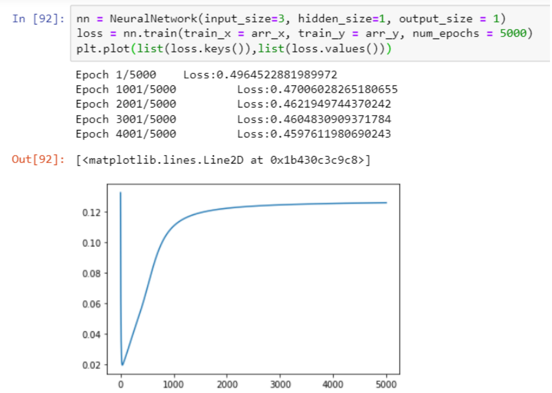
Assignment- 0

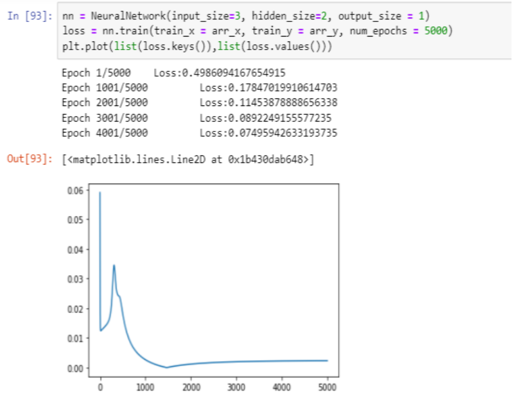
Shwetha S, ECE C

Inference from Task-1

- Changing the hidden_size parameter changes the number of neurons in the hidden layer.
- Initially as we increase the parameter from 1 the loss value reduces till the parameter's value is 5. After that value loss increases.
- Hidden _Size parameter needs to be tuned to an optimum value.
- One important thing to note is that each time the value of hidden_size is changed and the model is run, the values of the initialization matrix changes as it is just a random matrix. Hence, we must take this into account while examining the effect of hidden_size.
- The optimum value for hidden_size is either 4 or 5.



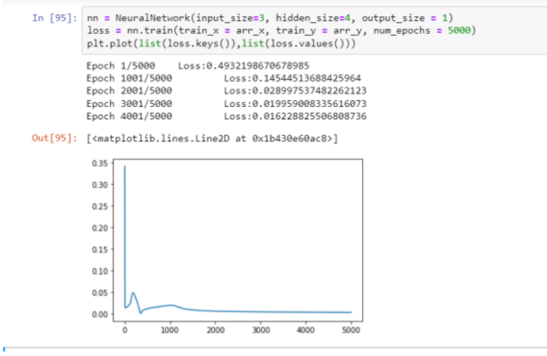
```
In [92]: nn = NeuralNetwork(input_size=3, hidden_size=1, output_size = 1)
         loss = nn.train(train_x = arr_x, train_y = arr_y, num_epochs = 5000)
         plt.plot(list(loss.keys()),list(loss.values()))

         Epoch 1/5000    Loss:0.4964522881989972
         Epoch 1001/5000         Loss:0.47006028265180655
         Epoch 2001/5000         Loss:0.4621949744370242
         Epoch 3001/5000         Loss:0.4604830909371784
         Epoch 4001/5000         Loss:0.4597611980690243

Out[92]: [<matplotlib.lines.Line2D at 0x1b430c3c9c8>]
```



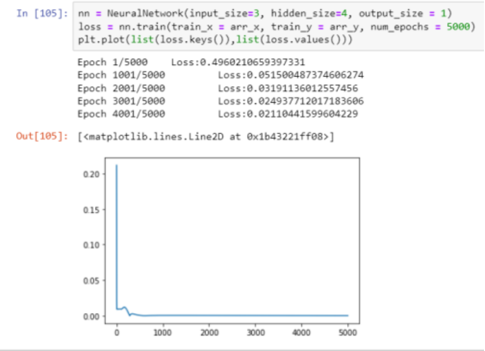
```
In [93]: nn = NeuralNetwork(input_size=3, hidden_size=2, output_size = 1)
         loss = nn.train(train_x = arr_x, train_y = arr_y, num_epochs = 5000)
         plt.plot(list(loss.keys()),list(loss.values()))

         Epoch 1/5000    Loss:0.4986094167654915
         Epoch 1001/5000         Loss:0.17847019910614703
         Epoch 2001/5000         Loss:0.11453878888656338
         Epoch 3001/5000         Loss:0.0892249155577235
         Epoch 4001/5000         Loss:0.07495942633193735

Out[93]: [<matplotlib.lines.Line2D at 0x1b430dab648>]
```

Fig 1: Hidden_size=1

Fig 2:   Hidden_size=2



```
In [95]: nn = NeuralNetwork(input_size=3, hidden_size=4, output_size = 1)
         loss = nn.train(train_x = arr_x, train_y = arr_y, num_epochs = 5000)
         plt.plot(list(loss.keys()),list(loss.values()))

         Epoch 1/5000    Loss:0.4932198670678985
         Epoch 1001/5000         Loss:0.14544513688425964
         Epoch 2001/5000         Loss:0.028997537482262123
         Epoch 3001/5000         Loss:0.019959008335616073
         Epoch 4001/5000         Loss:0.016228825506808736

Out[95]: [<matplotlib.lines.Line2D at 0x1b430e60ac8>]
```



```
In [105]: nn = NeuralNetwork(input_size=3, hidden_size=4, output_size = 1)
          loss = nn.train(train_x = arr_x, train_y = arr_y, num_epochs = 5000)
          plt.plot(list(loss.keys()),list(loss.values()))

          Epoch 1/5000    Loss:0.4960210659397331
          Epoch 1001/5000         Loss:0.051500487374606274
          Epoch 2001/5000         Loss:0.03191136012557456
          Epoch 3001/5000         Loss:0.024937712017183606
          Epoch 4001/5000         Loss:0.02110441599604229

Out[105]: [<matplotlib.lines.Line2D at 0x1b43221ff08>]
```

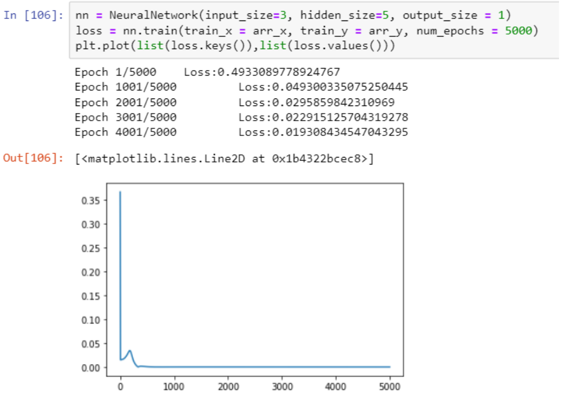
Fig 3: Hidden_size=4

Loss= 0.0162

Fig 4: Hidden_size=4

Loss=0.0211

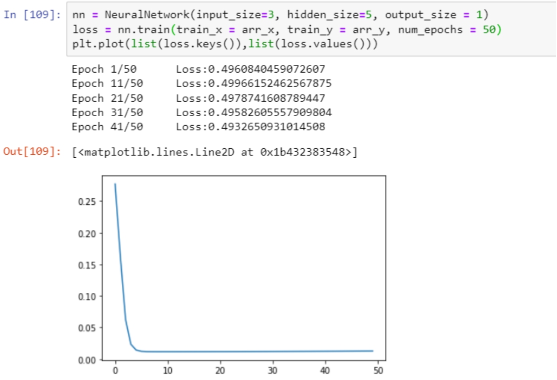
Fig 5: Hidden_Size=5 , Loss= 0.019



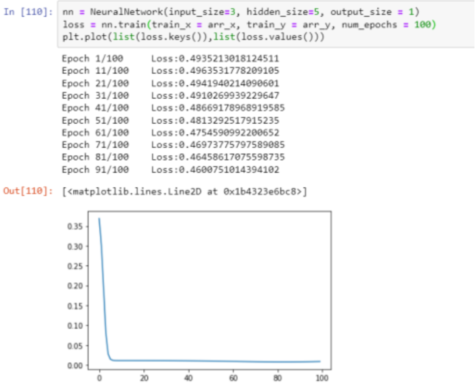
```
In [106]: nn = NeuralNetwork(input_size=3, hidden_size=5, output_size = 1)
          loss = nn.train(train_x = arr_x, train_y = arr_y, num_epochs = 5000)
          plt.plot(list(loss.keys()),list(loss.values()))

          Epoch 1/5000    Loss:0.4933089778924767
          Epoch 1001/5000         Loss:0.049300335075250445
          Epoch 2001/5000         Loss:0.0295859842310969
          Epoch 3001/5000         Loss:0.022915125704319278
          Epoch 4001/5000         Loss:0.019308434547043295

Out[106]: [<matplotlib.lines.Line2D at 0x1b4322bcec8>]
```

Here the lower value of loss can be for hidden_size=4 or hidden_size=5 depending on the result chosen.

- The number of iterations or the value of epochs determine the degree of reduction in loss. Greater the number of epochs, lower the loss. However, the loss saturates after a while and so we can stop after a certain number of iterations



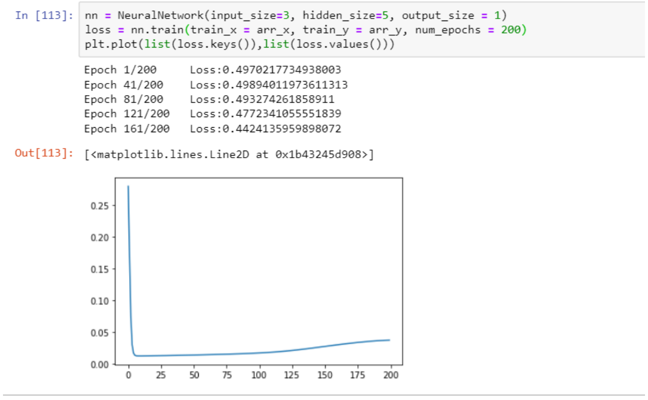
```
In [109]: nn = NeuralNetwork(input_size=3, hidden_size=5, output_size = 1)
          loss = nn.train(train_x = arr_x, train_y = arr_y, num_epochs = 50)
          plt.plot(list(loss.keys()),list(loss.values()))

          Epoch 1/50      Loss:0.4960840459072607
          Epoch 11/50     Loss:0.49966152462567875
          Epoch 21/50     Loss:0.4978741608789447
          Epoch 31/50     Loss:0.49582605557909804
          Epoch 41/50     Loss:0.4932650931014508

Out[109]: [<matplotlib.lines.Line2D at 0x1b432383548>]
```



```
In [110]: nn = NeuralNetwork(input_size=3, hidden_size=5, output_size = 1)
          loss = nn.train(train_x = arr_x, train_y = arr_y, num_epochs = 100)
          plt.plot(list(loss.keys()),list(loss.values()))

          Epoch 1/100     Loss:0.4935213018124511
          Epoch 11/100    Loss:0.4963531778209105
          Epoch 21/100    Loss:0.4941940214090601
          Epoch 31/100    Loss:0.4910269939229647
          Epoch 41/100    Loss:0.48669178968919585
          Epoch 51/100    Loss:0.4813292517915235
          Epoch 61/100    Loss:0.4754590992200652
          Epoch 71/100    Loss:0.46973775797589085
          Epoch 81/100    Loss:0.46458617075598735
          Epoch 91/100    Loss:0.4600751014394102

Out[110]: [<matplotlib.lines.Line2D at 0x1b4323e6bc8>]
```

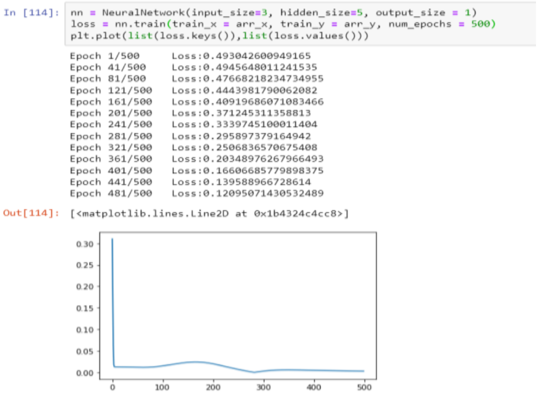
Fig 6: Epochs=50

Fig 7: Epochs=100



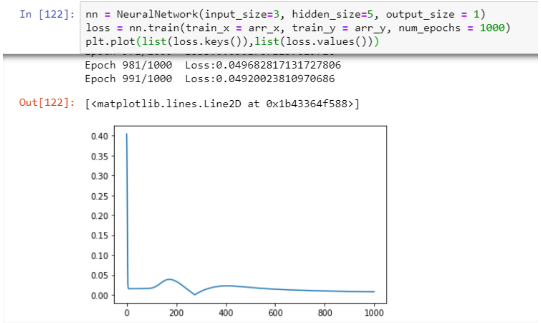
```
In [113]: nn = NeuralNetwork(input_size=3, hidden_size=5, output_size = 1)
          loss = nn.train(train_x = arr_x, train_y = arr_y, num_epochs = 200)
          plt.plot(list(loss.keys()),list(loss.values()))

          Epoch 1/200     Loss:0.4970217734938003
          Epoch 41/200    Loss:0.4989401197361131З
          Epoch 81/200    Loss:0.493274261858911
          Epoch 121/200   Loss:0.4772341055551839
          Epoch 161/200   Loss:0.4424135959898072

Out[113]: [<matplotlib.lines.Line2D at 0x1b43245d908>]
```

Fig 8: Epochs=200



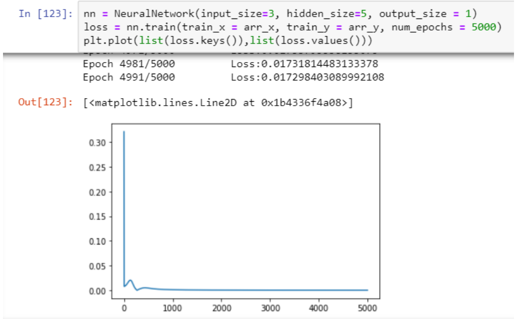
```
In [114]: nn = NeuralNetwork(input_size=3, hidden_size=5, output_size = 1)
          loss = nn.train(train_x = arr_x, train_y = arr_y, num_epochs = 500)
          plt.plot(list(loss.keys()),list(loss.values()))

          Epoch 1/500     Loss:0.493042600949165
          Epoch 41/500    Loss:0.4945648011241535
          Epoch 81/500    Loss:0.47668218234734955
          Epoch 121/500   Loss:0.4443981790062082
          Epoch 161/500   Loss:0.40919686071083466
          Epoch 201/500   Loss:0.371245311358813
          Epoch 241/500   Loss:0.3339745100011404
          Epoch 281/500   Loss:0.295897379164942
          Epoch 321/500   Loss:0.2506836570675408
          Epoch 361/500   Loss:0.20348976267966493
          Epoch 401/500   Loss:0.16606685779898375
          Epoch 441/500   Loss:0.139588966728614
          Epoch 481/500   Loss:0.12095071430532489

Out[114]: [<matplotlib.lines.Line2D at 0x1b4324c4cc8>]
```

Fig 9: Epochs=500



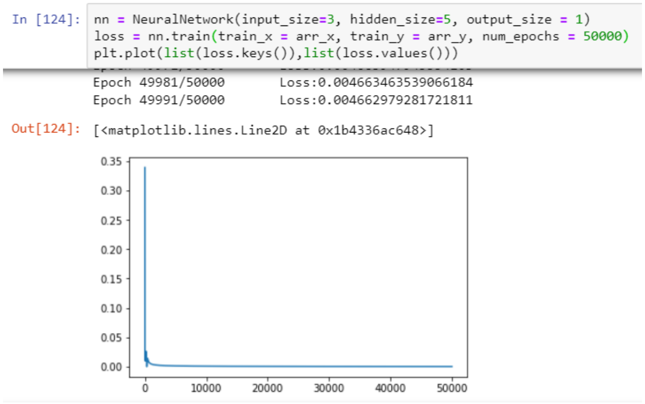
```
In [122]: nn = NeuralNetwork(input_size=3, hidden_size=5, output_size = 1)
          loss = nn.train(train_x = arr_x, train_y = arr_y, num_epochs = 1000)
          plt.plot(list(loss.keys()),list(loss.values()))

          Epoch 981/1000  Loss:0.049682817131727806
          Epoch 991/1000  Loss:0.04920023810970686

Out[122]: [<matplotlib.lines.Line2D at 0x1b43364f588>]
```

Fig 9: Epochs=1000



```
In [123]: nn = NeuralNetwork(input_size=3, hidden_size=5, output_size = 1)
          loss = nn.train(train_x = arr_x, train_y = arr_y, num_epochs = 5000)
          plt.plot(list(loss.keys()),list(loss.values()))

          Epoch 4981/5000         Loss:0.01731814483133378
          Epoch 4991/5000         Loss:0.017298403089992108

Out[123]: [<matplotlib.lines.Line2D at 0x1b4336f4a08>]
```

Fig 10: Epochs=5000

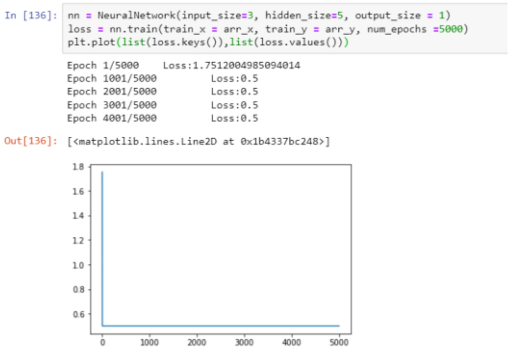


```
In [124]: nn = NeuralNetwork(input_size=3, hidden_size=5, output_size = 1)
          loss = nn.train(train_x = arr_x, train_y = arr_y, num_epochs = 50000)
          plt.plot(list(loss.keys()),list(loss.values()))

          Epoch 49981/50000       Loss:0.004663463539066184
          Epoch 49991/50000       Loss:0.004662979281721811

Out[124]: [<matplotlib.lines.Line2D at 0x1b4336ac648>]
```

Fig 11: Epochs=50000

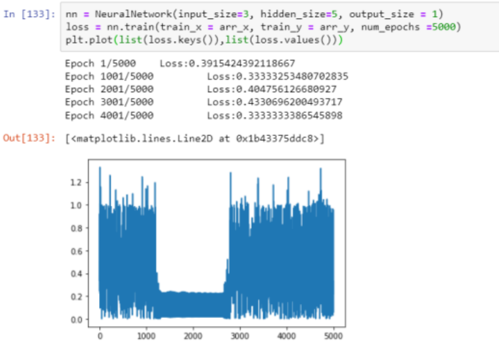
Inference from Task 2

- The activation functions for the hidden layer was changed.
- For each application a different activation function may give a better result.
- Hence we need to figure out the best activation function by trial and error.
- Sigmoid function has lower loss compared with tanh and relu for this model.



```
In [136]: nn = NeuralNetwork(input_size=3, hidden_size=5, output_size = 1)
          loss = nn.train(train_x = arr_x, train_y = arr_y, num_epochs =5000)
          plt.plot(list(loss.keys()),list(loss.values()))

          Epoch 1/5000    Loss:1.7512004985094014
          Epoch 1001/5000         Loss:0.5
          Epoch 2001/5000         Loss:0.5
          Epoch 3001/5000         Loss:0.5
          Epoch 4001/5000         Loss:0.5

Out[136]: [<matplotlib.lines.Line2D at 0x1b4337bc248>]
```

Fig 12: Relu function



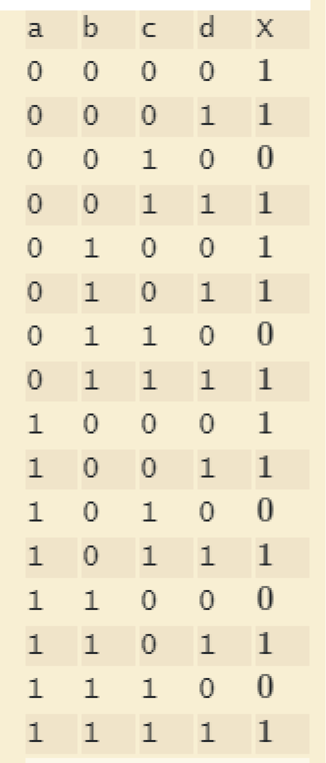
```
In [133]: nn = NeuralNetwork(input_size=3, hidden_size=5, output_size = 1)
          loss = nn.train(train_x = arr_x, train_y = arr_y, num_epochs =5000)
          plt.plot(list(loss.keys()),list(loss.values()))

          Epoch 1/5000    Loss:0.3915424392118667
          Epoch 1001/5000         Loss:0.33333253480702835
          Epoch 2001/5000         Loss:0.404756126680927
          Epoch 3001/5000         Loss:0.4330696200493717
          Epoch 4001/5000         Loss:0.3333333386545898

Out[133]: [<matplotlib.lines.Line2D at 0x1b43375ddc8>]
```

Fig 13 : Tanh function

Inference from Task 3

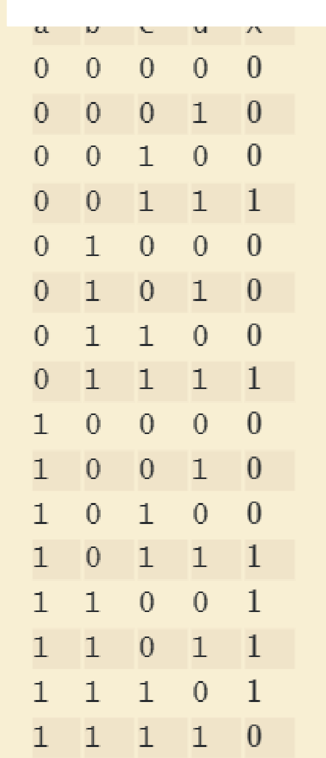
- The given task was to use the model for different data.

1. F = !((A.B)+C) + D



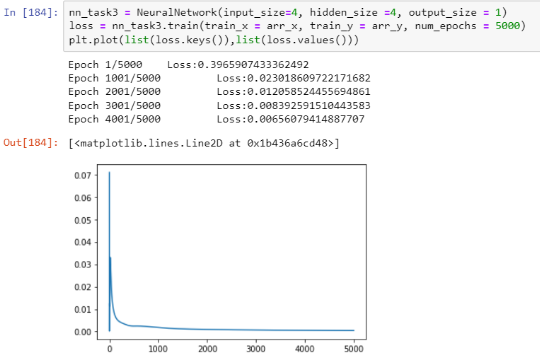
a	b	c	d	x
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

2. F = !(A.B) xor !(C.D)



a	b	c	d	x
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

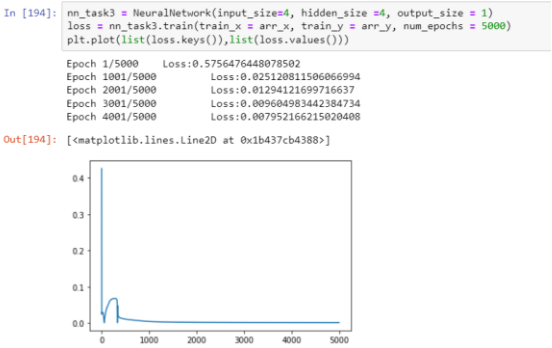
- Using sigmoid function as the activation function for hidden layer, these are the results:



```
In [184]: nn_task3 = NeuralNetwork(input_size=4, hidden_size =4, output_size = 1)
          loss = nn_task3.train(train_x = arr_x, train_y = arr_y, num_epochs = 5000)
          plt.plot(list(loss.keys()),list(loss.values()))

          Epoch 1/5000    Loss:0.3965907433362492
          Epoch 1001/5000         Loss:0.023018609722171682
          Epoch 2001/5000         Loss:0.012058524455694861
          Epoch 3001/5000         Loss:0.008392591510443583
          Epoch 4001/5000         Loss:0.00656079414887707

Out[184]: [<matplotlib.lines.Line2D at 0x1b436a6cd48>]
```

Fig 14:

The result after plugging in the training data for equation 1



```
In [194]: nn_task3 = NeuralNetwork(input_size=4, hidden_size =4, output_size = 1)
          loss = nn_task3.train(train_x = arr_x, train_y = arr_y, num_epochs = 5000)
          plt.plot(list(loss.keys()),list(loss.values()))

          Epoch 1/5000    Loss:0.5756476448078502
          Epoch 1001/5000         Loss:0.025120811506066994
          Epoch 2001/5000         Loss:0.01294121699716637
          Epoch 3001/5000         Loss:0.009604983442384734
          Epoch 4001/5000         Loss:0.007952166215020408

Out[194]: [<matplotlib.lines.Line2D at 0x1b437cb4388>]
```

Fig 15:

The result after plugging in the training data for equation 2

- Hidden size was optimum at 3 and 4 for equation one's model. Hidden size was optimum at 4 and 5 for equation two's model. Greater the number of epochs, lower the loss but again we can reduce the number of epochs by observing when the loss saturates.