MCA SEM V Minor

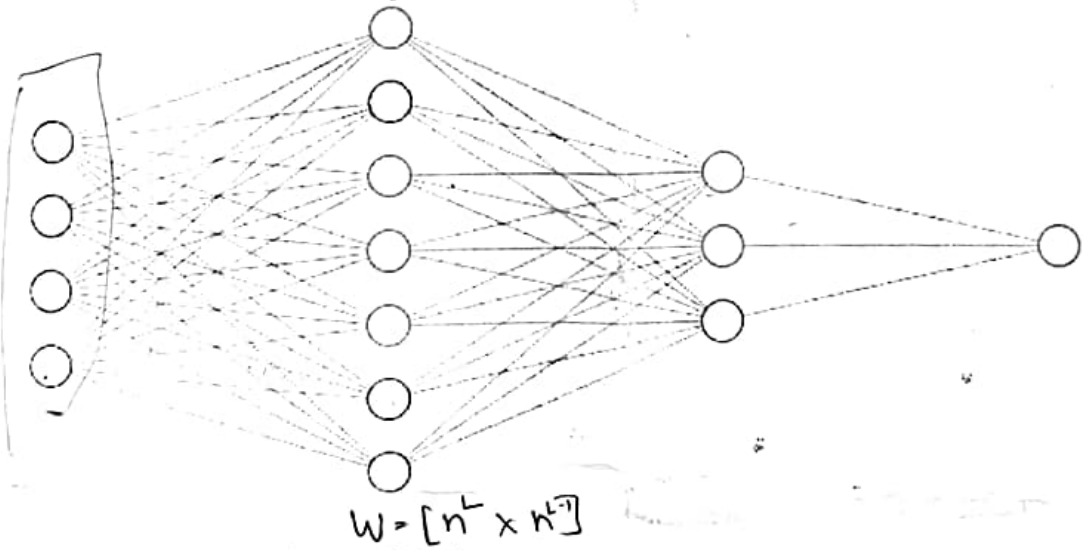
Time: 1 Hour **Max. Marks: 40**

1. For each of the following cases, address the questions w. r. t. the neural network and justify the answer. 12

   Case 1: For the hidden layers, which of the two activation functions: *sigmoid* and *tanh* is preferable?
   Case 2: For a classifier that distinguishes between man vs animal, which of the four activations should be employed at the output layer: Sigmoid, Leaky Relu, Elu and tanh.
   Case 3: For a network that predicts the probability of occurrence of the disease out of six diseases, which of the four activation functions should be employed: Sigmoid, Relu, softmax, and tanh.
   Case 4: Which initialization of weights and biases will result in all the neurons in the layer in consideration learn the same function even after several epochs of the gradient descent backpropagation algorithm?
   Case 5: Considering a sample space comprising 1 million instances, should 70-30 split for training-test partition be preferred?
   Case 6: Considering the following computation, determine the shape of B.

```
A = np.random.randn(4,3)
B = np.sum(A, axis = 1, keepdims = True)
```

2. Consider the following neural network, comprising four layers, called layer 1, layer2, layer3, and layer4. $W^{[i]}$ and $b^{[i]}$ denote the weights and biases of layer `i`. 8



$W = [n^{L} \times n^{L-1}]$

Determine the dimensions of following arrays:

$W^{[1]}$ $b^{[1]}$ $W^{[2]}$ $b^{[2]}$ $b^{[3]}$ $W^{[3]}$ $b^{[4]}$ $W^{[4]}$

For the above neural network, determine each of the following pieces for code, whether it will correctly initialize the network parameters. The components of the vector **`dimLayers`** denote the number of neurons in the respective unit. 8

a)
```
import np.random.randn as
npRand N = len(dimLayers)
    for(i in range(1,N)):
            parameter['W' + str(i)] = npRand (dimLayers[i], dimLayers[i - 1])) *
    0.01
            parameter['b' + str(i)] = npRand(dimLayers[i-1], 1) * 0.01
```

b)
```
import np.random.randn as
npRand N = len(dimLayers)
    for(i in range(1,N)):
            parameter['W' + str(i)] = npRand (dimLayers[i-1], dimLayers[i])) * 0.01
            parameter['b' + str(i)] = npRand (dimLayers[i-1], 1) * 0.01
```

c)
```
import np.random.randn as
npRand N = len(dimLayers)
    for(i in range(1,N)):
            parameter['W' + str(i)] = npRand (dimLayers[i], dimLayers[i - 1]), *
    0.01
            parameter['b' + str(i)] = npRand (dimLayers[i], 1) * 0.01
```

$n^L \times n^{L-1}$

d)
```
import np.random.randn as
npRand N = len(dimLayers)
    for(i in range(1,N)):
            parameter['W' + str(i)] = np.random.randn(dimLayers[i-1],
    dimLayers[i]))*0.01
            parameter['b' + str(i)] = np.random.randn(dimLayers[i], 1) * 0.01
```

3. State one advantage and one disadvantage of pooling when CNN is applied to images. Perform max pooling on the following input using a 3X3 filter and stride 2: 12 /3

7	25	13	1	18
24	12	5	18	6
11	4	17	11	22
2	15	8	20	14
18	7	20	13	3

The following Convolutional Neural Network (CNN) is applied to a database of images to predict the true class of the image out of 10 choices. Each image is of size 100 x 100 with 3 channels. Give the dimensions of the input layer of the CNN. If all convolution layers have stride 1 and all pooling functions have stride 2, determine the total number of parameters for each layer (ignore bias terms)

Input layer: As mentioned in the question above.

C1: Conv 3X3, padding same, 10 filters

C2: Conv 7X7, padding same, 20 filters

$n-f+1$

P1: 2X2 max pooling

C3: Conv 5X5, padding same, 10 filters

P2: 4X4 max pooling

FC1: fully connected

Output layer: Softmax

Note: C1, C2, C3 denote convolution layers, P1 and P2 denote pooling layers, and FC1 denotes fully connected layer.