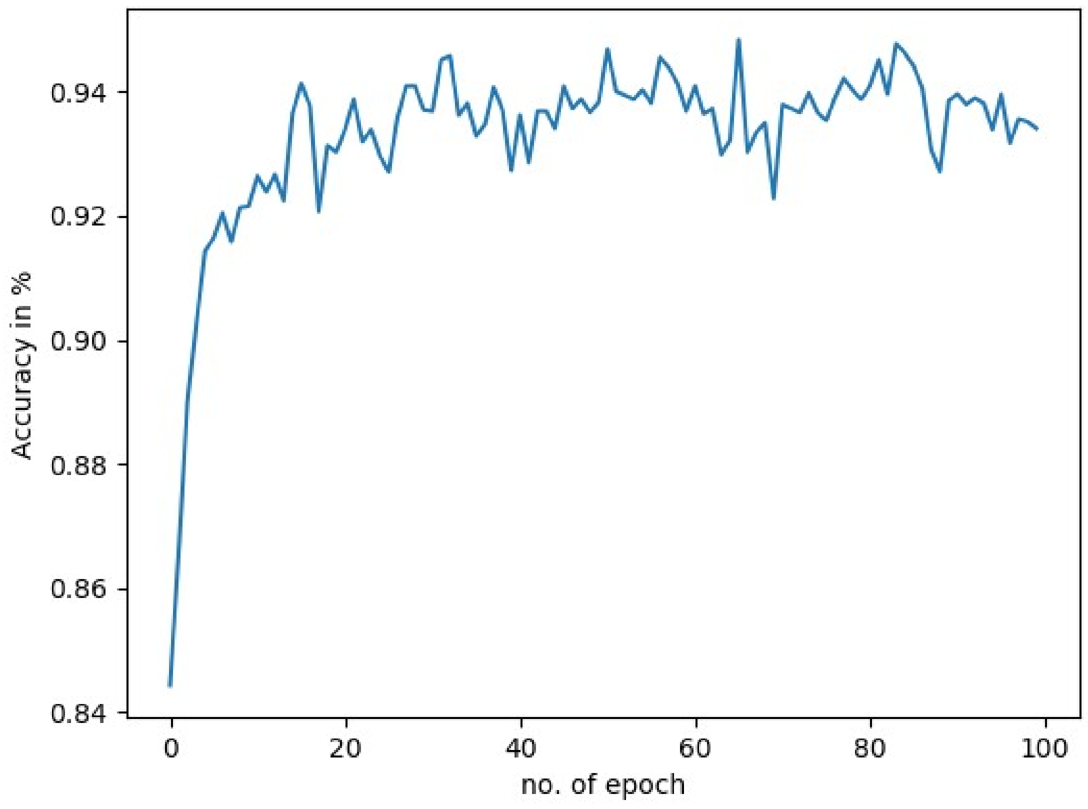
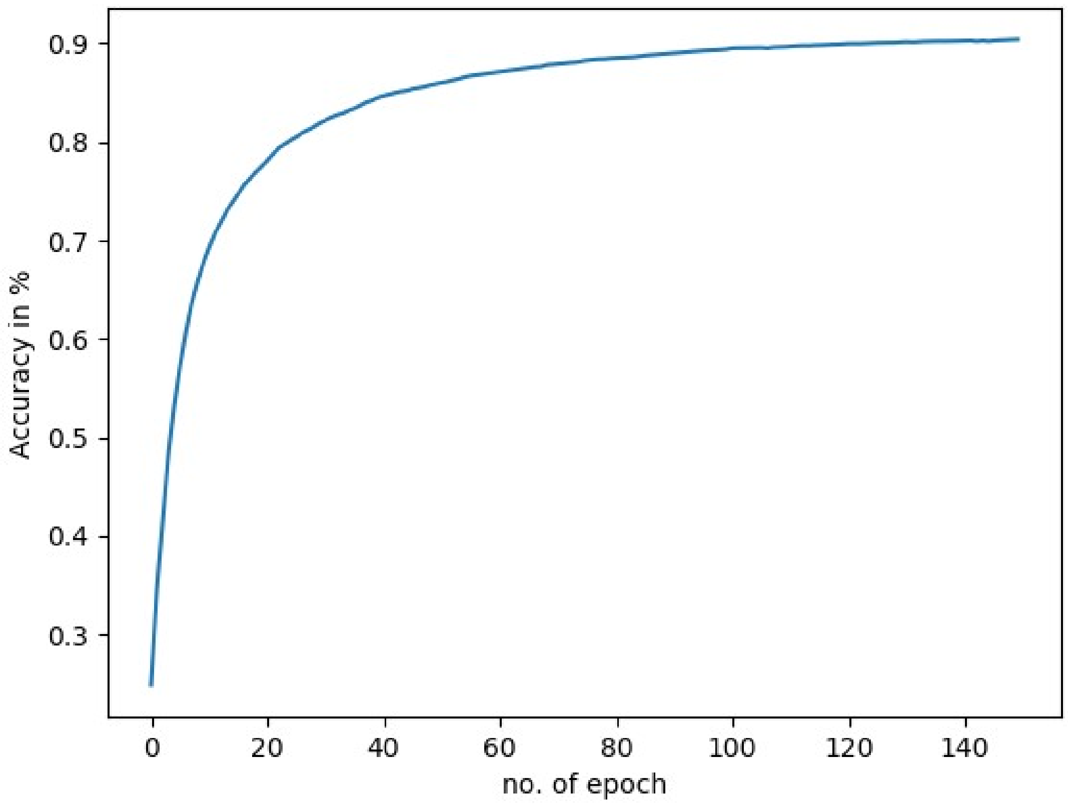
Q1_(a). Max Accuracy is around 94.26%

Where No. of epcoh= 100 and learning rate= 0.01

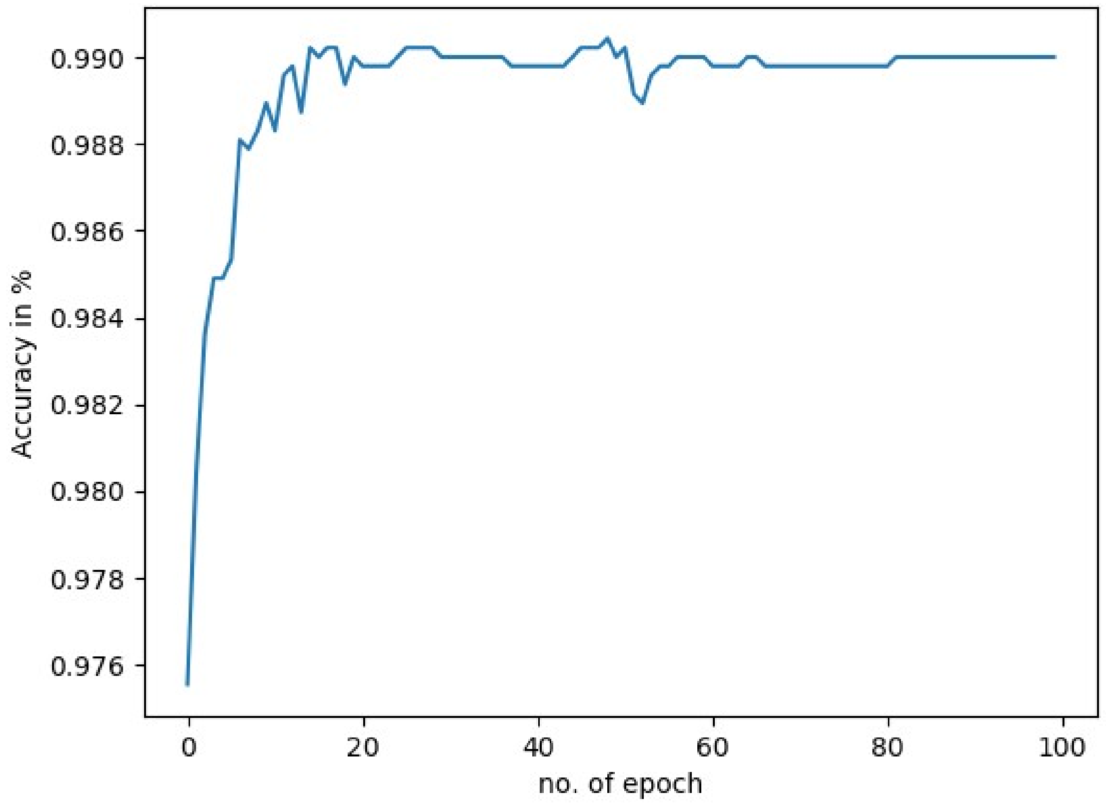


Q1_(b) Max Accuracy is around 90.43%
Where No. of epcoh= 150 and learning rate= 0.001



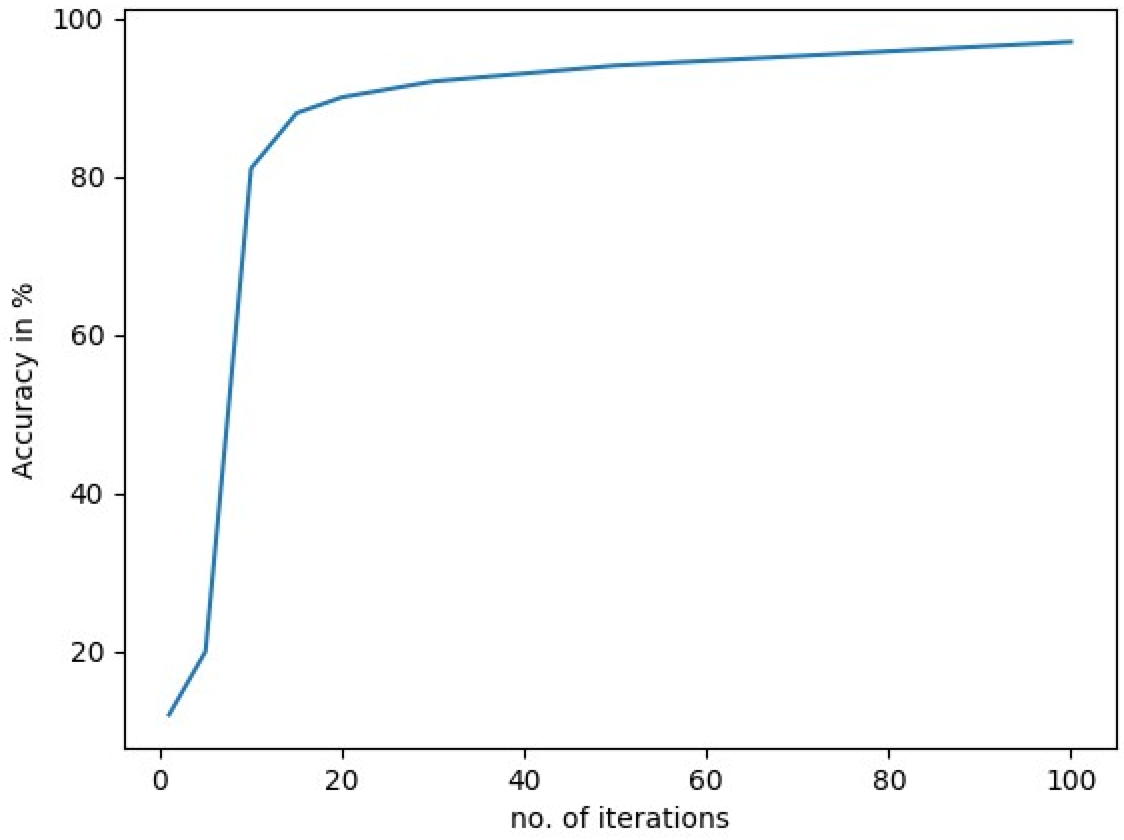
Q1_(c) Using Relu on Q1(a) part, Max Accuracy is around 99.04%

Where No. of epcoh= 150 and learning rate= 0.001

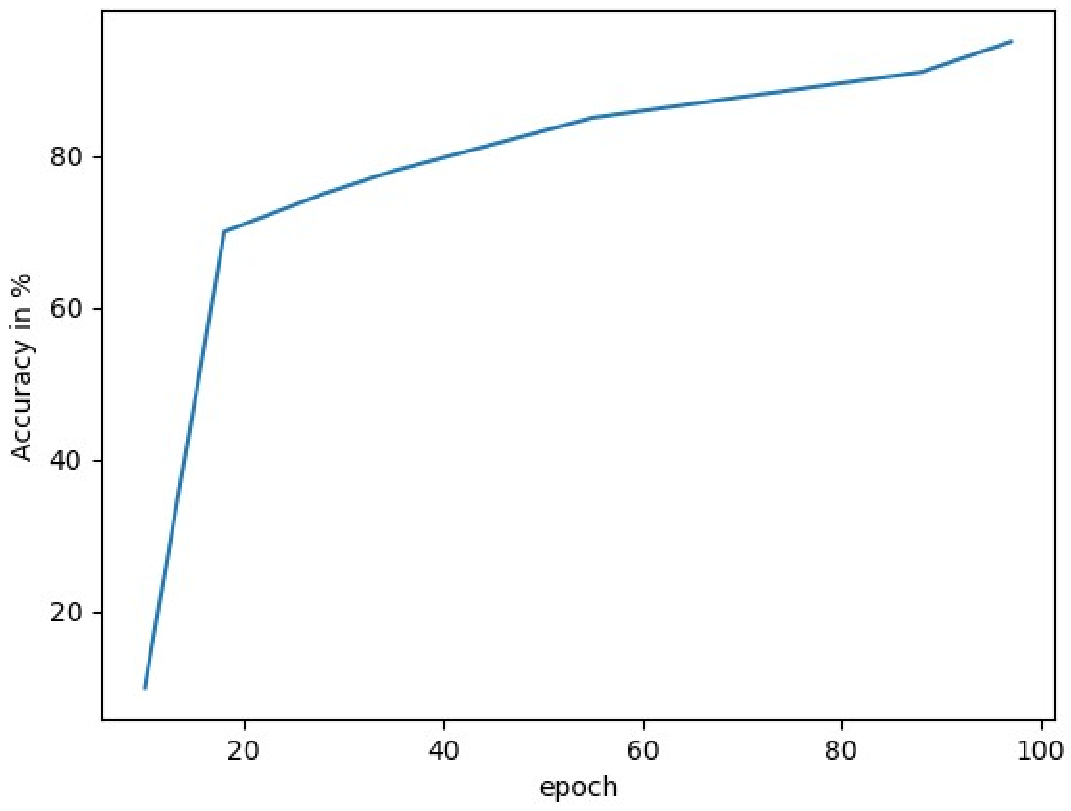


Q1_(c) Using Relu on Q1(b) part, Max Accuracy is around 94.24%

Where No. of epcoh= 100 and learning rate= 0.001

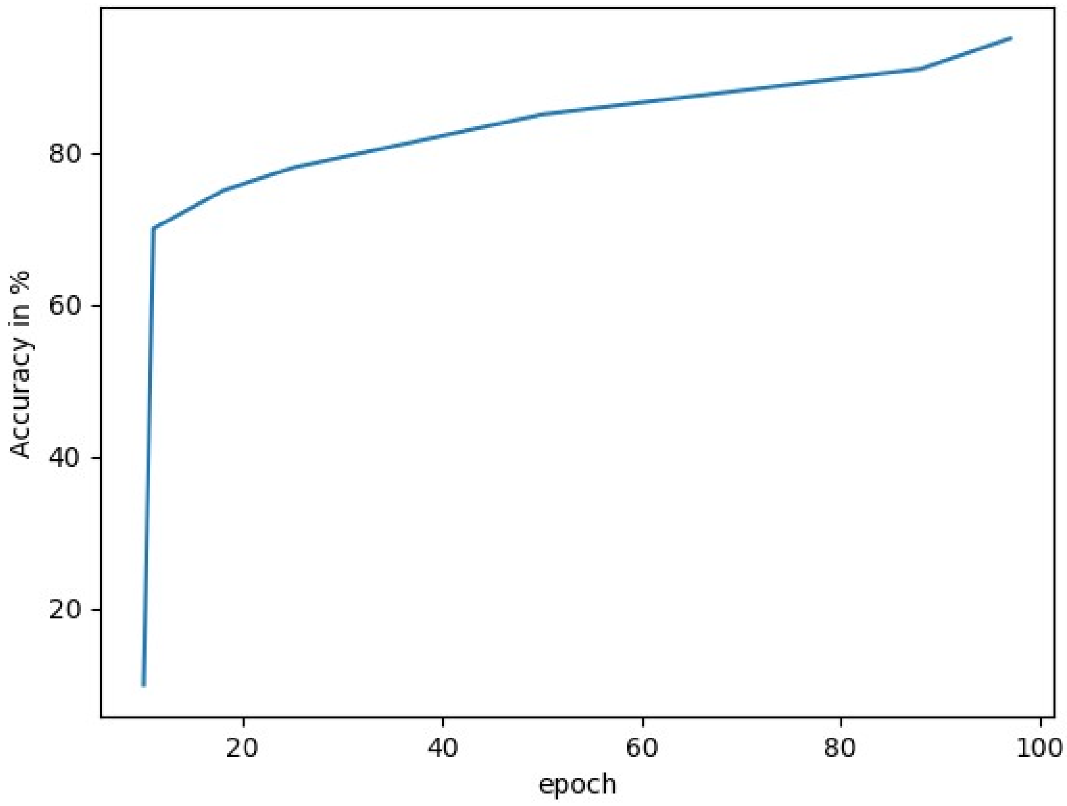


Q1_(c) Using Mxout a on Q1(a) part, Max Accuracy is around 98.04%
Where No. of epcoh= 100 and learning rate= 0.001

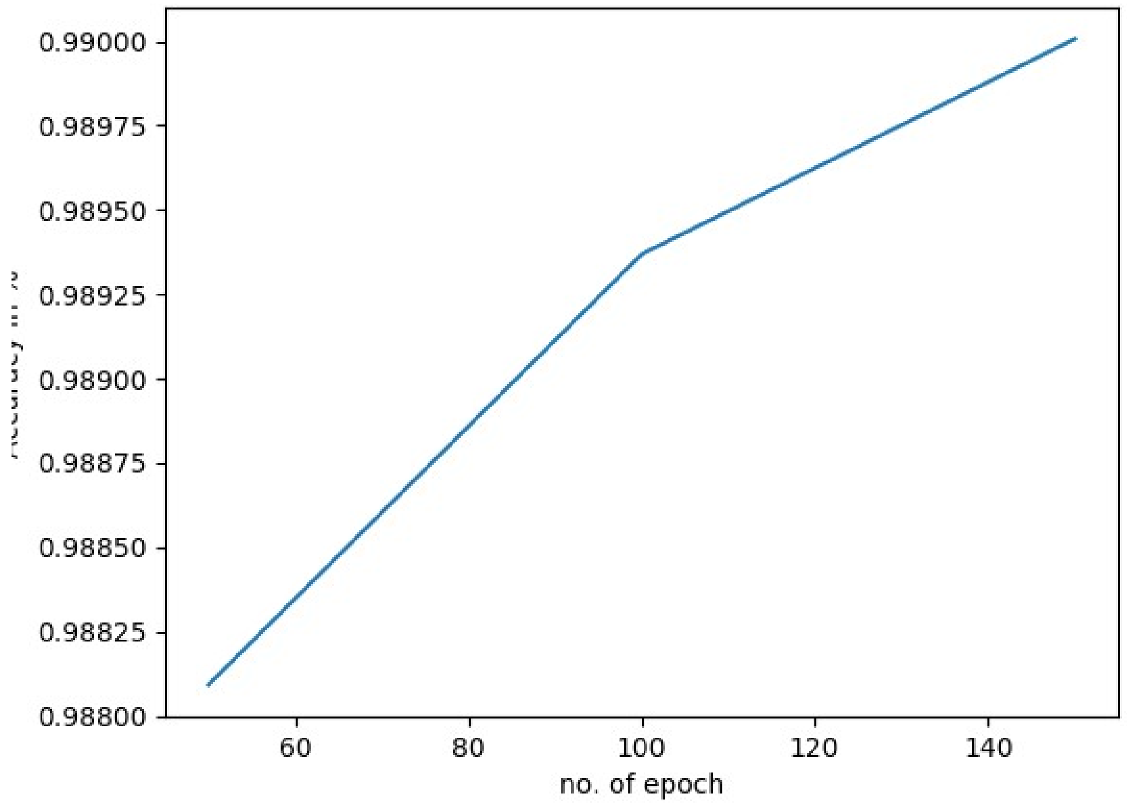


Q1_(c) Using Relu on Q1(b) part, Max Accuracy is around 94.24%

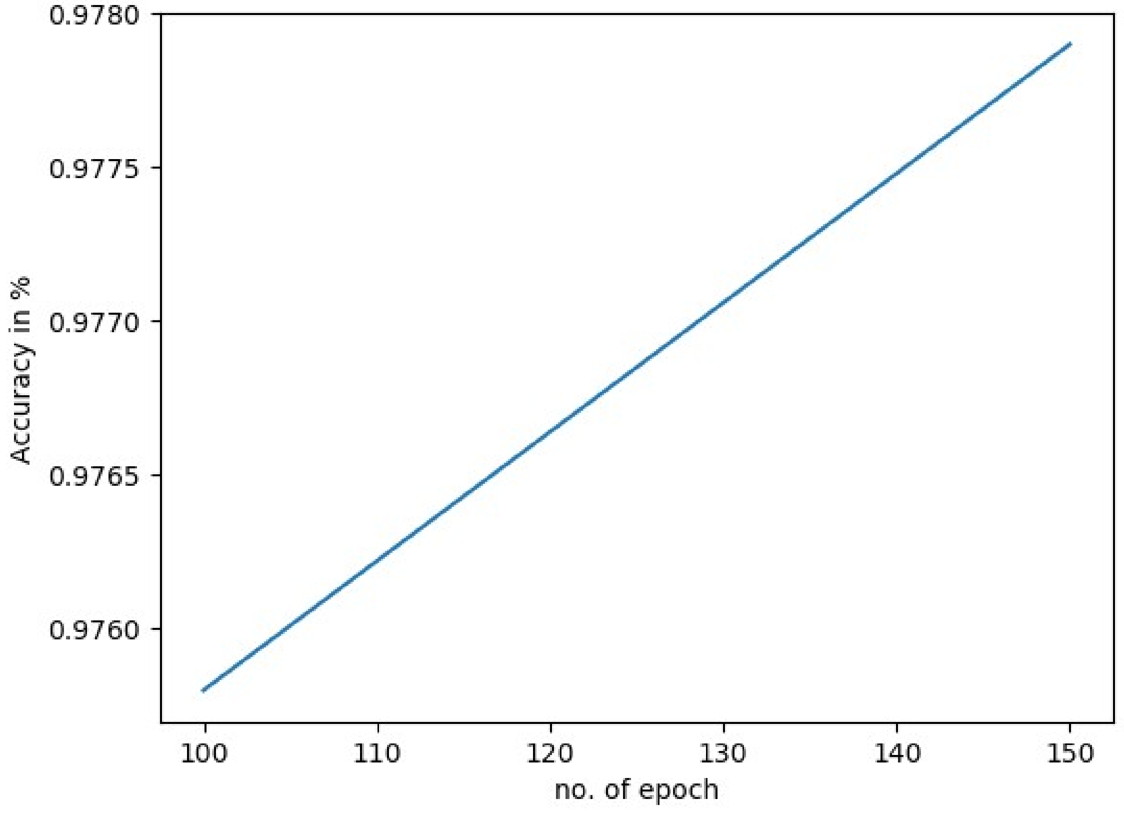
Where No. of epcoh= 100 and learning rate= 0.001



Q2 (a) Below graph in which no. of epoch is[50,100,150] and Accuracy is around 99.006%
Using gridSearch at learning rate = 0.001 and max_iteration = 150, I got max accuracy
I think accuracy difference due to taking different batch size for the input array in q1_a and learning rate also lead accuracy difference, I applied learning rate [0.01,0.1] using grid search which show me the 0.01 giving best accuracy.



Q2 (b) Below graph in which no. of epoch is[100,150] and Accuracy is around 97.79%
Using gridSearch at learning rate = 0.001 and max_iteration = 150, I got max accuracy
difference in accuracy due to learning rate and batch size, number of iterations, batch size.

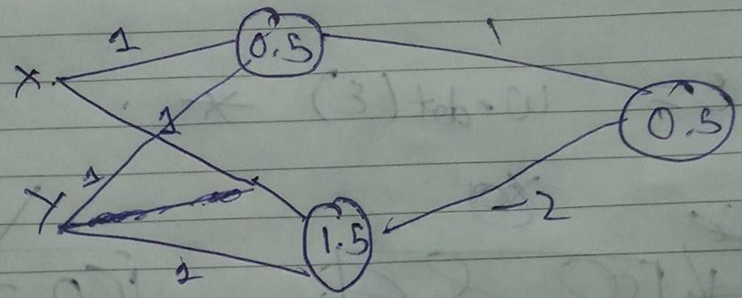


Q3:
Applying epoch =150 and learing =0.01, and by changing activation I got max accuracy using logistic and hidden layer=[200 100 50], and variable learning rate it will decay when go more number of itteration I got 99.4% accuracy.

Q4

M	T	W	T	F	S	S
31					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

Q1 Yes, using linear activation funct^n be used to model the XOR



0	0	0
0	1	1
1	0	1
1	1	0

⇒ So using these threshold we can model XOR funct^n

$(x \cdot 1 + y \cdot 1) > 0 \rightarrow 1 \times 1$
else $\rightarrow 0 \times 1$
$(x \cdot 1 + y \cdot 1) > 1.5 \rightarrow -2 \times 1$
else $\rightarrow -2 \times 0$

Q3 · In ~~the~~ back-propagation training, while using MSE we have gradient term as (output) * (1-output) which gets smaller and smaller
- the chang in weights gets smaller and smaller and training can stall
but in crossentropy above term is not there so weigh changes don't get smaller and smaller and training is not stall.

Q2 ~~A~~ It works better if x used RELU, I would normalise the data before applying, ~~to the~~ and the weight and bias i ~~choosed will normalised~~ randomly it should also be normalised. using (mean = 0, sigma = [illegible]