CSE4/574 Fall 2022 Introduction to Machine Learning
Programming Assignment 2

Classification and Regression

TA: Meng Ding

Team
Name: Mallikharjuna Rao Annam
UBIT Name : mallikha(50465177)
Name: Sai Sridhar Reddy Palli
UBIT Name : spalli (50468339)
Name : Kajol
UBIT Name : Kajol (50478691)

1. **How to choose the hyper-parameter for a Neural Network ?**

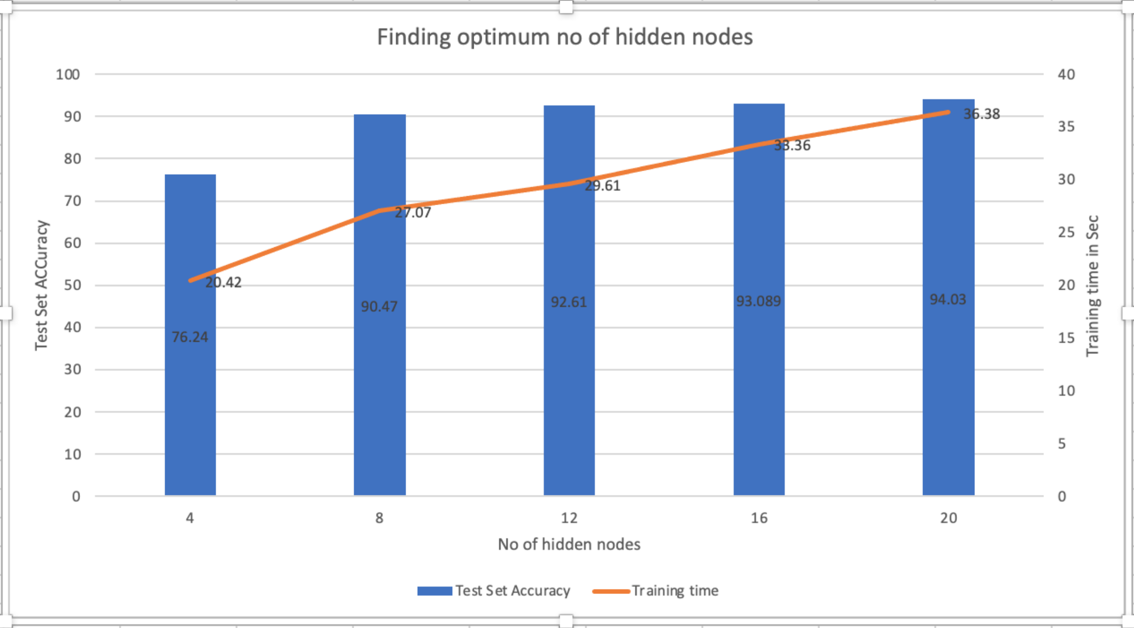
   To find the optimal hyper-parameter for a Neural Network. We have computed the results of Training score, Validation score, Test score and Training time for different values of hidden units and the same lambda value.

   Finding the Optimum Number of Hidden Units:

   Here we have calculated the values when **Lambda=0** for the different values of Hidden Units.(i.e. 4,8,12,16,20). We have also tried all the combinations of Lambda values(i.e. 0, 10, 20, 30, 40, 50, 60, 70, 80, 90) with different hidden units.

Lambda	Hidden Units	Training Score	Validation Score	Test Score	Time
0	4	77.082	75.89	76.24	20.42
0	8	92.47	91.7	90.47	27.07
0	12	94.638	93.74	92.61	29.61
0	16	95.984	94.16	93.089	33.36
0	20	95.994	94.8	94.03	36.38

Observations:
1. The table clearly shows that for a fixed value of Lambda, the Test score increases linearly along with the no of hidden units.
2. As the hidden units are increasing the time taken is also increasing along with the Training score.
3. Among the above calculated values the highest Train accuracy is obtained when the Hidden units are **20**.
4. So we can clearly say that the **optimum value of Hidden Units = 20**



The graph above depicts the time taken to train and test accuracy along with the increasing number of hidden nodes.

Finding the optimum value of Lambda:

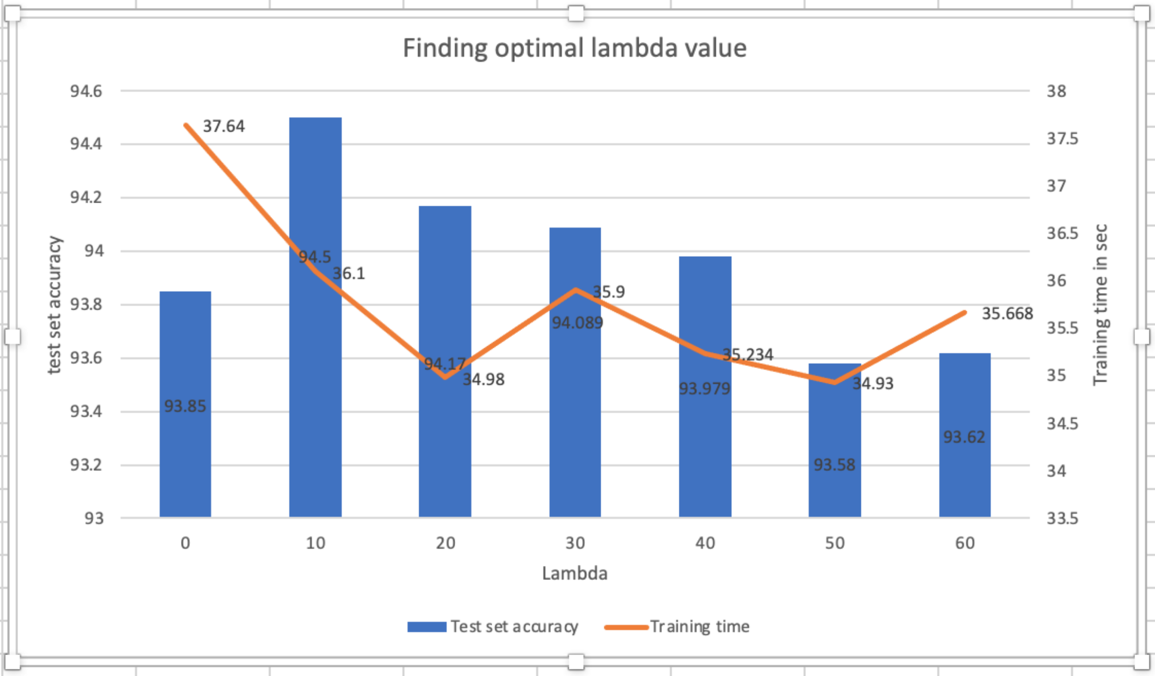
Now that we have obtained the optimum value of hidden units = 20 we will now try with different values of Lambda(i.e. 0,10,20,30,40,50,60).

Here we have calculated the values when **Hidden Unit=20** for the different values of Lambda.

Lambda	Hidden Units	Training Score	Validation Score	Test Score	Time
0	20	96.04	94.69	93.85	37.64
10	20	96.394	95.23	94.5	36.1
20	20	95.7	94.43	94.17	34.98
30	20	95.14	93.89	94.089	35.9
40	20	94.718	94.15	93.979	35.234
50	20	94.268	93.58	93.58	34.93
60	20	93.88	92.89	93.62	35.668

Observations:

1. From the above depicted graph we can clearly see that the highest Test score is obtained when the Lambda value is **10.**
2. Generally there won't be any kind of help in regularization technique with an increase in the Lambda value.
3. Among the above calculated values the highest Test score is obtained when the value of lambda = **10**.
4. So we can clearly say that the **optimum value of Lambda = 10.**



The graph above depicts the time taken to train and test accuracy for different Lambda values.

2. Accuracy of the classification method on the handwritten digit test data:

1. As shown in the above **2** tables, we can see that the optimum value of Hidden unit is **20** and for Lambda the optimum value is **10.**
2. When Lambda = 10

   Hidden Unit = 20

   The values are:

   Training Score = 96.394

   Validation Score = 95.23

   Test Score = 94.5

   Time = 36.1
3. Highest **Test Score = 94.5** is obtained when the **Lambda = 10** and **Hidden Unit = 20**

3. **Accuracy of classification method on the CelebA dataset:**

   1. As shown in the above **2** tables, we can see that the optimum value of Hidden unit is **20** and for Lambda the optimum value is **10.**
   2. We will be using the same values to classify the CelebA dataset.
   3. The observations are shown in the below table:

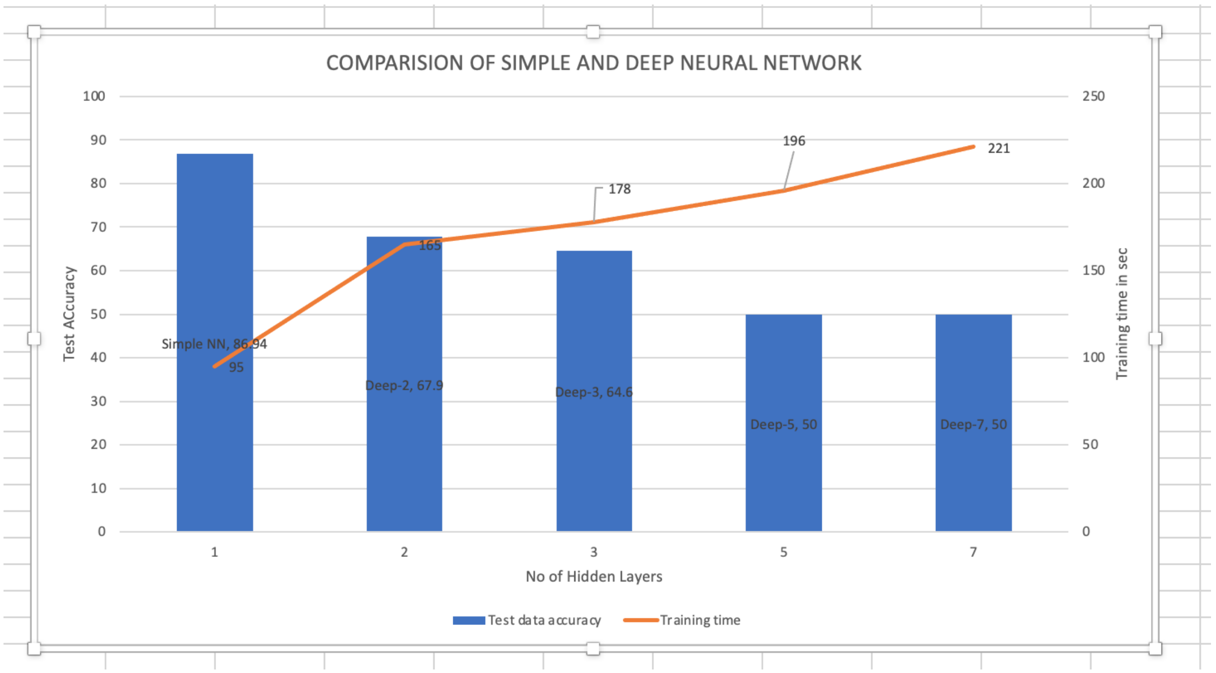
Lambda	HiddenUnits	Training Score	Validation Score	Test Score	Time
10	20	85.24	83.33	85.61	57.8

4. **Comparison of your neural network with a deep neural network using PyTorch in terms of accuracy and training time.**

Type of NN	Number of Hidden Layers	Test Score	Average Loss	Time taken to Train in sec
Simple Neural Network	1	86.94	0.600746	95
Deep Neural Network	2	67.9	0.674762	165
Deep Neural Network	3	64.6	0.688873	178
Deep Neural Network	5	50.0	0.693098	196
Deep Neural Network	7	50.0	0.6913164	221

Observations:

1. The above table clearly shows that the accuracy falls as the count of hidden layers increases because of over-fitting.
2. According to the above mentioned observations, Simple Neural Network with single hidden layer for optimum hyper-parameter values has an accuracy of 86.94%
3. In addition, the Simple Neural Network takes less time than the Deep Neural Network. Time increases as the number of layers increases due to greater complexity and calculation.



Above graph is plotted for Time taken to train and Test Score for different values of hidden units. From the above graph,we can conclude that the highest Test Score is obtained when the hidden layers are set to **1** which is Simple Neural Network.

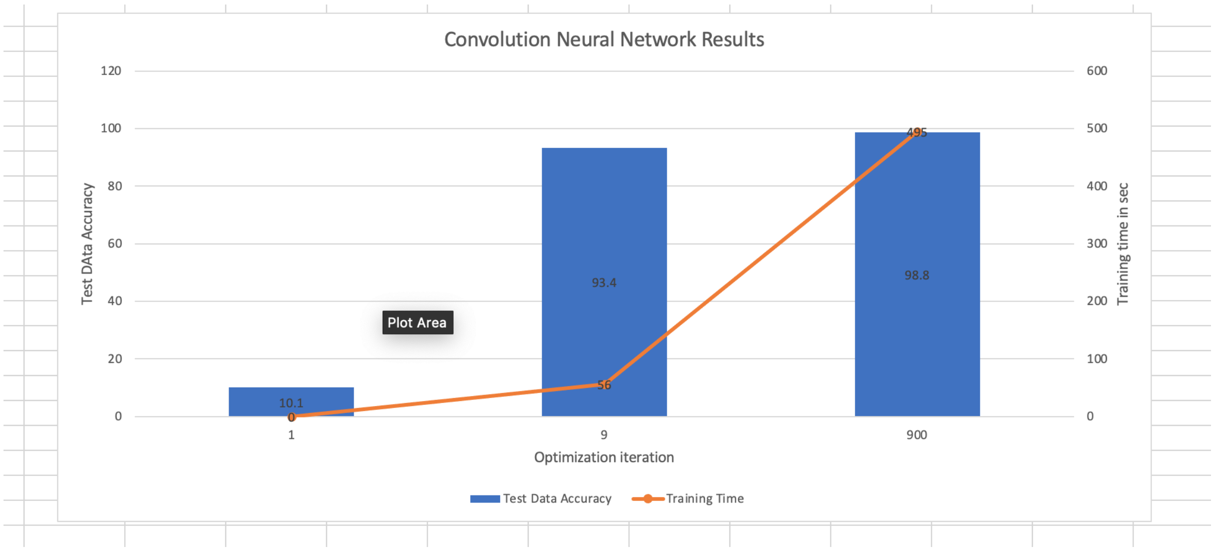
Conclusion: With the observations seen, we can say that Simple Neural Network will perform with better accuracy when compared with the Multiple layers of Deep Neural Network. Time taken for training in Simple Neural Network is also Less.

5. Convolutional Neural Network in terms of Accuracy and Training time:

No of Iterations	Test Score	Average Loss	Time to Training in sec
1	10.1	2.304254	0
9	93.4	0.222554	56
900	98.8	0.035	495

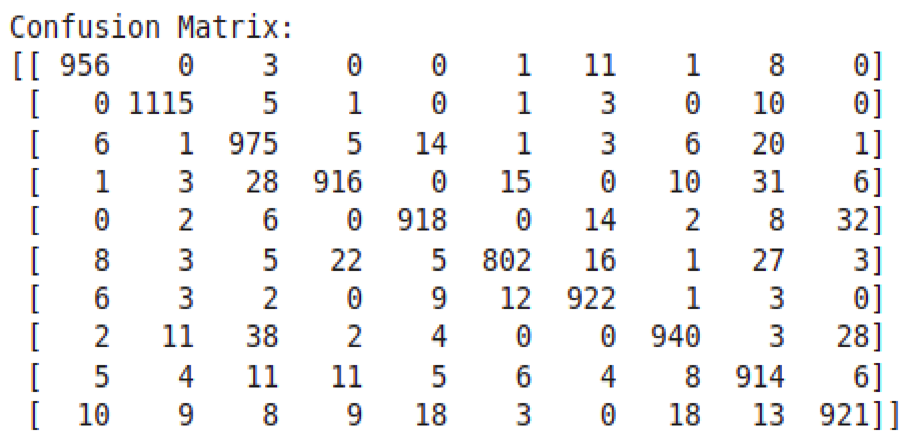
Observations:

1. From the above table, we can clearly see that the Test score is continuously increasing with the increase in the number of iterations.
2. Also, as the Test score increases the time taken to train also increases which can be seen clearly from the graph.



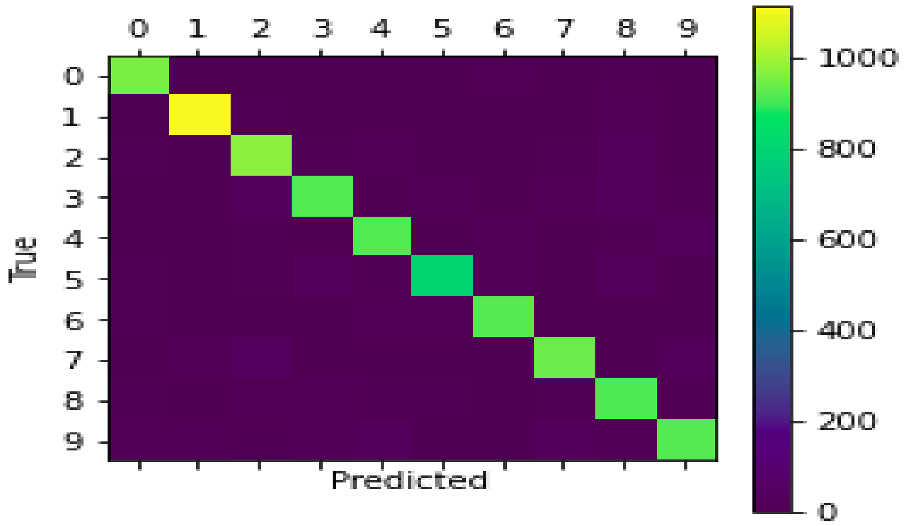
From the above graph we can clearly see that the Test Score is increasing with the increase in the number of iterations.

Confusion Matrix:

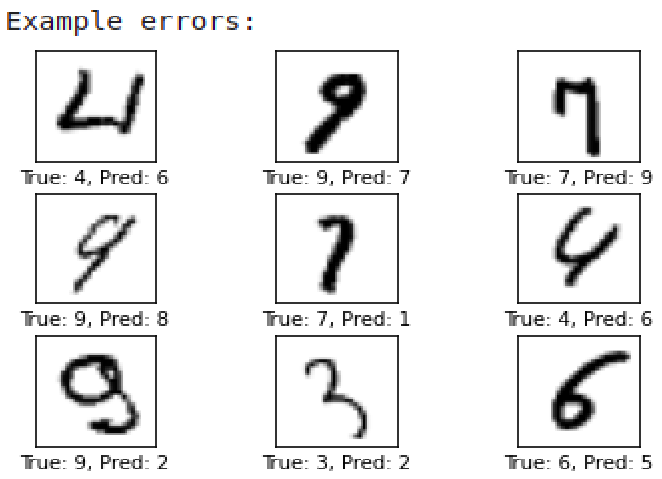


```
Confusion Matrix:
[[ 956    0    3    0    0    1   11    1    8    0]
 [    0 1115    5    1    0    1    3    0   10    0]
 [    6    1  975    5   14    1    3    6   20    1]
 [    1    3   28  916    0   15    0   10   31    6]
 [    0    2    6    0  918    0   14    2    8   32]
 [    8    3    5   22    5  802   16    1   27    3]
 [    6    3    2    0    9   12  922    1    3    0]
 [    2   11   38    2    4    0    0  940    3   28]
 [    5    4   11   11    5    6    4    8  914    6]
 [   10    9    8    9   18    3    0   18   13  921]]
```

Matlab Representation:



Sample Errors:



Extra Credits:

Comparison of the CNN vs DNN:

Both the CNN and DNN are trained on the same dataset with equal number of iterations and the following results are obtained.

Learning Rate: 0.002 for DNN script using MNIST:

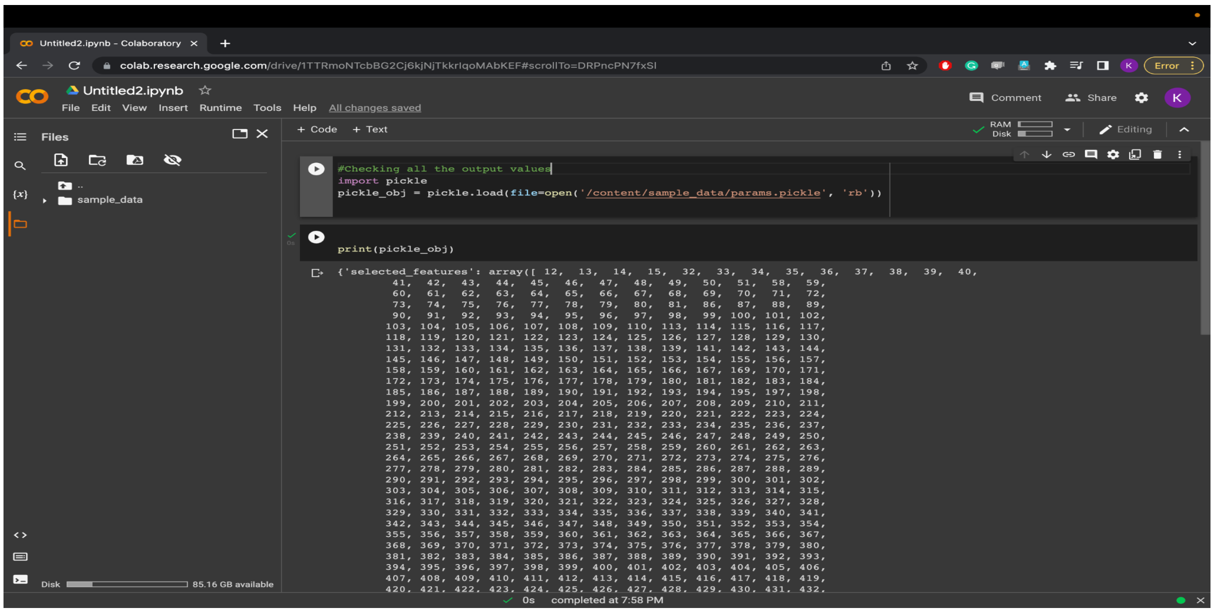
Hidden Units	Test Score	Time to Train
1	95.4	58.89
2	91.2	98.76
3	90.8	115.25
5	86.2	158.64
7	12.1	191.56

For CNN script using MNIST:

Iterations	Test Score	Time to Train
9	98.1	394.98
20	98.78	597.12

From this we can see that the highest accuracy is obtained in the CNN with MNIST when compared to the DNN script with the same dataset.

The “params.pickle” will contain the selected_features, optimal.n_hidden, w1, w2, optimal lambda.

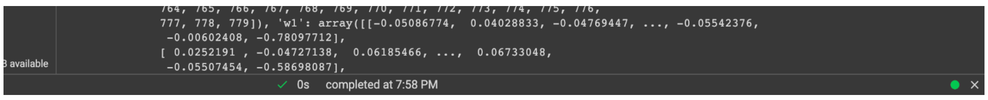


```
#Checking all the output values
import pickle
pickle_obj = pickle.load(file=open('/content/sample_data/params.pickle', 'rb'))
```

```
print(pickle_obj)
```

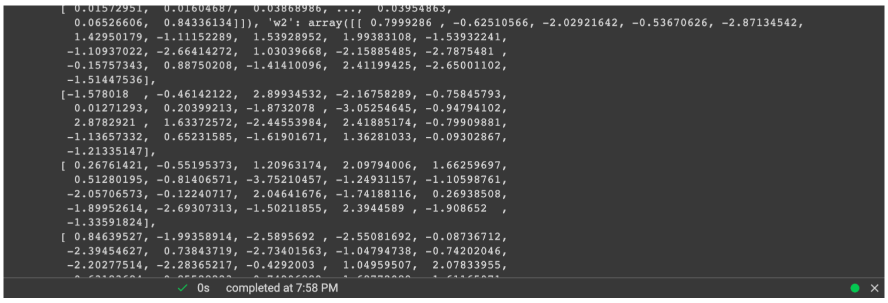
```
{'selected_features': array([ 12,  13,  14,  15,  32,  33,  34,  35,  36,  37,  38,  39,  40,
        41,  42,  43,  44,  45,  46,  47,  48,  49,  50,  51,  58,  59,
        60,  61,  62,  63,  64,  65,  66,  67,  68,  69,  70,  71,  72,
        73,  74,  75,  76,  77,  78,  79,  80,  81,  86,  87,  88,  89,
        90,  91,  92,  93,  94,  95,  96,  97,  98,  99, 100, 101, 102,
       103, 104, 105, 106, 107, 108, 109, 110, 113, 114, 115, 116, 117,
       118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130,
       131, 132, 133, 134, 135, 136, 137, 138, 139, 141, 142, 143, 144,
       145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157,
       158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 169, 170, 171,
       172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184,
       185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 197, 198,
       199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211,
       212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224,
       225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237,
       238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250,
       251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263,
       264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276,
       277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289,
       290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302,
       303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315,
       316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328,
       329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341,
       342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354,
       355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367,
       368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380,
       381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393,
       394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406,
       407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419,
       420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432,
```

✓ 0s completed at 7:58 PM



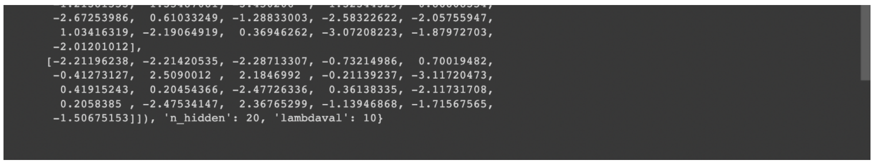
```
       764, 765, 766, 767, 768, 769, 770, 771, 772, 773, 774, 775, 776,
       777, 778, 779]), 'w1': array([[-0.05086774,  0.04028833, -0.04769447, ..., -0.05542376,
        -0.00602408, -0.78097712],
       [ 0.0252191 , -0.04727138,  0.06185466, ...,  0.06733048,
        -0.05507454, -0.58698087],
```

✓ 0s completed at 7:58 PM



```
       [ 0.01572951,  0.01604687,  0.03868986, ...,  0.03954863,
         0.06526606,  0.84336134]]), 'w2': array([[ 0.7999286 , -0.62510566, -2.02921642, -0.53670626, -2.87134542,
         1.42950179, -1.11152289,  1.53928952,  1.99383108, -1.53932241,
        -1.10937022, -2.66414272,  1.03039668, -2.15885485, -2.7875481 ,
        -0.15757343,  0.88750208, -1.41410096,  2.41199425, -2.65001102,
        -1.51447536],
       [-1.578018  , -0.46142122,  2.89934532, -2.16758289, -0.75845793,
         0.01271293,  0.20399213, -1.8732078 , -3.05254645, -0.94794102,
         2.8782921 ,  1.63372572, -2.44553984,  2.41885174, -0.79909881,
        -1.13657332,  0.65231585, -1.61901671,  1.36281033, -0.09302867,
        -1.21335147],
       [ 0.26761421, -0.55195373,  1.20963174,  2.09794006,  1.66259697,
         0.51280195, -0.81406571, -3.75210457, -1.24931157, -1.10598761,
        -2.05706573, -0.12240717,  2.04641676, -1.74188116,  0.26938508,
        -1.89952614, -2.69307313, -1.50211855,  2.3944589 , -1.908652  ,
        -1.33591824],
       [ 0.84639527, -1.99358914, -2.5895692 , -2.55081692, -0.08736712,
        -2.39454627,  0.73843719, -2.73401563, -1.04794738, -0.74202046,
        -2.20277514, -2.28365217, -0.4292003 ,  1.04959507,  2.07833955,
```

✓ 0s completed at 7:58 PM



```
        -2.67253986,  0.61033249, -1.28833003, -2.58322622, -2.05755947,
         1.03416319, -2.19064919,  0.36946262, -3.07208223, -1.87972703,
        -2.01201012],
       [-2.21196238, -2.21420535, -2.28713307, -0.73214986,  0.70019482,
        -0.41273127,  2.5090012 ,  2.1846992 , -0.21139237, -3.11720473,
         0.41915243,  0.20454366, -2.47726336,  0.36138335, -2.11731708,
         0.2058385 , -2.47534147,  2.36765299, -1.13946868, -1.71567565,
        -1.50675153]]), 'n_hidden': 20, 'lambdaval': 10}
```