Gabarito comentado

Avaliação Final (AF) · exemplo 2026.2

14 objetivas (7,0) + 1 dissertativa em três partes (3,0) = 10,0 pontos

Como usar. Resolva a prova inteira no papel, cronometrando 120 minutos, *antes* de abrir este gabarito. Confira as objetivas pela tabela e leia o comentário de cada uma, inclusive das que você acertou. Depois corrija a dissertativa com os critérios de cada etapa.

Como ler. ✓ **Por quê** explica a resposta certa; × **Armadilhas** mostra o erro das alternativas mais tentadoras. Cada questão indica a aula para revisar; os links abrem a aula no site.

Questões objetivas

Q1 a Q14 · 0,5 ponto cada

questão	1	2	3	4	5	6	7	8	9	10	11	12	13	14
resposta	B	D	A	E	C	B	E	A	D	C	A	C	B	D

Q1 B — Duas atualizações; $z = 0$ conta como 1

revisar: **4. Perceptron**

✓ **Por quê.** Os exemplos 1 ($z = 1$) e 2 ($z = -1$) já saem certos. No 3, $z = -1$ dá $\hat{y} = 0$ com $y = 1$: erro $+1$, $\mathbf{w} \leftarrow (1; -1) + 0{,}5\,(0; 1) = (1; -0{,}5)$ e $b \leftarrow 0{,}5$. No 4, $z = 1 - 1{,}5 + 0{,}5 = 0$; como $z \geq 0$ dá $\hat{y} = 1$, há erro -1: $\mathbf{w} \leftarrow (1; -0{,}5) - 0{,}5\,(1; 3) = (0{,}5; -2)$ e $b \leftarrow 0{,}5 - 0{,}5 = 0$.

× **Armadilhas. A**: tratar $z = 0$ como classe 0 pula a segunda atualização; o degrau da aula devolve 1 em $z \geq 0$. **C**: usar o erro com o sinal trocado ($\hat{y} - y$ no lugar de $y - \hat{y}$). **E**: atualizar $\mathbf{w}$ e esquecer o viés no exemplo 4. **D**: somar $\eta\,\mathbf{x}$ num falso positivo, quando o erro -1 manda subtrair.

Q2 D — XOR com uma camada oculta

revisar: **5. MLP**, **3. Redes Neurais**

✓ **Por quê.** h_1 só liga em $(1, 0)$ e h_2 só em $(0, 1)$; a saída é o OU dos dois. A coluna é 0, 1, 1, 0: o XOR, a função não linearmente separável que Minsky e Papert (1969) usaram contra o perceptron de uma camada. Com uma camada oculta ela sai; o que permitiu treinar redes assim a partir de dados foi a retropropagação de Rumelhart, Hinton e Williams (1986).

× **Armadilhas. E**: o XOR não é linearmente separável; nenhuma reta deixa $(0, 1)$ e $(1, 0)$ de um lado com $(0, 0)$ e $(1, 1)$ do outro. **C**: a sigmoide também é não linear; a aproximação universal (Cybenko, 1989) foi provada justamente para ela. **B**: em $(1, 1)$ as duas unidades ocultas ficam em 0.

Q3 A — A maior que não explode, dividida por dois

revisar: **6. Otimização**

✓ **Por quê.** É a busca que a aula recomenda: varrer η em potências de dez por algumas centenas de passos, ficar com a maior que não explode e dividir por dois. 10^{-1} passou do limite de estabilidade (a perda sobe desde o passo 12 e vira NaN); 10^{-2} desce rápido e estável; daí cerca de $5 \cdot 10^{-3}$ no treino longo.

× **Armadilhas. C**: perda caindo devagar e suave é o sintoma de η pequeno demais. **B**: a queda inicial de 10^{-1} não se sustenta; ela volta a subir antes do passo 20. **E**: no AdamW o passo tem tamanho da ordem de η qualquer que seja a escala do gradiente (ele divide pela raiz da média dos quadrados), então recortar o gradiente não encurta o passo; a aula manda dividir η por 10 primeiro.

Q4 E Validação aumentada mede outra coisa

revisar: **7. Regularização**

✓ **Por quê.** C e D têm o mesmo treino (92%); só a validação muda. Em C, cada foto de validação é recortada e recolorida ao acaso, então o número mede o modelo em fotos distorcidas, não nas que ele vai receber. Com a validação limpa (D), o mesmo modelo marca 91%, 5 pontos acima de B: o aumento de dados ajudou. A regra da aula é nunca aumentar a validação.

× **Armadilhas. A**: compara B com uma medida estragada. **C**: a validação deve imitar as fotos que chegam, sem sorteio; recorte e cor sorteados a cada avaliação não são o que o usuário envia. **B** e **D**: o alvo é a validação, não o treino; 100% no treino com 81% na validação é overfitting.

Q5 C O fator de uma camada é $n\sigma^2\rho$

revisar: **9. Redes Profundas**

✓ **Por quê.** Cada camada multiplica $\mathbb{E}[a^2]$ por $n\sigma^2\rho$, com $\rho = \frac{1}{2}$ na ReLU. Com $\sigma^2 = 0{,}01$ fixo: $100 \cdot 0{,}01 \cdot \frac{1}{2} = 0{,}5$, depois 1 e 2. Em 19 travessias, $0{,}5^{19} \approx 2 \cdot 10^{-6}$ e $2^{19} \approx 5 \cdot 10^{5}$, as ordens de grandeza do gráfico. Xavier e He põem n no denominador exatamente para cancelar a largura: com $\sigma^2 = 2/n$ o fator vale 1 em qualquer n.

× **Armadilhas. D**: esquecer o $\frac{1}{2}$ da ReLU. **B**: a ReLU não satura do lado positivo. **E**: nada foi treinado; a taxa de aprendizado não entra na inicialização. **A**: a profundidade é a mesma nas três redes.

Q6 B Formatos pela fórmula; pooling global

revisar: **10. Redes Convolucionais**

✓ **Por quê.** Com passo 2 e padding 1: $\lfloor (32 + 2 - 3)/2 \rfloor + 1 = 16$, então $32 \times 16 \times 16$; o segundo pooling leva a $32 \times 8 \times 8$ e o `Flatten` a 2.048 números. O `Linear` tem $2048 \cdot 10 + 10 = 20.490$ parâmetros. O pooling global tira a média de cada mapa: sobram 32 números, com $32 \cdot 10 + 10 = 330$ parâmetros.

× **Armadilhas. A**: esquecer o segundo `MaxPool2d`. **C**: esquecer o padding na convolução com passo 2 ($\lfloor 29/2 \rfloor + 1 = 15$, depois 7). **D**: esquecer o viés e tratar o pooling global como um número só (ele dá um por canal). **E**: o pooling global muda justamente a entrada da cabeça.

Q7 E LoRA treina $r(d + k)$ por matriz

revisar: **14. Transfer Learning**, **13. ViT**

✓ **Por quê.** Cada matriz 768×768 ganha B (768×4) e A (4×768): $r(d + k) = 4 \cdot 1536 = 6.144$. São 4 matrizes por bloco em 12 blocos: $48 \cdot 6.144 = 294.912$, cerca de 0,3% dos 86,6 milhões. O estado do AdamW acompanha os parâmetros *treináveis*: $294.912 \cdot 12 \approx 3{,}5$ MB, contra 86,6 milhões $\cdot\, 12 \approx 1$ GB no fine-tuning completo.

× **Armadilhas. A**: contar só um dos dois fatores. **B**: $\Delta W = BA$ nunca é guardada nem treinada; treinam-se B e A. **C**: congelar é justamente não ter gradiente nem estado de otimizador. **D**: os 12 bytes já incluem os dois momentos, guardados em fp32 com a cópia mestra.

Q8 A Escolher modos ou cobrir tudo

revisar: **16. Modelos Generativos**

✓ **Por quê.** Z custa 50 passadas por amostra: é o modelo de flow matching, cujo custo é o número de passos. Entre os dois de uma passada, X põe amostras nítidas em só 3 dos 8 modos: colapso de modos, o comportamento da KL reversa, típico da GAN. Y cobre o anel inteiro, inclusive o vale entre os aglomerados: o borrão da máxima verossimilhança (KL para frente), típico do VAE.

× **Armadilhas. C**: Y cobre todos os modos; colapso é o contrário. **B**: o borrão espalha as amostras, não as concentra. **E**: borrão e colapso são propriedades do objetivo, não de treino curto. **D**: 50 passadas por amostra é custo de difusão ou de fluxo; o VAE gera numa passada.

Q9 D — KL zero numa dimensão: ela não carrega nada

revisar: **17. VAE**

✓ **Por quê.** A KL de uma dimensão só é zero quando $q(z_i \mid x) = \mathcal{N}(0,1)$, o próprio prior, para toda imagem: ali o código é igual para todas as imagens e não diz nada sobre x. É o colapso do posterior em 4 das 8 dimensões; as outras 4 (cerca de 10 nats no total) carregam a informação.

× **Armadilhas. C**: $\sigma \to 0$ faz a KL explodir (o termo $-\log \sigma^2$), não zerar. **A**: KL perto de zero é justamente não carregar informação. **B**: o colapso total teria KL total perto de zero; aqui metade das dimensões trabalha. **E**: um β maior empurra mais dimensões para o prior; para reativá-las usa-se β menor, annealing da KL ou free bits.

Q10 C — A perda minimax satura quando D vence

revisar: **18. GANs**

✓ **Por quê.** `bce(`ℓ`, 0)` $= -\log(1 - D)$, então `loss_G` $= \log(1 - D(G(z)))$: a perda minimax do gerador. O gradiente dela no logit é $-D(G(z)) \approx -0{,}02$, quase nada, justamente quando o discriminador está vencendo. A perda não saturante, `bce(D(fake), ones)` $= -\log D(G(z))$, tem o mesmo ponto fixo e gradiente $D - 1 \approx -0{,}98$.

× **Armadilhas. A**: é o objetivo do artigo, mas o próprio artigo troca essa perda pela não saturante na prática. **D**: o sinal está certo (minimizar $\log(1 - D)$ empurra $D(G(z))$ para 1); tirar o menos faria o gerador ajudar o discriminador. **E**: no equilíbrio $D(G(z))$ fica perto de 0,5, não de 0,02. **B**: o `detach` é do passo do discriminador.

Q11 A — Softmax de 100 × cosseno

revisar: **19. CLIP**

✓ **Por quê.** Os logits são 29, 27 e 22. Dividindo tudo pelo maior, $p = 1/(1 + e^{-2} + e^{-7}) \approx 1/1{,}141 \approx 0{,}88$. A nova classe tem logit 20 e soma $e^{-9} \approx 0{,}0001$ ao denominador: o valor quase não muda. No zero-shot, uma classe nova é só mais um prompt e nada é treinado.

× **Armadilhas. B** e **C**: o cosseno não é probabilidade; normalizar os cossenos ignora a temperatura. **E**: cossenos de 0,2 a 0,35 são típicos de um par imagem–texto correto (o gap entre modalidades). **D**: os pesos do classificador zero-shot são os embeddings dos prompts.

Q12 C — Extrapolar a partir do incondicional

revisar: **20. Stable Diffusion**

✓ **Por quê.** $\tilde{\epsilon} = \epsilon_\theta(z_t, \varnothing) + w\,(\epsilon_\theta(z_t, c) - \epsilon_\theta(z_t, \varnothing))$. Componente 1: $0{,}10 + 7{,}5 \cdot 0{,}20 = 1{,}60$. Componente 2: a diferença é zero, então $\tilde{\epsilon} = -0{,}20$. O guidance amplifica só a direção em que o prompt muda a previsão.

× **Armadilhas. D**: partir da previsão condicional $(0{,}30 + 1{,}50)$ em vez da incondicional. **A**: isso é $w = 1$. **B** e **E**: multiplicar por w a previsão inteira, ou o termo que não depende do prompt.

Q13 B — A convenção muda x_t e o sentido do alvo

revisar: **21. Flow Matching**

✓ **Por quê.** Na aula, $x_t = (1 - t)\,x_0 + t\,x_1 = 0{,}75\,(1; -2) + 0{,}25\,(3; 2) = (1{,}5; -1)$ e o alvo é $x_1 - x_0 = (2; 4)$. No código do colega, $x_t = 0{,}75\,(3; 2) + 0{,}25\,(1; -2) = (2{,}5; 1)$ e a velocidade vai dos dados para o ruído: $(1; -2) - (3; 2) = (-2; -4)$. Misturar as duas convenções (um sampler de uma com um modelo da outra) dá lixo; é a primeira coisa a conferir num código alheio.

× **Armadilhas. E**: trocar a convenção sem inverter o sentido da velocidade. **A**: trocar os pesos do interpolante. **C**: confundir a velocidade com o dado ou o ruído, que são outros alvos possíveis. **D**: o interpolante não é simétrico; $t = 0{,}25$ fica perto de um dos extremos.

Q14 **D** **Qualidade por compute ou uma sequência só** revisar: **22. DiT**, **23. Geração AR**

✓ **Por quê.** Para texto para imagem isolado, os modelos abertos mais fortes são transformers de difusão ou de flow (SD3, FLUX.1, Qwen-Image): é o caminho para a melhor imagem por unidade de compute. Quando a imagem precisa viver no mesmo contexto que o texto (conversar, raciocinar, editar), o encaixe natural é um modelo único sobre uma sequência de texto e imagem: autorregressivo sobre tokens (Chameleon, Janus-Pro) ou com uma perda de difusão dentro do modelo de linguagem (Transfusion, BAGEL).

× **Armadilhas. B**: no MMDiT a imagem reescreve os tokens de texto, mas o modelo só prevê a imagem; ele não gera texto. **A**: verossimilhança exata não é qualidade de imagem. **C**: tokens discretos têm o teto do tokenizador; os modelos abertos que lideram em qualidade usam latentes contínuos. **E**: MaskGIT e VAR cortam passadas sequenciais; nenhum dos dois supera "qualquer DiT".

Questão 15 · Um projeto de deep learning de ponta a ponta

3,0 pontos · 1,0 por etapa

Resposta esperada, em uma frase por etapa.

■ **Modelo e transferência.** Classificação supervisionada em 5 classes sobre um backbone pré-treinado (a ResNet-50 é a escolha segura com 2.400 fotos), com cabeça nova de 5 saídas e entropia cruzada; primeiro um linear probe, depois fine-tuning com taxa pequena ou LoRA.

■ **Treino.** AdamW com warmup e cosseno, aumento de dados que preserva a cor, validação em `model.eval()` sem aumento e early stopping que restaura o melhor checkpoint.

■ **Diagnóstico.** Descongelar o backbone com $\eta = 10^{-2}$ destruiu as features pré-treinadas; o primeiro passo é voltar ao checkpoint da época 6 e ajustar o backbone com uma taxa de 100 a 1.000 vezes menor (ou com LoRA).

(a) Modelo e transferência — 1,0 ponto

Aprendizado supervisionado (os agrônomos deram os rótulos): classificação em 5 classes. As features vêm aprendidas do pré-treino, não desenhadas à mão.	**0,2**
Backbone pré-treinado, nunca treinado do zero com 2.400 fotos. A ResNet-50 é a escolha segura: o prior convolucional forte vence com poucos dados e ela é mais barata. O ViT-B/16 também vale se a justificativa for o pré-treino em escala (sem ele, o ViT perde para a CNN com poucos dados).	**0,25**
Ordem: primeiro um linear probe (encoder congelado, features calculadas uma vez, uma linha de base barata); depois fine-tuning completo ou dos últimos blocos com taxa pequena, ou LoRA. Com 1.920 fotos de treino, qualquer um dos dois, justificado, vale a nota.	**0,25**
Entrada $3 \times 224 \times 224$, normalizada como no pré-treino; saída de 5 logits (a cabeça antiga de 1.000 classes é trocada por `Linear(2048, 5)` ou `Linear(768, 5)`); perda de entropia cruzada com softmax.	**0,3**

(b) Treino — 1,0 ponto

Otimizador e agendamento: AdamW com weight decay, taxa pequena no backbone (perto de 10^{-5}) e maior na cabeça, warmup curto seguido de cosseno, com o agendador avançando a cada passo.	**0,25**
Regularização justificada pelo domínio: aumento de dados só no treino (espelhar e girar a folha não muda a doença), sem mexer no matiz, porque a cor é o que separa ferrugem de cercosporiose; weight decay.	**0,25**
Validação com `model.eval()` e `torch.no_grad()`, sem aumento, voltando a `model.train()` no treino.	**0,25**
Early stopping com paciência, guardando o melhor checkpoint da validação e restaurando-o no fim.	**0,25**

(c) Diagnóstico **1,0 ponto**

Leitura: tudo ia bem até a época 6 (validação perto de 83%); na época 7 as *duas* perdas saltam para perto de $\ln 5 \approx 1{,}61$ (a de um chute entre 5 classes) e a acurácia cai para perto de 30% sem voltar ao nível anterior. Não é overfitting: a perda de treino também piorou. **0,35**

Causa: $\eta = 10^{-2}$ servia para uma cabeça nova, mas é grande demais para pesos pré-treinados; os primeiros passos jogam fora as features que faziam o modelo funcionar (esquecimento catastrófico). **0,3**

Primeira mudança: voltar ao checkpoint da época 6 e ajustar o backbone com taxa de 100 a 1.000 vezes menor (10^{-5} a 10^{-4}, num grupo de parâmetros separado da cabeça), com warmup curto; ou usar LoRA ou descongelar só os últimos blocos. Porque os pesos pré-treinados já estão perto de uma boa solução: o ajuste fino deve mexer pouco neles. **0,35**

Detalhes de sintaxe não descontam. Treinar do zero zera o item do backbone em (a). Aplicar aumento de dados na validação zera o item da validação em (b). Em (c), chamar a queda de overfitting zera o item da leitura.

Um código que atende a (b), entre outros possíveis (os `imports` e a montagem dos `DataLoaders` foram omitidos):

```
model = torchvision.models.resnet50(weights='IMAGENET1K_V2')
model.fc = nn.Linear(2048, 5)                          # cabeça nova: 5 classes
treino_tf = T.Compose([T.RandomResizedCrop(224, scale=(0.7, 1.0)),
    T.RandomHorizontalFlip(), T.RandomVerticalFlip(), T.RandomRotation(20),
    T.ColorJitter(brightness=0.2, contrast=0.2),       # sem hue: a cor é a pista
    T.ToTensor(), T.Normalize(MEDIA, DESVIO)])         # MEDIA e DESVIO do ImageNet
val_tf = T.Compose([T.Resize(256), T.CenterCrop(224),  # validação sem aumento
    T.ToTensor(), T.Normalize(MEDIA, DESVIO)])

corpo = [p for n, p in model.named_parameters() if not n.startswith('fc.')]
opt = torch.optim.AdamW([{'params': corpo, 'lr': 3e-5},
                         {'params': model.fc.parameters(), 'lr': 1e-3}], weight_decay=0.05)
total, warmup = EPOCAS * len(treino_dl), len(treino_dl)
def fator(s):                                          # warmup linear, depois cosseno
    if s < warmup: return (s + 1) / warmup
    return 0.5 * (1 + math.cos(math.pi * (s - warmup) / (total - warmup)))
sched = torch.optim.lr_scheduler.LambdaLR(opt, fator)
perda = nn.CrossEntropyLoss()

melhor, paciencia, sem_melhora = float('inf'), 5, 0
for epoca in range(EPOCAS):
    model.train()
    for x, y in treino_dl:
        loss = perda(model(x), y)
        opt.zero_grad(); loss.backward(); opt.step(); sched.step()   # sched por passo
    model.eval()
    with torch.no_grad():
        val = sum(perda(model(x), y).item() * len(y) for x, y in val_dl) / len(val_dl.dataset)
    if val < melhor:
        melhor, sem_melhora = val, 0
        torch.save(model.state_dict(), 'melhor.pt')    # guarda o melhor checkpoint
    else:
        sem_melhora += 1
        if sem_melhora == paciencia:
            break
model.load_state_dict(torch.load('melhor.pt'))         # restaura o melhor
```

Erros comuns. Treinar a rede do zero com 2.400 fotos. Aumento de dados com `hue` forte, que apaga a pista da cor. Aplicar o aumento também na validação. Esquecer o `model.eval()`. Parar o treino sem restaurar o melhor checkpoint. Usar o teste da próxima safra para escolher o modelo. Em (c), chamar a queda de overfitting ou de "dados ruins" sem notar que ela começa exatamente na época em que o backbone foi descongelado.

Revisar: **1. Conceitos**, **6. Otimização**, **7. Regularização**, **10. Redes Convolucionais**, **13. Vision Transformers** e **14. Transfer Learning**.