4 Deep Discriminative Neural Networks

1. Give brief definitions of the following terms:

a. Feature map

The output produced by a single mask (before or after the application of the activation function, or pooling) in a convolutional layer of a CNN. It shows the responses produced by identical neurons at different spatial locations.

b. Sub-sampling

The process of reducing the width and height of a feature map by pooling (also the process of reducing the resolution of an image).

c. Adversarial example

A sample that has been manipulated so that it is classified differently to the original sample.

d. Data augmentation

A method of expanding the number of samples in a dataset by adding new samples that are the original samples after the application of class-preserving transformations.

e. Transfer learning

A method of training a classifier that utilises the results of pre-training the classifier using another dataset.

f. Dropout

A method of reducing over-fitting that works by giving an activation of zero to a random sample of neurons in a layer at each iteration during training.

g. Regularization

Any method intended to reduce over-fitting and improve generalisation.

2. Define what is meant by, and explain the causes of, the "vanishing gradient problem". Briefly describe four methods that can be used to reduce its effects.

When errors, backpropagated through a neural network, become smaller and smaller resulting in smaller and smaller weight updates. It is due to the error being multiplied at each layer of the network by derivatives or weights that are small. It can result in little or no learning occurring in the earlier layers of the network, and hence, causing these layer to fail to contribute to solving the task.

Methods that reduce the likelihood of gradients vanishing are:

- *using activation functions, like ReLU, which have a smaller range for which the derivative is* < 1.
- *initialising the weights in ways that have been found, empirically, to reduce the effects of vanishing gradients.*
- *using variations of standard backpropagation that use momentum and/or an adaptive learning rate.*
- *using batch normalisation to keep the mean and standard deviation of each neuron close to 0 and 1 respectively.*

- *using skip connections so the gradient can by-pass layers in the network where the gradient has vanished.*

3. Mathematically define the following activation functions:

a. ReLU

$\varphi(net_j) = net_j$ *if* $net_j \geq 0$ *or* 0 *if* $net_j < 0$

b. LReLU

$\varphi(net_j) = net_j$ *if* $net_j \geq 0$ *or* $a \times net_j$ *if* $net_j < 0$ *where* a *is a constant hyper-parameter.*

c. PReLU

$\varphi(net_j) = net_j$ *if* $net_j \geq 0$ *or* $a_j \times net_j$ *if* $net_j < 0$ *where* a_j *is a learnt parameter.*

4. The following array show the output produced by a mask in a convolutional layer of a CNN.

$$net_j = \begin{bmatrix} 1 & 0.5 & 0.2 \\ -1 & -0.5 & -0.2 \\ 0.1 & -0.1 & 0 \end{bmatrix}$$

Calculate the values produced by the application of the following activation functions:

a. ReLU,

b. LReLU when a=0.1,

c. tanh,

d. Heaviside function where each neuron has a threshold of 0.1 (define H(0) as 0.5).

a) ReLU:

$$\begin{bmatrix} 1 & 0.5 & 0.2 \\ 0 & 0 & 0 \\ 0.1 & 0 & 0 \end{bmatrix}$$

b) LReLU when a=0.1:

$$\begin{bmatrix} 1 & 0.5 & 0.2 \\ -0.1 & -0.05 & -0.02 \\ 0.1 & -0.01 & 0 \end{bmatrix}$$

c) tanh:

$$\begin{bmatrix} 0.7616 & 0.4621 & 0.1974 \\ -0.7616 & -0.4621 & -0.1974 \\ 0.0997 & -0.0997 & 0 \end{bmatrix}$$

d) Heaviside function where each neuron has a threshold of 0.1:

$$\begin{bmatrix} H(1-0.1) & H(0.5-0.1) & H(0.2-0.1) \\ H(-1-0.1) & H(-0.5-0.1) & H(-0.2-0.1) \\ H(0.1-0.1) & H(-0.1-0.1) & H(0-0.1) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0.5 & 0 & 0 \end{bmatrix}$$

5. The following arrays show the output produced by a convolutional layer to all 4 samples in a batch. $X_1 = \begin{bmatrix} 1 & 0.5 & 0.2 \\ -1 & -0.5 & -0.2 \\ 0.1 & -0.1 & 0 \end{bmatrix}$, $X_2 = \begin{bmatrix} 1 & -1 & 0.1 \\ 0.5 & -0.5 & -0.1 \\ 0.2 & -0.2 & 0 \end{bmatrix}$, $X_3 = \begin{bmatrix} 0.5 & -0.5 & -0.1 \\ 0 & -0.4 & 0 \\ 0.5 & 0.5 & 0.2 \end{bmatrix}$, $X_4 = \begin{bmatrix} 0.2 & 1 & -0.2 \\ -1 & -0.6 & -0.1 \\ 0.1 & 0 & 0.1 \end{bmatrix}$. **Calculate the corresponding outputs produced after the application of batch normalisation, assuming the following parameter values** $\beta = 0$**,** $\gamma = 1$**, and** $\epsilon = 0.1$ **which are the same for all neurons.**

Batch normalisation modifies the output of an individual neuron, x*, to become:*

$$BN(x) = \beta + \gamma \frac{x - E(x)}{\sqrt{Var(x) + \epsilon}}$$

Where $E(x)$ *is the mean, and* $Var(x)$ *is the variance, of* x *within the batch.*

For the top-left neuron:
$E(x) = (1 + 1 + 0.5 + 0.2)/4 = 0.675$,
$Var(x) = \frac{(1-0.675)^2+(1-0.675)^2+(0.5-0.675)^2+(0.2-0.675)^2}{4} = 0.1169$
$BN(x_1) = 0 + 1 \times \frac{1-0.675}{\sqrt{0.1169+0.1}} = 0.6979$
$BN(x_2) = 0 + 1 \times \frac{1-0.675}{\sqrt{0.1169+0.1}} = 0.6979$
$BN(x_3) = 0 + 1 \times \frac{0.5-0.675}{\sqrt{0.1169+0.1}} = -0.3758$
$BN(x_4) = 0 + 1 \times \frac{0.2-0.675}{\sqrt{0.1169+0.1}} = -1.0200$

Applying the same method to the output of each neuron gives:

$$BN(X_1) = \begin{bmatrix} 0.6979 & 0.5872 & 0.5657 \\ -0.8652 & 0 & -0.3086 \\ -0.3509 & -0.3612 & -0.2294 \end{bmatrix}, BN(X_2) = \begin{bmatrix} 0.6979 & -1.1744 & 0.2828 \\ 1.2112 & 0 & 0 \\ -0.0702 & -0.6019 & -0.2294 \end{bmatrix},$$

$$BN(X_3) = \begin{bmatrix} -0.3758 & -0.5872 & -0.2828 \\ 0.5191 & 0.3086 & 0.3086 \\ 0.7720 & 1.0835 & 0.3824 \end{bmatrix}, BN(X_4) = \begin{bmatrix} -1.0200 & 1.1744 & -0.5657 \\ -0.8652 & -0.3086 & 0 \\ -0.3509 & -0.1204 & 0.0765 \end{bmatrix}$$

Note, the top-left neuron's outputs were originally all positive, but after batch normalisation have been scaled to be both positive and negative; the top-middle neuron's outputs were originally well spread out in the range -1 to +1 and have not been changed much after batch normalisation; the top-right neuron's outputs were originally all similar, but after batch normalisation have been scaled to cover a wider range of values.

Note, that often the batch normalisation parameters β *and* γ *are not pre-defined hyper-parameters but are learnt, via back-propagation, at the same time as the weights of the network.*

Note, that for convolutional layers, rather than performing batch normalisation separately at each location in each feature map, it is common to apply the same normalisation to all locations in a feature map by calculating the mean and variance across all locations and all samples in the batch: so that different elements of the same feature map, at different locations, are normalised in the same way. However, for the purposes of the tutorial and exam questions, we will use the formulation used above.

Note, other forms of normalisation, such as layer normalisation, group normalisation, and instance normalisation, use the same method but calculate the mean and variance across different dimensions.

6. The following arrays show the feature maps that provide the input to a convolutional layer of a CNN.

$$X_1 = \begin{bmatrix} 0.2 & 1 & 0 \\ -1 & 0 & -0.1 \\ 0.1 & 0 & 0.1 \end{bmatrix},\ X_2 = \begin{bmatrix} 1 & 0.5 & 0.2 \\ -1 & -0.5 & -0.2 \\ 0.1 & -0.1 & 0 \end{bmatrix}$$

If a mask, H, has two channels defined as:

$$H_1 = \begin{bmatrix} 1 & -0.1 \\ 1 & -0.1 \end{bmatrix},\ H_2 = \begin{bmatrix} 0.5 & 0.5 \\ -0.5 & -0.5 \end{bmatrix}$$

Calculate the output produced by mask H when using:

- **a. padding=0, stride=1**
- **b. padding=1, stride=1**
- **c. padding=1, stride=2**
- **d. padding=0, stride=1, dilation=2**

a) padding=0, stride=1:

$$X_1 \star H_1 + X_2 \star H_2 =$$

$$\begin{bmatrix} (0.2 \times 1) + (1 \times \text{-}0.1) + (\text{-}1 \times 1) + (0 \times \text{-}0.1) & (1 \times 1) + (0 \times \text{-}0.1) + (0 \times 1) + (\text{-}0.1 \times \text{-}0.1) \\ (\text{-}1 \times 1) + (0 \times \text{-}0.1) + (0.1 \times 1) + (0 \times \text{-}0.1) & (0 \times 1) + (\text{-}0.1 \times \text{-}0.1) + (0 \times 1) + (0.1 \times \text{-}0.1) \end{bmatrix}$$
$$+ \begin{bmatrix} (1 \times 0.5) + (0.5 \times 0.5) + (\text{-}1 \times \text{-}0.5) + (\text{-}0.5 \times \text{-}0.5) & (0.5 \times 0.5) + (0.2 \times 0.5) + (\text{-}0.5 \times \text{-}0.5) + (\text{-}0.2 \times \text{-}0.5) \\ (\text{-}1 \times 0.5) + (\text{-}0.5 \times 0.5) + (0.1 \times \text{-}0.5) + (\text{-}0.1 \times \text{-}0.5) & (\text{-}0.5 \times 0.5) + (\text{-}0.2 \times 0.5) + (\text{-}0.1 \times \text{-}0.5) + (0 \times \text{-}0.5) \end{bmatrix}$$

$$= \begin{bmatrix} -0.9 & 1.01 \\ -0.9 & 0 \end{bmatrix} + \begin{bmatrix} 1.5 & 0.7 \\ -0.75 & -0.3 \end{bmatrix} = \begin{bmatrix} 0.6 & 1.71 \\ -1.65 & -0.3 \end{bmatrix}$$

b) padding=1, stride=1:

$$X_1' = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0.2 & 1 & 0 & 0 \\ 0 & -1 & 0 & -0.1 & 0 \\ 0 & 0.1 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix},\ X_2' = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0.5 & 0.2 & 0 \\ 0 & -1 & -0.5 & -0.2 & 0 \\ 0 & 0.1 & -0.1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$X_1' \star H_1 + X_2' \star H_2 =$$

$$\begin{bmatrix} \text{-}0.02 & 0.1 & 1 & 0 \\ 0.08 & \text{-}0.9 & 1.01 & \text{-}0.1 \\ 0.09 & \text{-}0.9 & 0 & 0 \\ \text{-}0.01 & 0.1 & \text{-}0.01 & 0.1 \end{bmatrix} + \begin{bmatrix} \text{-}0.5 & \text{-}0.75 & \text{-}0.35 & \text{-}0.1 \\ 1 & 1.5 & 0.7 & 0.2 \\ \text{-}0.55 & \text{-}0.75 & \text{-}0.3 & \text{-}0.1 \\ 0.05 & \text{-}0 & \text{-}0.05 & 0 \end{bmatrix} = \begin{bmatrix} \text{-}0.52 & \text{-}0.65 & 0.65 & \text{-}0.1 \\ 1.08 & 0.6 & 1.71 & 0.1 \\ \text{-}0.46 & \text{-}1.65 & \text{-}0.3 & \text{-}0.1 \\ 0.04 & 0.1 & \text{-}0.06 & 0.1 \end{bmatrix}$$

c) padding=1, stride=2:

using answer to previous part:

$$\begin{bmatrix} -0.52 & 0.65 \\ -0.46 & -0.3 \end{bmatrix}$$

d) padding=0, stride=1, dilation=2:

$$H'_1 = \begin{bmatrix} 1 & 0 & -0.1 \\ 0 & 0 & 0 \\ 1 & 0 & -0.1 \end{bmatrix},\ H'_2 = \begin{bmatrix} 0.5 & 0 & 0.5 \\ 0 & 0 & 0 \\ -0.5 & 0 & -0.5 \end{bmatrix}$$

$$X_1 \star H'_1 + X_2 \star H'_2$$

$$= \begin{bmatrix} 0.29 \end{bmatrix} + \begin{bmatrix} 0.55 \end{bmatrix} = \begin{bmatrix} 0.84 \end{bmatrix}$$

7. The following arrays show the feature maps that provide the input to a convolutional layer of a CNN.

$$X_1 = \begin{bmatrix} 0.2 & 1 & 0 \\ -1 & 0 & -0.1 \\ 0.1 & 0 & 0.1 \end{bmatrix},\ X_2 = \begin{bmatrix} 1 & 0.5 & 0.2 \\ -1 & -0.5 & -0.2 \\ 0.1 & -0.1 & 0 \end{bmatrix},\ X_3 = \begin{bmatrix} 0.5 & -0.5 & -0.1 \\ 0 & -0.4 & 0 \\ 0.5 & 0.5 & 0.2 \end{bmatrix}$$

Calculate the output produced by 1x1 convolution when the 3 channels of the 1x1 mask are: $[1, -1, 0.5]$**.**

$$Y =$$

$$\begin{bmatrix} (0.2 \times 1) + (1 \times \text{-}1) + (0.5 \times 0.5) & (1 \times 1) + (0.5 \times \text{-}1) + (\text{-}0.5 \times 0.5) & (0 \times 1) + (0.2 \times \text{-}1) + (\text{-}0.1 \times 0.5) \\ (\text{-}1 \times 1) + (\text{-}1 \times \text{-}1) + (0 \times 0.5) & (0 \times 1) + (\text{-}0.5 \times \text{-}1) + (\text{-}0.4 \times 0.5) & (\text{-}0.1 \times 1) + (\text{-}0.2 \times \text{-}1) + (0 \times 0.5) \\ (0.1 \times 1) + (0.1 \times \text{-}1) + (0.5 \times 0.5) & (0 \times 1) + (\text{-}0.1 \times \text{-}1) + (0.5 \times 0.5) & (0.1 \times 1) + (0 \times \text{-}1) + (0.2 \times 0.5) \end{bmatrix}$$

$$Y = \begin{bmatrix} -0.55 & 0.25 & -0.25 \\ 0 & 0.3 & 0.1 \\ 0.25 & 0.35 & 0.2 \end{bmatrix}$$

8. The following array shows the input to a pooling layer of a CNN.

$$X_1 = \begin{bmatrix} 0.2 & 1 & 0 & 0.4 \\ -1 & 0 & -0.1 & -0.1 \\ 0.1 & 0 & -1 & -0.5 \\ 0.4 & -0.7 & -0.5 & 1 \end{bmatrix}$$

Calculate the output produced by the pooling when using:

a. average pooling with a pooling region of 2x2 and stride=2

b. max pooling with a pooling region of 2x2 and stride=2

c. max pooling with a pooling region of 3x3 and stride=1

a) average pooling with a pooling region of 2x2 and stride=2:

$$\begin{bmatrix} (0.2 + 1 - 1 + 0)/4 & (0 + 0.4 - 0.1 - 0.1)/4 \\ (0.1 + 0 + 0.4 - 0.7)/4 & (-1 - 0.5 - 0.5 + 1)/4 \end{bmatrix} = \begin{bmatrix} 0.05 & 0.05 \\ -0.05 & -0.25 \end{bmatrix}$$

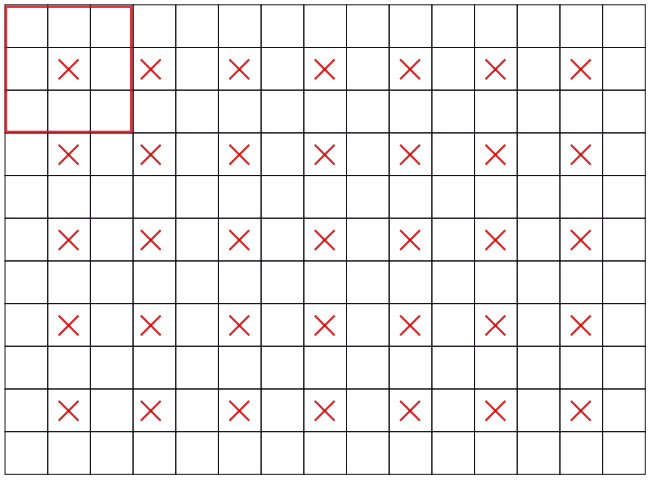
b) max pooling with a pooling region of 2x2 and stride=2:

$$\begin{bmatrix} 1 & 0.4 \\ 0.4 & 1 \end{bmatrix}$$

c) max pooling with a pooling region of 3x3 and stride=1:

$$\begin{bmatrix} 1 & 1 \\ 0.4 & 1 \end{bmatrix}$$

9. The input to a convolutional layer of a CNN consists of 6 feature maps each of which has a height of 11 and width of 15 (*i.e.*, input is $11 \times 15 \times 6$). What size will the output produced by a single mask with 6 channels and a width of 3 and a height of 3 (*i.e.*, $3 \times 3 \times 6$) when using a stride of 2 and padding of 0.



output size will be: $5 \times 7 \times 1$

Note, in general $\textit{outputDim} = 1 + \frac{(\textit{inputDim} - \textit{maskDim} + 2 \times \textit{padding})}{\textit{stride}}$.
So for this example:

$\textit{outputHeight} = 1 + \frac{(11-3+2\times 0)}{2} = 1 + 4 = 5.$

$\textit{outputWidth} = 1 + \frac{(15-3+2\times 0)}{2} = 1 + 6 = 7.$

10. A CNN processes an image of size 200x200x3 using the following sequence of layers:

- **convolution with 40 masks of size 5x5x3 with stride=1, padding=0**
- **pooling with 2x2 pooling regions stride=2**
- **convolution with 80 masks of size 4x4 with stride=2, padding=1**

- **1x1 convolution with 20 masks**

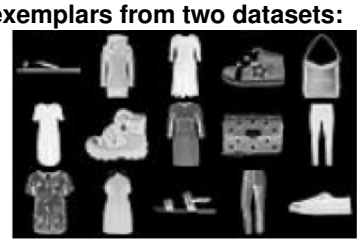
What is the size of the output once it has been flattened?

Using: $\textit{outputDim} = 1 + \frac{(\textit{inputDim} - \textit{maskDim} + 2 \times \textit{padding})}{\textit{stride}}$.

- *convolution with 40 masks of size 5x5x3 with stride=1, padding=0:*
  width=height=1+(200-5)/1=196
  number of channels=40 (as there are 40 masks)
  so size of output is 196x196x40
- *pooling with 2x2 pooling regions stride=2:*
  width=height=1+(196-2)/2=98
  number of channels=40 (as pooling does not change this)
  so size of output is 98x98x40
- *convolution with 80 masks of size 4x4 with stride=2, padding=1:*
  width=height=1+(98-4+2)/2=49
  number of channels=80 (as there are 80 masks)
  so size of output is 49x49x80
- *1x1 convolution with 20 masks:*
  width=height=49 (as 1x1 convolution does not change this)
  number of channels=20 (as there are 20 masks)
  so size of output is 49x49x20
- *after flatterning, length of feature vector is: 48020*

11. The following images show exemplars from two datasets:

MNIST



Fashion MNIST

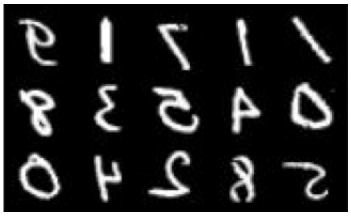
Each dataset is to be expanded using data augmentation. Which of the following transformations are appropriate:

a. rescaling

MNIST: yes, FashionMNIST: yes

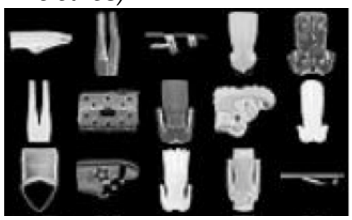
b. horizontal flip

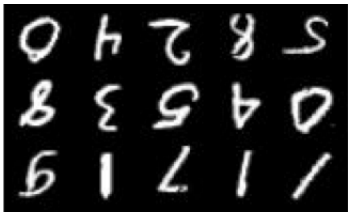
MNIST: no, FashionMNIST: yes



c. rotation

MNIST: no (small rotations would be OK), FashionMNIST: yes (if we want classifier to recognise up-side-down clothes)





d. cropping

MNIST: yes, FashionMNIST: yes
Assuming object still recognisable after crop