Frames\_used=18 (randomly selected)

batch\_size=20

epoch=50

min\_lr=0.00000001

## 

## 

## MobileNetV2 - Transfer Learning

Resnet50 did not give great results but to check if some other small already trained network works reasonably well, we chose to check MobileNetV2 as one more model. Here the MobileNetV2 is connected GRU using TimeDistributed then a Dense layer and finally to softmax layer.

|  |  |  |  |
| --- | --- | --- | --- |
| Sr. No. | Model | Result | Decision + Explanation |
| **70** | MobilenetV2 (last 10 layers trainable)  GRU - 128  Dropout - 0.5  Dense - 128  Dropout - 0.5  FC - 5 softmax  Trainable params: 8,663,877  Non-trainable params: 1,525,504  Params:  Frames\_used=18 (randomly selected)  batch\_size=32  epoch=30  min\_lr=0.00000001 | loss 0.107624  categorical\_accuracy 0.972851  val\_loss 0.800824  val\_categorical\_accuracy 0.690000  lr 0.000040 | **Train - 97, Val - 69**  **Using mobilenet with last 10 layers as trainable, the val accuracy is at 69. Model is cleary overfitting.** |
| **71** | MobilenetV2 (last 20 layers trainable)  GRU - 128  Dropout - 0.5  Dense - 128  Dropout - 0.4  FC - 5 softmax  Trainable params: 9,137,477  Non-trainable params: 1,051,904  Params:  Frames\_used=18 (randomly selected)  batch\_size=20  epoch=50  min\_lr=0.00000001 | loss 0.017967  categorical\_accuracy 0.995475  val\_loss 0.491741  val\_categorical\_accuracy 0.900000  lr 0.000010 | **Train - 99.5, Val - 90**  **With last 20 layers of the mobilenet being made trainable, the val accuracy has jumped to 90. But still the model is overfitting.**  **Tweak the LR further and also increase dropout.** |
| **72** | MobilenetV2 (last 20 layers trainable)  GRU - 128  Dropout - 0.5  Dense - 128  Dropout - 0.5  FC - 5 softmax  Trainable params: 9,137,477  Non-trainable params: 1,051,904  Params:  Frames\_used=18 (randomly selected)  batch\_size=20  epoch=50  min\_lr=0.0000000001 | loss 0.020126  categorical\_accuracy 0.995475  val\_loss 0.138011  val\_categorical\_accuracy 0.960000  lr 0.000002 | **Train - 99.5, Val - 96**  **The val accuracy has increased to 96 with an increase in dropout. In the last epochs the network val accuracy is increasing.**  **Increase dropout further to see if the network gives better accuracy.** |
| **73** | MobilenetV2 (last 20 layers trainable)  GRU - 128  Dropout - 0.6  Dense - 128  Dropout - 0.6  FC - 5 softmax  Trainable params: 9,137,477  Non-trainable params: 1,051,904  Params:  Frames\_used=18 (randomly selected)  batch\_size=20  epoch=50  min\_lr=0.0000000001 | loss 0.050275  categorical\_accuracy 0.986425  val\_loss 0.208935  val\_categorical\_accuracy 0.940000  lr 0.000008 | **Train - 99, Val - 94**  **Even with high dropouts the train is pretty high and val is also a nice 94% accuracy.** |
| **Final Model** | **Experiment 72**  MobilenetV2 (last 20 layers trainable)  GRU - 128  Dropout - 0.5  Dense - 128  Dropout - 0.5  FC - 5 softmax  Trainable params: 9,137,477  Non-trainable params: 1,051,904  Params:  Frames\_used=18 (randomly selected)  batch\_size=20  epoch=50  min\_lr=0.0000000001 | loss 2.495213e-02  categorical\_accuracy 9.939668e-01  val\_loss 4.590651e-01  val\_categorical\_accuracy 9.200000e-01  lr 3.200000e-07 | **Train - 99.4, Val - 92** |

### MobileNetV2 Results

Of the 3 models tried for CNN + RNN, MobileNetV2 + RNN gives the best results with val accuracy of over 95. The network is very simple were the mobilenet is connected directly the GRU layer. The trainable parameters are higher as compared to Resnet and Custom network but the val accuracy is pretty high.

The best model has the following params

Frames\_used=18 (randomly selected)

batch\_size=20

epoch=50

min\_lr=0.0000000001

# Conv3D Model Experiments

3D convolution is a natural extension to 2D convolution in which instead of a 2D filter 3D filter is used. This gives the network ability to work on entire frames in the video at one go as compared to 2D convolution where one frame is considered one at a time. We expect that the training parameters will be lesser than the CNN-RNN model.

|  |  |  |  |
| --- | --- | --- | --- |
| Sr. No. | Model | Result | Decision + Explanation |
| 1 | input\_shape=(15,120,120,3)  activation = elu  conv layer 1- 8 features, dropout=0.2  conv layer 2- 16 features, dropout=0.2  conv layer 3- 24 features, dropout=0.2  Dense layer - 64 neurons dropout= 0.5  Total params: 169,845  Trainable params: 169,621  Non-trainable params: 224  min\_lr=0.0001,  batch\_size=64  num\_epochs=5 | Categorical accuracy = 64.71%  Validation accuracy = 27% | Clearly a case of overfitting, If the number of epochs were increased then also it would have overfitted. Low Val accuracy.  Need to increase drop-out percentage to reduce overfitting and increase number of neurons in fully connected layer and add more convolution layers ti improve train + val accuracy. |
| **2** | input\_shape=(15,120,120,3)  activation = elu  conv layer 1- 16 features, dropout=0.25  conv layer 2- 32 features, dropout=0.25  conv layer 3- 64 features, dropout=0.25  Dense layer - 128 neurons dropout= 0.5  Dense layer - 256 neurons dropout= 0.5  Total params: 946,869  Trainable params: 945,877  Non-trainable params: 992  min\_lr=0.0001,  batch\_size=64  num\_epochs=5 | Categorical accuracy = 63.95%  Validation accuracy = 36% | **Conv layer changed from 8 to 16. Dropout increased.**  **There is slight improvement in overfitting**  **The network needs to train for longer.**  **Hence increase the number of epoch’s.** |
| **3** | input\_shape=(15,120,120,3)  activation = elu  conv layer 1- 16 features, dropout=0.25  conv layer 2- 32 features, dropout=0.25  conv layer 3- 64 features, dropout=0.25  Dense layer - 128 neurons dropout= 0.5  Dense layer - 256 neurons dropout= 0.5  Total params: 946,869  Trainable params: 945,877  Non-trainable params: 992  min\_lr=0.0001,  batch\_size=64  num\_epochs=20 | Categorical accuracy = 88.69%  Validation accuracy = 30% | **With increase in the number of epochs, train accuracy increased but val accuracy did not improve. Also the model is overfitting.**  **Try with lower dropout in fully connected layer and increased dropout in in convolution layers** |
| **4** | input\_shape=(15,120,120,3)  activation = elu  conv layer 1- 16 features, dropout=0.3  conv layer 2- 32 features, dropout=0.3  conv layer 3- 64 features, dropout=0.3  Dense layer - 128 neurons dropout= 0.3  Dense layer - 256 neurons dropout= 0.3  Total params: 946,869  Trainable params: 945,877  Non-trainable params: 992  min\_lr=0.0001,  batch\_size=64  num\_epochs=20 | Categorical accuracy = 90.8%  Validation accuracy = 24% | **Train accuracy has improved and val accuracy has decreased.**  **Learning rate continues to decrease but peaks out at the configured min\_lr.**  **Training accuracy is increasing with every epoch but validation accuracy has almost become stagnant.**  **Larger batch size speeds the computation since number of updates required for network to converge is less comparatively.**  **Try by decreasing the minimum learning rate for better results and increasing epoch slightly for network to reach to that learning rate. Also try with smaller batch size.** |
| **5** | input\_shape=(15,120,120,3)  activation = elu  conv layer 1- 16 features, dropout=0.3  conv layer 2- 32 features, dropout=0.3  conv layer 3- 64 features, dropout=0.3  Dense layer - 128 neurons dropout= 0.3  Dense layer - 256 neurons dropout= 0.3  Total params: 946,869  Trainable params: 945,877  Non-trainable params: 992  min\_lr=0.00000001,  batch\_size=40  num\_epochs=23 | **categorical\_accuracy: 0.9457**  **val\_categorical\_accuracy: 0.2500**  **lr: 8.0000e-06** | **Slight improvement as compared to previous model.**  **Minimum learning rate can be decreased further and epochs and drop outs can be increased to reduce overfitting** |
| **6** | input\_shape=(15,120,120,3)  activation = elu  conv layer 1- 16 features, dropout=0.1  conv layer 2- 32 features, dropout=0.35  conv layer 3- 64 features, dropout=0.35  Dense layer - 128 neurons dropout= 0.4  Dense layer - 256 neurons dropout= 0.4  Total params: 946,869  Trainable params: 945,877  Non-trainable params: 992  min\_lr=0.00000001,  batch\_size=40  num\_epochs=30 | **categorical\_accuracy: 0.8944**  **val\_categorical\_accuracy: 0.49**  **lr: 3.2000e-07** | **Train - 90, Val - 49.**  **As can be seen, decreasing the learning rate, increasing the dropouts and increasing the epochs helped in reducing the overfitting and increased the val accuracy.**  **To reduce overfitting further, we can try reducing the learning rate further, increasing the dropout, increasing the epochs and reduce batch size** |
| **7** | input\_shape=(15,120,120,3)  activation = elu  conv layer 1- 16 features, dropout=0.15  conv layer 2- 32 features, dropout=0.35  conv layer 3- 64 features, dropout=0.40  Dense layer - 128 neurons dropout= 0.45  Dense layer - 256 neurons dropout= 0.45  Total params: 946,869  Trainable params: 945,877  Non-trainable params: 992  min\_lr=0.000000000000001  batch\_size=35  num\_epochs=35 | **categorical\_accuracy: 0.8582**  **val\_categorical\_accuracy: 0.7600**  **lr: 3.2000e-07** | **Train - 85.5, Val - 76**  **As seen from the result, validation accuracy has increased because of reducing learning rate and increasing the number of epochs. Overfitting has reduced due to an increase in dropouts**  **We can further try to reduce the batch size**  **As seen from graph there is rising validation accuracy, so can increase number of epochs** |
| **8** | input\_shape=(15,120,120,3)  activation = elu  conv layer 1- 16 features, dropout=0.05  conv layer 2- 32 features, dropout=0.40  conv layer 3- 64 features, dropout=0.45  Dense layer - 128 neurons dropout= 0.5  Dense layer - 256 neurons dropout= 0.6  Total params: 946,869  Trainable params: 945,877  Non-trainable params: 992  min\_lr=0.000000000000001  ,  batch\_size=35  num\_epochs=45 | **categorical\_accuracy: 0.7873**  **val\_categorical\_accuracy: 0.7000**  **lr: 8.0000e-06** | **Slight reduction in overfitting due to increase in dropout**  **But there is a need to increase both validation and training accuracy**  **Let’s remove dropout in the first convolution layer**  **and also decrease dropout in subsequent convolution layers**  **and also lets reduce batch size** |
| **9** | input\_shape=(15,120,120,3)  activation = elu  conv layer 1- 16 features, no dropout  conv layer 2- 32 features, dropout=0.35  conv layer 3- 64 features, dropout=0.40  Dense layer - 128 neurons dropout= 0.5  Dense layer - 256 neurons dropout= 0.6  Total params: 946,869  Trainable params: 945,877  Non-trainable params: 992  min\_lr=0.000000000000001  ,  batch\_size=25  num\_epochs=45 | **categorical\_accuracy: 0.7934**  **val\_categorical\_accuracy: 0.7200**  **lr: 3.2000e-07** | **There is a slight improvement in validation accuracy and training accuracy by decreasing the batch size and decreasing the drop-outs.**  **we can further decrease the batch size to improve results.**  **Learning rate is not reducing below 1e-07. Hence we can increase it to 1e-07.** |
| **10** | input\_shape=(15,120,120,3)  activation = elu  conv layer 1- 16 features  conv layer 2- 32 features, dropout=0.35  conv layer 3- 64 features, dropout=0.40  Dense layer - 128 neurons dropout= 0.5  Dense layer - 256 neurons dropout= 0.6  Total params: 946,869  Trainable params: 945,877  Non-trainable params: 992  min\_lr=0.0000001  ,  batch\_size=20  num\_epochs=45 | **categorical\_accuracy: 0.7828**  **val\_categorical\_accuracy: 0.7200**  **lr: 8.0000e-06** | **There is slight reduction in overfitting , but both validation and training accuracy need to improve.**  **hence we can reduce drop out and reduce learning rate to increase accuracy** |
| **11** | input\_shape=(15,120,120,3)  activation = elu  conv layer 1- 16 features  conv layer 2- 32 features, dropout=0.30  conv layer 3- 64 features, dropout=0.40  Dense layer - 128 neurons dropout= 0.4  Dense layer - 256 neurons dropout= 0.55  Total params: 946,869  Trainable params: 945,877  Non-trainable params: 992  min\_lr=0.00000001  batch\_size=20  num\_epochs=45 | **categorical\_accuracy: 0.9578**  **val\_categorical\_accuracy: 0.8200**  **lr: 1.6000e-06** | **There is huge improvement in both the accuracies with slight increase in overfitting.**  **can try reducing the parameters** |
| **12** | input\_shape=(15,120,120,3)  activation = elu  conv layer 1- 16 features  conv layer 2- 32 features, dropout=0.30  conv layer 3- 64 features, dropout=0.45  Dense layer - 100 neurons dropout= 0.4  Dense layer - 200 neurons dropout= 0.55  Total params: 754,201  Trainable params: 753,377  Non-trainable params: 824  min\_lr=0.00000001  batch\_size=20  num\_epochs=45 | **categorical\_accuracy: 0.8839**  **val\_categorical\_accuracy: 0.8300**  **lr: 3.2000e-07** | **Reducing parameters has reduced complexity and reduced overfitting and improved validation accuracy.**  **This indicates there was lot of redundancy in fully connected layers.**  **we can can further reduce neurons in fully connected layers to reduce overfitting** |
| **13** | input\_shape=(15,120,120,3)  activation = elu  conv layer 1- 16 features  conv layer 2- 32 features, dropout=0.30  conv layer 3- 64 features, dropout=0.45  Dense layer - 80 neurons dropout= 0.4  Dense layer - 160 neurons dropout= 0.55  Total params: 618,501  Trainable params: 617,797  Non-trainable params: 704  min\_lr=0.00000001  batch\_size=20  num\_epochs=45 | **categorical\_accuracy: 0.9532**  **val\_categorical\_accuracy: 0.9100**  **lr: 1.6000e-06** | **Reducing parameters has reduced complexity and reduced overfitting and improved validation accuracy.**  **This indicates that there was lot of redundancy in fully connected layers.**  **can further reduce neurons in fully connected layers to reduce overfitting** |
| **14** | input\_shape=(15,120,120,3)  activation = elu  conv layer 1- 16 features  conv layer 2- 32 features, dropout=0.30  conv layer 3- 64 features, dropout=0.45  Dense layer - 70 neurons dropout= 0.4  Dense layer - 140 neurons dropout= 0.55  Total params: 551,251  Trainable params: 550,607  Non-trainable params: 644  min\_lr=0.00000001  batch\_size=20  num\_epochs=50 | **categorical\_accuracy: 0.8054**  **val\_categorical\_accuracy: 0.7800**  **lr: 6.4000e-08** | **Not a good idea to reduce the neurons, As it led to reduced learning.**  **hence we will restore the previous model and increase the epochs** |
| **15** | input\_shape=(15,120,120,3)  activation = elu  conv layer 1- 16 features  conv layer 2- 32 features, dropout=0.30  conv layer 3- 64 features, dropout=0.45  Dense layer - 80 neurons dropout= 0.4  Dense layer - 160 neurons dropout= 0.55  Total params: 618,501  Trainable params: 617,797  Non-trainable params: 704  min\_lr=0.00000001  batch\_size=20  num\_epochs=50 | **categorical\_accuracy: 0.8944**  **val\_categorical\_accuracy: 0.7900**  **lr: 1.6000e-06** | **Should be again decreasing the epochs, As more epochs has lead to increase in overfitting**  **Can reduce batch size slightly** |
| **16** | input\_shape=(15,120,120,3)  activation = elu  conv layer 1- 16 features  conv layer 2- 32 features, dropout=0.30  conv layer 3- 64 features, dropout=0.45  Dense layer - 80 neurons dropout= 0.4  Dense layer - 160 neurons dropout= 0.55  Total params: 618,501  Trainable params: 617,797  Non-trainable params: 704  min\_lr=0.00000001  batch\_size=18  num\_epochs=45 | **categorical\_accuracy: 0.9095**  **val\_categorical\_accuracy: 0.8200**  **lr: 3.2000e-07** | **There is clear improvement in validation accuracy and also reduction in overfitting** |
| **17** | input\_shape=(15,120,120,3)  activation = elu  conv layer 1- 16 features  conv layer 2- 32 features, dropout=0.30  conv layer 3- 64 features, dropout=0.45  Dense layer - 80 neurons dropout= 0.4  Dense layer - 160 neurons dropout= 0.55  Total params: 618,501  Trainable params: 617,797  Non-trainable params: 704  min\_lr=0.00000001  batch\_size=18  num\_epochs=50 | **categorical\_accuracy: 0.7572**  **val\_categorical\_accuracy: 0.1700**  **lr: 1.0000e-08** | **Model didn’t learned anything**  **From the graph of Experiment 16, it is evident that model stabilizes between 25 and 30 epochs.**  **Hence we can try some epoch between 25 and 30 in next experiment.** |
| **18** | input\_shape=(15,120,120,3)  activation = elu  conv layer 1- 16 features  conv layer 2- 32 features, dropout=0.30  conv layer 3- 64 features, dropout=0.45  Dense layer - 80 neurons dropout= 0.4  Dense layer - 160 neurons dropout= 0.55  Total params: 618,501  Trainable params: 617,797  Non-trainable params: 704  min\_lr=0.00000001  batch\_size=18  num\_epochs=27 | **categorical\_accuracy: 0.8854**  **val\_categorical\_accuracy: 0.8200**  **lr: 2.0000e-04** | **Final model has stabilized as predicted from experiment 16, but the best model remains that of experiment 13. Hence we can finalize the model of Experiment-13 as final model.** |
| **Final Model** | **Based on Experiment 13**  input\_shape=(15,120,120,3)  activation = elu  conv layer 1- 16 features  conv layer 2- 32 features, dropout=0.30  conv layer 3- 64 features, dropout=0.45  Dense layer - 80 neurons dropout= 0.4  Dense layer - 160 neurons dropout= 0.55  Total params: 618,501  Trainable params: 617,797  Non-trainable params: 704  min\_lr=0.00000001  batch\_size=20  num\_epochs=45 | **categorical\_accuracy: 0.9532**  **val\_categorical\_accuracy: 0.9100**  **lr: 1.6000e-06** | **train accuracy =95.32%**  **Val accuracy = 91%** |

### Conv3D Results

After conducting many experiments with different combinations of batch\_size, epochs, min\_lr, drop-outs and a different number of parameters in dense layers we arrived at a model with the best results which consist of 3 conv3D layers and 2 densely connected layers.

The total number of Trainable parameters are 617,797, which is the lowest among all the models that were tried. The size of H5 file is 7.2MB which is also the smallest among the H5 files of other models tried.

The parameters used for the best model is

Frames\_used=15

batch\_size=20

epoch=45

min\_lr=0.00000001

# 

# Conclusion

The following table shows the comparison of the 4 different models used and the best results.

|  |  |  |
| --- | --- | --- |
| Model | Trainable Params | Val Accuracy |
| Resnet50 + CNN + RNN | Trainable params: 5,353,557  Non-trainable params: 19,123,808  Params:  Frames\_used=18 (randomly selected)  batch\_size=32  epoch=50  min\_lr=0.00000001 | 62% |
| Custom CNN + RNN | Trainable params: 1,345,413  Non-trainable params: 992  Params:  Frames\_used=18 (randomly selected)  batch\_size=20  epoch=50  min\_lr=0.00000001 | 85% in first run and 81% on final model re-run  Size of h5 file = 15.5 MB |
| MobileNet + RNN | Trainable params: 9,137,477  Non-trainable params: 1,051,904  Params:  Frames\_used=18 (randomly selected)  batch\_size=20  epoch=50  min\_lr=0.0000000001 | 96% in first run and 92% on final model re-run  Size of h5 file = 108.9 MB |
| Conv3D CNN | Trainable params: 617,797  Non-trainable params: 704  Params:  Frames\_used=15  batch\_size=20  epoch=45  min\_lr=0.0000000001 | 91% in first run and 84% on final model re-run  Size of h5 file = 7.2 MB |

Since the model needs to be deployed on a webcam, it needs to be of very small size and be reasonably accurate. Conv3D model is the smallest at 7.2 MB with an accuracy of 91% in the first experiment and 84% on the final re-run. Hence it can operate reasonably well for this application. If the webcam can support larger models, then MobileNet + RNN model can be used which has an accuracy of 96% and 92%.

# Submission Files

The following h5 files are submitted for 2 models along with 2 notebooks. Resnet50 model is not submitted since its accuracy was poor and Mobilenet was not submitted because of its large size.

1. Conv3D Model - **Conv3D\_GestureRecognitionAssignment.ipynb**
2. CNN+RNN Model - **GestureRecognitionAssignment-CNN\_RNN.ipynb**

|  |  |
| --- | --- |
| Model | File |
| Custom CNN+RNN | model-00024-0.39372-0.85671-0.61336-0.81000.h5 |
| Conv3D | model-00043-0.39105-0.85671-0.48040-0.84000.h5 |