Deep Learning
Tutorial 1: Signals and Convolution Solutions

Q1: The curse of dimensionality

1a) For a one-dimensional space of real numbers between 0 and 1, 100 observations are required to adequately cover this space such that histograms can be calculated and conclusions can be drawn. How many observations would be required to adequately cover the space for the following number of dimensions?

i) 10 dimensions

A: 100^{10}

ii) 1000 dimensions

A: 100^{1000}

1b) Under what conditions is it still possible to perform Cluster analysis and outlier detection in high-dimensional spaces?

A: The pure number of dimensions has been shown not to be decisive to decide the number of observations required to cover an input-space, as additional relevant information can better seperate the data. Only additional dimensions that are 'irrelevant' to the separation cause the effect. While the exact distance values become more similar, the resulting order then remains stable and Cluster analysis and outlier detection is still possible with suitable methods.

1c) Why is the curse of dimensionality a serious hurdle in machine learning problems?

A: The curse of dimensional is a serious hurdle in machine learning problems that have to learn the structure of high-dimensional space from few sample elements. In other words, with increasing dimensions the number of observations required to learn from can

increase to such an extent that it becomes impractical/impossible.

1d) Why is using weight sharing common practice?

A: 1. It reduces the number of parameters of a model reducing comutational complexity. 2. Convolutions can be implemented through weight sharing i.e the weights of N spacial neighbours are identical and these shared weights can be interpreted as weights of a filter function.

1	1	2	3	2	1	1
1	2	6	6	5	2	1
1	5	6	7	7	6	2
1	5	6	6	6	4	2
1	1	4	5	4	1	1
1	1	3	4	3	1	0
0	0	1	2	1	0	1

Figure 1: Single channel image

0	1	0
0	-2	1
0	1	0

Figure 2: 3x3 Filter

Q2: Convolutions

2a) Suppose you have the following single-channel image shown in Figure 1:

i) Compute the result of max pooling of size 3x3 with stride=2.

A: see figure 3

ii) Compute the result of convolution of the input image with the 3x3 filter shown in Figure 2 using stride=2, no zero padding.

A: see figure 4

2bi) How would the filter shown in Figure 2 need to changed to obtain the correct definition of convolution rather than cross-correlation? Check your answer to **2aii)**.

A: The filter would need to be flipped horizontally (only horizontally as it is symmetric vertically in this case)

2bii) Why do we need to perform this change?

6	7	7
6	7	7
4	5	4

Figure 3: **A**: image after max pooling

3	4	8
-3	6	5
0	2	2

Figure 4: **A**: image after convolution with filter

A: To obtain the correct definition of convolution rather than correlation which are similar: for real-valued functions, of a continuous or discrete variable, it differs from cross-correlation $(f * g)$ only in that either $f(x)$ or $g(x)$ is reflected about the y-axis; thus it is a cross-correlation of $f(x)$ and $g(-x)$, or $f(-x)$ and $g(x)$.

2c) Which of the following properties hold for convolution? (True/False)

i) Non-Commutativity: $f * g \neq g * f$ **A: False**

ii) Associativity: $f * (g * h) = (f * g) * h$ **A: True**

iii) Non-Distributivity: $f * (g + h) \neq (f * g) + (f * h)$ **A: False**

iv) Associativity with scalar multiplication: $a(f * g) = (af) * g$ **A: True**

v) Derivative: $D(f * g) = (Df) * g = f * Dg$ **A: True**

Q3: Backpropagation applied

A Siamese network is a type of neural network architecture that consists of two or more identical subnetworks (called "twins") that share the same weights and architecture. Siamese networks are often used for tasks that involve comparing or matching inputs, such as verification, identification, and similarity learning.

The outputs of the twin networks are usually joined later on by more layers. Let's assume we have a two layer Siamese neural network, as defined

below:

$$
\begin{aligned}
z_1 &= W_1 x^{(i)} + b_1 \\
a_1 &= ReLU(z_1) \\
z_2 &= W_1 x'^{(i)} + b_1 \\
a_2 &= ReLU(z_2) \\
a &= a_1 - a_2 \\
z_3 &= W_2 a + b_2 \\
\hat{y}^{(i)} &= \sigma(z_3) \\
\mathcal{L}^{(i)} &= y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}) \\
J &= -\frac{1}{m} \sum_{i=1}^{m} \mathcal{L}^{(i)}
\end{aligned}
$$

Note that $(x^{(i)}, x'^{(i)})$ represents a pair of single input examples, and are each of shape $D \times 1$. Further $y^{(i)}$ is a single output label and is a scalar. There are m examples in our dataset. We use D_{a1} nodes in our first hidden layers; *i.e.*, z_1's and z_2's shape is $D_{a1} \times 1$. Note that the first two layers share the same weights.

What are the shapes of W_1, b_1, W_2, b_2? If we were vectorizing across multiple examples, what would the shapes of X and Y be instead?

follow $W_1 \in \mathbb{R}^{D_{a1} \times D_x}$, $b_1 \in \mathbb{R}^{D_{a1} \times 1}$, $W_2 \in \mathbb{R}^{1 \times D_{a1}}$, $b_2 \in \mathbb{R}^{1 \times 1}$

$X \in \mathbb{R}^{D_x \times m}$, $Y \in \mathbb{R}^{m \times 1}$ after vectoring.

Derive $\frac{\partial J}{\partial z_3}$ formally and write $\delta_1^i = ...$ You can simplify the equation in terms of $\hat{y}^{(i)}$.

$\delta_1^i = -\frac{1}{m}(\frac{y^{(i)}}{\hat{y}^{(i)}} - \frac{(1-y^{(i)})}{(1-\hat{y}^{(i)})} * \sigma(z_3)(1 - \sigma(z_3))$ or $\frac{1}{m}(\hat{y}^{(i)} - y^{(i)})$

Derive $\frac{\partial z_3}{\partial a}$ formally and write $\delta_2^i = ...$

$\delta_2^i = W_2$

Derive $\frac{\partial a}{\partial z_2}$ formally and write $\delta_3^i = ...$

$\delta_3^i = -H(z_2)$, where H is the step function