Tarea 2

Arquitecturas modernas de Redes Neuronales profundas

Profesor: Fernando Crema Garcia
Fecha: 18 de Septiembre de 2026

Evaluación. La nota final se basa en el promedio ponderado de las tareas (70 %) y el proyecto final (30 %). La nota de cada informe se multiplica por un factor de calidad $\alpha \in [0{,}8,\ 1{,}2]$ según la calidad del reporte asociado.

Tarea	Tema	Peso
Tarea 1	Fundamentos	30 %
Tarea 2	Componentes: CNN y U-NET	30 %
Tarea 3	Arquitecturas	30 %

Entrega soft	30/09/2026, 23:59 (hora Caracas) - Máximo 20 ptos
Entrega final	18/10/2026, 23:59 (hora Caracas) - Máximo 20 ptos
Penalización	2 ptos/día de retraso hasta un máximo de 5 días. **(no aplica)** Después del 18/10/2026 la nota es 0.

Nota final:

$$\text{mín}\Big\{\alpha\left(0{,}3\,T_1 + 0{,}3\,T_2 + 0{,}3\,T_3 + 0{,}1\,T_4\right),\ 20\Big\}$$

I. Introducción

I-A. Objetivos

El objetivo de esta actividad es consolidar los conocimientos sobre redes neuronales convolucionales y arquitecturas encoder-decoder en dos frentes: la resolución de una serie de notebooks del libro *Understanding Deep Learning* (UDL) de Simon Prince, y un caso práctico de segmentación de imágenes planteado como una pasantía corta en una empresa de AI.

Objetivos específicos

1. Resolver los siguientes notebooks del libro UDL, utilizando Jupyter/Colab:
   - **Notebook 10.1** – 1D convolution
   - **Notebook 10.2** – Convolution for MNIST-1D
   - **Notebook 10.3** – 2D convolution
   - **Notebook 10.4** – Downsampling & upsampling
   - **Notebook 10.5** – Convolution for MNIST
2. Responder las preguntas teóricas sobre CNN de la Sección III-A.
3. Mejorar el rendimiento de una red de segmentación siguiendo el protocolo de la Sección III-B, y documentarlo en un informe de no más de 4 páginas.

email: fernando.cremagarcia@kuleuven.be

I-B. Entregables

1. Los notebooks del UDL resueltos, con comentarios propios sobre los conceptos cubiertos y cómo se relacionan entre sí a lo largo del flujo de aprendizaje profundo.
2. El notebook `04_CNN_Segmentation_v3.ipynb` con todos sus experimentos, **ejecutado y con las salidas visibles** (tabla de resultados, gráficos, ejemplos). Un notebook sin salidas no se corrige.
3. El informe en PDF, de máximo 4 páginas sin contar anexos, con la estructura de la Sección III-C.

II. Arquitecturas Encoder-Decoder (Teoría y Fórmulas)

Un modelo **encoder-decoder** transforma una entrada $\mathbf{x} \in \mathbb{R}^n$ en una salida $\mathbf{y} \in \mathbb{R}^m$ mediante una representación latente $\mathbf{z} \in \mathbb{R}^d$, donde normalmente $d < n$.

II-A. Encoder

El **encoder** convierte la entrada $\mathbf{x}$ en una representación latente $\mathbf{z}$:

$$\mathbf{z} = f_{\text{enc}}(\mathbf{x}; \theta_e) \tag{1}$$

- f_{enc}: red neuronal (CNN, RNN, MLP, etc.)
- θ_e: parámetros del encoder
- $\mathbf{z}$: vector latente comprimido

Si el encoder tiene múltiples capas:

$$\mathbf{z} = f_L \circ f_{L-1} \circ \cdots \circ f_1(\mathbf{x}) \tag{2}$$

En términos de dominio y codominio:

$$f_{\text{enc}} : \mathbb{R}^n \to \mathbb{R}^d \tag{3}$$

- $\mathbf{x} \in \mathbb{R}^n$: entrada original
- $\mathbf{z} = f_{\text{enc}}(\mathbf{x}) \in \mathbb{R}^d$: representación latente
- Típicamente $d < n$ para compresión

II-B. Decoder

El **decoder** reconstruye o genera la salida a partir de $\mathbf{z}$:

$$\hat{\mathbf{y}} = f_{\text{dec}}(\mathbf{z}; \theta_d) \tag{4}$$

- f_{dec}: red decodificadora
- θ_d: parámetros del decoder
- $\hat{\mathbf{y}}$: salida generada o reconstruida

Reconstruye o genera la salida desde el espacio latente:

$$f_{\text{dec}} : \mathbb{R}^d \to \mathbb{R}^m \tag{5}$$

- $\mathbf{z} \in \mathbb{R}^d$: vector latente
- $\hat{\mathbf{y}} = f_{\text{dec}}(\mathbf{z}) \in \mathbb{R}^m$: salida reconstruida o generada

II-C. El modelo

El modelo completo es la composición de encoder y decoder:

$$f_{\text{dec}} \circ f_{\text{enc}} : \mathbb{R}^n \to \mathbb{R}^m \tag{6}$$

Es decir:

$$\hat{\mathbf{y}} = f_{\text{dec}}(f_{\text{enc}}(\mathbf{x})) \tag{7}$$

II-D. Función de Pérdida

El modelo se entrena minimizando la pérdida entre la salida generada $\hat{\mathbf{y}}$ y la salida real $\mathbf{y}$:

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = \mathcal{L}\left(\mathbf{y}, f_{\text{dec}}(f_{\text{enc}}(\mathbf{x}))\right) \tag{8}$$

II-D1. Ejemplos comunes

- **Reconstrucción (autoencoder):**

$$\mathcal{L} = \|\mathbf{x} - f_{\text{dec}}(f_{\text{enc}}(\mathbf{x}))\|^2 \tag{9}$$

- **Segmentación (por píxel):**

$$\mathcal{L} = \text{CrossEntropy}(\mathbf{y}, \hat{\mathbf{y}}) \tag{10}$$

- **Secuencia a secuencia (traducción):**

$$\mathcal{L} = -\sum_{t=1}^{T} \log P(y_t \mid \mathbf{z}, y_{<t}) \tag{11}$$

II-E. Segmentación como clasificación por píxel

En segmentación semántica la salida no es un vector sino un mapa $\hat{\mathbf{Y}} \in \mathbb{R}^{H \times W \times C}$: en cada píxel (i, j) hay una distribución softmax sobre las C clases. La pérdida es la entropía cruzada promediada sobre todos los píxeles:

$$\mathcal{L} = -\frac{1}{HW} \sum_{i,j} \log \hat{y}_{ij,\, y_{ij}} \tag{12}$$

Para que $\hat{\mathbf{Y}}$ tenga las mismas dimensiones espaciales que la entrada, el decoder debe deshacer *exactamente* las reducciones de resolución del encoder.

II-F. Dimensiones de salida

Para una convolución con kernel k, stride s y padding p por lado:

$$H_{\text{out}} = \left\lfloor \frac{H_{\text{in}} + 2p - k}{s} \right\rfloor + 1 \tag{13}$$

Para una convolución transpuesta:

$$H_{\text{out}} = (H_{\text{in}} - 1)\, s - 2p + k \tag{14}$$

En Keras, `padding="same"` equivale a $H_{\text{out}} = \lceil H_{\text{in}}/s \rceil$ para la convolución y a $H_{\text{out}} = s \cdot H_{\text{in}}$ para la transpuesta. Nótese que con $s = 1$ ninguna de las dos cambia el tamaño.

II-G. U-Net: skip connections

La U-Net (Ronneberger et al., 2015) es un encoder-decoder simétrico en el que el mapa de características del encoder en el nivel ℓ se *concatena* con el mapa del decoder cuando este vuelve a la misma resolución:

$$\mathbf{d}_\ell = g_\ell\big(\,[\,\text{Up}(\mathbf{d}_{\ell+1}) \,\|\, \mathbf{e}_\ell\,]\,\big) \tag{15}$$

donde $\mathbf{e}_\ell$ es la salida del encoder en el nivel ℓ, $\|$ denota concatenación por canales y g_ℓ son las convoluciones del bloque. Las skips dan al decoder acceso directo a la información espacial fina que se pierde en el cuello de botella. No confundir con una conexión *residual*, que suma $(\mathbf{x} + F(\mathbf{x}))$ la salida de un bloque con la del bloque inmediatamente anterior y exige que ambos tengan la misma forma.

II-H. Métrica: Intersection over Union

Para la clase c, con P_c los píxeles predichos como c y T_c los que realmente son c:

$$\text{IoU}_c = \frac{|P_c \cap T_c|}{|P_c \cup T_c|} = \frac{TP_c}{TP_c + FP_c + FN_c}, \qquad \text{mIoU} = \frac{1}{C} \sum_{c=1}^{C} \text{IoU}_c \tag{16}$$

A diferencia de la pixel accuracy, el mIoU da el mismo peso a cada clase sin importar cuántos píxeles ocupe.

III. Evaluación

III-A. CNN (teoría)

1. Considere la siguiente matriz

$$\mathbf{X} = \begin{bmatrix} 2 & 5 & 4 & 1 \\ 3 & 1 & 2 & 0 \\ 4 & 5 & 7 & 1 \\ 1 & 2 & 3 & 4 \end{bmatrix}$$

   a) Calcule la salida de una convolución con kernel de dimensión 2×2, sin padding y stride de 2.

$$\mathbf{K} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

   b) Calcule la salida de una convolución con kernel de dimensión 2×2, padding 2 y stride de 2.

$$\mathbf{K} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

2. ¿Cómo podemos determinar la dimensión de salida luego de aplicar la función de convolución?
3. ¿Qué beneficios tiene el uso de CNNs sobre las redes completamente conectadas?

III-B. CNN Segmentation: la pasantía

Imaginen que son contratados para una pasantía corta en una empresa de AI. Normalmente, estas pasantías involucran reutilizar código de un grupo e intentar mejorar el rendimiento actual. Este rendimiento, dependiendo claramente de cada empresa, se considera el *baseline*.

Intentemos emular una pasantía corta reutilizando el notebook 04_CNN_Segmentation_v3.ipynb. El objetivo es simple



Figura 1. Muestra de 21 imágenes del dataset Oxford-IIIT Pet redimensionadas a 160×160. Arriba: las imágenes de entrada. Abajo: sus trimaps, con el animal en negro, el fondo en gris y el borde en blanco. Nótese lo delgado del borde frente a las otras dos clases, y la variedad de fondos, poses y escalas.

de enunciar: mejorar el rendimiento de la red que reciben. Como en una pasantía real, lo primero es entender el código heredado; la Sección 2 del notebook explica cada componente y las preguntas de comprensión de más abajo verifican que lo hicieron.

III-B1. Los datos

El notebook usa el dataset *Oxford-IIIT Pet*: 7 390 fotografías de perros y gatos de 37 razas, tomadas en condiciones muy variadas de iluminación, fondo, pose y tamaño del animal dentro del cuadro, como muestra la Figura 1 (arriba). Cada fotografía viene acompañada de un *trimap*: una imagen del mismo tamaño con un entero por píxel que indica su clase, 1 = animal, 2 = fondo, 3 = borde (la franja ambigua en la frontera del animal). El notebook resta 1 para trabajar con las clases $0, 1, 2$. En la Figura 1 (abajo) el animal aparece en negro, el fondo en gris y el borde en blanco.

III-B2. Protocolo

Para que los resultados de distintas arquitecturas, y de distintos estudiantes, sean comparables, todo lo que **no** es la arquitectura está fijo en la celda de configuración del notebook y **no debe modificarse**:

- semilla, split de entrenamiento/validación (1000 / 1000 imágenes) y tamaño de imagen (160×160);
- batch, número de épocas, optimizador y tasa de aprendizaje;
- métricas: **mIoU** sobre validación como métrica principal y pixel accuracy como secundaria; el número que se reporta es el mejor val mIoU y la época en que ocurrió (lo escoge el `ModelCheckpoint`).

Paso 0: reproducir el baseline. Antes de cambiar nada, ejecuten el notebook tal cual y anoten el val mIoU que obtienen. Ese es el número a superar y debe aparecer al inicio del informe junto al valor de referencia dado en clase.

III-B3. Preguntas de comprensión

Deben responderse en el informe **antes** de presentar los experimentos. Todas se contestan con las fórmulas de la Sección II y las celdas de la Sección 2 del notebook.

1. **Datos I.** ¿Por qué las máscaras se redimensionan con vecino más cercano y las imágenes con interpolación bilineal? ¿Qué pasaría si se usara bilineal en las máscaras?
2. **Datos II.** ¿Qué puedes decir sobre la distribución de las clases?
3. **Formas.** Construya a mano la tabla de dimensiones (H, W, C) de cada etapa del baseline (bloque de entrada, tres bloques del encoder, cuatro bloques del decoder y capa de salida) para una entrada de $160 \times 160 \times 3$. ¿Qué tamaño tiene el cuello de botella y a cuántos píxeles de la imagen original corresponde cada posición de él? ¿Qué ocurriría con una entrada de 150×150? Justifique con las fórmulas.
4. **Upsampling.** El decoder del baseline usa `Conv2DTranspose(f, 3, padding="same")` sin especificar `strides`. ¿Cambia esa capa las dimensiones espaciales? ¿Cuál es entonces la capa que duplica la resolución? ¿Cuántos parámetros tiene esa `Conv2DTranspose` comparada con una `Conv2D` con los mismos argumentos? ¿Qué se gana y qué se arriesga al reemplazarla por `Conv2DTranspose(f, 3, strides=2, padding="same")`?
5. **Parámetros.** Calcule a mano los parámetros de una `Conv2D(128, 3)` y de una `SeparableConv2D(128, 3)` cuando la entrada tiene 128 canales (incluya los sesgos) y verifique con el notebook. ¿Cuántos parámetros tiene un bloque completo del encoder de 128 filtros, incluida la proyección residual?
6. **Residuales vs. skips.** Explique por qué el baseline *no* es una U-Net. ¿Qué conecta una conexión residual (`add`) y qué conecta una skip de U-Net (`Concatenate`)? ¿Qué condición de forma exige cada una y para qué sirve la proyección 1×1?
7. **Métrica.** Con las fracciones de píxeles por clase que imprime el notebook, estime qué pixel accuracy obtendría un modelo que acierta siempre el animal y el fondo pero nunca predice la clase *borde*. ¿Qué mIoU obtendría? Concluya por qué la métrica principal es el mIoU.

III-B4. Experimentos: requerimientos mínimos

1. Entrene y reporte **al menos cuatro arquitecturas** además del baseline. Al menos tres deben salir de la siguiente lista; la cuarta es libre:
   a) skip connections de concatenación encoder $\rightarrow$ decoder (convertir el baseline en una U-Net);

b) estrategia de upsampling: `UpSampling2D` + `Conv2D` frente a `Conv2DTranspose` con `strides=2`;
c) `SeparableConv2D` frente a `Conv2D` con el mismo número de filtros;
d) profundidad: `filters=(64,128)`, `(64,128,256)` y `(64,128,256,512)`;
e) tamaño de kernel: 3 frente a 5;
f) eliminar las conexiones residuales, o eliminar `BatchNormalization`.

2. Antes de entrenar cada variante, escriba en una línea qué espera que ocurra y por qué. Esa hipótesis va en el informe junto al resultado, se cumpla o no.
3. El foco está en los parámetros de la CNN. **No modifique** el batch, las épocas, el optimizador, la tasa de aprendizaje, el tamaño de imagen ni el split.
4. Preste mucha atención a las dimensiones: las de salida de una capa deben coincidir con las de entrada de la siguiente, y las conexiones residuales o skips exigen formas compatibles. Recuerde que algunas arquitecturas tardan mucho en entrenarse; planifique el uso de la GPU.
5. Toda arquitectura que se entrene se reporta, incluidas las que empeoran el baseline.

III-B5. Análisis de resultados

1. La tabla de resultados con **todas** las arquitecturas: nombre, número de parámetros, segundos por época, mejor época, val mIoU y val pixel accuracy (`results_table` en el notebook).
2. Dos gráficos como mínimo: val mIoU por época de todas las arquitecturas, y mejor val mIoU frente a número de parámetros (`compare_experiments`). Una arquitectura con la mitad de los parámetros y el mismo mIoU es una mejora.
3. IoU por clase y matriz de confusión del mejor modelo, comparadas con las del baseline. ¿Qué clase mejoró más? ¿Es consistente con la hipótesis?
4. Tres aciertos y tres fallos del mejor modelo sobre validación, con un comentario sobre qué tienen en común los fallos.
5. Tenga en cuenta la variabilidad: diferencias de $\pm 0{,}01$ en mIoU entre corridas idénticas son normales en GPU. Una mejora de ese orden no es una mejora. Si el tiempo lo permite, entrene la mejor arquitectura con 2–3 semillas y reporte media $\pm$ desviación.

III-C. Estructura del informe

Máximo 4 páginas; los notebooks y cualquier anexo van aparte.

1. **Baseline reproducido** ($\frac{1}{4}$ página): val mIoU y época obtenidos, y el valor de referencia.
2. **Teoría** (1 página): respuestas a la Sección III-A y a las preguntas de comprensión.
3. **Experimentos** (1 página): por cada arquitectura, qué se cambió, qué se esperaba y qué ocurrió.
4. **Análisis** (1 página): tabla, gráficos, IoU por clase y ejemplos.
5. **Handover** ($\frac{1}{2}$ página): como si dejaran el proyecto al equipo: qué funcionó, qué no, y qué probarían con diez veces más cómputo.

III-D. Distribución de la nota de la Tarea 2

Componente	**Puntos**
Notebooks UDL 10.1–10.5 resueltos y comentados	5
Preguntas teóricas de CNN (Sección III-A)	3
Preguntas de comprensión del baseline	3
Experimentos (protocolo respetado, hipótesis, ≥ 4 arquitecturas)	5
Análisis y handover	4
Total	**20**

El factor α de calidad del informe se aplica sobre este total.

IV. Uso de Inteligencia Artificial

El uso de herramientas de inteligencia artificial generativa (ChatGPT, Claude, Copilot, Gemini, etc.) está permitido **únicamente** en los componentes indicados a continuación. En el resto de la tarea, el trabajo debe ser realizado en su totalidad por el estudiante.

Componente	**Uso de AI**
Notebooks del libro UDL (10.1–10.5)	No permitido[1]
III-A. CNN (teoría)	No permitido
III-B3. Preguntas de comprensión	No permitido
III-B4. Experimentos: código de las arquitecturas	Permitido sólo en código
III-B4. Experimentos: hipótesis previas	No permitido
III-B5. Análisis de resultados	No permitido
III-C. Redacción del informe y handover	No permitido

Condiciones de uso. En los componentes donde el uso de AI está permitido, el estudiante debe declarar explícitamente en el informe qué herramienta utilizó y con qué propósito (por ejemplo: generar el esqueleto de una arquitectura, depurar un error de dimensiones, escribir una función de graficado). Toda celda del notebook con código generado o modificado con AI debe comenzar con el comentario `# generado con AI`. La declaración va al final del informe y no cuenta para el límite de páginas. El estudiante es responsable de la veracidad de todo el contenido entregado: los errores producidos por una AI se penalizan igual que los errores propios.

Incumplimiento. El uso de AI en componentes no autorizados se considera una falta de integridad académica y resulta en la anulación del componente correspondiente. El profesor puede solicitar una defensa oral del informe a cualquier estudiante.

[1] Esta es la sección más importante de la tarea. Si logran hacer esto solos, el resto es sencillo.