Examen Final

● Graded

Student

Luis Méndez Lázaro

Total Points

25.5 / 26 pts

Question 1

DeepLab vs PSPNet

1.5 / 1.5 pts

✔ **+ 1.5 pts Correcto!**

+ 0 pts Respuesta: b) *DeepLab utiliza atrous convolutions para ampliar el campo receptivo sin disminuir la resolución, mientras que PSPNet incorpora un pyramid pooling module que integra información contextual en múltiples escalas.*

DeepLab utiliza atrous (o dilated) convolutions para ampliar el campo receptivo sin reducir la resolución. PSPNet emplea un pyramid pooling module que aplica pooling a diferentes escalas para integrar información contextual de distintas áreas de la imagen.

Question 2

DeepLabV3+

1.5 / 1.5 pts

✔ **+ 1.5 pts Correcto!**

+ 0 pts Respuesta: c) *Mejora la precisión en la segmentación de detalles finos al fusionar características de alto nivel con información espacial de bajo nivel a través de un diseño encoder-decoder.*

Introduce un diseño encoder-decoder que no solo se basa en atrous convolutions para capturar contexto, sino que también fusiona características de alto nivel (capturadas en etapas profundas) con información espacial de bajo nivel (detalles finos de las primeras capas).

Question 3

SSIM

1.5 / 1.5 pts

✔ **+ 1.5 pts Correcto!**

+ 0 pts Respuesta: d) *SSIM mide la similitud estructural incorporando luminancia, contraste y estructura, lo que lo hace adecuado para evaluar la fidelidad en la reconstrucción de escenas en Depth Estimation.*

El SSIM (Structural Similarity Index) es una métrica diseñada para evaluar la similitud estructural entre dos imágenes, considerando tres componentes principales:

- Luminancia: Compara la intensidad promedio de las imágenes.
- Contraste: Evalúa la variación de intensidad en cada imagen.
- Estructura: Mide la correlación en la distribución de las intensidades, lo que resulta fundamental para capturar la relación espacial entre píxeles.

Question 4

SimCLR vs Triple Loss

1.5 / 1.5 pts

✔ **+ 1.5 pts Correcto!**

+ 0 pts Respuesta: c) *Se beneficia de un mayor tamaño de batch, lo que posibilita la inclusión de numerosos ejemplos negativos implícitos y mejora la discriminación en el espacio latente.*

SimCLR se apoya en data augmentation para generar pares positivos (dos vistas diferentes de la misma imagen) y utiliza el resto de las muestras en el batch como negativos implícitos. Una de las ventajas clave de SimCLR es que se beneficia de batch sizes grandes, lo que permite disponer de una gran cantidad de ejemplos negativos en cada iteración.

Question 5

BYOL

1.5 / 1.5 pts

✔ **+ 1.5 pts Correcto!**

+ 0 pts Respuesta: a) *Se fundamenta en una arquitectura teacher–student en la que el student aprende de un teacher por producto punto, eliminando la necesidad de las muestras negativas.*

BYOL se emplea un encoder online (student) y un encoder target (teacher), donde el teacher se actualiza mediante un promedio móvil (momentum) del student.

El entrenamiento se centra en minimizar la distancia entre las representaciones proyectadas del student y las del teacher para vistas (augmentations) diferentes de la misma imagen. Esta estrategia evita la necesidad de generar explícitamente pares negativos, ya que se basa únicamente en la consistencia entre dos vistas positivas de la misma imagen.

Question 6

iBOT vs BEiT

1.5 / 1.5 pts

✔ **+ 1.5 pts Correcto!**

+ 0 pts Respuesta: b) *iBOT incorpora señales de self-distillation que permiten aprender tanto del masking como de una supervisión implícita, mientras que BEiT se basa exclusivamente en la predicción de tokens.*

BEiT se basa en el enmascaramiento de imágenes y en la predicción de tokens preentrenados (tokenizer visual) para aprender representaciones.

iBOT extiende esta idea al incorporar señales de self-distillation además del masking, lo que permite que el modelo aprenda de dos fuentes: la predicción de tokens (masking), y la supervisión implícita derivada de self-distillation.

Question 7

CLIP

1.5 / 1.5 pts

✔ **+ 1.5 pts Correcto!**

+ 0 pts Respuesta: d) *La simetría entre el encoders de imagen y texto, combinada con una función de pérdida contrastiva que maximiza la similitud entre pares emparejados y separa los no emparejados.*

CLIP se entrena utilizando un gran conjunto de pares imagen-texto mediante una función de pérdida contrastiva. La simetría en la arquitectura es un punto clave en esta arquitectura, ya que se utilizan encoders separados (uno para imagen y otro para texto) que producen representaciones en espacios latentes compatibles.

Question 8

β-VAE

1.5 / 1.5 pts

✔ **+ 1.5 pts** **Correcto!**

+ 0 pts **Respuesta: c)** *Se refuerza la disentanglement de la representación latente.*

Al establecer $\beta > 1$, se aumenta la penalización sobre la divergencia entre la distribución latente y la distribución prior (usualmente una gaussiana). Esto fuerza al modelo a aprender una representación latente más disentangled, es decir, donde las variables latentes capturan de manera más independiente los factores de variación en los datos.

Question 9

SimSiam

3.5 / 3.5 pts

✔ **+ 3.5 pts Correcto!**

+ 0 pts

```
import torch
import torch.nn as nn
import torch.optim as optim

def negativeCosineSimilarity(p, z):
    p_norm = p / p.norm(dim=1, keepdim=True)
    z_norm = z / z.norm(dim=1, keepdim=True)
    return -(p_norm * z_norm).sum(dim=1).mean()

class SimSiamModel(nn.Module):
    def __init__(self, backbone):
        super().__init__()
        self.backbone = backbone            # encoder
        self.projection = ProjectionMLP()    # proyección
        self.prediction = PredictionMLP()    # predicción

    def forward(self, x):
        h = self.backbone  (x)
        z = self.projection(h)
        p = self.prediction(z)
        return z, p


backbone = Backbone()
model    = SimSiamModel(backbone)
optimizer = optim.Adam(model.parameters(), lr=3e-4)

for epoch in range(num_epochs):
    model.train()
    for (x1, x2) in dataloader:
        # x1, x2: dos vistas aumentadas de la misma imagen
        z1, p1 = model(x1)  # z1 = proyección, p1 = predicción
        z2, p2 = model(x2)  # z2 = proyección, p2 = predicción

        loss_12 = negativeCosineSimilarity(p1, z2.detach())
        loss_21 = negativeCosineSimilarity(p2, z1.detach())
        loss = 0.5 * (loss_12 + loss_21)

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
```

+ 3 pts Click here to replace this description.

+ 2.5 pts Click here to replace this description.

+ 1 pt Click here to replace this description.

Question 10

iBOT

3.5 / 3.5 pts

✔ **+ 0.5 pts** **Masking Correcto!**

✔ **+ 1 pt** **CLS Loss Correcto!**
Loss cruzado. Debe ser simetrico.

✔ **+ 1 pt** **Patch Loss Correcto!**
Loss en la misma vista. Debe ser simetrico.

✔ **+ 1 pt** **EMA Correcta!**

+ 0 pts

```
import torch
import torch.nn as nn
import torch.optim as optim
import torch.nn.functional as F
import copy

class iBOTModel(nn.Module):
    def __init__(self, backbone):
        super().__init__()
        self.backbone = backbone  # Tipo ViT

    def forward(self, x, mask=None):
        # mask indica que tokens se enmascaran
        cls_tokens, patch_tokens = self.backbone(x, mask=mask)
        return cls_tokens, patch_tokens

def iBOT_loss(student_cls, teacher_cls,
              student_patch, teacher_patch,
              mask, lambda_cls=1.0, lambda_patch=1.0):
    L_cls = F.cross_entropy(student_cls, teacher_cls.argmax(dim=1))
    masked_idx = mask.bool()  # [B, N_patches]

    student_masked = student_patch[masked_idx]
    teacher_masked = teacher_patch[masked_idx]

    if student_masked.shape[0] > 0:
        L_patch = F.cross_entropy(student_masked, teacher_masked.argmax(dim=1))
    else:
        # Si no hay patches enmascarados
        L_patch = torch.tensor(0.0, device=student_cls.device)

    loss = lambda_cls * L_cls + lambda_patch * L_patch
    return loss


# Parameters
n_epochs   =   10
lr         = 1e-3
emaMomemtum=0.999


backbone_student = Backbone()
backbone_teacher = copy.deepcopy(backbone_student)

student_model = iBOTModel(backbone_student)
```

```
teacher_model = iBOTModel(backbone_teacher)

# teacher sin gradientes
teacher_model.eval()
for param in teacher_model.parameters():
    param.requires_grad = False

optimizer = optim.Adam(student_model.parameters(), lr=lr)
for epoch in range(n_epochs):
    student_model.train()
    running_loss = 0.0

    for images in dataloader:
        # Generar vista global (sin alterar) y vistas enmascaradas
        global_view, masked_view, mask = create_masked_views(images)

        # Forward teacher
        with torch.no_grad():
            teacher_cls, teacher_patch = teacher_model(global_view, mask=None)

        # Forward student
        student_cls, student_patch = student_model(masked_view, mask=mask)

        loss = iBOT_loss(
            student_cls, teacher_cls,
            student_patch, teacher_patch,
            mask
        )

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        with torch.no_grad():
            for t_param, s_param in zip(teacher_model.parameters(), student_model.parameters()):
                t_param.data = emaMomemtum * t_param.data + (1.0 - emaMomemtum) * s_param.data
```

+ 0.5 pts Click here to replace this description.

+ 0.5 pts Click here to replace this description.

Question 11

CycleGAN

3.5 / 3.5 pts

✔ **+ 3.5 pts Correcto!**

+ 0 pts

```
import torch
import torch.nn as nn
import torch.optim as optim

MSELoss = nn.MSELoss()
L1Loss  = nn.L1Loss()

def generator_adversarial_loss(fake_pred):
   # real = 1
   target_real = torch.ones_like(fake_pred)
   return MSELoss(fake_pred, target_real)

def discriminator_adversarial_loss(real_pred, fake_pred):
   # real_pred = 1    fake_pred = 0.0
   target_real = torch.ones_like(real_pred)
   target_fake = torch.zeros_like(fake_pred)
   loss_real = MSELoss(real_pred, target_real)
   loss_fake = MSELoss(fake_pred, target_fake)
   return 0.5 * (loss_real + loss_fake)

# Parameters
n_epochs     = 100,
lambda_cycle = 10.0,
lr           =2e-4

optimizer_G = optim.Adam(
   list(G_AB.parameters()) + list(G_BA.parameters()),
   lr=lr, betas=(0.5, 0.999)
)
optimizer_D_A = optim.Adam(D_A.parameters(), lr=lr, betas=(0.5, 0.999))
optimizer_D_B = optim.Adam(D_B.parameters(), lr=lr, betas=(0.5, 0.999))

for epoch in range(n_epochs):
   for real_A, real_B in zip(dataloaderA, dataloaderB):
      #------------------------------------------------------------
      # Generator loss
      #------------------------------------------------------------
      optimizer_G.zero_grad()

      fake_B = G_AB(real_A)
      rec_A = G_BA(fake_B)

      fake_A = G_BA(real_B)
      rec_B = G_AB(fake_A)

      # Adversarial loss para G_AB
      pred_fake_B = D_B(fake_B)
      loss_GAB_adv = generator_adversarial_loss(pred_fake_B)

      # Adversarial loss para G_BA
      pred_fake_A = D_A(fake_A)
      loss_GBA_adv = generator_adversarial_loss(pred_fake_A)
```

```
        # Cycle-consistency loss
        loss_cycle_A = L1Loss(rec_A, real_A)
        loss_cycle_B = L1Loss(rec_B, real_B)
        loss_cycle   = loss_cycle_A + loss_cycle_B

        # Total loss
        loss_G = loss_GAB_adv + loss_GBA_adv + lambda_cycle * loss_cycle
        loss_G.backward()
        optimizer_G.step()

        #-----------------------------------------------------------
        # Discriminator loss
        #-----------------------------------------------------------
        optimizer_D_A.zero_grad()

        pred_real_A = D_A(real_A)
        pred_fake_A = D_A(fake_A.detach())
        loss_D_A = discriminator_adversarial_loss(pred_real_A, pred_fake_A)
        loss_D_A.backward()
        optimizer_D_A.step()

        optimizer_D_B.zero_grad()

        pred_real_B = D_B(real_B)
        pred_fake_B = D_B(fake_B.detach())
        loss_D_B = discriminator_adversarial_loss(pred_real_B, pred_fake_B)
        loss_D_B.backward()
        optimizer_D_B.step()
```

+ 3 pts Click here to replace this description.

+ 2.5 pts Click here to replace this description.

+ 1.5 pts Click here to replace this description.

Question 12

LLM **3** / 3.5 pts

+ 3.5 pts **Correcto!**

✔ **+ 3 pts** Click here to replace this description.

+ 2.5 pts Click here to replace this description.

+ 2 pts Click here to replace this description.

+ 1.5 pts Click here to replace this description.

+ 0 pts Click here to replace this description.

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Deep Learning
Examen Final
Pregrado 2025-o

Profesor: Victor Flores Benites

Apellidos: Méndez Lázaro Nombres: Luis Fernando

Sección: 1 Fecha: 27/02/2025 Nota:

Indicaciones:
La Duración es de **120 minutos**.
La evaluación consta de **12 preguntas**.

Pregunta 1 (1.5 puntos)
¿Cuál es la principal diferencia en la forma en que DeepLab y Pyramid Scene Parsing Network (PSPNet) abordan la integración del contexto en las imágenes?

a) DeepLab utiliza técnicas de pooling pyramid para capturar el contexto, mientras que PSPNet se apoya en convolutions atrous para preservar la resolución.
b) DeepLab utiliza atrous convolutions para ampliar el campo receptivo sin disminuir la resolución, mientras que PSPNet incorpora un pyramid pooling module que integra información contextual en múltiples escalas.
c) Ambas arquitecturas combinan pooling pyramid y atrous convolutions de manera idéntica para aumentar el campo receptivo sin perder detalles.
d) Ambas arquitecturas utilizan atrous convolutions, pero DeepLab añade un módulo de residual learning para refinar la segmentación, mientras que PSPNet no lo hace.
e) PSPNet se basa en un módulo de atrous spatial pyramid pooling (ASPP) para mejorar la segmentación, mientras que DeepLab depende únicamente de técnicas de upsampling clásico.

Pregunta 2 (1.5 puntos)
¿Qué ventaja aporta este diseño de DeepLabV3+ con respecto a enfoques previos?

a) Facilita la integración de información global, eliminando la necesidad de atrous convolutions en la red.
b) Elimina la necesidad de aplicar técnicas de regularización en el proceso de entrenamiento.
c) Mejora la precisión en la segmentación de detalles finos al fusionar características de alto nivel con información espacial de bajo nivel a través de un diseño encoder-decoder.
d) Utiliza pooling pyramid de manera exclusiva para incrementar la generalización, sin modificar la resolución de la imagen.
e) Implementa un mecanismo de atención que sustituye las atrous convolutions, lo que optimiza únicamente la velocidad de inferencia.

Pregunta 3 (1.5 puntos)
¿Cuál es el rol del SSIM en la evaluación de la calidad en depth-estimation?

a) SSIM se utilizado para calcular el error absoluto entre las imágenes de profundidad estimadas y las reales.
b) SSIM es una métrica que se centra en la comparación de la luminancia, e invariante a la estructura y el contraste.
c) SSIM se utiliza exclusivamente para calcular el error absoluto entre las imágenes de profundidad estimadas y las reales.
d) SSIM mide la similitud estructural incorporando luminancia, contraste y estructura, lo que lo hace adecuado para evaluar la fidelidad en la reconstrucción de escenas en Depth Estimation.
e) En Depth Estimation, el SSIM es reemplazado por el Mean Squared Error (MSE) ya que este último ofrece una información más detallada.

Pregunta 4 (1.5 puntos)
¿Cuál es la ventaja principal de SimCLR en comparación con Triplet Loss?

a) Permite entrenar sin necesidad de pares negativos, centrándose en la relación entre pares positivos.
b) Emplea un mecanismo de teacher–student para transferir el conocimiento de manera directa, evitando el uso de funciones de pérdida contrastiva.
c) Se beneficia de un mayor tamaño de batch, lo que posibilita la inclusión de numerosos ejemplos negativos implícitos y mejora la discriminación en el espacio latente.
d) Reduce la dependencia de data augmentation, ya que se enfoca en la normalización de las entradas para generar invariancias.
e) Se basa en una función de pérdida simétrica que penaliza por igual la distancia entre todas las muestras.

Pregunta 5 (1.5 puntos)

¿Cuál es la innovación que permite a BYOL no requerir la formación explícita de pares negativo?

a) Se fundamenta en una arquitectura teacher–student en la que el student aprende de un teacher por producto punto, eliminando la necesidad de las muestras negativas.
b) Emplea un mecanismo de data augmentation intensivo para generar negativos implícitos a partir de variaciones sutiles.
c) Utiliza exclusivamente técnicas de clustering para agrupar representaciones similares, evitando la comparación directa de pares.
d) Depende de una normalización interna que automáticamente ignora las diferencias entre muestras similares, sin emplear loss contrastiva.

Pregunta 6 (1.5 puntos)

¿Cuál es la innovación principal que introduce iBOT en comparación con BEiT?

a) iBOT reemplaza el uso de un tokenizer visual preentrenado con un mecanismo de clustering dinámico para generar tokens..
b) iBOT incorpora señales de self-distillation que permiten aprender tanto del masking como de una supervisión implícita, mientras que BEiT se basa exclusivamente en la predicción de tokens.
c) iBOT elimina la necesidad de enmascarar regiones de la imagen, utilizando únicamente un encoder simétrico para predecir la imagen completa.
d) iBOT se centra en la reconstrucción pixel a pixel de la imagen en lugar de aprender representaciones latentes robustas, a diferencia de BEiT.

Pregunta 7 (1.5 puntos)

¿Qué factor permite a CLIP obtener representaciones robustas y coherentes entre imagen y texto?

a) Uso intensivo de data augmentation únicamente en el encoder de imagen, ignorando las transformaciones en el texto.
b) La integración de un framework teacher–student dentro del entrenamiento, donde el teacher proporciona etiquetas suaves para la convergencia del modelo.
c) La utilización de loss functions que penalizan de forma idéntica tanto la similitud semántica como la magnitud de las representaciones.
d) La simetría entre el encoders de imagen y texto, combinada con una función de pérdida contrastiva que maximiza la similitud entre pares emparejados y separa los no emparejados.

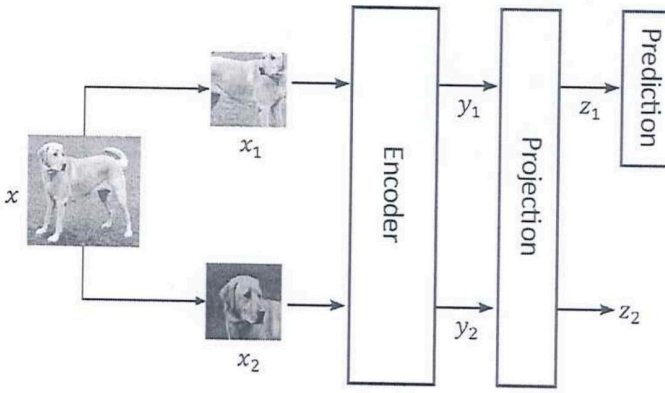
Pregunta 8 (1.5 puntos)

β-VAE introduce un factor que pondera el efecto de KL-divergence en el loss de la VAE. ¿Cuál es el efecto de establecer β>1?

a) Se mejora la calidad de la reconstrucción sin afectar las representaciones aprendidas.
b) Se favorece una mayor capacidad de reconstrucción.
c) Se refuerza la disentanglement de la representación latente.
d) Se reduce la sensibilidad a las variaciones de los datos, eliminando la necesidad de data augmentation.
e) Genera representaciones latentes simétricas.

Pregunta 9 (3.5 puntos)

Crea un pseucodigo para entrenamiento con SimSiam. Asuma que puede importar el backbone de su elección.



$$\mathcal{D}(p_1, z_2) = -\frac{p_1}{\|p_1\|_2} \cdot \frac{z_2}{\|z_2\|_2}$$

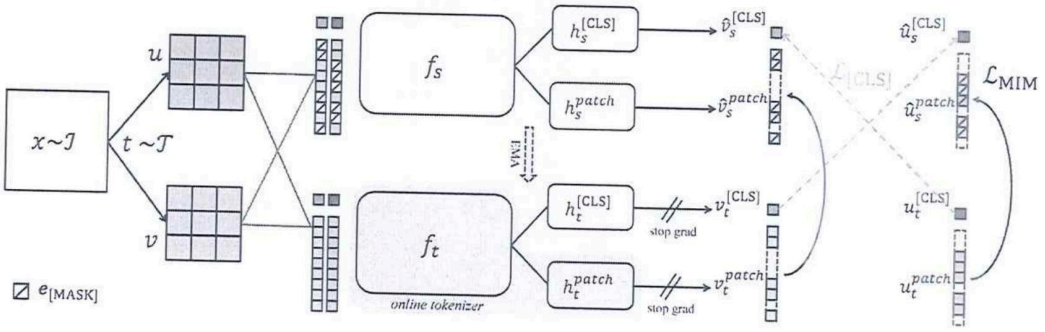
$$\mathcal{L} = \frac{1}{2}\mathcal{D}(p_1, z_2) + \frac{1}{2}\mathcal{D}(p_2, z_1)$$

```
for x in dataloader
    x1, x2 = augmentation(x), augmentation(x)
    y1 = Encoder(x1)
    y2 = Encoder(x2)
    z1 = Projection(y1)
    z2 = Projection(y2)
    p1 = Prediction(z1)
    p2 = Prediction(z2)
    L = 1/2 D(p1, z2) + 1/2 D(p2, z1)
    L.backward()
Update Encoder and Projection
```

```
def D(s, t)
    t = t.detach()
    s = l2-normalization(s)
    t = l2-normalization(t)
    return - s·t
    # - s/|s|_2 · t/|t|_2
    # without normalization
```

Pregunta 10 (3.5 puntos)

Crea un pseucodigo para entrenamiento con iBoT. Asuma que puede importar el backbone de su elección.



$$\mathcal{L}_{\text{cls}} = -\sum_{c} q^{c}_{cls} \log p^{c}_{cls} \quad \checkmark$$

$$\mathcal{L}_{\text{patch}} = -\sum_{m}\sum_{c} q^{c}_{m} \log p^{c}_{m} \quad \checkmark\checkmark$$

$$\mathcal{L}_{\text{iBOT}} = \lambda_1 \mathcal{L}_{\text{cls}} + \lambda_2 \mathcal{L}_{\text{patch}}$$

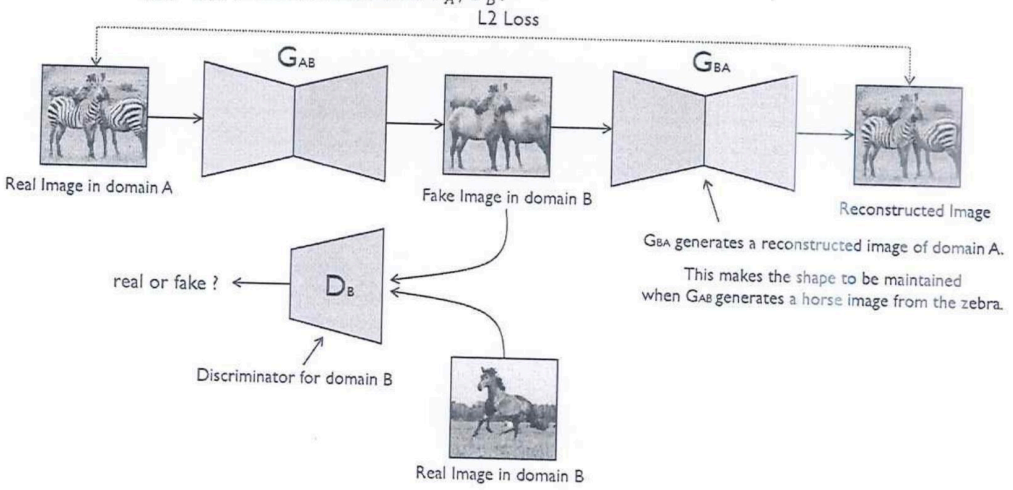
donde c recorre todas las vistas de la imagen original y m recorre los tokens enmascarados.

```
for x in dataloader:
    u, v = augmentation(x), augmentation(x)
    û, v̂ = bitwise_mask(u), bitwise_mask(v)

    # student network (MLPs included)
    v̂_s^[CLS], v̂_s^patch = f_s(v̂)        # f_s + h_s
    û_s^[CLS], û_s^patch = f_s(û)

    # teacher network (MLPs included)
    v_t^[CLS], v_t^patch = f_t(v)        # f_t + h_t
    u_t^[CLS], u_t^patch = f_t(u)
    # losses
    L_CLS = [H(v̂_s^[CLS], u_t^[CLS]) + H(û_s^[CLS], v_t^[CLS])]/2

    L_patch = [H(v̂_s^patch, v_t^patch) +
               H(û_s^patch, u_t^patch)]/2

    L_iBOT = λ1 L_cls + λ2 L_patch
    L_iBOT.backward()
    update(f_s)
    f_t.params = l f_t.params + (1-l) f_s.params   # EMA :)
    update c
```

```
def H(s, t)
    t = t.detach()
    s = softmax(s / τ_s)
    t = softmax(t-c/ τ_t)
    return cross_entropy(s, t)
```

Pregunta 11 (3.5 puntos)

Crea un pseucodigo para entrenamiento de la CycleGAN. Asuma que puede importar las redes generadoras G_{AB}, G_{BA}; y discriminadoras D_A, D_B.



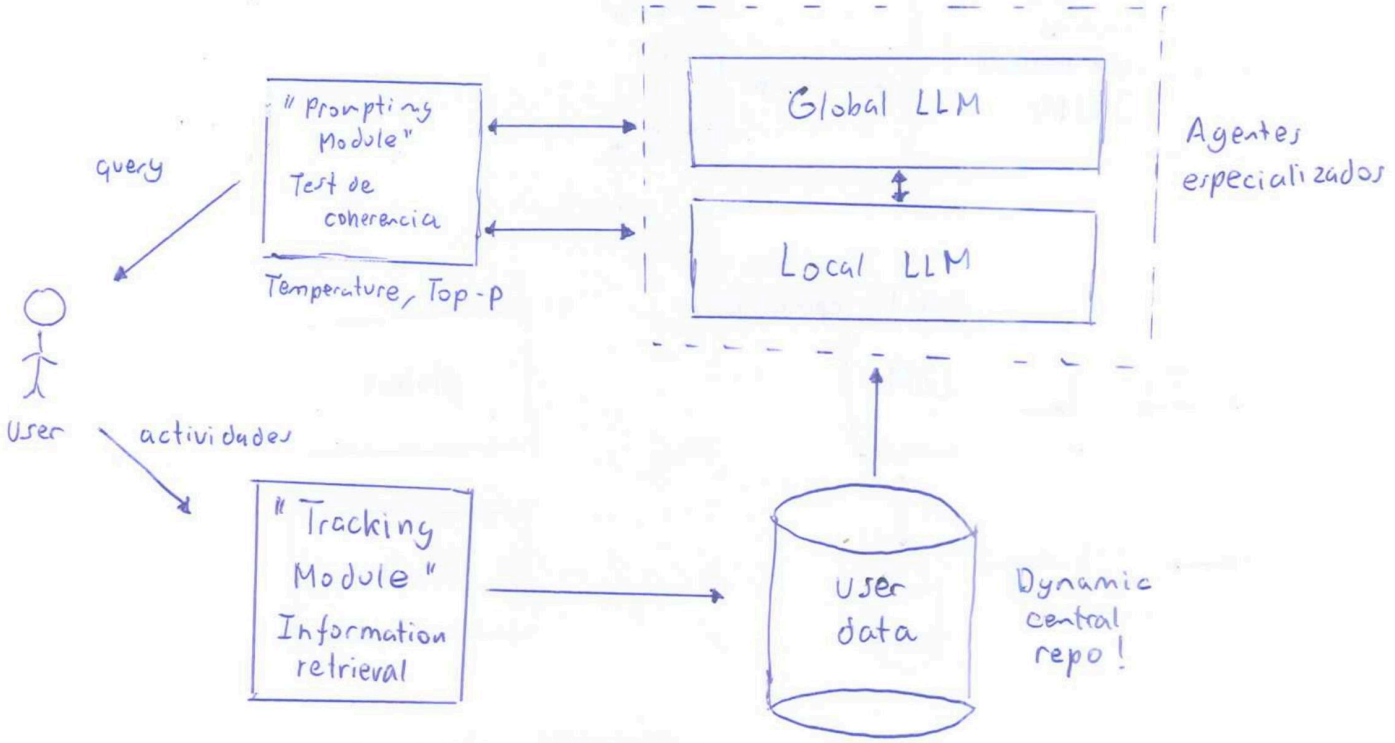
```
for A, B in dataloader:

    real_labels = torch.ones()
    fake_labels = torch.zeros()

    fake_B = G_AB(A)
    fake_A = G_BA(B)
    # train generators G_AB, G_BA
    loss_G_AB = criterion_gan( D_B(fake_B), real_labels )
    loss_G_BA = criterion_gan( D_A(fake_A), real_labels )
    recov_A = G_BA(fake_B)
    recov_B = G_AB(fake_A)
    loss_cycle_B = criterion_cycle(recov_A, A)
    loss_cycle_A = criterion_cycle(recov_B, B)
    loss_gen = loss_G_AB + loss_G_BA + λ(loss_cycle_B + loss_cycle_A)
    loss_gen.backward()
    optim_gen.step()  # update G_AB, G_BA parameters

    # train discriminator D_B
    loss_real_D_B = criterion_gan(D_B(B), real_labels)
    loss_fake_D_B = criterion_gan(D_B(fake_B.detach()), fake_labels)
    loss_D_B = (loss_real_D_B + loss_fake_D_B)/2
    loss_D_B.backward()
    optim_D_B.step()
    # train discriminator D_A
    loss_real_D_A = criterion_gan( D_A(A), real_labels)
    loss_fake_D_A = criterion_gan( D_A(fake_A.detach()), fake_labels)
    loss_D_A = (loss_real_D_A + loss_fake_D_A)/2
    loss_D_A.backward()
    optim_D_A.step()
```

Pregunta 12 (3.5 puntos)

Considere el diseño de un sistema de coaching virtual que interactúa en tiempo real con el usuario y, al mismo tiempo, realiza un seguimiento detallado de sus actividades, generando una planificación a dos niveles: una estrategia global que oriente el progreso a largo plazo y una planificación local para cada sesión. El sistema debe contar con agentes especializados que operen de forma independiente pero compartan información de un repositorio central dinámico de datos del usuario, permitiendo que cada módulo aporte su perspectiva sin perder la coherencia general.

Proponga un pipeline integral que articule estos componentes, definiendo cómo se intercambiarán y actualizarán los datos entre agentes y cómo se abordarán los posibles conflictos o inconsistencias en la información.

Pipeline: "Sistema de coaching virtual"



Propongo un sistema basado en RAG donde ambos agentes (LLMs) usan datos externos sobre el usuario. El comportamiento de los agentes es dinámico y pueden ser actualizados dependiendo de la coherencia del prompt final (Chain of Thought sobre ambos modelos). Ambos modelos tienen un temperature ≈ 0.9 para que exista variabilidad en las respuestas y a su vez coherencia.