Task 1
a)
The act of digitizing coordinate values of an image is called sampling.

b)
The act of digitizing the amplitude of an image is called quantization.

c)
A high contrast histogram has a wide range/spread of intensity values with peaks in frequency value.

d)

Pixel intensity	0	1	2	3	4	5	6	7
No. of pixels	1	1	0	1	2	2	4	4
prob	0.0666	0.0666	0	0.0666	0.1333	0.1333	0.2666	0.2666
Cum prob	0.0666	0.1333	0.1333	0.2	0.3333	0.4666	0.7333	1
Scaled by L-1(8-1)	0	0	0	1	2	3	5	7

The old image matrix

7	6	5	6	4
5	4	7	7	0
1	7	6	3	6

New matrix

7	5	3	5	2
3	2	7	7	0
0	7	5	1	5

e)
A log transform will brighten an image. The lower intensity values will see the most significant change in value, while the highest intensity values the least. The dynamic range will therefore be compressed.

f)
First we flipp the kernel left right then up down we then get:

-1	0	1
-2	0	2
-1	0	1

We then apply this filter to the image with a padding of one layer of zeros the image then looks like this

0	0	0	0	0	0	0
0	7	6	5	6	4	0
0	5	4	7	**7**	**0**	**0**
0	1	7	6	**3**	**6**	**0**
0	0	0	0	**0**	**0**	**0**

The new image will then be the same size as the old image

(1,1) = (0*-1) + (0 * 0) + (0* 1) + (0*-2) +(7*0) + (6* 2) + (0*-1) +(5 * 0) +(4 * 1) = 16
(1,2) = (0*-1) + (0 * 0) + (0* 1) + (7*-2) +(6*0) + (5* 2) + (5*-1) +(4 * 0) +(7 * 1) = -2
(1,3) = (0*-1) + (0 * 0) + (0* 1) + (6*-2) +(5*0) + (6* 2) + (4*-1) +(7 * 0) +(7 * 1) = 3
(1,4) = (0*-1) + (0 * 0) + (0* 1) + (5*-2) +(6*0) + (4* 2) + (7*-1) +(7 * 0) +(0 * 1) = -9
(1,5) = (0*-1) + (0 * 0) + (0* 1) + (6*-2) +(4*0) + (0* 2) + (7*-1) +(0 * 0) +(0 * 1) = -19
(2,1) = (0*-1) + (7 * 0) + (6* 1) + (0*-2) +(5*0) + (4* 2) + (0*-1) +(1 * 0) +(7 * 1) = 21
(2,2) = (7*-1) + (6 * 0) + (5* 1) + (5*-2) +(4*0) + (7* 2) + (1*-1) +(7 * 0) +(6 * 1) = 7
(2,3) = (6*-1) + (5 * 0) + (6* 1) + (4*-2) +(7*0) + (7* 2) + (7*-1) +(6 * 0) +(3 * 1) = 2
(2,4) = (5*-1) + (6 * 0) + (4* 1) + (7*-2) +(7*0) + (0* 2) + (6*-1) +(3 * 0) +(6 * 1) = -15

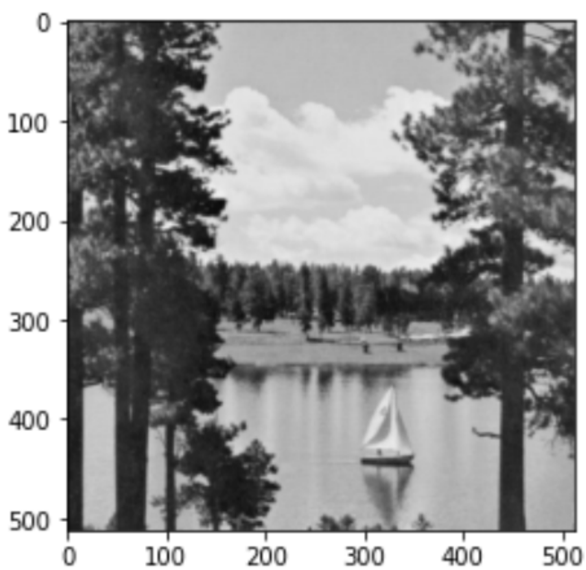
(2,5) = (6*-1) + (4 * 0) + (0* 1) + (7*-2) +(0*0) + (0* 2) + (3*-1) +(6 * 0) +(0 * 1) = -23
(3,1) = (0*-1) + (5 * 0) + (4* 1) + (0*-2) +(1*0) + (7* 2) + (0*-1) +(0 * 0) +(0 * 1) = 18
(3,2) = (5*-1) + (4 * 0) + (7* 1) + (1*-2) +(7*0) + (6* 2) + (0*-1) +(0 * 0) +(0 * 1) = 12
(3,3) = (4*-1) + (7 * 0) + (7* 1) + (7*-2) +(6*0) + (3* 2) + (0*-1) +(0 * 0) +(0 * 1) = -5
(3,4) = (7*-1) + (7 * 0) + (0* 1) + (6*-2) +(3*0) + (6* 2) + (0*-1) +(0 * 0) +(0 * 1) = -7
(3,5) = (7*-1) + (0 * 0) + (0* 1) + (3*-2) +(6*0) + (0* 2) + (0*-1) +(0 * 0) +(0 * 1) = -13

The new image is then:

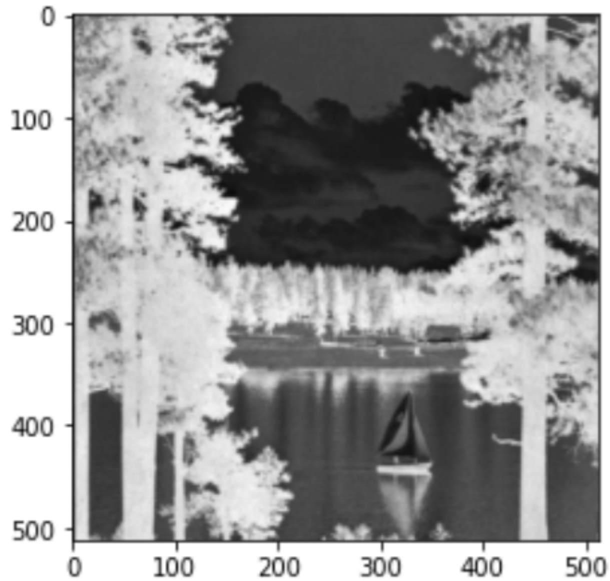
16	-2	3	-9	-19
21	7	2	-15	-23
18	12	-5	-7	-13

Task 2

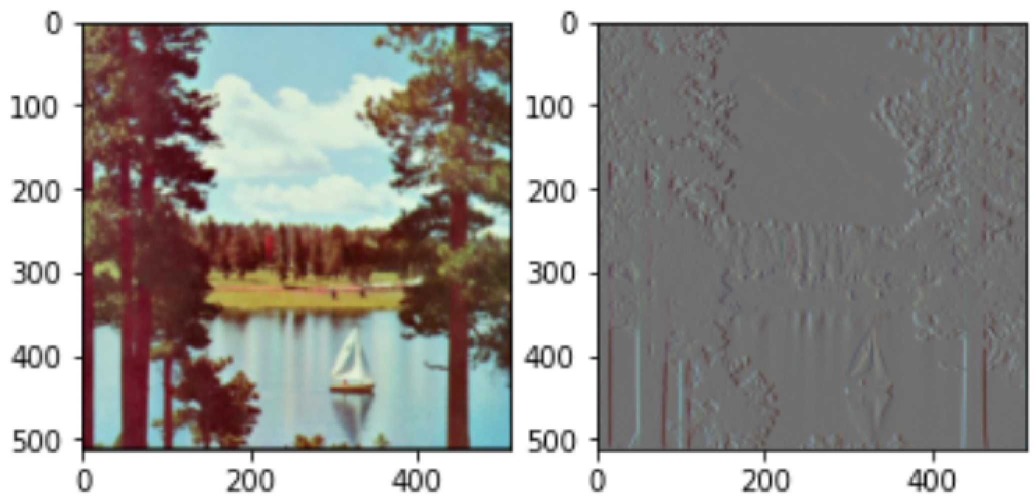
a)



b)

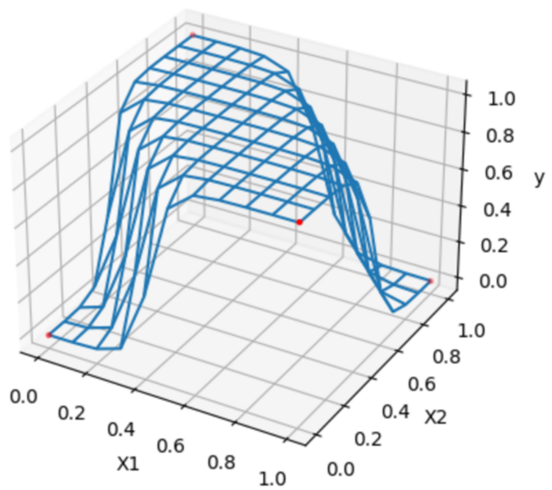


c)



Task 3

a)
XOR cannot be represented by a single layer neural network.



As you can see in this vizual representation of the XOR function it is not possible to represent it as a linear function thus it's not possible to represent it with a single layer neural network.

b)
Hyperparameters are variables that describe the network and its learning parameters. Variables that describe the network could be the number of nodes in a layer, or how many hidden layers in the network. Learning parameters could be learning rate, activation function, and so on.

c)
When we want to classify an object it is desired to know the probability predicted by the network. The softmax provides this by using the input to divide the probabilities to each of the output nodes.

d)
A1 = w1 * x1 = -1 * -1 = 1
A2 = w2 * x2 = 1 * 0 = 0
A3 = w3 * x3 = -1 * -1 = 1
A4 = w4 * x4 = -2 * 2 = -4
C1 = a1 + a2 + b1 = 1 + 0 +1 =2
C2 = a3 + a4 + b2 = 1 + -4 + -1 = -4

y^ = max(C1,C2) = max(2,-4) = 2
C = ½(y -y^)^2=½(1-2)^2 = ½(-1)^2 = ½

y^’ =-y^’ 2*1/2(y - y^) = -1 * (1-2) = 1
C1’ = y^’ * dy^/dc1 = y^’ * d/dc1 * c1 , this comes from y^ = max(c1,c2) = c1 = 1*1 = 1
C2’ = y^’ * dy^/dc2 = y^’*d/c2 *c1, since there is no c1 in c2 then d/c2* c1 = 0, = 1 * 0 = 0
B1’ = C1’ * dc1/db1 = c1’ * d/db1 * c1 = c1’ * d/db1 * (a1 + a2 +b1) = 1*1 = 1

B2’ = C2’ * dc2/db2 = 0* dc1/db2 = 0
A1’ = c1’ * dc1/da1 = c1’ * d/a1 * c1 = c1’ * d/da1 (a1 + a2 + b1) = 1 * 1 = 1
A2’ = c1’ * dc1/da2 = c1’ * d/a2 * c1 = c1’ * d/da2 (a1 + a2 + b1) = 1 * 1 = 1
A3’ = c2’ * dc2/da3 = c2’ * d/a3 * c2 = c2’ * d/a3 (a3 + a4 + b2) = 0 * 1 = 0
A4’ = c2’ * dc2/da4 = c2’ * d/a4 * c2 = c2’ * d/a4 (a3 + a4 + b2) = 0 * 1 = 0
W1’ = a1’ * da1/w1 = a1’ * d/w1 *a1 = a1’ * d/w1 (w1* x1) = a1’ * x1 = 1 * -1 = -1
W2’ = a2’ * da2/w2 = a2’ * d/w2 *a2 = a2’ * d/w2 (w2* x2) = a2’ * x2 = 1 * 0 = 0
W3’ = a3’ * da3/w3 = a3’ * d/w3 *a3 = a3’ * d/w3 (w3* x3) = a3’ * x3 = 0 * -1 = 0
W4’ = a4’ * da4/w4 = a4’ * d/w4 *a4 = a4’ * d/w4 (w4* x4) = a4’ * x4 = 0 * 2 = 0

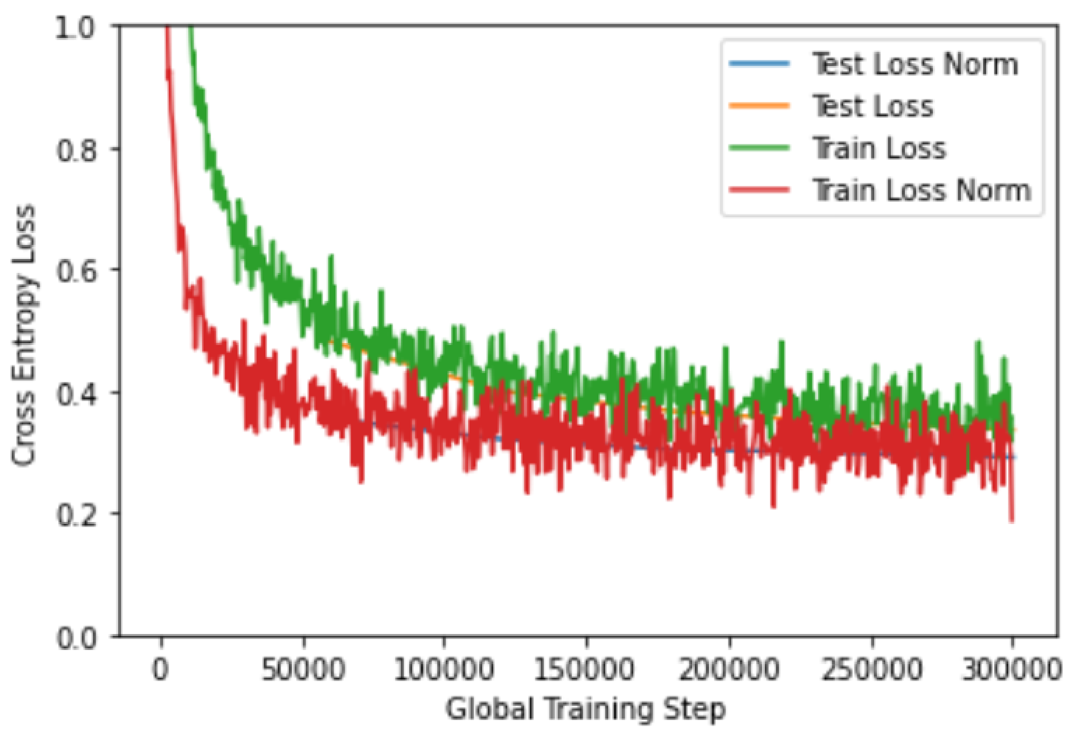
e)

W1_2 = w1_1 - 0.1*w1’ = -1 - 0.1*-1 = -0.9

W3_2 = w3_1 - 0.1*w3’ = -1 - 0.1*0 = -1

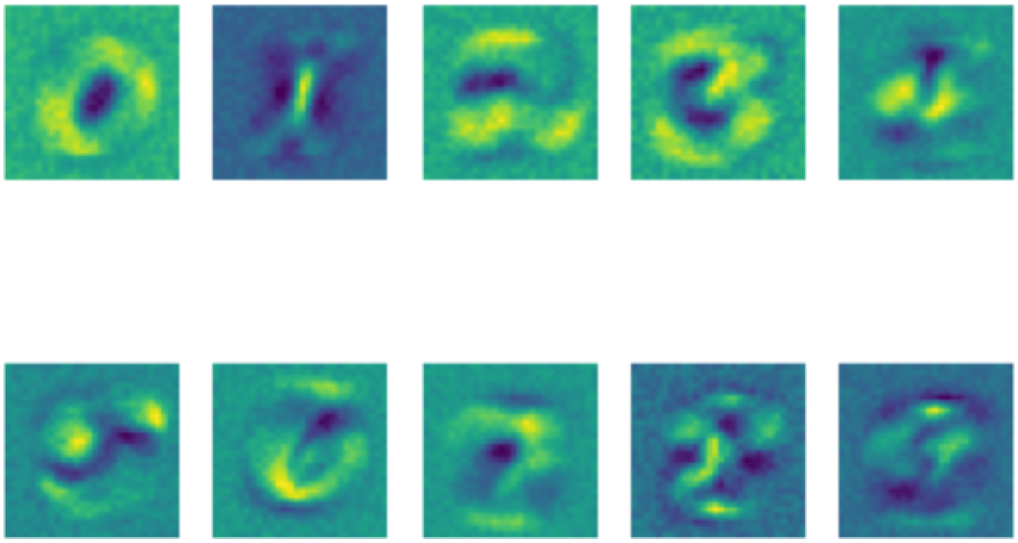
B1_2 = b1_1 -0.1 * b1’ = 1 - 0.1 * 1 = 0.9

Task 4
a)



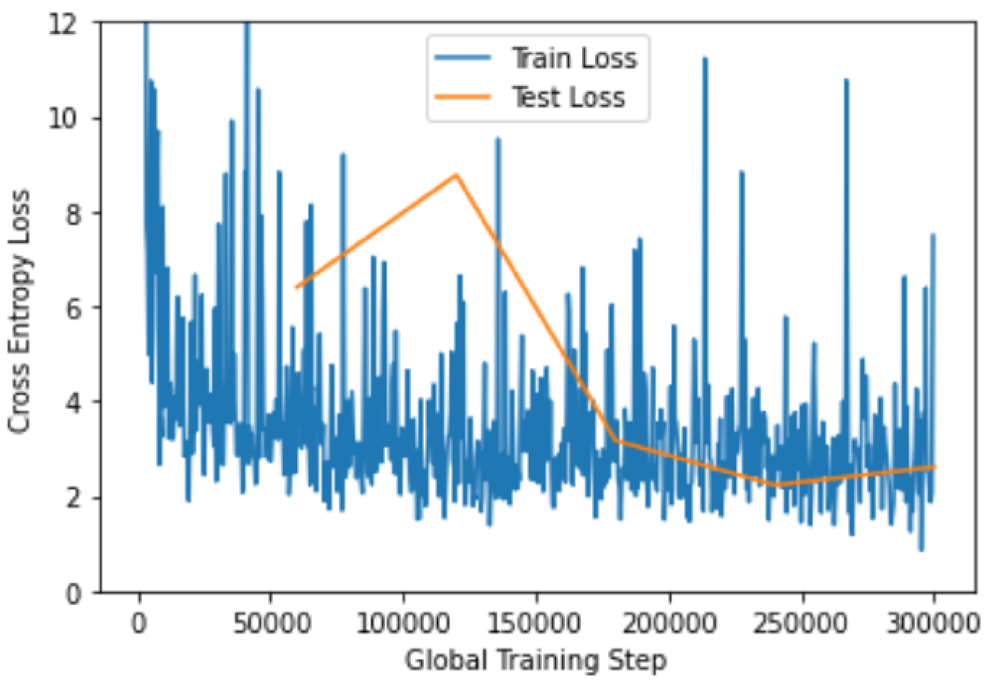
You can tell that the Normalized model gets a lower loss score than the non normalized model, on the same nr of steps.

b)



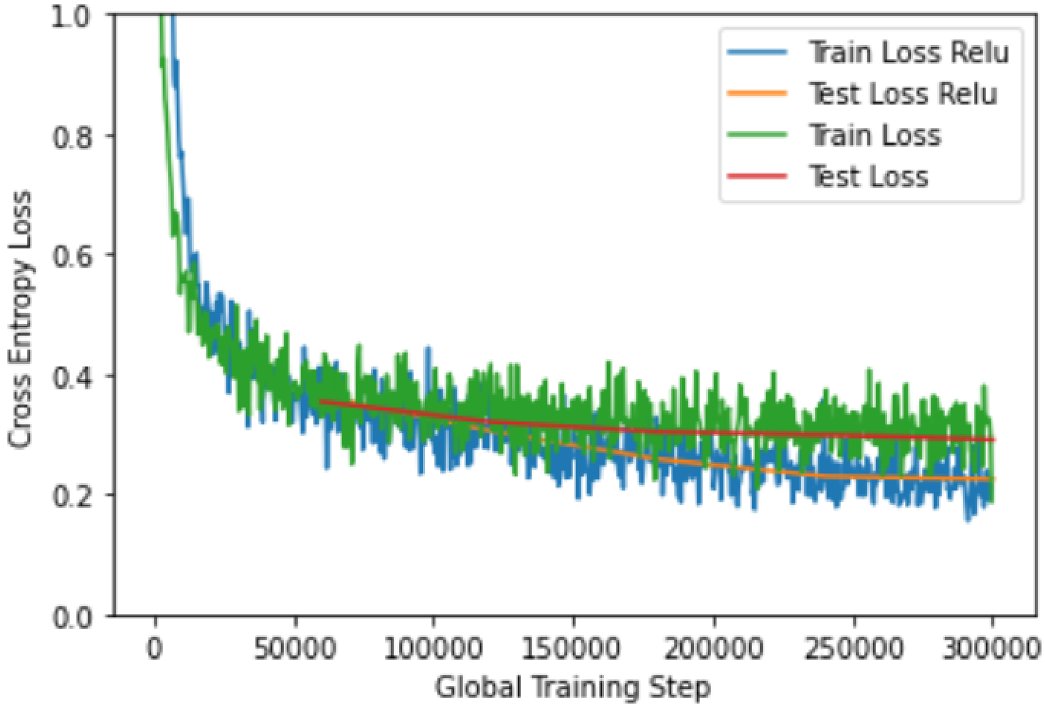
From the weight we can see some similarity to the shapes of the numbers. This is because the network only has 10 nodes which will lead to each node optimizing to calculate the pixel value to its given number.

c)



You can see that the loss is a lot higher and this is because when updating the weights we take a 100% of the rate of change of the weights(i.e wi+1 = wi - (a*wi’, this part is where we then take a 100% of)), which inn turn leads to over shouting of the hunt for the minimum of the function.

d)



We can see here that when adding relu and more nodes we get a better loss then the network from task 4a. Relu takes the max between a value in the network and zero meaning that all values are 0<=. Also more nodes makes it so that there are more weights to optimize and more specificity and variety among the weights in the network giving better results.