Deep Learning

Problem Sheet 3 Solutions —

1. (a) What is a pooling layer in a Convolutional Neural Network? What is the difference between Max pooling and Average pooling?

   Solution: A pooling layer performs a downsampling operation and is typically applied after a convolutional layer. They also allow flexibility in the location of certain features in the image, therefore allowing non-rigid, rotated or scaled objects to be recognised and they allow unions of features to be computed, e.g. blue eyes or green eyes. Both max and average pool are defined by their window size and stride parameters.

   Max pooling selects only the maximum value within the window view and in doing so extracts only the most significant of features.

   Average pooling averages the values over the window view. Average pooling is used in LeNet.

   (b) What are the stride and padding parameters in a convolutional layer?

   Solution: In a pooling or convolution operation, stride denotes how many pixels the window moves by after each operation.

   Padding determines pixels at the edges of an image are treated; for example zero padding replaces values 'outside' an image with zeroes. Padding allows the width and height of an image to be preserved.

   (c) A CNN architecture is described in the table below and takes as input an image and produces a 10-dimensional probability vector and is trained using cross-entropy loss. The architecture consists of max pooling layers as well as convolutional layers.

layer	0	1	2	3	4	5	6	7
type	input	conv	pool	conv	pool	conv	conv	loss
num. filters	-	5x5x1	2x2	5x5x20	2x2	4x4x50	1x1x500	-
stride	-	1	2	1	2	1	-	-
padding	-	0	0	0	0	0	0	-
data shape	1x28x28x1							
receptive field	1							

   The input is a $1 \times 28 \times 28 \times 1$ tensor representing $batch \times width \times height \times channels$. Calculate the data shape and receptive field for each layer.

Solution:

layer	1	2	3	4	5	6	7
type	conv	pool	conv	pool	conv	conv	loss
num. filters	5x5x1	2x2	5x5x20	2x2	4x4x50	1x1x500	-
stride	1	2	1	2	1	1	-
pad	0	0	0	0	0	0	-
data shape	1x24x24x20	1x12x12x20	1x8x8x50	1x4x4x50	1x1x1x500	1x1x1x10	1
receptive field	5	6	14	16	28	28	28

(d) Standard precision numbers take up 4 bytes per number. Half precision takes only 2 bytes per number. What are the advantages and disadvantages of using half precision?

Solution: Half precision reduces the memory required to store data and on the same GPU allows more data to be stored and enables more data to be moved in the same timeframe (with constant bandwidth) thereby decreasing training time. However, reducing precision reduces numerical accuracy and may result in underflow or overflow.

(e) Many CNN training problems are compounded by a lack of available training data. Describe some data augmentation techniques to artificially increase the amount of training data. Some of these techniques are performed at training time on the mini-batch before the forward pass. Suggest why?

Solution: Data augmentation artificially increases the amount of training data available by presenting the same image in different views. Some augmentation techniques include mirroring, random cropping, rotating, shearing colour shifting and addition of noise.

Performing augmentation during preprocessing increases the memory requirement as more samples have to be stored. An alternative is to perform augmentation during training time itself where one or more augmentations can be applied randomly to the training (mini) batch. This means less data has to be stored overall at the cost of slightly increased training time.

2. (a) In a classification problem with three classes, a model will output a 1×3 vector, c, that predicts how strongly the input belongs to a class. However, we want a probability vector, $p \in \mathbb{R}^{3\times 1}$. Such a probability vector must sum to 1 and each element must be positive. To ensure this a Softmax function is used which looks like:

$$\textbf{Given } c = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} \tag{1}$$

$$p_i = \frac{e^{c_i}}{\sum_{k=1}^{3} e^{c_k}} \forall i, k \in 1, 2, 3 \tag{2}$$

Verify whether p fulfils the necessary criteria of all elements being non-negative and summing up to 1.

Solution: For a real number c_i, the exponential e^{c_i} is always non-negative as is the sum $\sum_{k=1}^{3} e^{c_k}$, therefore the elements of p too are non-negative and they do add up to 1.

(b) Compute the derivative $\frac{\partial p_i}{\partial c_j}$. Consider the two scenarios when $i = j$ and $i \neq j$.

Solution:

$$\begin{aligned}
\text{If } i = j, \frac{\partial p_i}{\partial c_j} &= \frac{e^{c_i} \sum_k e^{c_k} - e_{c_i} e^{c_j}}{(\sum_k e^{c_k})^2} \\
&= \frac{e^{c_i}}{\sum_k e^{c_k}} \frac{\sum_k e^{c_k} - e^{c_j}}{\sum_k e^{c_k}} \\
&= p_i \cdot (1 - p_j) \\
\text{If } i \neq j, \frac{\partial p_i}{\partial c_j} &= \frac{-e^{c_i} e^{c_j}}{(sum_k e^{c_k})^2} \\
&= \frac{e^{c_i}}{\sum_k e^{c_k}} \frac{-e^{c_j}}{\sum_k e^{c_k}} \\
&= p_i \cdot -p_j
\end{aligned}$$

(c) Suppose that an image, I is corrupted by white (Gaussian) noise $n \sim N(0, \sigma^2)$ giving us the result $Y = I + n$. One approach to denoising the image is to take N snapshots of an object from the same view, yielding multiple images and then taking the average of all the images. Here, each image is given by $Y_i = I + n_i \forall i \in 1, ..., N$. Taking the average of the N noisy images yields the denoised image $Y_d = \frac{1}{N} \sum_N Y_N$. Derive the mean, μ_d, and variance, σ_d^2, of the denoised image Y_d.

Solution:

$$Y_d = \frac{1}{N}\sum_N Y_N = \frac{1}{N}\sum_i^N I + n_i = I + \frac{1}{N}\sum_i^N n_i$$

Hence, the noise in the denoised image is given by $e = Y - I = \frac{1}{N}\sum_i^N n_i$

The expectation $\mathbb{E}[e] = \mu_d = \frac{1}{N} = \sum_i^N \mathbb{E}[n_i] = 0$

The variance is given by $var(e) = \sigma_d^2 = \mathbb{E}[e^2] - \mathbb{E}[e]^2 = \frac{1}{N^2}\mathbb{E}[\sum_i^N n_i^2 + \sum_{i \neq j}^N n_i n_j] = \frac{1}{N}\sigma^2$

(d) Given an image:

3	4	8	10	22	45	50	65
3	4	8	10	22	45	50	65
3	4	8	10	22	45	50	65
3	4	8	10	22	45	50	65
3	4	8	10	22	45	50	65
3	4	8	10	22	45	50	65
3	4	8	10	22	45	50	65
3	4	8	10	22	45	50	65
3	4	8	10	22	45	50	65

And the horizontal and vertical 3×3 Prewitt filter kernels:

$$h_x = \begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix}$$

$$h_y = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix}$$

Assuming zero padding, compute the gradient magnitude of the second row after convolving with both kernels.

Solution: Remember to transpose the kernel matrices before convolving.

The convolution with the horizontal filter is given by:

$$g_x = I * h_x = \begin{pmatrix} 12 & 15 & 18 & 42 & 105 & 84 & 60 & -150 \end{pmatrix}$$

The convolution with the vertical filter is given by:

$$g_x = I * h_y = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

And computing the magnitude yields:

$$g = \sqrt{g_x^2 + g_y^2} = \begin{pmatrix} 12 & 15 & 18 & 42 & 105 & 84 & 60 & 150 \end{pmatrix}$$

(e) Repeat the previous question but now use reflection (mirrored) padding.

Solution: We have:

$$g_x = \begin{pmatrix} 0 & 15 & 18 & 42 & 105 & 84 & 60 & 0 \end{pmatrix}$$

$$g_x = I * h_y = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

And so:

$$g = \sqrt{g_x^2 + g_y^2} = \begin{pmatrix} 12 & 15 & 18 & 42 & 105 & 84 & 60 & 150 \end{pmatrix}$$

3. (a) What is the difference between invariance and equivariance? Is convolution either of these?

Solution: For invariance, a function f is invariant with respect to a transformation T if $f(T(x)) = f(x)$, meaning the output does not change even when the input is transformed.

For equivariance, a function f is equivariant with respect to a transformation T if $f(T(x)) = T(f(x))$, meaning the applying the transformation to x is the same as applying the transformation to $f(x)$.

Convolution is only equivariant to translation (shift equivariant) where the output is shifted by the same amount as the input. If the image were to be flipped, warped etc. then convolution is not equivariant. This is one of the reasons why data augmentation by flipping and cropping is performed to enable better model generalisation.

As a side note, a CNN can be made (approximately) shift invariant by including pooling layers.

(b) What is a 2D separable filter? Are all 2D filters separable?

Solution: A separable filter is a filter which can be written as a convolution of two or more simple filters. In the case of a 2D separable filter, it can typically be expressed as a combination of two 1D filters. An example of this is the Sobel operator, or as in question 2d, the Prewitt filter.

Not all 2D filters are separable. A separable filter must be able to be decomposed into a combination of simpler filters. Assume we are given an $M \times N$ 2D filter, F and asked to determine whether it is separable. If F can be decomposed into two filter of shape $M \times 1$ and $1 \times N$. We know that with Singular Value Decomposition (SVD) that we can decompose a matrix as $J = U\Sigma V^*$ and if J is real, then $U, V^* = V$ are real orthonormal matrices. The matrix Σ is a diagonal matrix containing only singular values which indicate how much "information" each corresponding vector contributes to making up J. Therefore, if J is separable, then it is rank 1 and it has only 1 singular value (and if rank 1, it would have determinant 0).

(c) Is the following 2D filter separable?

$$F = \begin{pmatrix} 2 & 3 \\ 1 & 1 \end{pmatrix}$$

If so separate it, otherwise explain why it is not separable.

Solution: There are multiple ways of doing this.

This filter is not separable as $det(F) = -1$. For separability, the determinant should be 0.

(d) A CNN has four consecutive 3×3 convolutional layers with stride 1 and no pooling. How large is the support of a neuron in the fourth layer?

Solution: With a stride of 1, no pooling and a 3×3 the outermost two-width pixels are removed as the input passes from one layer to the next. This means the dimension of the input is reduced from $n \times n$ to $(n-2) \times (n-2)$. As a result a 1×1 patch at the input grows as it passes through each layer as:
$1 \times 1 \rightarrow 3 \times 3 \rightarrow 5 \times 5 \rightarrow 7 \times 7 \rightarrow 9 \times 9$

Which means a neuron in the fourth layer has a support of 81 pixels.

(e) Why are skip connections used in deep CNN architectures such as ResNet?

Solution: Skip connections can help solve the vanishing gradient problem.

The standard update rule for a given layer with weights $w = [w_1, ..., w_N]$, with respect to the loss L is given by $w_i^{'} = w_i^{'} - \lambda \frac{\partial L}{\partial w_i}$ where λ is the learning rate and using partial derivatives we can write $\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial z} \frac{\partial z}{\partial w_i}$ where z is the output of the

layer. As errors propagate through the layers during back-propagation, the error itself decreases resulting in increasingly small updates to weights.

Skip connections introduce lower level features to a 'deeper' layer which means the partial derivative now looks like $\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial z}(\frac{\partial z}{\partial w_i} + 1) = \frac{\partial L}{\partial z}\frac{\partial z}{\partial w_i} + \frac{\partial L}{\partial z}$. This means the error with respect to a weight w_i is larger. Note that ResNet uses skip connection via addition, the other method being skip connections via concatenation in architectures like DenseNet.

(f) Calculate the (big O) computational complexity of applying both separable and non-separable $K \times K$ Gaussian filters to an $N \times N$ image.

Solution: The separable filter has complexity $O(N^2K)$.

The non-separable filter has complexity $O(N^2K^2)$.