OBSERVATIONS:

Neural Network:

First let us take the case of 'XOR'

Input	Output
[1,0]	1
[0,1]	1
[0,0]	0
[1,1]	0

We then, Add gaussian noise (with mean '0' and we can vary Standard deviation) to the input and the output labels to generate more samples

There will be a similar trend in for OR and AND gate

Number of Training samples dependence:

Standard deviation of noise:	**0.001**
NUMBER OF HIDDEN NODES	5
LEARNING RATE	0.1
EPOCHS	300
LOSS FUNCTION	Mean squared error

Case 1:[for one data set ie. with one set of noise]
For 20 training data points, the model gave an overall loss of 0.23. Three out of four predictions were correct with a very slight deviation from the threshold(0.5)
Four test cases given [0,0],[1,0],[0,1],[1,1]
Failed for [0,1]
Case 2:
For 40 training data points, the same model gave an overall loss of 0.11. Four out of four predictions were correct with a very slight deviation from the threshold(0.5)
Four test cases given [0,0],[1,0],[0,1],[1,1]
All outputs are correct
Thus increasing the number of uncorrelated or meaningful data while training will increase the performance of a fixed architecture

NOTE: Retesting with same parameters might result in slightly different loss value since running it again means taking different data set due to added noise in the training data. But the predictions will be same ie. the output will be within the same margin.

Number of nodes in the hidden layer:

Standard deviation of noise:	**0.001**
NUMBER OF TRAINING SAMPLES	40
LEARNING RATE	0.1
EPOCHS	100
LOSS FUNCTION	Mean squared error

Case 1:

5 hidden node:

Loss=0.04
Four test cases given [0,0],[1,1],[0,1],[1,0] with probability of being '1' is 0.2,0.21,0.76,0.81 thus labels are 0 0 1 1 respectively

10 hidden nodes:

Loss=0.0196
Four test cases given [0,0],[1,1],[0,1],[1,0] with probability of being '1' is 0.14,0.17,0.86,0.82 thus labels are 0 0 1 1 respectively

Better outputs than before since loss was less

This way we keep increasing the number of hidden nodes till

50 hidden nodes:
From the values obtained,
Here clear overfitting happens since the number parameters(250 weights including bias) are way higher than the training data points(40 points).

Thus increasing the number of hidden nodes will benefit only upto a point above which it affects the performance.

Learning Rate:

Learning rate is a measure of the convergence of model towards a optimal local minimum. Lower the Learning rate lower is the convergence of model thus requires more number epoches. Higher learning rate implies that there is chance to miss the optimal local minimum.

For,

Standard deviation of noise:	**0.001**
NUMBER OF TRAINING SAMPLES	40
NUMBER OF HIDDEN NODES	10
EPOCHS	100
LOSS FUNCTION	Mean squared error

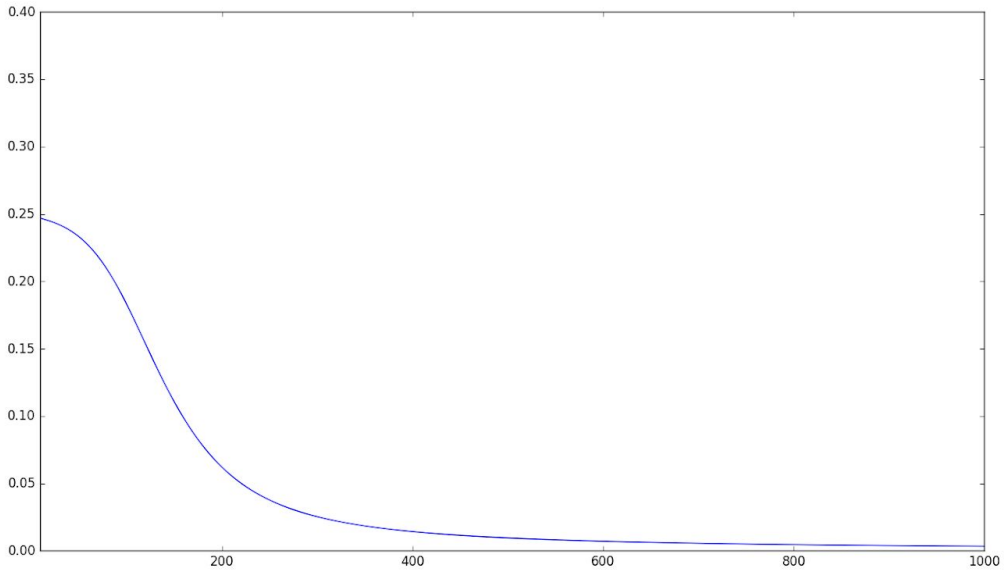
Learning rate=0.1 gave gave good output for 100 epochs (loss=0.0249)

Learning rate=0.01 had a slower convergence towards the optimal local minimum requires more epochs or iterations to get the same value of loss as above
(loss at the end of 100th epoch is 0.25)
(loss at the end of 500th epoch is 0.15) so on….

Learning rate=1 gave bad output since the value was way high

Increasing the number of epochs

Standard deviation of noise:	**0.001**
NUMBER OF TRAINING SAMPLES	160
NUMBER OF HIDDEN NODES	4
LEARNING RATE	0.01
LOSS FUNCTION	Mean squared error



Thus we can see that the loss keeps decreasing for increase in epoch number. Though the rate decreases as we proceed further 400 epochs.