Option in red is the correct answer

Q1. Consider a comparison between a sigmoid function and a rectified linear unit (ReLU). Which of following statement is NOT true?

A. Sigmoid function is more expensive to compute (sigmoid involves exp and division etc, while ReLU only involves checking if the input is negative or not.)

B. ReLU has non-zero gradient everywhere (on negative inputs, ReLU has 0 gradient)

C. Sigmoid has a large zone that has nonzero gradient (always nonzero)

D. The gradient of ReLU on positive inputs is 1 (ReLU(x) = x for positive inputs so gradient=1)

Q2. Let us compare a convolutional layer vs. a standard fully connected layer. Which of the following is TRUE?

A. Convolution layer has more parameters (they have few parameters)

B. Fully connected layer can be used to represent the convolution (correct. Recall that the connections in the convolutional layer are a subset of the fully connected layer's. If we have a fully connected layer, we can put the corresponding kernel weights on the connections in this subset, and put 0 weight on the other connections, then the output of this fully connect layer is the same as the convolution. That is, the fully connected layer can represent the convolution.)

C. Convolution layer can be used to represent fully connected layer (Not the other way around)

D. Fully connected layer is more efficient (they have more parameters and is less efficient)

Q3. Which one of the following layers has the **fewest** parameters to be learned during training?

A. A convolutional layer with 12 filters. Each filter is 3 x 3 operating on a single channel image.

B. A convolutional layer with 4 filters. Each filter is 6 x 6 operating on a single channel image.

C. A fully-connected layer that maps the outputs of 12 hidden units to 3 output units.

D. A max-pooling layer that reduces a 12 x 12 feature map to 6 x 6. (no parameters. The other options have nonzero numbers of parameters)

Q4. Which one of the following statement is true about neural networks?

A. The output of a max pooling layer is linear of its inputs. (max is not a linear function)

B. The performance of a neural network will always benefit by simply adding more layers. (No, we have seen that simply adding more layers can lead to worse performance. Simply adding more layers doesn't work and there can be multiple reasons: 1) more significant vanishing gradient issue; 2) instability issue (e.g., hard to learn identity function). Another reason is that more layers can mean larger models, and larger models may overfit.)

C. A convolutional layer can be represented by a fully connected layer. (correct. Explained in Q2)

D. The output of average pooling layer is not linear of its input. (mean is a linear function)

Q5. Consider one dimensional convolution. The input is [1, 2, 3, 4]. The kernel is [1, 0, 1]. Use no padding and use stride 1. What is the output?

Answer: [4, 6].

The first sliding window is [1,2,3], and after multiplying with the kernel we get 1*1 + 2*0 + 3*1 = 4. The second sliding window is [2,3,4] and the corresponding outcome is 6.

Q6. Consider max pooling on a one-dimensional vector with kernel (filter) size 4 and stride 1. With the input [7,7,3,9,2], what is the output?

Answer: [9, 9].

The first sliding window is [7, 7, 3, 9], and the maximum number is 9. The second sliding window is [7,3,9,2] and the maximum number is 9.

Q7. Consider a convolution layer with 32 filters. Each filter has size (height x width) 11x11, and 3 channels. Use padding 2x4, and stride 2x2. Given an input image of 32x32, and 3 channels, what is the output size? Write the answer in the format of #output channels x height x width.

Answer: 32 x 12 x 13.

There are 32 filters, so we have 32 output channels. The output shape for each channel can be computed using the expression:
$\lfloor (n_h - k_h + p_h + s_h)/s_h \rfloor \times \lfloor (n_w - k_w + p_w + s_w)/s_w \rfloor$

Where n_h is the input height, k_h is the kernel/filter height, p_h is the padding height, s_h is the stride for height. Similar for n_w, k_w, p_w and s_w.