# Use of Basic Functionality:

## Callbacks

The training process of the model does not incorporate termination of the training process when the model attains a human level accuracy in both the training set and the validation set. As the resulting curve of the training and validation accuracy is not exponential or the curve does not produce a standard linear function. The callback termination functional call, will disrupt training process and will not be the optimal way through which the model’s efficiency can be determined. The only callback function call used in during the model analysis is the csv callback function which records the model’s accuracy and loss function score in every epoch into a csv file. These files are then saved to get more further analysis of the benefactor of the model and what are the downsides of it.

## Convolutions

The usage of Residual networks has not only made the deep neural network a bit better but has also increased the overall path length of the neural network, in order to compute and produce one epoch the model takes a whooping 70-80 seconds. Even though the model has only 20 million parameter which is far less than the inception v4 neural network, but the inception neural network takes 35-45 seconds at each epoch for the same dataset. By path increase it technically means that there are more computations taking place at each layer than when compared to the computation on a relatively huge trainable matrix.

## Data Augmentation

The datasets used in evaluation of the TExNet model are all in .jpg files which under go data augmentation via ImageDataGenerator function of Keras library.

rescale = 1./255, each image is rescaled in 255 pixel ratio, the resolution of the image is multiplied by 1/255. The reason why we convert to 1/255 basis is because the pixel value in gray scale ranges from 0 to 255, where 0 is black and 255 is white. The same applies for RBG images where each color has 0 to 255 range for each individual color and the dimension is 3, as in (R(0-255),G(0-255),B(0-255)).

rotation\_range=40, the rotation function rotates the image at a particular angle to give the dataset some randomness in the viewpoint of the data in multiple angles.

width\_shift\_range=0.2, this function basically shifts the image by a fixed value in the horizontal direction. Does similar functionality in providing randomness by shifting the image in the horizontal direction.

height\_shift\_range=0.2, Similar to the width shift range function

shear\_range=0.2, turns the image by clockwise or anti clockwise direction by a certain angle depending on the value specified between -1 to 1.

zoom\_range=0.2, It zooms the image by the zoom percent, in this case the zoom percent mentioned is 0-20% of the overall image. At max 20% of the image is zoomed in to provide an enlarged/different view of the image.

horizontal\_flip=True, this functionality will generate the mirror reflection of the image

fill\_mode='nearest'. Here we have different options we can experiment on how the void portions of the image should be filled. There are 4 options, [‘constant’, ’nearest’, ’reflect’, ’wrap’]; the fill mode selected is nearest where the pixel closest to the void is naturally replicated. For example, ‘xxx’|xyz|’zzz’; where the quoted letters are

target size = 299,299, This basically reshapes the image resolution to our preference. It is an extremely useful functionality in the basis of applying transfer learning where the model is pretrained with a standard image pixel

batch size = 2. This helps in providing the given data set in the form of batches that can be sent to the model for training and depending on the size of the number of batches it would have a direct impact on the training of the model.

The above functionalities help in increasing the number of images by a certain amount. It helps in training the model to train certain images at various different angles and perspectives.

## Transfer Learning

Xxx

## Multiclass classification

Xxxx

# Test Analysis

The test analysis for each case is taken in such a way that the training and validation scores are taken for every epoch and the average of all is taken in to consideration. The significant part of this test is to analyse the difference between the use of 50 million parameters as in the inception v4 model to that of 20 million of the TExNet model. The main take away through these test will be to find ways to tweak the TExNet model in order to perform better than the inception model for the given dataset. The dataset used in this analysis is the Digit MNIST dataset which is the most common dataset used in comparing the performance of models and distinguishing them.

## model\_history\_log\_inception\_resnet\_v2\_Mnist\_Digits\_1

* Model used is Inception resnet or inception v4.
* Learining rate applied is e-3 or 10^-3.
* Training accuracy achieved is 89.5%
* Validation accuracy achieved is 81%.
* Analysis: the model stabilizes between 85-90% after 200 epochs, and the data collected is for 300 epochs. The time taken for each epoch is approx. 45-50 seconds. Even though the model consists of more than 50 million parameters the forward and bachward propagation step is cut short for the number of compution steps involved in this. Making it important to rework the Texnet model and reduce the computation step to a significant amount.

## model\_history\_log\_texnet\_test\_2

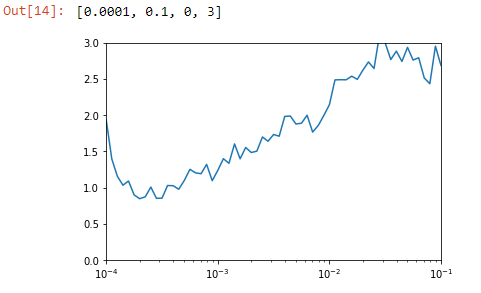
* Model used is TExNet.
* Learining rate applied is e-3 or 10^-3.
* Training accuracy achieved is 57%
* Validation accuracy achieved is 47%.
* Analysis: the model stabilizes between 75-80% after 200 epochs, and the data collected is for 300 epochs. The time taken for each epoch is approx. 70-80 seconds. Even though the model consists of more than 20 million parameters the forward and bachward propagation step takes a lot of time to execute as the number of computation and merging steps is almost twice as much as inception v4.The learning portion of this step is that the learning accuracy is the only portion that gets stabilized and the validation portion goes haywire in terms of having a constant increase. The problem arised in the dataset split where only 5% of the dataset is used for validation and 95% is used for training.

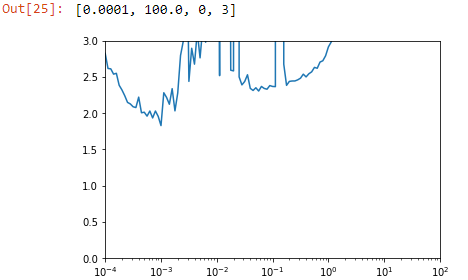
## model\_history\_log\_texnet\_test\_3

* Model used is TExNet.
* Learining rate applied is e-3 or 10^-3.
* Training accuracy achieved is 55%
* Validation accuracy achieved is 35%.
* Analysis: the model stabilizes between 75-80% after 250 epochs, and the data collected is for 300 epochs. The time taken for each epoch is approx. 70-80 seconds.This data is collected after solving the dataset split issue and turns out that the overall diiference between the training accuracy average and the validation accuracy average is significantly higher which is about 20% when compared to the 10%. As this can be explained through the use of the 5% of the data for validation which internally resulted in the accuracy for few of the epochs to go to human level accuracy of 99%. This rise in the accuracy eventually lead to the increase in the average accuracy for the validation. Rest assured that no such human level accuracy was achieved during this test as the max percentage reached by the model during the training was 79% in the validation accuracy. Now this brings the situation as to where the model can be tweaked, for both these test there is a window where the validation and the training scores is increasing pretty slowly when compared to the previous test for the inception v4. By this the conclusion achieved and the next test to be analysed will be check the most optimal learning rate that is to be applied on the model to get a fairly sophisticated and natural learning and validation curve for the given MNIST dataset.

## model\_history\_log\_texnet\_test\_4

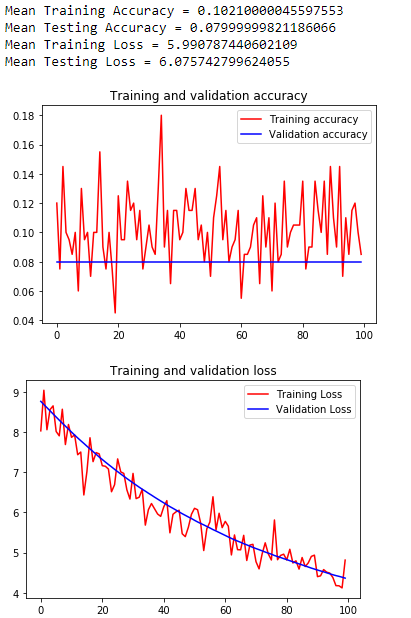
* Model used is TExNet.
* Learning rate applied is 7e-4 or 70^-4.
* Training accuracy achieved is 55%
* Validation accuracy achieved is 35%.
* Analysis: After readjusting the learning rate to a even smaller amount than the previous, the resulting learning curve is such that when comparison with the previous test, the validation loss score and the training loss score are very close to each other. The average score in the training and validation score is respectively.





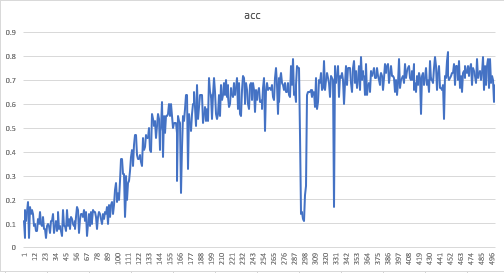
The top graph is that of the learning rate analysis of the Inception v4 model and the bottom is that of the TExNet. Both models seeming performs better with the learning rate being in-between e-4 and e-3. When tests were performed on the TExNet model with 7e-4 learning rate, the model performed slowly in terms of loss reduction. The next test to be performed will be having to tweak with the steps per epoch parameter and to find an optimal parameter to benefit the model in general case and as well as to perform better than the inception v4.

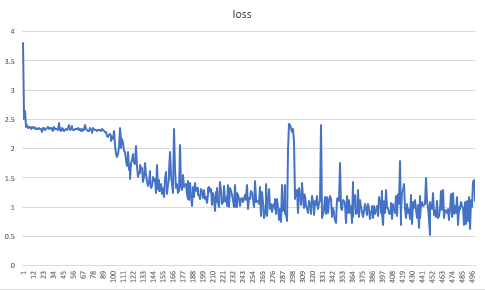
Topping it off with the main highlight is that the difference between the mean training loss and the mean testing loss, the model has achieved a significant result in this test by having both the curves extremely close to each other. This is possible only by finding the optimal learning rate. The optimal learning rate is directly dependent on the dataset than the actual model itself. So the learning rate will differ for any other given dataset and it is highly considered to test and find the optimal learning rate by using the learning rate callback.



## model\_history\_log\_texnet\_test\_5

* Model used is TExNet.
* Learining rate applied is 9e-4 or 9^-3.
* Training accuracy achieved is 53%
* Validation accuracy achieved is 47%.
* Analysis: Learning rate is readjusted and the results have been a slight improvement in general. The steps per epoch parameter was reduced to half and that resulted in reducing the time taken to finish one epoch by half. Reducing the steps per epoch resulted in a very jumpy curve and was looking less likely than the curve generated in the inception v4 model. This time the number of epochs for the training process was set to 500 epochs, this was done prior to the discovery of the reduced time taken to finish one epoch. The next test will be having to test out any form of layer tweaking or reduction of the overall layers used so that the time taken for one computation/epoch is as close to the inception v4 model. Another finding in this test was that the system used to test the model is not highly capable in discoverying the true potential of the TExnet model. The model’s main system requirement to generate propper and smooth curve is the need to use at least 12GB Ram, with this in mind necessary changes will be made to the model where the layers that require high demand of RAM will be tweaked/reduced so that the model can generate the training and validation sets with a batch size of 32. Batch size in general play a significant role in how good the model will perform during the training process.





## model\_history\_log\_texnet\_test\_6

Test resulted in failure as the model training crashed at epoch 56 and the main cause being the memory allocation in the RAM of the system. Overall, the whole system crashed in the process even though the model was optimized to support the current system specification.

## model\_history\_log\_texnet\_test\_7

* Model used is TExNet.
* Learining rate applied is 10^-3.
* Training accuracy achieved is 88.4%
* Validation accuracy achieved is 75.6%.
* Analysis: Learning rate is readjusted and also the batch size for both training and validation has been increased to 8 due to system limitations, and the results have sky rocketed. The training has been done for 500 epochs where it elapsed for almost 6 hours in total training time where the steps per epoch for training is set to 50 and the steps per epoch for validation has been set to 25. The average time taken for one epoch was estimated to be around 41 seconds which is again a little more than the inception v4 model. But this is due to the fact that the TExNet model has been updated in terms of the number of trainable parameters set, estimating to be around 1.5 million. When compared with the inception v4 model this model consists of lesser trainable parameters but consists of more computation steps in a single forward propagation step. Now coming to the difference in the training and validation steps, where the difference is around 13% in average which signifies that the model does not have issues in regards with over fitting and under fitting. This issue may not rise due to the fact of the number of training samples being used for training and validation per epoch. The MNIST dataset being used consists of about 32,000 training set images and 8,000 validation set images and this is set to the image generator function which in tern produces approx. 30+ images which get generated at random through the means of random rotation and other geometrical applications implied to it. So, judging by the number of samples the model is using and for the training time of about 6 hours for 500 epochs the model is able to get average validation score of 88% in the training set. As this will help in training more complex images and give us more satisfying and less time-consuming results when applied to other real life applications.

At epoch 437, the model reached its peak in both training and validation set. Where the training accuracy reached to almost 96.5% and the validation accuracy reached 95.5%. The next test will be adjusting the steps per epochs with the standard equation used by the deep learning community, this is mainly to reduce the jumpy effects seen in the validation accuracy curve

## model\_history\_log\_texnet\_test\_7

* Model used is TExNet.
* Learining rate applied is 10^-3.
* Training accuracy achieved is 87.6%
* Validation accuracy achieved is 71%.

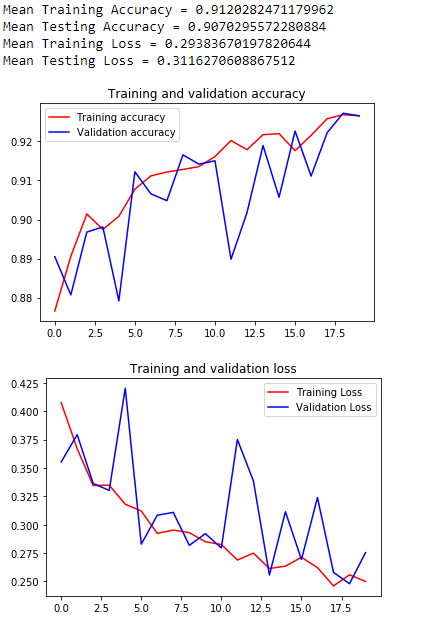
Analysis: The batch size for this test is reduced to 6 to help counter the memory issue when the inclusion of a saved model callback in initiated. With the help of this call back, the best epoch is noted and saved. Then the model weights are saved with regards to the best epoch so when applied to a real-life application the model will be able to perform efficiently better than some random epoch value or average weight of it. Since the model does not face issues with over fitting and under fitting, this approach seems to be the most optimal. The best epoch found is to be performing 93% accuracy on the training set and 95% on the validation set, where the difference is 2% which would be acceptable as the number of training set is almost more than 35-40 million images when compared to the use of 1-2 million validation images. The jumpiness of the curve of both the validation loss and accuracy can be reduced by increasing the batch size but due to system limitation this is not possible for the moment. The next test will be having to load and reuse the model and test it out on a real life application and check its performance.

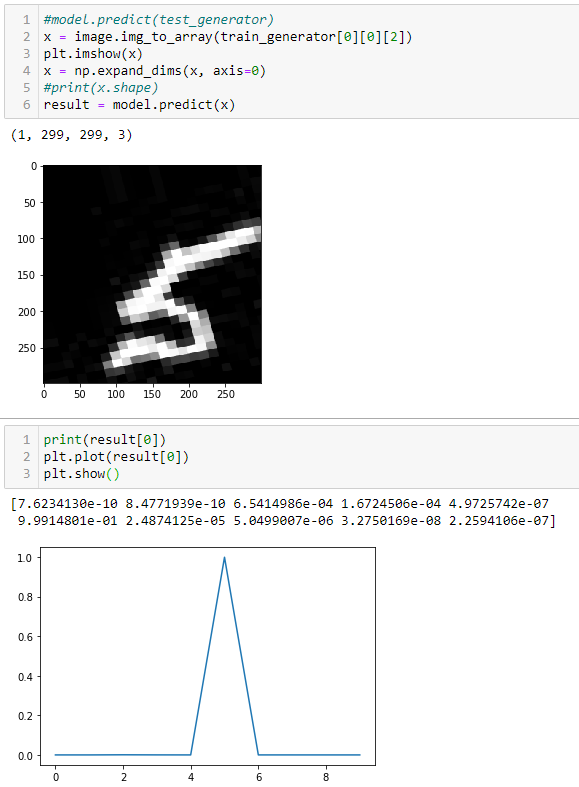
## model\_history\_log\_texnet\_test\_8

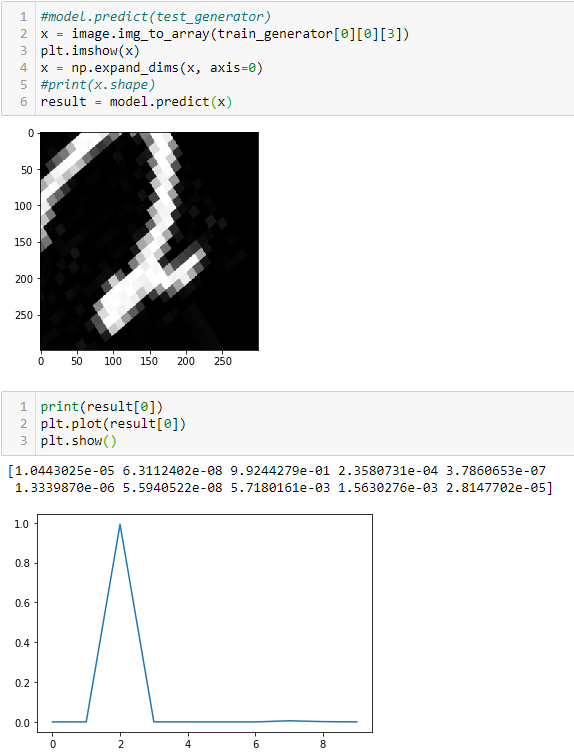
* Model used is TExNet.
* Learining rate applied is 10^-3.
* Training accuracy achieved is 91.2%
* Validation accuracy achieved is 90.7%.

Analysis: The batch size is set to the same value as the previous test and the only changed made was in the steps\_per\_epoch parameter where the equation used is,

The results produced an even better line curve and a constant learning rate, but at the cost of the time taken for each epoch. The time for each epoch elapsed to around 30 mins and the total training time elapsed to around 10 hours for 20 epochs. But the model now predicts the handwritten digits with an accuracy of about 91% on average which can be termed to human level perform if were to take in to consideration of the randomness of how the image is sent across the model to train.







# Reference Links

<https://stackoverflow.com/questions/49922252/choosing-number-of-steps-per-epoch>

<https://stats.stackexchange.com/questions/164876/tradeoff-batch-size-vs-number-of-iterations-to-train-a-neural-network>

<https://www.kaggle.com/cdeotte/25-million-images-0-99757-mnist>