**EXPERIMENTS**

Model 1

* **Model**: Conv3D

Batch Size = 40 and No. of Epochs = 15

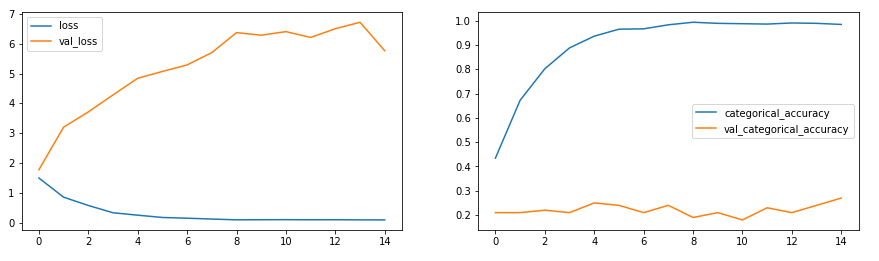
Image Resolution = 160x160

Frames to sample=20

* **Number of parameters**: 1,117,061
* **Result**:

Training accuracy: 0.9849

Validation accuracy: 0.2700



* **Decision + Explanation**:

Model is overfitting, increasing number of epochs and reducing the batch size

Model 2

* **Model**: Conv3D

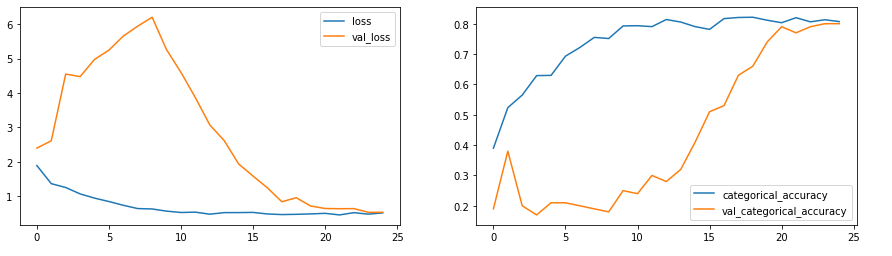
Batch size=20 and No. of epochs = 25

Image Resolution = 160x160

* **Number of parameters**: 3,638,981
* **Result**:

Training accuracy: 0.8069

Validation accuracy: 0.8000



* **Decision + Explanation**:

The validation accuracy of 80% and training accuracy of 80%. Good job!

Next, we will try to reduce the filter size and image resolution and see if get better results than this.

Moreover, since we see minor oscillations in loss, let's try lowering the learning rate to 0.0002

Model 3

* **Model**: Conv3D

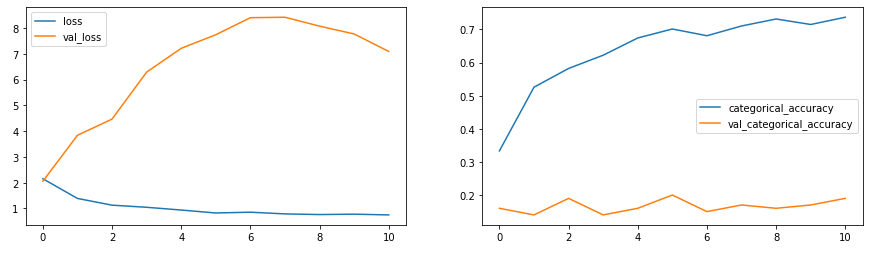
Batch size=30 and No. of epochs = 25

Image Resolution = 120x120

* **Number of parameters**: 1,762,613
* **Result:**

Training accuracy: 0.7368

Validation accuracy: 0.1900



* **Decision + Explanation**:

Model is significantly overfitting, adding more layers

Model 4

* **Model**: Conv3D

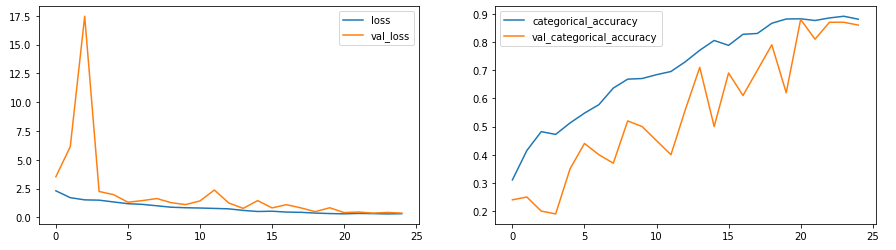
Batch size=20 and No. of epochs = 25

Image Resolution: 120x120

* **Number of parameters**: 2,556,533
* **Result:**

Training accuracy: 0.8808

Validation accuracy: 0.8600



* **Decision + Explanation**:

By adding more layers, we see much performance improvement. We get a best validation accuracy of 86% Let's try adding dropouts at the convolution layers in next model

Model 5

* **Model**: Conv3D

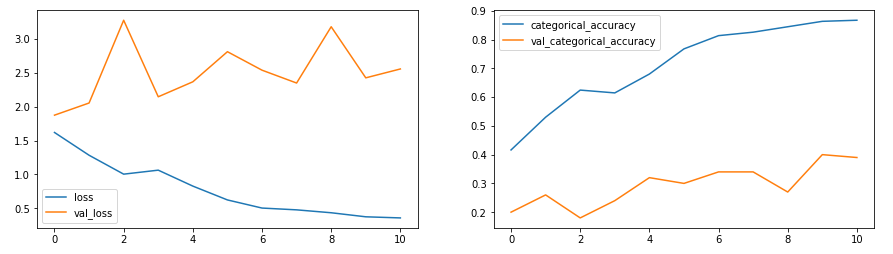
Batch size=20 and No. of epochs = 25

Image Resolution: 120x120

* **Number of parameters**: 2,556,533
* **Result:**

Training accuracy: 0.8673

Validation accuracy: 0.3900



* **Decision + Explanation**: By adding more dropout layers, model is overfitting. Let’s reduce the parameters

Model 6

* **Model**: Conv3D

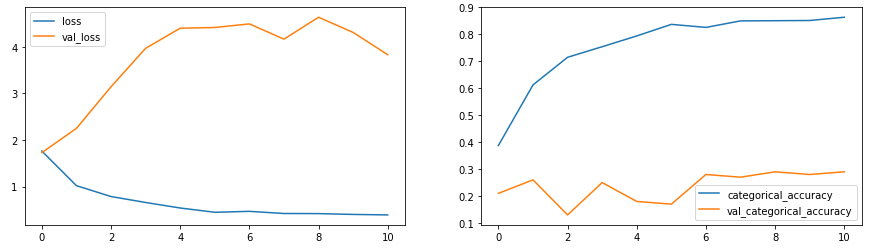
Batch size = 20 No. of epochs = 20

Image resolution = 100x100

* **Number of parameters**: 696,645
* **Result:**

Training accuracy: 0.8635

Validation accuracy: 0.2900



* **Decision + Explanation**:

By reducing the number of parameters, model is overfitting. Let’s further reduce the parameters to ascertain that the parameters is directly proportional to the accuracy

Model 7

* **Model**: Conv3D

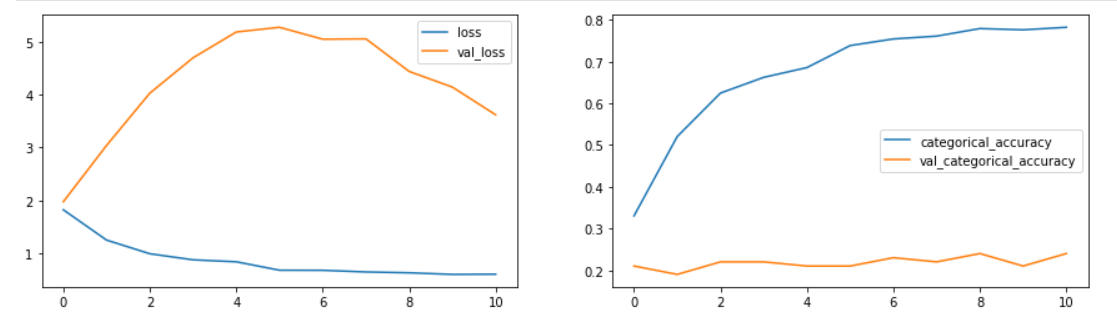
Batch size = 20 No. of epochs = 20

Image resolution = 100x100

* **Number of parameters**: 504,709
* **Result:**

Training accuracy: 0.7821

Validation accuracy: 0.2400



* **Decision + Explanation**:

Further as we reduce the number of parameters, the validation accuracy worsens. Let’s try out a sequential model

Model 8

* **Model**: CNN- LSTM Model

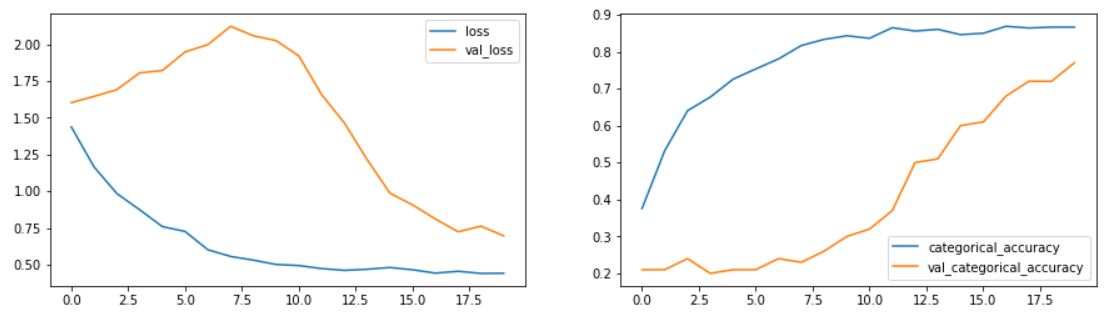
Batch size = 20 No. of epochs = 20

Image resolution = 120x120

* **Number of parameters**: 1,657,445
* **Result:**

Training accuracy: 0.8665

Validation accuracy: 0.7700



* **Decision + Explanation**: With the help of CNN-LSTM model, validation accuracy improved to 77%

Final Model

* **Model**: Conv3D

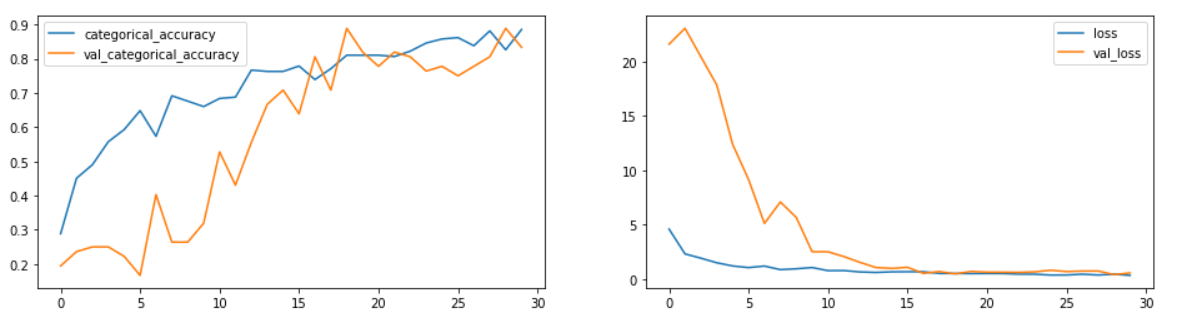
Batch size = 64 No. of epochs = 30

Image Resolution=84x84

* **Number of parameters**: 9,440,773
* **Result:**

Training accuracy: 88.54%

Validation accuracy: 83.33%



* **Decision + Explanation**:

I have selected model from experiment 9 as my final model for the following reasons:

* Using mean subtraction as normalizing technique for the batch gave substantially better performance than dividing pixels by 255. (Similar to VGG\_ILSVRC\_16\_layers Model, where BGR values are subtracted with [103.939, 116.779, 123.68])
* Model is able to capture the gesture from the alternative frames than last 18 consecutive frames.
* I used epoch=30 and batch size=64 for limitation of computational resources.
* Among different values, initial learning rate of 0.001 and momentum = 0.9 gave best accuracy.
* Adding further dropouts is not improving performance
* Most importantly both training and validation accuracy > 0.80