Insper

Artificial Neural Networks and Deep Learning

Final Exam – FE . 2026.2

Example

Dear Student,

You will have 120 minutes from the official start of the exam to complete it, so manage your time well. Read the following instructions and the exam questions carefully before you begin.

1. Write your name legibly on the last two pages of the exam.
2. This exam consists of 14 multiple-choice questions (0.5 point each) and 1 open-ended question in three parts (3.0 points), for a total of 10.0 points.
3. Check that the exam is complete and free of printing problems, and tell the proctor before you start. Later reports will not be considered. The exam has 13 pages.
4. **Return only the last two pages to the proctor (pages 12 and 13)**, with the answer sheet and your identification – copy your answers onto the sheet correctly and do not forget to write your name.
5. You may answer in pencil or in pen. Exams answered in pencil, however, will not be reviewed by the professor.
6. If you have a question about any item of this exam, write it down on the answer sheet or on its back, so that the professor can assess its relevance during grading.
7. Carrying any material other than the question sheet – mobile phone (on or off), tablet, laptop, notes and books – will constitute a violation of the Code of Ethics and Conduct and will lead to the sanctions it provides for.
8. Consulting classmates or any material other than the question sheet will constitute a violation of the Code of Ethics and Conduct and will lead to the sanctions it provides for. Do your work ethically!
9. You may leave the room only after handing in the exam. If you need to leave during the exam, ask the proctor for permission beforehand.

Good luck!

For the Professor's use only

Guidelines during the exam:

Scratch paper for notes allowed?
( ) YES (X) NO

Calculator allowed?
( ) YES (X) NO

Reference material allowed?
( ) YES (X) NO

After grading:

Exam eligible for review?
( ) YES ( ) NO

Q1. *(Perceptron / Learning rule)*

A perceptron with a step activation (output 1 if $z \geq 0$; output 0 if $z < 0$) and learning rate $\eta = 0.5$ starts an epoch with $\mathbf{w} = (1, -1)$ and $b = 0$. The examples are presented in this order and the perceptron learning rule is applied right after each error:

example	x_1	x_2	**label** y
1	2	1	1
2	1	2	0
3	0	1	1
4	1	3	0

At the end of the epoch, what are $\mathbf{w}$ and b?

( ) **A)** $\mathbf{w} = (1, -0.5)$ and $b = 0.5$, with a single update, at example 3.

( ) **B)** $\mathbf{w} = (0.5, -2)$ and $b = 0$, with two updates, at examples 3 and 4.

( ) **C)** $\mathbf{w} = (1, -1.5)$ and $b = -0.5$, with a single update, at example 3.

( ) **D)** $\mathbf{w} = (1.5, 1)$ and $b = 1$, with two updates, at examples 3 and 4.

( ) **E)** $\mathbf{w} = (0.5, -2)$ and $b = 0.5$, with two updates, at examples 3 and 4.

Q2. *(Multi-layer perceptron and history / The XOR problem)*

A network has two binary inputs, two hidden neurons and one output, all with the perceptron's step activation ($H(z) = 1$ if $z \geq 0$; $H(z) = 0$ if $z < 0$). The weights were set by hand:

$$h_1 = H(x_1 - x_2 - 0.5), \qquad h_2 = H(x_2 - x_1 - 0.5), \qquad y = H(h_1 + h_2 - 0.5).$$

x_1	x_2	y
0	0	?
0	1	?
1	0	?
1	1	?

What is the column y (in the order of the table) and what does it show?

( ) **A)** 0, 1, 1, 1: the network computes OR, which a single perceptron already represents; the hidden layer is superfluous.

( ) **B)** 0, 0, 0, 1: the network computes AND, the function that Minsky and Papert (1969) showed to be beyond the reach of the perceptron.

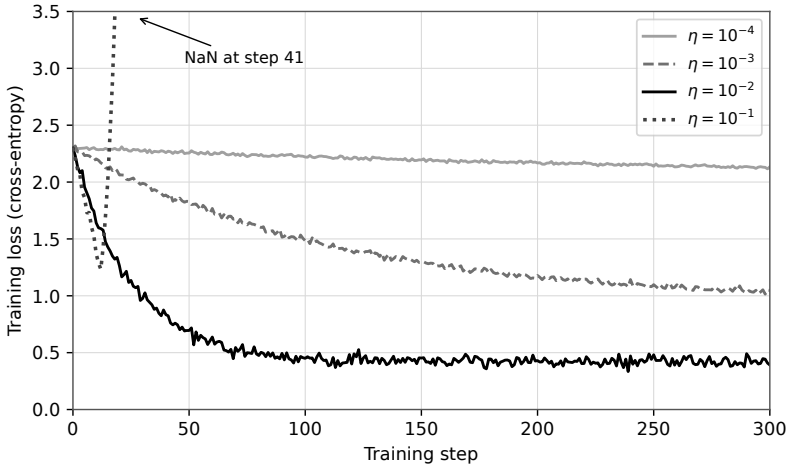
( ) **C)** 0, 1, 1, 0: the network computes XOR, but only because of the step; with the sigmoid, no hidden layer would solve it.

( ) **D)** 0, 1, 1, 0: it is XOR, beyond the reach of a single-layer perceptron; backpropagation (1986) made it possible to train networks like this one.

( ) **E)** 0, 1, 1, 0: the network computes XOR; since it is linearly separable in (x_1, x_2), a perceptron alone would also learn it.

Q3. *(Optimization / Learning rate)*

Before a long training run, a team trained the same model (AdamW, 10 classes) for 300 steps with four learning rates. The figure shows the training loss for each one; with $\eta = 10^{-1}$ the loss became NaN at step 41.



Which reading and which choice for the long run are correct?

( ) **A)** 10^{-1} went past the stability limit; 10^{-2} is the largest rate that does not blow up; for the long run, use about $5 \cdot 10^{-3}$.

( ) **B)** 10^{-1} is the best, because it is the one that brings the loss down the most in the first steps; just stop training before the NaN.

( ) **C)** 10^{-3}, because the smoothest curve of the four indicates the rate that will converge best in a long run.

( ) **D)** 10^{-4}, because a small rate never diverges and, in a long run, ends up reaching the loss of the others.

( ) **E)** 10^{-1} with gradient clipping, which prevents the NaN and keeps the faster descent that the plot shows.

Q4. *(Regularization / Data augmentation and validation)*

A classifier of 10 bird species was trained on 6,000 photos and evaluated on a separate validation split. The team measured four configurations (accuracy, in %):

	what changes	training	validation
A	no regularization	100	81
B	A + weight decay and early stopping	97	86
C	B + data augmentation (random crop, flip and color), applied to training **and to validation**	92	84
D	same as C, but validation is only resized and center-cropped	92	91

Which reading is correct?

( ) **A)** Data augmentation made the model worse: in C, validation accuracy dropped from 86% to 84% relative to B.

( ) **B)** A is the best configuration: it alone reached 100% on training, which shows that it learned all the patterns.

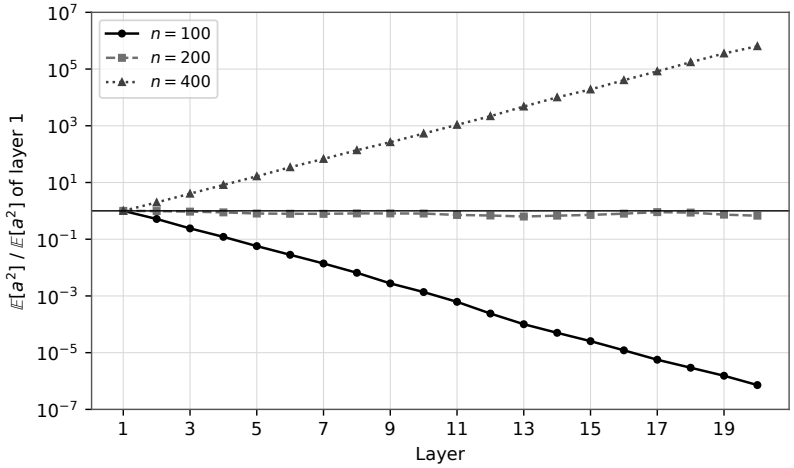
( ) **C)** D is optimistic: validating without augmentation hides the errors on real photos, which arrive cropped and with varied colors.

( ) **D)** B got worse than A, because training accuracy dropped from 100% to 97% with the regularization.

( ) **E)** C and D have the same training and the difference comes only from the augmented validation; data augmentation helped and D is the best.

Q5. *(Deep networks / Initialization and width)*

Three 20-layer MLPs with ReLU, with no shortcuts and no normalization, differ only in their width n (100, 200 or 400 units per layer). In all three, each weight was drawn from a normal distribution with zero mean and fixed variance $\sigma^2 = 0.01$. The plot shows, at initialization (before any training), the $\mathbb{E}[a^2]$ of each layer divided by that of layer 1, on a log scale (geometric mean of five networks drawn per width).



Why does the same initialization work for one width and fail for the others? What fixes it?

( ) **A)** Width changes the effective depth: wide networks have more active units per layer, which is why the signal grows in the 400-wide one.

( ) **B)** ReLU saturates in wide networks: summing more inputs pushes the pre-activations into the flat region of the function.

( ) **C)** The per-layer factor is $n\sigma^2 \cdot \frac{1}{2}$: 0.5, 1 and 2 for n = 100, 200 and 400; with $\sigma^2 = 2/n$ (He), it equals 1 in all three.

( ) **D)** The per-layer factor is $n\sigma^2$: 1, 2 and 4; only the 100-wide network should be stable and the figure points to a measurement error.

( ) **E)** It is the learning rate: wider networks need a smaller η and that is why the 400-wide one blows up already at initialization.

Q6. *(Convolutional networks / Shapes and parameters)*

A CNN takes 64×64 RGB images and ends in a 10-class `Linear`. The student needs to complete the table to write the input of that `Linear`:

layer	output shape
input	$3 \times 64 \times 64$
`Conv2d(3, 16, 3, padding=1)` + ReLU	$16 \times 64 \times 64$
`MaxPool2d(2)`	$16 \times 32 \times 32$
`Conv2d(16, 32, 3, stride=2, padding=1)` + ReLU	?
`MaxPool2d(2)`	?
`Flatten`	?
`Linear(?, 10)`	10

What is the shape before the `Flatten` and how many parameters (with bias) does the `Linear` have? How many would it have with `AdaptiveAvgPool2d(1)` (global pooling) right before the `Flatten`?

( ) **A)** $32 \times 16 \times 16$; `Linear(8192, 10)`, with 81,930 parameters; with global pooling, 330.

( ) **B)** $32 \times 8 \times 8$; `Linear(2048, 10)`, with 20,490 parameters; with global pooling, 330.

( ) **C)** $32 \times 7 \times 7$; `Linear(1568, 10)`, with 15,690 parameters; with global pooling, 330.

( ) **D)** $32 \times 8 \times 8$; `Linear(2048, 10)`, with 20,480 parameters; with global pooling, 20.

( ) **E)** $32 \times 8 \times 8$; `Linear(2048, 10)`, with 20,490 parameters; global pooling does not change this count.

Q7. *(Vision Transformers and transfer learning / LoRA)*

A team adapts a pretrained ViT-B/16 (about 86.6 million parameters, $d = 768$, 12 blocks) with rank $r = 4$ LoRA on the four attention projections of each block (W_q, W_k, W_v and W_o, all 768×768); the rest stays frozen. Not counting the new head, how many parameters are trained? With AdamW storing 12 bytes per trainable parameter (fp32 master copy and two moments), how much memory does the optimizer state take?

( ) **A)** 147,456, since each matrix gains $r \cdot d$ numbers; the AdamW state comes to about 1.8 MB.

( ) **B)** 28,311,552, since each $\Delta W = BA$ has 768×768 entries; the AdamW state comes to about 340 MB.

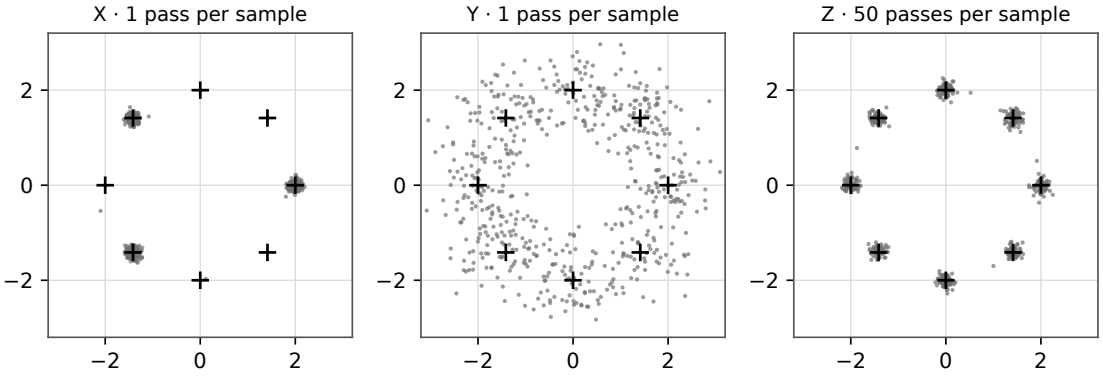
( ) **C)** 294,912; the AdamW state does not change, because the optimizer stores the moments of all the weights of the model.

( ) **D)** 294,912; the AdamW state takes about 0.6 MB, because the moments use 2 bytes per parameter.

( ) **E)** 294,912, about 0.3% of the model; the AdamW state drops from about 1 GB (full fine-tuning) to about 3.5 MB.

Q8. *(Generative models / What the samples reveal)*

Three models were trained on the same data, eight clusters arranged in a ring (the crosses mark the centers): a GAN, a VAE and a flow matching model, not necessarily in that order. The figure shows 600 samples from each one and how many passes through the network each sample cost.



Which reading is correct?

( ) **A)** X is the GAN, which picks modes like the reverse KL; Y is the VAE, which covers all of them (including the valley between them) like the forward KL.

( ) **B)** X is the VAE: the VAE's blur concentrates the samples on a few modes, while the GAN, Y, spreads the samples over the ring.

( ) **C)** Y is the GAN with mode collapse: the samples between the clusters show that it settled on a few of them.

( ) **D)** Z is the VAE, because only a model trained by likelihood covers the eight modes with samples this sharp.

( ) **E)** X and Y differ only because of short training: with more epochs, X would cover the eight modes and Y would stop placing samples between them.

Q9. *(Variational autoencoders / The KL of each dimension)*

A VAE with an 8-dimensional latent space was trained on photos of clothing with $\beta = 1$. On validation, the team recorded the mean KL to the prior $\mathcal{N}(0, I)$, separately for each latent dimension, in nats:

dimension	1	2	3	4	5	6	7	8
mean KL	3.1	2.4	0.01	1.7	0.02	0.00	2.9	0.01

Which reading is correct?

( ) **A)** Dimensions 3, 5, 6 and 8 are the most informative, because the encoder uses them while paying almost no KL.

( ) **B)** The total KL, of about 10 nats, indicates posterior collapse: the encoder ignores the input in every dimension.

( ) **C)** In the dimensions with KL near zero, $\sigma \to 0$: the encoder encodes them with no uncertainty at all.

( ) **D)** Dimensions 3, 5, 6 and 8 are inactive: in them $q(z \mid x) \approx \mathcal{N}(0, 1)$ for every image and the decoder has nothing to read.

( ) **E)** Increasing β would reactivate dimensions 3, 5, 6 and 8, because the KL term would weigh more on them.

Q10. *(GANs / The generator loss)*

In the training loop of a GAN, `D` returns logits and `bce(logits, target)` is the binary cross-entropy with a sigmoid (`BCEWithLogitsLoss`). The generator step reads:

```
fake = G(z)
loss_G = -bce(D(fake), torch.zeros(B))      # B: batch size
opt_G.zero_grad(); loss_G.backward(); opt_G.step()
```

Over the first 2,000 steps, $D(G(z))$ stays around 0.02 and the samples barely change. What explains this and what should be changed?

( ) **A)** Nothing is wrong: minimizing $-$`bce(D(fake), 0)` is the generator's objective in the original paper; it just needs more training.

( ) **B)** `fake.detach()` is missing: without it, the generator's gradient leaks into the discriminator, which therefore wins early.

( ) **C)** It is the minimax loss, $\log(1 - D(G(z)))$: with $D(G(z)) \approx 0.02$, the gradient at the logit is only -0.02; `bce(D(fake), ones)` gives -0.98.

( ) **D)** The sign is flipped: the generator is maximizing $\log(1 - D(G(z)))$; just remove the minus sign from the loss.

( ) **E)** The game has reached equilibrium, where the discriminator cannot tell real from fake; there is nothing to change.

Q11. *(CLIP / Zero-shot classification)*

A park uses CLIP to classify camera-trap photos, with one prompt per class and temperature $\tau = 0.01$ (logit = 100 $\times$ cosine). For one photo, the cosines with the prompts are:

class	maned wolf	crab-eating fox	ocelot
cosine	0.29	0.27	0.22

Use $e^{-2} \approx 0.14$, $e^{-7} \approx 0.001$ and $e^{-9} \approx 0.0001$. What is the predicted class and with what probability? What happens if the park adds the class "giant anteater", whose cosine with the photo is 0.20?

( ) **A)** Maned wolf, with about 0.88; the new class barely moves this value and requires no training at all.

( ) **B)** Maned wolf, with 0.29, the cosine itself; the new class does not change this value, which does not depend on the others.

( ) **C)** Maned wolf, with about 0.37 (0.29 over the sum of the cosines); the new class lowers it to about 0.30.

( ) **D)** Maned wolf, with about 0.88; the new class requires retraining the classifier's linear layer with photos of it.

( ) **E)** None: cosines below 0.3 show that the photo matches no prompt; the answer should be discarded.

Q12. *(Stable Diffusion / Classifier-free guidance)*

In a Stable Diffusion sampling step with guidance scale $w = 7.5$, the U-Net returned, for two components of the latent, the noise predictions below:

component	**without the prompt**, $\epsilon_\theta(z_t, \varnothing)$	**with the prompt**, $\epsilon_\theta(z_t, c)$
1	0.10	0.30
2	-0.20	-0.20

What is the guided prediction $\tilde{\epsilon}$ for these two components?

( ) **A)** $(0.30, -0.20)$: with the prompt present, the guided prediction is the conditional prediction itself.

( ) **B)** $(2.25, -1.50)$: guidance multiplies the conditional prediction by the scale w, component by component.

( ) **C)** $(1.60, -0.20)$: component 2 does not change, because there the prompt does not alter the noise prediction.

( ) **D)** $(1.80, -0.20)$: guidance starts from the conditional prediction and adds w times the difference between the two.

( ) **E)** $(1.60, -1.50)$: the scale w amplifies both components, including the one that does not depend on the prompt.

Q13. *(Flow matching / The interpolant and the time convention)*

A flow matching training step with the linear interpolant draws the noise $x_0 = (1, -2)$, the data point $x_1 = (3, 2)$ and $t = 0.25$. In the class convention, $t = 0$ is noise and $t = 1$ is data. A colleague's code follows the convention of the SD3 paper (data at $t = 0$, noise at $t = 1$) and drew the same two points and the same t. What input x_t and what regression target does each version use?

( ) **A)** Class: $x_t = (2.5, 1)$ and target $(2, 4)$. Colleague: $x_t = (1.5, -1)$ and target $(-2, -4)$.

( ) **B)** Class: $x_t = (1.5, -1)$ and target $(2, 4)$. Colleague: $x_t = (2.5, 1)$ and target $(-2, -4)$.

( ) **C)** Class: $x_t = (1.5, -1)$ and target $(3, 2)$, the data point. Colleague: the same x_t and target $(1, -2)$, the noise.

( ) **D)** Both: $x_t = (1.5, -1)$ and target $(2, 4)$, because the linear interpolant is symmetric in time.

( ) **E)** Class: $x_t = (1.5, -1)$ and target $(2, 4)$. Colleague: $x_t = (2.5, 1)$ and the same target, $(2, 4)$.

Q14. *(Diffusion Transformers and autoregressive generation / Choosing the generator)*

A company is about to launch two image generation products and needs to choose the design of the generator for each one:

product	what it needs to do
1	text-to-image API at 1024×1024, judged by quality per GPU-hour
2	an assistant that chats, reads the user's photos, reasons in text and returns the edited photo in the same conversation

Which choice is best justified?

( ) **A)** 1: autoregressive in raster order, for the exact likelihood; 2: DiT with cross-attention, which already reads the user's text.

( ) **B)** Both with MMDiT: in joint attention, the image rewrites the text tokens, so the same model already answers in text.

( ) **C)** Both autoregressive over VQ tokens, because a large codebook gives more fidelity than any continuous VAE.

( ) **D)** 1: a diffusion or flow transformer; 2: a single model over a sequence that interleaves text and image.

( ) **E)** 1: MaskGIT, because eight parallel rounds beat any DiT in quality; 2: VAR, which generates from coarse to fine.

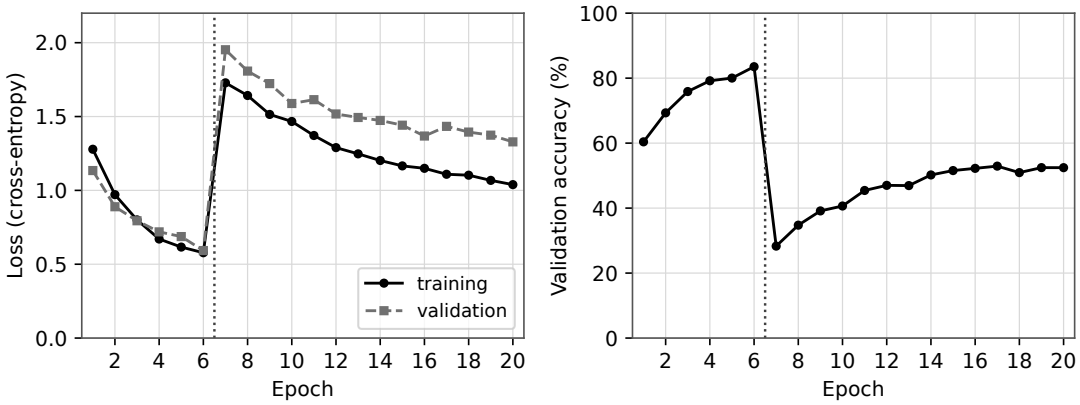
Q15. *(Open-ended – An end-to-end deep learning project – 3.0 points)*

A coffee growers' cooperative wants an app that classifies a photo of a coffee leaf, taken with a phone, into one of five classes: *healthy*, *rust*, *cercospora*, *leaf miner* or *phoma*. Agronomists labeled 2,400 photos (480 per class), already split into training (1,920) and validation (480); a test set will be collected in the next harvest. Rust and cercospora are told apart mainly by the color and the shape of the spots. The team has two backbones pretrained on ImageNet: a ResNet-50 (CNN, 25 million parameters) and a ViT-B/16 (86 million).

(a) Model and transfer (1.0). What kind of learning problem is this? Which backbone would you use and why? What would you freeze or adapt (linear probe, fine-tuning, LoRA), and in what order? State the input, the output and the loss.

(b) Training (1.0). Write, in Python/PyTorch (it may be schematic), the training loop with the optimizer, the learning rate schedule, the regularization you chose (justify the data augmentation for this domain), the validation in `model.eval()` and early stopping.

(c) Diagnosis (1.0). Another team trained only the new head for 6 epochs (AdamW, $\eta = 10^{-2}$). At epoch 7 it unfroze the whole backbone (changing nothing else) and got the curves below ($\ln 5 \approx 1.61$). What does the figure show? What would you change first, and why?



NAME: _____

	A	B	C	D	E
Q1					
Q2					
Q3					
Q4					
Q5					
Q6					
Q7					

	A	B	C	D	E
Q8					
Q9					
Q10					
Q11					
Q12					
Q13					
Q14					

Q15 (a) model and transfer:

NAME: _____

Q15 (b) training (schematic code is fine):

Q15 (c) diagnosis (use the back if needed):