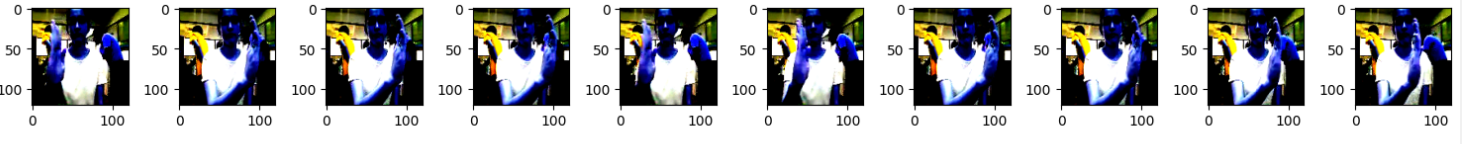
# GESTURE RECOGNITION ASSIGNMENT

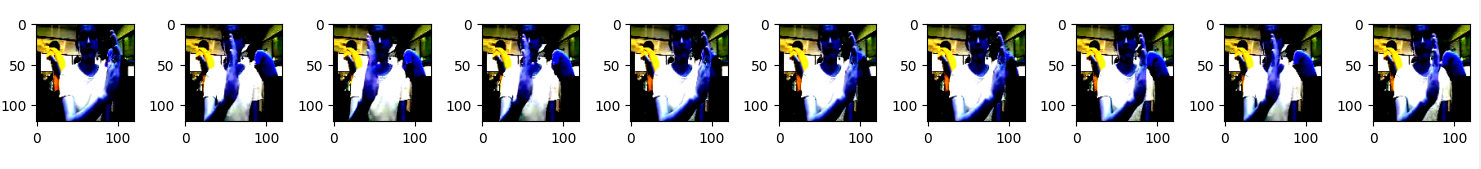
## GENERATOR FUNCTION

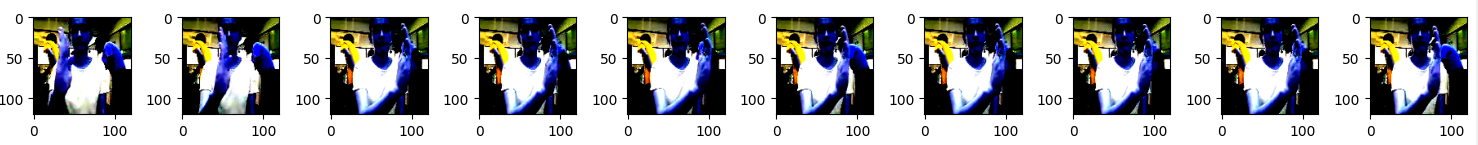
The generator function has been designed in such a way that it calculates how many total number of batches it needs to complete 1 epoch and then returns 1 batch of data as it loops through. The various things implemented in the generator function is:

* **Normalization:** Normalization is done to get rid of the outliers and to optimize the training of the model.
* **Augmentation:** Augmentation has been done since the model may overfit and we would have to deal with it. A random variable is generated to deal with 2 different types of augmentation that we have implemented in this function.
* **Remainder file:** With respect to the batch size we are selecting and the total video length we have as input and validation set it may not be possible to get each and every video in the last epoch. Therefore, a condition for the remainder files has been provided in our code.

## PLOTTING THE IMAGES CREATED BY THE GENERATOR FUNCTION







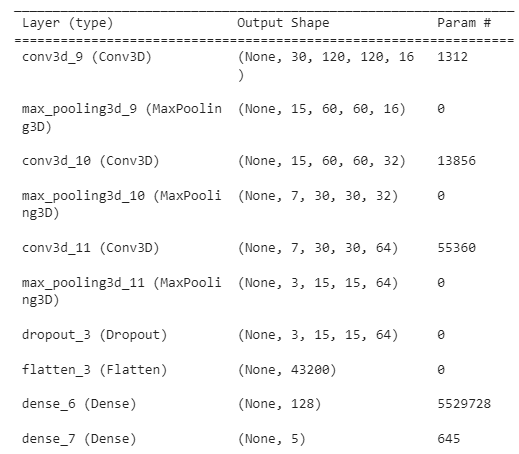
The above images represent 30 frames inside a video frame. This represents a short video with one command. In the above frames it is the left swipe.

## MODEL BUILDING

### CONV3D ARCHITECTURE

#### 1.EXPERIMENT-1

* The experiment follows a convolution layer with numbers of filters as 16,32,64 . Each followed by a Maxpool layer. Dropouts have been added and also number of neurons in the Dense layer is selected as 128.
* All the arguments and parameters of the experiment has been mentioned below. The experiment is clearly overfitting.
* Model Architecture

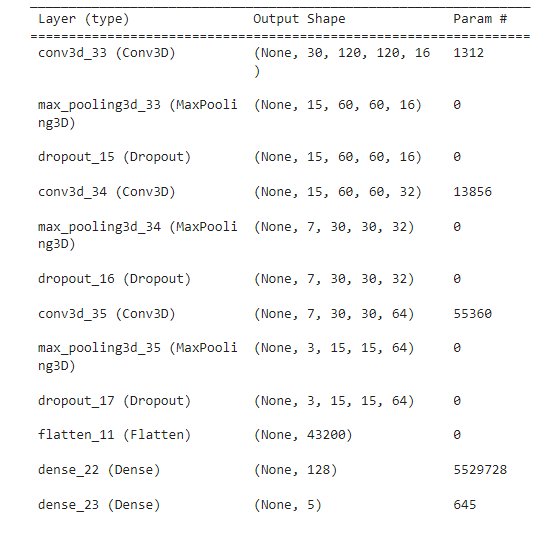


* Result

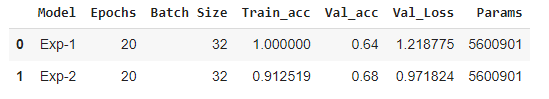


#### 2. EXPERIMENT-2

* In this experiment we have added Dropout layers after each convolution layer in hope of getting to reduce the overfitting we faced in earlier experiment.
* Model Architecture



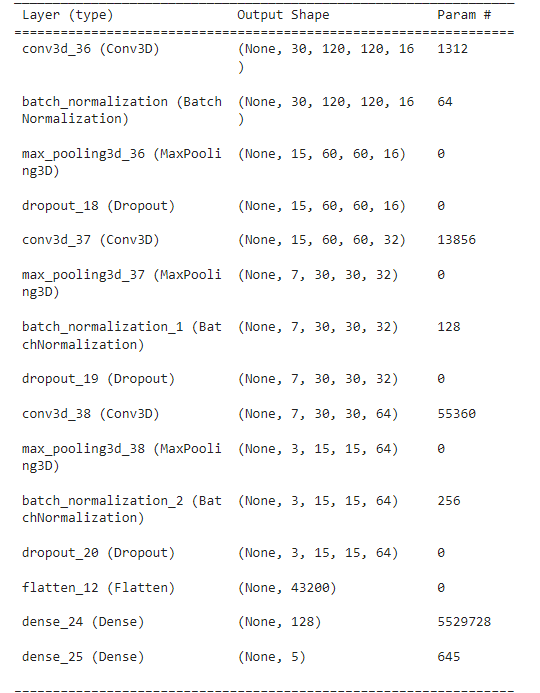
* Result🡪 Model is overfitting and apart from slightly improving the model it hasn’t left a significant mark.



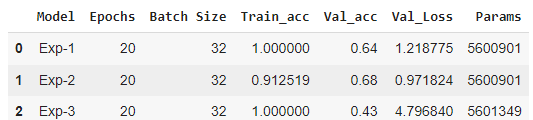
**Note: Exp-2** represents current experiment

#### 3.EXPERIMENT-3

* In Experiment-3 we have added batch normalization again in hopes of reducing overfitting.
* Model Architecture

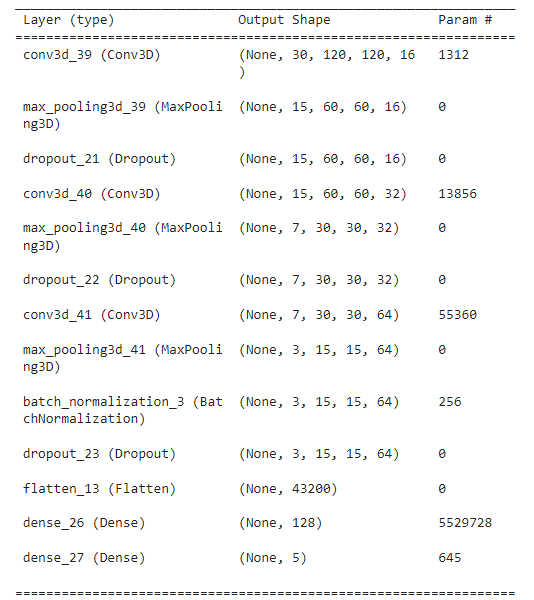


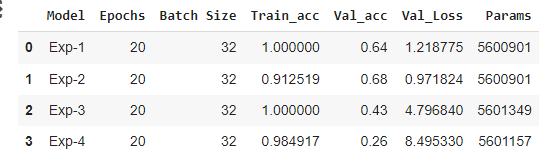
* Result🡪 Clearly adding batch normalization has made the matters worse.



#### 4.EXPERIMENT-4

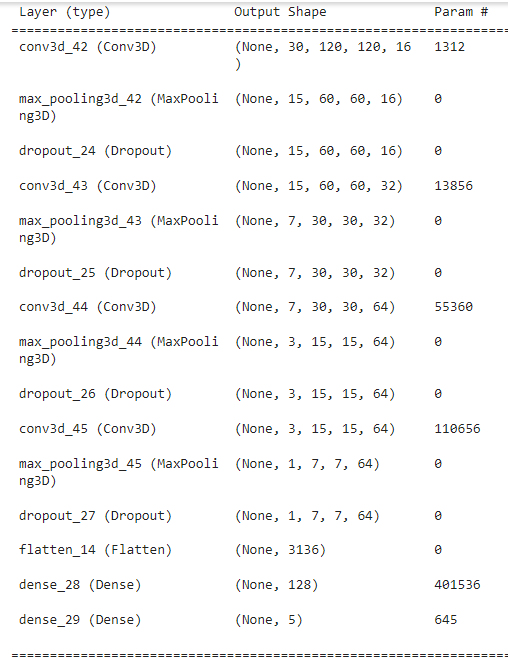
* In experiment-4 we have tweaked the model little bit by adding batch normalization to only one layer and increased dropout to 0.5 in the last layer.
* Model Architecture



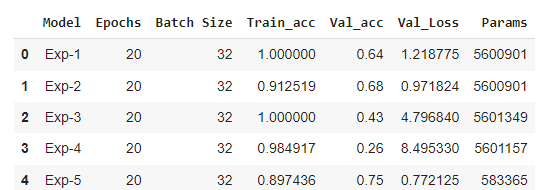
* Result🡪 Cleary this the worst model we have made until now.

#### 5. Experiment-5

* In Experiment-5 we have added an extra convolution layer and added dropouts to each layer in hopes of getting extra feature maps will improve the model performance.
* Model Architecture

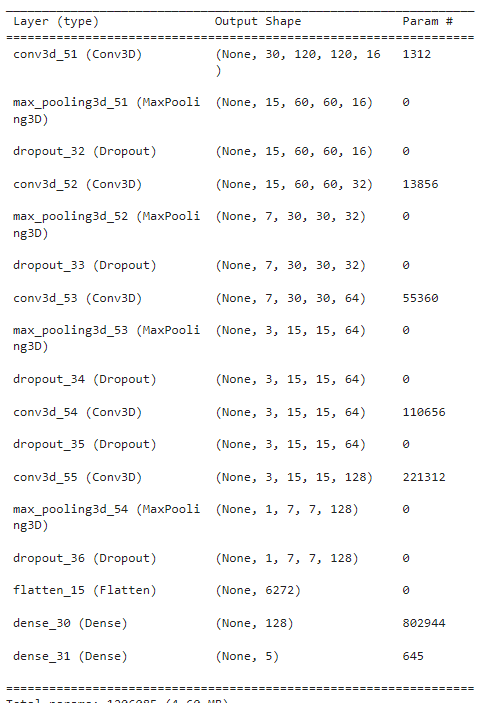


* Result🡪 We have improved the overfitting problem by a mile.

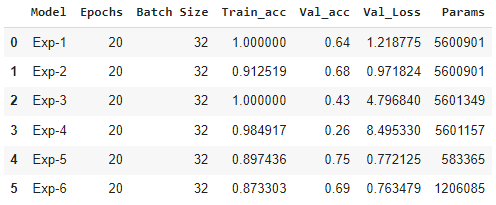


#### 6. EXPERIMENT-6

* In experiment-6 we have added the dropout layer after the dense layer to see if we can improve the overfitting situation.
* Model Architecutre

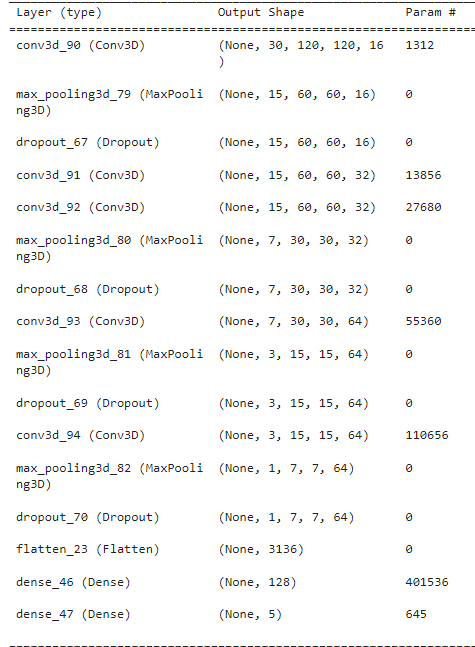


* Result🡪It has not improved the model performance as expected.

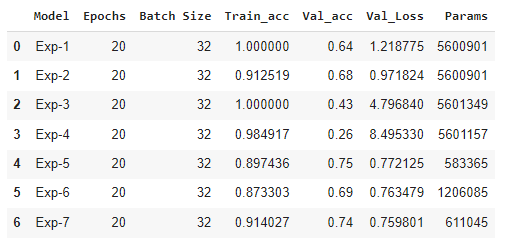


#### 7.EXPERIMENT-7

* In Experiment-7 we have further added one more convolutional layer of 32 features.
* Model Architecture

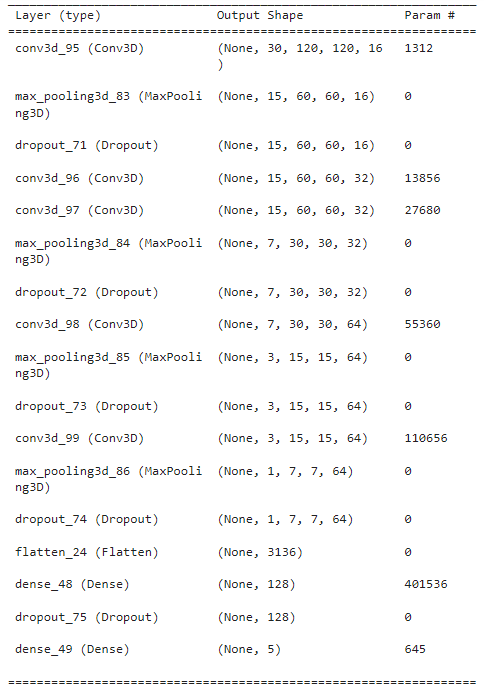


* Result🡪 Still there is overfitting, however the validation accuracy is one of the best we have got yet.
* One notable thing is the significant reduction of model parameters as compared to the previous models we have built.



#### 8. EXPERIMENT-8

* We have added a dropout layer to the above model itself after the dense layer in hope of getting overfitting reduced.
* Model Architecture

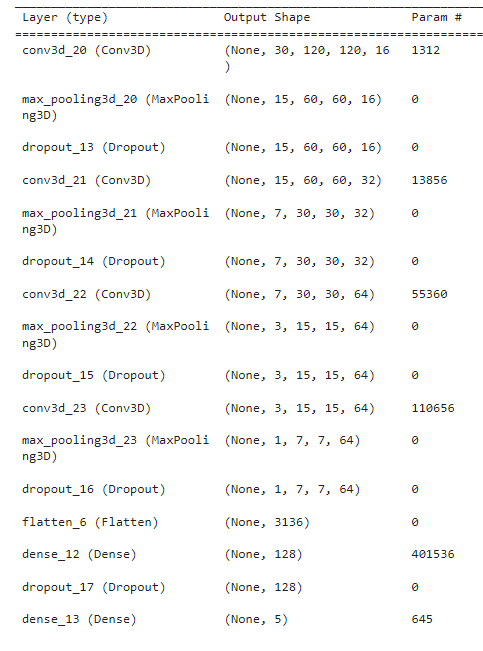


* Result🡪We are able to reduce the overfitting significantly however we haven’t completely got rid of it.

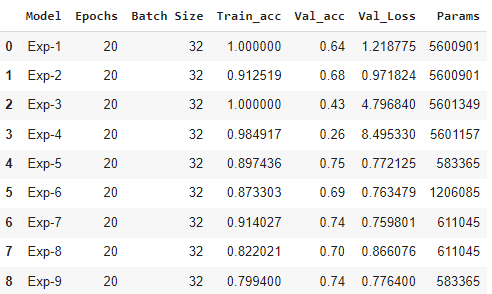


#### 9. EXPERIMENT-9

* We have introduced augmentation in this experiment.
* Model Architecture

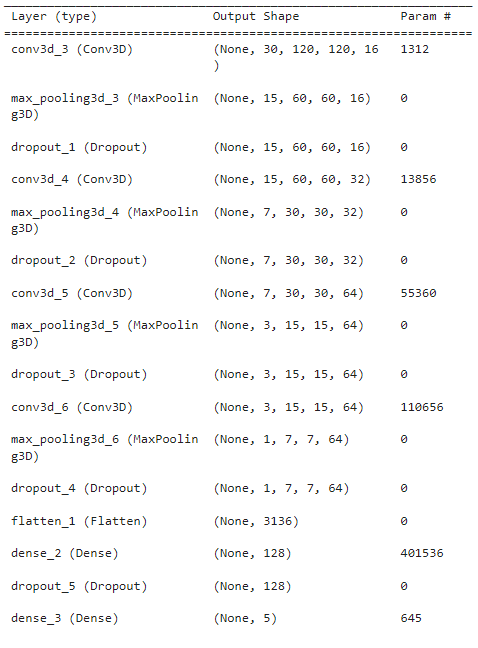


* Result🡪 This one seems to have handled the overfitting issues since validation and the training accuracies are pretty close and also above 70%.

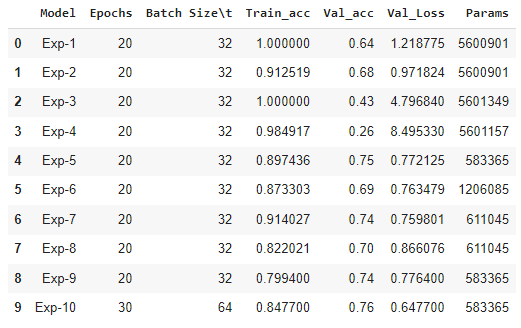


#### 10. EXPERIMENT-10

* We have experimented with batch size=64 and added more epochs also to checkout the results.
* Model Architecture

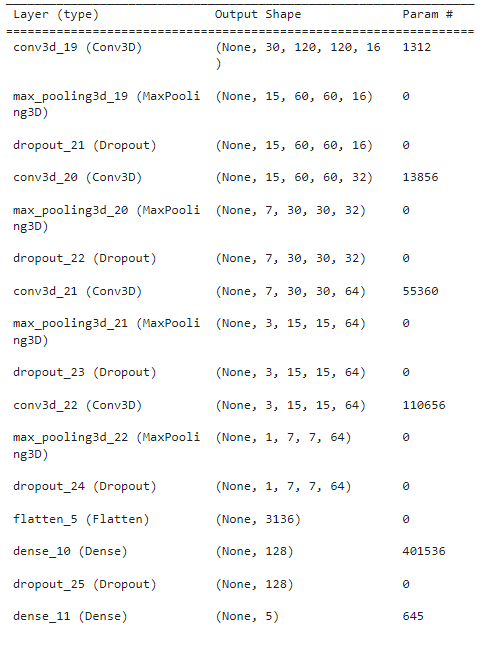


* Result🡪Model has improved a little bit and it is a good model with no overfitting.

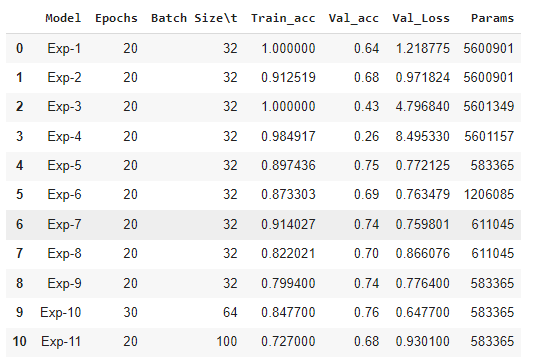


#### 11. EXPERIMENT-11

* In the experiment we have increased the batch size to 100 and tested on 20 epochs
* Model Architecture

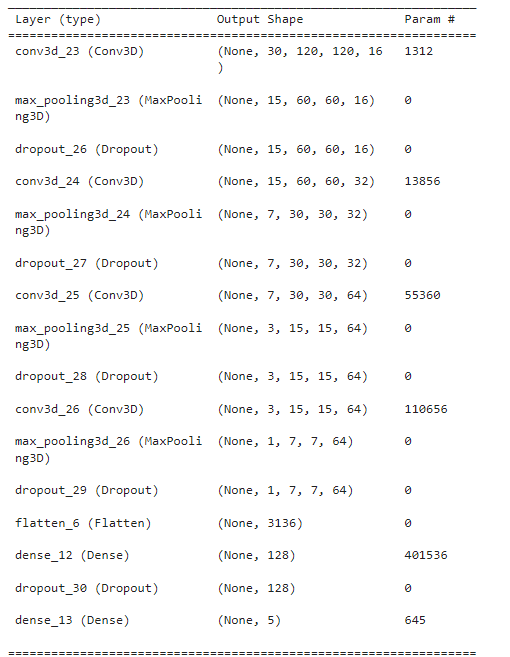


* Result🡪 Model is not overfitting however the training accuracy is just not too high.

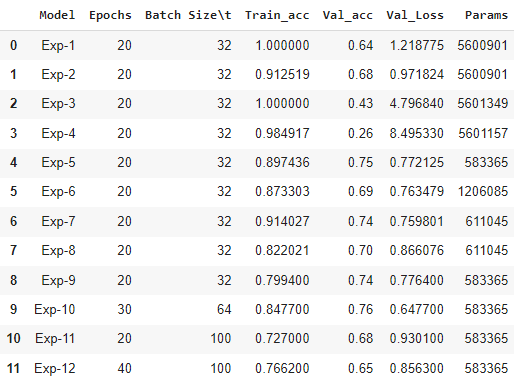


#### 12. EXPERIMENT-12

* In this experiment we have tried out with batch size of 100 and number of epochs 40.
* Model Architecture

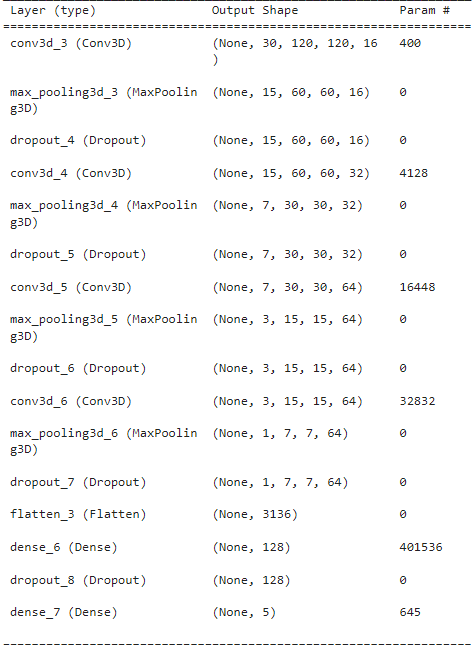


* Result🡪 Model has not improved whatsoever by increasing the batch size and the number of epochs.

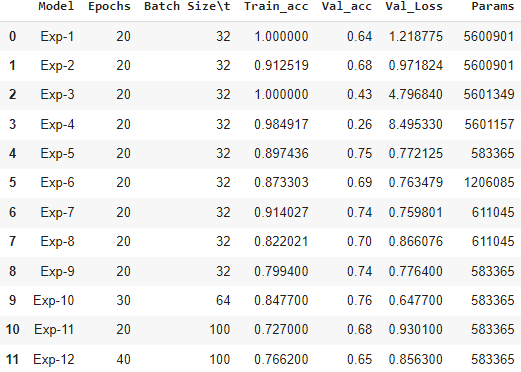


#### 13. EXERIMENT-17

* In this experiment we have changed the kernel size to 2 and performed an experiment just to see what different happens.
* The model was tweaked over the model that we have built in Exp-9.
* Model Architecture



* Result🡪One significant thing to note is that the model parameters got reduced significantly however model got overfit still.

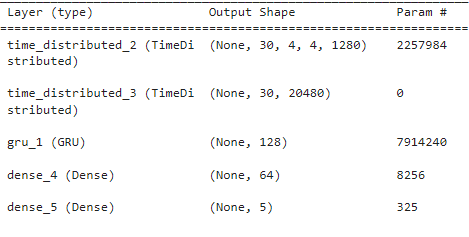




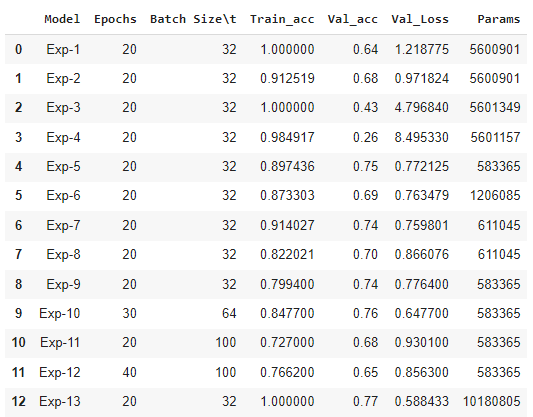
### CNN+RNN ARCHITECTURE

#### 14. EXPERIMENT-13

* Here we have introduced transfer learning and GRU units to perform the experiment.
* We have used MobileNetV2 since it is light weight and also had better accuracy than the MobileNet.
* Model Architecture

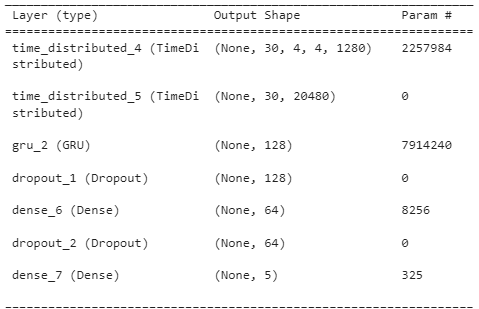


* Result🡪 The model is overfitting here. Also note that we have frozen the model layers of MobileNetV2.

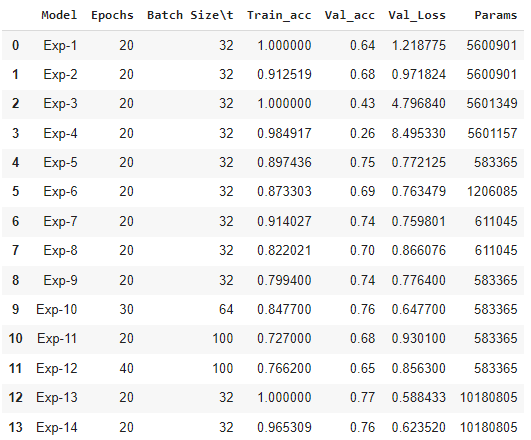


#### 15. EXPERIMENT-14

* Here we have added a Dropout layer to the previous model in hopes of getting overfitting reduced.
* Model Architecture

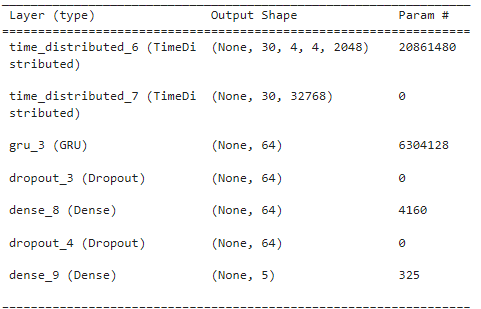


* Result🡪Model has not improved yet.



#### 16. EXPERIMENT-15

* We will try the transfer learning with respect to the Xception model now since it has the highest accuracy in the keras api.
* Also we have reduced the GRU units to 64 in hope of reducing overfitting.
* Model Architecture

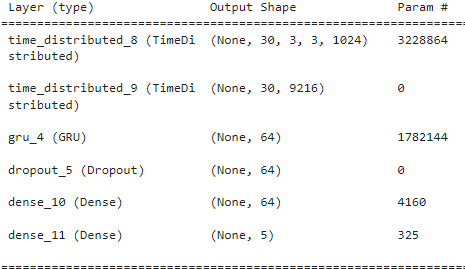


* Result🡪The model performance has got worse and not improved as expected.

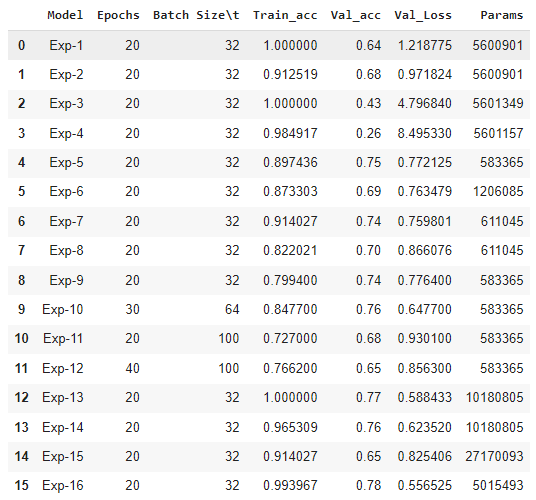


#### 17. EXPERIMENT-16

* In this experiment we are trying out transfer learning using MobileNet since it is the lightest model and having very few parameters as compared to other models.
* Model Architecture

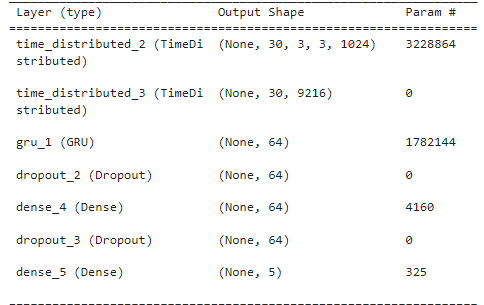


* Result🡪 Model is still overfitting however we are able to achieve the validation accuracy near 80%.



#### 18. EXPERIMENT-18

* In this experiment we have removed the frozen layers of MobileNet model in hope of tweaking all the parameters to improve the model.
* Model Architecture



* Result🡪 We were able to get to the validation accuracy mark of 85% and clearly we haven’t achieved it yet in any of the other experiments. It seems that unfreezing the layers let’s the model to tweak its weights as per the new data and it suits well for our problem statement. Due to limited computational resources we have stopped the training here and taken this as our **final model.**  We will attach the weights associated with it in the zip file. The model will be of the last epoch of Experiment-18.
* Note: A total of approximately 125 computational units have been used up in performing this experiment.
* Approximate time of training a model was around 45-50 minutes.

