CPSC 425 **Deep Learning / Neural Networks** **18 (Term 2)**

Practice Questions
Solutions

Multiple Part True/False Questions. For each question, indicate which of the statements, (A)–(D), are **true** and which are **false**? Note: Questions may have zero, one or multiple statements that are true.

Question 1. Which of the following statements about max-pooling are **true**? Which are **false**? [Adopted from Stanford's CS230]

(A) It allows a neuron in a network to have information about features in a larger part of the image, compared to a neuron at the same depth in a network without max pooling.

(B) It increases the number of parameters when compared to a similar network without max pooling.

(C) It increases the sensitivity of the network towards the position of features within an image.

Solution : (A) True, (B) False, (C) False

Question 2. Consider a simple convolutional neural network with one convolutional layer. Which of the following statements is **true** about this network? [Adopted from Stanford's CS230]

(A) It is scale invariant.

(B) It is rotation invariant.

(C) It is translation invariant.

(D) All of the above.

Solution : (A) False, (B) False, (C) True, (D) False

Short Answer Questions.

Question 3. What are the two passes of Backpropagation and their roles?

Solution : Backpropagation consists of a *forward* and *backward* pass. The role of forward pass is to evaluate the function implemented by a neural net (this also could be thought of as making a prediction). It is called a forward pass because it requires traversing the computational graph from left-to-right in topological order of the variables. The role of backwards pass is to compute all partial derivatives of the loss with respect to parameters of the network. The set of partial derivatives also gives a full Jacobian. Recall that loss is a function that tells how bad is your prediction with respect to the ground truth. You need the Jacobian (or partial derivatives) to update the parameters of the network. The reason this is called a backwards pass is that it traverses the computational graph from right-to-left, in the opposite order from the forward pass.

Question 4. Recall that using stochastic gradient descent the parameters are updated using the following equations: $\mathbf{W}_{i,j,k} = \mathbf{W}_{i,j,k} - \lambda \frac{\partial \mathcal{L}(f(\mathbf{x}_i), y_i)}{\partial \mathbf{W}_{i,j,k}}$, where $\mathbf{W}_{i,j,k}$ is an element of the weight matrix (or more broadly a parameter of the neural net), $f(\mathbf{x}_i)$ is the mapping function that neural network implements applied on sample $\mathbf{x}_i$, the y_i is the ground truth label for this sample and $\mathcal{L}$ is the loss. What is the λ called? What would you expect to happen as you set λ to be small? large?

Solution : λ is a scalar that is called *learning rate*. If it is small, it will take many gradient steps for optimization to converge. In other words, it will take long (many iterations) to learn parameters of the neural network. If it is large, you may never converge or find appropriate parameters of the neural network, because taking a large step may overshoot the optimum. This is the reason that setting appropriate λ is very important in practice.

Question 5. You want to map every possible image of size 64×64 to a binary category (*cat* or *non-cat*). Each image has 3 channels and each pixel in each channel can take an integer value between (and including) 0 and 255. How many bits do you need to represent this mapping? [Adopted from Stanford's CS230]

Solution : $256^{3 \times 64 \times 64}$

Question 6. The mapping from previous question clearly can not be stored in memory. Instead, it is typically encoded using a neural network classifier. Recall the simple single hidden layer classifier that was illustrated in class, containing an

input layer, a hidden layer and an output layer. Assuming we use a single hidden layer of size 100 for this task.
(a) What is the size of the weight matrices $\mathbf{W}_1$ and $\mathbf{W}_2$?
(b) What about the bias vectors $\mathbf{b}_1$ and $\mathbf{b}_2$?
(c) How many learnable parameters are there?
Each weight in the $\mathbf{W}_1$ and $\mathbf{W}_2$ matrices can be represented in memory using a float of size 64 bits (double precision).
(d) How many bits do you need to store your two layer neural network (you may ignore the biases $\mathbf{b}_1$ and $\mathbf{b}_2$)? [Adopted from Stanford's CS230]

Solution : (a) $\mathbf{W}_1$ is size $(64 \times 64 \times 3) \times 100$, $\mathbf{W}_2$ is size 100×1.
(b) $\mathbf{b}_1$ is size 100, $\mathbf{b}_2$ is size 1.
(c) $(64 \times 64 \times 3) \times 100 + 100 \times 1 + 100 + 1$.
(d) $64 \times ((64 \times 64 \times 3 \times 100) + (100 \times 1))$

Question 7(a). Consider a problem of detecting whether there is a *car* is in any given grayscale satellite image of size 1024×1024 pixels. Considering that the largest instance of the car that can appear in such an image is 11×11 pixels, design the simplest two-layer neural network consisting of one Convolutional and one Max Pooling layer. What would be a reasonable design? How many learnable parameters would such a network have?

Solution : A reasonable design would be a single convolutional layer with one filter (neuron) of size $11 \times 11 \times 1$ followed by a max pooling layer of size $(1024 - 10) \times (1024 - 10)$. The convolutional layer would be able to look at the object of the right size and determine if it is present at a given location. The resulting activation map will be of size $(1024 - 10) \times (1024 - 10)$. The max pooling layer will then integrate responses across the activation map. The number of parameters would be $11 \times 11 + 1 = 122$ (max pool layer has no parameters).

Question 7(b). Now consider that you have access to a library that ONLY supports $3 \times 3 \times 1$ convolutional filters. How could you implement the architecture that could conceptually have the same behavior? How many parameters would such an architecture have?

Solution : Replacing an original single CNN layer with $11 \times 11 \times 1$ filter by 5 consecutive layers of convolutional layers with filter sizes of $3 \times 3 \times 1$ would do the trick. Remember that the effective receptive field grows with the layers. A 5 layer CNN network with $3 \times 3 \times 1$ filters would have an 11×11 effective receptive field (same as with a single filter of size 11×11). The number of parameters of such a network is $5 \times (3 \times 3 + 1) = 50$. Note that this is precisely the reason that

recent convolutional architectures are deep and have smallest possible filter sizes ($3 \times 3 \times K$) at each layer.

Question 8. Consider an intermediate convolutional neural network layer within a CNN which receives as an input the output from the previous layer of size $512 \times 512 \times 128$ Assuming that the convolutional layer we are considering applies 48 7×7 filters at stride 2 with no padding. What is the number of parameters that need to be learned in this layer? What is the size of the resulting activation map?

Solution : Number of parameters: $48 \times (7 \times 7 \times 128 + 1) = 301,104$. The size of the resulting activation map: $\frac{512-6}{2} \times \frac{512-6}{2} \times 48$.

Question 9. Lets say you want to re-implement the template matching assignment on face detection using a neural network. What architectural design choices would you make and how would you setup the parameters? What would be the key difference between your original filtering implement and the neural network implementation?

Solution : You could implement something fairly similar by creating a single layer convolutional layer network with a single filter of the size of the template and a linear activation function: $a(x) = x$. It would be reasonable to initialize the weights of the layer with the template values and set bias to 0. This would implement the same operation as linear filtering, aside from the normalization.

Question 10. Consider a convolution layer. The input consists of 6 feature maps of size 20×20. The output consists of 8 feature maps, and the filters are of size 5×5. The convolution is done with a stride of 2 and with zero padding, so the output feature maps are of size 10×10. [Adopted from UofT's CSC321]

(A) Determine the number of weights in this convolution layer.

(B) Now suppose we made this a fully connected layer, but where the number of input and output units are kept the same as in the network described above. Determine the number of weights in this layer.

Solution : (A) $8 \times (6 \times 5 \times 5) = 1,200$ (this does not include 8 biases)

(B) $(6 \times 20 \times 20) \times (8 \times 10 \times 10) = 2400 \times 800 = 1,920,000$.