Task_3:

1)

a) As binary classification function is used there is only one output 0 or 1

zero for the negative classes and one for the positive classes

The first input points:

Output = W1.X1 + W2.X2 + b = 2 * 2 + -1 * 4 +0.5 = 0.5

∵ 0.5 > 0

∴ The class label =1

The second input points:

Output = W1.X1 + W2.X2 + b = 2 * 4 + -1 * 2 + 0.5 = 6.5

∵ 6.5 > 0

∴ The class label =1

The third input points:

Output = W1.X1 + W2.X2 + b = 2 * 2 + -1 * 2 + 0.5 = 2.5>0

∵ 2.5 > 0

∴ The class label =1

b) To find the decision boundary we need to equal the equation to zero

W1.X1 + W2.X2 + b = 0

2.X1 + -1.X2 + 0.5 = 0

X2 = 2.X1 + 0.5

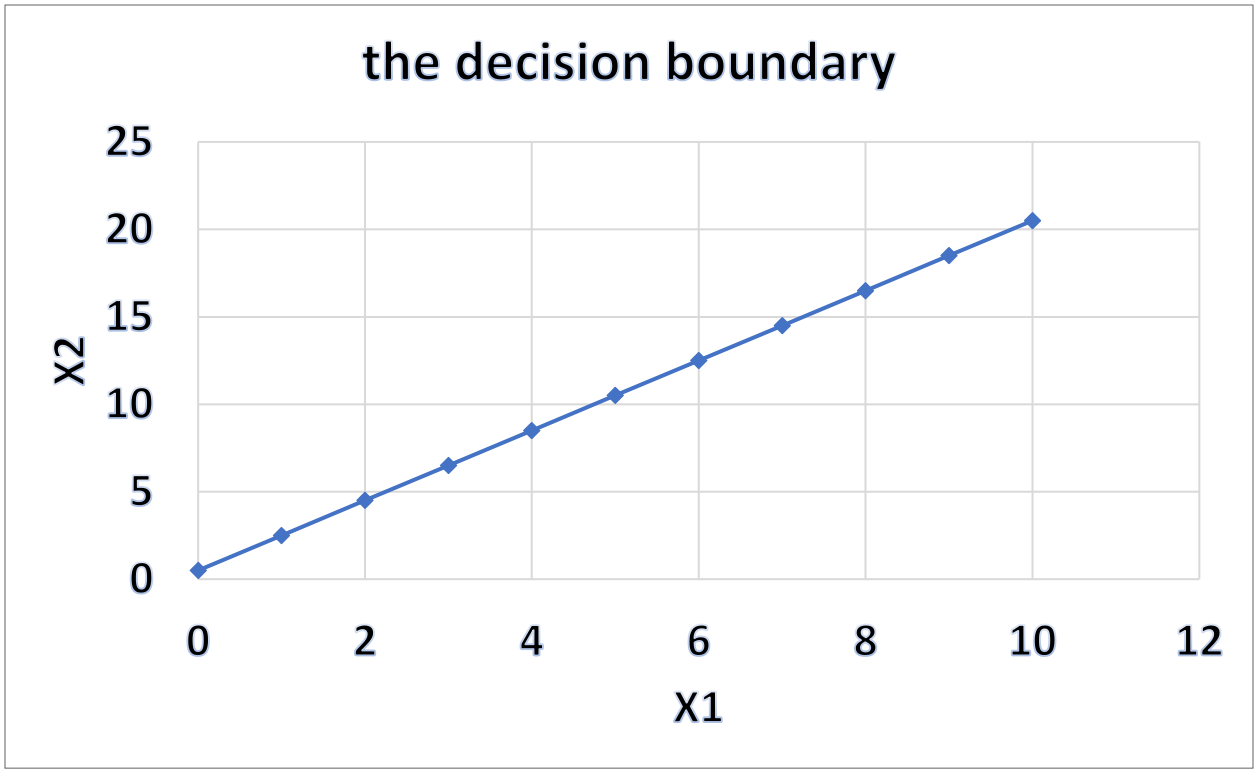

Figure 1.1

c) the limitations of the perceptron algorithm:

1. Only works with linearly separable data

2. Limited to binary classification

3. It is a single layer algorithm which can not model complex functions as it lacks hidden layers

2)

a)

Input layer: 4 neurons

Output layer: 2 neurons

Hidden layers: 12 neurons

Total neurons = 18

b) 4 hidden layers

c)

Learnable Parameters:

1. From input layer to the first hidden layer

Total for this layer = input layer neurons * first hidden layer neurons + first hidden layer neurons

= 4 * 2 + 2 = 10

2. From first hidden layer to second hidden layer

Total for this layer = 2*4 + 4 = 12

3. From second hidden layer to third hidden layer

Total for this layer = 4*2 + 2 = 10

4. From third hidden layer to fourth hidden layer

Total for this layer = 4*2 + 4 = 12

5. From fourth hidden layer to output layer

Total for this layer = 4*2 + 2 = 10

Total learnable parameters = 54

3)

The padded array = [0, 3, 4, 6, 3, 2, 3, 4, 0]

The filter = [1, 2, -1]

Output_1 = 1 * 0 + 2 * 3 + -1 * 4 = 2

Output_2 = 1 * 3 + 2 * 4 + -1 * 6 = 5

Output_3 = 1 * 4 + 2 * 6 + -1 * 3 = 13

Output_4 = 1 * 6 + 2 * 3 + -1 * 2 = 10

Output_5 = 1 * 3 + 2 * 2 + -1 * 3 = 4

Output_6 = 1 * 2 + 2 * 3 + -1 * 4 = 4

Output_7 = 1 * 3 + 2 * 4 + -1 * 0 = 11

The convolved array = [2, 5, 13, 10, 4, 4, 11]

4)

The expected output length = [(L(in) + 2P -K)/S] + 1

K: Kernel or filter size, K = F

P: convolution padding size, P = 0

S: convolution stride, Assume S = 1

- So, the formula without padding: L(out) = L(in) -F +1

The filter here can overlap with the input array when there are enough elements in the input array

- By adding padding into the input array for maintaining a constant output size and making the input size equals to the output size

L(out) = L(in)

L(in) + 2P – F +1 = L(in)

P = 0.5(F-1)

By providing an example

Assume L(in) = 6 & F = 3

P = 0.5(3-1) = 1 (one element of padding on each side)

L(out) = 6+2(1)-3+1 = 6

5)

By letting stride = 1

8	19	1
13	1	16
-10	28	20

6)

Alex Net has 5 convolution layers and 3 fully connected layers as shown in the figure below:

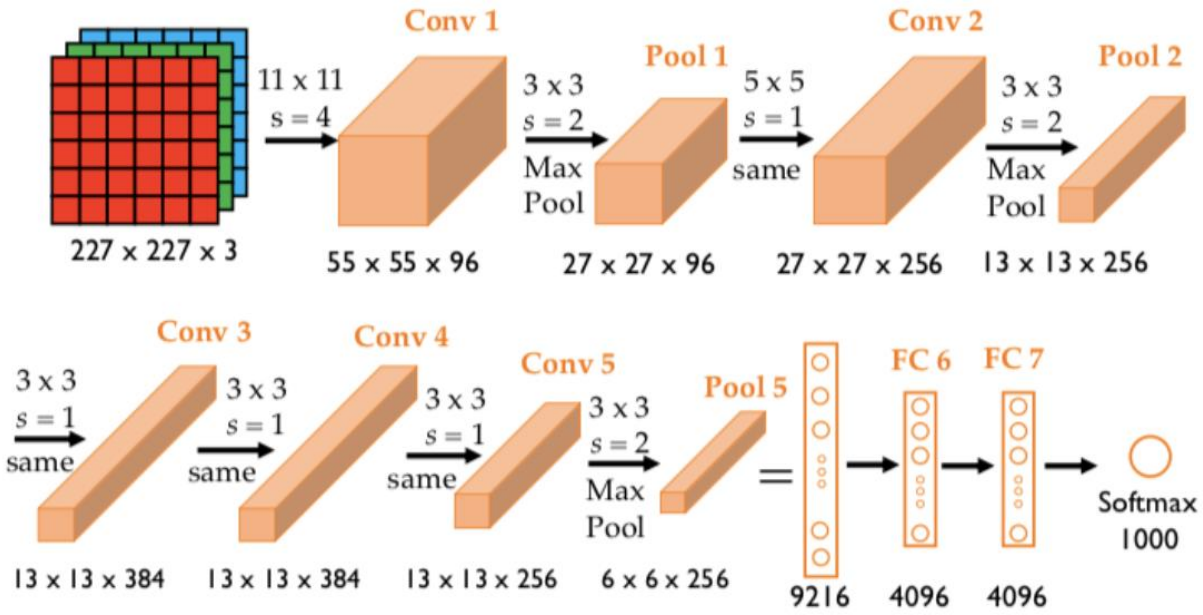

Figure 6.1

Input image shape: 512x512x3

Parameters = (filter width×filter height×number of input channels+1) ×number of Filters

Number of parameters of Conv1 = (11 * 11 * 96 *3 +1) *96 = 34,944

Pool1

Number of parameters of Conv2 = (5 * 5 *96+1) *256 = 614,656

Pool2

Number of parameters of Conv3 = (3*3*256+1) *384 = 885,120

Number of parameters of Conv4 = (3*3*384+1) *384 = 1,327,488

Number of parameters of Conv5 = (3*3*384+1) *256 = 884,992

Pool3

Number of parameters of neurons = 4096

Number of parameters of Fully connected layer1 = (13*13*256+1) *4096

= 177,213,440

Number of parameters of neurons = 4096

Number of parameters of Fully connected layer2 = (4096+1) *4096 = 16,781,312

Number of parameters of neurons = 1000

Number of parameters of Fully connected layer3 = (4096+1) *1000 = 4,097,000

Total parameters = 34,944 + 614,656 + 885,120 + 1,327,488 + 884,992 + 177,213,440 + 16,781,312 + 4,097,000