TÉCNICO LISBOA

Deep Learning

MSc in Computer Science and Engineering

MSc in Electrical and Computer Engineering

Final exam — February 12, 2022

Version A

Instructions

- You have 120 minutes to complete the exam.
- Make sure that your test has a total of 10 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 19 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)

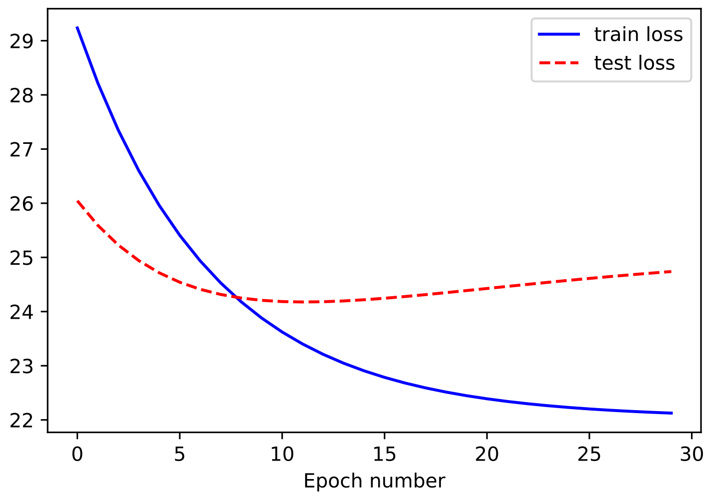
In each of the following questions, indicate your answer by *checking a single option.*

1. (4 points) An RNN-based sequence-to-sequence model with an attention mechanism translates an input sentence of M words into an output sentence with N words. How does the number of computational operations (algorithmic complexity) increase as a function of M and N?

   - ☐ $O(M + N)$
   - ■ $O(MN)$
   - ☐ $O(\max(M, N)^2)$
   - ☐ $O(M^N)$

   Solution: The correct option is $O(MN)$, since for each of the N generated words we need to attend to M representations for the source words.

2. (4 points) A model is trained for 30 epochs with gradient descent and it leads to the following plot for its training and test losses:



   Which of the following statements is a plausible explanation for what could be happening?

   - ☐ The model is underfitting the training data.
   - ■ **The model is overfitting the training data.**
   - ☐ The model generalizes well to unseen examples.
   - ☐ None the above.

   Solution: The training error is decreasing, while the test error is increasing, which suggests that the model is overfitting the training data.

3. (4 points) A neural network is overfitting its training data. What strategies could mitigate this?

   - ■ **Increase the dropout probability.**

- ☐ Decrease the amount of training data.
- ☐ Increase the number of hidden units.
- ☐ All the above.

Solution: More regularization should help, and this can be achieved by increasing the dropout probability.

4. (4 points) Let $\vee, \wedge, \oplus$ denote respectively the OR, AND, and XOR Boolean logical operators, and $\neg$ denote Boolean negation. Assume Boolean values are represented as -1 (False) and $+1$ (True). Which of these logical functions cannot be learned by a single perceptron with inputs A and B?

   ■ $(A \wedge \neg B) \vee (\neg A \wedge B)$

   □ $(A \oplus B) \wedge A$

   □ $A \vee B$

   □ $\neg A \wedge B$

   Solution: The answer is $(A \wedge \neg B) \vee (\neg A \wedge B)$, which is equal to $A \oplus B$.

5. (4 points) Let $L(\boldsymbol{w}) = \frac{1}{2}\sum_i (y_i - \boldsymbol{w}^\top \boldsymbol{\phi}(x_i))^2$ be the loss function corresponding to a linear regression problem. Which equation represents the stochastic gradient descent update for $\boldsymbol{w}$?

   ■ $\boldsymbol{w}^{(k+1)} \leftarrow \boldsymbol{w}^{(k)} - \eta(y_i - \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}_i))\boldsymbol{\phi}(x_i)$

   □ $\boldsymbol{w}^{(k+1)} \leftarrow \boldsymbol{w}^{(k)} - \eta \sum_i (y_i - \boldsymbol{w}^\top \boldsymbol{\phi}(x_i))\boldsymbol{\phi}(x_i)$

   □ $\boldsymbol{w}^{(k+1)} \leftarrow \boldsymbol{w}^{(k)} - \eta(y_i - \mathrm{sign}(\boldsymbol{w}^\top \boldsymbol{\phi}(x_i)))\boldsymbol{\phi}(x_i)$, where $\mathrm{sign}(\cdot)$ is the sign function

   □ $\boldsymbol{w}^{(k+1)} \leftarrow \boldsymbol{w}^{(k)} - \eta(y_i - \sigma(\boldsymbol{w}^\top \boldsymbol{\phi}(x_i)))\boldsymbol{\phi}(x_i)$, where $\sigma(z) = 1/(1+e^{-z})$ is the sigmoid function.

   Solution: Stochastic gradient updates depend only on a single example (or a mini-batch of examples). The option $\boldsymbol{w}^{(k+1)} \leftarrow \boldsymbol{w}^{(k)} - \eta \sum_i (y_i - \boldsymbol{w}^\top \boldsymbol{\phi}(x_i))\boldsymbol{\phi}(x_i)$ corresponds to gradient descent on the full batch.

6. (4 points) Which one of the following statements is true?

   □ Convolutional layers are equivariant to translations and rotations.

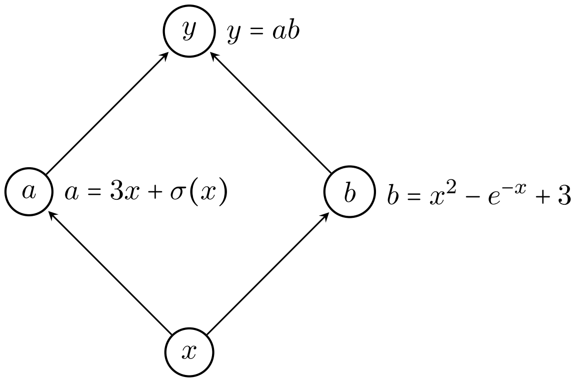
   □ Neural networks with a single hidden layer with linear activations are universal approximators.

   □ Auto-encoders with non-linear activations and a squared loss are equivalent to PCA.

   ■ **None of the above.**

   Solution: Convolutional layers are equivariant to translations, but not rotations. Neural networks with a single hidden layer with **linear** activations are equivalent to linear classifiers, which are not universal approximators. Auto-encoders with **linear** activations would correspond to PCA.

7. (4 points) Consider the following computation graph, where $\sigma(z) = 1/(1+e^{-z})$ is the sigmoid function. What is the derivative of y with respect to x?



- ■ $b(3 + \sigma(x)(1 - \sigma(x))) + a(2x + e^{-x})$**.**
- □ $a(3 + \sigma(x)(1 - \sigma(x))) + b(2x + e^{-x})$.
- □ $3 + \sigma(x)(1 - \sigma(x)) + 2x + e^{-x}$.
- □ 0.

Solution: It is $b(3 + \sigma(x)(1 - \sigma(x))) + a(2x + e^{-x})$:

$$\frac{\partial y}{\partial a} = b, \quad \frac{\partial y}{\partial b} = a$$
$$\frac{\partial a}{\partial x} = 3 + \sigma(x)(1 - \sigma(x)), \quad \frac{\partial b}{\partial x} = 2x + e^{-x}$$
$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial a}\frac{\partial a}{\partial x} + \frac{\partial y}{\partial b}\frac{\partial b}{\partial x} = b(3 + \sigma(x)(1 - \sigma(x))) + a(2x + e^{-x}).$$

8. (4 points) Which one of the following statements is **false**?
   - ■ **Gradient clipping can prevent vanishing gradients.**
   - □ Transformer models can be used for computer vision applications.
   - □ Distributed representations generally require fewer dimensions than local (one-hot) representations.
   - □ Upper level layers (closer to the output) tend to learn more abstract representations (shapes, forms, objects) compared to bottom level layers.

   Solution: Gradient clipping can prevent exploding gradients, not vanishing gradients.

Part 2: Short Answer Questions (18 points)

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

1. (6 points) Explain how dropout regularization works.

   Solution: At training time, for each example neurons are dropped randomly with probability p (i.e. their activations are masked to become zero) and the remaining activations are scaled by $1/(1 - p)$. This forces each neuron to depend less on other neurons' activations.

2. (6 points) Explain the role and need for positional encoding in transformers.

Solution: Without positional encodings, the self-attention in transformers is insensitive to the word positions being queried: permuting the words leads to a similar permutation in the self-attention responses. In order for transformers to be sensitive to the word order, each word embedding is augmented with a positional embedding.

3. (6 points) Mention one advantage of contextualized word embeddings (e.g. BERT) over static word embeddings (e.g. word2vec or GloVe).

Solution: Contextualized word embeddings can assign different representations to the same word being used in different contexts; this is particularly useful for polysemic words (such as “bank” which can be a river bank or a financial institution).

Part 3: Problems (50 points)

Problem 1: Convolutional Neural Networks (25 points)

In their retail store, Yolanda and Zach currently use a card punching system to register the entry and exit times of their 6 employees. However, they heard about recent advances in computer vision systems and decided to replace that system by face recognition using CNNs.

To train the system, they collected a large dataset of pictures from their 6 employees, Alice, Berta, Chad, Diane, Eric, and Frank. Each picture in the dataset is a 192×256 grayscale picture, similar to those depicted in Fig. 1, and is labeled according to the corresponding employee.

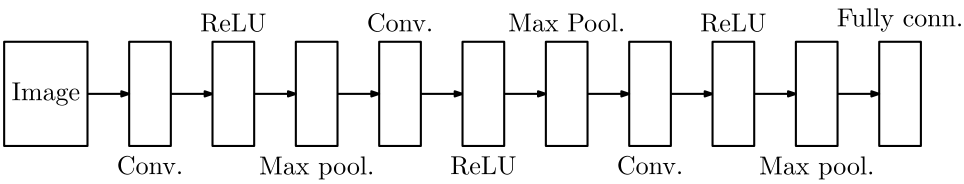
192×256 px	192×256 px	192×256 px	192×256 px	192×256 px	192×256 px
Class A	Class B	Class C	Class D	Class E	Class F

Figure 1: Sample pictures from the 6 classes that the CNN must recognize. Alice corresponds to class A, Berta to class B, etc.

1. (4 points) Briefly explain in 1-2 sentences why a CNN is an adequate choice of architecture for Yolanda and Zach’s task (image classification).

Solution: CNNs take advantage of the spacial structure of the image, unlike standard feed-forward networks. Moreover, convolutional and pooling layers exploit the fact that the same feature may appear in different parts of the image, enabling the network to process those occurrences in a similarly way.

2. (7 points) Suppose that, in their classifier, their use the following architecture:



which is specified using the following Pytorch code snippet:

```
nn.Sequential(
    nn.Conv2d(1, 5, kernel_size=3, stride=1, padding=1),
    nn.ReLU(),
    nn.MaxPool2d(kernel_size=2, stride=2),
    nn.Conv2d(5, 10, kernel_size=5, stride=1, padding=0),
    nn.ReLU(),
    nn.MaxPool2d(kernel_size=2, stride=2),
    nn.Conv2d(10, 20, kernel_size=2, stride=2, padding=0),
    nn.ReLU(),
    nn.MaxPool2d(kernel_size=5, stride=2),
    nn.Flatten(),
    nn.Linear(2800, 6))
```

Fill in the following table with the adequate values.

Layer	Output size	N. weights	N. biases
Input	$192 \times 256 \times 1$	0	0
1st conv. layer	$192 \times 256 \times 5$	45	5
1st pooling layer	$96 \times 128 \times 5$	0	0
2nd conv. layer	$92 \times 124 \times 10$	1250	10
2nd pooling layer	$46 \times 62 \times 10$	0	0
3rd conv. layer	$23 \times 31 \times 20$	800	20
3rd pooling layer	$10 \times 14 \times 20$	0	0
Output layer	6×1	$16,800$	6

3. (7 points) Consider the diagram in Fig. 2, containing the brightness values for the first window of pixels in one of the images in the dataset.

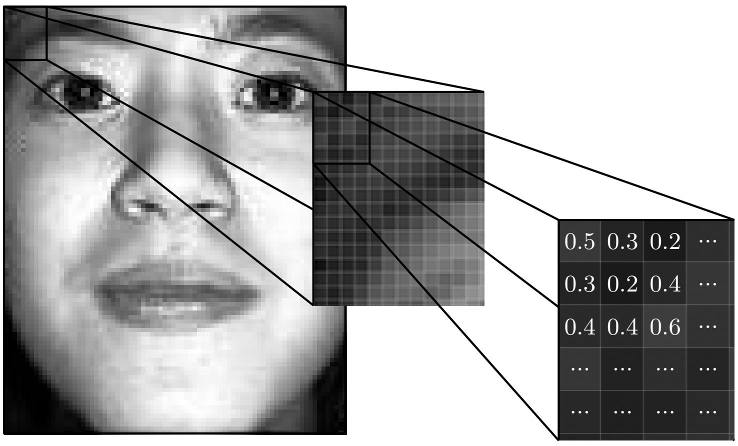


Figure 2: Brightness values for one of the images in the dataset.

Suppose that, after training, one of the filters in the first convolutional layer is defined by the parameters

$$\boldsymbol{K} = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \qquad\qquad b = 0.5.$$

For the filter provided, compute the the top-left-most value after the pooling layer. Do not forget that the first convolutional layer includes a padding of size 1 (use zeros as the padding value).

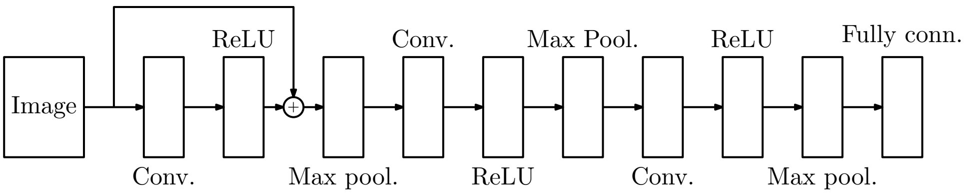
Solution: The input of the max-pool layer corresponding to the provided pixels is given by

$$\begin{aligned} \boldsymbol{h}_{\text{conv}} &= \text{ReLU}\left(\begin{bmatrix} -0.5 & 0.2 \\ -0.9 & 0.0 \end{bmatrix} + 0.5\right) \\ &= \begin{bmatrix} 0.0 & 0.7 \\ 0.0 & 0.5 \end{bmatrix}. \end{aligned}$$

At the output of the pooling layer, we thus have

$$h_{\text{pool}} = 0.7.$$

4. (7 points) Suppose that Yolanda and Zach decide to add some skip connections to their network, as indicated in the diagram:



Repeat Question 3, but now considering the skip connection indicated in the diagram above.

Solution: The input of the first max-pool layer corresponding to the provided pixels is given by

$$\begin{aligned} \boldsymbol{h}_{\text{conv}} &= \text{ReLU}\left(\begin{bmatrix} -0.5 & 0.2 \\ -0.9 & 0.0 \end{bmatrix} + \begin{bmatrix} 0.5 & 0.3 \\ 0.3 & 0.2 \end{bmatrix} + 0.5\right) \\ &= \begin{bmatrix} 0.5 & 1.0 \\ 0.0 & 0.7 \end{bmatrix}. \end{aligned}$$

At the output of the pooling layer, we thus have

$$h_{\text{pool}} = 1.0.$$

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

Bartholomew (known to his friends as Bart) had an idea for a project: building a system to summarize news articles into a short sentence (e.g., a tweet). He collected a dataset with news documents and their corresponding tweets, which he will use to train a summarization model.

1. (7 points) Bart's sister (Lisa) is taking a course on *deep learning* and she recommended using a sequence-to-sequence architecture based on a *recurrent neural network* (RNN) for this problem. Bart tested a simple RNN-based sequence-to-sequence model (without any attention mechanism) on a small-scale experiment. He is using a very small vocabulary (7 words, including the <STOP> symbol), shared between the source and target, and using the same embedding vectors for both sides. The embedding vectors are

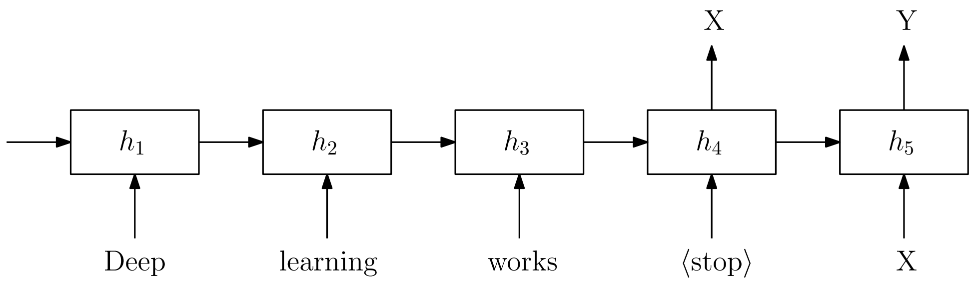
$$\boldsymbol{x}_{\text{Deep}} = [0,1]^\top, \quad \boldsymbol{x}_{\text{learning}} = [1,0]^\top, \quad \boldsymbol{x}_{\text{works}} = [-1,-1]^\top,$$
$$\boldsymbol{x}_{\text{!!!}} = [2,-1]^\top, \quad \boldsymbol{x}_{\text{\#deep}} = [-1,2]^\top, \quad \boldsymbol{x}_{\text{lol}} = [0,-1]^\top, \quad \boldsymbol{x}_{\text{<stop>}} = [1,1]^\top.$$

The initial hidden state of the RNN, $\boldsymbol{h}_0$, is all-zeros. The input-to-hidden matrix is

$$\boldsymbol{W}_{hx} = \begin{bmatrix} 0 & -1 \\ 2 & 0 \\ 1 & 1 \end{bmatrix}.$$

The recurrent matrix $\boldsymbol{W}_{hh}$ is the identity matrix. All biases are vectors of zeros. The RNN uses `relu` activations.

Compute the last state of the encoder RNN, $\boldsymbol{h}_4$, for the input document "Deep learning works". Show all your calculations.



Solution: We have:

$$\begin{aligned}
\boldsymbol{h}_1 &= \text{relu}(\boldsymbol{W}_{hx}\boldsymbol{x}_{\text{Deep}} + \boldsymbol{W}_{hh}\boldsymbol{h}_0) \\
&= \text{relu}([-1,0,1]^\top + [0,0,0]^\top) \\
&= [0,0,1]^\top. \\
\boldsymbol{h}_2 &= \text{relu}(\boldsymbol{W}_{hx}\boldsymbol{x}_{\text{learning}} + \boldsymbol{W}_{hh}\boldsymbol{h}_1) \\
&= \text{relu}([0,2,1]^\top + [0,0,1]^\top) \\
&= [0,2,2]^\top. \\
\boldsymbol{h}_3 &= \text{relu}(\boldsymbol{W}_{hx}\boldsymbol{x}_{\text{works}} + \boldsymbol{W}_{hh}\boldsymbol{h}_2) \\
&= \text{relu}([1,-2,-2]^\top + [0,2,2]^\top) \\
&= [1,0,0]^\top. \\
\boldsymbol{h}_4 &= \text{relu}(\boldsymbol{W}_{hx}\boldsymbol{x}_{\text{<stop>}} + \boldsymbol{W}_{hh}\boldsymbol{h}_3) \\
&= \text{relu}([-1,2,2]^\top + [1,0,0]^\top) \\
&= [0,2,2]^\top.
\end{aligned}$$

Therefore the last state is $\boldsymbol{h}_4 = [0,2,2]^\top$.

2. (8 points) Assume that in the previous question we obtained $\boldsymbol{h}_4 = [0, 2, 2]^\top$. Assume that the decoder RNN has the same parameters as the encoder RNN, and the hidden-to-output matrix is

$$\boldsymbol{W}_{yh} = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 1 & -1 \\ -2 & 1 & 0 \\ 0 & 0 & -1 \\ 1 & 2 & 0 \\ 0 & 0 & 0 \\ -2 & 0 & 1 \end{bmatrix}.$$

The target word probabilities at time step t are given by $\text{softmax}(\boldsymbol{W}_{yh}\boldsymbol{h}_t)$, where $\boldsymbol{h}_t$ is the corresponding state of the decoder RNN.

Compute the first two words of the generated tweet using **greedy decoding**.

Solution: We have:

$$\begin{aligned}
\boldsymbol{y}_1 &= \text{argmax}(\boldsymbol{W}_{yh}\boldsymbol{h}_4) \\
&= \text{argmax}([-2, 0, 2, -2, 4, 0, 2]) = \#\text{deep}. \\
\boldsymbol{h}_5 &= \text{relu}(\boldsymbol{W}_{hx}\boldsymbol{x}_{\#\text{deep}} + \boldsymbol{W}_{hh}\boldsymbol{h}_4) \\
&= \text{relu}([-2, -2, 1]^\top + [0, 2, 2]^\top) \\
&= [0, 0, 3]^\top. \\
\boldsymbol{y}_2 &= \text{argmax}(\boldsymbol{W}_{yh}\boldsymbol{h}_5) \\
&= \text{argmax}([0, -3, 0, -3, 0, 0, 3]) = <\text{stop}>.
\end{aligned}$$

The generated words are "#deep <stop>".

3. (6 points) After playing with this network for a while, Bart realized that it didn't work well for long documents and therefore decided to add an attention mechanism. In the first decoding step, using **scaled dot-product attention** with $\boldsymbol{h}_4$ as the query vector and $\boldsymbol{h}_1$, $\boldsymbol{h}_2$, $\boldsymbol{h}_3$ as the key and value vectors, compute the attention probabilities and the resulting context vector (use $\boldsymbol{h}_1 = [0, 0, 1]^\top$, $\boldsymbol{h}_2 = [0, 2, 2]^\top$, $\boldsymbol{h}_3 = [1, 0, 0]^\top$, $\boldsymbol{h}_4 = [0, 2, 2]^\top$).

Solution: We have:

$$\begin{aligned}
s_1 &= \frac{1}{\sqrt{3}}[0, 2, 2]^\top[0, 0, 1] = \frac{2}{\sqrt{3}} \\
s_2 &= \frac{1}{\sqrt{3}}[0, 2, 2]^\top[0, 2, 2] = \frac{8}{\sqrt{3}} \\
s_3 &= \frac{1}{\sqrt{3}}[0, 2, 2]^\top[1, 0, 0] = 0 \\
Z &= \exp\left(\frac{2}{\sqrt{3}}\right) + \exp\left(\frac{8}{\sqrt{3}}\right) + \exp(0) = 105.546 \\
\boldsymbol{p} &= \text{softmax}([s1, s2, s3]) = \exp([s_1, s_2, s_3])/Z = [.030, .960, .009] \\
\boldsymbol{c} &= \boldsymbol{h}_1 p_1 + \boldsymbol{h}_2 p_2 + \boldsymbol{h}_3 p_3 = [.009, 1.921, 1.951].
\end{aligned}$$

4. (4 points) Lisa's friend, Allison, who is also knowledgeable about deep learning, told Bart about transformers and large pretrained models. Give one example of a pretrained model that Bart could use for this task and the necessary steps to use it.

   Solution: Bart could use a pretrained decoder-only (e.g., GPT) or encoder-decoder model (e.g., T5, BART) and fine-tune it on the data he has available. Alternatively he could use the model without any fine-tuning and use prompting at test time. A possible prompt would be "<document> TL;DR: <answer>". Note: an encoder-only model (e.g. BERT) would not be suitable, since this an auto-regressive generation task.