ML Assignment – 3 Logistic Regression and SVM

Participants: Keshav Narayan Srinivasan (50610509)

Pramila Yadav (50613803)

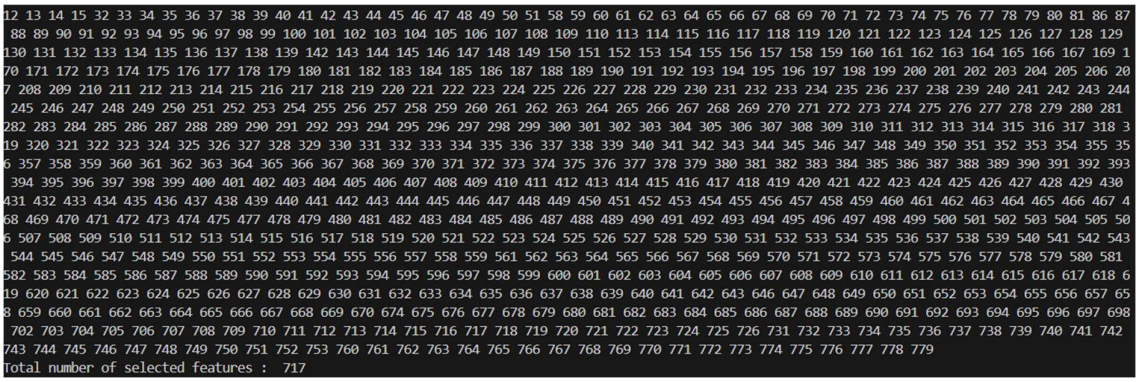
Hari Chandan Gooda (50614165)

Implement Logistic Regression and give the prediction results

Feature Selection Detail:

To optimize model training, we are ignoring features which remain same across all samples because it will not provide any learning value to training set and we are selecting features which will provide meaningful insights.

Total no. of selected features is 717



Use the Support Vector Machine (SVM) to perform classification:

For optimizing the neural network's hyperparameters focused on two key elements which are regularization coefficient (λ) and the number of hidden units (m).

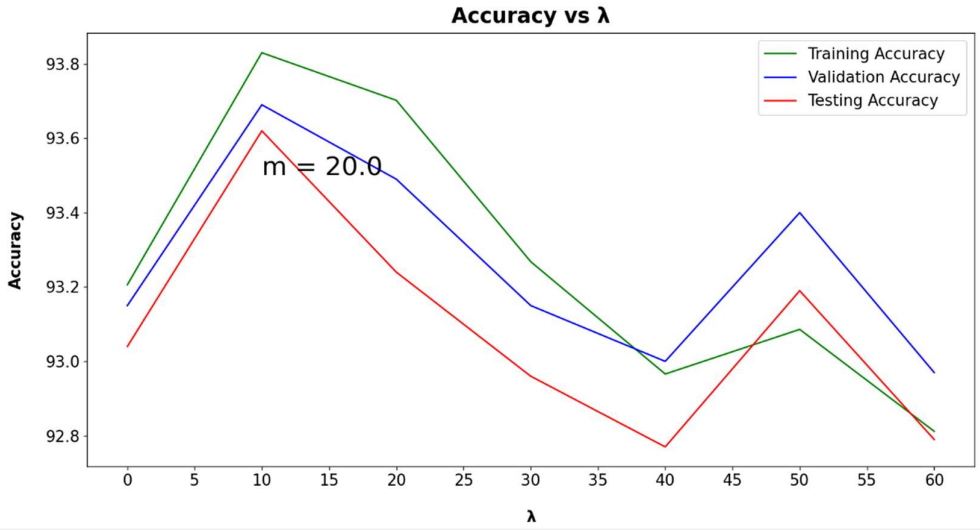
To address **overfitting**, we applied regularization and found that the number of hidden units significantly affects the model's performance. We adjusted m value from 4 to 20 (step size 4) and λ value from 0 to 60 (step size 10)

Lambda	Training Accuracy	Validation Accuracy	Test Accuracy	m
10	93.83	93.69	93.62	20
20	93.702	93.49	93.24	20
50	93.086	93.4	93.19	20
0	93.206	93.15	93.04	20
30	93.268	93.15	92.96	20
10	93.346	93.09	92.91	16
60	92.812	92.97	92.79	20
40	92.966	93	92.77	20
40	92.41	92.1	92.45	12
40	92.39	92.4	92.4	16
0	92.774	92.64	92.37	16
30	92.528	92.55	92.35	16
20	92.812	92.74	92.32	16

From the above table we can get optimum Lambda as 10 & m as 20 giving Test Accuracy of 93.62%.

Taking m value as 20

Lambda	Training Accuracy	Validation Accuracy	Test Accuracy	m
10	93.83	93.69	93.62	20
20	93.702	93.49	93.24	20
50	93.086	93.4	93.19	20
0	93.206	93.15	93.04	20
30	93.268	93.15	92.96	20
60	92.812	92.97	92.79	20
40	92.966	93	92.77	20



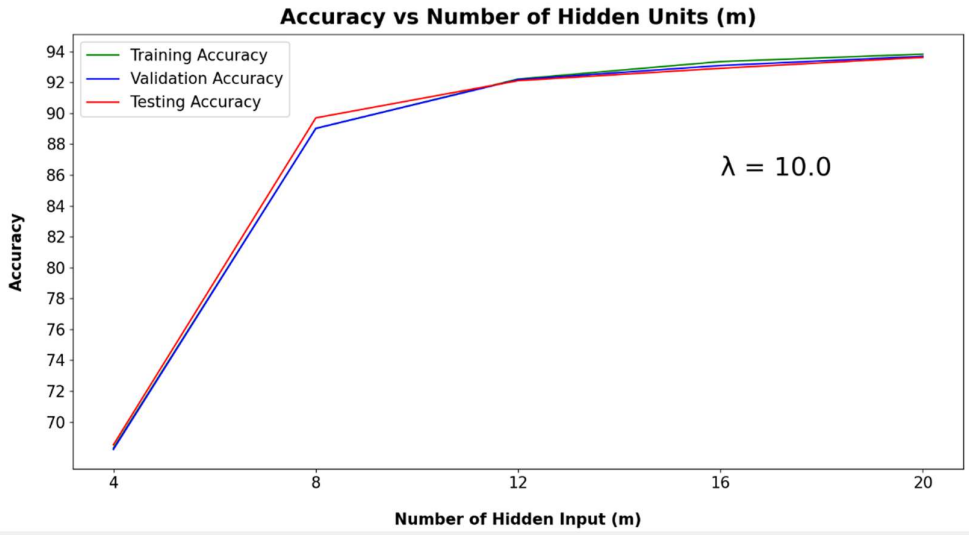
The highest accuracy in testing was observed when λ is set to 10. As λ increases, there are slight variations in training, testing and validation datasets. According to theory, accuracy should decrease after reaching the optimal point due to underfitting, and our plot shows the similar behaviour. There are some minor anomalies occur, which are justifiable in a practical setup, with similar inconsistencies seen across the training, testing and validation datasets.

Taking λ as 10 and explore different values for m.

Taking **λ** value as 10

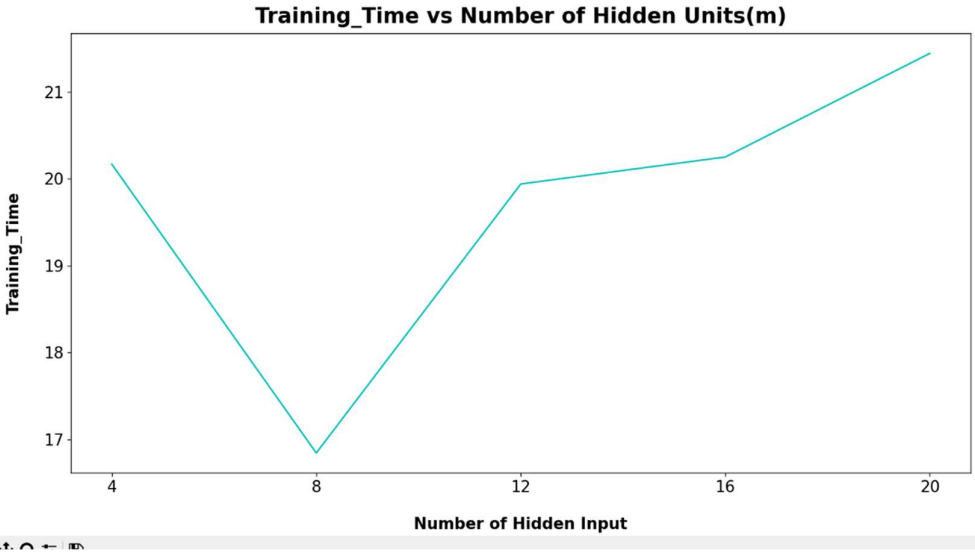
10	68.32	68.23	68.53	4
10	89.012	89.02	89.7	8
10	92.216	92.18	92.11	12
10	93.346	93.09	92.91	16
10	93.83	93.69	93.62	20

From above graph it is evident that when m is 20, we will get best result.



The plot shows that as m grows, accuracy also improves. There is a significant jump between 4 and 8, after which the increase is slow and the curve levels off from 16 to 20. This indicates that after a certain point, adding more hidden units does not significantly impact accuracy.

Plotting Training time vs no of hidden units:

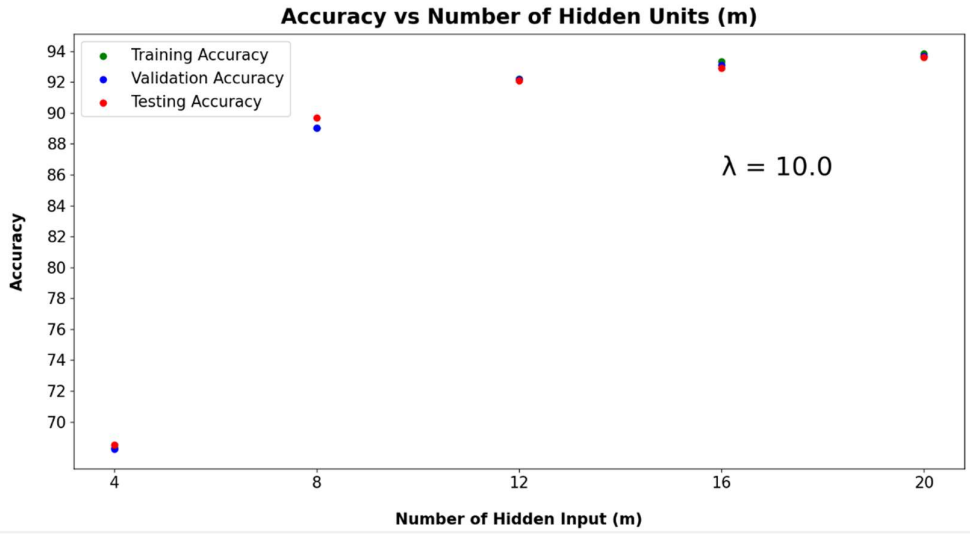


The plot indicates that as we add hidden units, training time increases. So, we should think carefully when to stop adding more units to avoid making the training time longer than needed, especially if accuracy stops improving.

Classification Accuracy on Handwritten Digits Test Data using nnScript.py:

Best Hyper Parameter combination: **λ** = 10, m = 20

scatter plot of accuracy v/s m



Classification accuracy on the celebA dataset:

Using implementations of the functions: sigmoid(), nnObjFunc(), and nPredict() from nnScript in facennScript. Below are the results obtained by running `facennScript.py` when λ is set to 10 :

```
Training set Accuracy:84.37440758293839%

Validation set Accuracy:82.92682926829268%

Test set Accuracy:84.21650264950796%
```

Compare the results of deep neural network and neural network with one hidden layer on the CelebA data set:

We analysed and compared the performance on the celebA dataset between a single-layer neural network and a multilayer deep neural network. By adjusting the layers from 2 to 7 (2, 3, 5, 7), we obtained the following results:

No of Layers	Script	Accuracy in %	Training Time in sec	Avg loss
1	facennScript.py	86.26	48.068	
2	deepnnScript.py	67.1	44.47	0.662335
3	deepnnScript.py	64.1	50.36	0.687593
5	deepnnScript.py	50.0	55.47	0.693011
7	deepnnScript.py	50.0	62.22	0.693166

Observation:

Adding more hidden layers makes the training time longer because the model becomes more complex when we add more hidden layers.

We also found that accuracy goes down with more layers. This might be because of overfitting, which causes good performance on training data but worse performance on test data.

CNN Results: Accuracy and Training Duration Analysis:

In term of Accuracy Comparison, The CNN model's accuracy is 98.7%, which is better than a simple or multilayer neural network

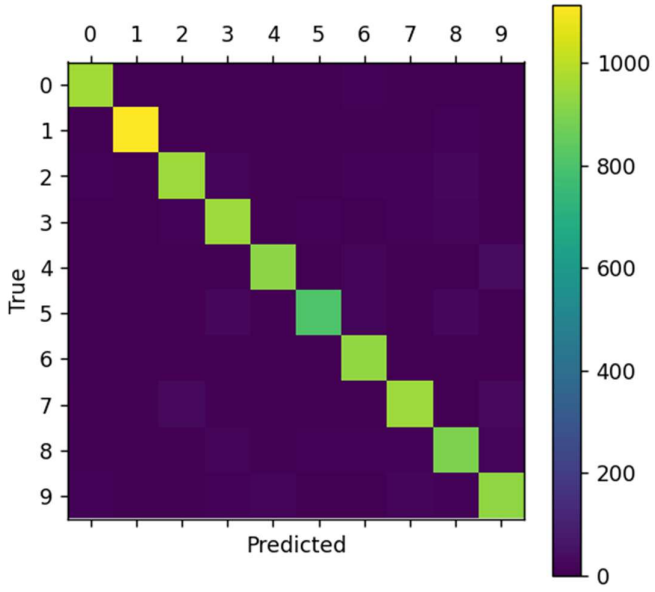
The reason why CNN Works Well is that CNN adapts its settings to match what the image needs, also it doesn't need prior knowledge to pick important features.

The Importance of Filters in CNN help break down the image into smaller sections and analysed each part which improves the accuracy.

The Training Time is 16 seconds for the CNN model using CUDA threading to achieve 93.3% accuracy.

```
loss: 0.461491  [44800/60000]
loss: 0.173865  [51200/60000]
loss: 0.283974  [57600/60000]
Time usage: 0:00:16
Test Error:
 Accuracy: 93.3%, Avg loss: 0.217007

Example errors:
Confusion Matrix:
[[ 956    0    0    1    0    4   12    2    5    0]
 [   0 1109    3    2    1    1    5    0   14    0]
 [  12    1  916   17   13    2   18   18   31    4]
 [   3    5    9  930    0   20    1   15   18    9]
 [   0    3    2    0  918    0   18    1    2   38]
 [   6    2    1   14    6  823   18    3   14    5]
 [   6    4    1    0    9   13  922    2    1    0]
 [   0    9   20    5    4    0    0  935    3   52]
 [   8    4    3   13   11   12   11    9  884   19]
 [   8    5    3    9   23    4    1   13    6  937]]
```



The matrix shows that nearly all predictions are correct resulting in a final accuracy of 93.3% and it has few mistakes.