CSE4/574: - Introduction to Machine Learning

Programming Assignment 2

Handwritten Digits Classification

Project Report

Name of Students: -

- *Apurva Chavan* *(50365703)*
- *Siddhi Thakur* *(50365530)*
- *Mehvish Shamshad* *(50374333)*

Introduction: -

- In this assignment, our task is to implement Perceptron Neural Network and evaluate its performance. As a part of it, we have tuned the hyperparameters, analysed the classification results and evaluated the results on two different data sets (MNIST and FACE_ALL).
- We have also implemented the Deep Neural Network on Face_all dataset using the TensorFlow Library.

Data Pre-processing:

As a first step towards building our Neural Network, we are loading the MNIST dataset and pre-processing it.
This includes feature selection, splitting the data into train and validation test.

Feature Selection:

- There are certain features in the Data Set for which the values are equal or same for all data points. These features will not contribute more to the classification and will not help model.
- So, we are removing such features from our dataset.
- **Total Number of features Selected** – ***717***

[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, 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, 196, 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, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 646, 647, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691, 692, 693, 694, 695, 696, 697, 698, 702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 720, 721, 722, 723, 724, 725, 726, 731, 732, 733, 734, 735, 736, 737, 738, 739, 740, 741, 742, 743, 744, 745, 746, 747, 748, 749, 750, 751, 752, 753, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770, 771, 772, 773, 774, 775, 776, 777, 778, 779]

Training the Neural Network:

Feed Forward Pass:

- In this step, with the help of input features and weights we are determining the class of a particular feature vector.

Backward Propagation:

- After the feed forward pass, we get the probabilities of features belonging to a particular digit.
- We classify it to the highest probable class.
- In the training process, we determine the error in classification of data and propagate that error backwards from output to input and update the weights.

Regularization:

- Regularization is an important step in model training in order to find the good balance without underfitting or overfitting the data.
- In order to do so, we vary the hyperparameters (lambda and hidden units). We select the optimal combination giving highest test accuracy.
- We need to select the hidden units and lambda, so that we get a generalized model. Say suppose if we have a single unit in the hidden layer, we could lose information and the model will be unable to predict accurately. On the other hand, if we have large number of hidden units, the model will learn accurately on the training data but its performance on validation and test data will be terrible.
- In order to get a better generalized model and improve the test accuracies we experiment and test on combinations of lambda and hidden units and get the optimal values respectively.

Selection of Hyper-parameters:

- Let's tune the hyper-parameter ***'λ'*** and the number of hidden units in the hidden units.
- We're varying ***'λ'*** from 0 to 60 in increments of 10 ***(0, 10, 20……, 60)*** and the number of hidden units in the hidden layer are varied in increments of 4 ***(4, 8, 12, 16, ……)***.

Below are the hyper-parameter combinations : -

[45]

	λ	hidden_units	Train_Accuracy	Validation_Accuracy	Test_Accuracy	Training_Time
6	0.0	28.0	94.944	93.73	94.56	73.352560
13	10.0	28.0	94.922	94.04	94.33	72.133948
18	20.0	20.0	94.374	93.40	94.14	56.767084
20	20.0	28.0	94.448	93.46	93.97	62.328781
41	50.0	28.0	94.142	93.35	93.91	62.799378
5	0.0	24.0	94.234	93.19	93.85	64.738766
34	40.0	28.0	94.108	93.09	93.74	65.638479
40	50.0	24.0	94.372	93.40	93.73	58.719894
19	20.0	24.0	94.128	93.19	93.73	54.843787
12	10.0	24.0	94.176	93.18	93.69	61.904392
4	0.0	20.0	94.214	93.38	93.69	58.961259
27	30.0	28.0	93.834	93.01	93.58	65.791053
32	40.0	20.0	93.690	92.83	93.43	57.621713
26	30.0	24.0	93.860	92.88	93.35	60.868069
25	30.0	20.0	93.896	92.80	93.20	55.379991
11	10.0	20.0	93.418	92.47	93.15	62.439204
33	40.0	24.0	93.658	92.65	92.94	63.242834

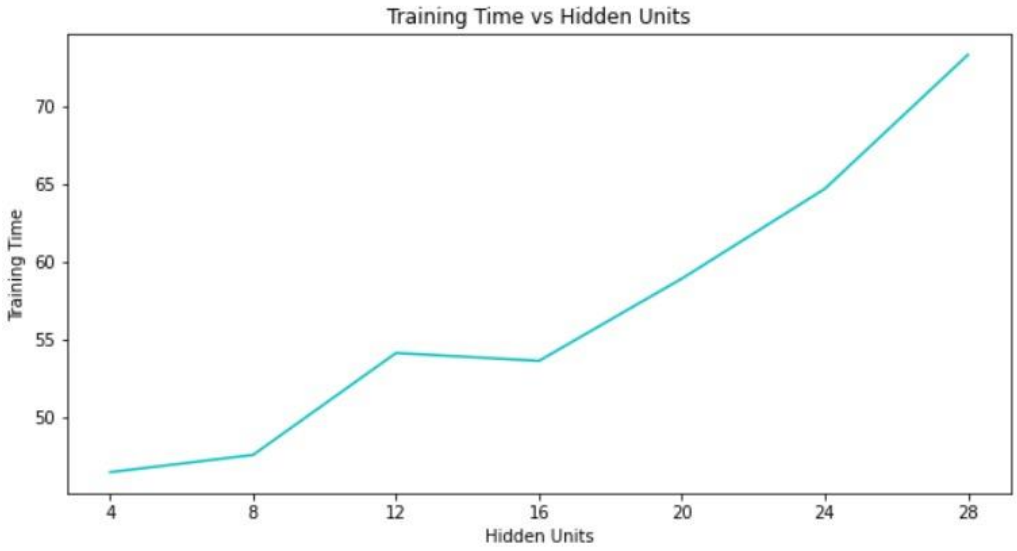
33	40.0	24.0	93.658	92.65	92.94	63.242834
39	50.0	20.0	93.462	92.54	92.91	60.535415
38	50.0	16.0	93.524	92.24	92.76	52.240110
10	10.0	16.0	93.016	91.77	92.72	56.076441
17	20.0	16.0	92.672	91.92	92.41	51.663628
24	30.0	16.0	92.758	91.71	92.34	55.199131
30	40.0	12.0	92.502	91.26	92.26	47.614496
31	40.0	16.0	92.408	91.18	92.04	53.943763
3	0.0	16.0	92.430	91.52	91.94	53.665437
9	10.0	12.0	92.108	90.73	91.90	55.789387
23	30.0	12.0	91.554	90.23	90.90	53.077883
16	20.0	12.0	91.108	90.28	90.81	50.481463
2	0.0	12.0	91.096	89.86	90.65	54.170208

- So, from the above table we can see that the best possible hyper parameter combination is **'λ' = 0** and the number of hidden units = 28, as we get the highest test accuracy from this combination (94.56%).

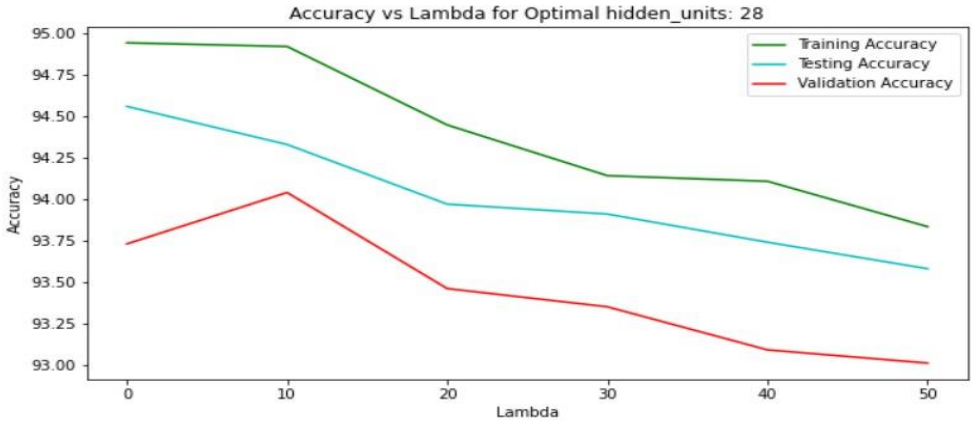
```
--- 72.81464767456055 seconds ---

 Training set Accuracy:94.94399999999999%

 Validation set Accuracy:93.73%

 Test set Accuracy:94.56%
```

Graphical Representations: -

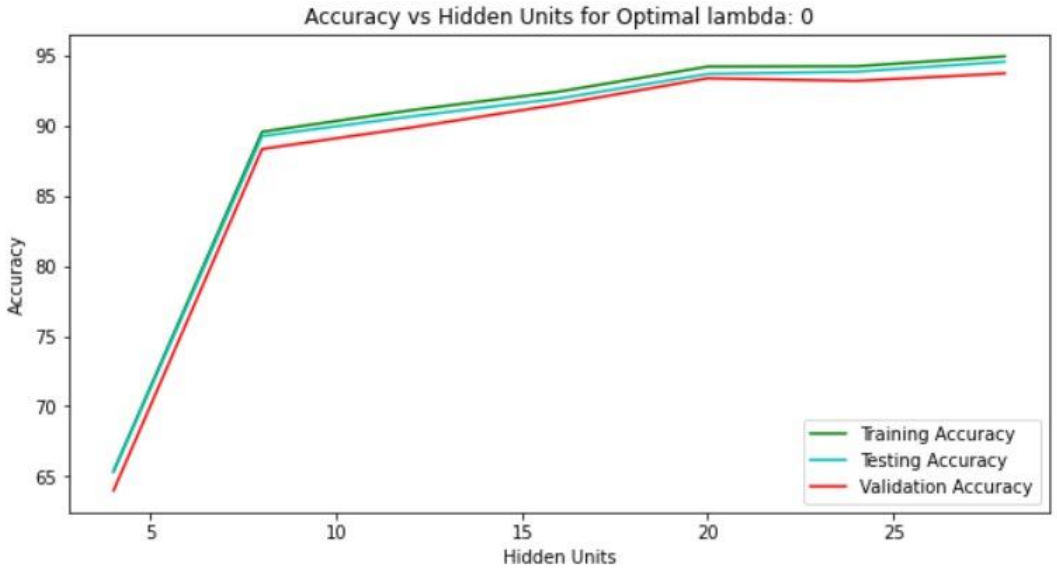
- Let's plot the ***'Training Time vs Hidden Units'*** graph.



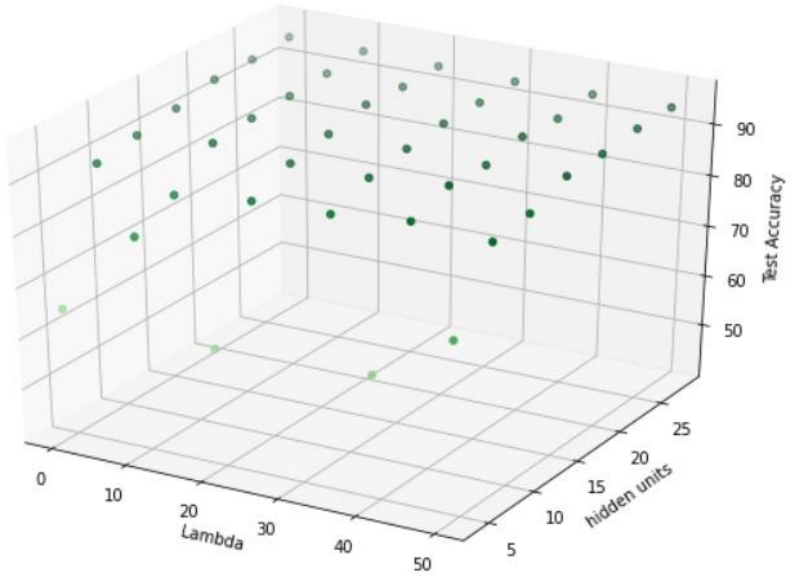
- Let's plot the ***'Accuracy vs Lambda for Optimal hidden units: 28'*** graph.



- Now, Let's plot the ***'Accuracy vs Hidden Units for Optimal Lambda: 0'*** graph.



- Finally let's plot the ***'lambda vs hidden units vs Accuracy'*** graph.



Increasing the maximum number of iterations: -

- We changed the value of ***'maxiter': 50*** to ***'maxiter': 100*** for the best hyper-parameter combination (***'λ'*** **= 0** and the number of hidden units = 28) and got the following result: -

```
--- 128.08856296539307 seconds ---

Training set Accuracy:96.614%

Validation set Accuracy:95.53%

Test set Accuracy:95.81%
```

- Again, we changed the value of ***'maxiter': 100*** to ***'maxiter': 200*** for the same hyper-parameter combination. The result is shown below: -

```
--- 244.06259942054749 seconds ---

Training set Accuracy:98.644%

Validation set Accuracy:95.75%

Test set Accuracy:95.59%
```

- From the above two operations, we can observe that as we increase the maximum number of iterations, the Training accuracy increases (***'maxiter': 100 = 96.614% to 'maxiter': 200 = 98.644%)*** but the Test accuracy decreases (***'maxiter': 100 = 95.81% to 'maxiter': 200 = 95.59%)*** which means that overfitting is increasing as we increase the maximum number of iterations.

Implementing our perceptron Neural Network on the *'FACE_ALL'* Data Set: -

- Now Let's implement our perceptron neural network on the ***'FACE_ALL'*** Data Set and observe the results.

  For ***'FACE_ALL'*** Data Set: -

```
Training set Accuracy:84.62085308056872%

Validation set Accuracy:83.30206378986867%

Test set Accuracy:84.33005299015896%

Time taken = 67.86341118812561
```

Deep Neural Network Performance on *'FACE_ALL'* Data Set

- First, we implemented using the TensorFlow library on the ***'FACE_ALL'*** Data Set with ***'1 Hidden Layer' and 28 units in the hidden layer.***

```
Time take = 35.81646013259888
Optimization Finished!
Accuracy: 0.8304315
```

- Next, we implemented using the TensorFlow library on the ***'FACE_ALL'*** Data Set with ***'2 Hidden Layers'***.

```
Time take = 108.86995649337769
Optimization Finished!
Accuracy: 0.78955334
```

- Next, we added one more hidden layer making it a total of ***'3 Hidden Layers'***.

```
Time take = 113.58905839920044
Optimization Finished!
Accuracy: 0.78046936
```

- Now we implemented using the TensorFlow library on the ***'FACE_ALL'*** Data Set with ***'5 Hidden Layers'***.

```
Time take = 121.17231130599976
Optimization Finished!
Accuracy: 0.75397426
```

- Finally, we used ***'7 hidden layers'*** on the ***'FACE_ALL'*** Data Set.

```
Time take = 139.78609442710876
Optimization Finished!
Accuracy: 0.7573808
```

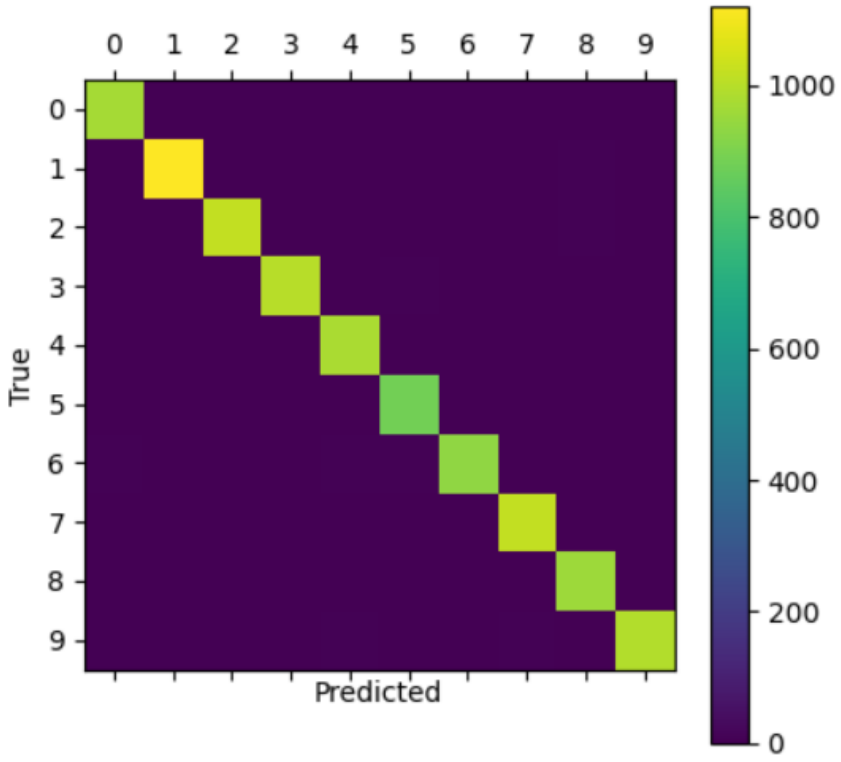
- We observe that the accuracy of Perceptron Neural Network is 84.33% and 83% using Tensor-flow. Though there is not much difference in the accuracy, but we see difference in the time taken.

- As we add more hidden layers and increase the hidden units to 256, the training time increases as the predictive model becomes more complex. Also, we also notice that the accuracy of the model decreases as the number of hidden layers increases.
- Again, this happens because of the overfitting of the model. The model fits the parameters as per the training data that it no longer generalizes to the data outside training data.
- We changed the value of number of features in each hidden layer to ***500*** for a 2-layer Deep Neural Network Model (***n_hidden_1 = 500 and n_hidden_2 = 500***) and observed the following: -

```
Time take = 158.09308409690857
Optimization Finished!
Accuracy: 0.8080999
```

Results from the '*Convolutional Neural Network*' in terms of Accuracy and Training Time and the Confusion Matrix.

```
Time usage: 0:02:58
Accuracy on Test-Set: 98.7% (9870 / 10000)
Confusion Matrix:
[[ 972    0    1    0    0    1    1    1    4    0]
 [   0 1121    4    0    1    0    1    1    7    0]
 [   0    0 1018    1    2    0    0    4    7    0]
 [   0    0    2  999    0    5    0    2    2    0]
 [   0    0    1    0  979    0    0    0    1    1]
 [   2    0    1    2    0  883    1    0    3    0]
 [   7    3    0    0    5    7  934    0    2    0]
 [   1    1    3    3    0    0    0 1017    1    2]
 [   3    0    2    3    2    1    0    2  957    4]
 [   0    2    0    2    6    2    1    5    1  990]]
```



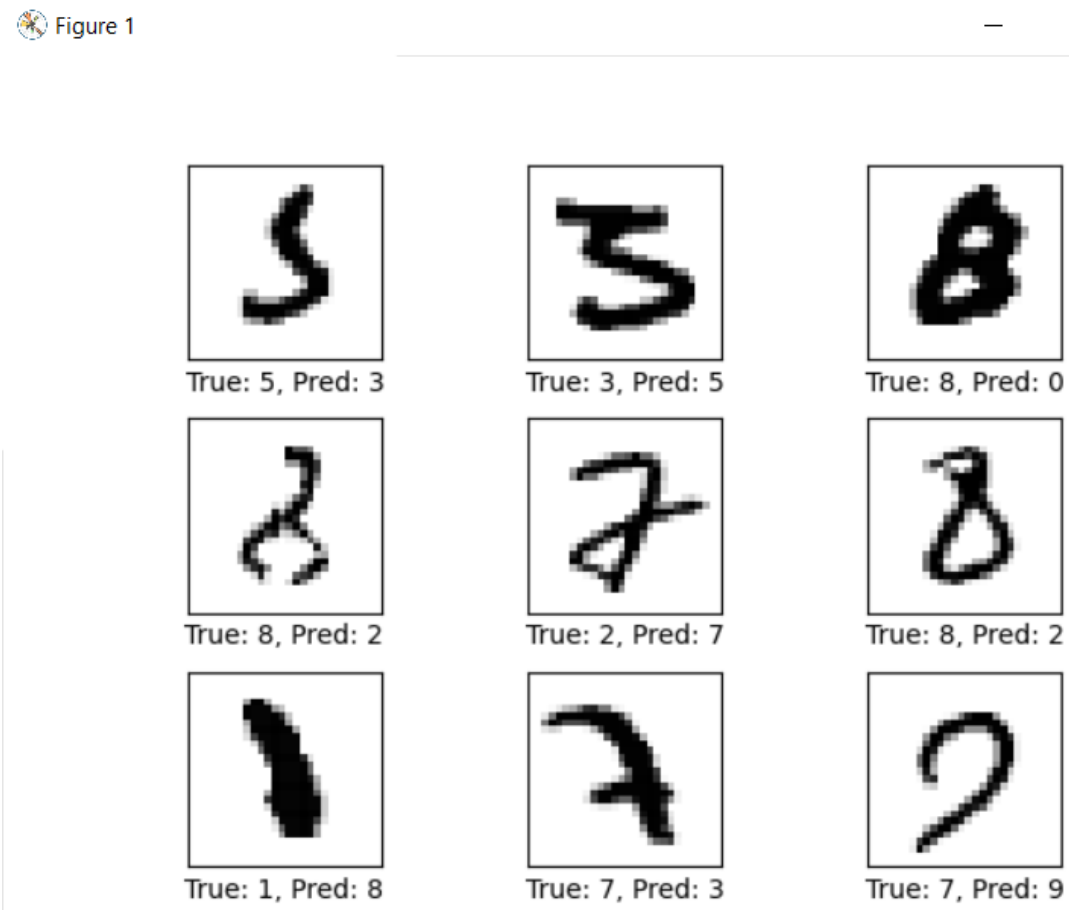


Figure 1
True: 5, Pred: 3
True: 3, Pred: 5
True: 8, Pred: 0
True: 8, Pred: 2
True: 2, Pred: 7
True: 8, Pred: 2
True: 1, Pred: 8
True: 7, Pred: 3
True: 7, Pred: 9