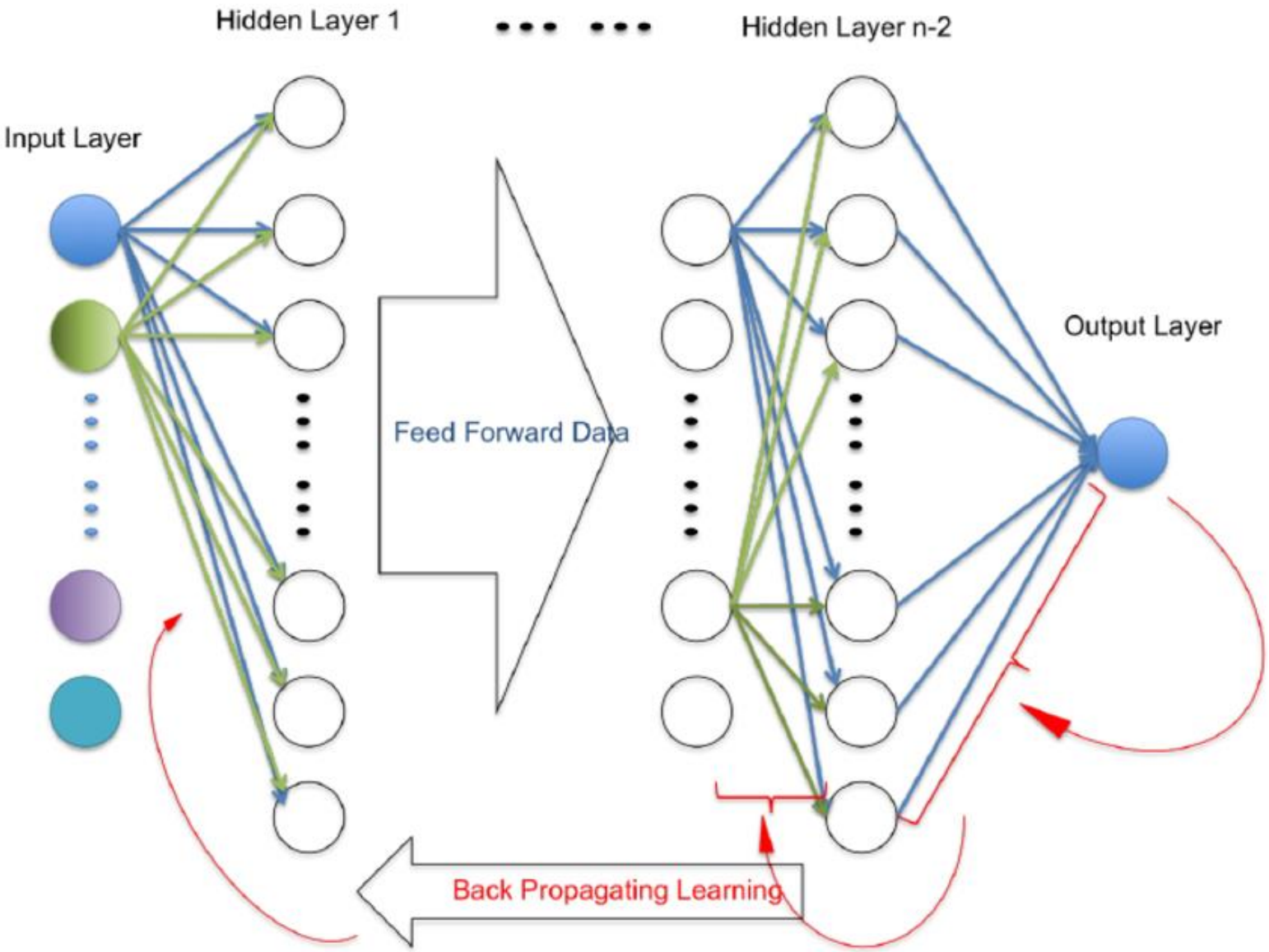
CSE574 Introduction to Machine Learning
Programming Assignment 1
Handwritten Digits Classification

Group 35

TARIQ SIDDIQUI **KAUSHIK RAMASUBRAMANIAN** **JUNAID SHAIKH**



1. Introduction:

This project implements a Multilayer Perceptron Neural Network and evaluate its performance in classifying handwritten digits. Further, same neural network implementation is used to analyze a more challenging face dataset and compare the performance of the neural network against a deep neural network using the TensorFlow library.

Our Machine Learning Model Implements Forward feed and back propagation methodology to adjust weights of model for predictive learning. The parameters in Neural Network model are the weights associated with the hidden layer units and the output layer units.

This project discusses relations and dependency of accuracy and training time of Neural network on parameters such as number of Hidden nodes, regularization coefficient, maximum iterations of conjugate gradient algorithm to perform optimization task.

2. Feature Selection in Pre-Processing Step

Feature selection is selection of attributes in our data set that are most relevant to the predictive modeling.

As mentioned in project description, there are many features for which values are exactly the same for all data points in the training set. With those features, the classification models cannot gain any more information about the difference (or variation) between data points. Therefore, we can ignore those features in the pre-processing step.

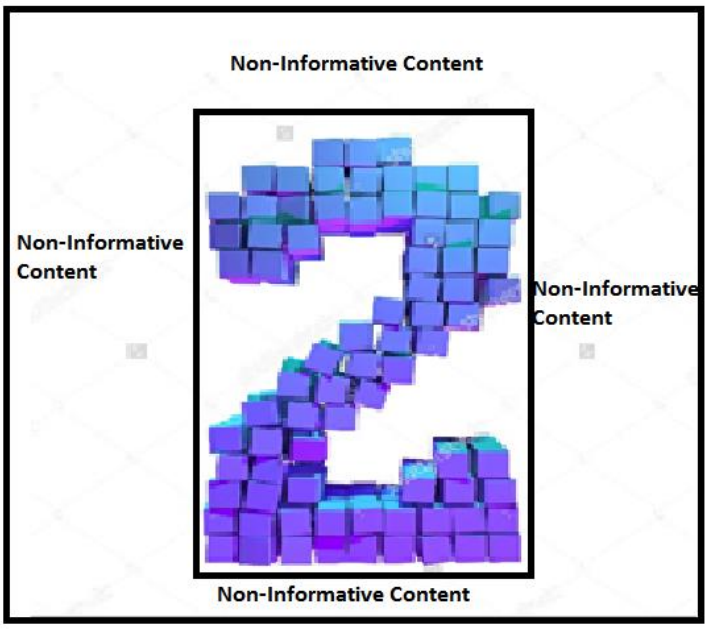


Figure1: Feature Selection Process

As shown above there are regions in Image which do not reveal much information and hence can be

removed/ignored before modelling neural network algorithm is applied. Aforesaid holds only when these non-informative attributes behave same across entire data set.

To substantiate, through our preprocessing which included removing common pixels' indexes from all the images with color same as background color, In params.pickle file submitted, we were able to identify such 71 pixel from all images and these pixel information was then removed from Training data set.

Total Number of Pixels = 28*28 = 784

Below Uninformative Pixels are removed before machine learning algorithm is applied:

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 52, 53, 54, 55, 56, 57, 82, 83, 84, 85, 111, 112, 140, 141, 168, 336, 392, 420, 448, 476, 560, 644, 645, 671, 672, 673, 699, 700, 701, 727, 728, 729, 730, 754, 755, 756, 757, 758, 759, 780, 781, 782, 783]

3. Simulation Approach

To see behavior of Training, Validation and test accuracy and training time for different values of maximum iterations, number of hidden nodes and Regularization coefficient, we created a script which iterated for maximum number of iterations for values of 50,100 and 500 along with number of hidden nodes from 4 to 100 and also varied regularization coefficient from 0 to 60. **Thus in total we evaluated 975 combinations** and in each combination we measured training, validation and test set accuracy along with training time taken.

```
matrixiteration=[0,1,2]

with open('ML_Regression.csv', 'a', newline='\n', encoding='utf-8') as csvFile:
    fieldNames = ['Matrix_iterations','Hidden', 'Lambda', 'Start time', 'End time', 'Execution time',
    writer = csv.DictWriter(csvFile, fieldnames=fieldNames)
    writer.writeheader()
    for index in matrixiteration:
        if index==0:
            numberofiterations=50
            opts = {'maxiter': 50}  # Preferred value.
        elif index==1:
            numberofiterations=100
            opts = {'maxiter': 100}  # Preferred value.
        elif index==2:
            numberofiterations=150
            opts = {'maxiter': 150}  # Preferred value.'


        for number_of_nodes in np.arange(4,104,4):
            n_hidden = number_of_nodes
            n_class = 10
            for lamb in np.arange(0,65,5):
                now_time = time.time()
                start_time = time.strftime("%X")
                lambdaval = lamb
                # initialize the weights into some random matrices
```

Figure 3-1: Python code to generate various combinations of neural network parameters

4. Simulation Summary of Best Accuracy

Below we present summary of achieved results, before we start discussing them in more detail.

Maximum Iterations	Hidden Nodes	Regularization coefficient	Training Data	Validation Accuracy	Test Accuracy
50	92	5	95.89%	95.47%	95.32%
100	100	5	98.39%	97.13%	97.25%
150	100	5	98.85%	97.22%	97.45%

Table 1: Summary for Best achieved Accuracies for different Parameters

5. Accuracy analysis of Neural Network for handwritten digits

We analyzed the performance of our implemented Neural network for percentage accuracy and time taken to train the model.

Further, we extended our analysis for 3 different values of Maximum number of iterations of conjugate gradient algorithm.

5.1 Accuracies for Maximum Iteration Count of 50

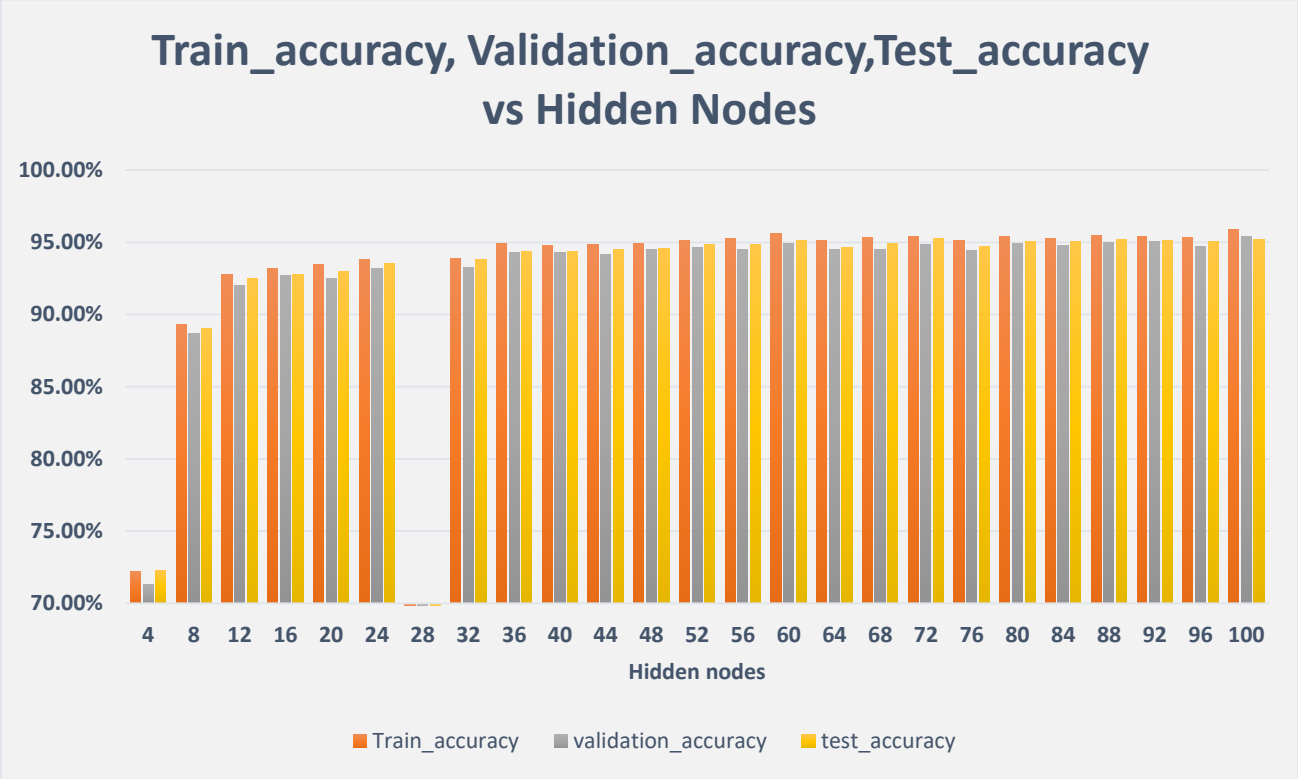


Figure 5-1-1: Variation of different Accuracies with Number of Hidden Nodes

5.2 Accuracies for Maximum Iteration Count of 100

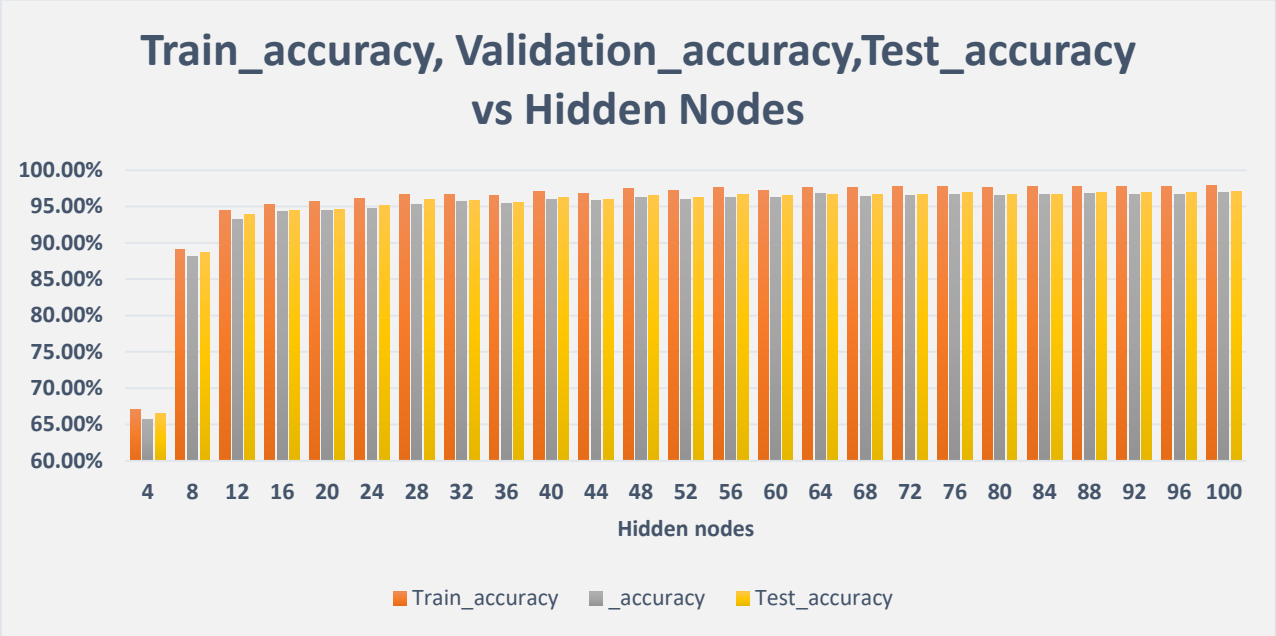


Figure 5-2-1: Variation of different Accuracies with Number of Hidden Nodes

5.3 Accuracies for Maximum Iteration Count of 150

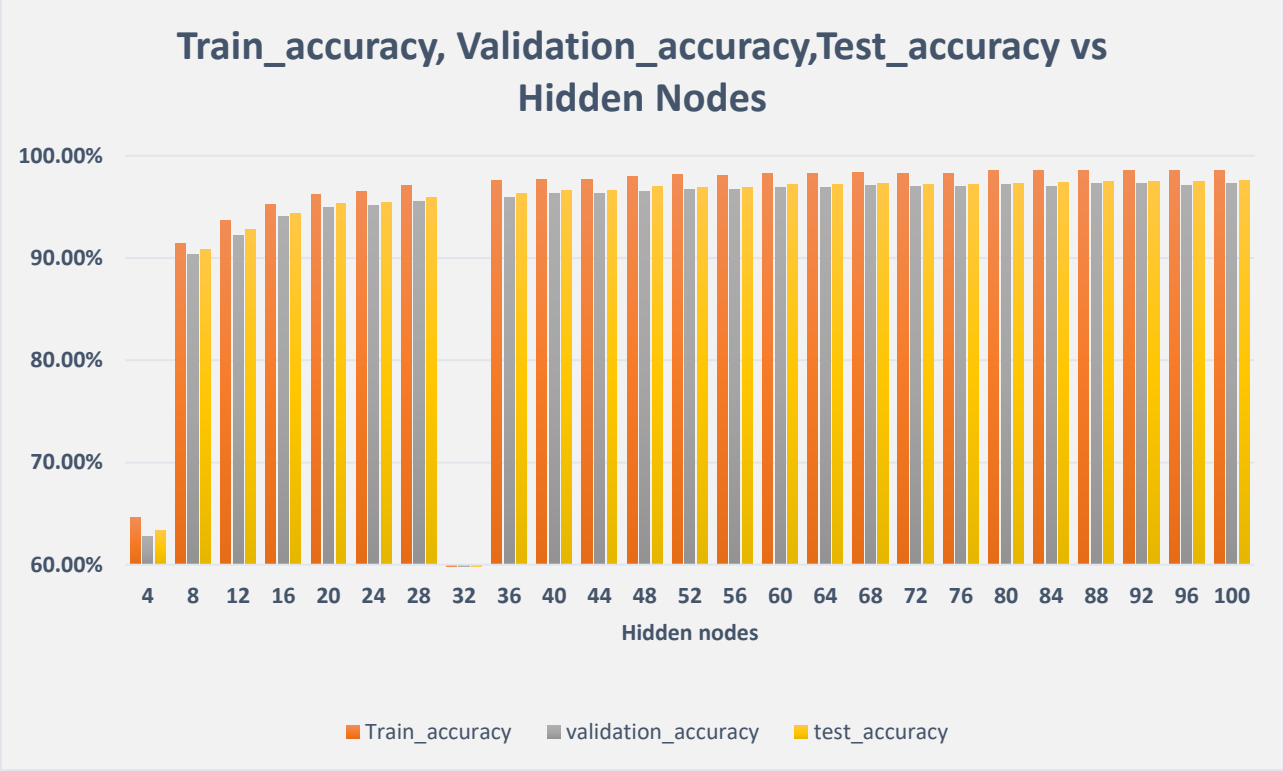


Figure 5-3-1: Variation of different Accuracies with Number of Hidden Nodes

As seen in all of above plots, training, validation and testing data accuracy increases as we increase number of hidden nodes and as we increase "maximum number of iterations count". Hence, we make our first observation as below:

Observation1: *Neural Network predictive accuracy increases as Number of Hidden units and Number of hidden layer increase:*

Increased Number of weights help to capture pattern of slightest variation of Input data. Additional hidden neural units can provide a better modeling of the data set through the additional weights corresponding to additional nodes added.

HENCE, WE HAVE CHOOSEN NUMBER OF NODES AS HIGH AS 100 IN THE FINAL SCRIPT.

Observation2: *Training, Validation and test accuracy Increases as we increase "maximum number of iterations" parameter of Optimization function:* *Explanation is implicit as optimized function is looped for larger number of times, hence weights converge to more optimum value.*

6. Training Time analysis of Neural Network for handwritten digits

In this section we study Training time of model as a function of number of hidden nodes:

*In below experiments (3 different experiments with maximum number of iterations as **50,100 and 150 in optimization function**) regularization coefficient is fixed to a constant value (10 here).

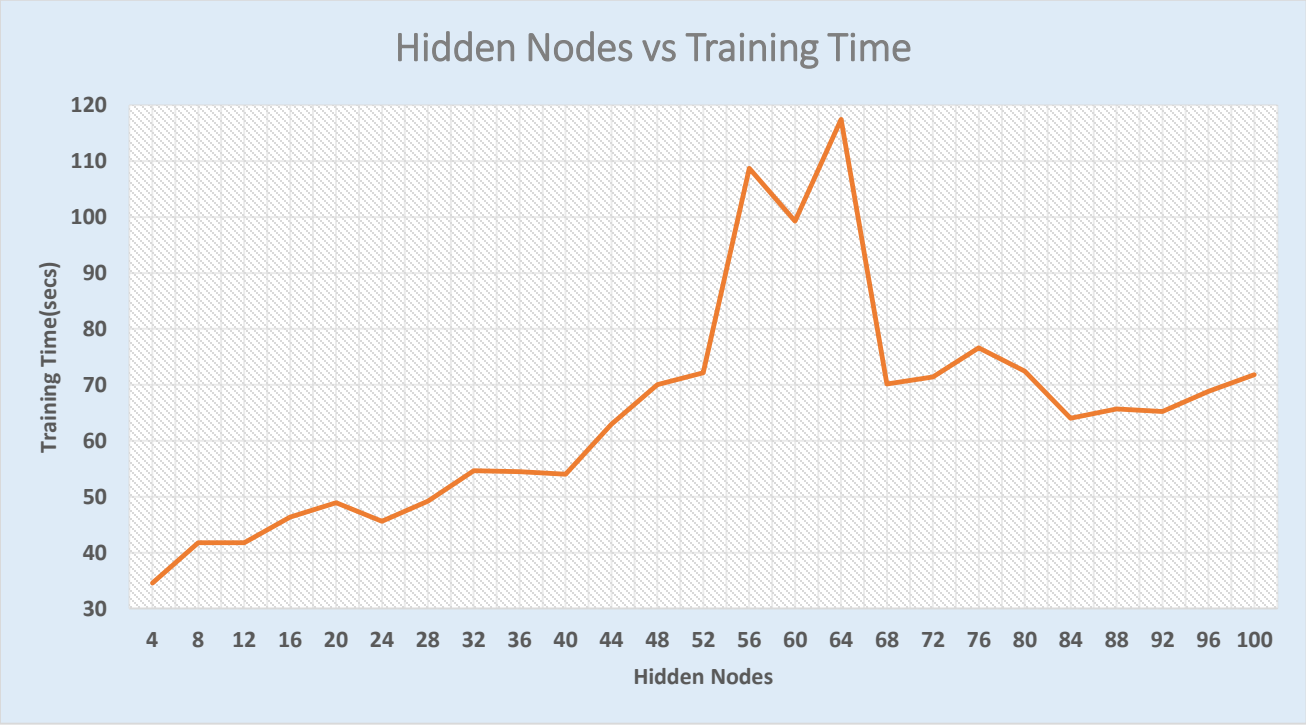


Figure 6-6-1:Variation of Training Time for different number of Hidden Nodes for maximum Iteration Value of 50

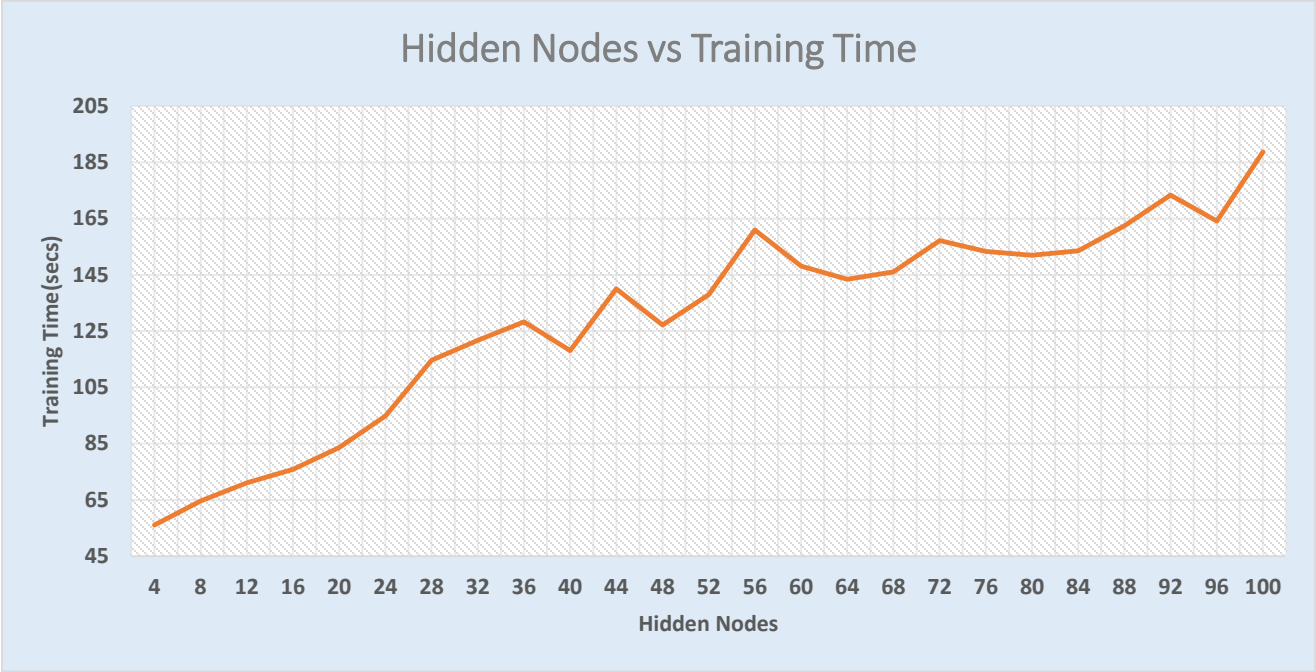


Figure 6-2: Variation of Training Time for different number of Hidden Nodes for maximum Iteration Value of 100

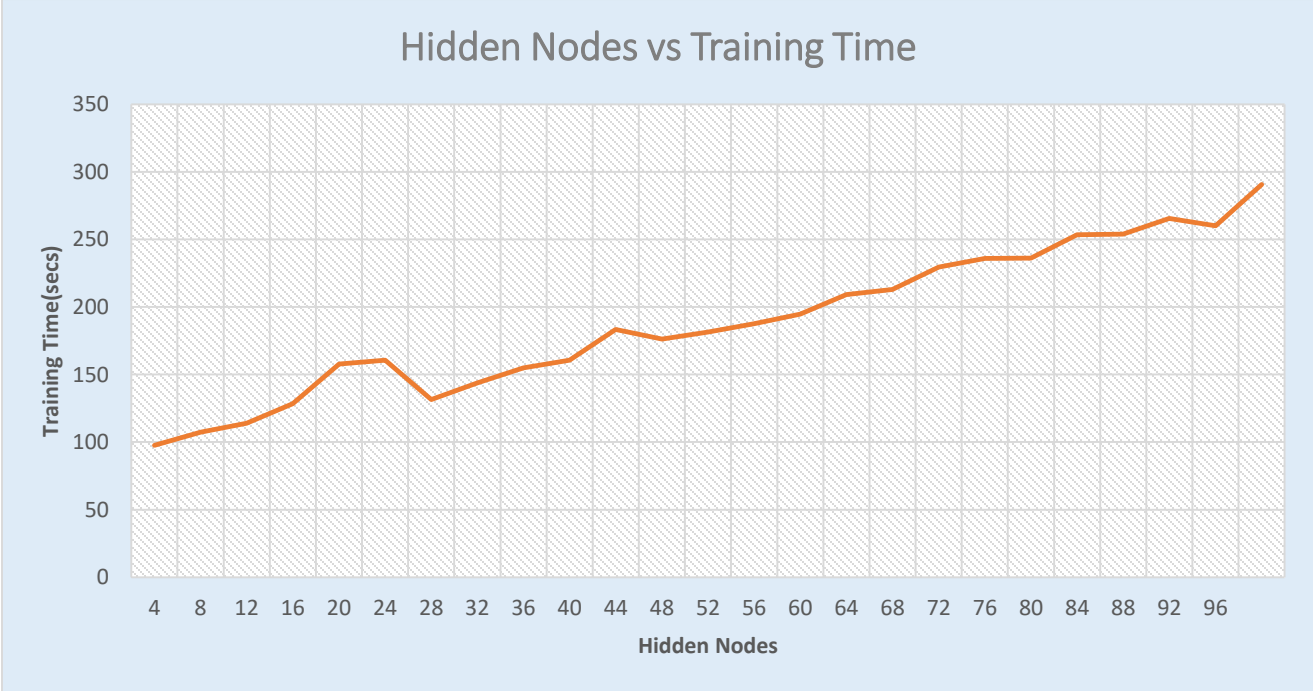


Figure 6-3: Variation of Training Time for different number of Hidden Nodes for maximum Iteration Value of 150

As seen in all of above plots (for different maximum number of iterations of optimization function: 50,100,150), training time of model increases as we increase number of nodes. Also, training time increases as we increase maximum number of iterations of optimization function.

Observation3: *Neural Network Training time with more number of "Maximum Number of Iterations in optimization function" is more:* *This is implicit as we loop in optimization function for longer duration to achieve better convergence.*

Observation4: *Training time grows as Number of Hidden units and Number of hidden layers' increase:* Too many hidden units may lead to the slow training phase. This is true as more number of hidden units imply greater size of hidden layer and output weight matrix and hence add complexity to the system. Hence any Minimization/Optimization algorithm will need more time in computing these increased weights and spitting out converge weight matrix with low error function value.

7. Impact of Regularization coefficient on accuracy

We use regularization in Neural Network to avoid overfitting problem. That is the learning model is best fit with the training data but give poor generalization when test with validation data.

Purpose of any Neural Network unit is to perform with good accuracy on TEST DATA once the model is exposed to outside world. Training data accuracy though important but is of practically no importance to a User as true labels of training data are known beforehand. Hence, Regularization coefficient (λ) is tucked in to solve the overfitting problem in statistical model by controlling the magnitude parameters in Neural Network. Hence, by choosing correcting value of Regularization coefficient (λ), we can avoid overfitting and under fitting problem.

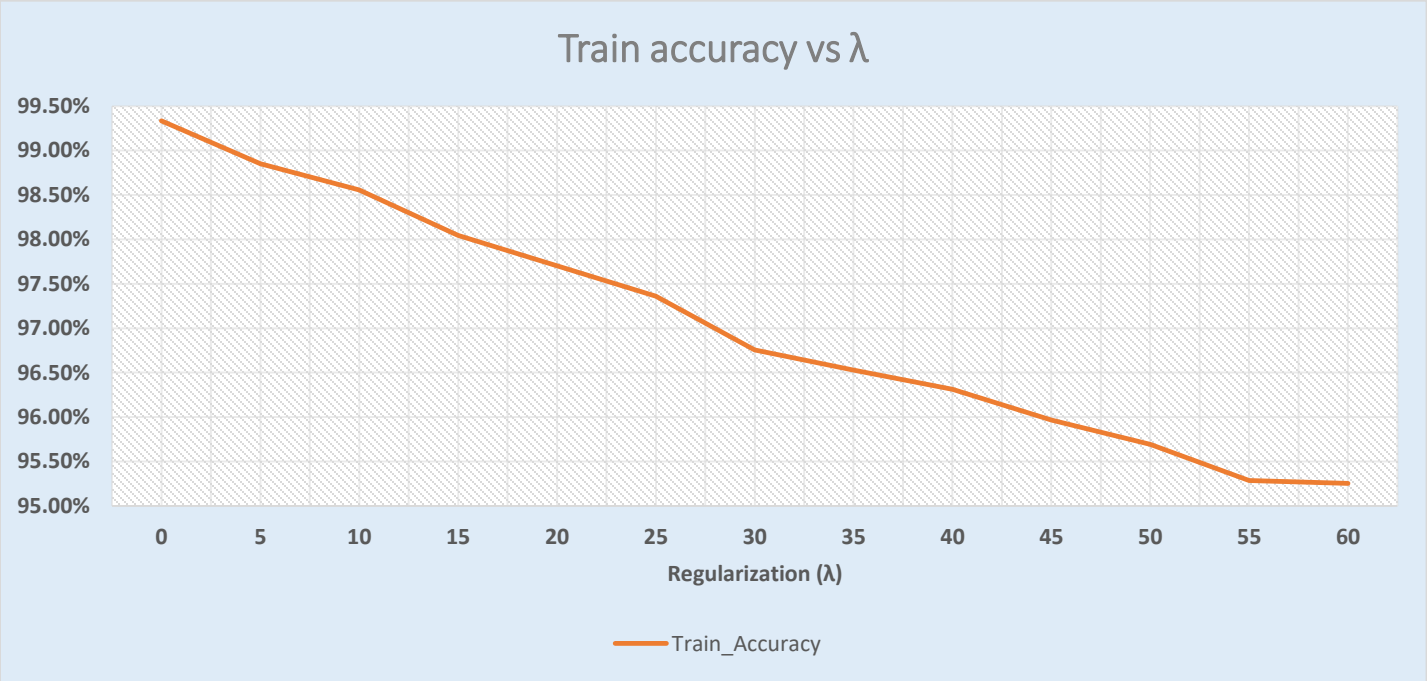


Figure 7-7-1: Variation of Training Accuracy with Regularization coefficient

As expected, Regularization coefficient (λ) is tucked in to solve the overfitting problem by controlling the magnitude parameters in Neural Network. Hence, weights computed in training phase are slightly modified by Regularization factor. Hence, this lowers training accuracy.

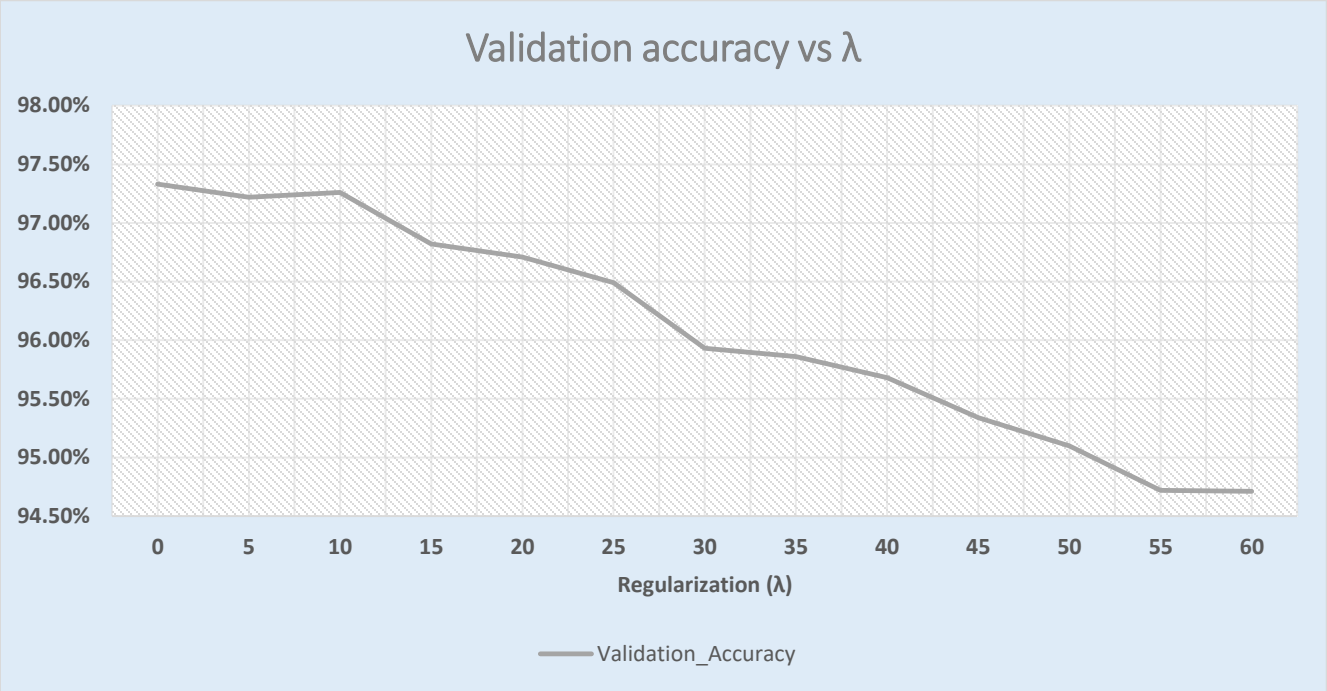


Figure 7-7-2: Variation of Validation Accuracy with Regularization coefficient for 150 maximum Iterations

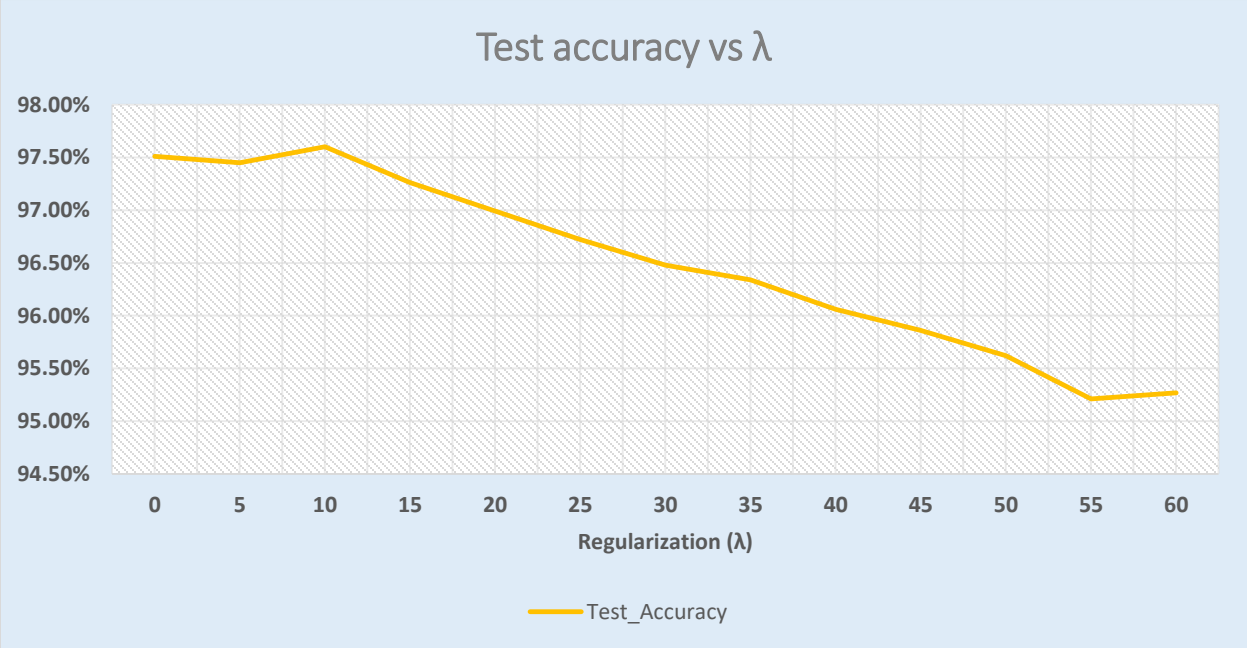


Figure 7-3: Variation of Test Accuracy with Regularization coefficient for 150 maximum Iterations

Ideally and theoretically we expected Validation and Test accuracy to increase with increase of regularization coefficient. However, observations as above are quite opposite to that. It appears that training, validation and testing data are closely related and hence applying more regularization coefficient backfires in this experiment.

HENCE, WE HAVE CHOOSEN LOWER VALUE OF LAMBDA (5) IN THE FINAL SCRIPT.

Observation5: *Neural Network predictive accuracy on VALIDATION and TRAINING DATA decreases as value of Regularization coefficient (λ) increases*

Regularization coefficient (λ) in error function is introduced to allow for better performance of Neural Network on Unseen data. Hence, this involves slightly changing original weight values which were updated based on training data examples. This slight change in weights causes training accuracy to go down as Regularization coefficient (λ) is increased.

Observation6: *Neural Network predictive accuracy on TEST DATA is expected to increase* ***theoretically*** *as value of Regularization coefficient(λ) is increased.* However, However, observations as above are quite opposite to that. It appears that training, validation and testing data are closely related and hence applying more regularization coefficient backfires in this experiment. However, for few values of Hidden nodes, observations are quite similar to expected, i.e., validation and test accuracy increases with regularization coefficient.

8. Comparative Analysis of Single Hidden layer implementation and Deep Neural Network on the CelebA data set

8.1 Performance of Implemented algorithm on face Dataset

We used the same forward feed and back propagation algorithm/code for CelebA dataset and below are the observations:

Training set Accuracy	88.6682464455%
Validation set Accuracy	87.8424015009%
Test set Accuracy	88.1907645723%
Evaluation time	372.58 (Start time: 17:10:08, finish time: 17:03:56)

8.2 Deep Neural Network Performance on face Dataset

A hidden layer can consist of number of neural network units. To interleave weights more closely and to achieve higher performance in terms of predictive analysis, we can introduce number of hidden layers instead of just once.

Intuitively, above implementation though increases predictive analysis performance but makes Neural network more complex and hence it takes more time to adjust weights of neural units.

These observations are seen in below table which show variation of execution time and Accuracy with Number of hidden Layer.

Number of Hidden Layers	Execution Time (Seconds)	Accuracy
2	361.96	78.539%
3	287.06	77.2521%
5	340.21	75.7759%
7	354.57	74.9054%

***Observation6**: In general, the accuracy of Neural Network tends to increase for increase in Neural Network units. But, after certain threshold addition of many hidden nodes can lead to decreasing accuracy. This is seen above. Hence, accuracy reduces with increase in number of hidden layers.*

Comparing Section 8.1 and 8.2, we present below conclusion:

***Simulation Result1:** Our implemented algorithm of Single Layer Forward feed and backward propagation surpasses Tensorflow library in terms of accuracy percentage. Our implementation algorithm gives accuracy of 88.1907645723% on test data whereas Tensorflow library gives best accuracy of 78.539% with 2 Hidden Layers.*

***Simulation Result2:** Tensorflow Implementation performs better than our implementation of forward feedback and backward propagation in terms of training time, though time difference is not very significant.*

9. Final Script Configuration

9.1 Increased Number of weights help to capture pattern of slightest variation of Input data. Additional hidden neural units can provide a better modeling of the data set through the additional weights corresponding to additional nodes added.
HENCE, WE HAVE CHOOSEN NUMBER OF NODES AS HIGH AS 100 IN THE FINAL SCRIPT.

9.2 Ideally and theoretically we expected Validation and Test accuracy to increase with increase of regularization coefficient. However, observations as above are quite opposite to that. It appears that training, validation and testing data are closely related and hence applying more regularization coefficient backfires in this experiment.
HENCE, WE HAVE CHOOSEN LOWER VALUE OF LAMBDA (5) IN THE FINAL SCRIPT.

10. Summary and Conclusion

Observation1: *Neural Network predictive accuracy increases as Number of Hidden units and Number of hidden layer increase.*

Observation2: *Training, Validation and test accuracy Increases as we increase "maximum number of iterations" parameter of Optimization function*

Observation3: *Neural Network Training time with more number of Maximum Number of Iterations in optimization function is more.*

Observation4: *Training time grows as Number of Hidden units and Number of hidden layers' increase.*

Observation5: *Neural Network predictive accuracy on TRAINING DATA decreases as value of Regularization coefficient (λ) increases*

Observation6: *Neural Network predictive accuracy on VALIDATION and TEST DATA is expected to increase* ***theoretically*** *as value of Regularization coefficient(λ) is increased.* However, However, observations as above are quite opposite to that. It appears that training, validation and testing data are closely related and hence applying more regularization coefficient backfires in this experiment. However, for few values of Hidden nodes, observations are quite similar to expected, i.e., validation and test accuracy increases with regularization coefficient.

Observation7*: In general, the accuracy of Neural Network tends to increase up for increase in Neural Network units. But, after certain threshold addition of many hidden nodes can lead to decreasing accuracy.*

Simulation Result1: *Our implemented algorithm of Forward feed and backward propagation surpasses Tensorflow library in terms of accuracy percentage. Our implementation algorithm gives accuracy of 88.1907645723% on test data whereas Tensorflow library gives best accuracy of 78.539% with 2 Hidden Layers.*

Simulation Result2: *Tensorflow Implementation performs better than our implementation of forward feedback and backward propagation in terms on training time, though time difference is not very significant.*