Handwritten Digits Classification

Group 26

Observations from running the nnScript.py :

We ran several iterations of this script for different combinations of λ and number of hidden units (m) with λ ranging from 0 to 60 in increments of 10 and m ranging from 4 to 20 in increments of 4.

Choosing the hyper-parameters for Neural Network:

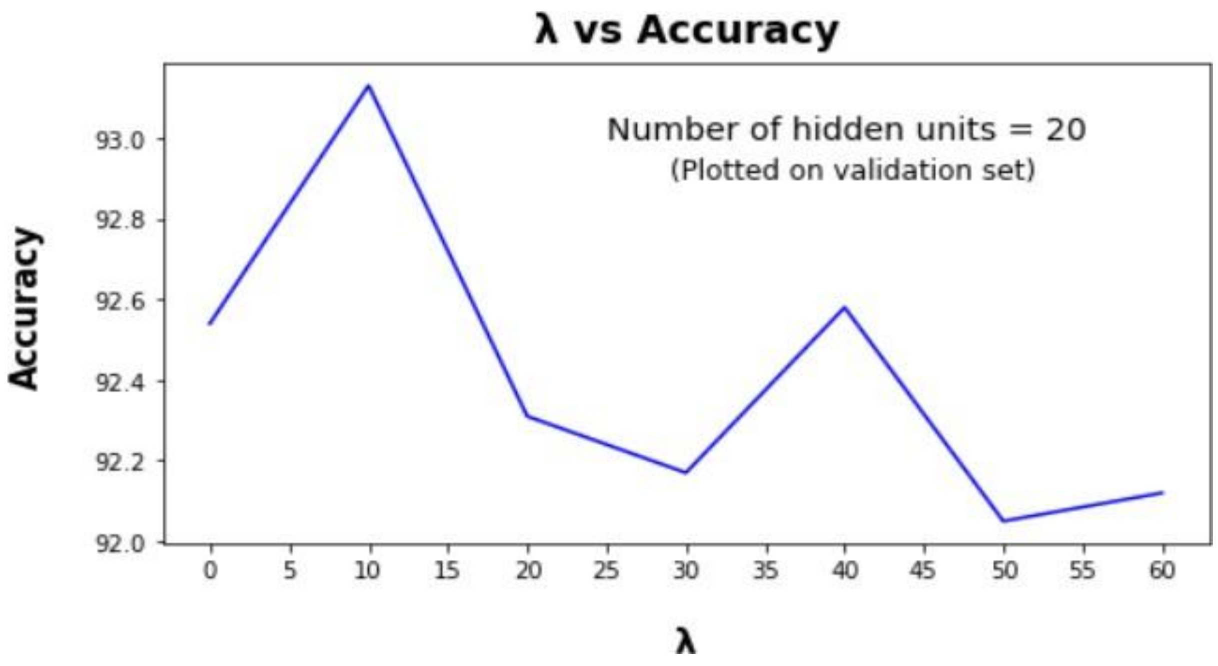
λ	m	Training set accuracy %	Validation set accuracy %	Test set accuracy %
0	4	59.87	58.06	59.46
	8	88.35	87.42	87.57
	12	91.07	90.68	90.78
	16	93.41	92.7	92.75
	20	93.16	92.4	92.9
10	4	62.38	60.67	61.68
	8	83.58	82.23	83.98
	12	91.52	90.66	91.58
	16	92.81	92.10	92.39
	20	**93.89**	**93.13**	**93.4**
20	4	67.20	65.83	66.8
	8	89.52	88.68	89.1
	12	92.53	92.05	92.36
	16	93.16	92.75	92.67
	20	93.11	92.31	92.83

30	4	66.24	65.14	65.75
	8	89.02	87.89	88.68
	12	90.72	89.95	90.29
	16	92.83	92.41	92.71
	20	92.94	92.17	92.74
40	4	79.36	78.49	79.46
	8	85.37	84.43	85.91
	12	91.90	91.4	91.81
	16	92.51	91.62	92.42
	20	93.11	92.58	92.86
50	4	76.10	75.6	75.55
	8	89.00	88.38	89.02
	12	91.77	91.13	91.43
	16	92.39	91.77	92.38
	20	92.84	92.05	92.85
60	4	60.06	58.69	59.4
	8	89.17	88.14	89.2
	12	89.94	89.11	90.0
	16	91.86	91.13	91.79
	20	92.66	92.12	92.54

From the table above, we observe that highest accuracy is achieved for the combination of **λ= 10 and m= 20 ,** which are the optimal values for λ and m for the given data set.

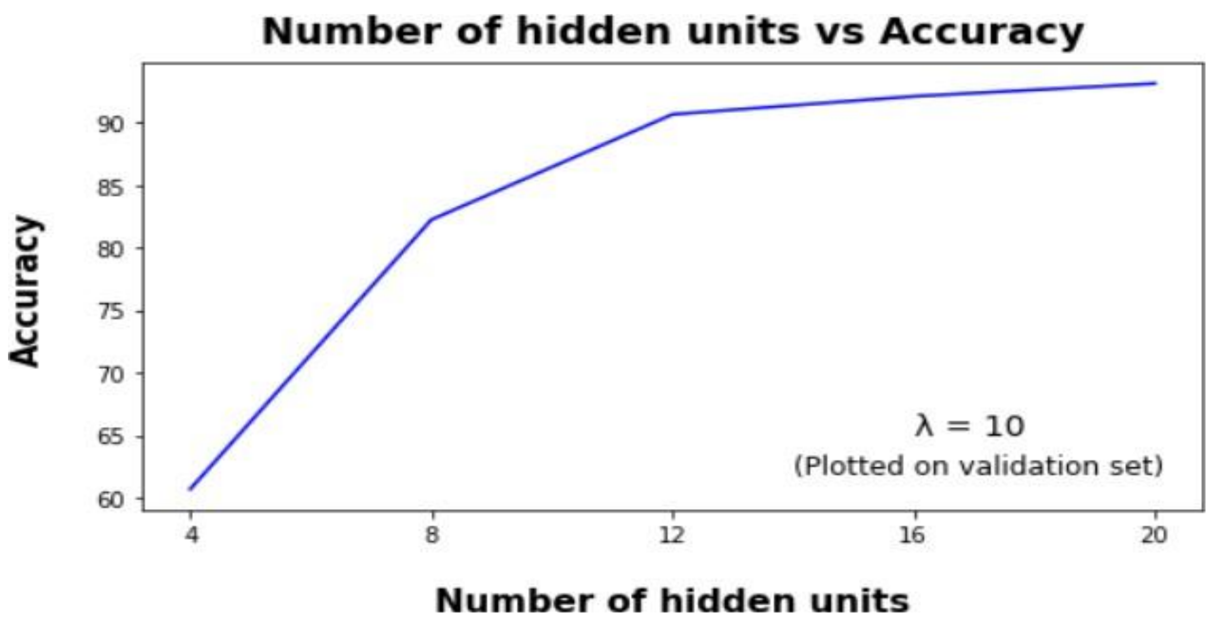
Performance Observation:

Now, we fix our m value to 20 and vary the λ values to observe its effect on the performance of the neural network. We use the validation set for this.

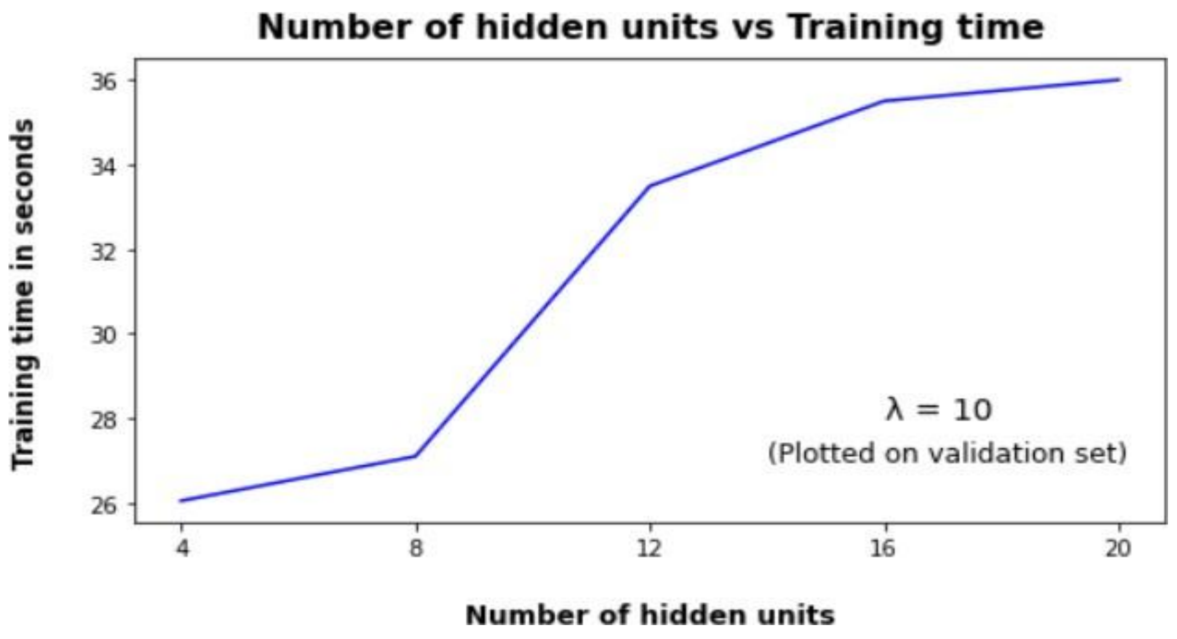


Now we observe from the above graph that there is a steady increase in accuracy till the optimal λ value, that is 10. After this, the accuracy fluctuates and decreases as the λ increases due to underfitting.

Next, we observe the behaviour of the neural network with respect to the number of hidden units used. For this, we fix the λ value to 10 and vary the m value. We also only use accuracy obtained from validation dataset.



As seen in the graph above, it is clear that the accuracy increases with increase in the number of hidden units. This is due to the fact that, number of hidden units represents the dimension of the learned features of the neural network. But this doesn't mean we can increase the number of hidden units in the neural network without a penalty. As indicated by the graph below, the training time of the neural network will increase with increase in the number of hidden units. So it is essential to choose m, number of hidden units that avoids underfitting as well as slow training.



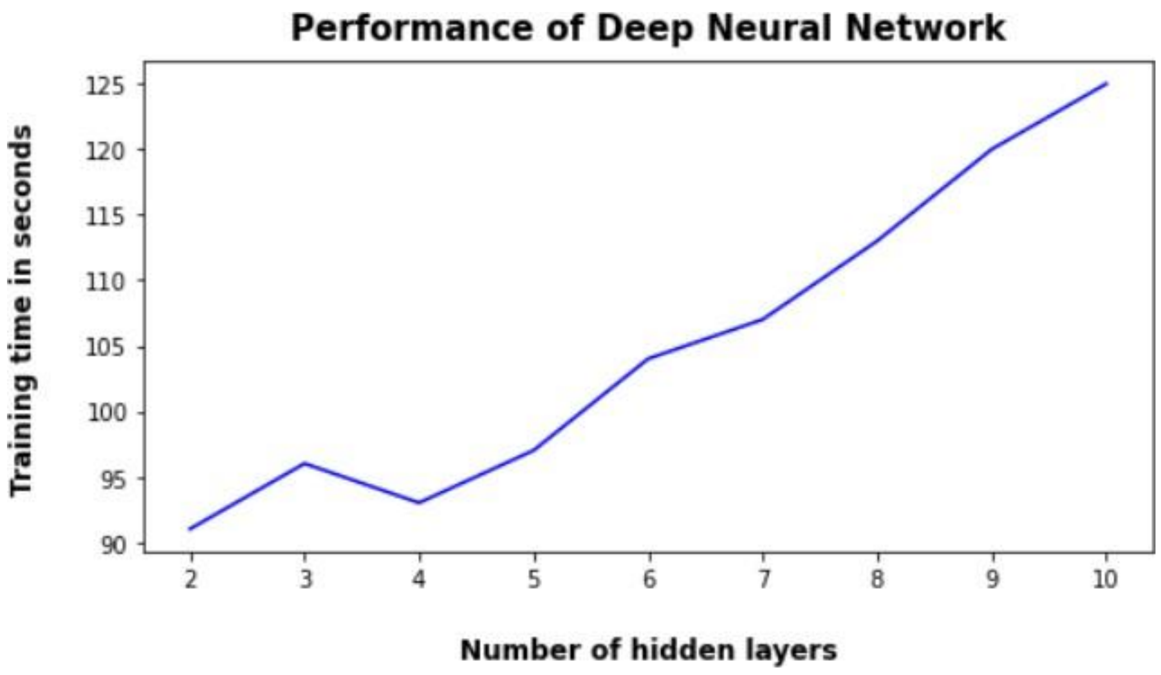
Observation from running the single layer Neural Network and multi-layer deep Neural Network:

We use the celebA dataset to observe the behaviour of single layer and multilayer Neural Network with respect to changes in the number of hidden layers.

	Training set Accuracy	**Validation set Accuracy**	**Test set Accuracy**	**Training time (In seconds)**
facennScript (1 hidden layer)	0.8547	0.8461	0.8565	77.05

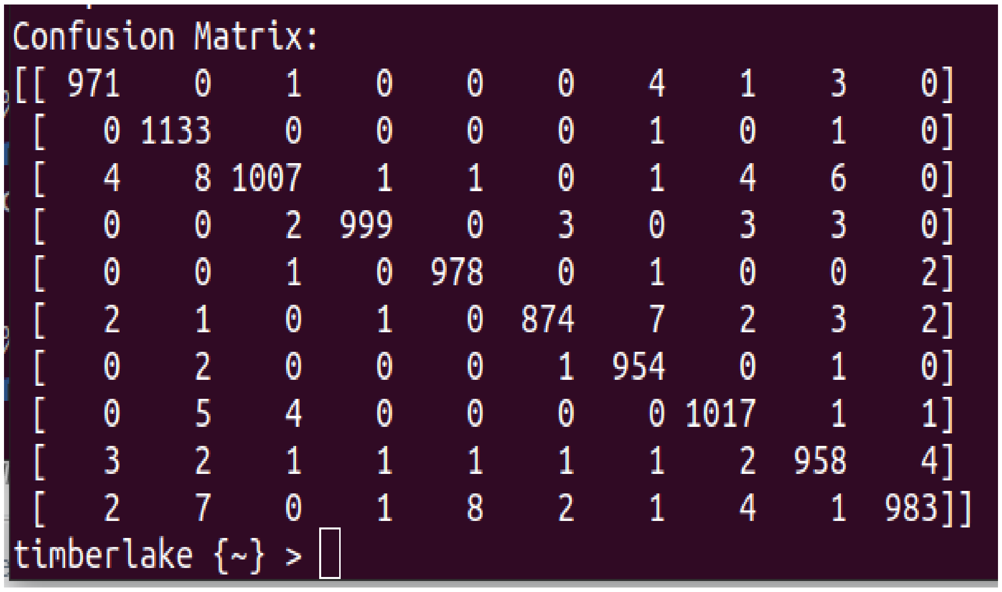
deepnnScript	**Accuracy**	**Training time (In Seconds)**
2 hidden layers	0.81	91.9182
3 hidden layers	0.7823	96.5389
5 hidden layers	0.76	97.8984
7 hidden layers	0.7392	107.8141

From the tables above, it is clear that as the number of hidden layers increases, the training time increases. Also the number of hidden layers does not improve the accuracy of the model. This is due to overfitting of the model. This observation is further cemented with the help of the graph below.



Observations from running the cnnScript.py :

We run the cnnscript.py for the given dataset which is an implementation of Convolutional Neural Network using TensorFlow library and obtained the confusion matrix:



```
Confusion Matrix:
[[ 971    0    1    0    0    0    4    1    3    0]
 [    0 1133    0    0    0    0    1    0    1    0]
 [    4    8 1007    1    1    0    1    4    6    0]
 [    0    0    2  999    0    3    0    3    3    0]
 [    0    0    1    0  978    0    1    0    0    2]
 [    2    1    0    1    0  874    7    2    3    2]
 [    0    2    0    0    0    1  954    0    1    0]
 [    0    5    4    0    0    0    0 1017    1    1]
 [    3    2    1    1    1    1    1    2  958    4]
 [    2    7    0    1    8    2    1    4    1  983]]
timberlake {~} > 
```

We make the following observations after running 10,000 iterations:

Training Time - 1840 seconds
Accuracy - 98.7%

As we can see accuracy is much better than all of the previous models we reported above. We should also note that the accuracy improved with the number of iterations. From all the observations combined, CNN becomes the best model for image classification.