

# Examen Parcial

● Graded

Student

Luis Méndez Lázaro

Total Points

20.5 / 25 pts

Question 1

CNN

1.5 / 1.5 pts

✓ + 1.5 pts Correcto!

+ 0 pts Respuesta: b) Las capas de pooling ayudan a que las representaciones sean invariantes a pequeñas traslaciones en la entrada.

Las capas de pooling reducen la sensibilidad de las representaciones a pequeñas variaciones o traslaciones en la entrada, permitiendo que el modelo mantenga características importantes independientemente de ligeros desplazamientos.

Question 2

ResNet

1.5 / 1.5 pts

✓ + 1.5 pts Correcto!

+ 0 pts Respuesta: c) Implementar conexiones residuales para afrontar el vanishing gradient.

ResNet introdujo las residual connections que permiten el flujo directo de información y gradientes a través de la red. Esto ayuda a mitigar el problema del vanishing gradient, facilitando el desarrollo de redes profundas.

Question 3

LSTM

1.5 / 1.5 pts

✓ + 1.5 pts Correcto!

+ 0 pts Respuesta: d) Determina cuánto del estado previo de la celda se debe conservar.

Forget gate en un LSTM controla la cantidad de información del cell state previo que se mantiene o descarta. Esto permite que la red olvide información irrelevante y retenga solo lo que es necesario para futuras predicciones.

Question 4

Transformers vs RNN

1.5 / 1.5 pts

✓ + 1.5 pts Correcto!

+ 0 pts Respuesta: c) Los Transformers dependen de mecanismos de atención, lo que permite un entrenamiento paralelo en la secuencia.

Los Transformers utilizan mecanismos de atención que permiten procesar todos los tokens de una secuencia simultáneamente, lo que facilita el entrenamiento paralelo. A diferencia de las RNNs, que procesan secuencias de manera secuencial, los Transformers son más eficientes y escalables para manejar entradas largas.

#### Question 5

##### MobileNet

0 / 1.5 pts

+ 1.5 pts Correcto!

✓ + 0 pts Respuesta: a) *Convoluciones separables en profundidad que factoriza la convolución estándar.*

MobileNet descomponen la convolución vanilla en una convolución depthwise seguida de una convolución pointwise ( $1 \times 1$  convolution). Esta factorization reduce significativamente el número de parámetros y la carga computacional.

#### Question 6

##### U-net

1.5 / 1.5 pts

✓ + 1.5 pts Correcto!

+ 0 pts Respuesta: b) *Recuperar información espacial perdida durante el downsampling.*

En U-Net, las skip connections conectan capas correspondientes del encoder y el decoder, permitiendo que la red recupere detalles espaciales que se pierden durante el proceso de downsampling.

#### Question 7

##### LSTM vs GRU

1.5 / 1.5 pts

✓ + 1.5 pts Correcto!

+ 0 pts Respuesta: d) *GRU carece de cell state.*

Las GRU (Gated Recurrent Units) simplifican la arquitectura de las LSTM al eliminar la cell state y combinar las funciones de las puertas en una única puerta de actualización, manteniendo solo el hidden state. Esto hace que las GRU sean más eficientes en términos de parámetros y computación.

#### Question 8

##### Faster R-CNN

1.5 / 1.5 pts

✓ + 1.5 pts Correcto!

+ 0 pts Respuesta: a) *Generar propuestas de regiones candidatas que probablemente contengan objetos.*

Region Proposal Network (RPN) se encarga de generar propuestas de regiones que tienen una alta probabilidad de contener objetos. Estas propuestas son utilizadas posteriormente por la red para clasificar y refinar las bounding boxes, mejorando así la eficiencia y precisión de la detección de objetos.

## Question 9

## Multi-head Self-attention

2.5 / 2.5 pts

✓ + 0.5 pts Linear Projections Correcto!

✓ + 1 pt Scaled Dot-Product Attention Correcto!

✓ + 1 pt Multi-head Correcto!

+ 0 pts **Multi-head Self-Attention** es un mecanismo detrás de los Transformer, el cual permite capturar diferentes tipos de relaciones y dependencias entre elementos de la secuencia, procesándolos en paralelo en múltiples subespacios de representación.

**1. Linear Projections:**

Los vectores de entrada se transforman mediante multiplicaciones con matrices de peso para generar tres conjuntos de señales: queries (Q), keys (K) y values (V). Esta proyección se realiza de forma independiente para cada head, lo que significa que el espacio de características original se divide en subespacios más pequeños.

**2. Scaled Dot-Product Attention:**

Cada head aplica el mecanismo de atención propio a sus respectivos Q, K y V.

$$\text{Attention}(Q, K, V) = \text{softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right) V$$

- Calcular la similitud entre los queries y keys usando el producto punto.
- Escalar el resultado dividiéndolo entre  $\sqrt{d_k}$  para evitar valores excesivamente grandes que podrían saturar el softmax.
- Aplicar softmax para obtener un vector de atención que determina la importancia de cada elemento en la secuencia.
- Multiplicar estos pesos por los values para obtener una representación ponderada que resuma la información relevante.

**3. Concatenación de los heads y Proyección Final:**

Una vez que cada head ha procesado su parte, sus salidas se concatenan. El vector resultante pasa a través de una última proyección lineal para integrar la información combinada de todos los subespacios, permitiendo que la red tenga una representación enriquecida y diversa de las relaciones en la secuencia.

+ 0.5 pts Click here to replace this description.

+ 1 pt Click here to replace this description.

✓ + 1 pt Diseño

+ 0.5 pts Manejar la variabilidad en las pruebas

+ 0.5 pts Soporte a nuevos tipos de exámenes

+ 0.5 pts Recursos limitados

+ 0.5 pts [Click here to replace this description.](#)

+ 0 pts Una solución es emplear una arquitectura modular y flexible que procese cada tipo de dato a través de una rama especializada y que, posteriormente, fusione sus representaciones en un espacio común para la toma de decisiones. La idea central es separar el procesamiento según la modalidad para aprovechar los modelos que se han demostrado eficientes en cada caso, y luego combinar la información de forma que se puedan capturar las interacciones entre ellas, sin asumir que siempre estarán disponibles todas.

Para ello, cada rama (por ejemplo, la encargada de señales EEG, la de imágenes de resonancia magnética, y la de reportes) se construye con un encoder específico que transforma la entrada en un vector de características. En el caso de las imágenes, se podría utilizar una red ligera inspiradas en MobileNet para ajustarse a los recursos limitados. Para las señales EEG, se podría emplear un módulo que combine convoluciones temporales con mecanismos de atención simples (solo una capa con pocos heads). Por último, los reportes textuales pueden procesarse mediante modelos de embeddings o arquitecturas basadas en transformers de baja complejidad (Reformer, por ejemplo), generando representaciones que capturen la información clínica relevante.

El desafío de la variabilidad en los exámenes (los pacientes realizan un sub-conjunto de exámenes) se aborda incorporando en la etapa de fusión un mecanismo que sea robusto a la ausencia de entradas. Esto puede lograrse utilizando máscaras o pesos adaptativos que ignoren las ramas no disponibles para un determinado paciente y normalicen la contribución de las modalidades presentes. De esta forma, el módulo de salida recibe siempre una representación coherente y comparable, sin penalizar la falta de información en una o más ramas.

Además, la escalabilidad a nuevos tipos de exámenes se facilita mediante la naturaleza modular del diseño. Cada nueva modalidad se implementaría como una rama adicional, con su propio encoder entrenado o adaptado, cuyo output se alinea en la misma dimensión del espacio latente común. La etapa de fusión puede adaptarse de manera que, al integrarse nuevos vectores de características, la red aprenda a ponderar y combinar la información de forma óptima, sin tener que rehacer completamente la arquitectura existente.

Finalmente, se pueden aplicar técnicas de transfer learning, utilizar modelos preentrenados y realizar cuantización de los modelos. Esto no solo reduce la carga computacional, sino que también permite que el sistema se entrene con menos datos, aprovechando el conocimiento ya adquirido en tareas relacionadas.

## Question 11

✓ + 4 pts Correcto!

+ 0 pts Empleando Maximum a Posteriori (MAP) buscamos:

$$\hat{w} = \arg \max_w p(w|D) = \arg \max_w p(D|w) p(w)$$

Aplicando logaritmo:

$$\hat{w} = \arg \max_w \log p(D|w) + \log p(w)$$

1. **Primer termino.** Desarrollamos:

$$p(D | w) = \prod_{i=1}^N p((x_i, y_i) | w) = \prod_{i=1}^N p(y_i | x_i, w)$$

Aplicamos logaritmo:

$$\log p(D | w) = \sum_{i=1}^N \log p(y_i | x_i, w) = \sum_{i=1}^N \log \left( \frac{1}{2b} \right) - \frac{1}{b} \sum_{i=1}^N |y_i - x_i^T w|$$

Simplificando:

$$\log p(D | w) = -N \log(2b) - \frac{1}{b} \sum_{i=1}^N |y_i - x_i^T w|$$

Ignoramos el termino constante:

$$\log p(D | w) \propto -\frac{1}{b} \sum_{i=1}^N |y_i - x_i^T w|$$

2. **Segundo termino.** Desarrollamos:

$$p(w) = \prod_{j=1}^d \frac{1}{2b'} \exp \left( -\frac{|w_j|}{b'} \right)$$

Aplicamos logaritmo:

$$\log p(w) = \sum_{j=1}^d \log \left( \frac{1}{2b'} \right) - \frac{1}{b'} \sum_{j=1}^d |w_j| = -d \log(2b') - \frac{1}{b'} \sum_{j=1}^d |w_j|$$

Ignoramos el termino constante:

$$\log p(w) \propto -\frac{1}{b'} \sum_{j=1}^d |w_j|$$

3. **Derivamos el estimador MAP.** Unimos los terminos:

$$\hat{w} \propto \max_w \left\{ -\frac{1}{b} \sum_{i=1}^N |y_i - x_i^T w| - \frac{1}{b'} \sum_{j=1}^d |w_j| \right\}$$

Maximizar una función es equivalente a minimizar la negativa de la misma:

$$\hat{w} = \arg \min_w \left\{ \sum_{i=1}^N |y_i - x_i^T w| + \lambda \sum_{j=1}^d |w_j| \right\}$$

donde  $\lambda = \frac{b}{b'}$ .


+ 2 pts Click here to replace this description.

+ **0.5 pts** Click here to replace this description.

+ **1 pt** Click here to replace this description.

+ **2.5 pts** Click here to replace this description.

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

 cambia ese e y es perfecto

## Question 12



✓ + 1 pt Backbone Correcto!

+ 1 pt Spatial Attention Correcto!

✓ + 1.5 pts Loop LSTM Correcto!

+ 0.5 pts Outlayer Correcto!

+ 0 pts

```
import torch
import torch.nn as nn

def Encoder():
    return nn.Sequential(
        # Conv1 [n_batch,24,80,40]
        nn.Conv2d(3, 24, kernel_size=5, stride=2, padding=2),
        nn.BatchNorm2d(24),
        nn.ReLU(inplace=True),

        # Conv2 [n_batch,36,40,20]
        nn.Conv2d(24, 36, kernel_size=5, stride=2, padding=2),
        nn.BatchNorm2d(36),
        nn.ReLU(inplace=True),

        # Conv3 [n_batch,48,20,10]
        nn.Conv2d(36, 48, kernel_size=3, stride=2, padding=1),
        nn.BatchNorm2d(48),
        nn.ReLU(inplace=True),

        # Conv4 [n_batch,64,20,10]
        nn.Conv2d(48, 64, kernel_size=3, stride=1, padding=1),
        nn.BatchNorm2d(64),
        nn.ReLU(inplace=True),

        # Conv5 [n_batch,64,20,10]
        nn.Conv2d(64, 64, kernel_size=3, stride=1, padding=1),
        nn.BatchNorm2d(64),
        nn.ReLU(inplace=True)
    )

class SpatialAttention(nn.Module):
    def __init__(self, n_hidden, in_chan=64,h=20,w=10, mode='single_channel'):
        super(SpatialAttention, self).__init__()
        self.h = h
        self.w = w

        self.convLayer = nn.Conv2d(in_chan, 1, kernel_size=1, stride=1, padding=0)
        self.fcLayer = nn.Linear(n_hidden, h*w)
        self.Softmax = nn.Softmax(dim=-1) # Se ejecuta en la ultima dimension

    def forward(self, featureMap, hidden):
        # feature_map: [n_batch,64,20,10]
        sptLogits = self.convLayer(featureMap) # [n_batch,1,20,10]
        hddLogits = self.fcLayer(hidden) # [n_batch,h*w]
        hddLogits = hddLogits.view(-1, 1, self.h, self.w) # [n_batch,1,20,10]
```

```
attnLogits = sptLogits + hddLogits
attnMapp = self.Softmax(attnLogits)
```

```
return attnMapp
```

```
class DrivingNet(nn.Module):
```

```
def __init__(self, n_chan=64, n_hidden=128, h=10, w=20, n_layers=1):
```

```
    super(DrivingNet, self).__init__()
```

```
    self.h = h
```

```
    self.w = w
```

```
    self.n_chan = n_chan
```

```
    self.n_hidden = n_hidden
```

```
    self.n_layers = n_layers
```

```
    self.encoder = Encoder()
```

```
    self.spatialAttn = SpatialAttention(n_hidden,in_chan=64)
```

```
    self.LSTM = nn.LSTMCell(input_size =n_chan*h*w, # feature map final es [n_batch,n_chan,h,w]
                            hidden_size=n_hidden)
```

```
    self.fc = nn.Linear(n_hidden+n_chan*h*w, 1)
```

```
def forward(self, x, h0=None, c0=None):
```

```
    # x: [n_batch, n_sequence, n_chan, h, w]
```

```
    x = x.permute(1, 0, 2, 3, 4) # [n_sequence, n_batch, n_chan, h, w]
```

```
    n_sequence, n_batch, _, _ = x.size()
```

```
    # Inicialización
```

```
    if h0 is None: ht = x.new_zeros(n_batch, self.hidden_size)
```

```
    else          : ht = h0
```

```
    if c0 is None: ct = x.new_zeros(n_batch, self.hidden_size)
```

```
    else          : ct = c0
```

```
    u = []
```

```
    for t in range(n_sequence):
```

```
        xt = x[t] # [n_batch,n_chan,h,w]
```

```
        # Extraer features con el encoder
```

```
        ft = self.encoder(xt) # [n_batch,64,20,10]
```

```
        # Aplicar spatial attention
```

```
        at = self.spatialAttn(ft) # [n_batch,1,20,10]
```

```
        ft = ft * at              # [n_batch,64,20,10]
```

```
        yt = ft.view(n_batch, -1) # [n_batch,64*20*10]
```

```
        # La red recibe el estado anterior ht-1 y la nueva informacion
```

```
        # Si ht es input de fc habría redundancia de información
```

```
        zt = torch.cat([yt,ht],dim=1)
```

```
        ut = self.fc(zt)
```

```
        # ht: [n_batch,n_hidden]
```

```
        ht,ct = self.LSTM(yt,(ht,ct))
```

```
        u.append(ut)
```

```
    out = torch.stack(u, dim=1) # [n_batch,n_sequence,1]
```

```
    return out
```

+ **0.5 pts** [Click here to replace this description.](#)

Profesor: Victor Flores Benites

Apellidos: Méndez Lázaro

Nombres: Luis Fernando

Sección: 1 Fecha: 30/01/2025

Nota:

Indicaciones:

La Duración es de **120 minutos**.

La evaluación consta de **12 preguntas**.

Pregunta 1 (1.5 puntos)

Sobre las capas pooling empleadas en Convolutional Neural Networks (CNNs):

- a) Las capas de pooling aumentan las dimensiones espaciales de los feature maps para preservar detalles. ☒
- b) Las capas de pooling ayudan a que las representaciones sean invariantes a pequeñas translaciones en la entrada. ☒
- c) Las capas de pooling son responsables de aprender representaciones jerárquicas. ☒
- d) Las capas de pooling reemplazan las capas de convolución para reducir la complejidad computacional. ☐
- e) Las capas de pooling normalizan los datos de entrada para tener media cero y varianza unitaria. ☒

Pregunta 2 (1.5 puntos)

¿Qué innovación clave introdujo la arquitectura ResNet?

- a) Usar convoluciones de  $1 \times 1$  para reducir el número de parámetros. ☐
- b) Introducir batch normalization después de cada capa de convolución. ☒
- c) Implementar conexiones residuales para afrontar el vanishing gradient. ☒
- d) Utilizar convoluciones dilatadas para aumentar el campo receptivo sin parámetros adicionales. ☐
- e) Aplicar capas de dropout para prevenir el overfitting. ☐

Pregunta 3 (1.5 puntos)

¿Cuál es la función principal de la forget gate en LSTM?

$$C_t = f \cdot C_{t-1} + i \cdot \tilde{C}_t$$

- a) Filtra características irrelevantes de los datos de entrada antes de procesarlos. ☐
- b) Escala la salida del cell state para producir la salida final. ☐
- c) Decide cuánta información nueva agregar al cell state desde la entrada actual. ☐
- d) Determina cuánto del estado previo de la celda se debe conservar. ☒
- e) Olvida información del hidden state a partir de la nueva información introducida. ☐

Pregunta 4 (1.5 puntos)

¿Cuál es la principal ventaja de los Transformers sobre los modelos basados en RNN?

- a) Los Transformers procesan las secuencias de forma secuencial, haciéndolos más eficientes para entradas largas. ☒
- b) Los Transformers utilizan capas convolucionales para capturar eficientemente las dependencias locales. ☒
- c) Los Transformers dependen de mecanismos de atención, lo que permite un entrenamiento paralelo en la secuencia. ☒
- d) Los Transformers usan conexiones recurrentes para mantener información sobre largas secuencias. ☒
- e) Los Transformers están limitados a secuencias de longitud fija debido a su arquitectura. ☐

Pregunta 5 (1.5 puntos)

¿Cuál es mejora de MobileNet en comparación con las arquitecturas clásicas basadas en CNN?

- a) Convoluciones separables en profundidad que factoriza la convolución estándar. ☒
- b) Reduce el número de capas para disminuir la cantidad de parámetros. ☒
- c) Reemplaza capas de convolución con capas completamente conectadas. ☐
- d) Incorpora conexiones residuales para facilitar el entrenamiento de redes más profundas. ☐
- e) Usa kernels de convolución más grandes para aumentar el campo receptivo. ☐

Pregunta 6 (1.5 puntos)

¿Cuál es la función de skip connections en U-net?

- a) Reducir el overfitting regularizando la red. ☒
- b) Recuperar información espacial perdida durante el downsampling. ☒
- c) Ampliar el campo receptivo de las capas convolucionales. ☒
- d) Permitir la fusión de características a múltiples escalas desde diferentes capas. ☒
- e) Facilitar el entrenamiento mitigando el vanishing gradient. ☒

Pregunta 7 (1.5 puntos)

¿Cuál es una diferencia clave entre LSTM y GRU?

- a) LSTM combinan el input y forget gate, mientras que las GRU las separan. ☒
- b) LSTM utilizan funciones de activación lineal, mientras que las GRU utilizan tanh y sigmoid. ☒
- c) GRU son más adecuadas para secuencias largas debido a su estructura más eficiente. ☒
- d) GRU carece de cell state. ☒
- e) LSTM no pueden manejar secuencias con dependencias a largo plazo tan efectivamente como las GRU. ☒

Pregunta 8 (1.5 puntos)

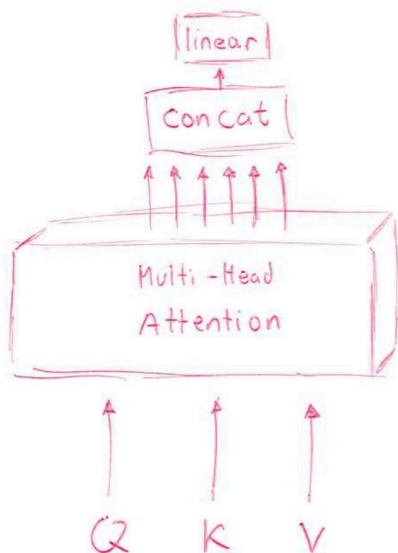
¿Cuál es la función principal de la Region Proposal Network (RPN) en Faster R-CNN?

- a) Generar propuestas de regiones candidatas que probablemente contengan objetos. ☒
- b) Clasificar directamente los objetos sin necesidad de propuestas de región. ☒
- c) Refinar las cajas delimitadoras producidas por el modelo base para mejorar la precisión. ☒
- d) Fusionar información de múltiples escalas para detectar objetos de diferentes tamaños. ☒
- e) Convertir el problema de detección en uno de segmentación semántica. ☒

Pregunta 9 (2.5 puntos)

Explique Multi-head Attention. Su función y los bloques que los componen a detalle.

- Computa la matriz de self-attention a partir de las matrices  $Q, K, V$
- De forma paralela, se computan las matrices  $head_i$  para luego concatenarse.



$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$\text{Multi-Head}(Q, K, V) =$$

$$\text{Concat}(\text{head}_1, \text{head}_2, \dots, \text{head}_n)$$

$$\text{head}_i = \text{Attention}(Q^i, K^i, V^i)$$

- Cada head se especializa en una tarea concreta.



Pregunta 10 (2.5 puntos)

En un hospital especializado, varios pacientes con desórdenes neurológicos generan datos simultáneamente de: señales cerebrales (monitoreo EEG), estudios de imagen (por ejemplo, resonancias magnéticas) y reportes textuales (observaciones clínicas, notas de enfermería). Con recursos limitados y la necesidad de diagnósticos rápidos, el equipo médico requiere un sistema unificado que integre toda esta información heterogénea y proporcione alertas tempranas de anomalías.

Proponga una arquitectura escalable a nuevos tipos de exámenes (cada paciente tiene un conjunto distinto de exámenes) y capaz de aprender la interacción entre diferentes modalidades de datos.

Sea detallado en su respuesta.

*Propongo una arquitectura multimodal con los siguientes módulos:*

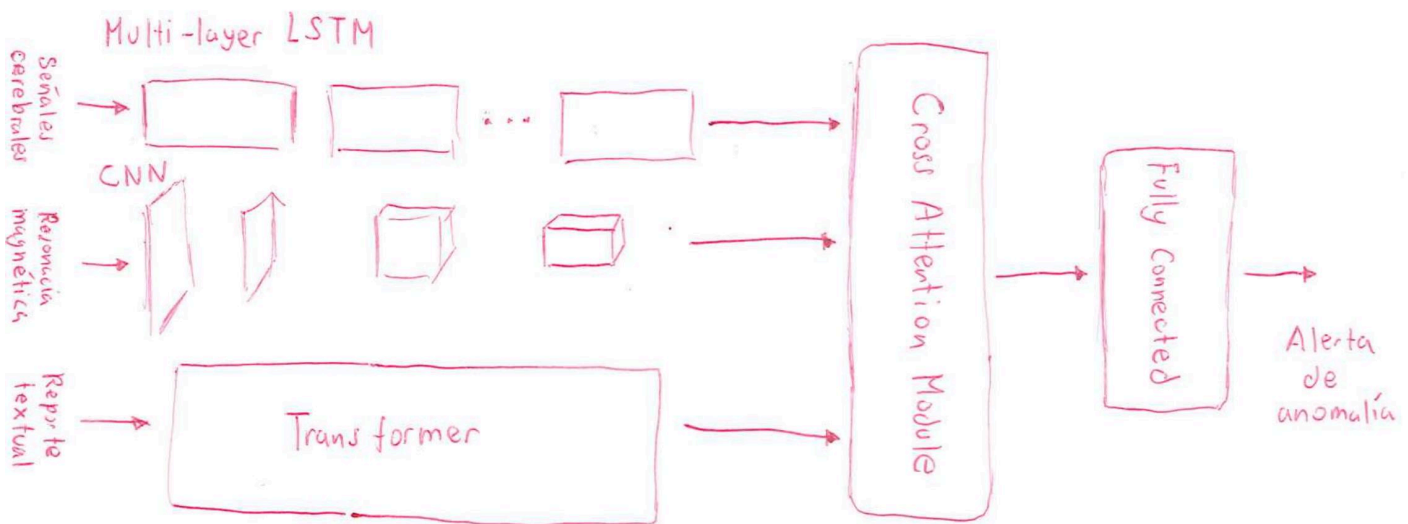
*1.- 3 ramas de preprocesamiento (1 para cada modalidad de dato)*

- Un LSTM multi-layer para señales cerebrales.*
- Un CNN para resonancias magnéticas (imágenes).*
- Un transformer para reportes textuales.*

*2.- Módulo de Cross Attention (Adaptado de Cross ViT)*

- Para agregar contexto global e enriquecer los feature maps.*

*3.- Fully connected para clasificación (alertas)*



Pregunta 11 (4 puntos)

Se tiene un modelo de regresión lineal definido como:  $y = Xw + \epsilon$ , donde:  $X \in \mathbb{R}^{n \times d}$  son los datos,  $w \in \mathbb{R}^d$  es el vector de parámetros,  $y \in \mathbb{R}^n$  es el vector de respuestas. Se asume que el ruido sigue una distribución Laplaciana  $\epsilon \sim \text{Laplace}(0, b)$ . Usando el enfoque de Máxima a Posteriori (MAP), deriva la función de costo resultante considerando un prior Laplaciano sobre los parámetros del modelo.

La función de densidad de probabilidad (PDF) de la variable aleatoria  $\epsilon$  con distribución Laplaciana es:

$$p(\epsilon) = \frac{1}{2b} \exp\left(-\frac{|\epsilon|}{b}\right)$$

La distribución del prior también sigue una distribución Laplaciana:

$$p(y|X, w) = \prod_{i=1}^n \frac{1}{2b} \exp\left(-\frac{|y_i - X_i w|}{b}\right)$$

Por comodidad  $p(y_i | x_i, w) = \frac{1}{2b} e^{-\frac{|y_i - x_i w|}{b}}$

Luego:  $\hat{w} = \arg \max_w p(w|D) = \arg \max_w p(D|w) p(w)$

$$= \frac{1}{\prod_{i=1}^n p(x_i)} p(y; | x; , w) = \frac{1}{\prod_{i=1}^n p(x_i)} \prod_{i=1}^n p(y_i | x_i, w)$$

$$= \sum_{i=1}^n \log p(x_i) + \sum_{i=1}^n \log p(y_i | x_i, w)$$

(\*)

(\*)  $\sum_{i=1}^n \log p(y_i | x_i, w) = \sum_{i=1}^n \log \left( \frac{1}{2b} e^{-\frac{|y_i - x_i w|}{b}} \right) = \sum_{i=1}^n \log \left( \frac{1}{2b} \right) + \sum_{i=1}^n \left( -\frac{|y_i - x_i w|}{b} \right)$

$$= -n \log(2b) - \frac{1}{b} \sum_{i=1}^n |y_i - x_i w|$$

Además:

$$\log p(\epsilon) = \log \left( \frac{1}{2b} e^{-\frac{|\epsilon|}{b}} \right) = -\log(2b) - \frac{|\epsilon|}{b}$$

Por lo tanto:

$$\hat{w} = \arg \max_w \left[ -\frac{1}{b} \sum_{i=1}^n |y_i - x_i w| - \frac{|\epsilon|}{b} \right]$$

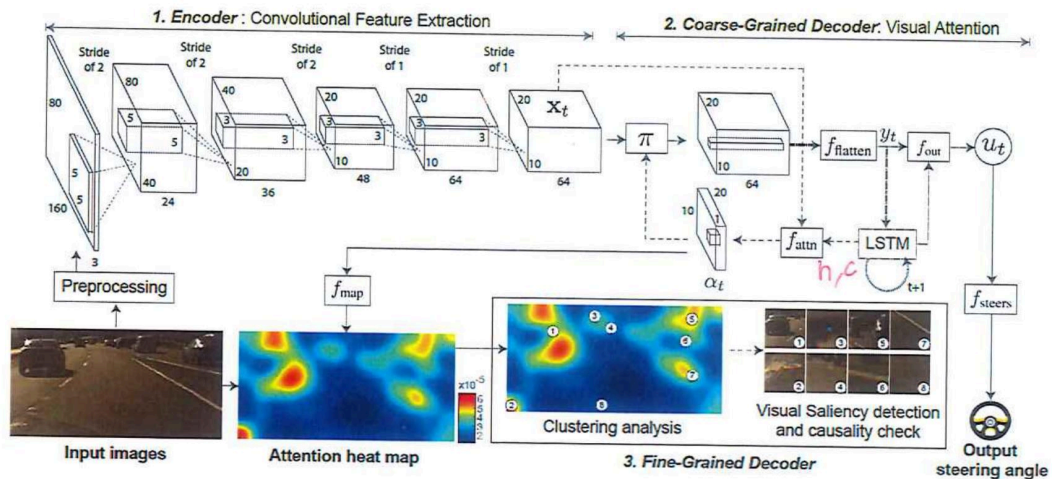
$$\hat{w} = \arg \min_w \frac{1}{b} \left[ \underbrace{\sum_{i=1}^n |y_i - x_i w|}_{\text{Loss}} + \underbrace{|\epsilon|}_{\text{Regularization}} \right]$$

Loss

Regularization

Pregunta 12 (4 puntos)

Para un proyecto de conducción autónoma, se tiene como tarea implementar la arquitectura mostrada en la figura, la cual emplea un mecanismo de spatial attention. Proponga un pseudocódigo, asegurando que las dimensiones los tensores internos sean las adecuadas.



En la figura, el símbolo  $\pi$  representa una multiplicación punto a punto en la dimensión espacial. Los bloques  $f_{attn}$  y  $f_{out}$  son redes que calculan los mapas de atención y calculan la señal  $u_t$ , respectivamente. Asuma que la salida de la red es la señal de control  $u_t$ . Además, omita el bloque  $f_{map}$ .

Simple Module :

init ()

**cnn** = Sequential (

Conv2d(3, 24, Kernel-size=5, padding=2, stride=2),

Conv2d(24, 36, Kernel-size=5, padding=2, stride=2),

Conv2d(36, 48, Kernel-size=3, padding=1, stride=2),

Conv2d(48, 64, Kernel-size=3, padding=1, stride=4),

Conv2d(64, 64, Kernel-size=3, padding=1, stride=1)

)

lstm = LSTM (input-size=64, hidden-size=20, num.layers=1)

fc\_attention = Linear (64\*10\*20, 10\*20)

flatten = flatten()

fc\_out = Linear (10\*20, 1)

forward(x) atrás de la  
página →