TÉCNICO LISBOA

Deep Learning

MSc in Computer Science and Engineering

MSc in Electrical and Computer Engineering

Final exam — January 25, 2023

Version A

Instructions

- You have 120 minutes to complete the exam.
- Make sure that your test has a total of 14 pages and is not missing any sheets, then write your full name and student n. on this page (and your number in all others).
- The test has a total of 17 questions, with a maximum score of 100 points. The questions have different levels of difficulty. The point value of each question is provided next to the question number.
- Please provide your answer in the space below each question. If you make a mess, clearly indicate your answer.
- The exam is open book and open notes. You may use a calculator, but any other type of electronic or communication equipment is not allowed.
- Good luck.

Part 1	Part 2	Part 3, Pr. 1	Part 3, Pr. 2	Total
32 points	18 points	25 points	25 points	100 points

Part 1: Multiple Choice Questions (32 points)

In each of the following questions, indicate your answer by *checking a single option.*

1. (4 points) The softplus function is defined as $\text{softplus}(t) = \log(1 + \exp(t))$. The logistic function is $\sigma(t) = 1/(1 + \exp(-t))$. Which of the following options is the derivative of softplus at t?

   - ☐ $\sigma(t)(1 - \sigma(t))$
   - ■ $\sigma(t)$
   - ☐ $\sigma(-t)$
   - ☐ $\text{softplus}(t)(1 - \text{softplus}(t))$

   Solution: The derivative of softplus is the logistic function, $\sigma(t) = 1/(1 + \exp(-t))$.

2. (4 points) Which of the following pairs of functions always give the same result when commuted?

   - ■ **ReLU activations and max-pooling layer.**
   - ☐ Convolutional layer and max-pooling layer.
   - ☐ Convolutional layer and ReLU activations.
   - ☐ None of the above.

   Solution: Applying ReLU activations followed by a max-pooling layer gives the same result as a max-pooling layer followed by ReLU activations.

3. (4 points) Let $\mathcal{D} = \{(\boldsymbol{x}^{(i)}, y^{(i)})_{i=1}^N\}$ be a linearly separable dataset for binary classification, and let $\mathcal{D}' = \{(2\boldsymbol{x}^{(i)}, y^{(i)})_{i=1}^N\}$ be a scaled version of the same dataset where the input vectors are scaled by a factor of 2. Consider the perceptron algorithm without a bias parameter and where all the weights are initialized to zero. On which of the datasets the perceptron will make more mistakes until it converges?

   - ☐ The number of mistakes can be larger for $\mathcal{D}$ than for $\mathcal{D}'$.
   - ☐ The number of mistakes can be smaller for $\mathcal{D}$ than for $\mathcal{D}'$.
   - ■ **The number of mistakes must be the same for both datasets.**
   - ☐ There can be infinitely many mistakes for both datasets.

   Solution: For $\mathcal{D}$, the updates will be $\boldsymbol{w} \leftarrow \boldsymbol{w} + y^{(i)}\boldsymbol{x}^{(i)}$. For $\mathcal{D}'$, the updates will be $\boldsymbol{w} \leftarrow \boldsymbol{w} + 2y^{(i)}\boldsymbol{x}^{(i)}$. Since $\boldsymbol{w}$ is initialized to 0, the mistakes will be exactly the same for both datasets and the final $\boldsymbol{w}$ for $\mathcal{D}'$ will be twice the final $\boldsymbol{w}$ for $\mathcal{D}$.

4. (4 points) Assume a transformer with two attention heads, one with projection matrices $\boldsymbol{W}_Q^{(1)}, \boldsymbol{W}_K^{(1)}, \boldsymbol{W}_V^{(1)}$ and another with projection matrices $\boldsymbol{W}_Q^{(2)}, \boldsymbol{W}_K^{(2)}, \boldsymbol{W}_V^{(2)}$ where $\boldsymbol{W}_Q^{(2)} = -\boldsymbol{W}_Q^{(1)}$, $\boldsymbol{W}_K^{(2)} = -\boldsymbol{W}_K^{(1)}$, and $\boldsymbol{W}_V^{(2)} = -\boldsymbol{W}_V^{(1)}$. Let $\boldsymbol{P}^{(1)}$ and $\boldsymbol{P}^{(2)}$ be the attention probability matrices and $\boldsymbol{Z}^{(1)}$ and $\boldsymbol{Z}^{(2)}$ be the context representations obtained by the two attention heads. What is the relation between $\boldsymbol{P}^{(1)}$ and $\boldsymbol{P}^{(2)}$ and between $\boldsymbol{Z}^{(1)}$ and $\boldsymbol{Z}^{(2)}$?

   - ☐ $\boldsymbol{P}^{(1)} = \boldsymbol{P}^{(2)}$ and $\boldsymbol{Z}^{(1)} = \boldsymbol{Z}^{(2)}$.
   - ☐ $\boldsymbol{P}^{(1)} = -\boldsymbol{P}^{(2)}$ and $\boldsymbol{Z}^{(1)} = \boldsymbol{Z}^{(2)}$.

- ■ $\boldsymbol{P}^{(1)} = \boldsymbol{P}^{(2)}$ **and** $\boldsymbol{Z}^{(1)} = -\boldsymbol{Z}^{(2)}$**.**
- □ None of the above.

Solution: We have $\boldsymbol{P}^{(2)} = \text{Softmax}(\boldsymbol{Q}^{(2)}(\boldsymbol{K}^{(2)})^\top/\sqrt{K}) = \text{Softmax}(-\boldsymbol{Q}^{(1)}(-\boldsymbol{K}^{(1)})^\top/\sqrt{K}) = \text{Softmax}(\boldsymbol{Q}^{(1)}(\boldsymbol{K}^{(1)})^\top/\sqrt{K}) = \boldsymbol{P}^{(1)}$. We have $\boldsymbol{Z}^{(2)} = \boldsymbol{P}^{(2)}\boldsymbol{V}^{(2)} = -\boldsymbol{P}^{(1)}\boldsymbol{V}^{(1)} = -\boldsymbol{Z}^{(1)}$.

5. (4 points) Consider a neural network layer with preactivation $\boldsymbol{z}^{(\ell)} = \mathbf{W}\boldsymbol{h}^{(\ell-1)} + \boldsymbol{b}$ and activation $\boldsymbol{h}^{(\ell)} = \boldsymbol{g}(\boldsymbol{z}^{(\ell)})$ where the activation function $g(z_i) = \log(1 + \text{relu}(z_i))$ is applied elementwise. Let L be the loss function associated to this neural network. Which of these expressions is the correct derivative of L with respect to $\boldsymbol{h}^{(\ell-1)}$?

- ☐ $\frac{\partial L}{\partial \boldsymbol{h}^{(\ell-1)}} = \boldsymbol{W}^\top \left(\frac{\partial L}{\partial \boldsymbol{h}^{(\ell)}} \odot 1(\boldsymbol{z}^{(\ell)} \geq \mathbf{0}) \right)$.
- ■ $\frac{\partial L}{\partial \boldsymbol{h}^{(\ell-1)}} = \boldsymbol{W}^\top \left(\frac{\partial L}{\partial \boldsymbol{h}^{(\ell)}} \odot \frac{1(\boldsymbol{z}^{(\ell)} \geq \mathbf{0})}{1+\text{relu}(\boldsymbol{z}^{(\ell)})} \right)$.
- ☐ $\frac{\partial L}{\partial \boldsymbol{h}^{(\ell-1)}} = \boldsymbol{W}^\top \left(\frac{\partial L}{\partial \boldsymbol{h}^{(\ell)}} \odot (1 - (\boldsymbol{h}^{(\ell)})^2) \right)$.
- ☐ $\frac{\partial L}{\partial \boldsymbol{h}^{(\ell-1)}} = \boldsymbol{W}^\top \frac{\partial L}{\partial \boldsymbol{h}^{(\ell)}}$.

Solution: We have $\frac{\partial L}{\partial z_i^{(\ell)}} = \frac{\partial L}{\partial h_i^{(\ell)}} g'(z_i^{(\ell)})$, i.e., $\frac{\partial L}{\partial \boldsymbol{z}^{(\ell)}} = \frac{\partial L}{\partial \boldsymbol{h}^{(\ell)}} \odot \boldsymbol{g}'(\boldsymbol{z}^{(\ell)})$ and $\frac{\partial L}{\partial h_i^{(\ell-1)}} = \sum_j \frac{\partial L}{\partial z_j^{(\ell)}} W_{ji}$, i.e., $\frac{\partial L}{\partial \boldsymbol{h}^{(\ell-1)}} = W^\top \frac{\partial L}{\partial \boldsymbol{z}^{(\ell)}}$. We have $g'(z_i) = \frac{1(z_i \geq 0)}{1+\text{relu}(z_i)}$, i.e., $\boldsymbol{g}'(\boldsymbol{z}) = \frac{1(\boldsymbol{z} \geq \mathbf{0})}{1+\text{relu}(\boldsymbol{z})}$. Therefore:

$$\frac{\partial L}{\partial \boldsymbol{h}^{(\ell-1)}} = \boldsymbol{W}^\top \left(\frac{\partial L}{\partial \boldsymbol{h}^{(\ell)}} \odot \boldsymbol{g}'(\boldsymbol{z}^{(\ell)}) \right) = \boldsymbol{W}^\top \left(\frac{\partial L}{\partial \boldsymbol{h}^{(\ell)}} \odot \frac{1(\boldsymbol{z}^{(\ell)} \geq \mathbf{0})}{1 + \text{relu}(\boldsymbol{z}^{(\ell)})} \right).$$

6. (4 points) Assume an auto-encoder that learns representations of 28×28 images using 3 hidden layers of sizes 250, 75, 250, respectively. Which of the following statements about the architecture of the auto-encoder is correct?

- ☐ The output layer has K (number of classes) units; the latent representation has dimension 75; the loss function of the auto-encoder is the cross-entropy loss.
- ■ **The output layer has 784 units; the latent representation has dimension 75; the loss function is the mean squared error.**
- ☐ The output layer has 784 units; the latent representation has dimension 75; the loss function is the cross-entropy loss.
- ☐ The output layer has 784 units; the latent representation has dimension 250; the loss function is the mean squared error.

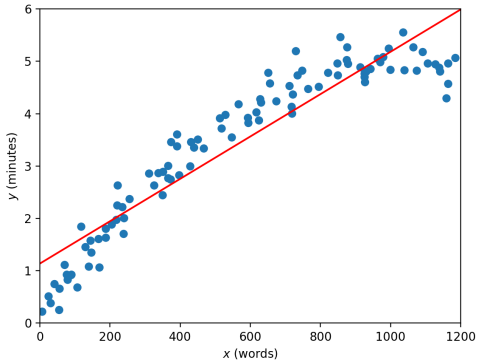
Solution: We train the auto-encoder on the task of predicting the same image it received as input, *i.e.*, on the regression task of predicting the value of each pixel value, therefore the output layer will have 784 units and the loss function will be the mean squared error. The learned representation will be encoded in the network's bottleneck: 75 units.

7. (4 points) Which of these layers computes its output using the dot product between patches of the input and different filters?

- ■ **Convolution layer.**
- ☐ Max-pooling layer.
- ☐ Fully-connected layer.
- ☐ None of the above.

Solution: The convolution layer does exactly what is described.

8. (4 points) The figure below represents a regression model (in red) fit to a given training dataset by minimizing the mean squared error loss. Can you describe what is happening and how would you fix it?



- ☐ Underfitting. Utilize dropout.
- ☐ Overfitting. Utilize ℓ_2-regularization.
- ■ **Underfitting. Increase the complexity of the model.**
- ☐ Overfitting. Train the model for a longer duration.

Solution: We need to increase the complexity of the model (e.g. the degree of the polynomial), since the model is underfitting the data.

Part 2: Short Answer Questions (18 points)

Please provide **brief** answers (1-2 sentences) to the following questions.

1. (6 points) What is exposure bias in sequence-to-sequence models and what causes it?

   Solution: Exposure bias is the tendency of sequence-to-sequence models to be exposed only to correct target sequence prefixes at training time, and never to their own predictions. This makes them having trouble to recover from their own incorrect predictions at test time, if they are produced early on in the sequence. Exposure bias is caused by auto-regressive teacher forcing, where models are trained to maximize the probability of target sequences and are always assigned the previous target symbols as context.

2. (6 points) In sequence models for natural language processing, why are sentences usually sorted by length when batching?

   Solution: Within the same batch, all sequences must have the same length, and for this reason they must be padded with padding symbols for making them as long as the longest sentence in the batch. Since in NLP sentences can have very different lengths, if we don't sort sentences by length, we can end up with very unbalanced batches, where some sentences are very short and others are very long, which makes it necessary to add a lot of padding symbols. This process is inefficient and can make training more time consuming. For this reason, sentences are usually sorted by length, which makes each batch more balanced.

3. (6 points) Mention two advantages of self-attention encoders over RNN encoders.

   Solution: They can better capture long-range relations between elements of the sequence, since information does not propagate sequentially (words can be compared with a constant number of operations). This usually reflects in more accurate models. Also, the training is usually faster, since the self-attention computation can be parallelized, unlike RNNs, that require sequential computation.

Part 3: Problems (50 points)

Problem 1: Convolutional Neural Networks (25 points)

GoogLeNet is a convolutional neural network architecture that makes use of a particular block structure known as *inception block*. It was proposed by Szegedy and collaborators in 2015 in an article published at the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. The architecture of GoogLeNet is summarized in Fig. 1. Assume that all convolution layers shown include a ReLU non-linearity, besides the convolution operation.

Each inception block in the network (gray blocks) corresponds to a number of parallel convolutional branches, and its architecture is detailed in Fig. 2, where we write $\oplus$ to denote concatenation along the channel dimension and, as before, all convolution layers include a ReLU non-linearity, besides the convolution operation.

1. (5 points) The inception block includes some 1×1 convolutional layers. What does such a layer do, and why may it be useful?

   Note: Note that, for a $H \times W \times C$ image, a 1×1 convolutional layer comprises a filter with a $1 \times 1 \times C$.

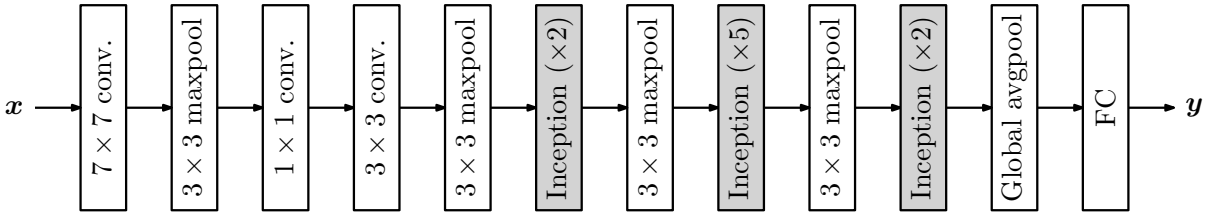


Figure 1: GoogLeNet architecture.

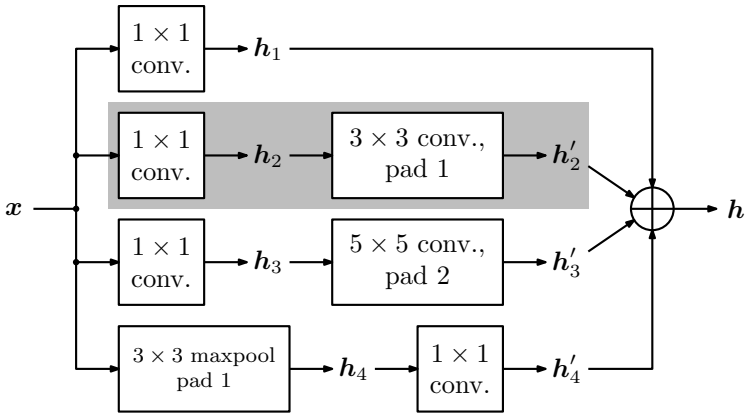


Figure 2: Inception block. The operation $\oplus$ denotes *concatenation* along the channel dimension. The branch used in question (c) is outlined in gray.

Solution: A 1×1 convolutional layer maps each input "pixel" (with all its channels) to one output pixel in a nonlinear way. It can be thought of as applying a fully connected layer to each input "pixel". This can be used, for example, to reduce or enlarge the number of channels in an image in a "pixelwise" manner, performing a nonlinear transformation on each of them.

2. (8 points) Consider an inception block where all convolutions have a single output channel. Suppose that the input $\boldsymbol{x}$ for such block is an $H \times W \times C$ image (C corresponds to the number of channels). Fill in the following table, where the dimensions refer to the different computed elements in Fig. 2 ($\boldsymbol{h}_1$, $\boldsymbol{h}_2$, $\boldsymbol{h}'_2$, etc.), and the number of parameters refer to the layers that compute them. Assume that stride = 1 and padding = 0, **unless where explicitly stated otherwise**. Indicate all relevant computations.

	# **param.**	**Dimension**
$\boldsymbol{x}$	–	$H \times W \times C$
$\boldsymbol{h}_1$	$C + 1$	$H \times W$
$\boldsymbol{h}_2$	$C + 1$	$H \times W$
$\boldsymbol{h}'_2$	$3 \times 3 + 1 = 10$	$H \times W$
$\boldsymbol{h}_3$	$C + 1$	$H \times W$
$\boldsymbol{h}'_3$	$5 \times 5 + 1 = 26$	$H \times W$
$\boldsymbol{h}_4$	0	$H \times W \times C$
$\boldsymbol{h}'_4$	$C + 1$	$H \times W$
$\boldsymbol{h}$	–	$H \times W \times 4$

3. (12 points) Consider once again the inception block in Fig. 2 and an input image $\boldsymbol{x}$ with dimensions $3 \times 3 \times 2$. Suppose that, after training, the parameters of the second branch in the block (outlined in gray) are

$$\boldsymbol{K}_{1\times1} = \big[\big[0.5\big], \big[0.5\big]\big], \qquad b_{1\times1} = 0,$$

for the 1×1 convolutional layer, and

$$\boldsymbol{K}_{3\times3} = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}, \qquad b_{3\times3} = 0,$$

for the 3×3 convolutional layer. Compute $\boldsymbol{h}_2'$ for the input

$$\boldsymbol{x} = \left[\begin{bmatrix} 1.03 & 0.94 & 0.98 \\ 0.78 & 0.49 & 0.82 \\ 0.94 & 0.77 & 0.85 \end{bmatrix}, \begin{bmatrix} 0.97 & 1.06 & 1.02 \\ 0.82 & 0.51 & 0.78 \\ 0.86 & 0.83 & 0.95 \end{bmatrix}\right].$$

Indicate all relevant computations.

Note: If you are unable to compute $\boldsymbol{h}_2$, you can use

$$\boldsymbol{h}_2 = \begin{bmatrix} 1.0 & 1.0 & 1.0 \\ 0.8 & 0.5 & 0.8 \\ 0.9 & 0.8 & 0.9 \end{bmatrix}$$

in the computation of $\boldsymbol{h}_2'$.

Solution: We can observe, from $\boldsymbol{K}_{1\times1}$, that the 1×1 convolutional layer merely computes the average of the two input channels for the image, yielding

$$\boldsymbol{h}_2 = \begin{bmatrix} 1.0 & 1.0 & 1.0 \\ 0.8 & 0.5 & 0.8 \\ 0.9 & 0.8 & 0.9 \end{bmatrix}.$$

To compute $\boldsymbol{h}_2'$ we first add a padding of 1 and then apply the convolution to the resulting image:

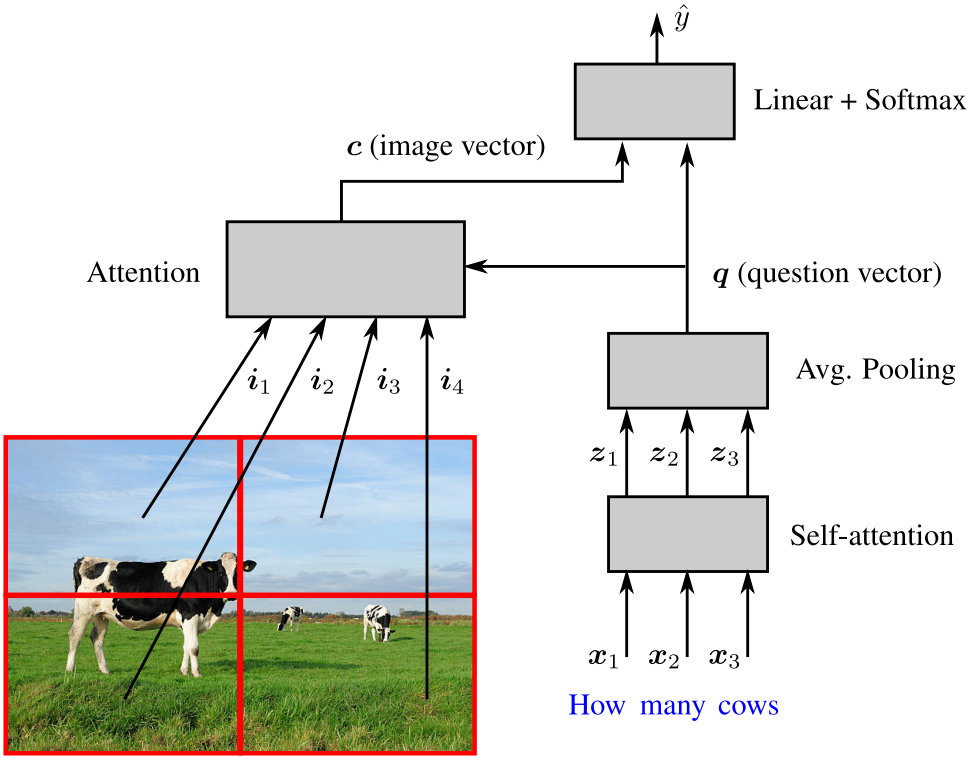
$$\boldsymbol{z}_2' = \begin{bmatrix} 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 1.0 & 1.0 & 0.0 \\ 0.0 & 0.8 & 0.5 & 0.8 & 0.0 \\ 0.0 & 0.9 & 0.8 & 0.9 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix} * \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1.3 & 2.1 & 1.3 \\ -0.3 & -0.4 & -0.3 \\ -1.3 & -2.1 & -1.3 \end{bmatrix}.$$

Finally, applying the ReLU, we get

$$\boldsymbol{h}_2' = \mathrm{ReLU}\left(\begin{bmatrix} 1.3 & 2.1 & 1.3 \\ -0.3 & -0.4 & -0.3 \\ -1.3 & -2.1 & -1.3 \end{bmatrix}\right) = \begin{bmatrix} 1.3 & 2.1 & 1.3 \\ 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \end{bmatrix}.$$

Problem 2: Sequence-to-Sequence Models (25 points)

Consider the network represented below for a visual question answering task. The task is as follows: given an image (in the example, a picture of cows) and a natural language question about the image (such as "How many cows?"), the goal is to output an answer from a predefined list of answers—this can be seen as a classification task.



The network has the following architecture:

- The image is processed by a convolution neural network (not represented), resulting in 4 feature representations $\boldsymbol{i}_1, \boldsymbol{i}_2, \boldsymbol{i}_3, \boldsymbol{i}_4$, where each $\boldsymbol{i}_i \in \mathbb{R}^2$ as shown in the figure.
- The words of the question are encoded as word embeddings, $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_n$, added to positional encodings $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n$, and provided as input to a small transformer model. Each $\boldsymbol{x}_i = \boldsymbol{w}_i + \boldsymbol{p}_i \in \mathbb{R}^3$. In the example, $n = 3$ (there is no stop symbol for simplicity).
- The transformer model has one layer of self-attention with a single head, with 2×3 projection matrices $\mathbf{W}_Q$, $\mathbf{W}_K$, $\mathbf{W}_V$. Scaled dot product attention is used to compute the attention probabilities. There is no feed-forward layer after the self-attention, no layer normalization, and no residual connections.
- The representations in the final layer of the transformer, $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n$, are average-pooled, leading to a single vector that represents the question, $\boldsymbol{q} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{z}_i$.
- This question vector $\boldsymbol{q}$ is then used in another attention mechanism as a query to attend over the image representations $\boldsymbol{i}_1, \boldsymbol{i}_2, \boldsymbol{i}_3, \boldsymbol{i}_4$ (again with scaled dot product attention), which are used both as keys and values. The result of this attention operation is an image vector $\boldsymbol{c}$.

- Finally, $\boldsymbol{q}$ and $\boldsymbol{c}$ are concatenated and go through an output linear layer, leading to a vector of answer probabilities

$$\text{softmax}\left(\mathbf{A}\begin{bmatrix}\boldsymbol{q}\\ \boldsymbol{c}\end{bmatrix}+\boldsymbol{b}\right),$$

where $\mathbf{A}$ and $\boldsymbol{b}$ are model parameters.

- The model parameters are as follows. The embedding vectors are

$$\boldsymbol{w}_{\text{How}} = [-1,0,1]^\top, \quad \boldsymbol{w}_{\text{many}} = [1,-1,0]^\top, \quad \boldsymbol{w}_{\text{cows}} = [-1,-1,-1]^\top,$$
$$\boldsymbol{p}_1 = [1,0,0]^\top, \quad \boldsymbol{p}_2 = [0,1,0]^\top, \quad \boldsymbol{p}_3 = [0,0,1]^\top.$$

The projection matrices of the transformer are

$$\mathbf{W}_Q = \mathbf{W}_K = \mathbf{W}_V = \begin{bmatrix} 0 & 2 \\ -1 & 0 \\ 2 & 1 \end{bmatrix}.$$

The parameters of the last output layer are:

$$\mathbf{A} = \begin{bmatrix} 2 & 0 & 1 & 0 \\ -1 & -2 & 0 & 2 \\ 0 & -1 & 2 & 1 \end{bmatrix}, \quad \boldsymbol{b} = \mathbf{0},$$

where the rows of $\mathbf{A}$ correspond (respectively) to the words "One", "Two", "Three".

1. (5 points) Compute the query matrix $\mathbf{Q}$, the key matrix $\mathbf{K}$, and the value matrix $\mathbf{V}$ associated to the words in the question. (Note: don't forget to account for the positional encodings.)

Solution: By summing the word and positional encodings, we obtain the following embedding matrix:

$$\mathbf{X} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ -1 & -1 & 0 \end{bmatrix}$$

The query, key, and value matrices are:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_Q = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ -1 & -1 & 0 \end{bmatrix}\begin{bmatrix} 0 & 2 \\ -1 & 0 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 0 & 2 \\ 1 & -2 \end{bmatrix}.$$

Since all projection matrices are the same, we have $\mathbf{Q} = \mathbf{K} = \mathbf{V}$.

2. (10 points) Assume that in the previous question we obtained

$$\mathbf{Q} = \mathbf{K} = \mathbf{V} = \begin{bmatrix} 2 & 1 \\ 0 & 2 \\ 1 & -2 \end{bmatrix}.$$

Compute the question vector $\boldsymbol{q}$.

Solution: The attention probabilities are

$$\mathbf{P} = \text{Softmax}\left(\frac{1}{\sqrt{2}}\mathbf{Q}\mathbf{K}^\top\right) = \text{Softmax}\left(\frac{1}{\sqrt{2}}\begin{bmatrix} 2 & 1 \\ 0 & 2 \\ 1 & -2 \end{bmatrix}\begin{bmatrix} 2 & 1 \\ 0 & 2 \\ 1 & -2 \end{bmatrix}^\top\right) = \text{Softmax}\left(\frac{1}{\sqrt{2}}\begin{bmatrix} 5 & 2 & 0 \\ 2 & 4 & -4 \\ 0 & -4 & 5 \end{bmatrix}\right)$$

$$= \begin{bmatrix} 0.87 & 0.10 & 0.03 \\ 0.20 & 0.80 & 0.00 \\ 0.03 & 0.00 & 0.97 \end{bmatrix}.$$

The output if the self-attention is:

$$\mathbf{Z} = \mathbf{P}\mathbf{V} = \begin{bmatrix} 1.77 & 1.03 \\ 0.39 & 1.79 \\ 1.03 & -1.91 \end{bmatrix}.$$

Finally, the query vector is

$$\boldsymbol{q} = \frac{1}{3}\mathbf{Z}^\top\mathbf{1} = [1.06, 0.30]^\top.$$

3. (10 points) Assume that in the previous question we obtained $\boldsymbol{q} = [1.06, 0.30]^\top$. Let the image feature maps be:

$$\boldsymbol{i}_1 = [1, 1]^\top, \quad \boldsymbol{i}_2 = [1, 1]^\top, \quad \boldsymbol{i}_3 = [0, 0]^\top, \quad \boldsymbol{i}_4 = [4, 4]^\top.$$

Compute the attention probabilities over the image feature maps (using scaled dot product attention) and the most probable answer returned by the system.

Solution: We can form the key matrix (which equals the value matrix):

$$\mathbf{I} = [\boldsymbol{i}_1^\top, \boldsymbol{i}_2^\top, \boldsymbol{i}_3^\top, \boldsymbol{i}_4^\top]^\top = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 0 & 0 \\ 4 & 4 \end{bmatrix}.$$

The attention probabilities are

$$\boldsymbol{a} = \text{softmax}\left(\frac{1}{\sqrt{2}}\mathbf{I}\boldsymbol{q}\right) = \text{softmax}\left(\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 0 & 0 \\ 4 & 4 \end{bmatrix}\begin{bmatrix} 1.06 \\ 0.30 \end{bmatrix}\right) = [0.05, 0.05, 0.02, 0.88]^\top.$$

The image vector is:

$$\boldsymbol{c} = \mathbf{I}^\top\boldsymbol{a} = \begin{bmatrix} 1 & 1 & 0 & 4 \\ 1 & 1 & 0 & 4 \end{bmatrix}\begin{bmatrix} 0.05 \\ 0.05 \\ 0.02 \\ 0.88 \end{bmatrix} = [3.63, 3.63]^\top.$$

The logits for the answer are

$$\boldsymbol{s} = \mathbf{A}\begin{bmatrix} \boldsymbol{q} \\ \boldsymbol{c} \end{bmatrix} + \boldsymbol{b} = \begin{bmatrix} 2 & 0 & 1 & 0 \\ -1 & -2 & 0 & 2 \\ 0 & -1 & 2 & 1 \end{bmatrix}\begin{bmatrix} 1.06 \\ 0.30 \\ 3.63 \\ 3.63 \end{bmatrix} = [5.8, 5.6, 10.6]^\top.$$

Therefore, the answer is "Three".