**Technical Report of MNIST Dataset using Neural Network**

**Author Name -> Ajit Chavhan**

**Contact No. -> 8652027955**

**Email ->** [**ajit.chavhan@gmail.com**](mailto:ajit.chavhan@gmail.com)

**Algorithm:-** Tensorflow/keras

**Preparation of data**

We have already prepared dataset from keras in below format

List of 2

$ train:List of 2

..$ x: int [1:60000, 1:28, 1:28] 0 0 0 0 0 0 0 0 0 0 ...

..$ y: int [1:60000(1d)] 5 0 4 1 9 2 1 3 1 4 ...

$ test :List of 2

..$ x: int [1:10000, 1:28, 1:28] 0 0 0 0 0 0 0 0 0 0 ...

..$ y: int [1:10000(1d)] 7 2 1 0 4 1 4 9 5 9 ...

I have made the four variable from this dataset

trainx <- mnist$train$x

trainy <- mnist$train$y

testx <- mnist$test$x

testy <- mnist$test$y

### it is showing how many images we have in train y matrix

# 0 1 2 3 4 5 6 7 8 9

# 0 5923 0 0 0 0 0 0 0 0 0

# 1 0 6742 0 0 0 0 0 0 0 0

# 2 0 0 5958 0 0 0 0 0 0 0

# 3 0 0 0 6131 0 0 0 0 0 0

# 4 0 0 0 0 5842 0 0 0 0 0

# 5 0 0 0 0 0 5421 0 0 0 0

Then I changed the matrix dimension (reshaping) the matrix and converting it to on dimensions rather than to 28 by 28 from trainx and test x both

##num [1:60000, 1:784] 0 0 0 0 0 0 0 0 0 0 ...

Now I have converted number from 0 to 254 to 0 to 1 by below command

trainx <- trainx/255

testx <- testx/255



Then I have applied one hot encoding using below command

trainy <- to\_categorical(trainy,10)

# [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]

# [1,] 0 0 0 0 0 0 0 1 0 0

# [2,] 0 0 1 0 0 0 0 0 0 0

# [3,] 0 1 0 0 0 0 0 0 0 0

# [4,] 1 0 0 0 0 0 0 0 0 0

# [5,] 0 0 0 0 1 0 0 0 0 0

# [6,] 0 1 0 0 0 0 0 0 0 0

**Model Making**

model %>%

layer\_dense(units = 128, activation = 'relu' , input\_shape = c(784)) %>% ## it is indicating we have one hidden layer with

## 128 neurons and 784 input neurons and activation function is relu

layer\_dense(units = 10, activation = 'softmax')

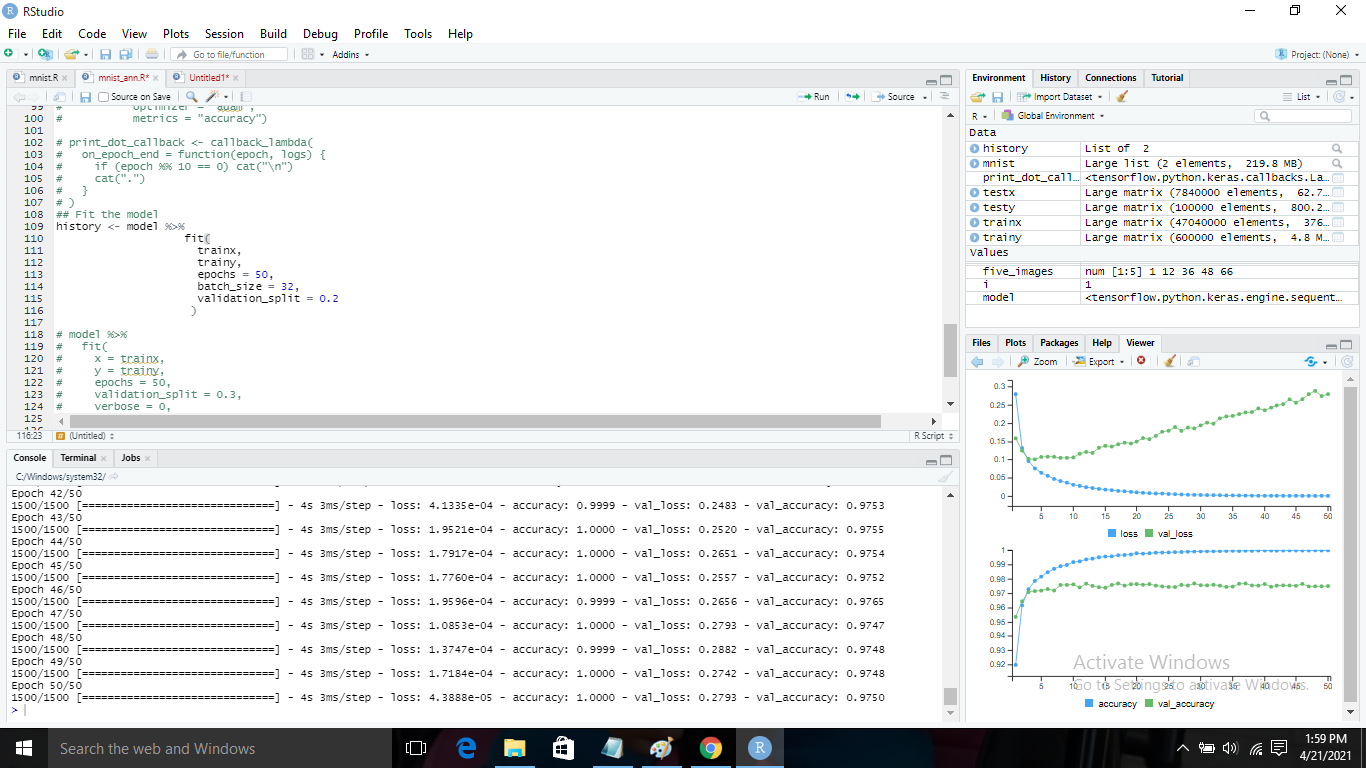
**Hidden layer -> 1**

**Neurons -> 128**

**Activation function -> relu/softmax**

##8s 4ms/step - loss: 0.4434 - accuracy: 0.8748 - val\_loss: 0.1630 - val\_accuracy: 0.9526

##4s 3ms/step - loss: 0.1327 - accuracy: 0.9607 - val\_loss: 0.1296 - val\_accuracy: 0.9633



As you can see the graph of loss and validation loss and accuracy and validation accuracy having very big gap which indicates the model is not good and facing the issue of over fitting which require parameter tuning

**Parameter Tuning**

model %>%

layer\_dense(units = 128, activation = 'relu' , input\_shape = c(784)) %>%

layer\_dropout(rate = 0.3) %>%

layer\_dense(units = 64, activation = 'relu') %>%

layer\_dropout(rate = 0.2) %>%

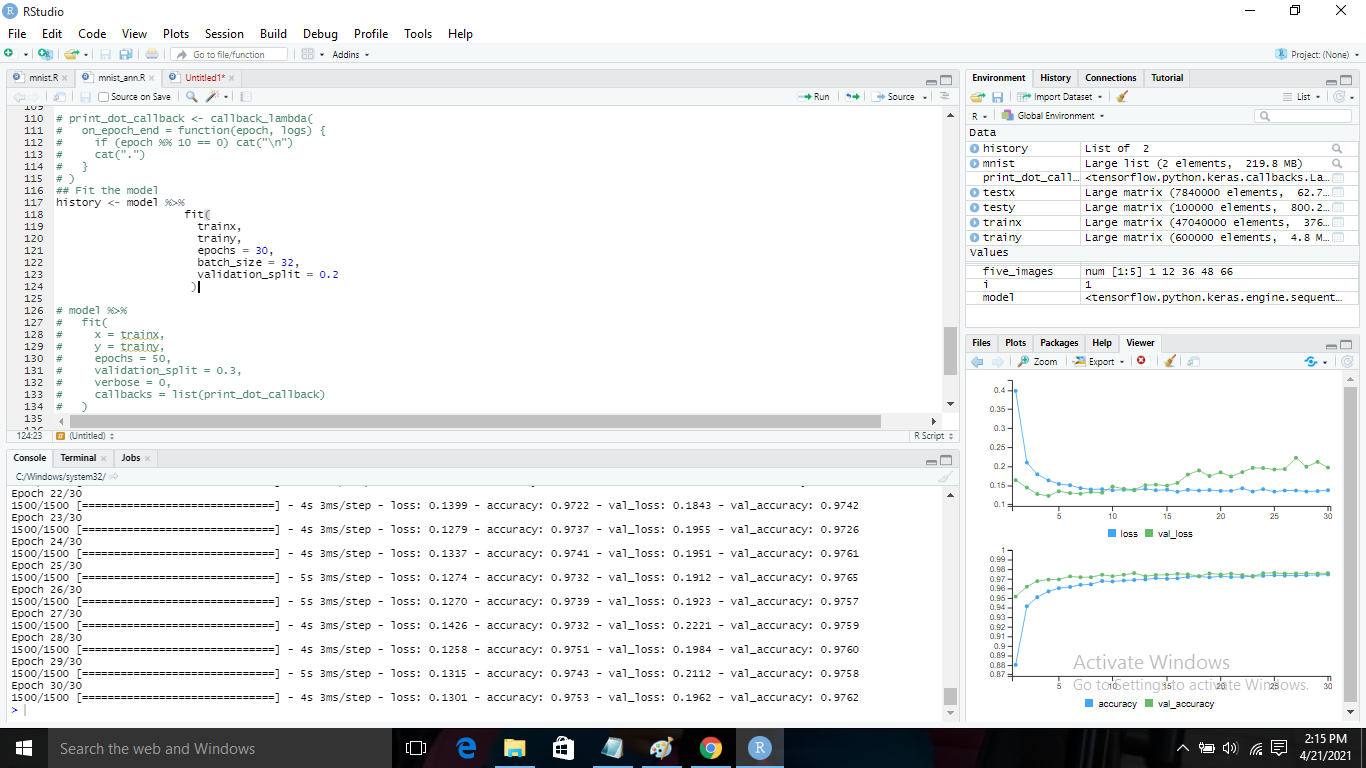
layer\_dense(units = 10, activation = 'softmax')

**Hidden layer -> 2**

**Neurons ->** 128 in first layer with dropout of .3 percent rate to handle the over fitting problem

And on second hidden layer there are 64 neurons with dropout rate of 2 percent

**Activation function ->** relu/softmax



In above picture you can see the gap between loss vs validation loss and accuracy vs validation accuracy reduced significantly so this model is far better than above one.

**Prediction and Accuracy**

In this model I have got the loss and accuracy as below

# loss accuracy

**# 0.1795492 0.9768000**

Below is the confusion matrix of testy variable

# pred 0 1 2 3 4 5 6 7 8 9

# 0 971 0 2 0 2 2 6 3 5 2

# 1 0 1125 2 0 1 0 3 3 1 2

# 2 2 3 1011 6 3 0 0 9 3 0

# 3 1 1 2 982 0 6 1 6 7 4

# 4 0 0 1 0 949 1 5 1 3 7

# 5 2 1 0 9 0 874 3 0 8 4