Instructions

- *This homework is* **not** *to be handed in or given a grade.*
- *There will be a* **quiz in class** *: Section B : 25/02/2026, Section A : 26/02/2026*
- *TA for this assignment is (theoretical part) :* ***Armand Collin****.*

Question 1. Recall the definition of the softmax function : $S(\boldsymbol{x})_i = e^{\boldsymbol{x}_i} / \sum_j e^{\boldsymbol{x}_j}$.

1. Show that softmax is translation-invariant, that is : $S(\boldsymbol{x}+c) = S(\boldsymbol{x})$, where c is a scalar constant.
2. Let $\boldsymbol{x}$ be a 2-dimensional vector. One can represent a 2-class categorical probability using softmax $S(\boldsymbol{x})$. Show that $S(\boldsymbol{x})$ can be reparameterized using sigmoid function, i.e. $S(\boldsymbol{x}) = [\sigma(z), 1-\sigma(z)]^\top$ where z is a scalar function of $\boldsymbol{x}$.
3. Let $\boldsymbol{x}$ be a K-dimensional vector ($K \geq 2$). Show that $S(\boldsymbol{x})$ can be represented using $K-1$ parameters, i.e. $S(\boldsymbol{x}) = S([0, y_1, y_2, ..., y_{K-1}]^\top)$, where y_i is a scalar function of $\boldsymbol{x}$ for $i \in \{1, ..., K-1\}$.
4. Show that the Jacobian of the softmax function $J_{\text{softmax}}(\boldsymbol{x})$ can be expressed as : $\mathbf{Diag}(\boldsymbol{p}) - \boldsymbol{p}\boldsymbol{p}^\top$, where $\boldsymbol{p} = S(\boldsymbol{x})$.

Question 2. Consider the differentiable functions $f : \mathbb{R}^\ell \to \mathbb{R}^m$, $g : \mathbb{R}^m \to \mathbb{R}^n$, and $h : \mathbb{R}^n \to \mathbb{R}^o$. Let $F : \mathbb{R}^\ell \to \mathbb{R}^o$ be the composition of these functions, i.e. $F = h \circ g \circ f$. For this question, denote the Jacobian matrix of F evaluated at $x \in \mathbb{R}^\ell$ as $J_F(x) \in \mathbb{R}^{o \times \ell}$, and analogously for any other function.

1. Using the chain rule, express $J_F(x)$ using J_f, J_g and J_h. Your answer should be in matrix form. In your expression, make sure it is clear at which point each Jacobian matrix is evaluated.
2. Provide a simple pseudo code which computes $J_F(x)$ using *forward mode accumulation* (Section 6.5.9 in DL book) given x. You can call the functions f, g, h, J_f, J_g and J_h **only once each** (to maximize efficiency). You can call the function `matmul`$(\cdot, \cdot)$ which performs matrix multiplication (no limit on the number of calls). Your pseudo code should also return $F(x)$.
3. Provide a simple pseudo code which computes $J_F(x)$ in *reverse mode accumulation* (Section 6.5.9 in DL book) given x. You can call the functions f, g, h, J_f, J_g and J_h **only once each** (to maximize efficiency). You can call the function `matmul`$(\cdot, \cdot)$ which performs matrix multiplication (no limit on the number of calls). Your pseudo code should also return $F(x)$.
4. Assume evaluating f and J_f cost $O(\ell m)$, evaluating g and J_g cost $O(mn)$ and evaluating h and J_h cost $O(no)$. What is the time complexity of your forward mode pseudo code ? your reverse mode pseudo code ?

Question 3. In this question, we will explore the convergence of the gradient descent algorithm.

1. **Convex Function Convergence** Consider the following function:

$$f(x) = \begin{cases} \frac{3}{4}(1-x)^2 - 2(1-x) & \text{if } x > 1 \\ \frac{3}{4}(1+x)^2 - 2(1+x) & \text{if } x < -1 \\ x^2 - 1 & \text{otherwise} \end{cases} \tag{1}$$

   Show that f is a convex function. Find its unique minimizer and its gradient. Consider the following algorithm : $x_t = x_{t-1} - \eta f'(x_{t-1})$ where $\eta = 1$. Will this algorithm converge to a stationary point if it starts at point x_0, where $x_0 > 1$? Why or why not?

2. **Prove Convergence of Gradient Descent to Stationary Point in Non-Convex case** Suppose we are trying to minimize the function $F(w)$ that is L-smooth. Let F_* be the minimal function value (i.e. the value at the global minima). Using $\eta = \frac{1}{L}$, prove that gradient descent will "almost" converge to a stationary point in a bounded (and polynomial) number of steps. Precisely,

$$\min_{k<K} \|\nabla F(w^{(k)})\|^2 \leq \frac{2L}{K}(F(w^{(0)}) - F_*) \tag{2}$$

   Hints:

   (a) L-smoothness implies that:

$$F(w^{(k+1)}) \leq F(w^{(k)}) - \eta\|\nabla F(w^{(k)})\|^2 + \frac{1}{2}\eta^2 L\|\nabla F(w^{(k)})\|^2 \tag{3}$$

   Combine this with $\eta = \frac{1}{L}$

   (b) Use the fact that the minimum of a sequence of elements is less or equal than the average of the sequence.

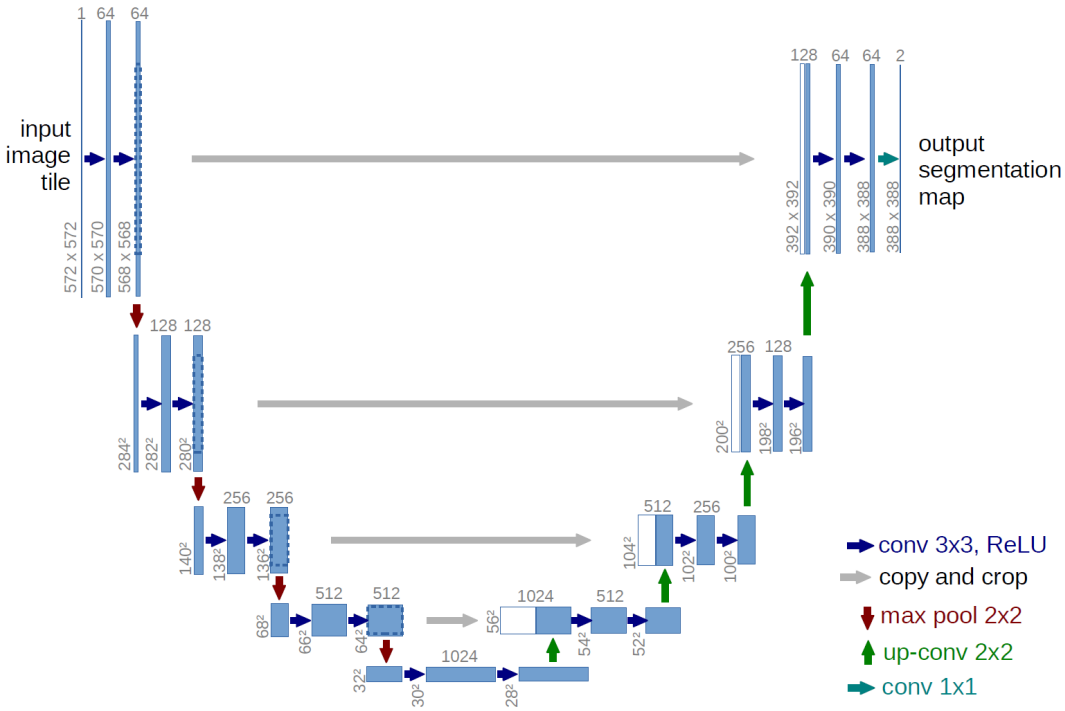

FIGURE 1 – (Borrowed from the U-Net paper : `https://arxiv.org/pdf/1505.04597`) This architecture consists of a contracting path (encoder), a bottleneck at the bottom and an expansive path (decoder). The model depicted has 4 stages. Every stage of the encoder doubles the number of feature channels, whereas every stage of the decoder halves them. The decoder receives the input of the corresponding encoder block at every stage via a copy-and-crop operation.

Question 4. The U-Net is a popular architecture for biomedical image segmentation. Suppose an input RGB image of size $444 \times 444 \times 3$. We want to implement a 3-stage U-Net to process this data and output segmentation maps with 5 classes. *NOTE : These dimensions are different than in Fig. 1*

1. **Encoder dimensions :** Complete Table 1 by specifying the output shape and the number of parameters of every layer in the encoder and bottleneck. Layers are expressed as follows :
   - CONV(H,C) : An unpadded convolution layer with filters of size $H \times H$ with C channels and appropriate weights and biases (stride 1).
   - RELU : ReLU activation.
   - MAXPOOLING(N) : $N \times N$ max pooling (stride 2).
2. **Skip connections :** The U-Net has skip connections between the encoder and decoder. Feature maps from the encoder are center-cropped and concatenated with the up-sampled feature maps. Does the inclusion of these skip connections increase the total number of learnable parameters compared to a network with the same architecture but no skip connections ? Explain why or why not.
3. **Output shape :** What are the spatial dimensions $H_O \times W_O$ of the network's final output, after the decoder and the 1×1 convolution ? *Hint : The* up-conv 2x2 *operation doubles spatial dimensions and halves the number of channels.*

Layer	Output Dimensions	Number of Parameters
Encoder Stage 1		
Input	$444 \times 444 \times 3$	0
Conv(3,64) + ReLU		
Conv(3,64) + ReLU		
MaxPooling(2)		
Encoder Stage 2		
Conv(3,128) + ReLU		
Conv(3,128) + ReLU		
MaxPooling(2)		
Encoder Stage 3		
Conv(3,256) + ReLU		
Conv(3,256) + ReLU		
MaxPooling(2)		
Bottleneck		
Conv(3,512) + ReLU		
Conv(3,512) + ReLU		

Table 1 – Table to fill out for part 1.

4. **Inference :** Notice that the output segmentation map is smaller than the input size. To apply our model to images of arbitrary size, during inference, image tiles are sampled with overlap, and the model is applied to every tile. At the image borders, the image is mirrored to simulate missing context (see Fig. 2 in the U-Net paper). Suppose we have an image of size 1068×712, and we sample tiles with an 88-pixel overlap. How many tiles are required to apply our model ?
5. **Conditioning the network :** U-Nets are also widely used in Denoising Diffusion Probabilistic Models (DDPM), which will be covered later in class. In these applications, the network must process the image differently depending on a "time-step" t (a scalar indicating the noise level). To achieve this, we embed t into a vector v_t of size 42. We want to inject this vector into the output of the first convolution of each Encoder stages (1, 2, 3) and the Bottleneck. For each stage, the process is :
   (a) Pass v_t through a learnable Linear (Fully Connected) layer.
   (b) Reshape the output of the Linear layer to shape $(1, C, 1, 1)$.
   (c) Add this result element-wise to the feature map (broadcasting over the spatial dimensions).

   Based on your answer in Table 1, specify the input and output dimensions of the Linear layer required for each of the four locations (Encoder 1, 2, 3 and Bottleneck).