IFT-3395/6390 Fundamentals of Machine Learning,
Professors : Guillaume Rabusseau / Ioannis Mitliagkas

Solution for Homework 3

Question 1

A)
The derivative of the relu function can be expressed as the following :

$$g'(x) = \begin{cases} 1, & \text{when } 0 < x \\ 0, & \text{when } 0 \geq x \end{cases}$$

Since the derivative of x with respect to x is 1.

Note that technically, the derivative does not exist at x=0, but the question says we can put 0 in that case.

B)
Knowing that the sigmoid function is : $\sigma(x) = \frac{1}{1+e^{-x}}$, we can calculate its derrivative.

$$\begin{aligned}
\frac{\partial \sigma(x)}{\partial x} = \frac{\partial \frac{1}{1+e^{-x}}}{\partial x} &= \frac{\partial (1+e^{-x})^{-1}}{\partial x} \\
&= -(1+e^{-x})^{-2} \times -e^{-x} \quad \text{(Chain Rule)} \\
&= \frac{e^{-x}}{(1+e^{-x})^2} \\
&= \frac{1}{(1+e^{-x})} \frac{e^{-x}}{(1+e^{-x})} \\
&= \sigma(x) \frac{e^{-x}}{(1+e^{-x})} \\
&= \sigma(x) \left(\frac{1+e^{-x}-1}{(1+e^{-x})} \right) \\
&= \sigma(x) \left(\frac{1+e^{-x}}{(1+e^{-x})} - \frac{1}{(1+e^{-x})} \right) \\
&= \sigma(x)(1-\sigma(x))
\end{aligned}$$

C)

$$\begin{aligned}
\sigma(x) =& \frac{1}{2}\left(\tanh\frac{1}{2}x + 1\right) \\
=& \frac{1}{2}\left(\frac{e^{x/2} - e^{-x/2}}{e^{x/2} + e^{-x/2}} + \frac{e^{x/2} + e^{-x/2}}{e^{x/2} + e^{-x/2}}\right) \\
=& \frac{1}{2}\left(\frac{2e^{x/2}}{e^{x/2} + e^{-x/2}}\right) \\
=& \frac{e^{x/2}}{e^{x/2} + e^{-x/2}} \\
=& \left(\frac{e^{x/2}}{e^{x/2} + e^{-x/2}}\right)\left(\frac{e^{-x/2}}{e^{-x/2}}\right) \\
=& \frac{1}{1 + e^{-x}} \\
=& \sigma(x)
\end{aligned}$$

D)
Showing that $ln(\sigma(x)) = -softplus(-x)$

$$\begin{aligned}
ln(\sigma(x)) &= ln\left(\frac{1}{1 + e^{-x}}\right) \\
&= ln(1) - ln(1 + e^{-x}) \quad \text{(Log properties)} \\
&= 0 - ln(1 + e^{-x}) = -softplus(-x)
\end{aligned}$$

E)
Showing that softplus(x) - softplus(-x) = x :

$$\begin{aligned}
softplus(x) - softplus(-x) &= ln(1 + e^{x}) - ln(1 + e^{-x}) \\
&= ln(1 + e^{x}) - ln(e^{-x}(1 + e^{x})) \\
&= ln\left(\frac{1 + e^{x}}{e^{-x}(1 + e^{x})}\right) \\
&= ln\left(\frac{1}{e^{-x}}\right) \\
&= ln(1) - ln(e^{-x}) \\
&= 0 + x \\
&= x
\end{aligned}$$

F) We define the sign function as :

$$\text{sign}(x) = \begin{cases} 1 & x > 0 \\ 0 & x = 0 \\ -1 & x < 0 \end{cases}$$

Using only indicator functions, the sign function can be written as :

$$\text{sign}(x) = \mathbb{1}_{x\geq 0}(x) - \mathbb{1}_{x\leq 0}(x)$$

G) $\frac{\partial \|\mathbf{x}\|_2^2}{\partial \mathbf{x}}$ is the gradient of a scalar in respect to a vector. Given y, a scalar, and $\mathbf{x}$, a vector, we have :

$$\frac{\partial y}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y}{\partial x_1} \\ \frac{\partial y}{\partial x_2} \\ \vdots \\ \frac{\partial y}{\partial x_n} \end{bmatrix}$$

In that case we must compute $\frac{\partial \|\mathbf{x}\|_2^2}{\partial x_k}$ to find the value of each element.

$$\frac{\partial \|\mathbf{x}\|_2^2}{\partial x_k} = \frac{\partial \sum_i x_i^2}{\partial x_k} = \frac{\partial (x_1^2 + x_1^2 + \dots x_n^2)}{\partial x_k} = 2x_k$$

The gradient of the squared L_2 norm in a vector form can be written as :

$$\frac{\partial \|\mathbf{x}\|_2^2}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial \|\mathbf{x}\|_2^2}{\partial x_1} \\ \frac{\partial \|\mathbf{x}\|_2^2}{\partial x_2} \\ \vdots \\ \frac{\partial \|\mathbf{x}\|_2^2}{\partial x_n} \end{bmatrix} = \begin{bmatrix} 2x_1 \\ 2x_2 \\ \vdots \\ 2x_n \end{bmatrix}$$

H)
Just like in the last question, in order to express the gradient of L_1, we must find the derivative of L_1 in respect of each element of $\mathbf{x}$.

$$\frac{\partial \|\mathbf{x}\|_1}{\partial x_k} = \frac{\partial \sum_i |x_i|}{\partial x_k} = \frac{\partial (|x_1| + |x_1| + \dots |x_n|)}{\partial x_k} = \frac{x_k}{|x_k|} \qquad \text{*not differentiable at } x = 0$$

The gradient of the L_1 norm in a vector form can be written as :

$$\frac{\partial \|\mathbf{x}\|_1}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial \|\mathbf{x}\|_1}{\partial x_1} \\ \frac{\partial \|\mathbf{x}\|_1}{\partial x_2} \\ \vdots \\ \frac{\partial \|\mathbf{x}\|_1}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \frac{x_1}{|x_1|} \\ \frac{x_2}{|x_2|} \\ \vdots \\ \frac{x_n}{|x_n|} \end{bmatrix}$$

I) We will proof by contradiction. Let's assume that $S(\mathbf{x})_i = S(c\mathbf{x})_i$.

$$\begin{aligned}
S(\mathbf{x})_i =& S(c\mathbf{x})_i \\
\frac{e^{x_i}}{\sum_j e^{x_j}} =& \frac{e^{cx_i}}{\sum_j e^{cx_j}} \\
\frac{e^{-x_i}}{e^{-x_i}} \frac{e^{x_i}}{\sum_j e^{x_j}} =& \frac{e^{-cx_i}}{e^{-cx_i}} \frac{e^{cx_i}}{\sum_j e^{cx_j}} \\
\frac{1}{e^{-x_i} \sum_j e^{x_j}} =& \frac{1}{e^{-cx_i} \sum_j e^{cx_j}} \\
e^{-x_i} \sum_j e^{x_j} =& e^{-cx_i} \sum_j e^{cx_j} \\
\sum_j e^{x_j} e^{-x_i} =& \sum_j e^{cx_j} e^{-cx_i} \\
\sum_j e^{x_j - x_i} \neq& \sum_j e^{c(x_j - x_i)}
\end{aligned}$$

Therefore $S(\mathbf{x})_i \neq S(c\mathbf{x})_i$, for $c \neq 1$ and if x_j are not all equal.

J)
Showing that the Softmax is translation-invariant :

$$s(x+c)_i = \frac{e^{x_i + c}}{\sum_j e^{x_j + c}} = \frac{e^{x_i} e^c}{\sum_j e^{x_j} e^c} = \frac{e^c e^{x_i}}{e^c \sum_j e^{x_j}} = \frac{e^{x_i}}{\sum_j e^{x_j}} = s(x)_i$$

Therefore, the softmax function is translation invariant.

K)
Let's first calculate the partial derivative when i=j (is a diagonal element) :

$$\begin{aligned}
\frac{\partial \frac{e^{x_i}}{\sum_k e^{x_k}}}{\partial x_j} &= \frac{e^{x_i} \sum_k e^{x_k} - e^{x_j} e^{x_i}}{\left(\sum_k e^{x_k}\right)^2} \quad \text{(Quotient rule)} \\
&= \frac{e^{x_i}}{\sum_k e^{x_k}} \frac{\sum_k e^{x_k} - e^{x_j}}{\sum_k e^{x_k}} \\
&= S(x)_i \left(\frac{\sum_k e^{x_k}}{\sum_k e^{x_k}} - \frac{e^{x_j}}{\sum_k e^{x_k}} \right) \\
&= S(x)_i \left(1 - S(x)_j\right) \\
&= S(x)_i - S(x)_i S(x)_j
\end{aligned}$$

Now, let's calculate the partial derivative when $i \neq j$ (is a off-diagonal element) :

$$\begin{aligned}
\frac{\partial \frac{e^{x_i}}{\sum_k e^{x_k}}}{\partial x_j} &= \frac{0 \sum_k e^{x_k} - e^{x_j} e^{x_i}}{\left(\sum_k e^{x_k}\right)^2} \quad \text{(Quotient rule)} \\
&= \frac{-e^{x_i} e^{x_j}}{\sum_k e^{x_k}} \\
&= -\frac{e^{x_i}}{\sum_k e^{x_k}} \frac{ex^j}{\sum_k e^{x_k}} \\
&= -S(x)_i S(x)_j
\end{aligned}$$

Therefore, $\frac{\partial S(x)_i}{\partial x_j} = S(x)_i 1_{i=j} - S(x)_i S(x)_j$

L)

$$\frac{\partial S(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial S(\mathbf{x})_1}{\partial \mathbf{x}} \\ \frac{\partial S(\mathbf{x})_2}{\partial \mathbf{x}} \\ \vdots \\ \frac{\partial S(\mathbf{x})_n}{\partial \mathbf{x}} \end{bmatrix} = \begin{bmatrix} \frac{\partial S(\mathbf{x})_1}{\partial x_1} & \frac{\partial S(\mathbf{x})_1}{\partial x_2} & \dots & \frac{\partial S(\mathbf{x})_1}{\partial x_n} \\ \frac{\partial S(\mathbf{x})_2}{\partial x_1} & \frac{\partial S(\mathbf{x})_2}{\partial x_2} & \dots & \frac{\partial S(\mathbf{x})_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial S(\mathbf{x})_n}{\partial x_1} & \frac{\partial S(\mathbf{x})_n}{\partial x_2} & \dots & \frac{\partial S(\mathbf{x})_n}{\partial x_n} \end{bmatrix}$$

$$= \begin{bmatrix} S(\mathbf{x})_1(1 - S(\mathbf{x})_1) & S(\mathbf{x})_1(0 - S(\mathbf{x})_2) & \dots & S(\mathbf{x})_1(0 - S(\mathbf{x})_n) \\ S(\mathbf{x})_2(0 - S(\mathbf{x})_1) & S(\mathbf{x})_2(1 - S(\mathbf{x})_2) & \dots & S(\mathbf{x})_2(0 - S(\mathbf{x})_n) \\ \vdots & \vdots & \ddots & \vdots \\ S(\mathbf{x})_n(0 - S(\mathbf{x})_1) & S(\mathbf{x})_n(0 - S(\mathbf{x})_2) & \dots & S(\mathbf{x})_n(1 - S(\mathbf{x})_n) \end{bmatrix}$$

$$= \underbrace{diag(S(\mathbf{x}))}_{n \times n} - \underbrace{\underbrace{S(\mathbf{x})}_{n \times 1} \underbrace{S(\mathbf{x})^T}_{1 \times n}}_{n \times n}$$

We assume that vectors are by default column vectors.

M) Let's start by proving the case for the logistic sigmoid function $\sigma(\mathbf{x})$. We will express the Jacobian matrix $\frac{\partial \sigma(\mathbf{x})}{\partial \mathbf{x}}$.

$$\frac{\partial \sigma(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial \sigma(\mathbf{x}_1)}{\partial \mathbf{x}} \\ \frac{\partial \sigma(\mathbf{x}_2)}{\partial \mathbf{x}} \\ \vdots \\ \frac{\partial \sigma(\mathbf{x}_n)}{\partial \mathbf{x}} \end{bmatrix} = \begin{bmatrix} \frac{\partial \sigma(\mathbf{x}_1)}{\partial x_1} & \frac{\partial \sigma(\mathbf{x}_1)}{\partial x_2} & \dots & \frac{\partial \sigma(\mathbf{x}_1)}{\partial x_n} \\ \frac{\partial \sigma(\mathbf{x}_2)}{\partial x_1} & \frac{\partial \sigma(\mathbf{x}_2)}{\partial x_2} & \dots & \frac{\partial \sigma(\mathbf{x}_2)}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \sigma(\mathbf{x}_n)}{\partial x_1} & \frac{\partial \sigma(\mathbf{x}_n)}{\partial x_2} & \dots & \frac{\partial \sigma(\mathbf{x}_n)}{\partial x_n} \end{bmatrix}$$

$\forall \left(\frac{\partial \sigma(\mathbf{x})}{\partial \mathbf{x}}\right)_{ij}$ where $i \neq j$, we have $\left(\frac{\partial \sigma(\mathbf{x})}{\partial \mathbf{x}}\right)_{ij} = 0$

$\forall \left(\frac{\partial \sigma(\mathbf{x})}{\partial \mathbf{x}}\right)_{ij}$ where $i = j$, we have $\left(\frac{\partial \sigma(\mathbf{x})}{\partial \mathbf{x}}\right)_{ij} = \sigma(x_i)(1 - \sigma(x_i)$ *see Q1b

Therefore, the Jacobian matrix looks like this :

$$\frac{\partial \sigma(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} \sigma(x_1)(1-\sigma(x_1 1) & 0 & \dots & 0 \\ 0 & \sigma(x_2)(1-\sigma(x_2) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma(x_n)(1-\sigma(x_n) \end{bmatrix}$$

$$\nabla_{\mathbf{x}} L = \left(\frac{\partial \sigma(\mathbf{x})}{\partial \mathbf{x}}\right)^{\top} \frac{\partial L}{\partial \mathbf{x}} \quad = \underbrace{diag\left(\frac{\partial \sigma(\mathbf{x})}{\partial \mathbf{x}}\right)}_{n\times 1} \odot \underbrace{\frac{\partial L}{\partial \mathbf{x}}}_{n\times 1}$$

Where $\odot$ is the Hadamard product.
An Hadamard product of two $n \times 1$ vectors takes $O(n)$ computational time.

For the softmax function, the Jacobian is not diagonal like the sigmoid, so there is no simplification with the diag function. Nonetheless there is a simplification possible. Let's compute $\frac{\partial L}{\partial x_k}$, one element of the resulting vector, $\nabla_{\mathbf{x}} L$.

$$\begin{aligned}
\frac{\partial L}{\partial x_k} &= \sum_{i=1}^{n} \frac{\partial S(\mathbf{x})_i}{\partial x_k} \frac{\partial L}{\partial S(\mathbf{x})_i} \\
&= \sum_{i=1}^{n} \underbrace{(S(\mathbf{x})_i \mathbf{1}_{i=k} - S(\mathbf{x})_i S(\mathbf{x})_k)}_{\text{Q1k}} \frac{\partial L}{\partial S(\mathbf{x})_i} \\
&= \sum_{i=1}^{n} S(\mathbf{x})_i \mathbf{1}_{i=k} \frac{\partial L}{\partial S(\mathbf{x})_i} - \sum_{i=1}^{n} S(\mathbf{x})_i S(\mathbf{x})_k \frac{\partial L}{\partial S(\mathbf{x})_i} \\
&= S(\mathbf{x})_k \frac{\partial L}{\partial S(\mathbf{x})_k} - S(\mathbf{x})_k \underbrace{\sum_{i=1}^{n} S(\mathbf{x})_i \frac{\partial L}{\partial S(\mathbf{x})_i}}_{\text{dot product}} \\
&= \underbrace{S(\mathbf{x})_k}_{\text{scalar}} \Bigg(\underbrace{\frac{\partial L}{\partial S(\mathbf{x})_k}}_{\text{scalar}} - \underbrace{\underbrace{S(\mathbf{x})^{\top}}_{1\times n} \underbrace{\frac{\partial L}{\partial S(\mathbf{x})}}_{n\times 1}}_{\text{scalar}} \Bigg) \\
\nabla_{\mathbf{x}} L &= S(\mathbf{x}) \odot \left(\frac{\partial L}{\partial S(\mathbf{x})} - S(\mathbf{x})^{\top} \frac{\partial L}{\partial S(\mathbf{x})} \mathbf{1}_{n,1} \right)
\end{aligned}$$

We need to compute the dot product of $S(\mathbf{x})^{\top} \frac{\partial L}{\partial S(\mathbf{x})}$ only one time since it's the same for each element k of $\nabla_{\mathbf{x}} L$. Then it's constant time complexity operations and finally a Hadamard product of two $n \times 1$ vectors. The complexity is reduce to $O(n+n) \rightarrow O(n)$.

Question 2

A) The dimension of $b^{(1)}$ a vector of dimension $d_h \times 1$ which is equal to the number of neurons of the hidden layer.

The formula for the pre-activation vector can be expressed as (matrix form) - (where $P^{(0)}(x)$ is the post activation of the first (input layer) layer) :

$$h^a = W^{(1)}x + b^{(1)}$$

To calculate the specific element h^a_j, we get :

$$h^a_j = b^{(1)}_j + \sum_i W^{(1)}_{j,i} x_i$$

Each elements of the output vector of the hidden layer h^s can be written as :

$$h^s_j = \begin{cases} h^a_j & \text{if } h^a_j > 0 \\ \alpha(e^{h^a_j} - 1) & \text{otherwise} \end{cases}$$

In vector form, it is :

$$h^s = ELU_\alpha(h^a)$$

B) The dimension of $W^{(2)}$ is of dimension $m \times d_h$, which is the dimension of the output layer times the dimension of the hidden layer. The dimension of $b^{(2)}$ is a vector of size $m \times 1$.
The output layer before activation (pre-activation) is calculated (in matrix form) as :

$$o^a = W^{(2)}h^s + b^{(2)}$$

We can also write it in detailed form for o^a_k as :

$$o^a_k = b^{(2)}_k + \sum_j W^{(2)}_{k,j} h^s_j$$

C) We can write o^s_k as a function of o^a_k and o^a_j.

$$\begin{aligned} o^s_k &= \text{softmax}(\mathbf{o}^a)_k \\ &= \frac{\exp(o^a_k)}{\sum_{j=1}^m \exp(o^a_j)} \end{aligned}$$

All o^s_k are positive because the exponential function is stricly positive : $e^x >$

$0 \quad \forall x \in R$. The softmax() function is a quotient of a exponential function (positive) by a sum of exponential functions (sum of positive values results in a positive value). The quotient of two positive values is also positive. Now let's prove that $\sum_k o_k^s = 1$.

$$
\begin{aligned}
\sum_k o_k^s &= \sum_k \text{softmax}(\mathbf{o}^a)_k \\
&= \sum_k \frac{\exp(o_k^a)}{\sum_j \exp(o_j^a)} \\
&= \frac{\exp(o_1^a)}{\sum_j \exp(o_j^a)} + \frac{\exp(o_2^a)}{\sum_j \exp(o_j^a)} + \dots \frac{\exp(o_m^a)}{\sum_j \exp(o_j^a)} \\
&= \frac{\sum_j \exp(o_j^a)}{\sum_j \exp(o_j^a)} \\
&= 1
\end{aligned}
$$

The fact that $\sum_k o_k^s = 1$ is useful because it acts as a probability distribution over the different possible outcomes. By probability theory the sum of a discrete probability function must equals to 1.

D)

$$
L(x,y) = -\log o_y^s(x) = -\log\left(\frac{e^{o_y^a}}{\sum_{i=1} d_o e^{o_i^a}}\right) = -o_y^a + \log(\sum_{i=1}^{d_o} e^{o_i^a})
$$

E)

$$
\hat{R}(\theta) = \sum_D L(x^{(i)}, y^{(i)})
$$

The parameters θ are the two sets of W and b connecting the input layer to the hidden layer and the hidden layer to the output layer. In total, there are $n_{theta} = d_h d + d_h + d_o d_h + d_o$. The optimization problem is

$$
\arg\ \min_\theta \hat{R}(\theta)
$$

$$
\hat{R}(\theta) = \sum_D L(x^{(i)}, y^{(i)})
$$

The parameters θ are the two sets of W and b connecting the input layer to the hidden layer and the hidden layer to the output layer. In total, there are $n_{theta} = d_h d + d_h + d_o d_h + d_o$. The optimization problem is

$$
\arg\ \min_\theta \hat{R}(\theta)
$$

F)

$$\theta_{t+1} = \theta_t - \eta \nabla \hat{R}(\theta)$$

G) For this problem, we have to show that :

$$\frac{\partial L}{\partial o^a} = o^s - \text{onehot}_m(y)$$

Using the expression of L as a function of o_k^a We have :

$$L(o^a, y) = -log(e^{o_y^a}) + log(\sum_j e^{o_j^a}) = log(\sum_j e^{o_j^a}) - o_y^a$$

When $k \neq y$:

$$\begin{aligned}\frac{\partial L}{\partial o_k^a} &= \frac{1}{\sum_j e^{o_j^a}} \frac{\partial \sum_j e^{o_j^a}}{\partial o_k^a} - 0 \\ &= \frac{e^{o_k^a}}{\sum_j e^{o_j^a}} = o_k^s\end{aligned}$$

When $k = y$:

$$\begin{aligned}\frac{\partial L}{\partial o_k^a} &= \frac{1}{\sum_j e^{o_j^a}} \frac{\partial \sum_j e^{o_j^a}}{\partial o_k^a} - \frac{\partial o_y^a}{\partial o_y^a} \\ &= \frac{e^{o_k^a}}{\sum_j e^{o_j^a}} - 1 = o_k^s - 1\end{aligned}$$

Therefore, we can see that there is a minus one only when $k = y$, which means we can express :

$$\frac{\partial L}{\partial o^a} = o^s - \text{onehot}_m(y)$$

H)

$$\frac{\partial L}{\partial \mathbf{W}_{kj}^{(2)}} = \frac{\partial L}{\partial \mathbf{o}_k^a} \mathbf{h_j^s}$$

$$\frac{\partial L}{\partial \mathbf{b}_k^{(2)}} = \frac{\partial L}{\partial \mathbf{o}_k^a}$$

I)

$$\frac{\partial L}{\partial \mathbf{W}^{(2)}} = \frac{\partial L}{\partial \mathbf{o}^a} (\mathbf{h^s})^T$$

$$\frac{\partial L}{\partial \mathbf{b}^{(2)}} = \frac{\partial L}{\partial \mathbf{o}^a}$$

where $\frac{\partial L}{\partial \mathbf{o}^a}$ is a $d_o \times 1$ vector and h^s is a $d_h \times 1$ vector.

```
grad\_W2 = grad\_oa * hs.T \\
grad\_b2 = grad\_oa
```

J)
The partial derivative of the loss L with respect to the ouput of the neurons at the hidden layer is :

$$\begin{aligned}
\frac{\partial L}{\partial h^s_j} &= \sum_{k=1}^{m} \frac{\partial L}{\partial o^a_k} \frac{\partial o^a_k}{\partial h^s_j} \\
&= \sum_{k=1}^{m} \frac{\partial L}{\partial o^a_k} \frac{\partial \left(b^{(2)}_k + \sum_j W^{(2)}_{k,j} h^s_j \right)}{\partial h^s_j} \\
&= \sum_{k=1}^{m} \frac{\partial L}{\partial o^a_k} W^{(2)}_{k,j}
\end{aligned}$$

K)

$$\frac{\partial L}{\partial \mathbf{h}^s_j} = (\mathbf{W}^{(2)})^T \frac{\partial L}{\partial \mathbf{o}^a}$$

```
grad_hs = w2.T * grad_oa
```

where the dimension of $(\mathbf{W}^{(2)})^T$ is of size $d_h \times m$, $\frac{\partial L}{\partial o^a}$ is also a vector of size m $\times$ 1 and $\frac{\partial L}{\partial h^s}$ is a vector of size $d_h \times 1$.

L) Starting by calculating the derivative of the ELU function :

$$\begin{aligned}
\frac{\partial ELU_\alpha(z)}{\partial z} &= ELU^{'}_\alpha(z) \\
&= \begin{cases} 1 & \text{if } z > 0 \\ \alpha e^z & \text{if } z < 0 \end{cases}
\end{aligned}$$

Note that the derivative at zero only exists (and is 1) if $\alpha = 1$.
Now, calculating the partial derivative with respect to the. activation of the neurons at the hidden layer :

$$\begin{aligned}
\frac{\partial L}{\partial \mathbf{h}^a_j} &= \frac{\partial L}{\partial \mathbf{h}^s_j} \frac{\partial \mathbf{h}^s_j}{\partial \mathbf{h}^a_j} \\
&= \frac{\partial L}{\partial \mathbf{h}^s_j} \frac{\partial ELU\alpha(\mathbf{h}^a_j)}{\partial \mathbf{h}^a_j}
\end{aligned}$$

Which now breaks the case in 2, the first one being that $\mathbf{h}_j^a > 0$, which leads to the equation :

$$\begin{aligned}\frac{\partial L}{\partial \mathbf{h}_j^a} &= \frac{\partial L}{\partial \mathbf{h}_j^s} \frac{\partial ELU\alpha(\mathbf{h}_j^a)}{\partial \mathbf{h}_j^a} \\ &= \frac{\partial L}{\partial \mathbf{h}_j^s} \times 1\end{aligned}$$

and in the case that $h_j^a < 0$, it leads to the equation :

$$\begin{aligned}\frac{\partial L}{\partial \mathbf{h}_j^a} &= \frac{\partial L}{\partial \mathbf{h}_j^s} \frac{\partial ELU\alpha(\mathbf{h}_j^a}{\partial \mathbf{h}_j^a} \\ &= \frac{\partial L}{\partial \mathbf{h}_j^s} \times \alpha e^{\mathbf{h}_j^a}\end{aligned}$$

M) The gradient of the last equation can be written as :

$$\frac{\partial L}{\partial \mathbf{h}^a} = \frac{\partial L}{\partial \mathbf{h}^s} \odot ELU_\alpha^{'}(\mathbf{h}^a)$$

Where $\odot$ represents the element wise product and $ELU_\alpha^{'}(\mathbf{h}^a)$, is element wise calculation of the derivative of the ELU for all components of vector $\mathbf{h}^a$.

The dimension of $\frac{\partial L}{\partial \mathbf{h}^s}$ is a vector of size $d_h \times 1$, (calcualted above).
The dimension of $ELU_\alpha^{'}(\mathbf{h}^a)$ is a vector of size $d_h \times 1$.
The dimension of $\frac{\partial L}{\partial \mathbf{h}^a}$ is a vector of size $d_h \times 1$.

N) The gradient of the loss with respect to $\mathbf{W}^{(1)}$ is :

$$\begin{aligned}\frac{\partial L}{\partial \mathbf{W}_{ji}^{(1)}} &= \frac{\partial L}{\partial \mathbf{h}_j^a} \frac{\partial \mathbf{h}_j^a}{\partial \mathbf{W}_{ji}^{(1)}} \quad \text{Chain rule} \\ &= \frac{\partial L}{\partial \mathbf{h}_j^a} \frac{\partial \left(\mathbf{b}_j^{(1)} + \sum_l \mathbf{W}_{j,l}^{(1)} x_l\right)}{\partial \mathbf{W}_{ji}^{(1)}} \\ &= \frac{\partial L}{\partial \mathbf{h}_j^a} x_i\end{aligned}$$

O) The gradient of previous question can be written like :

$$\frac{\partial L}{\partial \mathbf{W}^{(1)}} = \frac{\partial L}{\partial \mathbf{h}^a} x^T$$

$$\frac{\partial L}{\partial \mathbf{b}^{(1)}} = \frac{\partial L}{\partial \mathbf{h}^a} \times 1$$

The dimension of $\frac{\partial L}{\partial \mathbf{h}^a}$ is a vector of size $d_h \times 1$.
The dimension of x^T is a vector of size $1 \times d$.
The dimension of $\frac{\partial L}{\partial \mathbf{W}^{(1)}}$ is therefore a matrix of size $d_h \times d$.
The dimension of $\frac{\partial L}{\partial \mathbf{b}^{(1)}}$ is a vector of size $d_h \times 1$

P)

$$\frac{\partial L}{\partial \mathbf{x}_j} = \sum_{k=1}^{m} \frac{\partial L}{\partial \mathbf{h}^a_k} \mathbf{W}^{(1)}_{kj}$$

$$\frac{\partial L}{\partial \mathbf{x}} = (\mathbf{W}^{(1)})^T \frac{\partial L}{\partial \mathbf{h}^a}$$

Question 3

1. (a) Let's describe each layers of the CNN and compute the output size of each.

   Input layer :
   $3 \times 128 \times 128$

   First layer :
   Convolution layer
   in (width = height) : 128
   convolutions : 32
   kernel size (k) : 8×8
   stride : 2
   padding : 0
   dilatation : 1 (no dilatation)
   out : $\mathbf{32} \times \mathbf{61} \times \mathbf{61}$

$$\begin{aligned} \text{out} &= \left\lfloor \frac{\text{in} + 2p - d(k-1) - 1}{s} \right\rfloor + 1 \\ &= \left\lfloor \frac{128 + 2(0) - 1(8-1) - 1}{2} \right\rfloor + 1 = 61 \end{aligned}$$

   Second layer :
   Max pooling layer
   in (width = height) : 61
   kernel size (k) : 5×5
   stride : 5 (no overlapping)
   padding : 0

dilatation : 1 (no dilatation)
out : $\mathbf{32 \times 12 \times 12}$

$$\begin{aligned} \text{out} &= \left\lfloor \frac{\text{in} + 2p - d(k-1) - 1}{s} \right\rfloor + 1 \\ &= \left\lfloor \frac{61 + 2(0) - 1(5-1) - 1}{5} \right\rfloor + 1 = 12 \end{aligned}$$

Third layer :
Convolution layer
in (width = height) : 12
convolutions : 128
kernel size (k) : 4×4
stride : 1
padding : 1
dilatation : 1 (no dilatation)
out : $\mathbf{128 \times 11 \times 11}$

$$\begin{aligned} \text{out} &= \left\lfloor \frac{\text{in} + 2p - d(k-1) - 1}{s} \right\rfloor + 1 \\ &= \left\lfloor \frac{12 + 2(1) - 1(4-1) - 1}{1} \right\rfloor + 1 = 11 \end{aligned}$$

There is $128 \times 11 \times 11 = 15488$ dimensions (scalars) for this third and last layer. **Note : According to our previous announcement, we accept both answers with/without using the floor operator and instead adding a padding = 2 in the max-pooling layer, to turn the fraction into an integer.**

(b) We can calculate the number of parameters (without biases) of the last layer as the number of kernels : 128, multiplied by the size of each kernel : 4×4 and finally multiplied by the number of feature maps of the layer above : 32. Therefore, the number of parameters of the last layer can be expressed as :

$$p = 128 \times 4 \times 4 \times 32 = 65536$$

2. Next,

(a) Assuming $p = 0$, $d = 1$

$$\begin{aligned} \text{out} &= \left\lfloor \frac{\text{in} + 2p - d(k-1) - 1}{s} \right\rfloor + 1 \\ 4 &= \left\lfloor \frac{64 + 2(0) - 1(k-1) - 1}{s} \right\rfloor + 1 \\ 3 &= \left\lfloor \frac{64 - k}{s} \right\rfloor \\ &\text{possible answer} \rightarrow k = 49, s = 5 \end{aligned}$$

$k = 49$, $s = 5$, $p = 0$, $d = 1$

(b) Assuming $d = 2$, $p = 2$

$$\text{out} = \left\lfloor \frac{\text{in} + 2p - d(k-1) - 1}{s} \right\rfloor + 1$$
$$4 = \left\lfloor \frac{64 + 2(2) - 2(k-1) - 1}{s} \right\rfloor + 1$$
$$3 = \left\lfloor \frac{64 + 4 - 2k + 2 - 1}{s} \right\rfloor$$
$$3 = \left\lfloor \frac{69 - 2k}{s} \right\rfloor$$
$$\text{possible answer} \rightarrow k = 27, s = 5$$

$k = 27$, $s = 5$, $p = 2$, $d = 2$

(c) Assuming $p = 1$, $d = 1$

$$\text{out} = \left\lfloor \frac{\text{in} + 2p - d(k-1) - 1}{s} \right\rfloor + 1$$
$$4 = \left\lfloor \frac{64 + 2(1) - 1(k-1) - 1}{s} \right\rfloor + 1$$
$$3 = \left\lfloor \frac{64 + 2 - k + 1 - 1}{s} \right\rfloor$$
$$3 = \left\lfloor \frac{66 - k}{s} \right\rfloor$$
$$\text{possible answer} \rightarrow k = 51, s = 5$$

$k = 51$, $s = 5$, $p = 1$, $d = 1$

Note : We accept all the possible solutions of k,s,p,d as long as they satisfy the equation.