ECS659P- Stef Abolade - 230985072

In [1]:
```
import numpy as np
```

Question 1a

In [2]:
```
import numpy as np

def coefficient(x):
    return np.array([x, 1])

x_num = np.array([1, 2, 3, 4, 5])
x_transform = np.array([coefficient(x) for x in x_num])

x_transform
```

Out[2]:
```
array([[1, 1],
       [2, 1],
       [3, 1],
       [4, 1],
       [5, 1]])
```

Question 1b

In [12]:
```
X = np.array([[0, 2], [1, -1], [-1, 1]])
W = np.array([[0, 1], [1, 0]])
B = np.array([[1, 2], [1, 2], [1, 2]])

logits = np.dot(X, W) + B

print('The logits matrix is \n',  logits)
```

```
The logits matrix is
 [[3 2]
 [0 3]
 [2 1]]
```

In [4]:
```
Y = np.array([[1, 0], [0, 1], [1, 0]])

predicted= np.argmax(logits, axis=1)
actual = np.argmax(Y, axis=1)

accuracy = np.mean(predicted == actual)

print('the accuracy is' , accuracy)
```

```
the accuracy is 1.0
```

Question 2

In [5]:
```
W1_shape = (10, 128)
B1_shape = (1, 128)
```

```
W2_shape = (128, 256)
B2_shape = (1, 256)
W3_shape = (256, 10)
B3_shape = (1, 10)

W1_parameters= W1_shape[0] * W1_shape[1]
B1_parameters = B1_shape[1]
W2_parameters= W2_shape[0] * W2_shape[1]
B2_parameters = B2_shape[1]
W3_parameters = W3_shape[0] * W3_shape[1]
B3_parameters= B3_shape[1]

total = (W1_parameters + B1_parameters +
                    W2_parameters + B2_parameters +
                    W3_parameters + B3_parameters)

print("W1 shape:", W1_shape)
print("B1 shape:", B1_shape)
print("W2 shape:", W2_shape)
print("B2 shape:", B2_shape)
print("W3 shape:", W3_shape)
print("B3 shape:", B3_shape)

print('Total number of parameters:', total)
```

```
W1 shape: (10, 128)
B1 shape: (1, 128)
W2 shape: (128, 256)
B2 shape: (1, 256)
W3 shape: (256, 10)
B3 shape: (1, 10)
Total number of parameters: 37002
```

Question 3

In [6]:

```
A = np.array([
    [[1, 2, 3], [3, 2, 1], [1, 3, 2]],
    [[2, 1, 3], [1, 2, 3], [3, 2, 1]]
])

B = np.array([
    [[1, 0], [0, 1]],
    [[0, 2], [2, 0]]
])

output = (A.shape[1] - B.shape[1] + 1, A.shape[2] - B.shape[2] + 1)
C = np.zeros((output[0], output[1]))

for i in range(output[0]):
    for j in range(output[1]):
        C[i, j] = np.sum(A[:, i:i+B.shape[1], j:j+B.shape[2]] * B)

print('The output of image C is \n', C)
```

```
The output of image C is
 [[ 7. 13.]
 [16. 14.]]
```

Question 3b

In [7]:
```
#image shape
input_shape = (3, 32, 32)

#1st Convolutional Layer
conv1 = (16, 32, 32)

#1st Pooling Layer
pool1 = (16, 16, 16)

#2ndConvolutional Layer
conv2 = (64, 16, 16)

#2nd Average Pooling Layer
pool2 = (64, 8, 8)

# Fully Connected Layer
fc_input = 64 * 8 * 8
fc_output = 10

print('The shape of the output image of each of the first four steps are \n:', conv1,
```

```
The shape of the output image of each of the first four steps are
: (16, 32, 32) (16, 16, 16) (64, 16, 16) (64, 8, 8)
```

Question 4

In [13]:
```
B = np.array([[4, 6], [6, 8], [4, 8], [6, 6]])

gamma = np.array([1, 1])
beta = np.array([0, 0])

mean = np.mean(B, axis=0)
sd = np.std(B, axis=0)

normalised = (B - mean) / sd * gamma + beta

print('The mean vector is:', mean, '\n The standard deviation vectir is:', sd, '\n The
```

```
The mean vector is: [5. 7.]
 The standard deviation vectir is: [1. 1.]
 The output batch is:
 [[-1. -1.]
 [ 1.  1.]
 [-1.  1.]
 [ 1. -1.]]
```

Question 4b

In [14]:
```
def gradient_loss(w):
    return 2 * w

w = np.array([2.0, 4.0])
lr = 0.25
```

```
momentum  = 0.5
velocity = np.array([0.0, 0.0])

points = [w.copy()]
for _ in range(2):
    g = gradient_loss(w)
    velocity = momentum  *velocity - lr * g
    w += velocity
    points.append(w.copy())

print('The initial weight vector is:', points[0], 'The next 2 points are:', points[1],
```

```
The initial weight vector is: [2. 4.] The next 2 points are: [1. 2.] and [0. 0.]
```