# Tarea 2

## Arquitecturas modernas de Redes Neuronales profundas

## CNN & UNETs

Profesor: Fernando Crema Garcia
Fecha: 16 de Julio de 2025

La nota final será basada el promedio ponderado de cada tarea (70 %) y 30 % del proyecto final. La nota de los informes será multiplicada por un **extra** ($\alpha \in [0{,}8, 1{,}2]$) dependiendo de la calidad del reporte asociado.

| Tarea 1 | Tarea 2 | Tarea 3 | Tarea 4 |
|---|---|---|---|
| Fundamentos | CNN & UNETs | Transformer | Optimizando |
| 30 % | 30 % | 30 % | 10 % |

**Fecha de entrega hard:**
23/07/2025 23:59 pm Caracas.
**Penalizaciones:**
2 ptos / día. Máximo 10 ptos.
Luego del 05/08/2025 tienen 0.
**Nota final:**

$$\text{mín}\left\{\alpha\left(0{,}3T1 + 0{,}3T2 + 0{,}3T3 + 0{,}1T4\right), 20\right\}$$

## I. Introducción

### I-A. Objetivos

El objetivo de esta actividad es consolidar los conocimientos fundamentales sobre redes neuronales profundas a través de la resolución de una serie de notebooks del libro Understanding Deep Learning de Simon Prince. Los notebooks seleccionados cubren aspectos clave de Redes Neuronales Convolucionales y arquitecturas Encoder-Decoder.

Objetivos específicos

1. Resolver los siguientes notebooks del libro UDL de Simon Prince, 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

Objetivos específicos

Aunado a la resolución de los notebooks[1] se desea la elaboración de un informe de no más de 4 páginas donde resuelvan dos preguntas prácticas usando solamente el código generado dentro de sus notebooks.

1. Documentar el proceso de resolución de los notebooks en un Jupyter Notebook explicativo, incluyendo reflexiones personales sobre los conceptos fundamentales cubiertos y cómo se relacionan entre sí a lo largo del flujo de aprendizaje profundo.
2. De ser posible, diseñar un experimento final donde se integre lo aprendido: entrenar un modelo simple utilizando los principios vistos (activaciones, arquitectura, función de pérdida, optimización, etc.) y almacenarlo como modelo óptimo.

email: fernando.cremagarcia@kuleuven.be
[1]Que serán agregados en el entregable

## 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 $\mathbf{z}$:

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

- $f_{\text{enc}}$: red neuronal (CNN, RNN, MLP, etc.)
- $\theta_e$: 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 $\mathbf{z}$:

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

- $f_{\text{dec}}$: red decodificadora
- $\theta_d$: 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):**

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

## III. Evaluación

### III-A. CNN

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 de dimensión del kernek $2 \times 2$ sin padding y stride de 2.

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

   *b*) Calcule la salida de una convolución de dimensión del kernek $2 \times 2$ 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

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.ipynb. El objetivo es simple: mejorar el rendimiento de la red que reciben.

### III-C. Requerimientos mínimos

1. Pruebe diferentes cantidades de capas, combinaciones de diferentes tipos de capas, número de filtros y tamaños de kernel.
2. Tenga en cuenta que el enfoque no está en experimentar con el tamaño del batch ni con las épocas, sino en los parámetros específicos de las CNN.
3. Preste mucha atención al ajustar los parámetros de una capa convolucional, ya que las dimensiones de entrada y salida entre capas deben coincidir. Analice sus resultados. Recuerde que algunas arquitecturas tardan mucho en entrenarse
4. Deben incluir como mínimo un gráfico con el rendimiento de todas sus arquitecturas. La o las métricas quedan a juicio de ustedes.