# Gesture Recognition Case Study

## Team Member:

- Manish Kumar
- Pratik Patil

## Problem Statement

Imagine you are working as a data scientist at a home electronics company which manufactures state of the art smart televisions. You want to develop a cool feature in the smart-TV that can recognise five different gestures performed by the user which will help users control the TV without using a remote.

The gestures are continuously monitored by the webcam mounted on the TV. Each gesture corresponds to a specific command:

| Gesture | Corresponding Action |
|---:|---:|
| Thumbs Up | Increase the volume. |
| Thumbs Down | Decrease the volume. |
| Left Swipe | 'Jump' backwards 10 seconds. |
| Right Swipe | 'Jump' forward 10 seconds. |
| Stop | Pause the movie. |

Each video is a sequence of 30 frames (or images).

**Objectives:**

1. **Generator**: The generator should be able to take a batch of videos as input without any error. Steps like cropping, resizing and normalization should be performed successfully.
2. **Model**: Develop a model that is able to train without any errors which will be judged on the total number of parameters (as the inference(prediction) time

should be less) and the accuracy achieved. As suggested by Snehansu, start training on a small amount of data and then proceed further.

3. **Write up**: This should contain the detailed procedure followed in choosing the final model. The write up should start with the reason for choosing the base model, then highlight the reasons and metrics taken into consideration to modify and experiment to arrive at the final model.

## Packages used:

- *Imageio*
- *Skimage*
- *Datetime*
- *Numpy*
- *OS*
- *Abc*
- *OpenCV*
- *Matplotlib*
- *random*
- *Tensorflow*
- *Keras*
- *Mobilenet*

## Models

Here you make the model using different functionalities that Keras provides. Remember to use `Conv3D` and `MaxPooling3D` and not `Conv2D` and `Maxpooling2D` for a 3D convolution model. You would want to use `TimeDistributed` while building a Conv2D + RNN model. Also remember that the last layer is the softmax. Design the network in such a way that the model is able to give good accuracy on the least number of parameters so that it can fit in the memory of the webcam.

## 1. CNN 3D - 3D Convolution Models

### Model 1 - Sample model used for initial experiments

The sample model provides a starting point for the experiments to understand various hyperparameters. Based on the results of the models, we can optimize the hyperparameters and get the better accuracies and reduce the loss.

*Below are the experiments to see how training time is affected by*

1. image resolution,
2. number of images in sequence and
3. batch size

A. As we see from the above experiments **"image resolution"** and **number of frames** in sequence have more impact on training time than **batch_size**
B. We can consider the Batch Size around 15-40 based on model performance
C. We will change the resolution 160 X 160, 120 X 120, 100 X 100 and 80 X 80 according the model performance

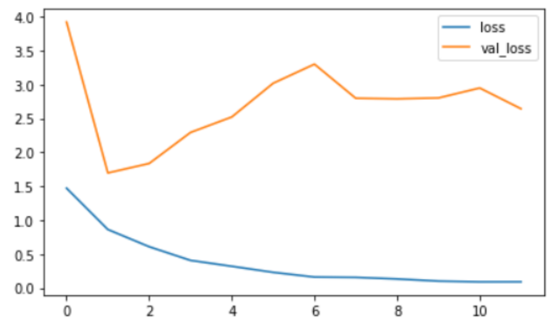
# Model 2

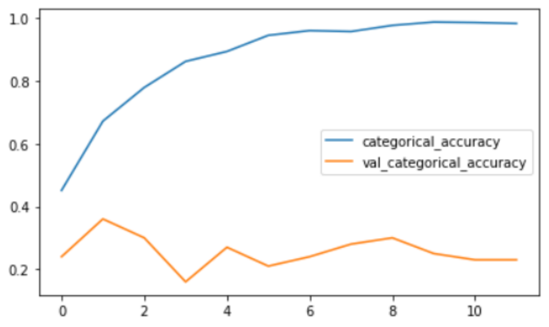
## Base Model

**Image Size = 160 X 160, Batch Size = 40, No. of Epochs = 15, Frames to Sample = 20**

To start with the experiments of building models using CNN3D, the initial parameters are set. Once the model is executed, we can start optimizing hyperparameters based on the model performance.

**<u>Training & Validation Accuracy/Loss Plot</u>**





*Model is clearly overfitting. There have not been any improvement in the model since last 10 epoch, so early stopping is executed @epoch 12. The model is not able to converge and hence there is no point in executing this model. We will reduce the batch size and increase the number of epochs to see if there is any improvement in the model. Also, we can add dropout 50% to address overfitting*

1. Total No. of parameters = 65,833,861
2. Categorical Accuracy = 98.19%
3. Validation Accuracy = 23.00%

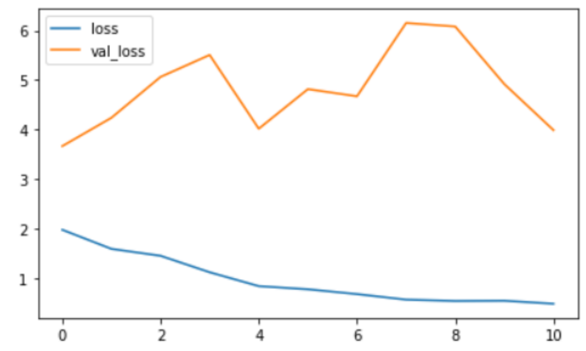
# Model 3

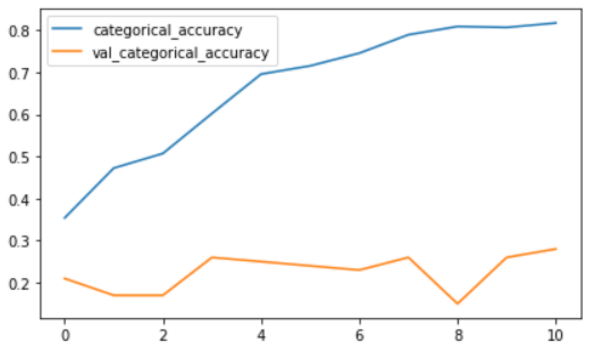
## Adding dropout to Base Model

*Image Size = 160 X 160, Batch Size = 20, No. of Epochs = 25, Frames to Sample = 20, Dense Neurons = 256, Dropout = 50%*

Batch size is reduced, and number of epochs have been increased based on the above model. Also added dropout of 50% to address overfitting. We will now check the performance of this model and then further optimize.

**Training & Validation Accuracy/Loss Plot**





*We can see val_loss did not improve for last 10 epochs, so early stopping stops the epoch automatically!! The model is highly overfit as well with no signs or convergence. Changing batch size and number of epochs did not help at all. We will reduce the image size to 120 X 120 and see if the model is converging. Also, we can test with reduced filter size.*

1. Total No. of parameters = 262,506,181
2. Categorical Accuracy = 81.67%
3. Validation Accuracy = 28.00%

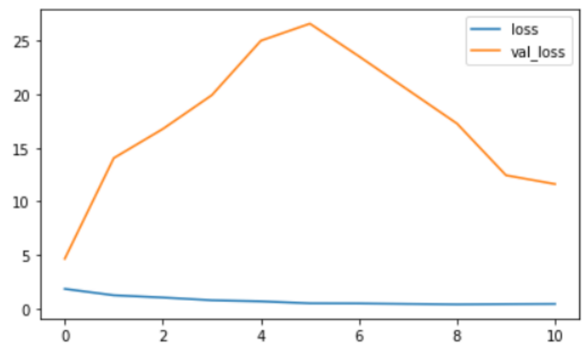
## Model 4

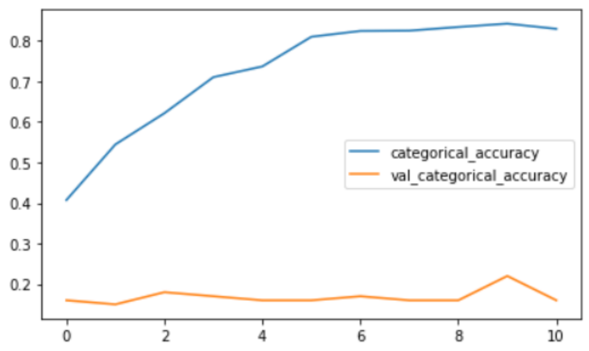
### Reduce filter size to (2,2,2) and image size to 120 x 120

*Image Size = 120 X 120, Batch Size = 30, No. of Epochs = 25, Frames to Sample = 16*

Image size is reduced, and filter size is also reduced. Now we will check the model performance and then further optimize.

**Training & Validation Accuracy/Loss Plot**





*The model performance still did not improve at all, instead, there is dip in categorical accuracy. This model also stopped early, without executing all the epochs. The model is clearly not converging at all. We will go back to filter size of (3,3,3) and reduce the number of epochs, since all previous models have stopped early. Also, adding dropout at all convolution layers as well.*

1. Total No. of parameters = 118,121,781
2. Categorical Accuracy = 82.96%
3. Validation Accuracy = 16.00%

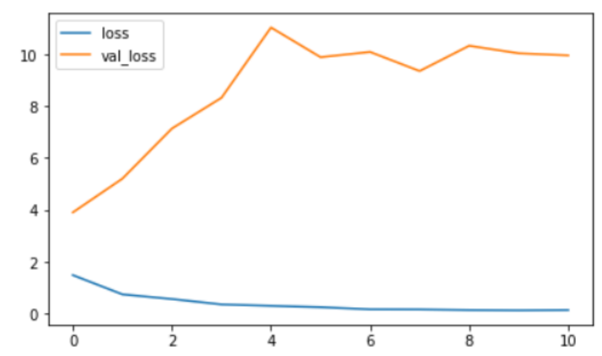
## Model 5

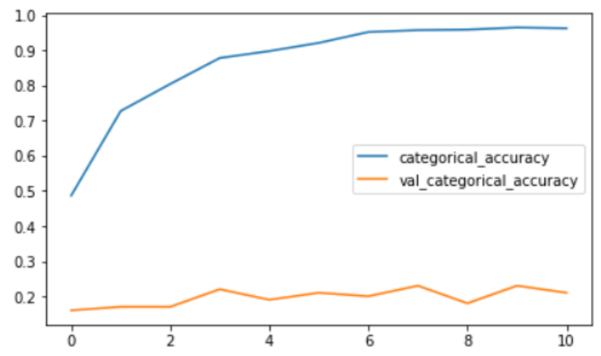
### Adding dropout at convolution layers

*Image Size = 120 X 120, Batch Size = 20, No. of Epochs = 15, Frames to Sample = 16*

Filter size is set back to (3,3,3) and number of epochs have been reduced to 15. Adding dropout at all the convolution layers as well. We will now check the model performance.

<u>**Training & Validation Accuracy/Loss Plot**</u>





*Overfitting again!! Adding dropouts have also not helped. The model doesn't seem to generalise well. All the experimental models above have more than 1 million parameters. Let's try to reduce the model size and see the performance*

1. Total No. of parameters = 118,326,981
2. Categorical Accuracy = 96.23%
3. Validation Accuracy = 21.00%

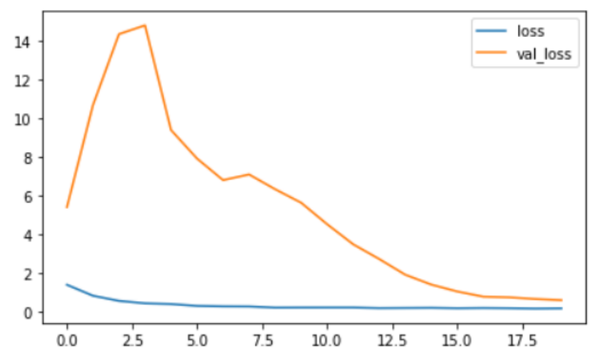
## Model 6

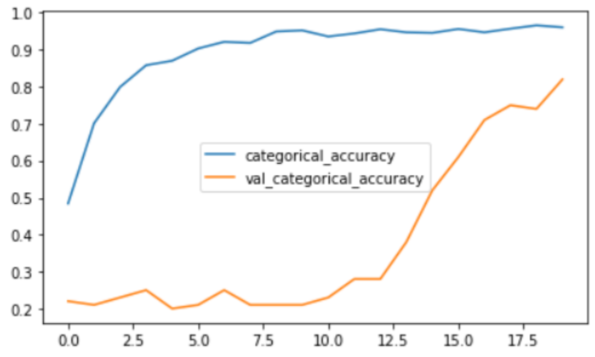
### Reducing the number of parameters and image size

*Image Size = 100 X 100, Batch Size = 20, No. of Epochs = 20, Frames to Sample = 16, Dense Neurons = 128, Dropout = 25%*

Image Size, Dense Neurons and dropout have been reduced. We will check the model performance and optimize further

**Training & Validation Accuracy/Loss Plot**





*Reducing the image size and number of parameters have really helped the model to converge. We have got a good validation accuracy of 82%. Though the model is still overfitting. We can reduce the number of parameters again and check if we can better this accuracy*

1. Total No. of parameters = 41,066,821
2. Categorical Accuracy = 96.08%
3. Validation Accuracy = 82.00%

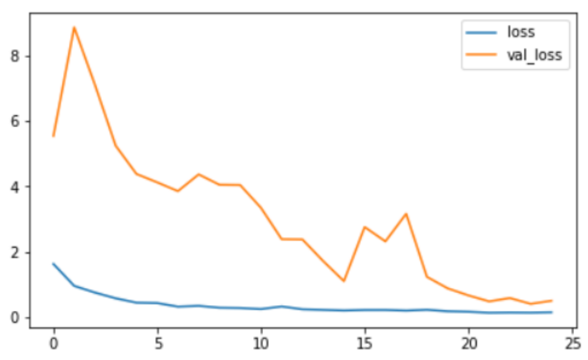
## Model 7

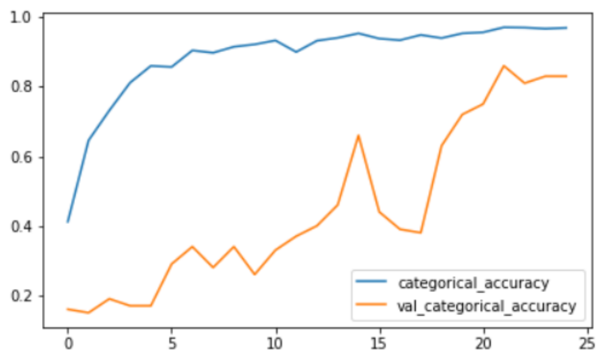
### Reducing the number of parameters again

*Image Size = 100 X 100, Batch Size = 20, No. of Epochs = 25, Frames to Sample = 16, Dense Neurons = 64, Dropout = 25%*

Further reducing the dense layer and dropout. We will check the model performance and optimize further.

**<u>Training & Validation Accuracy/Loss Plot</u>**





*Reducing the parameters further did not help much compared to previous model. We have similar accuracies. So, now, we will further reduce the image size to 80 X 80 and reduce the number of hidden layers.*

1. Total No. of parameters = 20,583,301
2. Categorical Accuracy = 96.91%
3. Validation Accuracy = 83.00%

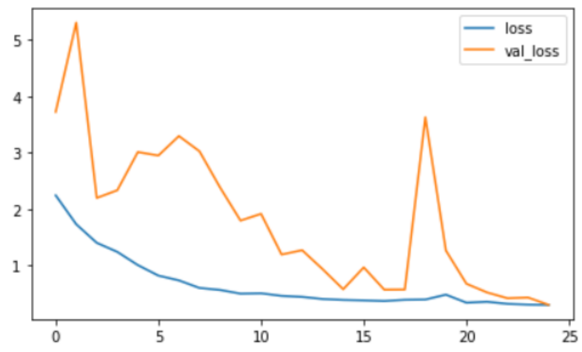
## Model 8

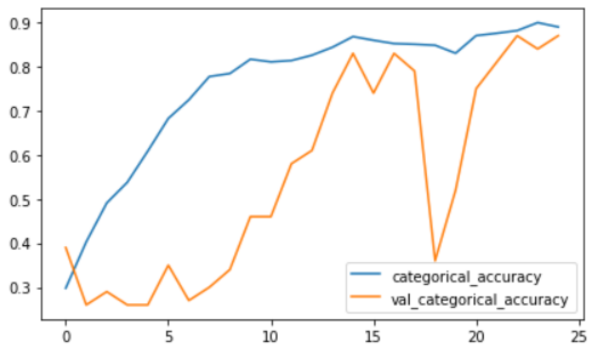
### Reducing number of hidden layers and image size = 80 X 80

*Image Size = 80 X 80, Batch Size = 15, No. of Epochs = 25, Frames to Sample = 16, Dense Neurons = 128, Dropout = 50%*

Removing a CNN3D layer and reducing image size to 80 X 80. We have to reduce the batch size to 15. With batch size 20, disk space of more than 20GB was supported at Jarvis cloud. We were running out of space after epoch 20 itself. So, reducing batch size to 15.

**<u>Training & Validation Accuracy/Loss Plot</u>**





*The model is finally fit with a very good training accuracy of ~89% and validation accuracy of 87%. For now, this seems to be a very good accuracy and we finally have our best CNN3D model.*

1. Total No. of parameters = 210,013,701
2. Categorical Accuracy = 88.99%
3. Validation Accuracy = 87.00%

*Now, we can try experimenting and building models in CNN2D + RNN.*

## 2. CNN 2D + RNN Models

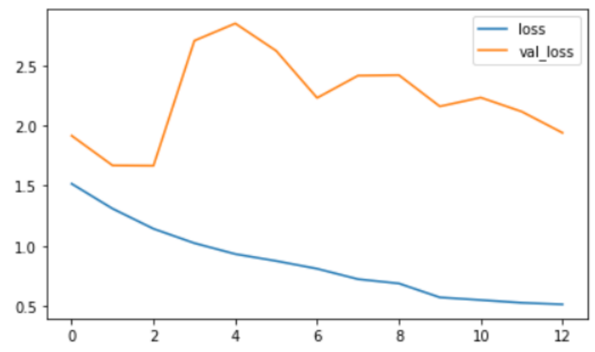
## Model 9

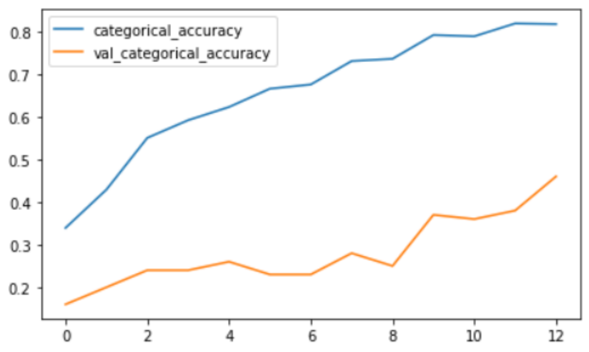
### CNN2D - LSTM Model

*Image Size = 120 X 120, Batch Size = 20, No. of Epochs = 20, Frames to Sample = 18, Dense Neurons = 128, Dropout = 25%, LSTM Cells = 128*

Initially to start with, we have less number of layers configured for CNN2D and image size is 120X120. We will check the model performance and optimize further.

**Training & Validation Accuracy/Loss Plot**





*The model started to converge but could not. It stooped early as there were no convergence for 10 epoch. The model is overfit with a huge gap in training and validation accuracies. We will try and reduce the image size to 100 x 100 and check the performance.*

1. Total No. of parameters = 14,922,949
2. Categorical Accuracy = 81.75%
3. Validation Accuracy = 46.00%

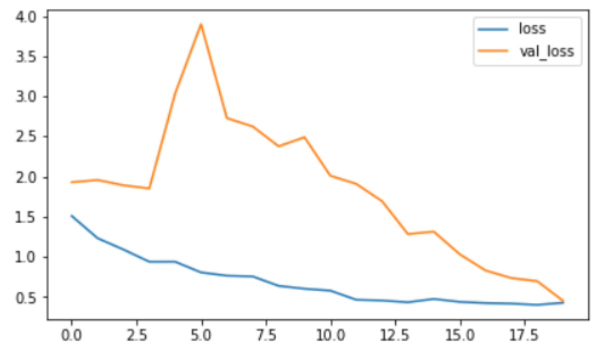
## Model 10

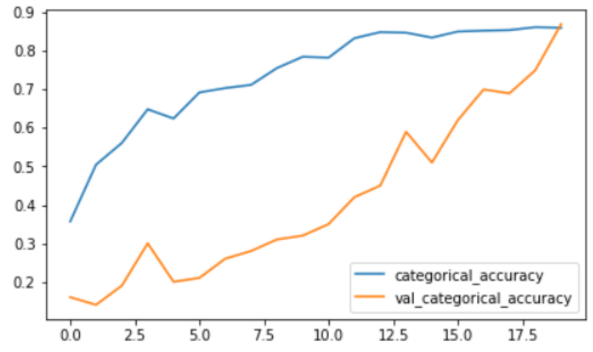
### CNN - LSTM Model - Reducing image size

*Image Size = 100 X 100, Batch Size = 20, No. of Epochs = 20, Frames to Sample = 18, Dense Neurons = 128, Dropout = 25%, LSTM Cells = 128*

Reducing the image size to 100 x 100 and checking its performance.

**Training & Validation Accuracy/Loss Plot**





*We have got a good training accuracy of 86% and validation accuracy of 87%. The model is fit and seems to be best model in CNN2D + RNN combination. We will reduce the image size further to check if we can achieve better accuracy.*

1. Total No. of parameters = 9,614,533
2. Categorical Accuracy = 86.05%
3. Validation Accuracy = 87.00%

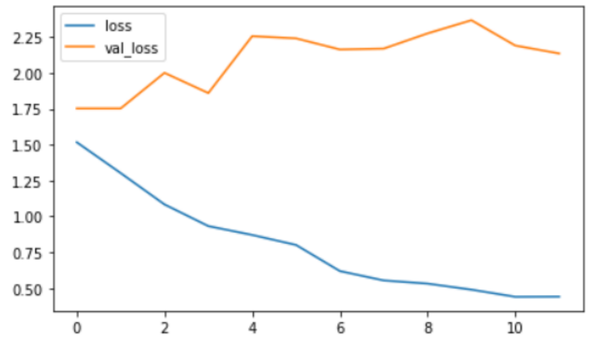
## Model 11

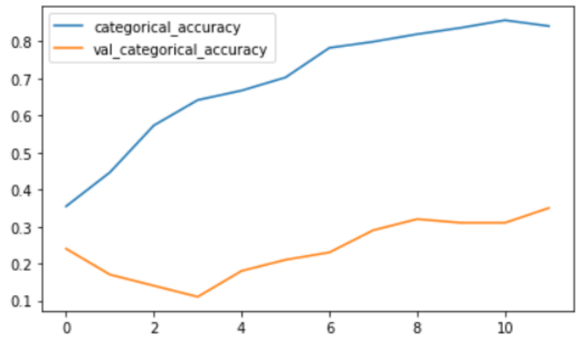
### CNN2D - LSTM Model - Further reducing image size

*Image Size = 80 X 80, Batch Size = 20, No. of Epochs = 20, Frames to Sample = 18, Dense Neurons = 128, Dropout = 25%, LSTM Cells = 128*

Reducing the image size further to check if it improves the accuracy further.

**Training & Validation Accuracy/Loss Plot**





*The model is overfit, failing to converge further. It stopped early as there were no convergence for last 10 epochs. So, clearly, image size of 80x80 has not improved, instead, drastically reduced the performance.*

1. Total No. of parameters = 6,730,949
2. Categorical Accuracy = 84.16%
3. Validation Accuracy = 35.00%

*We will try now building model with GRU and check its performance.*

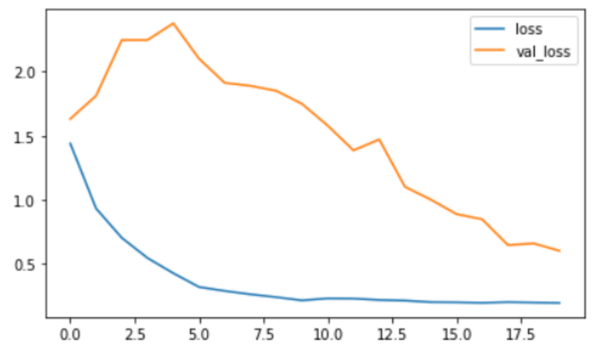
## Model 12

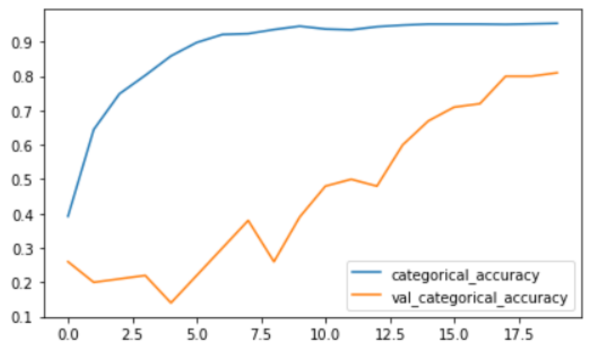
### CNN2D LSTM with GRU

*Image Size = 100 X 100, Batch Size = 20, No. of Epochs = 20, Frames to Sample = 18, Dense Neurons = 128, Dropout = 25%, LSTM Cells = 128*

Taking the best CNN LSTM model (model with image size 100 x 100) and adding GRU layer to it. Checking its performance

**Training & Validation Accuracy/Loss Plot**





*Adding GRU layer improved the training accuracy but did not improve the validation accuracy, hence overfitting the model.*

1. Total No. of parameters = 1,934,949
2. Categorical Accuracy = 95.40%
3. Validation Accuracy = 81.00%

## 3. Transfer Learning Models

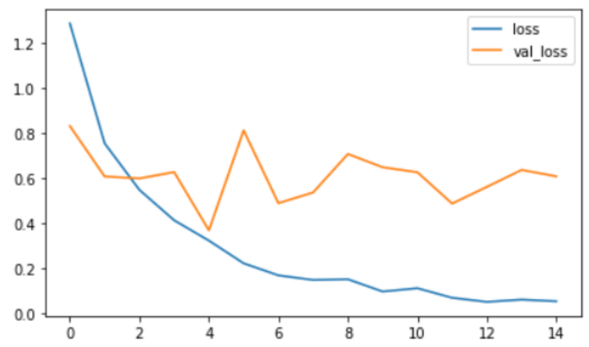
## Model 13 - Transfer Learning

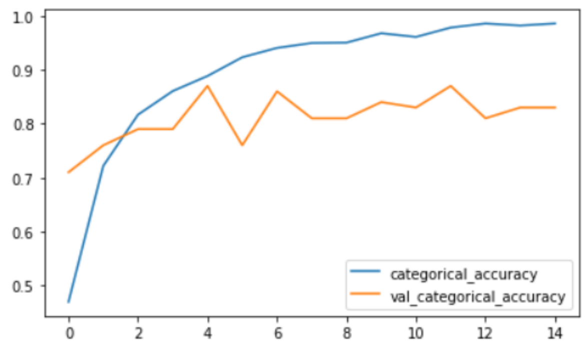
### Using Mobilenet

*Image Size = 100 X 100, Batch Size = 5, No. of Epochs = 20, Frames to Sample = 16, Dense Neurons = 128, Dropout = 25%, LSTM Cells = 128*

**(Optional)**

**Training & Validation Accuracy/Loss Plot**





*We are not training the mobilenet weights and we see validation accuracy is poor and model if overfit. Let's train them as well and observe if there is performance improvement*

1. Total No. of parameters = 3,840,453
2. Categorical Accuracy = 98.57%
3. Validation Accuracy = 83.00%

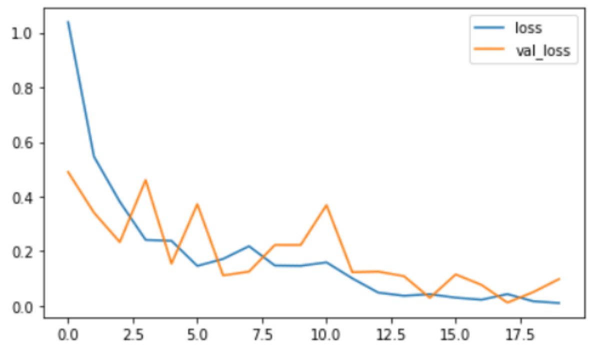
## Model 14 - Transfer Learning with GRU

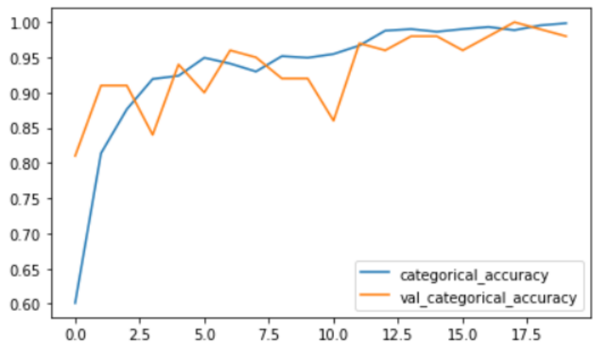
### Training all weights

*Image Size = 100 X 100, Batch Size = 5, No. of Epochs = 20, Frames to Sample = 16, Dense Neurons = 128, Dropout = 25%, LSTM Cells = 128*

**(Optional)**

**Training & Validation Accuracy/Loss Plot**



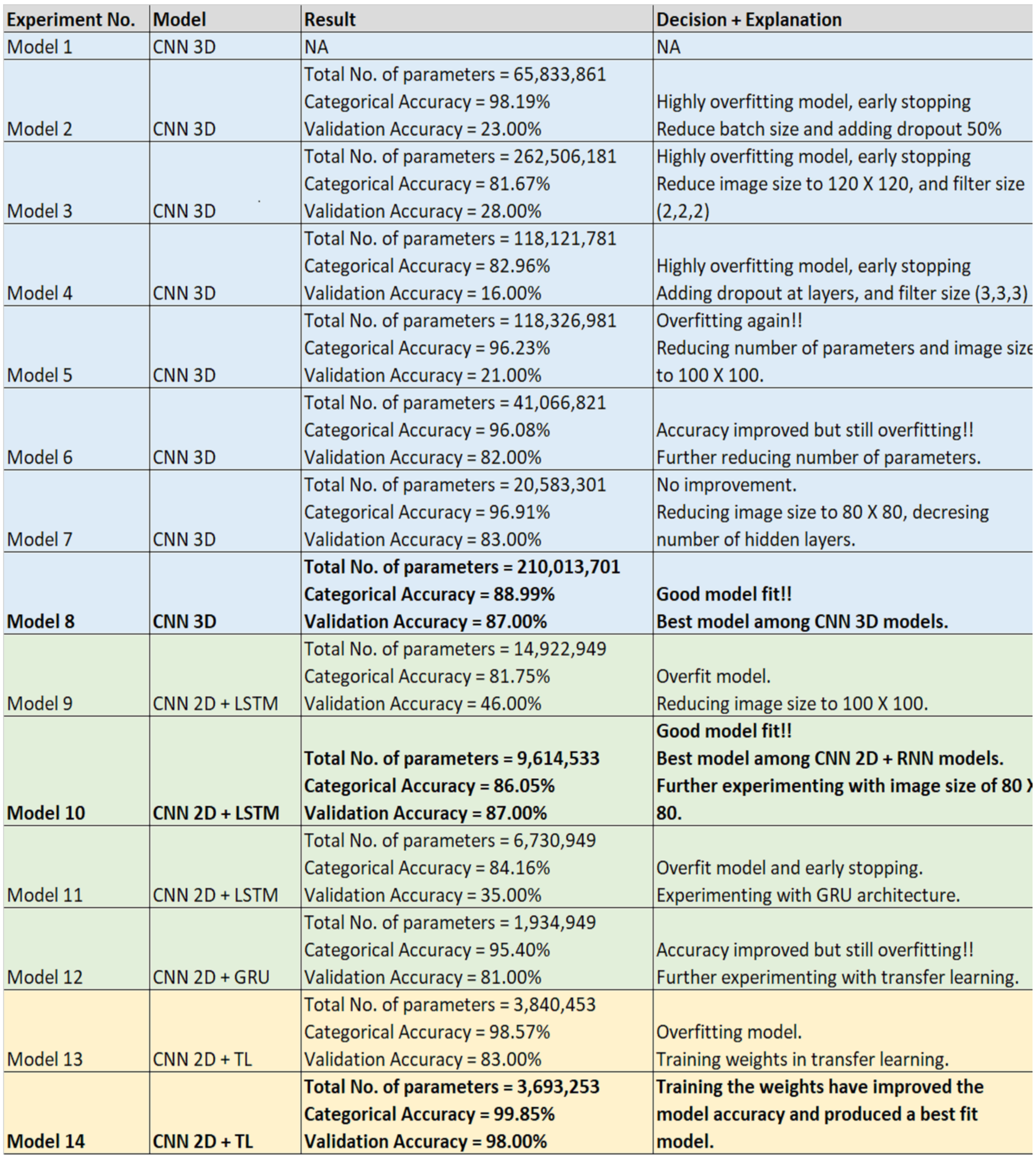


### Awesome results! 99.85% Training accuracy and 98% validation accuracy :)

*Training the weights have improved the model accuracy and produced a best fit model.*

1. Total No. of parameters = 3,693,253
2. Categorical Accuracy = 99.85%
3. Validation Accuracy = 98.00%

# Consolidated Model Summary



| Experiment No. | Model | Result | Decision + Explanation |
|---|---|---|---|
| Model 1 | CNN 3D | NA | NA |
| Model 2 | CNN 3D | Total No. of parameters = 65,833,861<br>Categorical Accuracy = 98.19%<br>Validation Accuracy = 23.00% | Highly overfitting model, early stopping<br>Reduce batch size and adding dropout 50% |
| Model 3 | CNN 3D | Total No. of parameters = 262,506,181<br>Categorical Accuracy = 81.67%<br>Validation Accuracy = 28.00% | Highly overfitting model, early stopping<br>Reduce image size to 120 X 120, and filter size (2,2,2) |
| Model 4 | CNN 3D | Total No. of parameters = 118,121,781<br>Categorical Accuracy = 82.96%<br>Validation Accuracy = 16.00% | Highly overfitting model, early stopping<br>Adding dropout at layers, and filter size (3,3,3) |
| Model 5 | CNN 3D | Total No. of parameters = 118,326,981<br>Categorical Accuracy = 96.23%<br>Validation Accuracy = 21.00% | Overfitting again!!<br>Reducing number of parameters and image size to 100 X 100. |
| Model 6 | CNN 3D | Total No. of parameters = 41,066,821<br>Categorical Accuracy = 96.08%<br>Validation Accuracy = 82.00% | Accuracy improved but still overfitting!!<br>Further reducing number of parameters. |
| Model 7 | CNN 3D | Total No. of parameters = 20,583,301<br>Categorical Accuracy = 96.91%<br>Validation Accuracy = 83.00% | No improvement.<br>Reducing image size to 80 X 80, decresing number of hidden layers. |
| **Model 8** | **CNN 3D** | **Total No. of parameters = 210,013,701**<br>**Categorical Accuracy = 88.99%**<br>**Validation Accuracy = 87.00%** | **Good model fit!!**<br>**Best model among CNN 3D models.** |
| Model 9 | CNN 2D + LSTM | Total No. of parameters = 14,922,949<br>Categorical Accuracy = 81.75%<br>Validation Accuracy = 46.00% | Overfit model.<br>Reducing image size to 100 X 100. |
| **Model 10** | **CNN 2D + LSTM** | **Total No. of parameters = 9,614,533**<br>**Categorical Accuracy = 86.05%**<br>**Validation Accuracy = 87.00%** | **Good model fit!!**<br>**Best model among CNN 2D + RNN models.**<br>**Further experimenting with image size of 80 X 80.** |
| Model 11 | CNN 2D + LSTM | Total No. of parameters = 6,730,949<br>Categorical Accuracy = 84.16%<br>Validation Accuracy = 35.00% | Overfit model and early stopping.<br>Experimenting with GRU architecture. |
| Model 12 | CNN 2D + GRU | Total No. of parameters = 1,934,949<br>Categorical Accuracy = 95.40%<br>Validation Accuracy = 81.00% | Accuracy improved but still overfitting!!<br>Further experimenting with transfer learning. |
| Model 13 | CNN 2D + TL | Total No. of parameters = 3,840,453<br>Categorical Accuracy = 98.57%<br>Validation Accuracy = 83.00% | Overfitting model.<br>Training weights in transfer learning. |
| **Model 14** | **CNN 2D + TL** | **Total No. of parameters = 3,693,253**<br>**Categorical Accuracy = 99.85%**<br>**Validation Accuracy = 98.00%** | **Training the weights have improved the model accuracy and produced a best fit model.** |

# Conclusion

**After doing all the experiments, we found that the Model 8 (CNN 3D) and Model 10 (Cnn2D + LSTM) have best results.**

**Lets compare these 2 models to get a better understanding, which will help us decide the best model between these two models.**

*Model 8 (CNN 3D):*

1. Total No. of parameters = 210,013,701
2. Categorical Accuracy = 88.99%
3. Validation Accuracy = 87.00%

*Model 10 (CNN2D + LSTM):*

1. Total No. of parameters = 9,614,533
2. Categorical Accuracy = 86.05%
3. Validation Accuracy = 87.00%

Based on the above details about the model, Model 10 (CNN2D + LSTM), is the BEST model, because:

1. Both models have similar accuracy, but model 10 was able to produce same result with very less number of parameters (only 9,614,533) as compared to Model 8 which uses much more parameters (210,013,701).
2. Model 10 uses LSTM, which is better suited to handle sequential data.
3. Since Model 10 has very less parameters, processing time required in Model 10 is much less compared to Model 8. i.e. Model 10 is much faster.
4. The image size used is Model 10 is 100X100 which is greater than the image size of Model 8, Hence not much information loss compared to Model 8.

**The best weights of CNN-LSTM: model_init_2022-05-1612_27_45.403017/model-00020-0.42984-0.86048-0.45405-0.87000.h5 (110 MB). We should consider this weight for model testing**